## C55x v3.x CPU Algebraic Instruction Set Reference Guide

Literature Number: SWPU068E June 2009



## **Preface**

## **Read This First**

#### **About This Manual**

The C55x<sup>TM</sup> is a fixed-point digital signal processor (DSP) in the TMS320<sup>TM</sup> DSP family, and it can use either of two forms of the instruction set: a mnemonic form or an algebraic form. This book is a reference for the algebraic form of the instruction set. It contains information about the instructions used for all types of operations. For information on the mnemonic instruction set, see C55x v3.x CPU Mnemonic Instruction Set Reference Guide, SWPU067.

This release is updated with the 3.0 Revision of the TMS320C55x DSP. Information not affected by the revision remains identical to the previous manual. The main new features of this revision are:

Relaxed parallelism restrictions:
Total size of both instructions may be up to 8 bytes.
Constant buses (KAB and KDB) are no longer a source of conflict.
New instructions:
Ismf
36 dual mac instructions with double coefficient features
MPY, MAC, and MAS instructions with unsigned coefficients
lock

## **Notational Conventions**

Thi	s book uses the following conventions.
	In syntax descriptions, the instruction is in a <b>bold typeface</b> . Portions of a syntax in <b>bold</b> must be entered as shown. Here is an example of an instruction syntax:
	Ims(Xmem, Ymem, ACx, ACy)
	<b>Ims</b> is the instruction, and it has four operands: <i>Xmem</i> , <i>Ymem</i> , <i>ACx</i> , and <i>ACy</i> . When you use <b>Ims</b> , the operands should be actual dual datamemory operand values and accumulator values. A comma and a space (optional) must separate the four values.
	Square brackets, [ and ], identify an optional parameter. If you use an optional parameter, specify the information within the brackets; do not type the brackets themselves.

#### Related Documentation From Texas Instruments

The following books describe the C55x<sup>™</sup> devices and related support tools. To obtain a copy of any of these TI documents, call the Texas Instruments Literature Response Center at (800) 477-8924. When ordering, please identify the book by its title and literature number.

- **TMS320C55x Technical Overview** (SPRU393). This overview is an introduction to the TMS320C55x<sup>™</sup> digital signal processor (DSP). The TMS320C55x is the latest generation of fixed-point DSPs in the TMS320C5000<sup>™</sup> DSP platform. Like the previous generations, this processor is optimized for high performance and low-power operation. This book describes the CPU architecture, low-power enhancements, and embedded emulation features of the TMS320C55x.
- **C55x CPU Reference Guide** (literature number SWPU073) describes the architecture, registers, and operation of the CPU for the TMS320C55x<sup>™</sup> digital signal processors (DSPs).
- **C55x CPU Mnemonic Instruction Set Reference Guide** (literature number SWPU067) describes the mnemonic instructions individually. It also includes a summary of the instruction set, a list of the instruction opcodes, and a cross-reference to the algebraic instruction set.
- **TMS320C55x Programmer's Guide** (literature number SPRU376) describes ways to optimize C and assembly code for the TMS320C55x<sup>™</sup> DSPs and explains how to write code that uses special features and instructions of the DSP.
- TMS320C55x Optimizing C Compiler User's Guide (literature number SPRU281) describes the TMS320C55x™ C Compiler. This C compiler accepts ANSI standard C source code and produces assembly language source code for TMS320C55x devices.
- TMS320C55x Assembly Language Tools User's Guide (literature number SPRU280) describes the assembly language tools (assembler, linker, and other tools used to develop assembly language code), assembler directives, macros, common object file format, and symbolic debugging directives for TMS320C55x™ devices.

#### **Trademarks**

TMS320, TMS320C54x, TMS320C55x, C54x, and C55x are trademarks of Texas Instruments.

# **Contents**

ı	Term	s, Symbols, and Abbreviations 1	-1
		and defines the terms, symbols, and abbreviations used in the TMS320C55x DSP	
		raic instruction set summary and in the individual instruction descriptions.	
	1.1	Instruction Set Terms, Symbols, and Abbreviations	-2
	1.2	Instruction Set Conditional (cond) Fields	
	1.3	Affect of Status Bits	
		1.3.1 Accumulator Overflow Status Bit (ACOVx)	
		1.3.2 C54CM Status Bit	
		1.3.3 CARRY Status Bit	
		1.3.4 FRCT Status Bit	
		1.3.5 INTM Status Bit	
		1.3.6 M40 Status Bit	
		1.3.7 RDM Status Bit	
		1.3.8 SATA Status Bit	
		1.3.9 SATD Status Bit	
		1.3.10 SMUL Status Bit	13
		1.3.11 SXMD Status Bit	13
		1.3.12 Test Control Status Bit (TCx)	13
	1.4	Instruction Set Notes and Rules	14
		1.4.1 Notes	14
		1.4.2 Rules	14
	1.5	Nonrepeatable Instructions 1-2	20
2		elism Features and Rules	-1
	Desc	ibes the parallelism features and rules of the TMS320C55x DSP algebraic instruction set.	
	2.1	Parallelism Features 2	-2
	2.2	Parallelism Basics	-3
	2.3	Resource Conflicts	
		2.3.1 Operators	-4
		2.3.2 Address Generation Units	-4
		2.3.3 Buses	-5
	2.4	Soft-Dual Parallelism	-5
		2.4.1 Soft-Dual Parallelism of MAR Instructions	-6
	2.5	Execute Conditionally Instructions	-6
	2.6	Other Exceptions 2	-7

3	Intro	duction	to Addressing Modes	3-1
	Prov	rides an i	introduction to the addressing modes of the TMS320C55x DSP.	
	3.1	Introdu	uction to the Addressing Modes	3-2
	3.2	Absolu	ute Addressing Modes	3-3
		3.2.1	k16 Absolute Addressing Mode	3-3
		3.2.2	k23 Absolute Addressing Mode	3-3
		3.2.3	I/O Absolute Addressing Mode	3-3
	3.3	Direct	Addressing Modes	3-4
		3.3.1	DP Direct Addressing Mode	3-4
		3.3.2	SP Direct Addressing Mode	3-5
		3.3.3	Register-Bit Direct Addressing Mode	3-5
		3.3.4	PDP Direct Addressing Mode	3-5
	3.4	Indired	et Addressing Modes	3-6
		3.4.1	AR Indirect Addressing Mode	3-6
		3.4.2	Dual AR Indirect Addressing Mode	3-14
		3.4.3	CDP Indirect Addressing Mode	3-16
		3.4.4	Coefficient Indirect Addressing Mode	3-19
	3.5	Circula	ar Addressing	3-21
4	Instr	ruction S	Set Summary	4-1
•			ummary of the TMS320C55x DSP algebraic instruction set.	
			•	
5			Set Descriptions	5-1
			mation on the TMS320C55x DSP algebraic instruction set.	
			ance (abdst)	
	Abso	olute Valu	Je	5-4
	Addi	tion		5-7
	Addi	tion with	Absolute Value	5-27
	Addi	tion with	Parallel Store Accumulator Content to Memory	5-29
	Addi	tion or S	ubtraction Conditionally (adsc)	5-31
	Addi	tion or S	ubtraction Conditionally with Shift (ads2c)	5-33
	Addi	tion, Sub	straction, or Move Accumulator Content Conditionally (adsc)	5-36
	Bitwi	ise AND		5-38
			Memory with Immediate Value and Compare to Zero	
	Bitwi	ise OR .		5-48
	Bitwi	ise Exclu	sive OR (XOR)	5-57
	Bran	ch Cond	itionally (if goto)	5-66
	Bran	ch Unco	nditionally (goto)	5-70
	Bran	ich on Au		F 74
			uxiliary Register Not Zero (if goto)	5-74
	Oun	Contaition	uxiliary Register Not Zero (if goto)	
				5-77
	Call	Uncondi	nally (if call)	5-77 5-83

Clear Memory Bit	
Clear Status Register Bit	. 5-90
Compare Accumulator, Auxiliary, or Temporary Register Content	. 5-93
Compare Accumulator, Auxiliary, or Temporary Register Content with AND	. 5-95
Compare Accumulator, Auxiliary, or Temporary Register Content with OR	5-100
Compare Accumulator, Auxiliary, or Temporary Register Content Maximum (max)	5-105
Compare Accumulator, Auxiliary, or Temporary Register Content Minimum (min)	5-108
Compare and Branch (compare goto)	5-111
Compare and Select Accumulator Content Maximum (max_diff)	5-114
Compare and Select Accumulator Content Minimum (min_diff)	5-120
Compare Memory with Immediate Value	5-126
Complement Accumulator, Auxiliary, or Temporary Register Bit (cbit)	5-128
Complement Accumulator, Auxiliary, or Temporary Register Content	5-129
Complement Memory Bit (cbit)	5-130
Compute Exponent of Accumulator Content (exp)	5-131
Compute Mantissa and Exponent of Accumulator Content (mant, exp)	5-132
Count Accumulator Bits (count)	5-134
Dual 16-Bit Additions	5-135
Dual 16-Bit Addition and Subtraction	5-140
Dual 16-Bit Subtractions	5-145
Dual 16-Bit Subtraction and Addition	5-154
Execute Conditionally (if execute)	5-159
Expand Accumulator Bit Field (field_expand)	5-166
Extract Accumulator Bit Field (field_extract)	5-167
Finite Impulse Response Filter, Antisymmetrical (firsn)	5-168
Finite Impulse Response Filter, Symmetrical (firs)	5-170
Idle	5-172
Least Mean Square (Ims)	5-173
Least Mean Square (Imsf)	5-175
Linear Addressing Qualifier (linear)	5-179
Load Accumulator from Memory	5-180
Load Accumulator from Memory with Parallel Store Accumulator Content to Memory	5-189
Load Accumulator Pair from Memory	5-191
Load Accumulator with Immediate Value	5-196
Load Accumulator, Auxiliary, or Temporary Register from Memory	5-199
Load Accumulator, Auxiliary, or Temporary Register with Immediate Value	5-205
Load Auxiliary or Temporary Register Pair from Memory	5-209
Load CPU Register from Memory	5-210
Load CPU Register with Immediate Value	5-213
Load Extended Auxiliary Register from Memory	5-215
Load Extended Auxiliary Register with Immediate Value	5-216
Load Memory with Immediate Value	5-217
Lock Access Qualifier	
Memory Delay (delay)	5-220

Memory-Mapped Register Access Qualifier (mmap)	5-221
Modify Auxiliary Register Content (mar)	5-222
Modify Auxiliary Register Content with Parallel Multiply	5-224
Modify Auxiliary Register Content with Parallel Multiply and Accumulate	5-226
Modify Auxiliary Register Content with Parallel Multiply and Subtract	5-231
Modify Auxiliary or Temporary Register Content (mar)	5-233
Modify Auxiliary or Temporary Register Content by Addition (mar)	5-237
Modify Auxiliary or Temporary Register Content by Subtraction (mar)	5-241
Modify Data Stack Pointer	
Modify Extended Auxiliary Register Content (mar)	5-246
Modify Extended Auxiliary Register Content by Addition (mar)	5-249
Modify Extended Auxiliary Register Content by Subtraction (mar)	5-251
Move Accumulator Content to Auxiliary or Temporary Register	5-253
Move Accumulator, Auxiliary, or Temporary Register Content	5-254
Move Auxiliary or Temporary Register Content to Accumulator	5-256
Move Auxiliary or Temporary Register Content to CPU Register	5-257
Move CPU Register Content to Auxiliary or Temporary Register	5-259
Move Extended Auxiliary Register Content	5-261
Move Memory to Memory	5-262
Multiply	5-269
Multiply with Parallel Multiply and Accumulate	5-283
Multiply with Parallel Multiply and Subtract	5-295
Multiply with Parallel Store Accumulator Content to Memory	5-305
Multiply and Accumulate (MAC)	5-308
Multiply and Accumulate with Parallel Delay	5-325
Multiply and Accumulate with Parallel Load Accumulator from Memory	5-327
Multiply and Accumulate with Parallel Multiply	5-329
Multiply and Accumulate with Parallel Multiply and Subtract	5-347
Multiply and Accumulate with Parallel Store Accumulator Content to Memory	5-367
Multiply and Subtract	5-369
Multiply and Subtract with Parallel Load Accumulator from Memory	5-379
Multiply and Subtract with Parallel Multiply	5-381
Multiply and Subtract with Parallel Multiply and Accumulate	5-390
Multiply and Subtract with Parallel Store Accumulator Content to Memory	5-401
Negate Accumulator, Auxiliary, or Temporary Register Content	5-403
No Operation (nop)	5-405
Parallel Modify Auxiliary Register Contents (mar)	5-406
Parallel Multiplies	5-407
Parallel Multiply and Accumulates	5-419
Parallel Multiply and Subtracts	
Peripheral Port Register Access Qualifiers	5-466
Pop Accumulator or Extended Auxiliary Register Content from Stack Pointers (popboth) .	5-468
Pop Top of Stack (pop)	5-469
Push Accumulator or Extended Auxiliary Register Content to Stack Pointers (pshboth)	5-476

Duch to Top of Stock (puch)	E 177
Push to Top of Stack (push)	
Repeat Single Instruction Conditionally (while/repeat)	
Repeat Single Instruction Unconditionally (repeat)	
Repeat Single Instruction Unconditionally and Decrement CSR (repeat)	
Repeat Single Instruction Unconditionally and Increment CSR (repeat)	
• • • • • • • • • • • • • • • • • • • •	
Return Conditionally (if return)	
Return from Interrupt (return_int)	
Rotate Left Accumulator, Auxiliary, or Temporary Register Content	
Rotate Right Accumulator, Auxiliary, or Temporary Register Content	
Round Accumulator Content (rnd)	
Saturate Accumulator Content (saturate)	
Set Accumulator, Auxiliary, or Temporary Register Bit	
Set Memory Bit	
Set Status Register Bit	
Shift Accumulator Content Conditionally (sftc)	
Shift Accumulator Content Logically	
Shift Accumulator, Auxiliary, or Temporary Register Content Logically	
Signed Shift of Accumulator Content	
Signed Shift of Accumulator, Auxiliary, or Temporary Register Content	
Software Interrupt (intr)	
Software Reset (reset)	
Software Trap (trap)	
Square	
Square and Accumulate	
Square and Subtract	
Square Distance (sqdst)	
Store Accumulator Content to Memory	
Store Accumulator Pair Content to Memory	
Store Accumulator, Auxiliary, or Temporary Register Content to Memory	
Store Auxiliary or Temporary Register Pair Content to Memory	
Store CPU Register Content to Memory	
Store Extended Auxiliary Register Content to Memory	
Subtract Conditionally (subc)	. 5-601
Subtraction	
Subtraction with Parallel Store Accumulator Content to Memory	
Swap Accumulator Content (swap)	5-629
Swap Accumulator Pair Content (swap)	5-630
Swap Auxiliary Register Content (swap)	5-631
Swap Auxiliary Register Pair Content (swap)	. 5-632
Swap Auxiliary and Temporary Register Content (swap)	
Swap Auxiliary and Temporary Register Pair Content (swap)	. 5-635
Swap Auxiliary and Temporary Register Pairs Content (swap)	5-637

#### Contents

	Swap Temporary Register Content (swap)	
	Swap Temporary Register Pair Content (swap)	5-640
	Test Accumulator, Auxiliary, or Temporary Register Bit	5-641
	Test Accumulator, Auxiliary, or Temporary Register Bit Pair	5-643
	Test Memory Bit	
	Test and Clear Memory Bit	5-648
	Test and Complement Memory Bit	5-649
	Test and Set Memory Bit	5-650
6	Instruction Opcodes in Sequential Order	6-1
	The opcode in sequential order for each TMS320C55x DSP instruction syntax.	
	6.1 Instruction Set Opcodes	6-2
	6.2 Instruction Set Opcode Symbols and Abbreviations	
7	Cross-Reference of Algebraic and Mnemonic Instruction Sets	7-1
	Cross-Reference of TMS320C55x DSP Algebraic and Mnemonic Instruction Sets.	
Ω.	Index	ndev-1
J		

# **Figures**

5–1	Status Registers Bit Mapping	5-92
5–2	Legal Uses of Repeat Block of Instructions Unconditionally (localrepeat)	
	Instruction	. 5-488
5–3	Status Registers Bit Mapping	5-526
5–4	Effects of a Software Reset on Status Registers	. 5-554
	<b>T</b> _	<b>L</b> I
	la	bles

1–1	Instruction Set Terms, Symbols, and Abbreviations	1-2
1–2	Operators Used in Instruction Set	
1–3	Instruction Set Conditional (cond) Field	
1–4	Nonrepeatable Instructions	
3–1	Addressing-Mode Operands	
3–2	Absolute Addressing Modes	
3–2 3–3	Direct Addressing Modes	
3–4	Indirect Addressing Modes	
3–5	DSP Mode Operands for the AR Indirect Addressing Mode	
3–6	Control Mode Operands for the AR Indirect Addressing Mode	
3–7	Dual AR Indirect Operands	
3–8	CDP Indirect Operands	
3–9	Coefficient Indirect Operands	
3–10	Circular Addressing Pointers	
4–1	Algebraic Instruction Set Summary	
5–1	Opcodes for Load CPU Register from Memory Instruction	
5–2	Opcodes for Load CPU Register with Immediate Value Instruction	
5–3	Opcodes for Move Auxiliary or Temporary Register Content to CPU Register	0211
0 0	Instruction	5-258
5–4	Opcodes for Move CPU Register Content to Auxiliary or Temporary Register	
	Instruction	
5–5	Effects of a Software Reset on DSP Registers	5-552
5–6	Opcodes for Store CPU Register Content to Memory Instruction	5-599
6–1	Instruction Set Opcodes	
6–2	Instruction Set Opcode Symbols and Abbreviations	6-19
7–1	Cross-Reference of Algebraic and Mnemonic Instruction Sets	7-2

## **Chapter 1**

# Terms, Symbols, and Abbreviations

This chapter lists and defines the terms, symbols, and abbreviations used in the TMS320C55 $x^{\text{TM}}$  DSP algebraic instruction set summary and in the individual instruction descriptions. Also provided are instruction set notes and rules and a list of nonrepeatable instructions.

# Topic Page 1.1 Instruction Set Terms, Symbols, and Abbreviations 1-2 1.2 Instruction Set Conditional (cond) Fields 1-7 1.3 Affect of Status Bits 1-9 1.4 Instruction Set Notes and Rules 1-14 1.5 Nonrepeatable Instructions 1-20

## 1.1 Instruction Set Terms, Symbols, and Abbreviations

Table 1–1 lists the terms, symbols, and abbreviations used and Table 1–2 lists the operators used in the instruction set summary and in the individual instruction descriptions.

Table 1–1. Instruction Set Terms, Symbols, and Abbreviations

Symbol	Meaning
[]	Optional operands
ACB	Bus that brings D-unit registers to A-unit and P-unit operators
ACOVx	Accumulator overflow status bit: ACOV0, ACOV1, ACOV2, ACOV3
ACw, ACx, ACy, ACz	Accumulator: AC0, AC1, AC2, AC3
ARn_mod	Content of selected auxiliary register (ARn) is premodified or postmodified in the address generation unit.
ARx, ARy	Auxiliary register: AR0, AR1, AR2, AR3, AR4, AR5, AR6, AR7
AU	A unit
Baddr	Register bit address
BitIn	Shifted bit in: Test control flag 2 (TC2) or CARRY status bit
BitOut	Shifted bit out: Test control flag 2 (TC2) or CARRY status bit
BORROW	Logical complement of CARRY status bit
C, Cycles	Execution in cycles. For conditional instructions, x/y field means: x cycle, if the condition is true. y cycle, if the condition is false.
CA	Coefficient address generation unit
CARRY	Value of CARRY status bit
Cmem	Coefficient indirect operand referencing a 16-bit or 32-bit value in data space
cond	Condition based on accumulator value (ACx), auxiliary register (ARx) value, temporary register (Tx) value, test control (TCx) flag, or CARRY status bit. See section 1.2.
CR	Coefficient Read bus
CSR	Computed single-repeat register

Table 1–1. Instruction Set Terms, Symbols, and Abbreviations (Continued)

Symbol	Meaning
DA	Data address generation unit
DR	Data Read bus
dst	Destination accumulator (ACx), lower 16 bits of auxiliary register (ARx), or temporary register (Tx): AC0, AC1, AC2, AC3 AR0, AR1, AR2, AR3, AR4, AR5, AR6, AR7 T0, T1, T2, T3
DU	D unit
DW	Data Write bus
Dx	Data address label coded on x bits (absolute address)
E	Indicates if the instruction contains a parallel enable bit.
kx	Unsigned constant coded on x bits
Kx	Signed constant coded on x bits
Lmem	Long-word single data memory access (32-bit data access). Same legal inputs as Smem.
lx	Program address label coded on x bits (unsigned offset relative to program counter register)
Lx	Program address label coded on x bits (signed offset relative to program counter register)
M40	If the optional M40 keyword is applied to the instruction, the instruction provides the option to locally set M40 to 1 for the execution of the instruction
Operator	Operator(s) used by an instruction.
Pipe, Pipeline	Pipeline phase in which the instruction executes:  AD Address  D Decode  R Read  X Execute
Px	Program or data address label coded on x bits (absolute address)
RELOP	Relational operators:
	== equal to < less than >= greater than or equal to != not equal to

Table 1–1. Instruction Set Terms, Symbols, and Abbreviations (Continued)

Symbol	Meaning
rnd	If the optional rnd keyword is applied to the instruction, rounding is performed in the instruction
RPTC	Single-repeat counter register
S, Size	Instruction size in bytes.
SA	Stack address generation unit
saturate	If the optional saturate keyword is applied to the input operand, the 40-bit output of the operation is saturated
SHFT	4-bit immediate shift value, 0 to 15
SHIFTW	6-bit immediate shift value, -32 to +31
Smem	Word single data memory access (16-bit data access)
SP	Data stack pointer
src	Source accumulator (ACx), lower 16 bits of auxiliary register (ARx), or temporary register (Tx): AC0, AC1, AC2, AC3 AR0, AR1, AR2, AR3, AR4, AR5, AR6, AR7 T0, T1, T2, T3
SSP	System stack pointer
STx	Status register: ST0, ST1, ST2, ST3
TAx, TAy	Auxiliary register (ARx) or temporary register (Tx): AR0, AR1, AR2, AR3, AR4, AR5, AR6, AR7 T0, T1, T2, T3
TCx, TCy	Test control flag: TC1, TC2
TRNx	Transition register: TRN0, TRN1
Тх, Ту	Temporary register (Tx): T0, T1, T2, T3
uns	If the optional uns keyword is applied to the input operand, the operand is zero extended
XACdst	Destination extended register: All 23 bits of coefficient data pointer (XCDP), and extended auxiliary register (XARx): XAR0, XAR1, XAR2, XAR3, XAR4, XAR5, XAR6, XAR7

Table 1–1. Instruction Set Terms, Symbols, and Abbreviations (Continued)

Symbol	Meaning
XACsrc	Source extended register: All 23 bits of coefficient data pointer (XCDP), and extended auxiliary register (XARx): XAR0, XAR1, XAR2, XAR3, XAR4, XAR5, XAR6, XAR7
XAdst	Destination extended register: All 23 bits of data stack pointer (XSP), system stack pointer (XSSP), and data page pointer (XDP)
XARx	All 23 bits of extended auxiliary register: XAR0, XAR1, XAR2, XAR3, XAR4, XAR5, XAR6, XAR7
XAsrc	Source extended register: All 23 bits of data stack pointer (XSP), system stack pointer (XSSP), and data page pointer (XDP)
xdst	Accumulator: AC0, AC1, AC2, AC3
	Destination extended register: All 23 bits of data stack pointer (XSP), system stack pointer (XSSP), data page pointer (XDP), coefficient data pointer (XCDP), and extended auxiliary register (XARx):  XAR0, XAR1, XAR2, XAR3, XAR4, XAR5, XAR6, XAR7
xsrc	Accumulator: AC0, AC1, AC2, AC3
	Source extended register: All 23 bits of data stack pointer (XSP), system stack pointer (XSSP), data page pointer (XDP), coefficient data pointer (XCDP), and extended auxiliary register (XARx): XAR0, XAR1, XAR2, XAR3, XAR4, XAR5, XAR6, XAR7
Xmem, Ymem	Indirect dual data memory access (two data accesses)

Table 1-2. Operators Used in Instruction Set

Symbols		Operators	Evaluation
+ -	~	Unary plus, minus, 1s complement	Right to left
* /	%	Multiplication, division, modulo	Left to right
+	_	Addition, subtraction	Left to right
<<	>>	Signed left shift, right shift	Left to right
< < <	>>>	Logical left shift, logical right shift	Left to right
<	<=	Less than, less than or equal to	Left to right
>	>=	Greater than, greater than or equal to	Left to right
==	!=	Equal to, not equal to	Left to right
&		Bitwise AND	Left to right
1		Bitwise OR	Left to right
۸		Bitwise exclusive OR (XOR)	Left to right

**Note:** Unary +, -, and \* have higher precedence than the binary forms.

## 1.2 Instruction Set Conditional (cond) Fields

Table 1–3 lists the testing conditions available in the cond field of the conditional instructions.

Table 1-3. Instruction Set Conditional (cond) Field

Bit or Register	Condition (cond) Field	For Condition to be True	
Accumulator	Tests the accumulator (ACx) content against 0. The comparison against 0 depends on M40 status bit:		
	$\Box$ If M40 = 0, ACx(31–0)	is compared to 0.	
	$\Box$ If M40 = 1, ACx(39–0)	is compared to 0.	
	ACx == #0	ACx content is equal to 0	
	ACx < #0	ACx content is less than 0	
	ACx > #0	ACx content is greater than 0	
	ACx != #0	ACx content is not equal to 0	
	ACx <= #0	ACx content is less than or equal to 0	
	ACx >= #0	ACx content is greater than or equal to 0	
Accumulator Overflow Status Bit	Tests the accumulator overflow status bit (ACOVx) against 1; when a optional! symbol is used before the bit designation, the bit can be to against 0. When this condition is used, the corresponding ACOVx is cleared to 0.		
	overflow(ACx)	ACOVx bit is set to 1	
	!overflow(ACx)	ACOVx bit is cleared to 0	
Auxiliary Register	Tests the auxiliary register (	ARx) content against 0.	
	ARx == #0	ARx content is equal to 0	
	ARx < #0	ARx content is less than 0	
	ARx > #0	ARx content is greater than 0	
	ARx != #0	ARx content is not equal to 0	
	ARx <= #0	ARx content is less than or equal to 0	
	ARx >= #0	ARx content is greater than or equal to 0	
CARRY Status Bit		against 1; when the optional ! symbol is used he bit can be tested against 0.	
	CARRY	CARRY bit is set to 1	
	!CARRY	CARRY bit is cleared to 0	

Table 1–3. Instruction Set Conditional (cond) Field (Continued)

Bit or Register	Condition (cond) Field	For Condition to be True
Temporary Register	Tests the temporary registe	er (Tx) content against 0.
	Tx == #0	Tx content is equal to 0
	Tx < #0	Tx content is less than 0
	Tx > #0	Tx content is greater than 0
	Tx != #0	Tx content is not equal to 0
	Tx <= #0	Tx content is less than or equal to 0
	Tx >= #0	Tx content is greater than or equal to 0
Test Control Flags		(TC1 and TC2) independently against 1; when ed before the flag designation, the flag can be set 0.
	TCx	TCx flag is set to 1
	!TCx	TCx flag is cleared to 0
	TC1 and TC2 can be comb logical bit combinations:	oined with an AND (&), OR ( ), and XOR (^)
	TC1 & TC2	TC1 AND TC2 is equal to 1
	!TC1 & TC2	TC1 AND TC2 is equal to 1
	TC1 & !TC2	TC1 AND TC2 is equal to 1
	!TC1 & !TC2	TC1 AND TC2 is equal to 1
	TC1   TC2	TC1 OR TC2 is equal to 1
	!TC1   TC2	TC1 OR TC2 is equal to 1
	TC1   !TC2	TC1 OR TC2 is equal to 1
	!TC1   !TC2	TC1 OR TC2 is equal to 1
	T0.4.1 T00	TO 1 VOD TOO!
	TC1 ^ TC2	TC1 XOR TC2 is equal to 1
	!TC1 ^ TC2	TC1 XOR TC2 is equal to 1
	TC1 ^ !TC2	TC1 XOR TC2 is equal to 1
	!TC1 ^ !TC2	TC1 XOR TC2 is equal to 1

## 1.3 Affect of Status Bits

1.3.1	Accumulator	Ov	erflow Status Bit (ACOVx)
		The	e ACOV[0-3] depends on M40:
			When M40 = 0, overflow is detected at bit position 31
			When M40 = 1, overflow is detected at bit position 39
		If a	n overflow is detected, the destination accumulator overflow status bit is set l.
1.3.2	C54CM Status	s B	it
			When C54CM = 0, the enhanced mode, the CPU supports code originally developed for a TMS320C55 $x^{TM}$ DSP.
			When C54CM = 1, the compatible mode, all the C55x CPU resources remain available; therefore, as you translate code, you can take advantage of the additional features on the C55x DSP to optimize your code. This mode must be set when you are porting code that was originally developed for a TMS320C54 $x^{TM}$ DSP.
1.3.3	CARRY Statu	s B	it
			When M40 = 0, the carry/borrow is detected at bit position 31
			When M40 = 1, the carry/borrow is detected at bit position 39
			en performing a logical shift or signed shift that affects the CARRY status and the shift count is zero, the CARRY status bit is cleared to 0.
1.3.4	FRCT Status	Bit	
			When $FRCT = 0$ , the fractional mode is $OFF$ and results of multiply operations are not shifted.
			When FRCT = 1, the fractional mode is ON and results of multiply operations are shifted left by 1 bit to eliminate an extra sign bit.
1.3.5	INTM Status B	Bit	
			e INTM bit globally enables or disables the maskable interrupts. This bit has effect on nonmaskable interrupts (those that cannot be blocked by software).
			When INTM = 0, all unmasked interrupts are enabled.
			When INTM = 1, all maskable interrupts are disabled.

#### 1.3.6 M40 Status Bit

- $\square$  When M40 = 0:
  - overflow is detected at bit position 31
  - the carry/borrow is detected at bit position 31
  - saturation values are 00 7FFF FFFFh (positive overflow) or FF 8000 0000h (negative overflow)
  - TMS320C54x<sup>™</sup> DSP compatibility mode
  - for conditional instructions, the comparison against 0 (zero) is performed on 32 bits, ACx(31–0)
- $\square$  When M40 = 1:
  - overflow is detected at bit position 39
  - the carry/borrow is detected at bit position 39
  - saturation values are 7F FFFF FFFFh (positive overflow) or 80 0000 0000h (negative overflow)
  - for conditional instructions, the comparison against 0 (zero) is performed on 40 bits, ACx(39–0)

#### 1.3.6.1 M40 Status Bit When Sign Shifting

In D-unit shifter:

- ☐ When shifting to the LSBs:
  - when M40 = 0, the input to the shifter is modified according to SXMD and then the modified input is shifted according to the shift quantity:
    - if SXMD = 0, 0 is substituted for the guard bits (39–32) as the input, instead of ACx(39–32), to the shifter
    - if SXMD = 1, bit 31 of the source operand is substituted for the guard bits (39–32) as the input, instead of ACx(39–32), to the shifter
  - bit 39 is extended according to SXMD
  - the shifted-out bit is extracted at bit position 0
- ☐ When shifting to the MSBs:
  - 0 is inserted at bit position 0
  - if M40 = 0, the shifted-out bit is extracted at bit position 31
  - $\blacksquare$  if M40 = 1, the shifted-out bit is extracted at bit position 39

	After shifting, unless otherwise noted, when $M40 = 0$ :
	<ul> <li>overflow is detected at bit position 31 (if an overflow is detected, the destination ACOVx bit is set)</li> </ul>
	■ the carry/borrow is detected at bit position 31
	■ if SATD = 1, when an overflow is detected, ACx saturation values are 00 7FFF FFFFh (positive overflow) or FF 8000 0000h (negative overflow)
	■ TMS320C54x <sup>™</sup> DSP compatibility mode
	After shifting, unless otherwise noted, when M40 = 1:
	<ul> <li>overflow is detected at bit position 39 (if an overflow is detected, the destination ACOVx bit is set)</li> </ul>
	■ the carry/borrow is detected at bit position 39
	■ if SATD = 1, when an overflow is detected, ACx saturation values are 7F FFFF FFFFh (positive overflow) or 80 0000 0000h (negative overflow)
In A	-unit ALU:
	When shifting to the LSBs, bit 15 is sign extended
	When shifting to the MSBs, 0 is inserted at bit position 0
	After shifting, unless otherwise noted:
	<ul> <li>overflow is detected at bit position 15 (if an overflow is detected, the destination ACOVx bit is set)</li> </ul>
	■ if SATA = 1, when an overflow is detected, register saturation values are 7FFFh (positive overflow) or 8000h (negative overflow)
it W	hen Logically Shifting

#### 1.3.6.2 M40 Status Bi

In D-unit shifter:

- ☐ When shifting to the LSBs:
  - $\blacksquare$  if M40 = 0, 0 is inserted at bit position 31 and the guard bits (39–32) of the destination accumulator are cleared
  - if M40 = 1, 0 is inserted at bit position 39
  - the shifted-out bit is extracted at bit position 0 and stored in the CARRY status bit

- ☐ When shifting to the MSBs:
  - 0 is inserted at bit position 0
  - if M40 = 0, the shifted-out bit is extracted at bit position 31 and stored in the CARRY status bit, and the guard bits (39–32) of the destination accumulator are cleared
  - if M40 = 1, the shifted-out bit is extracted at bit position 39 and stored in the CARRY status bit

#### In A-unit ALU:

- ☐ When shifting to the LSBs:
  - 0 is inserted at bit position 15
  - the shifted-out bit is extracted at bit position 0 and stored in the CARRY status bit
- ☐ When shifting to the MSBs:
  - 0 is inserted at bit position 0
  - the shifted-out bit is extracted at bit position 15 and stored in the CARRY status bit

#### 1.3.7 RDM Status Bit

When the optional rnd or R keyword is applied to the instruction, then rounding is performed in the D-unit shifter. This is done according to RDM:

- When RDM = 0, the biased rounding to the infinite is performed. 8000h  $(2^{15})$  is added to the 40-bit result of the shift result.
- When RDM = 1, the unbiased rounding to the nearest is performed. According to the value of the 17 LSBs of the 40-bit result of the shift result, 8000h (2¹⁵) is added:

```
if( 8000h < bit(15-0) < 10000h)
   add 8000h to the 40-bit result of the shift result.
else if( bit(15-0) == 8000h)
   if( bit(16) == 1)
   add 8000h to the 40-bit result of the shift result.</pre>
```

If a rounding has been performed, the 16 lowest bits of the result are cleared to 0.

#### 1.3.8 SATA Status Bit

This status bit controls operations performed in the A unit.

- $\square$  When SATA = 0, no saturation is performed.
- ☐ When SATA = 1 and an overflow is detected, the destination register is saturated to 7FFFh (positive overflow) or 8000h (negative overflow).

#### 1.3.9 SATD Status Bit

1.3.3	OAID Status	Dit	
		Thi	s status bit controls operations performed in the D unit.
			When SATD = 0, no saturation is performed.
			When SATD = 1 and an overflow is detected, the destination register is saturated.
1.3.10	SMUL Status	Bit	
			When SMUL = 0, the saturation mode is OFF.
			When SMUL = 1, the saturation mode is ON. When SMUL = 1, FRCT = 1, and SATD = 1, the result of $18000h \times 18000h$ is saturated to $00.7FFFFFFF$ (regardless of the value of the M40 bit). This forces the product of the two negative numbers to be a positive number. For multiply-and-accumulate/subtract instructions, the saturation is performed after the multiplication and before the addition/subtraction.
1.3.11	SXMD Status	Bit	
		Thi	s status bit controls operations performed in the D unit.
			When SXMD = 0, input operands are zero extended.

 $\Box$  When SXMD = 1, input operands are sign extended.

## 1.3.12 Test Control Status Bit (TCx)

The test/control status bits (TC1 or TC2) hold the result of a test performed by the instruction.

#### 1.4 Instruction Set Notes and Rules

#### 1.4.1 Notes

Algebraic syntax keywords and operand modifiers are case insensitive. You can write:

```
abdst(*AR0, *ar1, AC0, ac1)
or
aBdST(*ar0, *aR1, aC0, Ac1)
```

- Operands for commutative operations (+, \*, &, |, ^) can be arranged in any order.
- ☐ Expression qualifiers can be specified in any order. For example, these two instructions are equivalent:

```
AC0 = m40 (rnd (uns (*AR0) * uns (*AR1)))
AC0 = rnd (m40 (uns (*AR0) * uns (*AR1)))
```

Algebraic instructions must use parenthesis in the exact form shown in the instruction set. For example, this instruction is legal:

```
AC0 = AC0 + (AC1 << T0)
```

while both of these instructions are illegal:

```
AC0 = AC0 + ((AC1 << T0))

AC0 = AC0 + AC1 << T0
```

#### 1.4.2 Rules

☐ Simple instructions are not allowed to span multiple lines. One exception, single instructions that use the "," notation to imply parallelism. These instructions may be split up following the "," notation.

The following example shows a single instruction (dual multiply) occupying two lines:

```
ACx = m40(rnd(uns(Xmem) * uns(coef(Cmem)))),
ACy = m40(rnd(uns(Ymem) * uns(coef(Cmem))))
```

☐ User-defined parallelism instructions (using || notation) are allowed to span multiple lines. For example, all of the following instructions are legal:

☐ The block repeat syntax uses braces to delimit the block that is to be repeated:

```
blockrepeat {
    instr
    instr
    instr
}
localrepeat {
    instr
    instr
    instr
    instr
}
```

The left opening brace must appear on the same line as the repeat keyword. The right closing brace must appear alone on a line (trailing comments allowed).

Note that a label placed just inside the closing brace of the loop is effectively outside the loop. The following two code sequences are equivalent:

A label is the address of the first construct following the label that gets assembled into code in the object file. A closing brace does not generate any code and so the label marks the address of the first instruction that generates code, that is, instr3.

In this example, "goto Label" exits the loop, which is somewhat unintuitive:

```
localrepeat {
     goto Label
     instr2
Label:
     }
     instr3
```

#### 1.4.2.1 Reserved Words

Register names and algebraic syntax keywords are reserved. They may not be used as names of identifiers, labels, etc.

#### 1.4.2.2 Literal and Address Operands

Literals in the algebraic strings are denoted as K or k fields. In the Smem address modes that require an offset, the offset is also a literal (K16 or k3). 8-bit and 16-bit literals are allowed to be linktime-relocatable; for other literals, the value must be known at assembly time.

Addresses are the elements of the algebraic strings denoted by P, L, and I. Further, 16-bit and 24-bit absolute address Smem modes are addresses, as is the dma Smem mode, denoted by the '@' syntax. Addresses may be assembly-time constants or symbolic linktime-known constants or expressions.

Both literals and addresses follow syntax rule 1. For addresses only, rules 2 and 3 also apply.

#### Rule 1

A valid address or literal is a # followed by one of the following:

$\Box$	a number	(#123)

an identifier (#FOO)

a parenthesized expression (#(FOO + 2))

Note that # is not used inside the expression.

#### Rule 2

When an address is used in a dma, the address does not need to have a leading #, be it a number, a symbol or an expression. These are all legal:

```
@#123
@123
@#foo
@foo
@#(foo+2)
```

@(foo+2)

#### Rule 3

When used in contexts other than dma (such as branch targets or Smemabsolute address), addresses generally need a leading #. As a convenience, the # may be omitted in front of an identifier. These are all legal:

Branch	Absolute Address
goto #123	*(#123)
goto #foo	*(#foo)
goto foo	*(foo)
goto #(foo+2)	*(#(foo+2))
These are illegal:	
goto 123	*(123)
goto (foo+2)	*((foo+2))

#### 1.4.2.3 Memory Operands

- ☐ Syntax of Smem is the same as that of Lmem or Baddr.
- ☐ In the following instruction syntaxes, Smem **cannot** reference to a memory-mapped register (MMR). No instruction can access a byte within a memory-mapped register. If Smem is an MMR in one of the following syntaxes, the DSP sends a hardware bus-error interrupt (BERRINT) request to the CPU.

```
dst = uns(high_byte(Smem))
dst = uns(low_byte(Smem))
ACx = low_byte(Smem) << #SHIFTW
ACx = high_byte(Smem) << #SHIFTW
high_byte(Smem) = src
low byte(Smem) = src</pre>
```

- Syntax of Xmem is the same as that of Ymem.
- ☐ Syntax of coefficient operands, Cmem:

```
*CDP

*CDP+

*CDP-

*(CDP + T0), when C54CM = 0

*(CDP + AR0), when C54CM = 1
```

When an instruction uses a Cmem operand with paralleled instructions, the pointer modification of the Cmem operand must be the same for both instructions of the paralleled pair or the assembler generates an error. For example:

```
AC0 = AC0 + (*AR2+ * coef(*CDP+)),
AC1 = AC1 + (*AR3+ * coef(*CDP+))
```

☐ An optional mmr prefix is allowed to be specified for indirect memory operands, for example, mmr (\*ARO). This is an assertion by you that this is an access to a memory-mapped register. The assembler checks whether such access is legal in given circumstances.

The mmr prefix is supported for Xmem, Ymem, indirect Smem, indirect Lmem, and Cmem operands. It is not supported for direct memory operands; it is expected that an explicit mmap() parallel instruction is used in conjunction with direct memory operands to indicate MMR access.

Note that the mmr prefix is part of the syntax. It is an implementation restriction that mmr cannot exchange positions with other prefixes around the memory operand, such as dbl or uns. If several prefixes are specified, mmr must be the innermost prefix. Thus, uns(mmr(\*AR0)) is legal, but mmr(uns(\*AR0)) is not legal.

☐ The following indirect operands **cannot** be used for accesses to I/O space. An instruction using one of these operands requires a 2-byte extension for the constant. This extension would prevent the use of the port() qualifier needed to indicate an I/O-space access.

```
*ARn(#K16)
*+ARn(#K16)
*CDP(#K16)
*+CDP(#K16)
```

Also, the following instructions that include the delay operation cannot be used for accesses to I/O space:

```
delay(Smem)
ACx = rnd(ACx + (Smem * coef(Cmem))) [,T3 = Smem],
delay(Smem)
```

Any illegal access to I/O space will generate a hardware bus-error interrupt (BERRINT) to be handled by the CPU.

#### 1.4.2.4 Operand Modifiers

Operand modifiers look like function calls on operands. Note that uns is an operand modifier and an instruction modifier meaning unsigned. The operand modifier uns is used when the operand is modified on the way to the rest of the operation (multiply-and-accumulate). The instruction modifier uns is used when the whole operation is affected (multiply, register compare, compare and branch).

Modifier	Meaning
dbl	Access a true 32-bit memory operand
dual	Access a 32-bit memory operand for use as two independent 16-bit halves of the given operation
HI	Access upper 16 bits of the accumulator
high_byte	Access the high byte of the memory location
LO	Access lower 16 bits of the accumulator
low_byte	Access the low byte of the memory location
pair	Dual register access
rnd	Round
saturate	Saturate
uns	Unsigned operand

When an instruction uses a Cmem operand with paralleled instructions and the Cmem operand is defined as unsigned (uns), both Cmem operands of the paralleled pair must be defined as unsigned (and reciprocally).

When an instruction uses both Xmem and Ymem operands with paralleled instructions and the Xmem operand is defined as unsigned (uns), Ymem operand must also be defined as unsigned (and reciprocally).

#### 1.4.2.5 Operator Syntax Rules

Instructions that read and write the same operand can also be written in op-assign form. For example:

```
AC0 = AC0 + *AR4
```

can also be written:

This form is supported for these operations: +=, -=, &=, |=,  $^=$ 

Note that in certain instances use of op-assign notation results in ambiguous algebraic assembly. This happens if the op-assign operator is not delimited by white space, for example:

```
*AR0+=#4 is ambiguous, is it *AR0 += #4 or *AR0+ = #4?
```

The assembler always parses adjacent += as plus-assign; therefore, this instructions is parsed as \*AR0 += #4.

```
*AR0+=*AR1 is ambiguous, is it *AR0 += *AR1 or *AR0+ =*AR1?
```

Once again, the first form, \*AR0 += \*AR1, is used. This is not a valid instruction -- an error is printed.

## 1.5 Nonrepeatable Instructions

Table 1–4 lists the instructions that cannot be used in a repeatable instruction.

Table 1-4. Nonrepeatable Instructions

Instruction Description	Algebraic Syntax That Cannot Be Repeated
Branch Conditionally	if (cond) goto I4
	if (cond) goto L8
	if (cond) goto L16
	if (cond) goto P24
Branch Unconditionally	goto ACx
	goto L7
	goto L16
	goto P24
Branch on Auxiliary Register Not Zero	if (ARn_mod != #0) goto L16
Call Conditionally	if (cond) call L16
	if (cond) call P24
Call Unconditionally	call ACx
	call L16
	call P24
Clear Status Register Bit	bit(STx, k4) = #0
Compare and Branch	compare (uns(src RELOP K8)) goto L8
Execute Conditionally	if (cond) execute(AD_Unit)
	if (cond) execute(D_Unit)
Idle	idle
Load CPU Register from Memory	DP = Smem
	RETA = dbl(Lmem)
Load CPU Register with Immediate Value	DP = k16
Move CPU Register Content to Auxiliary or Temporary Register	TAx = RPTC
Repeat Block of Instructions Unconditionally	localrepeat{}
	blockrepeat{}
Repeat Single Instruction Conditionally	while (cond && (RPTC < k8)) repeat

Table 1-4. Nonrepeatable Instructions (Continued)

Instruction Description	Algebraic Syntax That Cannot Be Repeated
Repeat Single Instruction Unconditionally	repeat(k8)
	repeat(k16)
	repeat(CSR)
Repeat Single Instruction Unconditionally and Decrement CSR	repeat(CSR), CSR -= k4
Repeat Single Instruction Unconditionally and Increment CSR	repeat(CSR), CSR += TAx
	repeat(CSR), CSR += k4
Return Conditionally	if (cond) return
Return Unconditionally	return
Return from Interrupt	return_int
Round Accumulator Content	ACy = rnd(ACx)
Set Status Register Bit	bit(STx, k4) = #1
Software Interrupt	intr(k5)
Software Reset	reset
Software Trap	trap(k5)
Store CPU Register Content to Memory	dbl(Lmem) = RETA

# **Parallelism Features and Rules**

This chapter describes the parallelism features and rules of the TMS320C55x $^{\text{TM}}$  DSP algebraic instruction set.

Горіс	С	P	age
2.1	Parallelism Features		2-2
2.2	Parallelism Basics		2-3
2.3	Resource Conflicts		2-4
2.4	Soft-Dual Parallelism		2-5
2.5	Execute Conditionally Instructions		2-6
2.6	Other Exceptions		2-7

#### 2.1 Parallelism Features

The C55x<sup>™</sup> DSP architecture enables you to execute two instructions in parallel within the same cycle of execution. The types of parallelism are:

☐ Built-in parallelism within a single instruction.

Some instructions perform two different operations in parallel. A comma is used to separate the two operations. This type of parallelism is also called implied parallelism. For example:

```
AC0 = *AR0 * coef(*CDP),
AC1 = *AR1 * coef(*CDP)
```

This is a single instruction. The data referenced by AR0 is multiplied by the coefficient referenced by CDP. At the same time, the data referenced by AR1 is multiplied by the same coefficient (CDP).

User-defined parallelism between two instructions.

Two instructions may be paralleled by you or the C compiler. The parallel bars, ||, are used to separate the two instructions to be executed in parallel. For example:

```
AC1 = *AR1- * *AR2+
|| T1 = T1 ^ AR2
```

The first instruction performs a multiplication in the D-unit. The second instruction performs a logical operation in the A-unit ALU.

☐ Built-in parallelism can be combined with user-defined parallelism. Parenthesis separators can be used to determine boundaries of the two instructions. For example:

```
(AC2 = *AR3+ * AC1,
T3 = *AR3+)
| AR1 = #5
```

The first instruction includes implied parallelism. The second instruction is paralleled by you.

2-2 Parallelism Features and Rules

## 2.2 Parallelism Basics

In t	In the parallel pair, all of these constraints must be met:			
	No resource conflicts as detailed in section 2.3.			
	One instruction must have a parallel enable bit or the pair must qualify for soft-dual parallelism as detailed in section 2.4.			
	No memory operand may use an addressing mode that requires a constant that is 16 bits or larger:			
	<ul> <li>*abs16(#k16)</li> <li>*(#k23)</li> <li>*port(#k16)</li> <li>*ARn(K16)</li> <li>*+ARn(K16)</li> <li>*CDP(K16)</li> <li>*+CDP(K16)</li> </ul>			
	The following instructions cannot be in parallel:			
	<pre>if (cond) goto P24 if (cond) call P24 idle intr(k5) reset trap(k5)</pre>			
	Neither instruction in the parallel pair can use any of these instruction or operand modifiers:			
	<pre>circular() linear() mmap() readport() writeport()</pre>			
	A particular register or memory location can only be written once per pipeline phase. Violations of this rule take many forms. Loading the same register twice is a simple case. Other cases include:			

■ Conflicting address mode modifications (for example, \*AR2+ versus

■ Combining a SWAP instruction (modifies all of its registers) with any

other instruction that writes one of the same registers

\*AR2-)

		☐ Data stack pointer (XSP) or system stack pointer (XSSP) modifications cannot be combined with any of the following instructions:
		<ul> <li>Call Conditionally, (if (cond) call instructions)</li> <li>Call Unconditionally, (call instructions)</li> <li>Push to top of Stack (push instructions)</li> <li>Pop from top of Stack (pop instructions)</li> <li>Return Conditionally, (if (cond) return instructions)</li> <li>Return Unconditionally, (return instructions)</li> <li>Return from Interrupt, (return_int, instructions)</li> <li>trap or intr instructions</li> </ul>
		☐ When both instructions in a parallel pair modify a status bit, the value of that status bit becomes undefined.
2.3 F	Resource Co	nflicts
		Every instruction uses some set of operators, address generation units, and buses, collectively called resources, while executing. To determine which resources are used by a specific instruction, see Table 4–1. Two instructions in parallel use all the resources of the individual instructions. A resource conflict occurs when two instructions use a combination of resources that is not supported on the C55x device. This section details the resource conflicts.
2.3.1	Operators	
		You may use each of these operators only once:
		<ul> <li>□ D Unit ALU</li> <li>□ D Unit Shift</li> <li>□ D Unit Swap</li> <li>□ A Unit Swap</li> <li>□ A Unit ALU</li> <li>□ P Unit</li> </ul>
		For an instruction that uses multiple operators, any other instruction that uses one or more of those same operators may not be placed in parallel.
2.3.2	Address Gen	neration Units
		You may use no more than the indicated number of data address generation units:
		<ul> <li>2 Data Address (DA) Generation Units</li> <li>1 Coefficient Address (CA) Generation Unit</li> <li>1 Stack Address (SA) Generation Unit</li> </ul>

Parallelism Features and Rules

#### **2.3.3 Buses**

You may use no more than the indicated number of buses:

2 Data Read (DR) Buses
 1 Coefficient Read (CR) Bus
 2 Data Write (DW) Buses
 1 ACB Bus – brings D-unit registers to A-unit and P-unit operators

#### 2.4 Soft-Dual Parallelism

Instructions that reference memory operands do not have parallel enable bits. Two such instructions may still be combined with a type of parallelism called soft-dual parallelism. The constraints of soft-dual parallelism are:

- ☐ Both memory operands must meet the constraints of the dual AR indirect addressing mode (Xmem and Ymem), as described in section 3.4.2. The operands available for the dual AR indirect addressing mode are:
  - \*ARn
  - \*ARn+
  - \*ARn-
  - \*(ARn + AR0)
  - \*(ARn + T0)
  - \*(ARn AR0)
  - \*(ARn T0)
  - \*ARn(AR0)
  - \*ARn(T0)
  - \*(ARn + T1)
  - \*(ARn − T1)
- Neither instruction can contain any of the following:
  - Instructions embedding high\_byte(Smem) and low\_byte(Smem).
    - dst = uns(high byte(Smem))
    - dst = uns(low byte(Smem))
    - ACx = low\_byte(Smem) << #SHIFTW
    - ACx = high byte(Smem) << #SHIFTW
    - high byte(Smem) = src
    - low byte(Smem) = src

■ These instructions that read and write the same memory location:

```
    cbit(Smem, src)
    bit(Smem, src) = #0
    bit(Smem, src) = #1
    TCx = bit(Smem, k4), bit(Smem, k4) = #1
    TCx = bit(Smem, k4), bit(Smem, k4) = #0
    TCx = bit(Smem, k4), cbit(Smem, k4)
```

☐ With regard to soft-dual parallelism, the mar(Smem) and XAdst = mar(Smem) instructions have the same properties as any memory reference instruction.

#### 2.4.1 Soft-Dual Parallelism of MAR Instructions

Although the following modify auxiliary register (MAR) instructions do not reference memory and do not have parallel enable bits, they may be combined together or with any other memory reference instructions (not limited to Xmem/Ymem) to form soft-dual parallelism.

```
    mar(TAy + TAx)
    mar(TAx + k8)
    mar(TAy = TAx)
    mar(TAx = k8)
    mar(TAy - TAx)
    mar(TAx - k8)
```

Note that this is not the full list of MAR instructions; instructions mar(TAx = D16) is not included.

## 2.5 Execute Conditionally Instructions

The parallelization of the execute conditionally, if (cond) execute, instructions does not adhere to the descriptions in this chapter. All of the specific instances of legal parallelism are covered in the execute conditionally descriptions in Chapter 5.

2-6 Parallelism Features and Rules

# 2.6 Other Exceptions

The following are other exceptions not covered elsewhere in this chapter.

- ☐ An instruction that reads the repeat counter register (RPTC) may not be combined with any single-repeat instruction:
  - repeat()
  - repeat(CSR)
  - $\blacksquare$  while (cond) repeat

SWPU068E

2-8 Parallelism Features and Rules SWPU068E

# Chapter 3

# **Introduction to Addressing Modes**

This chapter provides an introduction to the addressing modes of the TMS320C55 $x^{\text{TM}}$  DSP.

Горі	C	Page
3.1	Introduction to the Addressing Modes	3-2
3.2	Absolute Addressing Modes	3-3
3.3	Direct Addressing Modes	3-4
3.4	Indirect Addressing Modes	3-6
3.5	Circular Addressing	3-21

## 3.1 Introduction to the Addressing Modes

The TMS320C55x DSP supports three types of addressing modes that enable flexible access to data memory, to memory-mapped registers, to register bits, and to I/O space:

- ☐ The absolute addressing mode allows you to reference a location by supplying all or part of an address as a constant in an instruction.
- ☐ The direct addressing mode allows you to reference a location using an address offset.
- ☐ The indirect addressing mode allows you to reference a location using a pointer.

Each addressing mode provides one or more types of operands. An instruction that supports an addressing-mode operand has one of the following syntax elements listed in Table 3–1.

Table 3–1. Addressing-Mode Operands

Syntax Element(s)	Description
Baddr	When an instruction contains Baddr, that instruction can access one or two bits in an accumulator (AC0–AC3), an auxiliary register (AR0–AR7), or a temporary register (T0–T3). Only the register bit test/set/clear/complement instructions support Baddr. As you write one of these instructions, replace Baddr with a compatible operand.
Cmem	When an instruction contains Cmem, that instruction can access a single word (16 bits) of data from data memory. As you write the instruction, replace Cmem with a compatible operand.
HI(Cmem)/ LO(Cmem)	When an instruction contains HI(Cmem)/LO(Cmem), that instruction can access a long word (32 bits) of data from data memory. As you write the instruction, replace Cmem with a compatible operand.
Lmem	When an instruction contains Lmem, that instruction can access a long word (32 bits) of data from data memory or from a memory-mapped registers. As you write the instruction, replace Lmem with a compatible operand.
Smem	When an instruction contains Smem, that instruction can access a single word (16 bits) of data from data memory, from I/O space, or from a memory-mapped register. As you write the instruction, replace Smem with a compatible operand.
Xmem and Ymem	When an instruction contains Xmem and Ymem, that instruction can perform two simultaneous 16-bit accesses to data memory. As you write the instruction, replace Xmem and Ymem with compatible operands.

## 3.2 Absolute Addressing Modes

Table 3–2 lists the absolute addressing modes available.

Table 3–2. Absolute Addressing Modes

Addressing Mode	Description
k16 absolute	This mode uses the 7-bit register called DPH (high part of the extended data page register) and a 16-bit unsigned constant to form a 23-bit data-space address. This mode is used to access a memory location or a memory-mapped register.
k23 absolute	This mode enables you to specify a full address as a 23-bit unsigned constant. This mode is used to access a memory location or a memory-mapped register.
I/O absolute	This mode enables you to specify an I/O address as a 16-bit unsigned constant. This mode is used to access a location in I/O space.

### 3.2.1 k16 Absolute Addressing Mode

The k16 absolute addressing mode uses the operand \*abs16(#k16), where k16 is a 16-bit unsigned constant. DPH (the high part of the extended data page register) and k16 are concatenated to form a 23-bit data-space address. An instruction using this addressing mode encodes the constant as a 2-byte extension to the instruction. Because of the extension, an instruction using this mode cannot be executed in parallel with another instruction.

## 3.2.2 k23 Absolute Addressing Mode

The k23 absolute addressing mode uses the \*(#k23) operand, where k23 is a 23-bit unsigned constant. An instruction using this addressing mode encodes the constant as a 3-byte extension to the instruction (the most-significant bit of this 3-byte extension is discarded). Because of the extension, an instruction using this mode cannot be executed in parallel with another instruction.

Instructions using the operand \*(#k23) to access the memory operand Smem cannot be used in a repeatable instruction. See Table 1–4 for a list of these instructions.

#### 3.2.3 I/O Absolute Addressing Mode

The I/O absolute addressing mode uses the \*port(#k16) operand, where k16 is a 16-bit unsigned constant. An instruction using this addressing mode encodes the constant as a 2-byte extension to the instruction. Because of the extension, an instruction using this mode cannot be executed in parallel with another instruction. The delay() instruction cannot use this mode.

## 3.3 Direct Addressing Modes

Table 3–3 lists the direct addressing modes available.

Table 3-3. Direct Addressing Modes

Addressing Mode	Descriptio	n		
DP direct	page regis	uses the main data page specified by DPH (high part of the extended data er) in conjunction with the data page register (DP). This mode is used to emory location or a memory-mapped register.		
SP direct	pointers) in	this mode uses the main data page specified by SPH (high part of the extended stack ointers) in conjunction with the data stack pointer (SP). This mode is used to access tack values in data memory.		
Register-bit direct	This mode uses an offset to specify a bit address. This mode is used to access one register bit or two adjacent register bits.			
PDP direct	This mode uses the peripheral data page register (PDP) and an offset to specify an I/O address. This mode is used to access a location in I/O space.			
		The DP direct and SP direct addressing modes are mutually exclusive. The mode selected depends on the CPL bit in status register ST1_55:		
	CPL	Addressing Mode Selected		
	0	DP direct addressing mode		
	1	SP direct addressing mode		

The register-bit and PDP direct addressing modes are independent of the CPL bit.

## 3.3.1 DP Direct Addressing Mode

When an instruction uses the DP direct addressing mode, a 23-bit address is formed. The 7 MSBs are taken from DPH that selects one of the 128 main data pages (0 through 127). The 16 LSBs are the sum of two values:

- ☐ The value in the data page register (DP). DP identifies the start address of a 128-word local data page within the main data page. This start address can be any address within the selected main data page.
- A 7-bit offset (Doffset) calculated by the assembler. The calculation depends on whether you are accessing data memory or a memory-mapped register (using the mmap() qualifier).

The concatenation of DPH and DP is called the extended data page register (XDP). You can load DPH and DP individually, or you can use an instruction that loads XDP.

## 3.3.2 SP Direct Addressing Mode

When an instruction uses the SP direct addressing mode, a 23-bit address is formed. The 7 MSBs are taken from SPH. The 16 LSBs are the sum of the SP value and a 7-bit offset that you specify in the instruction. The offset can be a value from 0 to 127. The concatenation of SPH and SP is called the extended data stack pointer (XSP). You can load SPH and SP individually, or you can use an instruction that loads XSP.

On the first main data page, addresses 00 0000h–00 005Fh are reserved for the memory-mapped registers. If any of your data stack is in main data page 0, make sure it uses only addresses 00 0060h–00 FFFFh on that page.

### 3.3.3 Register-Bit Direct Addressing Mode

In the register-bit direct addressing mode, the offset you supply in the operand, @bitoffset, is an offset from the LSB of the register. For example, if bitoffset is 0, you are addressing the LSB of a register. If bitoffset is 3, you are addressing bit 3 of the register.

Only the register bit test/set/clear/complement instructions support this mode. These instructions enable you to access bits in the following registers only: the accumulators (AC0–AC3), the auxiliary registers (AR0–AR7), and the temporary registers (T0–T3).

## 3.3.4 PDP Direct Addressing Mode

When an instruction uses the PDP direct addressing mode, a 16-bit I/O address is formed. The 9 MSBs are taken from the 9-bit peripheral data page register (PDP) that selects one of the 512 peripheral data pages (0 through 511). Each page has 128 words (0 to 127). You select a particular word by specifying a 7-bit offset (Poffset) in the instruction. For example, to access the first word on a page, use an offset of 0.

You must use a readport() or writeport() instruction qualifier to indicate that you are accessing an I/O-space location rather than a data-memory location. You place the readport() or the writeport() instruction qualifier in parallel with the instruction that performs the I/O-space access.

## 3.4 Indirect Addressing Modes

Table 3–4 list the indirect addressing modes available. You may use these modes for linear addressing or circular addressing.

Table 3-4. Indirect Addressing Modes

Addressing Mode	Description
AR indirect	This mode uses one of eight auxiliary registers (AR0–AR7) to point to data. The way the CPU uses the auxiliary register to generate an address depends on whether you are accessing data space (memory or memory-mapped registers), individual register bits, or I/O space.
Dual AR indirect	This mode uses the same address-generation process as the AR indirect addressing mode. This mode is used with instructions that access two or more data-memory locations.
CDP indirect	This mode uses the coefficient data pointer (CDP) to point to data. The way the CPU uses CDP to generate an address depends on whether you are accessing data space (memory or memory-mapped registers), individual register bits, or I/O space.
Coefficient indirect	This mode uses the same address-generation process as the CDP indirect addressing mode. This mode is available to support instructions that can access a coefficient in data memory at the same time they access two other data-memory values using the dual AR indirect addressing mode.

## 3.4.1 AR Indirect Addressing Mode

The AR indirect addressing mode uses an auxiliary register ARn (n = 0, 1, 2, 3, 4, 5, 6, or 7) to point to data. The way the CPU uses ARn to generate an address depends on the access type:

For An Access To	ARn Contains	
Data space (memory or registers)	The 16 least significant bits (LSBs) of a 23-bit address. The 7 most significant bits (MSBs) are supplied by ARnH, which is the high part of extended auxiliary register XARn. For accesses to data space, use an instruction that loads XARn; ARn can be individually loaded, but ARnH cannot be loaded.	
A register bit (or bit pair)	A bit number. Only the register bit test/set/clear/complement instructions support AR indirect accesses to register bits. These instructions enable you to access bits in the following registers only: the accumulators (AC0–AC3), the auxiliary registers (AR0–AR7), and the temporary registers (T0–T3).	
I/O space	A 16-bit I/O address.	

The AR indirect addressing-mode operand available depends on the ARMS bit of status register ST2\_55:

ARMS	DSP Mode or Control Mode
0	DSP mode. The CPU can use the list of DSP mode operands (Table 3–5), which provide efficient execution of DSP-intensive applications.
1	Control mode. The CPU can use the list of control mode operands (Table 3–6), which enable optimized code size for control system applications.

Table 3–5 (page 3-8) introduces the DSP operands available for the AR indirect addressing mode. Table 3–6 (page 3-12) introduces the control mode operands. When using the tables, keep in mind that:

- □ Both pointer modification and address generation are linear or circular according to the pointer configuration in status register ST2\_55. The content of the appropriate 16-bit buffer start address register (BSA01, BSA23, BSA45, or BSA67) is added only if circular addressing is activated for the chosen pointer.
- All additions to and subtractions from the pointers are done modulo 8M. ARnH is updated by the hardware when the pointer modification crosses the main data pages' boundary.

Table 3–5. DSP Mode Operands for the AR Indirect Addressing Mode

Operand	Pointer Modification	Supported Access Types
*ARn	ARn is not modified.	Data-memory (Smem, Lmem)
		Memory-mapped register (Smem, Lmem)
		Register bit (Baddr)
		I/O-space (Smem)
*ARn+	ARn is incremented after the address is generated:	Data-memory (Smem, Lmem)
	If 16-bit/1-bit operation: ARn = ARn + 1 If 32-bit/2-bit operation: ARn = ARn + 2	Memory-mapped register (Smem, Lmem)
		Register bit (Baddr)
		I/O-space (Smem)
*ARn-	ARn is decremented after the address is generated:	Data-memory (Smem, Lmem)
	If 16-bit/1-bit operation: ARn = ARn - 1 If 32-bit/2-bit operation: ARn = ARn - 2	Memory-mapped register (Smem, Lmem)
		Register bit (Baddr)
		I/O-space (Smem)
*+ARn	ARn is incremented before the address is generated: If 16-bit/1-bit operation: ARn = ARn + 1 If 32-bit/2-bit operation: ARn = ARn + 2	Data-memory (Smem, Lmem)
		Memory-mapped register (Smem, Lmem)
		Register bit (Baddr)
		I/O-space (Smem)
*–ARn	ARn is decremented before the address is generated: If 16-bit/1-bit operation: $ARn = ARn - 1$ If 32-bit/2-bit operation: $ARn = ARn - 2$	Data-memory (Smem, Lmem)
		Memory-mapped register (Smem, Lmem)
		Register bit (Baddr)
		I/O-space (Smem)
*(ARn + AR0)	The 16-bit signed constant in AR0 is added to ARn after	Data-memory (Smem, Lmem)
	the address is generated: ARn = ARn + AR0	Memory-mapped register (Smem, Lmem)
	This operand is available when C54CM = 1. This operand is usable when .c54cm_on is active at assembly time.	Register bit (Baddr)
		I/O-space (Smem)

Table 3–5. DSP Mode Operands for the AR Indirect Addressing Mode (Continued)

Operand	Pointer Modification	Supported Access Types
*(ARn + T0)	The 16-bit signed constant in T0 is added to ARn after the address is generated:  ARn = ARn + T0  This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at assembly time.	Data-memory (Smem, Lmem)
		Memory-mapped register (Smem, Lmem)
		Register bit (Baddr)
	to double when too fort on to double at accomply time.	I/O-space (Smem)
*(ARn – AR0)	The 16-bit signed constant in AR0 is subtracted from ARn	Data-memory (Smem, Lmem)
	after the address is generated: ARn = ARn – AR0	Memory-mapped register (Smem, Lmem)
	This operand is available when C54CM = 1. This operand is usable when .c54cm_on is active at assembly time.	Register bit (Baddr)
	is assure when to toni_on is active at assembly time.	I/O-space (Smem)
*(ARn – T0)	The 16-bit signed constant in T0 is subtracted from ARn	Data-memory (Smem, Lmem)
	after the address is generated: ARn = ARn - T0	Memory-mapped register (Smem, Lmem)
	This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at assembly time.	Register bit (Baddr)
		I/O-space (Smem)
*ARn(AR0)	16-bit signed constant in AR0 is used as an offset from that base pointer.  This operand is available when C54CM = 1. This operand is usable when .c54cm_on is active at assembly time.	Data-memory (Smem, Lmem)
		Memory-mapped register (Smem, Lmem)
		Register bit (Baddr)
		I/O-space (Smem)
*ARn(T0)		Data-memory (Smem, Lmem)
	16-bit signed constant in T0 is used as an offset from that base pointer.	Memory-mapped register (Smem, Lmem)
	This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at assembly time.	Register bit (Baddr)
	is usable when .6346m_on is active at assembly time.	I/O-space (Smem)
*ARn(T1)	ARn is not modified. ARn is used as a base pointer. The	Data-memory (Smem, Lmem)
	16-bit signed constant in T1 is used as an offset from that base pointer.	Memory-mapped register (Smem, Lmem)
		Register bit (Baddr)
		I/O-space (Smem)

Table 3–5. DSP Mode Operands for the AR Indirect Addressing Mode (Continued)

Operand	Pointer Modification	Supported Access Types
*(ARn + T1)	The 16-bit signed constant in T1 is added to ARn after the address is generated: ARn = ARn + T1	Data-memory (Smem, Lmem)  Memory-mapped register
		(Smem, Lmem) Register bit (Baddr) I/O-space (Smem)
*(ARn – T1)	The 16-bit signed constant in T1 is subtracted from ARn after the address is generated:  ARn = ARn - T1	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register bit (Baddr) I/O-space (Smem)
*(ARn + AR0B)	The 16-bit signed constant in AR0 is added to ARn after the address is generated:  ARn = ARn + AR0  (The addition is done with reverse carry propagation)  This operand is available when C54CM = 1. This operand is usable when .c54cm_on is active at assembly time.	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register bit (Baddr) I/O-space (Smem)
	Note: When this bit-reverse operand is used, ARn cannot be used as a circular pointer. If ARn is configured in ST2_55 for circular addressing, the corresponding buffer start address register value (BSAxx) is added to ARn, but ARn is not modified so as to remain inside a circular buffer.	
*(ARn + T0B)	The 16-bit signed constant in T0 is added to ARn after the address is generated:  ARn = ARn + T0  (The addition is done with reverse carry propagation)	Data-memory (Smem, Lmem)  Memory-mapped register (Smem, Lmem)  Register bit (Baddr)
	This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at assembly time.  Note: When this bit-reverse operand is used, ARn cannot be used as a circular pointer. If ARn is configured in ST2_55 for circular addressing, the corresponding buffer	I/O-space (Smem)
	start address register value (BSAxx) is added to ARn, but ARn is not modified so as to remain inside a circular buffer.	

Table 3–5. DSP Mode Operands for the AR Indirect Addressing Mode (Continued)

Operand	Pointer Modification	Supported Access Types
*(ARn – AR0B)	The 16-bit signed constant in AR0 is subtracted from ARn after the address is generated:  ARn = ARn - AR0  (The subtraction is done with reverse carry propagation)	Data-memory (Smem, Lmem)  Memory-mapped register (Smem, Lmem)
	This operand is available when C54CM = 1. This operand is usable when .c54cm_on is active at assembly time.	Register bit (Baddr) I/O-space (Smem)
	Note: When this bit-reverse operand is used, ARn cannot be used as a circular pointer. If ARn is configured in ST2_55 for circular addressing, the corresponding buffer start address register value (BSAxx) is added to ARn, but ARn is not modified so as to remain inside a circular buffer.	
*(ARn – T0B)	The 16-bit signed constant in T0 is subtracted from ARn after the address is generated:  ARn = ARn - T0  (The subtraction is done with reverse carry propagation)	Data-memory (Smem, Lmem)  Memory-mapped register (Smem, Lmem)
	This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at assembly time.	Register bit (Baddr) I/O-space (Smem)
	Note: When this bit-reverse operand is used, ARn cannot be used as a circular pointer. If ARn is configured in ST2_55 for circular addressing, the corresponding buffer start address register value (BSAxx) is added to ARn, but ARn is not modified so as to remain inside a circular buffer.	
*ARn(#K16)	ARn is not modified. ARn is used as a base pointer. The 16-bit signed constant (K16) is used as an offset from that base pointer.	Data-memory (Smem, Lmem)
		Memory-mapped register (Smem, Lmem)
	Note: When an instruction uses this operand, the constant is encoded in a 2-byte extension to the instruction. Because of the extension, an instruction using this operand cannot be executed in parallel with another instruction.	Register bit (Baddr)
*+ARn(#K16)	The 16-bit signed constant (K16) is added to ARn before	Data-memory (Smem, Lmem)
	the address is generated: ARn = ARn + K16	Memory-mapped register (Smem, Lmem)
	Note: When an instruction uses this operand, the constant is encoded in a 2-byte extension to the instruction. Because of the extension, an instruction using this operand cannot be executed in parallel with another instruction.	Register bit (Baddr)

Table 3-6. Control Mode Operands for the AR Indirect Addressing Mode

Operand	Pointer Modification	Supported Access Types
*ARn	ARn is not modified.	Data-memory (Smem, Lmem)
		Memory-mapped register (Smem, Lmem)
		Register bit (Baddr)
		I/O-space (Smem)
*ARn+	ARn is incremented after the address is generated:	Data-memory (Smem, Lmem)
	If 16-bit/1-bit operation: ARn = ARn + 1 If 32-bit/2-bit operation: ARn = ARn + 2	Memory-mapped register (Smem, Lmem)
		Register bit (Baddr)
		I/O-space (Smem)
*ARn-	ARn is decremented after the address is generated:	Data-memory (Smem, Lmem)
	If 16-bit/1-bit operation: ARn = ARn – 1 If 32-bit/2-bit operation: ARn = ARn – 2	Memory-mapped register Smem, Lmem)
		Register bit (Baddr)
		I/O-space (Smem)
*(ARn + AR0)	The 16-bit signed constant in AR0 is added to ARn after the address is generated: ARn = ARn + AR0 This operand is available when C54CM = 1. This operand is usable when .c54cm_on is active at assembly time.	Data-memory (Smem, Lmem)
		Memory-mapped register (Smem, Lmem)
		Register bit (Baddr)
		I/O-space (Smem)
*(ARn + T0)	The 16-bit signed constant in T0 is added to ARn after the address is generated:  ARn = ARn + T0	Data-memory (Smem, Lmem)
		Memory-mapped register (Smem, Lmem)
	This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at	Register bit (Baddr)
	assembly time.	I/O-space (Smem)
*(ARn – AR0)	The 16-bit signed constant in AR0 is subtracted from	Data-memory (Smem, Lmem)
	ARn after the address is generated: ARn = ARn – AR0	Memory-mapped register (Smem, Lmem)
	This operand is available when C54CM = 1. This	Register bit (Baddr)
	operand is usable when .c54cm_on is active at assembly time.	I/O-space (Smem)

Table 3–6. Control Mode Operands for the AR Indirect Addressing Mode (Continued)

Operand	Pointer Modification	Supported Access Types
*(ARn – T0)	The 16-bit signed constant in T0 is subtracted from ARn after the address is generated: $ARn = ARn - T0$	Data-memory (Smem, Lmem)
		Memory-mapped register (Smem, Lmem)
	This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at	Register bit (Baddr)
	assembly time.	I/O-space (Smem)
*ARn(AR0)	ARn is not modified. ARn is used as a base pointer. The	Data-memory (Smem, Lmem)
	16-bit signed constant in AR0 is used as an offset from that base pointer.	Memory-mapped register (Smem, Lmem)
	This operand is available when C54CM = 1. This operand is usable when .c54cm_on is active at assembly time.	Register bit (Baddr)
		I/O-space (Smem)
*ARn(T0)	ARn is not modified. ARn is used as a base pointer. The 16-bit signed constant in T0 is used as an offset from that base pointer.  This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at assembly time.	Data-memory (Smem, Lmem)
		Memory-mapped register (Smem, Lmem)
		Register bit (Baddr)
		I/O-space (Smem)
*ARn(#K16)	ARn is not modified. ARn is used as a base pointer. The 16-bit signed constant (K16) is used as an offset from that base pointer.	Data-memory (Smem, Lmem)
		Memory-mapped register (Smem, Lmem)
	Note: When an instruction uses this operand, the constant is encoded in a 2-byte extension to the instruction. Because of the extension, an instruction using this operand cannot be executed in parallel with another instruction.	Register bit (Baddr)

Table 3-6. Control Mode Operands for the AR Indirect Addressing Mode (Continued)

Operand	Pointer Modification	Supported Access Types
*+ARn(#K16)	The 16-bit signed constant (K16) is added to ARn before the address is generated: ARn = ARn + K16  Note: When an instruction uses this operand, the constant is encoded in a 2-byte extension to the instruction. Because of the extension, an instruction using this operand cannot be executed in parallel with another instruction.	Data-memory (Smem, Lmem)
		Memory-mapped register (Smem, Lmem)
		Register bit (Baddr)
*ARn(short(#k3))	ARn is not modified. ARn is used as a base pointer. The 3-bit unsigned constant (k3) is used as an offset from that base pointer. k3 is in the range 1 to 7.	Data-memory (Smem, Lmem)
		Memory-mapped register (Smem, Lmem)
		Register bit (Baddr)
		I/O-space (Smem)

## 3.4.2 Dual AR Indirect Addressing Mode

The dual AR indirect addressing mode enables you to make two data-memory accesses through the eight auxiliary registers, AR0–AR7. As with single AR indirect accesses to data space, the CPU uses an extended auxiliary register to create each 23-bit address. You can use linear addressing or circular addressing for each of the two accesses.

You may use the dual AR indirect addressing mode for:

■ Executing an instruction that makes two 16-bit data-memory accesses. In this case, the two data-memory operands are designated in the instruction syntax as Xmem and Ymem. For example:

```
ACx = (Xmem << #16) + (Ymem << #16)
```

■ Executing two instructions in parallel. In this case, both instructions must each access a single memory value, designated in the instruction syntaxes as Smem or Lmem. For example:

```
dst = Smem
|| dst = src & Smem
```

The operand of the first instruction is treated as an Xmem operand, and the operand of the second instruction is treated as a Ymem operand.

The available dual AR indirect operands are a subset of the AR indirect operands. The ARMS status bit does not affect the set of dual AR indirect operands available.

#### Note:

The assembler rejects code in which dual operands use the same auxiliary register with two different auxiliary register modifications. You can use the same ARn for both operands, if one of the operands is \*ARn or \*ARn(T0); neither modifies ARn.

Table 3–7 (page 3-15) introduces the operands available for the dual AR indirect addressing mode. Note that:

- □ Both pointer modification and address generation are linear or circular according to the pointer configuration in status register ST2\_55. The content of the appropriate 16-bit buffer start address register (BSA01, BSA23, BSA45, or BSA67) is added only if circular addressing is activated for the chosen pointer.
- All additions to and subtractions from the pointers are done modulo 8M. ARnH is updated by the hardware when the pointer modification crosses the main data pages' boundary.

Table 3-7. Dual AR Indirect Operands

Operand	Pointer Modification	Supported Access Types
*ARn	ARn is not modified.	Data-memory (Smem, Lmem, Xmem, Ymem)
*ARn+	ARn is incremented after the address is generated:  If 16-bit operation: ARn = ARn + 1  If 32-bit operation: ARn = ARn + 2	Data-memory (Smem, Lmem, Xmem, Ymem)
*ARn–	ARn is decremented after the address is generated: If 16-bit operation: $ARn = ARn - 1$ If 32-bit operation: $ARn = ARn - 2$	Data-memory (Smem, Lmem, Xmem, Ymem)
*(ARn + AR0)	The 16-bit signed constant in AR0 is added to ARn after the address is generated: ARn = ARn + AR0	Data-memory (Smem, Lmem, Xmem, Ymem)
	This operand is available when C54CM = 1. This operand is usable when .c54cm_on is active at assembly time.	
*(ARn + T0)	The 16-bit signed constant in T0 is added to ARn after the address is generated: $ARn = ARn + T0$	Data-memory (Smem, Lmem, Xmem, Ymem)
	This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at assembly time.	

Table 3–7. Dual AR Indirect Operands (Continued)

Operand	Pointer Modification	Supported Access Types
*(ARn – AR0)	The 16-bit signed constant in AR0 is subtracted from ARn after the address is generated: $ARn = ARn - AR0$	Data-memory (Smem, Lmem, Xmem, Ymem)
	This operand is available when C54CM = 1. This operand is usable when .c54cm_on is active at assembly time.	
*(ARn – T0)	The 16-bit signed constant in T0 is subtracted from ARn after the address is generated: $ARn = ARn - T0$	Data-memory (Smem, Lmem, Xmem, Ymem)
	This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at assembly time.	
*ARn(AR0)	ARn is not modified. ARn is used as a base pointer. The 16-bit signed constant in AR0 is used as an offset from that base pointer.	Data-memory (Smem, Lmem, Xmem, Ymem)
	This operand is available when C54CM = 1. This operand is usable when .c54cm_on is active at assembly time.	
*ARn(T0)	ARn is not modified. ARn is used as a base pointer. The 16-bit signed constant in T0 is used as an offset from that base pointer.	Data-memory (Smem, Lmem, Xmem, Ymem)
	This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at assembly time.	
*(ARn + T1)	The 16-bit signed constant in T1 is added to ARn after the address is generated: $ARn = ARn + T1$	Data-memory (Smem, Lmem, Xmem, Ymem)
*(ARn – T1)	The 16-bit signed constant in T1 is subtracted from ARn after the address is generated: $ARn = ARn - T1$	Data-memory (Smem, Lmem, Xmem, Ymem)

## 3.4.3 CDP Indirect Addressing Mode

The CDP indirect addressing mode uses the coefficient data pointer (CDP) to point to data. The way the CPU uses CDP to generate an address depends on the access type:

For An Access To	CDP Contains
Data space (memory or registers)	The 16 least significant bits (LSBs) of a 23-bit address. The 7 most significant bits (MSBs) are supplied by CDPH, the high part of the extended coefficient data pointer (XCDP).
A register bit (or bit pair)	A bit number. Only the register bit test/set/clear/complement instructions support CDP indirect accesses to register bits. These instructions enable you to access bits in the following registers only: the accumulators (AC0–AC3), the auxiliary registers (AR0–AR7), and the temporary registers (T0–T3).
I/O space	A 16-bit I/O address.

Table 3–8 (page 3-17) introduces the operands available for the CDP indirect addressing mode. Note that:

- Both pointer modification and address generation are linear or circular according to the pointer configuration in status register ST2\_55. The content of the 16-bit buffer start address register BSAC is added only if circular addressing is activated for CDP.
- All additions to and subtractions from the pointers are done modulo 8M. CDPH is updated by the hardware when the pointer modification crosses the main data pages' boundary.

Table 3–8. CDP Indirect Operands

Operand	Pointer Modification	Supported Access Types
*CDP	CDP is not modified.	Data-memory (Smem, Lmem)
		Memory-mapped register (Smem, Lmem)
		Register-bit (Baddr)
		I/O-space (Smem)
*CDP+	CDP is incremented after the address is generated:	Data-memory (Smem, Lmem)
	If 16-bit/1-bit operation: CDP = CDP + 1 If 32-bit/2-bit operation: CDP = CDP + 2	Memory-mapped register (Smem, Lmem)
		Register-bit (Baddr)
		I/O-space (Smem)

Table 3–8. CDP Indirect Operands (Continued)

Operand	Pointer Modification	Supported Access Types
*CDP-	CDP is decremented after the address is generated:  If 16-bit/1-bit operation: CDP = CDP - 1  If 32-bit/2-bit operation: CDP = CDP - 2	Data-memory (Smem, Lmem)
		Memory-mapped register (Smem, Lmem)
		Register-bit (Baddr)
		I/O-space (Smem)
*CDP(#K16)	CDP is not modified. CDP is used as a base pointer. The	Data-memory (Smem, Lmem)
	16-bit signed constant (K16) is used as an offset from that base pointer.	Memory-mapped register (Smem, Lmem)
	Note: When an instruction uses this operand, the constant is encoded in a 2-byte extension to the instruction. Because of the extension, an instruction using this operand cannot be executed in parallel with another instruction.	Register-bit (Baddr)
*+CDP(#K16)	The 16-bit signed constant (K16) is added to CDP before the address is generated: CDP = CDP + K16	Data-memory (Smem, Lmem)
		Memory-mapped register (Smem, Lmem)
	Note: When an instruction uses this operand, the constant is encoded in a 2-byte extension to the instruction. Because of the extension, an instruction using this operand cannot be executed in parallel with another instruction.	Register-bit (Baddr)

## 3.4.4 Coefficient Indirect Addressing Mode

The coefficient indirect addressing mode uses the same address-generation process as the CDP indirect addressing mode for data-space accesses. The coefficient indirect addressing mode is supported by select memory-to-memory move and memory initialization instructions and by the following arithmetical instructions:

Dual multiply (accumulate/subtract)
Finite impulse response filter
Multiply
Multiply and accumulate
Multiply and subtract

Instructions using the coefficient indirect addressing mode to access data are mainly instructions performing operations with three memory operands per cycle. Two of these operands (Xmem and Ymem) are accessed with the dual AR indirect addressing mode. The third operand (Cmem) is accessed with the coefficient indirect addressing mode. The Cmem operand is carried on the BB bus.

Keep the following facts about the BB bus in mind as you use the coefficient indirect addressing mode:

The BB bus is not connected to external memory. If a Cmem operand is
accessed through the BB bus, the operand must be in internal memory.

Although the following instructions access Cmem operands, they do not
use the BB bus to fetch the 16-bit or 32-bit Cmem operand.

Instruction Syntax	Description of Cmem Access	Bus Used to Access Cmem
Smem = Cmem	16-bit read from Cmem	DB
Cmem = Smem	16-bit write to Cmem	EB
Lmem = dbl(Cmem)	32-bit read from Cmem	CB for most significant word (MSW) DB for least significant word (LSW)
dbl(Cmem) = Lmem	32-bit write to Cmem	FB for MSW EB for LSW

Consider the following instruction syntax. In one cycle, two multiplications can be performed in parallel. One memory operand (Cmem) is common to both multiplications, while dual AR indirect operands (Xmem and Ymem) are used for the other values in the multiplication.

```
ACx = Xmem * Cmem,
ACy = Ymem * Cmem
```

To access three memory values (as in the above example) in a single cycle, the value referenced by Cmem must be located in a memory bank different from the one containing the Xmem and Ymem values.

Table 3–9 introduces the operands available for the coefficient indirect addressing mode. Note that:

- □ Both pointer modification and address generation are linear or circular according to the pointer configuration in status register ST2\_55. The content of the 16-bit buffer start address register BSAC is added only if circular addressing is activated for CDP.
- All additions to and subtractions from the pointers are done modulo 8M. CDPH is updated by the hardware when the pointer modification crosses the main data pages' boundary.

Table 3–9. Coefficient Indirect Operands

Operand	Pointer Modification	Supported Access Type
*CDP	CDP is not modified.1	Data-memory
*CDP+	CDP is incremented after the address is generated:  If 16-bit operation: CDP = CDP + 1  If 32-bit operation: CDP = CDP + 2	Data-memory
*CDP-	CDP is decremented after the address is generated: If 16-bit operation: CDP = CDP $-$ 1 If 32-bit operation: CDP = CDP $-$ 2	Data-memory
*(CDP + AR0)	The 16-bit signed constant in AR0 is added to CDP after the address is generated: CDP = CDP + AR0	Data-memory
	This operand is available when C54CM = 1. This operand is usable when .c54cm_on is active at assembly time.	
*(CDP + T0)	The 16-bit signed constant in T0 is added to CDP after the address is generated: CDP = CDP + T0	Data-memory
	This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at assembly time.	

## 3.5 Circular Addressing

Circular addressing can be used with any of the indirect addressing modes. Each of the eight auxiliary registers (AR0–AR7) and the coefficient data pointer (CDP) can be independently configured to be linearly or circularly modified as they act as pointers to data or to register bits, see Table 3–10. This configuration is done with a bit (ARnLC) in status register ST2\_55. To choose circular modification, set the bit.

Table 3-10. Circular Addressing Pointers

Pointer	Linear/Circular Configuration Bit	Supplier of Main Data Page	Buffer Start Address Register	Buffer Size Register
AR0	ST2_55(0) = AR0LC	AR0H	BSA01	BK03
AR1	ST2_55(1) = AR1LC	AR1H	BSA01	BK03
AR2	ST2_55(2) = AR2LC	AR2H	BSA23	BK03
AR3	ST2_55(3) = AR3LC	AR3H	BSA23	BK03
AR4	ST2_55(4) = AR4LC	AR4H	BSA45	BK47
AR5	ST2_55(5) = AR5LC	AR5H	BSA45	BK47
AR6	ST2_55(6) = AR6LC	AR6H	BSA67	BK47
AR7	ST2_55(7) = AR7LC	AR7H	BSA67	BK47
CDP	ST2_55(8) = CDPLC	CDPH	BSAC	BKC

Each auxiliary register ARn has its own linear/circular configuration bit in ST2\_55:

ARnLC	ARn Is Used For
0	Linear addressing
1	Circular addressing

The CDPLC bit in status register ST2\_55 configures the DSP to use CDP for linear addressing or circular addressing:

CDPLC	CDP Is Used For
0	Linear addressing
1	Circular addressing

You can use the circular addressing instruction qualifier, circular(), if you want every pointer used by the instruction to be modified circularly, just add the circular() qualifier in parallel with the instruction. The circular addressing instruction qualifier overrides the linear/circular configuration in ST2\_55.

# **Instruction Set Summary**

This chapter provides a summary of the TMS320C55x<sup>™</sup> DSP algebraic instruction set (Table 4–1). With each instruction, you will find the availability of a parallel enable bit, word count (size), cycle time, what pipeline phase the instruction executes, in what operator unit the instruction executes, how many of each address generation unit is used, and how many of each bus is used.

Table 4–1 does not list all of the resources that may be used by an instruction, it only lists those that may result in a resource conflict, and thus prevent two instructions from being in parallel. If an instruction lists nothing in a particular column, it means that particular resource will never be in conflict for that instruction.

The column heads of Table 4–1 are:

- □ Instruction: In cases where the resource usage of an instruction varies with the kinds of registers, you see the notation <name>-AU for A-unit registers and <name>-DU for D-unit registers. So, dst-AU is a destination that is an A-unit register and src-DU is a source that is a D-unit register. In the few cases where that notation is insufficient, you see the cases listed in the Notes column.
- ☐ E: Whether that instruction has a parallel enable bit
- ☐ S: The size of the instruction in bytes
- ☐ C: Number of cycles required for the instruction
- ☐ Pipe: The pipeline phase in which the instruction executes:

Name	Phase
AD	Address
D	Decode
R	Read
Χ	Execute

Operator: Which operator(s) are used by this instruction. When an instruction uses multiple operators, any other instruction that uses one or more of those same operators may not be placed in parallel.

☐ Address Generation Unit: How many of each address generation unit is used. The address generation units are:

Name	Unit
DA	Data Address Generation Unit
CA	Coefficient Address Generation Unit
SA	Stack Address Generation Unit

Buses: How many of each bus is used. The buses are:

Name	Bus								
DR	Data Read								
CR	Coefficient Read								
DW	Data Write								

ACB Brings D unit registers to A unit and P unit operators

4-2 Instruction Set Summary SWPU068E

Table 4–1. Algebraic Instruction Set Summary

								Addres eration			В	uses		
No.	Instruction	E	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
Abs	solute Distance (page 5-2)													
	abdst(Xmem, Ymem, ACx, ACy)	N	4	1	Х	DU_ALU + DU_MAC1	2			2				
Abs	solute Value (page 5-4)													
	dst-AU =  src-AU	Υ	2	1	Х	AU_ALU								1
	dst-AU =  src-DU	Υ	2	1	Х	AU_ALU							1	
	dst-DU =  src	Υ	2	1	Х	DU_ALU								See Note 1.
Add	dition (page 5-7)	•					•							,
[1]	dst-AU = dst-AU + src-AU	Υ	2	1	Х	AU_ALU								
	dst-AU = dst-AU + src-DU	Υ	2	1	Х	AU_ALU							1	
	dst-DU = dst-DU + src	Υ	2	1	Х	DU_ALU								See Note 1.
[2]	dst-AU = dst-AU + k4	Υ	2	1	Х	AU_ALU								
	dst-DU = dst-DU + k4	Υ	2	1	Х	DU_ALU								
[3]	dst-AU = src-AU + K16	N	4	1	Х	AU_ALU								
	dst-AU = src-DU + K16	N	4	1	Х	AU_ALU							1	
	dst-DU = src + K16	N	4	1	Х	DU_ALU								See Note 1.
[4]	dst-AU = src-AU + Smem	N	3	1	Х	AU_ALU	1			1				
	dst-AU = src-DU + Smem	N	3	1	Х	AU_ALU	1			1			1	
	dst-DU = src + Smem	N	3	1	Χ	DU_ALU	1			1				See Note 1.
[5]	ACy = ACy + (ACx << Tx)	Υ	2	1	Χ	DU_SHIFT				•				
[6]	ACy = ACy + (ACx << #SHIFTW)	Υ	3	1	Х	DU_SHIFT								
[7]	ACy = ACx + (K16 << #16)	N	4	1	Χ	DU_ALU								
[8]	ACy = ACx + (K16 << #SHFT)	N	4	1	Х	DU_SHIFT								
[9]	ACy = ACx + (Smem << Tx)	N	3	1	Х	DU_SHIFT	1			1				
[10]	ACy = ACx + (Smem << #16)	N	3	1	Х	DU_ALU	1			1				
[11]	ACy = ACx + uns(Smem) + CARRY	Ν	3	1	Х	DU_ALU	1			1				1

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

<sup>2)</sup> dst-DU, src-AU or dst-AU, src-DU

Table 4–1. Algebraic Instruction Set Summary (Continued)

								Address neration Unit			В	uses		
No.	Instruction	E	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
[12]	ACy = ACx + uns(Smem)	N	3	1	Х	DU_ALU	1			1				
[13]	ACy = ACx + (uns(Smem) << #SHIFTW)	N	4	1	Χ	DU_SHIFT	1		•	1				
[14]	ACy = ACx + dbl(Lmem)	N	3	1	Х	DU_ALU	1			2				
[15]	ACx = (Xmem << #16) + (Ymem << #16)	N	3	1	Χ	DU_ALU	2			2				
[16]	Smem = Smem + K16	N	4	1	Χ	DU_ALU	1			1		1		
Add	dition with Absolute Value (page 5-27)													
	$ACy = \frac{\text{rnd}(ACy +  ACx )}{ACx}$	Υ	2	1	Х	DU_MAC1			•					
Add	dition with Parallel Store Accumulator Content to Memory (pa	age	5-2	9)										
	ACy = ACx + (Xmem << #16), Ymem = HI(ACy << T2)	N	4	1	X	DU_ALU + DU_SHIFT	2	-		2		2		
Add	dition or Subtraction Conditionally (page 5-31)													
[1]	ACy = adsc(Smem, ACx, TC1)	N	3	1	Х	DU_SHIFT	1			1				
[2]	ACy = adsc(Smem, ACx, TC2)	N	3	1	Х	DU_SHIFT	1		•	1				
Add	dition or Subtraction Conditionally with Shift (page 5-33)													
	ACy = ads2c(Smem, ACx, Tx, TC1, TC2)	N	3	1	Х	DU_SHIFT	1			1				
Add	dition, Subtraction, or Move Accumulator Content Conditiona	ally	(pa	ige 5	5-36)									
	ACy = adsc(Smem, ACx, TC1, TC2)	N	3	1	Х	DU_SHIFT	1			1				
Bitv	wise AND (page 5-38)	•					•			•				
[1]	dst-AU = dst-AU & src-AU	Υ	2	1	Х	AU_ALU								
	dst-AU = dst-AU & src-DU	Υ	2	1	Х	AU_ALU							1	
	dst-DU = dst-DU & src	Υ	2	1	Х	DU_ALU								See Note 1.
[2]	dst-AU = src-AU & k8	Υ	3	1	Х	AU_ALU								
	dst-AU = src-DU & k8	Υ	3	1	Х	AU_ALU							1	
	dst-DU = src & k8	Υ	3	1	Х	DU_ALU								See Note 1.

Instruction Set Summary

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

Table 4–1. Algebraic Instruction Set Summary (Continued)

								ddres eration			В	uses		
No.	Instruction	E	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
[3]	dst-AU = src-AU & k16	N	4	1	Х	AU_ALU							•	
	dst-AU = src-DU & k16	N	4	1	Х	AU_ALU							1	
	dst-DU = src & k16	N	4	1	X	DU_ALU								See Note 1.
[4]	dst-AU = src-AU & Smem	N	3	1	Χ	AU_ALU	1			1				
	dst-AU = src-DU & Smem	N	3	1	Χ	AU_ALU	1			1			1	
	dst-DU = src & Smem	N	3	1	Х	DU_ALU	1			1				See Note 1.
[5]	ACy = ACy & (ACx <<< #SHIFTW)	Υ	3	1	X	DU_SHIFT								
[6]	ACy = ACx & (k16 <<< #16)	N	4	1	X	DU_ALU								
[7]	ACy = ACx & (k16 <<< #SHFT)	N	4	1	Χ	DU_SHIFT								
[8]	Smem = Smem & k16	N	4	1	Х	AU_ALU	1			1		1		
Bitv	vise AND Memory with Immediate Value and Compare to Zer	<b>o</b> (p	pag	e 5-4	17)									
[1]	TC1 = Smem & k16	N	4	1	Х	AU_ALU	1			1			•	
[2]	TC2 = Smem & k16	N	4	1	Х	AU_ALU	1			1				
Bitv	vise OR (page 5-48)	•												•
[1]	dst-AU = dst-AU   src-AU	Υ	2	1	Х	AU_ALU							•	
	dst-AU = dst-AU   src-DU	Υ	2	1	Х	AU_ALU							1	
	dst-DU = dst-DU   src	Υ	2	1	Х	DU_ALU								See Note 1.
[2]	dst-AU = src-AU   k8	Υ	3	1	Х	AU_ALU								
	dst-AU = src-DU   k8	Υ	3	1	Х	AU_ALU							1	
	dst-DU = src   k8	Υ	3	1	Х	DU_ALU								See Note 1.
[3]	dst-AU = src-AU   k16	N	4	1	Х	AU_ALU								
	dst-AU = src-DU   k16	N	4	1	Х	AU_ALU							1	
	dst-DU = src   k16	N	4	1	Х	DU_ALU								See Note 1.
[4]	dst-AU = src-AU   Smem	N	3	1	Х	AU_ALU	1			1			•	
	dst-AU = src-DU   Smem	N	3	1	Х	AU_ALU	1			1			1	
	dst-DU = src   Smem	N	3	1	Х	DU_ALU	1			1				See Note 1.

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

4-6

Table 4–1. Algebraic Instruction Set Summary (Continued)

								Address Generation Unit		Buses				-
No.	Instruction	E	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
[5]	ACy = ACy   (ACx <<< #SHIFTW)	Υ	3	1	Х	DU_SHIFT								
[6]	ACy = ACx   (k16 <<< #16)	N	4	1	X	DU_ALU								
[7]	ACy = ACx   (k16 <<< #SHFT)	N	4	1	X	DU_SHIFT								
[8]	Smem = Smem   k16	N	4	1	X	AU_ALU	1			1		1		
Bitv	vise Exclusive OR (XOR) (page 5-57)													
[1]	dst-AU = dst-AU ^ src-AU	Υ	2	1	Х	AU_ALU								
	dst-AU = dst-AU ^ src-DU	Υ	2	1	Х	AU_ALU							1	
	dst-DU = dst-DU ^ src	Υ	2	1	Х	DU_ALU								See Note 1.
[2]	dst-AU = src-AU ^ k8	Υ	3	1	Х	AU_ALU								
	dst-AU = src-DU ^ k8	Υ	3	1	Х	AU_ALU							1	
	dst-DU = src ^ k8	Υ	3	1	Х	DU_ALU								See Note 1.
[3]	dst-AU = src-AU ^ k16	N	4	1	X	AU_ALU								
	dst-AU = src-DU ^ k16	N	4	1	X	AU_ALU							1	
	dst-DU = src ^ k16	N	4	1	Х	DU_ALU			•					See Note 1.
[4]	dst-AU = src-AU ^ Smem	N	3	1	Χ	AU_ALU	1			1				
	dst-AU = src-DU ^ Smem	N	3	1	Χ	AU_ALU	1			1			1	
	dst-DU = src ^ Smem	N	3	1	Χ	DU_ALU	1			1				See Note 1.
[5]	$ACy = ACy \land (ACx \iff SHIFTW)$	Υ	3	1	Χ	DU_SHIFT								
[6]	$ACy = ACx ^ (k16 <<< #16)$	N	4	1	Χ	DU_ALU								
[7]	$ACy = ACx ^ (k16 <<< \#SHFT)$	N	4	1	Χ	DU_SHIFT								
[8]	Smem = Smem ^ k16	N	4	1	Χ	AU_ALU	1			1		1		
Bra	nch Conditionally (page 5-66)													
[1]	if (cond) goto l4	N	2	6/5†	R	PU_UNIT								
[2]	if (cond) goto L8	Υ	3	6/5†	R	PU_UNIT								
[3]	if (cond) goto L16	Ζ	4	6/5†	R	PU_UNIT							<u> </u>	

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

Table 4–1. Algebraic Instruction Set Summary (Continued)

							Address Generation Unit				В	uses		
No.	Instruction	E	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
[4]	if (cond) goto P24	N	5	5/5†	R	PU_UNIT								
† <sub>x/y</sub>	cycles: x cycles = condition true, y cycles = condition false	•					•			•			•	
Bra	nch Unconditionally (page 5-70)													
[1]	goto ACx	N	2	10	Х	PU_UNIT							1	
[2]	goto L7	Υ	2	6†	AD	PU_UNIT								
[3]	goto L16	Υ	3	6†	AD	PU_UNIT								
[4]	goto P24	N	4	5	D	PU_UNIT							•	
† The	ese instructions execute in 3 cycles if the addressed instruction is in the instruction buffer unit.													
Bra	nch on Auxiliary Register Not Zero (page 5-74)													
	if (ARn_mod != #0) goto L16	N	4	6/5†	AD	PU_UNIT	1							
† <sub>x/y</sub>	cycles: x cycles = condition true, y cycles = condition false	•					•			•			•	
Cal	l Conditionally (page 5-77)													
[1]	if (cond) call L16	N	4	6/5†	R	PU_UNIT PU_UNIT	1		1			2		
[2]	if (cond) call P24	N	5	5/5†	R	PU_UNIT	1		1			2		
† <sub>X/y</sub>	cycles: x cycles = condition true, y cycles = condition false	•					•			•			•	
Cal	l Unconditionally (page 5-83)													
[1]	call ACx	N	2	10	Х	PU_UNIT	1		1			2	1	
[2]	call L16	Υ	3	6	AD	PU_UNIT	1		1			2		
[3]	call P24	N	4	5	D	PU_UNIT	1		1			2		
Cir	cular Addressing Qualifier (page 5-87)													
	circular()	N	1	1	AD									
Cle	ar Accumulator, Auxiliary, or Temporary Register Bit (page 5-	88	)				•			•			•	
	bit(src-AU, Baddr) = #0	N	3	1	Х	AU_ALU	1							
	bit(src-DU, Baddr) = #0	N	3	1	Х	DU_BIT	1							

- Notes: 1) dst-DU, src-AU or dst-DU, src-DU
  - 2) dst-DU, src-AU or dst-AU, src-DU

Table 4–1. Algebraic Instruction Set Summary (Continued)

							Addres eration			В	uses		
No. Instruction	E	s	С	Pipe	Operator	DA	CA	SA	DR	CR		ACB	Notes
Clear Memory Bit (page 5-89)													
bit(Smem, src) = #0	N	3	1	Х	AU_ALU	1			1		1	•	
Clear Status Register Bit (page 5-90)						•			•				•
[1] bit(ST0, k4) = #0	Υ	2	1	Х	AU_ALU						•		
[2] $bit(ST1, k4) = #0$	Υ	2	1	Х	AU_ALU								
[3] $bit(ST2, k4) = #0$	Υ	2	1	Х	AU_ALU								
[4] bit(ST3, k4) = #0	Υ	2	1†	Х	AU_ALU								
<sup>†</sup> When this instruction is decoded to modify status bit CAFRZ (15), CAEN (14), or CACLR (13), the G	CPU p	oipelin	e is flu	ushed an	d the instruction i	s exec	uted in	5 cycl	es rega	ırdless	of the i	nstruction	n context.
Compare Accumulator, Auxiliary, or Temporary Register Conte	nt (p	page	e 5-9	93)									
[1] TC1 = uns(src-AU RELOP dst-AU)	Υ	3	1	Х	AU_ALU								
TC1 = uns(src RELOP dst)	Υ	3	1	Х	AU_ALU							1	See Note 2.
TC1 = uns(src-DU RELOP dst-DU)	Υ	3	1	Х	DU_ALU								
[2] TC2 = uns(src-AU RELOP dst-AU)	Υ	3	1	Χ	AU_ALU								
TC2 = uns(src RELOP dst)	Υ	3	1	Χ	AU_ALU							1	See Note 2.
TC2 = uns(src-DU RELOP dst-DU)	Υ	3	1	Χ	DU_ALU								
Compare Accumulator, Auxiliary, or Temporary Register Conte	nt w	vith	ANI	<b>)</b> (pag	e 5-95)								
[1] TCx = TCy & uns(src-AU RELOP dst-AU)	Υ	3	1	Х	AU_ALU								
TCx = TCy & uns(src RELOP dst)	Υ	3	1	Х	AU_ALU							1	See Note 2.
TCx = TCy & uns(src-DU RELOP dst-DU)	Υ	3	1	Х	DU_ALU								
[2] TCx = !TCy & uns(src-AU RELOP dst-AU)	Υ	3	1	Х	AU_ALU								
TCx = !TCy & uns(src RELOP dst)	Υ	3	1	Х	AU_ALU							1	See Note 2.

Y 3 1

DU\_ALU

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

TCx = !TCy & uns(src-DU RELOP dst-DU)

Table 4–1. Algebraic Instruction Set Summary (Continued)

									ddres eration			В	uses		
No.	Instruction	E	s	c	;	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
Со	mpare Accumulator, Auxiliary, or Temporary Register Conten	t w	vith	O	<b>R</b> (p	age	5-100)								
[1]	TCx = TCy   uns(src-AU RELOP dst-AU)	Υ	3	1		Χ	AU_ALU								
	TCx = TCy   uns(src RELOP dst)	Υ	3	1		Χ	AU_ALU							1	See Note 2.
	TCx = TCy   uns(src-DU RELOP dst-DU)	Υ	3	1		Χ	DU_ALU								
[2]	TCx = !TCy   uns(src-AU RELOP dst-AU)	Υ	3	1		Χ	AU_ALU								
	TCx = !TCy   uns(src RELOP dst)	Υ	3	1		Χ	AU_ALU							1	See Note 2.
	TCx = !TCy   uns(src-DU RELOP dst-DU)	Υ	3	1		Χ	DU_ALU								
Со	mpare Accumulator, Auxiliary, or Temporary Register Conten	t N	lax	im	um	(pag	e 5-105)	•			•				•
	dst-AU = max(src-AU, dst-AU)	Υ	2	1		Х	AU_ALU								
	dst-AU = max(src-DU, dst-AU)	Υ	2	1		Χ	AU_ALU							1	
	dst-DU = max(src, dst-DU)	Υ	2	1		Χ	DU_ALU			•					See Note 1.
Со	mpare Accumulator, Auxiliary, or Temporary Register Conten	t N	/lini	mι	ım	(pag	e 5-108)	•			•				•
	dst-AU = min(src-AU, dst-AU)	Υ	2	1		Χ	AU_ALU								
	dst-AU = min(src-DU, dst-AU)	Υ	2	1		Χ	AU_ALU							1	
	dst-DU = min(src, dst-DU)	Υ	2	1		Х	DU_ALU								See Note 1.
Со	mpare and Branch (page 5-111)							ı							
	compare (uns(src-AU RELOP K8)) goto L8	N	4	7/6	3†	Х	AU_ALU + PU_UNIT			•			•		
	compare (uns(src-DU RELOP K8)) goto L8	N	4	7/6	5†	Х	DU_ALU + PU_UNIT								
† x/y	cycles: x cycles = condition true, y cycles = condition false							•							•
Со	mpare and Select Accumulator Content Maximum (page 5-114	4)													
[1]	max_diff(ACx, ACy, ACz, ACw)	Υ	3	1		Х	DU_ALU								
[2]	max_diff_dbl(ACx, ACy, ACz, ACw, TRNx)	Υ	3	1		Χ	DU_ALU								

Table 4–1. Algebraic Instruction Set Summary (Continued)

								Addres eration			В	uses		
No.	Instruction	Е	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
Coı	mpare and Select Accumulator Content Minimum (page 5-120	)												
[1]	min_diff(ACx, ACy, ACz, ACw)	Υ	3	1	X	DU_ALU								
[2]	min_diff_dbl(ACx, ACy, ACz, ACw, TRNx)	Υ	3	1	Х	DU_ALU			•				-	
Coı	mpare Memory with Immediate Value (page 5-126)													
[1]	TC1 = (Smem == K16)	N	4	1	X	AU_ALU	1			1			•	
[2]	TC2 = (Smem == K16)	Ν	4	1	Х	AU_ALU	1			1				
Coı	mplement Accumulator, Auxiliary, or Temporary Register Bit	(ра	ige :	5-12	3)									
	cbit(src-AU, Baddr)	N	3	1	X	AU_ALU	1							
	cbit(src-DU, Baddr)	Ν	3	1	X	DU_BIT	1		•					
Coı	mplement Accumulator, Auxiliary, or Temporary Register Cor	ite	nt (	oage	5-129	9)								
	dst-AU = ~src-AU	Υ	2	1	Х	AU_ALU							•	
	dst-AU = ~src-DU	Υ	2	1	X	AU_ALU			•				1	
	dst-DU = ~src	Υ	2	1	Х	DU_ALU								See Note
Coı	mplement Memory Bit (page 5-130)													
	cbit(Smem, src)	Ν	3	1	X	AU_ALU	1			1		1		
Coı	npute Exponent of Accumulator Content (page 5-131)						•			•				
	Tx = exp(ACx)	Υ	3	1	Х	DU_BIT + AU_ALU		٠	•				1	
Coı	mpute Mantissa and Exponent of Accumulator Content (page	5-	132	)										
	ACy = mant(ACx), Tx = exp(ACx)	Υ	3	1	Х	DU_BIT + DU_SHIFT + AU_ALU		٠	•			٠	1	
Coi	unt Accumulator Bits (page 5-134)						•			•				•
[1]	Tx = count(ACx, ACy, TC1)	Υ	3	1	Х	DU_BIT + AU_ALU							1	
[2]	Tx = count(ACx, ACy, TC2)	Υ	3	1	X	DU_BIT + AU_ALU							1	

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

Table 4–1. Algebraic Instruction Set Summary (Continued)

								Addres eration			В	uses		
No.	Instruction	E	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
Dua	al 16-Bit Additions (page 5-135)													
[1]	HI(ACy) = HI(Lmem) + HI(ACx), LO(ACy) = LO(Lmem) + LO(ACx)	N	3	1	Х	DU_ALU	1	•	•	2		•		
[2]	HI(ACx) = HI(Lmem) + Tx, LO(ACx) = LO(Lmem) + Tx	N	3	1	Х	DU_ALU	1			2				
Dua	al 16-Bit Addition and Subtraction (page 5-140)													
[1]	HI(ACx) = Smem + Tx, LO(ACx) = Smem - Tx	N	3	1	Х	DU_ALU	1			1				
[2]	HI(ACx) = HI(Lmem) + Tx, LO(ACx) = LO(Lmem) - Tx	N	3	1	Х	DU_ALU	1	•	•	2		•		
Dua	al 16-Bit Subtractions (page 5-145)									•			•	
[1]	HI(ACy) = HI(ACx) - HI(Lmem), LO(ACy) = LO(ACx) - LO(Lmem)	N	3	1	Х	DU_ALU	1			2				
[2]	HI(ACy) = HI(Lmem) - HI(ACx), LO(ACy) = LO(Lmem) - LO(ACx)	N	3	1	Х	DU_ALU	1			2				
[3]	HI(ACx) = Tx - HI(Lmem), LO(ACx) = Tx - LO(Lmem)	N	3	1	Х	DU_ALU	1			2			•	
[4]	HI(ACx) = HI(Lmem) - Tx, LO(ACx) = LO(Lmem) - Tx	N	3	1	Х	DU_ALU	1			2				
Dua	al 16-Bit Subtraction and Addition (page 5-154)													
[1]	HI(ACx) = Smem - Tx, LO(ACx) = Smem + Tx	N	3	1	Х	DU_ALU	1			1				
[2]	$\begin{aligned} & \text{HI}(\text{ACx}) = \text{HI}(\text{Lmem}) - \text{Tx}, \\ & \text{LO}(\text{ACx}) = \text{LO}(\text{Lmem}) + \text{Tx} \end{aligned}$	N	3	1	Х	DU_ALU	1			2				
Exe	ecute Conditionally (page 5-159)	•					•			•			•	
[1]	if (cond) execute(AD_Unit)	N	2	1	AD	PU_UNIT								
[2]	if (cond) execute(D_Unit)	N	2	1	Х	PU_UNIT								

Table 4–1. Algebraic Instruction Set Summary (Continued)

							Addres eration			В	uses		
No. Instruction	E	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
Expand Accumulator Bit Field (page 5-166)													
dst-AU = field_expand(ACx, k16)	N	4	1	Х	DU_BIT + AU_ALU							1	
dst-DU = field_expand(ACx, k16)	N	4	1	X	DU_BIT								
Extract Accumulator Bit Field (page 5-167)													
dst-AU = field_extract(ACx, k16)	N	4	1	Х	DU_BIT + AU_ALU		٠					1	
dst-DU = field_extract(ACx, k16)	N	4	1	X	DU_BIT								
Finite Impulse Response Filter, Antisymmetrical (page 5-168)													
firsn(Xmem, Ymem, coef(Cmem), ACx, ACy)	N	4	1	Х	DU_ALU + DU_MAC1	2	1		2	1			
Finite Impulse Response Filter, Symmetrical (page 5-170)													
firs(Xmem, Ymem, coef(Cmem), ACx, ACy)	N	4	1	X	DU_ALU + DU_MAC1	2	1		2	1			
<b>Idle</b> (page 5-172)													
idle	N	4	?	D	PU_UNIT								
Least Mean Square (LMS) (page 5-173)	·					•			•			•	
Ims(Xmem, Ymem, ACx, ACy)	N	4	1	Х	DU_ALU + DU_MAC1	2			2				
Least Mean Square (LMSF) (page 5-175)													
Imsf(Xmem, Ymem, ACx, ACy)	N	4	1	Х	DU_MAC1 + DU_MAC2 + DU_ALU	2			2		1		
Linear Addressing Qualifier (page 5-179)	Ī											ļ	
linear()	N	1	1	AD								.	

Table 4–1. Algebraic Instruction Set Summary (Continued)

								Addres eration			В	uses		
No.	Instruction	Е	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
Loa	d Accumulator from Memory (page 5-180)													
[1]	ACx = md(Smem << Tx)	N	3	1	Х	DU_SHIFT	1			1		•		
[2]	ACx = low_byte(Smem) << #SHIFTW	N	3	1	X	DU_SHIFT	1			1				
[3]	ACx = high_byte(Smem) << #SHIFTW	N	3	1	Х	DU_SHIFT	1		•	1		•		
[4]	ACx = Smem << #16	N	2	1	Х	DU_LOAD	1		•	1		•		
[5]	ACx = uns(Smem)	N	3	1	Х	DU_LOAD	1			1				
[6]	ACx = uns(Smem) << #SHIFTW	N	4	1	Х	DU_SHIFT	1			1				
[7]	$ACx = \frac{M40(dbl(Lmem))}{}$	N	3	1	Х	DU_LOAD	1			2				
[8]	LO(ACx) = Xmem, HI(ACx) = Ymem	N	3	1	Χ	DU_LOAD	2			2				
Loa	d Accumulator Pair from Memory (page 5-191)													!
[1]	pair(HI(ACx)) = Lmem	N	3	1	X	DU_LOAD	1			2				
[2]	pair(LO(ACx)) = Lmem	N	3	1	Х	DU_LOAD	1			2				
Loa	d Accumulator with Immediate Value (page 5-196)													
[1]	ACx = K16 << #16	N	4	1	Х	DU_LOAD						•		
[2]	ACx = K16 << #SHFT	N	4	1	Х	DU_SHIFT								
Loa	d Accumulator from Memory with Parallel Store Accumulato	r C	ont	ent	to Me	mory (pag	e 5-1	89)						
	ACy = Xmem << #16, Ymem = HI(ACx << T2)	N	4	1	Х	DU_LOAD + DU_SHIFT	2		•	2	٠	2		
Loa	d Accumulator, Auxiliary, or Temporary Register from Memo	ry	(pa	ge 5	-199)		•			•				
[1]	dst-AU = Smem	N	2	1	Х	AU_LOAD	1			1				
	dst-DU = Smem	N	2	1	Х	DU_LOAD	1			1				
[2]	dst-AU = uns(high_byte(Smem))	N	3	1	Х	AU_LOAD	1		•	1		•		
	dst-DU = uns(high_byte(Smem))	N	3	1	Х	DU_LOAD	1			1				
[3]	dst-AU = uns(low_byte(Smem))	N	3	1	Х	AU_LOAD	1			1				
	dst-DU = uns(low_byte(Smem))	N	3	1	Х	DU_LOAD	1			1				

Table 4–1. Algebraic Instruction Set Summary (Continued)

								Addres eration			В	uses		
No.	Instruction	E	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	АСВ	Notes
Loa	nd Accumulator, Auxiliary, or Temporary Register with Immed	liat	e V	alue	(page	e 5-205)								
[1]	dst-AU = k4	Υ	2	1	Х	AU_LOAD								
	dst-DU = k4	Υ	2	1	Х	DU_LOAD								
[2]	dst-AU = -k4	Υ	2	1	Х	AU_LOAD								
	dst-DU = -k4	Υ	2	1	Х	DU_LOAD								
[3]	dst-AU = K16	N	4	1	Х	AU_LOAD								
	dst-DU = K16	N	4	1	Х	DU_LOAD								
Loa	nd Auxiliary or Temporary Register Pair from Memory (page 5	-20	)9)											
	pair(TAx) = Lmem	N	3	1	Х	AU_LOAD	1			2				
Loa	d CPU Register from Memory (page 5-210)	•											•	
[1]	BK03 = Smem	N	3	1	Х	AU_LOAD	1			1				
[2]	BK47 = Smem	N	3	1	Х	AU_LOAD	1			1				
[3]	BKC = Smem	N	3	1	Х	AU_LOAD	1			1				
[4]	BSA01 = Smem	N	3	1	Х	AU_LOAD	1			1				
[5]	BSA23 = Smem	N	3	1	Х	AU_LOAD	1			1				
[6]	BSA45 = Smem	N	3	1	Х	AU_LOAD	1			1				
[7]	BSA67 = Smem	N	3	1	Х	AU_LOAD	1			1				
[8]	BSAC = Smem	N	3	1	Х	AU_LOAD	1			1				
[9]	BRC0 = Smem	N	3	1	Х		1			1				
[10]	BRC1 = Smem	N	3	1	Χ		1			1				
[11]	CDP = Smem	N	3	1	Х	AU_LOAD	1			1				
[12]	CSR = Smem	N	3	1	Х		1			1				
[13]	DP = Smem	N	3	1	Х	AU_LOAD	1			1				
[14]	DPH = Smem	N	3	1	Х	AU_LOAD	1			1				
[15]	PDP = Smem	N	3	1	Х	AU_LOAD	1			1			•	

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

Table 4–1. Algebraic Instruction Set Summary (Continued)

								Addres eration			В	uses		
No.	Instruction	E	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
[16]	SP = Smem	N	3	1	Х	AU_LOAD	1			1				
[17]	SSP = Smem	N	3	1	Х	AU_LOAD	1	•		1				
[18]	TRN0 = Smem	N	3	1	Х	DU_LOAD	1	•		1				
[19]	TRN1 = Smem	N	3	1	Х	DU_LOAD	1	•		1				
[20]	RETA = dbl(Lmem)	N	3	5	Х		1	•		2				
Loa	d CPU Register with Immediate Value (page 5-213)	•					•			•				
[1]	BK03 = k12	Υ	3	1	AD	AU_LOAD								
[2]	BK47 = k12	Υ	3	1	AD	AU_LOAD								
[3]	BKC = k12	Υ	3	1	AD	AU_LOAD								
[4]	BRC0 = k12	Υ	3	1	AD									
[5]	BRC1 = k12	Υ	3	1	AD									
[6]	CSR = k12	Υ	3	1	AD									
[7]	DPH = k7	Υ	3	1	AD	AU_LOAD								
[8]	PDP = k9	Υ	3	1	AD	AU_LOAD								
[9]	BSA01 = k16	N	4	1	AD	AU_LOAD								
[10]	BSA23 = k16	N	4	1	AD	AU_LOAD								
[11]	BSA45 = k16	N	4	1	AD	AU_LOAD								
[12]	BSA67 = k16	N	4	1	AD	AU_LOAD								
[13]	BSAC = k16	N	4	1	AD	AU_LOAD								
[14]	CDP = k16	N	4	1	AD	AU_LOAD								
[15]	DP = k16	N	4	1	AD	AU_LOAD								
[16]	SP = k16	N	4	1	AD	AU_LOAD								
[17]	SSP = k16	N	4	1	AD	AU_LOAD								
Loa	d Extended Auxiliary Register from Memory (page 5-215)	-								-			•	
	XAdst = dbl(Lmem)	N	3	1	Х		1			2				

- Notes: 1) dst-DU, src-AU or dst-DU, src-DU
  - 2) dst-DU, src-AU or dst-AU, src-DU

Instruction Set Summary

Table 4–1. Algebraic Instruction Set Summary (Continued)

									Addres eration			Е	uses		
No.	Instruction	E	s	(	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
Loa	ad Extended Auxiliary Register with Immediate Value (page 5-	21	6)												
	XAdst = k23	N	6		1	AD	AU_LOAD	1			1				
Loa	ad Memory with Immediate Value (page 5-217)														
[1]	Smem = K8	N	3		1	Χ		1					1		
[2]	Smem = K16	Ν	4		1	Χ		1	•	•			1	•	
Loc	ck Access Qualifier (page 5-218)	_						_			_				
	lock()	Ν	2		1	D			•	•				•	
Me	mory Delay (page 5-220)	_						_			_				
	delay(Smem)	Ν	2		1	Χ		2	1		1	1	1	•	
Me	mory-Mapped Register Access Qualifier (page 5-221)														
	mmap()	N	1		1	D									
Мо	dify Auxiliary Register Content (page 5-222)														
	mar(Smem)	N	2		1	AD		1			1				
Мо	dify Auxiliary Register Content with Parallel Multiply (page 5-	224	4)												
	mar(Xmem), ACx = M40(rnd(uns(Ymem) * uns(coef(Cmem))))	N	4		1	Х	DU_MAC1	2	1		2	1			
Мо	dify Auxiliary Register Content with Parallel Multiply and Acc	un	nula	ate	<b>)</b> (p	age 5	5-226)								
[1]	mar(Xmem), ACx = M40(rnd(ACx + (uns(Ymem) * uns(coef(Cmem)))))	N	4		1	Х	DU_MAC1	2	1		2	1			
[2]	$\begin{aligned} & mar(Xmem), \\ & ACx = \frac{M40(rnd((ACx >> #16) + (uns(Ymem) * uns(coef(Cmem))))))}{} \end{aligned}$	N	4		1	X	DU_MAC1	2	1		2	1			
Мо	dify Auxiliary Register Content with Parallel Multiply and Sub	tra	ct	(pa	age	5-23	1)								
	mar(Xmem), ACx = M40(rnd(ACx - (uns(Ymem) * uns(coef(Cmem)))))	N	4		1	Х	DU_MAC1	2	1	•	2	1	٠	ě	

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

Table 4–1. Algebraic Instruction Set Summary (Continued)

								Addres eration			Ві	ises		
No.	Instruction	E	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
Мо	dify Auxiliary or Temporary Register Content (page 5-233)													
[1]	mar(TAy = TAx)	N	3	1	AD		1							
[2]	mar(TAx = P8)	N	3	1	AD		1							
[3]	mar(TAx = D16)	N	4	1	AD		1							
Мо	dify Auxiliary or Temporary Register Content by Addition (pa	ge	5-2	37)										
[1]	mar(TAy + TAx)	N	3	1	AD		1							
[2]	mar(TAx + P8)	N	3	1	AD		1							
Мо	dify Auxiliary or Temporary Register Content by Subtraction	(pa	ige	5-24	11)									
[1]	mar(TAy – TAx)	N	3	1	AD		1							
[2]	mar(TAx – P8)	N	3	1	AD		1							
Мо	dify Data Stack Pointer (SP) (page 5-245)	•								•				
	SP = SP + K8	Υ	2	1	AD									
Мо	dify Extended Auxiliary Register Content (page 5-246)													
[1]	XAdst = mar(Smem)	N	3	1	AD		1			1				
[2]	mar(XACdst = XACsrc)	Υ	3	1	AD		1							
Мо	dify Extended Auxiliary Register Content by Addition (page 5	-24	l9)											
	mar(XACdst + XACsrc)	Υ	3	1	AD		1							
Мо	dify Extended Auxiliary Register Content by Subtraction (pag	je 5	5-25	51)						u				
	mar(XACdst – XACsrc)	Υ	3	1	AD		1							
Мо	ve Accumulator Content to Auxiliary or Temporary Register (	, pac	ge 5	5-25	3)		ı			ļ				
	TAx = HI(ACx)	1	2	1	x	AU_ALU	١.			۱.			1	

Table 4–1. Algebraic Instruction Set Summary (Continued)

								Addres eratior			В	uses		
No.	Instruction	E	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
Мо	ve Accumulator, Auxiliary, or Temporary Register Content (pa	age	5-	254	)									
	dst-AU = src-AU	Υ	2	1	Х	AU_ALU								
	dst-AU = src-DU	Υ	2	1	Х	AU_ALU	•					•	1	
	dst-DU = src	Υ	2	1	Х	DU_ALU								See Note 1.
Мо	ve Auxiliary or Temporary Register Content to Accumulator (	pa	ge :	5-25	56)									
	HI(ACx) = TAx	Υ	2	1	Х	DU_ALU								
Мо	ve Auxiliary or Temporary Register Content to CPU Register	(pa	age	5-2	57)									r.
[1]	BRC0 = TAx	Υ	2	1	Х	AU_ALU								
[2]	BRC1 = TAx	Υ	2	1	Х	AU_ALU								
[3]	CDP = TAx	Υ	2	1	Х	AU_ALU								
[4]	CSR = TAx	Υ	2	1	X	AU_ALU								
[5]	SP = TAx	Υ	2	1	Х	AU_ALU								
[6]	SSP = TAx	Υ	2	1	Х	AU_ALU								
Мо	ve CPU Register Content to Auxiliary or Temporary Register	(pa	age	5-2	59)									
[1]	TAx = BRC0	Υ	2	1	Х	AU_ALU								
[2]	TAx = BRC1	Υ	2	1	Х	AU_ALU								
[3]	TAx = CDP	Υ	2	1	Х	AU_ALU								
[4]	TAx = RPTC	Υ	2	1	Х	AU_ALU								
[5]	TAx = SP	Υ	2	1	Х	AU_ALU								
[6]	TAx = SSP	Υ	2	1	Х	AU_ALU								
Мо	ve Extended Auxiliary Register Content (page 5-261)													
	xdst-AU = xsrc-AU	N	2	1	Х	AU_ALU								
	xdst-AU = xsrc-DU	N	2	1	Х	AU_ALU							1	
	xdst-DU = xsrc	N	2	1	Χ	DU_ALU								See Note 1.

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

Table 4–1. Algebraic Instruction Set Summary (Continued)

								Addres eration			В	uses		
No.	Instruction	E	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
Мо	ve Memory to Memory (page 5-262)													
[1]	Smem = coef(Cmem)	N	3	1	Х		2			1		1		
[2]	coef(Cmem) = Smem	N	3	1	Х		2		•	1		1		
[3]	Lmem = dbl(coef(Cmem))	N	3	1	Х		2		•	2		2		
[4]	dbl(coef(Cmem)) = Lmem	N	3	1	Х		2			2		2		
[5]	dbl(Ymem) = dbl(Xmem)	N	3	1	Х		2			2		2		
[6]	Ymem = Xmem	N	3	1	Х		2			2		2		
Mu	tiply (MPY) (page 5-269)												•	
[1]	ACy = rnd(ACy * ACx)	Υ	2	1	Х	DU_MAC1			•					
[2]	ACy = rnd(ACx * Tx)	Υ	2	1	Х	DU_MAC1								
[3]	ACy = rnd(ACx * K8)	Υ	3	1	Х	DU_ALU								
[4]	$ACy = \frac{\text{rnd}(ACx * K16)}{\text{rnd}(ACx * K16)}$	N	4	1	Х	DU_ALU							•	
[5]	ACx = rnd(Smem * coef(Cmem))[, T3 = Smem]	N	3	1	Х	DU_ALU	1	1		1	1			
[6]	ACy = md(Smem * ACx)[, T3 = Smem]	N	3	1	Х	DU_ALU	1			1			•	
[7]	ACx = rnd(Smem * K8)[, T3 = Smem]	N	4	1	Х	DU_MAC1	1			1			•	
[8]	ACx = M40(rnd(uns(Xmem) * uns(Ymem)))[, T3 = Xmem]	N	4	1	Х	DU_MAC1	2		•	2				
[9]	ACx = rnd(uns(Tx * Smem))[, T3 = Smem]	N	3	1	Х	DU_MAC1	1		•	1				
[10]	ACx = md(Smem * uns(coef(Cmem)))	N	3	1	Х	DU_MAC1	1	1		1	1			
Mu	tiply with Parallel Multiply and Accumulate (page 5-283)													
[1]	ACx = M40(rnd(uns(Xmem) * uns(coef(Cmem)))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(coef(Cmem)))))	N	4	1	Х	DU_MAC1 + DU_MAC2	2	1		2	1			
[2]	ACy = M40(rnd(uns(Smem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx + (uns(Smem) * uns(LO(coef(Cmem))))))	N	4	1	Х	DU_MAC1 + DU_MAC2	1	1	•	1	2	•		
[3]	$ \begin{aligned} &ACy = \frac{\text{M40(rnd(uns(HI(Lmem)) * uns(HI(coef(Cmem)))))}}{\text{ACx} = \frac{\text{M40(rnd(ACx + (uns(LO(Lmem)) * uns(LO(coef(Cmem)))))}}} \end{aligned} $	N	4	1	Х	DU_MAC1 + DU_MAC2	1	1		2	2			
[4]	ACy = M40(rnd(uns(Ymem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx + uns(Xmem) * uns(LO(coef(Cmem)))))	N	5	1	Х	DU_MAC1 + DU_MAC2	2	1	•	2	2	•	•	

Table 4–1. Algebraic Instruction Set Summary (Continued)

								Addres eration			В	uses		
No.	Instruction	E	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
Mu	Itiply with Parallel Multiply and Subtract (page 5-295)													
[1]	ACy = M40(rnd(uns(Smem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx - (uns(Smem) * uns(LO(coef(Cmem))))))	N	4	1	Х	DU_MAC1 + DU_MAC2	1	1		1	2			
[2]	$ \begin{aligned} &ACy = \frac{\text{M40(rnd(uns(HI(Lmem)) * uns(HI(coef(Cmem)))))}}{\text{ACx} = \frac{\text{M40(rnd(ACx - (uns(LO(Lmem))) * uns(LO(coef(Cmem)))))}}} \end{aligned} $	N	4	1	Х	DU_MAC1 + DU_MAC2	1	1		2	2			
[3]	$ \begin{aligned} & ACy = \frac{M40(\text{rnd(uns(Ymem) * uns(HI(coef(Cmem)))))}}{ACx = \frac{M40(\text{rnd(ACx - uns(Xmem) * uns(LO(coef(Cmem)))))}}{} \end{aligned} $	N	5	1	Х	DU_MAC1 + DU_MAC2	2	1		2	2	•		
Mu	Itiply with Parallel Store Accumulator Content to Memory (pa	ge	5-3	05)										
	ACy = md(Tx * Xmem), $Ymem = HI(ACx << T2) [, T3 = Xmem]$	N	4	1	Х	DU_MAC1 + DU_SHIFT	2			2		2		
Mu	Itiply and Accumulate (MAC) (page 5-308)													
[1]	ACy = rnd(ACy + (ACx * Tx))	Υ	2	1	Х	DU_MAC1								
[2]	ACy = rnd((ACy * Tx) + ACx)	Υ	2	1	Х	DU_MAC1								
[3]	ACy = rnd(ACx + (Tx * K8))	Υ	3	1	Х	DU_MAC1							-	
[4]	$ACy = \frac{\text{rnd}(ACx + (Tx * K16))}{ACx + (Tx * K16))}$	N	4	1	Х	DU_MAC1						•		
[5]	$ACx = \frac{\text{rnd}(ACx + (Smem * coef(Cmem)))[, T3 = Smem]}{}$	N	3	1	Х	DU_MAC1	1	1		1	1	•		
[6]	ACy = rnd(ACy + (Smem * ACx))[, T3 = Smem]	N	3	1	Х	DU_MAC1	1			1		•		
[7]	ACy = rnd(ACx + (Tx * Smem))[, T3 = Smem]	N	3	1	Х	DU_MAC1	1			1		•		
[8]	$ACy = \frac{\text{rnd}(ACx + (Smem * K8))[, T3 = Smem]}{}$	N	4	1	Х	DU_MAC1	1			1		•		
[9]	ACy = M40(md(ACx + (uns(Xmem) * uns(Ymem))))[, T3 = Xmem]	N	4	1	Х	DU_MAC1	2			2		•		
[10]	ACy = M40(md((ACx >> #16) + (uns(Xmem) * uns(Ymem))))[, T3 = Xmem]	N	4	1	Х	DU_MAC1	2			2		-		
[11]	ACx = rnd(ACx + (Smem * uns(coef(Cmem))))	N	3	1	Х	DU_MAC1	1	1		1	1	•		
Mu	tiply and Accumulate with Parallel Delay (page 5-325)												•	
	ACx = rnd(ACx + (Smem * coef(Cmem)))[, T3 = Smem], delay(Smem)	N	3	1	Х	DU_MAC1	2	1		1	1	1		
Mu	Itiply and Accumulate with Parallel Load Accumulator from N	len	nor	<b>y</b> (p	age 5-	327)	•			•			•	
	ACx = rnd(ACx + (Tx * Xmem)), ACy = Ymem << #16 [, T3 = Xmem]	N	4	1	Х	DU_MAC1	2			2				

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

Table 4–1. Algebraic Instruction Set Summary (Continued)

								Addres eration			В	uses		
No.	Instruction	E	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
Mu	tiply and Accumulate with Parallel Multiply (page 5-329)													
[1]	ACx = M40(rnd(ACx + (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd(uns(Ymem) * uns(coef(Cmem))))	N	4	1	Χ	DU_MAC1 + DU_MAC2	2	1		2	1			
[2]	ACy = M40(rnd(ACy + (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(uns(Smem) * uns(LO(coef(Cmem)))))	N	4	1	Х	DU_MAC1 + DU_MAC2	1	1		1	2			
[3]	$\begin{split} ACy &= \frac{\text{M40(rnd((ACy >> #16) + (uns(Smem) * uns(HI(coef(Cmem))))))}}{\text{ACx} &= \frac{\text{M40(rnd(uns(Smem) * uns(LO(coef(Cmem)))))}}{\text{M40(rnd(uns(Smem) * uns(LO(coef(Cmem))))))}} \end{split}$	N	4	1	Х	DU_MAC1 + DU_MAC2	1	1		1	2			
[4]	$\begin{split} ACy &= \frac{M40(\text{rnd}(ACy + (\text{uns}(HI(Lmem)) * \text{uns}(HI(coef(Cmem))))))}{ACx &= \frac{M40(\text{rnd}(\text{uns}(LO(Lmem))) * \text{uns}(LO(coef(Cmem)))))}{ACx} \end{split}$	N	4	1	Х	DU_MAC1 + DU_MAC2	1	1		2	2			
[5]	$\begin{split} ACy &= \frac{M40(\text{rnd}((ACy >> #16) + (uns(HI(Lmem)) * uns(HI(coef(Cmem))))))}{ACx &= \frac{M40(\text{rnd}(uns(LO(Lmem))) * uns(LO(coef(Cmem)))))}{ACx} \end{split}$	N	4	1	Х	DU_MAC1 + DU_MAC2	1	1		2	2			
[6]	ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(uns(Xmem) * uns(LO(coef(Cmem)))))	N	5	1	Х	DU_MAC1 + DU_MAC2	2	1		2	2			
Mu	tiply and Accumulate with Parallel Multiply and Subtract (page	ge :	5-34	<del>1</del> 7)			•						•	
[1]	$\begin{split} ACy &= \text{M40(rnd(ACy + (uns(Smem) * uns(HI(coef(Cmem)))))),} \\ ACx &= \text{M40(rnd(ACx - (uns(Smem) * uns(LO(coef(Cmem))))))} \end{split}$	N	4	1	Х	DU_MAC1 + DU_MAC2	1	1		1	2			
[2]	$ \begin{split} ACy &= \text{M40(rnd((ACy >> \#16) + (uns(Smem) * uns(HI(coef(Cmem))))))}, \\ ACx &= \text{M40(rnd(ACx - (uns(Smem) * uns(LO(coef(Cmem))))))} \end{split} $	N	4	1	Х	DU_MAC1 + DU_MAC2	1	1		1	2			
[3]	$ \begin{split} ACy &= \frac{M40(\text{rnd}(ACy + (\text{uns}(HI(Lmem)) * \text{uns}(HI(coef(Cmem))))))}{ACx}, \\ ACx &= \frac{M40(\text{rnd}(ACx - (\text{uns}(LO(Lmem))) * \text{uns}(LO(coef(Cmem)))))}{ACx}, \end{split} $	N	4	1	Х	DU_MAC1 + DU_MAC2	1	1		2	2			
[4]	$ \begin{split} ACy &= \frac{\text{M40(rnd((ACy >> #16) + (uns(HI(Lmem)) * uns(HI(coef(Cmem))))))}}{\text{ACx} &= \frac{\text{M40(rnd(ACx - (uns(LO(Lmem))) * uns(LO(coef(Cmem))))))}}{\text{M40(rnd(ACx - (uns(LO(Lmem))) * uns(LO(coef(Cmem))))))}} \\ \end{split} $	N	4	1	Х	DU_MAC1 + DU_MAC2	1	1		2	2			
[5]	ACy = M40(rnd(ACy + uns(Ymem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx - uns(Xmem) * uns(LO(coef(Cmem)))))	N	5	1	Х	DU_MAC1 + DU_MAC2	2	1		2	2			
[6]	$\begin{split} ACy &= \frac{\text{M40(rnd((ACy >> #16) + (uns(Ymem) * uns(HI(coef(Cmem))))))}}{\text{ACx} &= \frac{\text{M40(rnd(ACx - (uns(Xmem) * uns(LO(coef(Cmem))))))}}{\text{N40(rnd(ACx - (uns(Xmem) * uns(LO(coef(Cmem)))))))}} \end{split}$	N	5	1	Х	DU_MAC1 + DU_MAC2	2	1		2	2			
Mu	tiply and Accumulate with Parallel Store Accumulator Conte	nt i	to N	/lem	ory (p	age 5-367)								
	$ACy = \frac{\text{rnd}(ACy + (Tx * Xmem))}{\text{Ymem} = \text{HI}(ACx << \text{T2}) [, \text{T3} = \text{Xmem}]}$	N	4	1	Х	DU_MAC1 + DU_SHIFT	2			2		2		
Mu	tiply and Subtract (MAS) (page 5-369)												•	
[1]	ACy = rnd(ACy - (ACx * Tx))	Υ	2	1	Х	DU_MAC1								
[2]	ACx = rnd(ACx - (Smem * coef(Cmem)))[, T3 = Smem]	Ν	3	1	Х	DU_MAC1	1	1		1	1			

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

<sup>2)</sup> dst-DU, src-AU or dst-AU, src-DU

Table 4–1. Algebraic Instruction Set Summary (Continued)

								Addres eratio			В	uses		
No.	Instruction	E	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
[3]	ACy = rnd(ACy - (Smem * ACx))[, T3 = Smem]	N	3	1	Х	DU_MAC1	1			1				
[4]	$ACy = \frac{\text{rnd}(ACx - (Tx * Smem))[, T3 = Smem]}{}$	N	3	1	Х	DU_MAC1	1			1				
[5]	ACy = M40(md(ACx - (uns(Xmem) * uns(Ymem))))[, T3 = Xmem]	N	4	1	X	DU_MAC1	2			2				
[6]	ACx = md(ACx - (Smem * uns(coef(Cmem))))	N	3	1	X	DU_MAC1	1	1		1	1			
Mu	tiply and Subtract with Parallel Load Accumulator from Mem	ory	<b>y</b> (p	age	5-379	)	•						•	
	ACx = md(ACx - (Tx * Xmem)), ACy = Ymem << #16 [, T3 = Xmem]	N	4	1	Х	DU_MAC1	2			2				
Mu	tiply and Subtract with Parallel Multiply (page 5-381)	•					•						•	
[1]	ACx = M40(rnd(ACx - (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd(uns(Ymem) * uns(coef(Cmem))))	N	4	1	Х	DU_MAC1 + DU_MAC2	2	1		2	1			
[2]	ACy = M40(rnd(ACy - (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(uns(Smem) * uns(LO(coef(Cmem)))))	N	4	1	Х	DU_MAC1 + DU_MAC2	1	1		1	2			
[3]	$ \begin{split} ACy &= \text{M40(rnd(ACy - (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))}, \\ ACx &= \text{M40(rnd(uns(LO(Lmem)) * uns(LO(coef(Cmem)))))} \end{split} $	N	4	1	Х	DU_MAC1 + DU_MAC2	1	1		2	2			
Mu	tiply and Subtract with Parallel Multiply and Accumulate (page	ge :	5-39	90)			•						•	
[1]	$ \begin{aligned} &ACx = \frac{M40(\text{rnd}(ACx - (\text{uns}(\text{Xmem}) * \text{uns}(\text{coef}(\text{Cmem})))))}{ACy}, \\ &ACy = \frac{M40(\text{rnd}(ACy + (\text{uns}(\text{Ymem}) * \text{uns}(\text{coef}(\text{Cmem})))))}{ACy}. \end{aligned} $	N	4	1	Х	DU_MAC1 + DU_MAC2	2	1		2	1			
[2]	$ \begin{aligned} &ACx = \frac{\text{M40(rnd(ACx - (uns(Xmem) * uns(coef(Cmem))))),}}{ACy = \frac{\text{M40(rnd((ACy >> #16) + (uns(Ymem) * uns(coef(Cmem)))))}}} \end{aligned} $	N	4	1	Х	DU_MAC1 + DU_MAC2	2	1		2	1	٠		
[3]	$ \begin{aligned} & ACy = \frac{M40(rnd(ACy - (uns(Smem)\ ^*\ uns(HI(coef(Cmem))))))}{ACx = \frac{M40(rnd(ACx + (uns(Smem)\ ^*\ uns(LO(coef(Cmem))))))}{N40} \end{aligned} $	N	4	1	Х	DU_MAC1 + DU_MAC2	1	1	•	1	2	•		
[4]	$ \begin{split} &ACy = \text{M40(rnd(ACy - (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))),} \\ &ACx = \text{M40(rnd(ACx + (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))} \end{split} $	N	4	1	Х	DU_MAC1 + DU_MAC2	1	1	•	2	2	•		
Mu	tiply and Subtract with Parallel Store Accumulator Content t	o N	lem	ory	(page	5-401)	•						•	
	ACy = md(ACy - (Tx * Xmem)), Ymem = HI(ACx << T2) [, T3 = Xmem]	N	4	1	Х	DU_MAC1 + DU_SHIFT	2			2		2		
Neg	gate Accumulator, Auxiliary, or Temporary Register Content (	pag	ge 5	-403	3)					ű.				
	dst-AU = -src-AU	Υ	2	1	X	AU_ALU								
	dst-AU = -src-DU	Υ	2	1	Х	AU_ALU							1	

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

Table 4–1. Algebraic Instruction Set Summary (Continued)

								Addres eration			В	ıses		
No.	Instruction	E	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
	dst-DU = -src	Υ	2	1	Х	DU_ALU			•					See Note 1.
No	Operation (NOP) (page 5-405)													
[1]	nop	Υ	1	1	D									
[2]	nop_16	Υ	2	1	D									
Par	allel Modify Auxiliary Register Contents (page 5-406)													
	mar(Xmem) , mar(Ymem) , mar(coef(Cmem))	N	4	1	Х		2	1		2	1			
Par	rallel Multiplies (page 5-407)													
[1]	ACx = M40(rnd(uns(Xmem) * uns(coef(Cmem)))), ACy = M40(rnd(uns(Ymem) * uns(coef(Cmem))))	N	4	1	Х	DU_MAC1 + DU_MAC2	2	1		2	1			
[2]	ACy = M40(rnd(uns(Smem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(uns(Smem) * uns(LO(coef(Cmem)))))	N	4	1	Х	DU_MAC1 + DU_MAC2	1	1		1	2			
[3]	$ \begin{aligned} & ACy = \frac{M40(\text{rnd(uns(HI(Lmem))} * \text{uns(HI(coef(Cmem))))})}{ACx = \frac{M40(\text{rnd(uns(LO(Lmem))} * \text{uns(LO(coef(Cmem))))})}{ACx = \frac{M40(\text{rnd(uns(LO(Lmem))} * \text{uns(LO(coef(Cmem))))})}{ACx = \frac{M40(\text{rnd(uns(HI(Lmem))} * \text{uns(LO(coef(Cmem))))})}{ACx = \frac{M40(\text{rnd(uns(HI(Lmem))} * \text{uns(LO(coef(Cmem))))})}{ACx = \frac{M40(\text{rnd(uns(HI(Lmem))} * \text{uns(LO(coef(Cmem))))})}{ACx = \frac{M40(\text{rnd(uns(HI(Lmem))} * \text{uns(LO(Lmem))} * \text{uns(LO(Lmem))})}{ACx = \frac{M40(\text{rnd(uns(LO(Lmem))} * \text{uns(LO(Lmem))})}{ACx = \frac{M40(\text{rnd(uns(Lo(Lmem))} * \text{uns(Lo(Lmem))} * \text{uns(Lo(Lmem))})})}{ACx = \frac{M40(\text{rnd(uns(Lo(Lmem))} * \text{uns(Lo(Lmem))} * \text{uns(Lo(Lmem))})}{ACx = \frac{M40(\text{rnd(uns(Lo(Lmem))} * \text{uns(Lo(Lmem))} * \text{uns(Lo(Lmem))})}{ACx = \frac{M40(\text{rnd(uns(Lo(Lmem))} * \text{uns(Lo(Lmem))} * \text{uns(Lo(Lmem))})})}{ACx = \frac{M40(\text{rnd(uns(Lo(Lmem))} * \text{uns(Lo(Lmem))} * \text{uns(Lo(Lmem))})}{ACx = \frac{M40(\text{rnd(uns(Lo(Lmem))} * \text{uns(Lo(Lmem))} * \text{uns(Lo(Lmem))})}{ACx = \frac{M40(\text{rnd(uns(Lo(Lmem))} * \text{uns(Lo(Lmem))})}{ACx = \frac{M40(\text{rnd(uns(Ln(Lmem))} * \text{uns(Lo(Lmem))})}{ACx = M40(\text{r$	N	4	1	Х	DU_MAC1 + DU_MAC2	1	1		2	2			
[4]	ACy = M40(rnd(uns(Ymem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(uns(Xmem) * uns(LO(coef(Cmem)))))	N	5	1	Х	DU_MAC1 + DU_MAC2	2	1		2	2			
Par	rallel Multiply and Accumulates (page 5-419)						•			•				
[1]	ACx = M40(rnd(ACx + (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd(ACy + (uns(Ymem) * uns(coef(Cmem)))))	N	4	1	Х	DU_MAC1 + DU_MAC2	2	1		2	1			
[2]	$ \begin{aligned} &ACx = \frac{M40(rnd((ACx >> \#16) + (uns(Xmem) * uns(coef(Cmem)))))}{ACy = \frac{M4(rnd(ACy + (uns(Ymem) * uns(coef(Cmem)))))}{N4(NCy + (NCy + (NCy + NCy) * NCy)))} \end{aligned} $	N	4	1	Х	DU_ALU	2	1		2	1			
[3]	$ \begin{aligned} & ACx = \frac{M40(\text{rnd}((ACx >> \#16) + (\text{uns}(Xmem) * \text{uns}(\text{coef}(Cmem)))))}{ACy = \frac{M40(\text{rnd}((ACy >> \#16) + (\text{uns}(Ymem) * \text{uns}(Ymem) * \text{uns}(\text{coef}(Cmem)))))}{ACy = \frac{M40(\text{rnd}((ACy >> \#16) + (\text{uns}(Ymem) * \text{uns}(Ymem) * un$	N	4	1	Х	DU_MAC1 + DU_MAC2	2	1		2	1			
[4]	$ \begin{aligned} & ACy = \frac{M40(\text{rnd}(ACy + (\text{uns}(Smem) * \text{uns}(Hl(coef(Cmem))))))}{ACx = \frac{M40(\text{rnd}(ACx + (\text{uns}(Smem) * \text{uns}(LO(coef(Cmem))))))}{ACx = \frac{M40(\text{rnd}(ACx + (\text{uns}(Smem) * \text{uns}(ACx + (uns$	N	4	1	Х	DU_MAC1 + DU_MAC2	1	1		1	2			
[5]	$ \begin{split} & ACy = \frac{M40(md(ACy + (uns(Smem) * uns(Hl(coef(Cmem))))))}{ACx = \frac{M40(md((ACx >> #16) + (uns(Smem) * uns(LO(coef(Cmem))))))}{} \end{split} $	N	4	1	Х	DU_MAC1 + DU_MAC2	1	1		1	2			
[6]	$ \begin{aligned} & ACy = \frac{M40(\text{rnd}((ACy >> \#16) + (uns(Smem) * uns(HI(coef(Cmem))))))}{ACx = \frac{M40(\text{rnd}((ACx >> \#16) + (uns(Smem) * uns(LO(coef(Cmem))))))}{ACx = \frac{M40(\text{rnd}((ACx >> \#16) + (uns(Smem) * uns(LO(coef(Cmem))))))}{ACx = \frac{M40(\text{rnd}((ACx >> \#16) + (uns(Smem) * uns(LO(coef(Cmem))))))}{ACx = \frac{M40(\text{rnd}((ACx >> \#16) + (uns(Smem) * uns(HI(coef(Cmem))))))}{ACx = \frac{M40(\text{rnd}((ACx >> \#16) + (uns(Smem) * uns(HI(coef(Cmem)))))}{ACx = \frac{M40(\text{rnd}((ACx >> \#16) + (uns(Smem) * uns(HI(coef(Cmem)))))}{ACx = \frac{M40(\text{rnd}((ACx >> \#16) + (uns(Smem) * uns(HI(coef(Cmem)))))}{ACx = \frac{M40(\text{rnd}((ACx >> \#16) + (uns(Smem) * uns(HI(coef(Cmem))))}{ACx = \frac{M40(\text{rnd}((ACx >> \#16) + (uns(Gmem) * uns(HI(coef(Cmem))))}{ACx = \frac{M40(\text{rnd}((ACx >> \#16) + (uns(Gmem) * uns(HI(coef(Cmem) * uns(HI(coef(Cmem))))))}{ACx = \frac{M40(\text{rnd}((ACx >> \#16) + (uns(Gmem) * uns(HI(coef(Cmem) * uns(HI(coef(Cmem))))))}{ACx = \frac{M40(\text{rnd}((ACx >> \#16) + (uns(Gmem) * uns(HI(coef(Cmem)))))}{ACx = \frac{M40(\text{rnd}((ACx $	N	4	1	Х	DU_MAC1 + DU_MAC2	1	1		1	2			
[7]	$ \begin{aligned} & ACy = \frac{\text{M40(nd(ACy + (uns(HI(Lmem)) * uns(HI(coef(Cmem))))))}}{\text{ACx} = \frac{\text{M40(nd(ACx + (uns(LO(Lmem)) * uns(LO(coef(Cmem)))))}}} \end{aligned} $	N	4	1	Х	DU_MAC1 + DU_MAC2	1	1		2	2			

No.

Instruction

E S

С

Pipe

Operator

Address Generation Unit

DA CA SA

Buses

CR DW

ACB

Notes

- Notes: 1) dst-DU, src-AU or dst-DU, src-DU
  - 2) dst-DU, src-AU or dst-AU, src-DU

Table 4–1. Algebraic Instruction Set Summary (Continued)

								Addres eration			Bu	ses		
No.	Instruction	E	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
Pop	Top of Stack (page 5-469)	•												
[1]	dst1-AU, $dst2-AU = pop()$	Υ	2	1	Х	AU_LOAD	1		1	2				
	dst1-DU, $dst2-DU = pop()$	Υ	2	1	Х	DU_LOAD	1		1	2				
	dst1-AU, dst2-DU = pop()	Υ	2	1	Х	AU_LOAD + DU_LOAD	1	٠	1	2				
	dst1-DU, $dst2-AU = pop()$	Υ	2	1	Х	DU_LOAD + AU_LOAD	1	٠	1	2		•		
[2]	dst-AU = pop()	Υ	2	1	X	AU_LOAD	1		1	1				
	dst-DU = pop()	Υ	2	1	Х	DU_LOAD	1		1	1				
[3]	dst-AU, Smem = pop()	N	3	1	Х	AU_LOAD	1		1	2		1		
	dst-DU, Smem = pop()	N	3	1	Х	DU_LOAD	1		1	2		1		
[4]	ACx = dbl(pop())	Υ	2	1	Х	DU_LOAD	1		1	2				
[5]	Smem = pop()	N	2	1	Х		1		1	1		1		
[6]	dbl(Lmem) = pop()	N	2	1	Х		1		1	2		2		
Pus	sh Accumulator or Extended Auxiliary Register Content to St	acl	k Pc	ointe	ers (pa	age 5-476)	•			•				
	pushboth(xsrc)	Υ	2	1	Х		1		1			2		
Pus	sh to Top of Stack (page 5-477)	•												
[1]	push(src1, src2)	Υ	2	1	Х		1		1			2		
[2]	push(src)	Υ	2	1	Х		1		1			1		
[3]	push(src, Smem)	N	3	1	Х		1		1	1		2		
[4]	dbl(push(ACx))	Υ	2	1	Х		1		1			2		
[5]	push(Smem)	N	2	1	Х		1		1	1		1		
[6]	push(dbl(Lmem))	N	2	1	Х		1		1	2		2		
Rep	peat Block of Instructions Unconditionally (page 5-484)	•					•			•				
[1]	localrepeat{}	Υ	2	1	AD	PU_UNIT								
[2]	blockrepeat{}	Υ	3	1	AD	PU_UNIT								

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

<sup>2)</sup> dst-DU, src-AU or dst-AU, src-DU

Table 4–1. Algebraic Instruction Set Summary (Continued)

								Addres eration			В	uses		
No.	Instruction	Е	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
Re	peat Single Instruction Conditionally (page 5-495)													
	while (cond && (RPTC < k8)) repeat	Υ	3	1	AD	PU_UNIT								
Re	peat Single Instruction Unconditionally (page 5-498)	•								•			•	•
[1]	repeat(k8)	Υ	2	1	AD	PU_UNIT								
[2]	repeat(k16)	Υ	3	1	AD	PU_UNIT								
[3]	repeat(CSR)	Υ	2	1	AD	PU_UNIT								
Re	peat Single Instruction Unconditionally and Decrement CSR	(ра	ge :	5-50	3)									
	repeat(CSR), CSR -= k4	Υ	2	1	Х	AU_ALU + PU_UNIT		-			-			
Re	peat Single Instruction Unconditionally and Increment CSR (	pag	je 5	-505	5)									
[1]	repeat(CSR), CSR += TAx	Υ	2	1	Х	AU_ALU + PU_UNIT		•			•	•		
[2]	repeat(CSR), CSR += k4	Υ	2	1	Х	AU_ALU + PU_UNIT		•			•	•		
Re	turn Conditionally (page 5-508)	•								•			'	•
	if (cond) return	Υ	3	5/5†	R	PU_UNIT	1		1	2				
t x/y	cycles: x cycles = condition true, y cycles = condition false	Į					1						Į.	<u>I</u>
Re	turn Unconditionally (page 5-510)													
	return	Υ	2	5	D	PU_UNIT	1		1	2				
Re	turn from Interrupt (page 5-512)						ı						'	ı
	return_int	N	2	5	D	PU_UNIT	1		1	2			.	
Ro	tate Left Accumulator, Auxiliary, or Temporary Register Cont	' ent	(pa	ige :	5-514)		ı			1			!	I
	dst-AU = BitOut \\ src-AU \\ BitIn	Υ	3	1	Х	AU_ALU								
	dst-AU = BitOut \\ src-DU \\ BitIn	Υ	3	1	Х	AU_ALU							1	
	dst-DU = BitOut \\ src-AU \\ BitIn	Υ	3	1	Х	DU_SHIFT								See Note

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

Table 4–1. Algebraic Instruction Set Summary (Continued)

								Addres eration			В	uses		
No.	Instruction	E	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
Ro	tate Right Accumulator, Auxiliary, or Temporary Register Cor	ntei	nt (p	oage	5-516	6)								
	dst-AU = BitIn // src-AU // BitOut	Υ	3	1	Х	AU_ALU								
	dst-AU = BitIn // src-DU // BitOut	Υ	3	1	X	AU_ALU							1	
	dst-DU = BitIn // src-AU // BitOut	Υ	3	1	X	DU_SHIFT								See Note 1
Ro	und Accumulator Content (page 5-518)													
	ACy = rnd(ACx)	Υ	2	1	Х	DU_ALU								
Sat	turate Accumulator Content (page 5-520)									•				•
	ACy = saturate(rnd(ACx))	Υ	2	1	Х	DU_ALU				.				
Set	Accumulator, Auxiliary, or Temporary Register Bit (page 5-52	22)					ı			ı				ı
	bit(src-AU, Baddr) = #1	N	3	1	Х	AU_ALU	1			.				
	bit(src-DU, Baddr) = #1	N	3	1	Х	DU_BIT	1							
Set	Memory Bit (page 5-523)	•								•				•
	bit(Smem, src) = #1	N	3	1	Х	AU_ALU	1			1		1		
Set	Status Register Bit (page 5-524)	•								•				•
[1]	bit(ST0, k4) = #1	Υ	2	1	Х	AU_ALU								
[2]	bit(ST1, k4) = #1	Υ	2	1	X	AU_ALU								
[3]	bit(ST2, k4) = #1	Υ	2	1	Х	AU_ALU								
[4]	bit(ST3, k4) = #1	Υ	2	1 <sup>†</sup>	Х	AU_ALU								
† WI	nen this instruction is decoded to modify status bit CAFRZ (15), CAEN (14), or CACLR (13), the C	PU p	oipelin	e is flu	ished an	d the instruction i	is exec	uted in	5 cycl	es rega	ardless	of the in	nstruction	context.
Shi	ift Accumulator Content Conditionally (page 5-527)													
[1]	ACx = sftc(ACx, TC1)	Υ	2	1	X	DU_SHIFT								
[2]	ACx = sftc(ACx, TC2)	Υ	2	1	X	DU_SHIFT								
Shi	ift Accumulator Content Logically (page 5-529)													:
[1]	ACy = ACx <<< Tx	Υ	2	1	Х	DU_SHIFT				1.				

- Notes: 1) dst-DU, src-AU or dst-DU, src-DU
  - 2) dst-DU, src-AU or dst-AU, src-DU

Table 4–1. Algebraic Instruction Set Summary (Continued)

								Addres eration			В	ıses		
No.	Instruction	E	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Note
[2]	ACy = ACx <<< #SHIFTW	Υ	3	1	Х	DU_SHIFT								
Shi	ft Accumulator, Auxiliary, or Temporary Register Content Lo	gic	ally	(pa	ge 5-5	32)	•			•			•	
[1]	dst-AU = dst-AU <<< #1	Υ	2	1	Х	AU_ALU + DU_SHIFT		-						
	dst-DU = dst-DU <<< #1	Υ	2	1	Х	DU_SHIFT								
[2]	dst-AU = dst-AU >>> #1	Υ	2	1	Х	AU_ALU + DU_SHIFT		-						
	dst-DU = dst-DU >>> #1	Υ	2	1	X	DU_SHIFT								
Sig	ned Shift of Accumulator Content (page 5-535)	•								•				
[1]	ACy = ACx << Tx	Υ	2	1	Х	DU_SHIFT								
[2]	ACy = ACx << #SHIFTW	Υ	3	1	Х	DU_SHIFT								
[3]	$ACy = ACx \ll Tx$	Υ	2	1	Х	DU_SHIFT								
[4]	ACy = ACx < <c #shiftw<="" td=""><td>Υ</td><td>3</td><td>1</td><td>Х</td><td>DU_SHIFT</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></c>	Υ	3	1	Х	DU_SHIFT								
Sig	ned Shift of Accumulator, Auxiliary, or Temporary Register C	on	tent	(pa	ge 5-	544)								
[1]	dst-AU = dst-AU >> #1	Υ	2	1	Х	AU_ALU + DU_SHIFT								
	dst-DU = dst-DU >> #1	Υ	2	1	Х	DU_SHIFT								
[2]	dst-AU = dst-AU << #1	Υ	2	1	Х	AU_ALU + DU_SHIFT		٠			•			
	dst-DU = dst-DU << #1	Υ	2	1	Х	DU_SHIFT								
Sof	tware Interrupt (page 5-549)	•					•			•			•	
	intr(k5)	N	2	3	D	PU_UNIT	1		1			2		
Sof	tware Reset (page 5-551)	•					•			·			'	
	reset	N	2	?	D	PU_UNIT							.	
Sof	tware Trap (page 5-555)	•					•			ı,			ı	
	trap(k5)	N	2	?	D	PU_UNIT	1		1	۱.		2	.	

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

Table 4–1. Algebraic Instruction Set Summary (Continued)

								Addres eration			В	uses		
No.	Instruction	E	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
Sqı	<b>uare</b> (page 5-557)													
[1]	ACy = rnd(ACx * ACx)	Υ	2	1	Х	DU_MAC1								
[2]	ACx = md(Smem * Smem)[, T3 = Smem]	N	3	1	Х	DU_MAC1	1		•	1				
Squ	uare and Accumulate (page 5-560)													
[1]	ACy = rnd(ACy + (ACx * ACx))	Υ	2	1	Х	DU_MAC1								
[2]	ACy = md(ACx + (Smem * Smem))[, T3 = Smem]	N	3	1	Х	DU_MAC1	1			1				
Sqı	uare and Subtract (page 5-563)	•					•			•				
[1]	ACy = md(ACy - (ACx * ACx))	Υ	2	1	Х	DU_MAC1								
[2]	ACy = md(ACx - (Smem * Smem))[, T3 = Smem]	N	3	1	Х	DU_MAC1	1			1				
Sqı	uare Distance (page 5-566)	•					•			•				
	sqdst(Xmem, Ymem, ACx, ACy)	N	4	1	Х	DU_ALU + DU_MAC1	2			2				
Sto	re Accumulator Content to Memory (page 5-568)	-					-			•				
[1]	Smem = HI(ACx)	N	2	1	Х		1					1		
[2]	Smem = HI(md(ACx))	N	3	1	Х	DU_SHIFT	1					1		
[3]	Smem = LO(ACx << Tx)	N	3	1	Х	DU_SHIFT	1					1		
[4]	Smem = HI(md(ACx << Tx))	N	3	1	Х	DU_SHIFT	1					1		
[5]	Smem = LO(ACx << #SHIFTW)	N	3	1	Х	DU_SHIFT	1					1		
[6]	Smem = HI(ACx << #SHIFTW)	N	3	1	Χ	DU_SHIFT	1		•			1		
[7]	Smem = HI(md(ACx << #SHIFTW))	N	4	1	Х	DU_SHIFT	1		•			1		
[8]	Smem = HI(saturate(uns(rnd(ACx))))	N	3	1	Χ	DU_SHIFT	1		•			1		
[9]	Smem = HI(saturate(uns(rnd(ACx << Tx))))	N	3	1	Χ	DU_SHIFT	1		•			1		
[10]	Smem = HI(saturate(uns(rnd(ACx << #SHIFTW))))	N	4	1	Χ	DU_SHIFT	1		•			1		
[11]	dbi(Lmem) = ACx	N	3	1	Χ		1		•			2		
[12]	dbl(Lmem) = saturate(uns(ACx))	N	3	1	Х	DU_SHIFT	1		•			2		

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

<sup>2)</sup> dst-DU, src-AU or dst-AU, src-DU

Table 4–1. Algebraic Instruction Set Summary (Continued)

								Addres eration			В	uses		
No.	Instruction	E	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
[13]	HI(Lmem) = HI(ACx) >> #1, LO(Lmem) = LO(ACx) >> #1	N	3	1	Х	DU_SHIFT	1					2		
[14]	Xmem = LO(ACx), Ymem = HI(ACx)	N	3	1	Х		2					2		
Sto	re Accumulator Pair Content to Memory (page 5-588)	,											•	
[1]	Lmem = pair(HI(ACx))	N	3	1	Х		1					2		
[2]	Lmem = pair(LO(ACx))	N	3	1	Х		1					2		
Sto	re Accumulator, Auxiliary, or Temporary Register Content to	Ме	emo	ry (	oage (	5-591)							'	
[1]	Smem = src	N	2	1	Х		1					1		
[2]	high_byte(Smem) = src	N	3	1	Х		1					1		
[3]	low_byte(Smem) = src	N	3	1	X		1					1		
Sto	re Auxiliary or Temporary Register Pair Content to Memory (	oag	ge 5	-595	5)		•			•			•	
	Lmem = pair(TAx)	N	3	1	X		1					2		
Sto	re CPU Register Content to Memory (page 5-596)	,					•			•			•	
[1]	Smem = BK03	N	3	1	Х		1					1		
[2]	Smem = BK47	N	3	1	Х		1					1		
[3]	Smem = BKC	N	3	1	Х		1					1		
[4]	Smem = BSA01	N	3	1	Х		1					1		
[5]	Smem = BSA23	N	3	1	Х		1					1		
[6]	Smem = BSA45	N	3	1	Х		1					1		
[7]	Smem = BSA67	N	3	1	Х		1					1		
[8]	Smem = BSAC	N	3	1	Х		1					1		
[9]	Smem = BRC0	N	3	1	Х		1					1		
[10]	Smem = BRC1	N	3	1	Х		1					1		
[11]	Smem = CDP	N	3	1	Х		1					1		
[12]	Smem = CSR	N	3	1	Х		1					1		

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

Table 4–1. Algebraic Instruction Set Summary (Continued)

								Addres eration			В	uses		
No.	Instruction	E	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
[13]	Smem = DP	N	3	1	Х		1					1		
[14]	Smem = DPH	N	3	1	Х		1					1		
[15]	Smem = PDP	N	3	1	Х		1					1		
[16]	Smem = SP	N	3	1	Х		1					1		
[17]	Smem = SSP	N	3	1	Х		1					1		
[18]	Smem = TRN0	N	3	1	Х		1					1		
[19]	Smem = TRN1	N	3	1	Х		1					1		
[20]	dbl(Lmem) = RETA	N	3	5	Х		1					2		
Sto	re Extended Auxiliary Register Content to Memory (page 5-60	00)												•
	dbl(Lmem) = XAsrc	N	3	1	Х		1					2		
Sul	otract Conditionally (page 5-601)													
	subc(Smem, ACx, ACy)	N	3	1	Х	DU_ALU	1			1				
Sub	otraction (page 5-603)	•												•
[1]	dst-AU = dst-AU - src-AU	Υ	2	1	Х	AU_ALU						•		
	dst-AU = dst-AU - src-DU	Υ	2	1	Х	AU_ALU							1	
	dst-DU = dst-DU - src	Υ	2	1	Х	DU_ALU								See Note 1.
[2]	dst-AU = dst-AU - k4	Υ	2	1	Х	AU_ALU								
	dst-DU = dst-DU - k4	Υ	2	1	Х	DU_ALU								
[3]	dst-AU = src-AU - K16	N	4	1	Х	AU_ALU								
	dst-AU = src-DU - K16	N	4	1	Х	AU_ALU							1	
	dst-DU = src - K16	N	4	1	Х	DU_ALU						•		See Note 1.
[4]	dst-AU = src-AU - Smem	N	3	1	Х	AU_ALU	1			1				
	dst-AU = src-DU - Smem	N	3	1	Х	AU_ALU	1			1			1	
	dst-DU = src - Smem	N	3	1	Х	DU_ALU	1			1				See Note 1.

Table 4–1. Algebraic Instruction Set Summary (Continued)

								Addres eration			В	uses		
No.	Instruction	E	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
[5]	dst-AU = Smem – src-AU	N	3	1	Х	AU_ALU	1			1			•	
	dst-AU = Smem - src-DU	N	3	1	Х	AU_ALU	1			1			1	
	dst-DU = Smem – src	N	3	1	Х	DU_ALU	1			1				See Note 1.
[6]	ACy = ACy - (ACx << Tx)	Υ	2	1	Х	DU_SHIFT								
[7]	ACy = ACy - (ACx << #SHIFTW)	Υ	3	1	Х	DU_SHIFT								
[8]	ACy = ACx - (K16 << #16)	N	4	1	Х	DU_ALU								
[9]	ACy = ACx - (K16 << #SHFT)	N	4	1	Х	DU_SHIFT								
[10]	ACy = ACx - (Smem << Tx)	N	3	1	Х	DU_SHIFT	1			1				
[11]	ACy = ACx - (Smem << #16)	N	3	1	Х	DU_ALU	1			1				
[12]	ACy = (Smem << #16) – ACx	N	3	1	Х	DU_ALU	1			1				
[13]	ACy = ACx - uns(Smem) - BORROW	N	3	1	Х	DU_ALU	1			1				
[14]	ACy = ACx – uns(Smem)	N	3	1	Х	DU_ALU	1			1				
[15]	ACy = ACx - (uns(Smem) << #SHIFTW)	N	4	1	Х	DU_SHIFT	1			1				
[16]	ACy = ACx - dbl(Lmem)	N	3	1	Х	DU_ALU	1			2				
[17]	ACy = dbl(Lmem) - ACx	N	3	1	Х	DU_ALU	1			2				
[18]	ACx = (Xmem << #16) - (Ymem << #16)	N	3	1	Х	DU_ALU	2			2				
Sub	otraction with Parallel Store Accumulator Content to Memory	(p	age	e 5-	627)									•
	ACy = (Xmem << #16) – ACx, Ymem = HI(ACy << T2)	N	4	1	Х	DU_ALU + DU_SHIFT	2			2		2		
Swa	ap Accumulator Content (page 5-629)													
[1]	swap(AC0, AC2)	Υ	2	1	Х	DU_SWAP								
[2]	swap(AC1, AC3)	Υ	2	1	Х	DU_SWAP								
Swa	ap Accumulator Pair Content (page 5-630)						1			1				1
	swap(pair(AC0), pair(AC2))	Υ	2	1	Х	DU_SWAP								
Swa	ap Auxiliary Register Content (page 5-631)									1				1
[1]	swap(AR0, AR1)	Υ	2	1	AD	AU_SWAP	1							I

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

<sup>2)</sup> dst-DU, src-AU or dst-AU, src-DU

Table 4–1. Algebraic Instruction Set Summary (Continued)

								Address Generation Unit			Buses			
No.	Instruction	E	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
[2]	swap(AR0, AR2)	Υ	2	1	AD	AU_SWAP								
[3]	swap(AR1, AR3)	Υ	2	1	AD	AU_SWAP								
Sw	ap Auxiliary Register Pair Content (page 5-632)													
	swap(pair(AR0), pair(AR2))	Υ	2	1	AD	AU_SWAP								
Sw	ap Auxiliary and Temporary Register Content (page 5-633)													•
[1]	swap(AR4, T0)	Υ	2	1	AD	AU_SWAP								
[2]	swap(AR5, T1)	Υ	2	1	AD	AU_SWAP								
[3]	swap(AR6, T2)	Υ	2	1	AD	AU_SWAP								
[4]	swap(AR7, T3)	Υ	2	1	AD	AU_SWAP								
Sw	ap Auxiliary and Temporary Register Pair Content (page 5-63	5)												
[1]	swap(pair(AR4), pair(T0))	Υ	2	1	AD	AU_SWAP								
[2]	swap(pair(AR6), pair(T2))	Υ	2	1	AD	AU_SWAP								
Sw	ap Auxiliary and Temporary Register Pairs Content (page 5-6	37)	)											
	swap(block(AR4), block(T0))	Υ	2	1	AD	AU_SWAP								
Sw	ap Temporary Register Content (page 5-639)	•												-
[1]	swap(T0, T2)	Υ	2	1	AD	AU_SWAP								
[2]	swap(T1, T3)	Υ	2	1	AD	AU_SWAP								
Sw	ap Temporary Register Pair Content (page 5-640)													-
	swap(pair(T0), pair(T2))	Υ	2	1	AD	AU_SWAP								
Tes	t Accumulator, Auxiliary, or Temporary Register Bit (page 5-6	41	)				•			•				1
[1]	TC1 = bit(src-AU, Baddr)	N	3	1	Х	AU_ALU	1							
	TC1 = bit(src-DU, Baddr)	N	3	1	Х	DU_BIT	1							
[2]	TC2 = bit(src-AU, Baddr)	N	3	1	Х	AU_ALU	1							
	TC2 = bit(src-DU, Baddr)	N	3	1	Х	DU_BIT	1							

- Notes: 1) dst-DU, src-AU or dst-DU, src-DU
  - 2) dst-DU, src-AU or dst-AU, src-DU

Table 4–1. Algebraic Instruction Set Summary (Continued)

							Address Generation Unit			t Buses					
No.	Instruction	Е	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes	
Tes	t Accumulator, Auxiliary, or Temporary Register Bit Pair (pag	e 5	5-64	3)											
	bit(src-AU, pair(Baddr))	N	3	1	Х	AU_ALU	1						•		
	bit(src-DU, pair(Baddr))	N	3	1	Х	DU_BIT	1	•							
Tes	t Memory Bit (page 5-645)														
[1]	TCx = bit(Smem, src)	N	3	1	Х	AU_ALU AU_ALU	1			1					
[2]	TCx = bit(Smem, k4)	N	3	1	Х	AU_ALU	1	•		1					
Tes	t and Clear Memory Bit (page 5-648)														
[1]	TC1 = bit(Smem, k4), bit(Smem, k4) = #0	N	3	1	Х	AU_ALU	1			1		1			
[2]	TC2 = bit(Smem, $k4$ ), bit(Smem, $k4$ ) = #0	N	3	1	Х	AU_ALU	1			1		1	•		
Tes	t and Complement Memory Bit (page 5-649)						·			•				•	
[1]	TC1 = bit(Smem, k4), cbit(Smem, k4)	N	3	1	Х	AU_ALU	1			1		1			
[2]	TC2 = bit(Smem, k4), cbit(Smem, k4)	N	3	1	Х	AU_ALU	1			1		1			
Tes	t and Set Memory Bit (page 5-650)						·			•					
[1]	TC1 = bit(Smem, k4), bit(Smem, k4) = #1	N	3	1	Х	AU_ALU	1			1		1			
[2]	TC2 = bit(Smem, k4), bit(Smem, k4) = #1	N	3	1	Х	AU_ALU	1			1		1	-		

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

# **Instruction Set Descriptions**

This chapter provides detailed information on the TMS320C55x  $^{\text{\tiny TM}}$  DSP algebraic instruction set.

See Section 1.1, *Instruction Set Terms, Symbols, and Abbreviations*, for definitions of symbols and abbreviations used in the description of each instruction. See Chapter 4 for a summary of the instruction set.

#### abdst

#### Absolute Distance

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	abdst(Xmem, Ymem, ACx, ACy)	No	4	1	X

**Opcode** 

| 1000 0110 | XXXM MMYY | YMMM DDDD | 1111 xxn%

**Operands** 

ACx, ACy, Xmem, Ymem

Description

This instruction executes two operations in parallel: one in the D-unit MAC and one in the D-unit ALU:

$$ACy = ACy + |HI(ACx)|$$
  
 $ACx = (Xmem << #16) - (Ymem << #16)$ 

The absolute value of accumulator ACx content is computed and added to accumulator ACy content through the D-unit MAC. When an overflow is detected according to M40:

- ☐ the destination accumulator overflow status bit (ACOVy) is set
- ☐ the destination register (ACy) is saturated according to SATD

The Ymem content shifted left 16 bits is subtracted from the Xmem content shifted left 16 bits in the D-unit ALU.

- ☐ Input operands (Xmem and Ymem) are sign extended to 40 bits according to SXMD.
- ☐ CARRY status bit depends on M40. Subtraction borrow bit is reported in CARRY status bit. It is the logical complement of CARRY status bit.
- ☐ When an overflow is detected according to M40:
  - the destination accumulator overflow status bit (ACOVx) is set
  - the destination register (ACx) is saturated according to SATD

### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

When C54CM = 1, the subtract operation does not have any overflow detection, report, and saturation after the shifting operation.

**Status Bits** 

Affected by C54CM, FRCT, M40, SATD, SXMD

Affects ACOVx, ACOVy, CARRY

Repeat

This instruction can be repeated.

See Also

See the following other related instructions:

□ Square Distance

# Example

Syntax	Description
abdst(*AR0+, *AR1, AC0, AC1)	The absolute value of the content of AC0 is added to the content of AC1 and the result is stored in AC1. The content addressed by AR1 is subtracted from the content addressed by AR0 and the result is stored in AC0. The content of AR0 is incremented by 1.

Before				After			
AC0	00	0000	0000	AC0	00	4500	0000
AC1	00	E800	0000	AC1	00	E800	0000
AR0			202	AR0			203
AR1			302	AR1			302
202			3400	202			3400
302			EF00	302			EF00
ACOV0			0	ACOV0			0
ACOV1			0	ACOV1			0
CARRY			0	CARRY			0
M40			1	M40			1
SXMD			1	SXMD			1

## **ABS**

#### Absolute Value

#### **Syntax Characteristics**

No. S	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1] d	st =  src	Yes	2	1	Х

#### **Opcode**

0011 001E FSSS FDDD

# Operands

dst, src

#### Description

This instruction computes the absolute value of the source register (src).

- ☐ When the destination register (dst) is an accumulator:
  - The operation is performed on 40 bits in the D-unit ALU.
  - If an auxiliary or temporary register is the source operand of the instruction, the 16 LSBs of the auxiliary or temporary register are sign extended to 40 bits according to SXMD.
  - If M40 = 0, the sign of the source register is extracted at bit position 31. If src(31) = 1, the source register content is negated. If src(31) = 0, the source register content is moved to the destination accumulator.
  - If M40 = 1, the sign of the source register is extracted at bit position 39. If src(39) = 1, the source register content is negated. If src(39) = 0, the source register content is moved to the destination accumulator.
  - During the 40-bit move operation, an overflow and CARRY bit status are detected according to M40:
    - The destination accumulator overflow status bit (ACOVx) is set.
    - The destination register is saturated according to SATD.
    - The CARRY status bit is updated as follows: If the result of the operation stored in the destination register is 0, CARRY is set; otherwise, CARRY is cleared.
- ☐ When the destination register (dst) is an auxiliary or temporary register:
  - The operation is performed on 16 bits in the A-unit ALU.
  - If an accumulator is the source operand of the instruction, the 16 LSBs of the accumulator are used to perform the operation.
  - The sign of the source register is extracted at bit position 15. If src(15) = 1, the source register content is negated. If src(15) = 0, the source register content is moved to the destination register. Overflow is detected at bit position 15.
  - The destination register is saturated according to SATA.

# Compatibility with C54x devices (C54CM = 1)

When C54CM =1, this instruction is executed as if M40 status bit was locally set to 1. To ensure compatibility versus overflow detection and saturation of destination accumulator, this instruction must be executed with M40 = 0.

**Status Bits** Affected by C54CM, M40, SATA, SATD, SXMD

> Affects ACOVx, CARRY

Repeat This instruction can be repeated.

See Also See the following other related instructions:

☐ Addition with Absolute Value

# Example 1

Syntax	Description
AC1 =  AC0	The absolute value of the content of AC0 is stored in AC1.

Before			After			
AC1	00 000	00 2000	AC1	7D	FFFF EDCC	7
AC0	82 000	00 1234	AC0	82	0000 1234	Ł
M40		1	M40		1	L

# Example 2

Syntax	Description
AC1 =  AR1	The absolute value of the content of AR1 is stored in AC1.

Before				After			
AC1	00	0000	2000	AC1	00	0000	0000
AR1			0000	AR1			0000
CARRY			0	CARRY			1

# Example 3

Syntax	Description
AC1 =  AR1	The absolute value of the content of AR1 is stored in AC1. Since SXMD = 1, AR1 content
	is sign extended. The resulting 40-bit data is negated since M40 = 0 and AR1(31) = 1.

Before				After				
AC1	00	0000	2000	AC1	00	0000	7900	
AR1			8700	AR1			8700	
M40			0	M40			0	
SXMD			1	SXMD			1	

# Example 4

Syntax	Description
T1 =  AC0	The absolute value of the content of AC0(15–0) is stored in T1. The sign bit is extracted at
	AC0(15). Since AC0(15) = 0, T1 = AC0(15–0).

Before			After			
T1		2000	T1			1234
AC0	80 0002	1234	AC0	80	0002	1234

# Example 5

Syntax	Description
T1 =  AC0	The absolute value of the content of AC0(15–0) is stored in T1. The sign bit is extracted at AC0(15). Since AC0(15) = 1, T1 equals the negated value of AC0(15–0).

Before				After			
T1			2000	T1			6DCC
AC0	80	0002	9234	AC0	80	0002	9234

ADD

**Addition** 

# **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = dst + src	Yes	2	1	Х
[2]	dst = dst + k4	Yes	2	1	X
[3]	dst = src + K16	No	4	1	X
[4]	dst = src + Smem	No	3	1	X
[5]	$ACy = ACy + (ACx \ll Tx)$	Yes	2	1	X
[6]	ACy = ACy + (ACx << #SHIFTW)	Yes	3	1	X
[7]	ACy = ACx + (K16 << <b>#16</b> )	No	4	1	X
[8]	ACy = ACx + (K16 << #SHFT)	No	4	1	X
[9]	ACy = ACx + (Smem << Tx)	No	3	1	X
[10]	ACy = ACx + (Smem << #16)	No	3	1	X
[11]	ACy = ACx + uns(Smem) + CARRY	No	3	1	X
[12]	ACy = ACx + uns(Smem)	No	3	1	X
[13]	ACy = ACx + (uns(Smem) << #SHIFTW)	No	4	1	X
[14]	ACy = ACx + <b>dbl(</b> Lmem <b>)</b>	No	3	1	X
[15]	ACx = (Xmem << #16) + (Ymem << #16)	No	3	1	X
[16]	Smem = Smem + K16	No	4	1	X

These instructions perform an addition operation. Description

**Status Bits** Affected by CARRY, C54CM, M40, SATA, SATD, SXMD

> Affects ACOVx, ACOVy, CARRY

See Also	See	e the following other related instructions:
		Addition or Subtraction Conditionally
		Addition or Subtraction Conditionally with Shift
		Addition with Absolute Value
		Addition with Parallel Store Accumulator Content to Memory
		Addition, Subtraction, or Move Accumulator Content Conditionally
		Dual 16-Bit Additions
		Dual 16-Bit Addition and Subtraction
		Dual 16-Bit Subtraction and Addition
		Subtraction

#### Addition

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = dst + src	Yes	2	1	Х
Opcod	e	001	0 01	0E FSS	S FDDD

# **Operands**

dst, src

# Description

This instruction performs an addition operation between two registers.

- ☐ When the destination (dst) operand is an accumulator:
  - The operation is performed on 40 bits in the D-unit ALU.
  - Input operands are sign extended to 40 bits according to SXMD.
  - If an auxiliary or temporary register is the source (src) operand of the instruction, the 16 LSBs of the auxiliary or temporary register are sign extended according to SXMD.
  - Overflow detection and CARRY status bit depends on M40.
  - When an overflow is detected, the accumulator is saturated according to SATD.
- ☐ When the destination (dst) operand is an auxiliary or temporary register:
  - The operation is performed on 16 bits in the A-unit ALU.
  - If an accumulator is the source (src) operand of the instruction, the 16 LSBs of the accumulator are used to perform the operation.
  - Addition overflow detection is done at bit position 15.
  - When an overflow is detected, the destination register is saturated according to SATA.

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

**Status Bits** Affected by M40, SATA, SATD, SXMD

> Affects ACOVx, CARRY

Repeat This instruction can be repeated.

# **Example**

Syntax	Description
AC0 = AC0 + AC1	The content of AC1 is added to the content of AC0 and the result is stored in AC0.

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	dst = dst + k4	Yes	2	1	Х
Opcod	е	010	0 00	0E kkk	k FDDD

## **Operands**

dst, k4

#### Description

This instruction performs an addition operation between a register content and a 4-bit unsigned constant, k4.

- ☐ When the destination (dst) operand is an accumulator:
  - The operation is performed on 40 bits in the D-unit ALU.
  - Overflow detection and CARRY status bit depends on M40.
  - When an overflow is detected, the accumulator is saturated according to SATD.
- ☐ When the destination (dst) operand is an auxiliary or temporary register:
  - The operation is performed on 16 bits in the A-unit ALU.
  - Addition overflow detection is done at bit position 15.
  - When an overflow is detected, the destination register is saturated according to SATA.

## Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

**Status Bits** Affected by M40, SATA, SATD

> Affects ACOVx, CARRY

Repeat This instruction can be repeated.

Syntax	Description
AC0 = AC0 + k4	The content of AC0 is added to an unsigned 4-bit value and the result is stored in AC0.

## **Syntax Characteristics**

No.	Syntax							Paralle Enable I	· <del>-</del>	Cycles	Pipeline
[3]	dst = src + K16							No	4	1	Х
Opcod	le				0111	1011	KKKK	KKKK :	KKKK KI	KKK FDI	DD FSSS
Opera	nds	dst	t, K1	6, src							
Descri	ption				n perforn d consta		•	peration b	etween a	register of	content and
			Wh	en the	destinat	ion (ds	) operar	nd is an a	ccumulat	or:	
				The o	peration	is perfe	ormed or	n 40 bits i	n the D-u	ınit ALU.	
			If an auxiliary or temporary register is the source (src) operand of instruction, the 16 LSBs of the auxiliary or temporary register are s extended according to SXMD.								
			■ The 16-bit constant, K16, is sign extended to 40 bits accord SXMD.								ccording to
				Overfl	low dete	ction ar	nd CARF	RY status	bit deper	nds on M4	40.
				When to SA		flow is c	letected,	the accur	mulator is	saturated	d according
			☐ When the destination (dst) operand is an auxiliary or temporary regis								ıry register:
				The o	peration	is perfe	ormed or	n 16 bits i	n the A-u	nit ALU.	
								, ,	perand o		ruction, the ration.
				Additi	on overf	low det	ection is	done at b	oit positio	n 15.	
					an over		detected	d, the des	stination	register is	s saturated
		Co	mpa	atibility	with C	54x de	vices (C	<b>54CM</b> = 1	1)		
		Wł	nen t	his inst	truction i	s execu	ited with	M40 = 0	, compati	bility is er	nsured.
Status	Bits	Aff	ecte	d by	M40,	SATA, S	SATD, S	XMD			
		Aff	ects		ACOV	/x, CAR	RY				

# Repeat Example

Syntax	Description
AC1 = AC0 + #2E00h	The content of AC0 is added to the signed 16-bit value (2E00h) and the result is stored in AC1.

This instruction can be repeated.

#### **Syntax Characteristics**

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline
[4]	dst = src + Smem		No	3	1	Х
Opcod	e	1101	0110 AA	AA AZ	AAI FDI	DD FSSS

**Operands** 

dst, Smem, src

Description

This instruction performs an addition operation between a register content and the content of a memory (Smem) location.

- ☐ When the destination (dst) operand is an accumulator:
  - The operation is performed on 40 bits in the D-unit ALU.
  - If an auxiliary or temporary register is the source (src) operand of the instruction, the 16 LSBs of the auxiliary or temporary register are sign extended according to SXMD.
  - The content of the memory location is sign extended to 40 bits according to SXMD.
  - Overflow detection and CARRY status bit depends on M40.
  - When an overflow is detected, the accumulator is saturated according to SATD.
- ☐ When the destination (dst) operand is an auxiliary or temporary register:
  - The operation is performed on 16 bits in the A-unit ALU.
  - If an accumulator is the source (src) operand of the instruction, the 16 LSBs of the accumulator are used to perform the operation.
  - Addition overflow detection is done at bit position 15.
  - When an overflow is detected, the destination register is saturated according to SATA.

## Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

**Status Bits** Affected by M40, SATA, SATD, SXMD

> Affects ACOVx, CARRY

# Repeat

This instruction can be repeated.

Syntax	Description
T1 = T0 + *AR3+	The content of T0 is added to the content addressed by AR3 and the result is
	stored in T1. AR3 is incremented by 1.

Before		After	
AR3	0302	AR3	0303
302	EF00	302	EF00
T0	3300	T0	3300
T1	0	T1	2200
CARRY	0	CARRY	1

# **Syntax Characteristics**

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline			
[5]	ACy = ACy + (A)	ACx <	< Tx)	Yes	2	1	Х			
Opcod	е			010	)1 10	1E DDS	SS ss00			
Operar	nds	AC	x, ACy, Tx							
Descri	ption	This instruction performs an addition operation between an accumulator content ACy and an accumulator content ACx shifted by the content of Tx.								
			☐ The operation is performed on 40 bits in the D-unit shifter.							
			Input operands are sign extended to 40 bits according to SXMD.							
			The shift operation is equivalent to the signed shift instruction.							
			Overflow detection and CARRY status bit depends on M40.							
			When an overflow is detected, the accumulator is saturated according to SATD.							
			Compatibility with C54x devices (C54CM = 1)							
			When this instruction is executed with $M40 = 0$ , compatibility is ensured. When $C54CM = 1$ :							
			An intermediary shift operation is properties no overflow detection, report, and			-				

#### **Status Bits**

Affected by C54CM, M40, SATD, SXMD

Affects ACOVy, CARRY

operation.

to −1.

Repeat

This instruction can be repeated.

# **Example**

Syntax	Description
AC0 = AC0 + (AC1 << T0)	The content of AC1 shifted by the content of T0 is added to the content of AC0 and the result is stored in AC0.

☐ The 6 LSBs of Tx are used to determine the shift quantity. The 6 LSBs of Tx define a shift quantity within -32 to +31. When the value is between -32to -17, a modulo 16 operation transforms the shift quantity to within -16

# **Syntax Characteristics**

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline				
[6]	ACy = ACy +	(ACx << #SHIFTW)	Yes	3	1	Х				
Opcode	е	0001	000E DDS	ss oc	)11   xxS	SH IFTW				
Operar	nds	ACx, ACy, SHIFTW								
Descrip	otion	·	This instruction performs an addition operation between an accumulator content ACy and an accumulator content ACx shifted by the 6-bit value, SHIFTW.							
		☐ The operation is performed on 4	☐ The operation is performed on 40 bits in the D-unit shifter.							
		☐ Input operands are sign extended to 40 bits according to SXMD.								
		☐ The shift operation is equivalent to the signed shift instruction.								
		<ul><li>Overflow detection and CARRY</li></ul>	☐ Overflow detection and CARRY status bit depends on M40.							
		☐ When an overflow is detected, the accumulator is saturated according to SATD.								
		Compatibility with C54x devices (C54CM = 1)								
		When this instruction is executed with $M40 = 0$ , compatibility is ensured. When $C54CM = 1$ , an intermediary shift operation is performed as if $M40$ is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.								
Status	Bits	Affected by C54CM, M40, SATD	, SXMD							
		Affects ACOVy, CARRY								
Repeat	:	This instruction can be repeated.								

Syntax	Description
· · · · · · · · · · · · · · · · · · ·	The content of AC1 shifted left by 31 bits is added to the content of AC0 and the result is stored in AC0.

## **Syntax Characteristics**

Cyntax	Characteris										
No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline			
[7]	ACy = ACx +	+ <b>(</b> K16 <b>&lt;&lt; #16)</b>	(16 <b>&lt;&lt; #16)</b>				1	Х			
Opcode         0111 1010   KKKK					KKKK KKK	K KK	KK SSD	D 000x			
Operar	nds	ACx, ACy, K16	;								
Descri	ption		This instruction performs an addition operation between an accumulator content ACx and a 16-bit signed constant, K16, shifted left by 16 bits.								
		☐ The operat	☐ The operation is performed on 40 bits in the D-unit ALU.								
		Input opera	☐ Input operands are sign extended to 40 bits according to SXMD.								
		The shift op	☐ The shift operation is equivalent to the signed shift instruction.								
		Overflow detection and CARRY status bit depends on M40.									
		☐ When an o SATD.	<b>_</b>								
		Compatibility	Compatibility with C54x devices (C54CM = 1)								
		C54CM = 1, ar set to 1 and no	When this instruction is executed with C54CM = 1, an intermediary shift operate set to 1 and no overflow detection, reshifting operation.				as if M4	0 is locally			
Status	Bits	Affected by	C54CM, I	M40, SATD,	SXMD						
		Affects	ACOVy, C	CARRY							

# Example

Repeat

Syntax	Description
AC0 = AC1 + (#2E00h << #16)	A signed 16-bit value (2E00h) shifted left by 16 bits is added to the content of AC1 and the result is stored in AC0.
	Content of ACT and the result is stored in ACC.

This instruction can be repeated.

# **Syntax Characteristics**

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline				
[8]	ACy = ACx +	(K16 << #SHFT)	No	4	1	Х				
Opcod	е	0111 0000   KKKK	KKKK KKI	KK KI	KKK SSI	DD SHFT				
Operar	nds	ACx, ACy, K16, SHFT								
Descri	ption	·	This instruction performs an addition operation between an accumulator content ACx and a 16-bit signed constant, K16, shifted left by the 4-bit value, SHFT.							
		☐ The operation is performed on 4	☐ The operation is performed on 40 bits in the D-unit shifter.							
		☐ Input operands are sign extended to 40 bits according to SXMD.								
	The shift operation is equivalent to the signed shift instruction									
		<ul><li>Overflow detection and CARRY</li></ul>	n M40.							
When an overflow is detected, the accumulator is satu SATD.					turated a	ccording to				
		Compatibility with C54x devices (C54CM = 1)								
When this instruction is executed with M40 = 0, compatibility is ensured the companion of t					0 is locally					
Status	Bits	Affected by C54CM, M40, SATD	, SXMD							
		Affects ACOVy, CARRY								
Repeat	t	This instruction can be repeated.								

Syntax	Description
AC0 = AC1 + (#2E00h << #15)	A signed 16-bit value (2E00h) shifted left by 15 bits is added to the content of AC1 and the result is stored in AC0.

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline	
[9]	ACy = ACx + (S	mem	n <b>&lt;&lt;</b> Tx <b>)</b>			No	3	1	Х	
Opcode	e				1101	1101 AA	AA AA	AI SSI	D ss00	
Operar	nds	AC	x, ACy, Tx,	Smem						
Descri	otion	This instruction performs an addition operation between an accumulator content ACx and the content of a memory (Smem) location shifted by the content of Tx.								
			The operation is performed on 40 bits in the D-unit shifter.							
			Input operands are sign extended to 40 bits according to SXMD.							
			☐ The shift operation is equivalent to the signed shift instruction.							
			Overflow detection and CARRY status bit depends on M40.							
			When an overflow is detected, the accumulator is saturated according to SATD.							
		Compatibility with C54x devices (C54CM = 1)								
		When this instruction is executed with M40 = 0, compatibility is ensured. When $C54CM = 1$ :								
				diary shift ope w detection, r				-		
			Tx define a	s of Tx are us shift quantity nodulo 16 ope	within –32	to +31. Whe	n the v	alue is be	etween –32	
Status	Bits	Aff	ected by	C54CM, M4	0, SATD,	SXMD				
		Aff	ects	ACOVy, CAI	RRY					
Repeat		This instruction can be repeated.								

Syntax	Description			
AC0 = AC1 + (*AR1 << T0)	The content addressed by AR1 shifted left by the content of T0 is added to the			
	content of AC1 and the result is stored in AC0.			

Before				After			
AC0	00	0000	0000	AC0	00	2330	0000
AC1	00	2300	0000	AC1	00	2300	0000
T0			000C	T0			000C
AR1			0200	AR1			0200
200			0300	200			0300
SXMD			0	SXMD			0
M40			0	M40			0
ACOV0			0	ACOV0			0
CARRY			0	CARRY			1

# **Syntax Characteristics**

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline	
[10]	ACy = ACx + (S)	Smerr	<< #16)		No	3	1	Х	
Opcode	9			1101	1110 AA	AA AA	AI SSI	DD 0100	
Operan	ıds	ACx, ACy, Smem							
Descrip	otion	This instruction performs an addition operation between an accumulator content ACx and the content of a memory (Smem) location shifted left by 16 bits.							
			The operat	tion is performed on 4	0 bits in the D	-unit A	LU.		
		☐ Input operands are sign extended to 40 bits according to SXMD.							
			☐ The shift operation is equivalent to the signed shift instruction.						
<ul> <li>Overflow detection and CARR of the addition generates a care the CARRY status bit is not aff</li> </ul>				tion generates a carry	, the CARRY				
			When an o	overflow is detected, th	ne accumulato	or is sat	turated a	ccording to	
		Со	mpatibility	with C54x devices (	C54CM = 1)				
When this instruction is executed with $M40 = 0$ , compatibility is ensigned to 1 and no overflow detection, report, and saturation is dornaliting operation.						0 is locally			
Status	Bits	Aff	ected by	C54CM, M40, SATE	, SXMD				
		Affe	ects	ACOVy, CARRY					

## Example

Repeat

Syntax	Description
AC0 = AC1 + (*AR3 << #16)	The content addressed by AR3 shifted left by 16 bits is added to the content of AC1 and the result is stored in AC0.

This instruction can be repeated.

# **Syntax Characteristics**

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline
[11]	ACy = ACx + ur	ns(Sr	nem) + CAR	RY		No	3	1	Х
Opcod	e				1101	1111 AAA	A AA	AI SSD	D 100u
Operar	nds	AC	x, ACy, Sm	nem					
Descri	ption	This instruction performs an addition operation of the accumulator content ACx, the content of a memory (Smem) location, and the value of the CARRY status bit.							
			The opera	ation is performed	l on 40	bits in the D	-unit A	LU.	
			Input ope	rands are extende	ed to 4	0 bits accord	ing to	uns.	
				optional uns keyw memory location			-		he content
				optional uns key nt of the memory l D.		• • •			-
			Overflow	detection and CA	RRY s	tatus bit dep	ends o	n M40.	
			When an SATD.	overflow is detect	ed, the	e accumulato	r is sat	urated ad	cording to
		Со	mpatibility	/ with C54x devi	ces (C	54CM = 1)			
		Wh	nen this ins	truction is execute	ed with	M40 = 0, co	mpatib	oility is en	sured.
Status	Bits	Aff	ected by	CARRY, M40,	SATD,	SXMD			
		Aff	ects	ACOVy, CARR	Υ				
Repeat	t	This instruction can be repeated.							
Evenn	.la								

Syntax	Description
AC0 = AC1 + uns(*AR3) + CARRY	The CARRY status bit and the unsigned content addressed by AR3 are added to the content of AC1 and the result is stored in AC0.

## **Syntax Characteristics**

Syntax	Characteristic	S							
No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline
[12]	ACy = ACx + ur	n <mark>s(</mark> Sr	mem)			No	3	1	Х
Opcod	е				1101	1111 AA	AA AA	AI SSI	DD 110u
Operar	nds	AC	x, ACy, Sm	em					
Descri	ption	This instruction performs an addition operation between an accumulator content ACx and the content of a memory (Smem) location.							
			The opera	tion is performe	ed on 40	bits in the D	-unit A	LU.	
			Input opera	ands are exten	ded to 4	0 bits accord	ling to	uns.	
				optional uns key memory locatio			-	•	the content
				optional uns ke nt of the memory	-				
			Overflow o	letection and C	ARRY s	tatus bit dep	ends o	n M40.	
			When an o	overflow is dete	cted, the	accumulato	r is sat	urated a	ccording to
		Co	mpatibility	with C54x de	vices (C	54CM = 1)			
		Wł	nen this inst	ruction is execu	uted with	M40 = 0, co	mpatik	oility is en	sured.
Status	Bits	Aff	ected by	M40, SATD,	SXMD				
		Aff	ects	ACOVy, CAR	RY				
Repeat	i	This instruction can be repeated.							

Syntax	Description
AC0 = AC1 + uns(*AR3)	The unsigned content addressed by AR3 is added to the content of AC1 and the result is stored in AC0.

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[13]	ACy = ACx + (uns(Smem) << #SHIFTW)	No	4	1	Χ

# **Opcode Operands**

1111 1001 AAAA AAAI uxsh iftw ssdd ooxx ACx, ACy, SHIFTW, Smem

#### Description

This instruction performs an addition operation between an accumulator content ACx and the content of a memory (Smem) location shifted by the 6-bit value, SHIFTW.

- ☐ The operation is performed on 40 bits in the D-unit shifter.
- ☐ Input operands are extended to 40 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 40 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 40 bits according to SXMD.
- ☐ The shift operation is equivalent to the signed shift instruction.
- Overflow detection and CARRY status bit depends on M40.
- When an overflow is detected, the accumulator is saturated according to SATD.

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, an intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.

#### **Status Bits**

Affected by C54CM, M40, SATD, SXMD

Affects ACOVy, CARRY

#### Repeat

This instruction can be repeated.

Syntax	Description
AC0 = AC1 + (uns(*AR3) << #31)	The unsigned content addressed by AR3 shifted left by 31 bits is added to the content of AC1 and the result is stored in AC0.

# **Syntax Characteristics**

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline		
[14]	ACy = ACx + dk	ol(Lmem)		No	3	1	Х		
Opcod	e		1110	1101 AA	AA AA	AI SSI	D 000n		
Operar	nds	ACx, ACy, Li	mem						
Descri	otion	This instruction performs an addition operation between an accumulator content ACx and the content of data memory operand dbl(Lmem).							
		☐ The data	memory operand dbl(Ln	nem) addres	ses ar	e aligned	:		
		■ if Lmem address is even: most significant word = Lmem, least significant word = Lmem + 1							
		■ if Lmem address is odd: most significant word = Lmem, least significant word = Lmem - 1							
		☐ The operation is performed on 40 bits in the D-unit ALU.							
		☐ Input operands are sign extended to 40 bits according to SXMD.							
		<ul> <li>Overflow detection and CARRY status bit depends on M40.</li> </ul>							
		☐ When an overflow is detected, the accumulator is saturated according SATD.					ccording to		
		Compatibili	ty with C54x devices (C	254CM = 1)					
		When this in	struction is executed with	M40 = 0, co	mpatil	oility is er	sured.		
Status	Bits	Affected by	M40, SATD, SXMD						
		Affects	ACOVy, CARRY						
Repeat	:	This instructi	on can be repeated.						
Evamn	lo.								

Syntax	Description
` ,	The content (long word) addressed by AR3 and AR3 + 1 is added to the content of AC1 and the result is stored in AC0. Because this instruction is a long-operand instruction, AR3 is incremented by 2 after the execution.

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline			
[15]	ACx = (Xmem	<< #16) + (Ymen	n <b>&lt;&lt; #16)</b>		No	3	1	Х			
Opcod	е			1000	0001 XXX	KM MM	IYY YMM	IM 00DD			
Operar	nds	ACx, Xmem,	Ymem								
Descri	ption	memory oper	This instruction performs an addition operation between the content of data memory operand Xmem shifted left 16 bits, and the content of data memory operand Ymem shifted left 16 bits.								
		☐ The oper	ation is perfoi	med on 40	bits in the D	-unit A	LU.				
		☐ Input ope	☐ Input operands are sign extended to 40 bits according to SXMD.								
		☐ The shift operation is equivalent to the signed shift instruction.									
		<ul> <li>Overflow detection and CARRY status bit depends on M40.</li> </ul>									
		☐ When an SATD.	☐ When an overflow is detected, the accumulator is saturated according to SATD.								
		Compatibility with C54x devices (C54CM = 1)									
		When this instruction is executed with $M40 = 0$ , compatibility is ensured. When $C54CM = 1$ , an intermediary shift operation is performed as if $M40$ is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.									
Status	Bits	Affected by	C54CM, M	140, SATD,	SXMD						
		Affects	ACOVx, C	ARRY							
Repeat	t	This instruction can be repeated.									
Examp	le										
Syntax	(		Descripti	on							

Syntax	Description
AC0 = (*AR3 << #16) + (*AR4 << #16)	The content addressed by AR3 shifted left by 16 bits is added to the content addressed by AR4 shifted left by 16 bits and the result is stored in AC0.

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline	
[16]	Smem = Smem	+ K1	16			No	4	1	Х	
Opcode	9			1111 0	111 AAAA	AAAI KK	KK KK	KK KK	K KKKK	
Operan	ds	K1	6, Smem							
Descrip	otion			•	s an addition ontent of a m	•			bit signed	
			The operat	tion is perf	ormed on 40	bits in the D	-unit A	LU.		
		☐ Input operands are sign extended to 40 bits according to SXMD and shifted by 16 bits to the MSBs before being added.							SXMD and	
		Addition overflow is detected at bit position 31. If an overflow is detected, accumulator 0 overflow status bit (ACOV0) is set.								
			Addition carry report in CARRY status bit is extracted at bit position 31.							
			☐ If SATD is 1 when an overflow is detected, the result is saturated before being stored in memory. Saturation values are 7FFFh or 8000h.							
		Со	mpatibility	with C54	x devices (C	54CM = 1)				
		Wh	nen this instr	ruction is e	executed with	M40 = 0, co	mpatik	oility is er	sured.	
Status	Bits	Aff	ected by	SATD, S	XMD					
		Aff	ects	ACOV0,	CARRY					
Repeat		Thi	is instruction	n can be re	epeated.					
Examp	le									

Syntax	Description
*AR3 = *AR3 + #2E00h	The content addressed by AR3 is added to a signed 16-bit value (2E00h) and the result is stored back into the location addressed by AR3.

# **ADDV**

# Addition with Absolute Value

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(ACy)	+  AC	Cx  <mark>)</mark>		Yes	2	1	Х
Opcod	е				010	01 01	.0E DDS	SS 000%
Operar	nds	AC	х, АСу					
Descri	ption			computes the absolut nulator ACy. This instr				
				ute value of accumu 6) by 00001h or 1FFI or.		•	•	
			If FRCT =	1, the absolute value	is multiplied b	y 2.		
			Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.					
			Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.					
		☐ When an addition overflow is detected, the accumulator is saturate according to SATD.						saturated
			The result part of AC	of the absolute value	of the higher	part of	ACx is ir	the lower
		Co	mpatibility	with C54x devices (	C54CM = 1)			
		Wh	en this insti	ruction is executed with	th M40 = 0, co	mpatik	oility is en	sured.
Status	Bits	Affe	ected by	FRCT, M40, RDM, S	SATD, SMUL			
		Affe	ects	ACOVy				
Repeat	t	Thi	s instructior	n can be repeated.				

See Also	See the following other related instructions:
	☐ Absolute Value
	☐ Addition
	☐ Addition or Subtraction Conditionally
	☐ Addition or Subtraction Conditionally with Shift
	Addition Subtraction or Move Accumulator Content Conditionally

Syntax	Description
AC0 = AC0 +  AC1	The absolute value of AC1 is added to the content of AC0 and the result is stored in AC0.

# ADD::MOV

## Addition with Parallel Store Accumulator Content to Memory

#### **Syntax Characteristics**

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline	
[1]	ACy = ACx + Ymem = <b>HI(</b> A	•	•	No	4	1	Х	
Opcod	e		1000 0111   X	MYY YYMM MXXX	M SS	DD   100	x xxxx	
Operar	nds	AC	x, ACy, T2, Xmem, Ymem					
Descri	ption	Thi	s instruction performs two op	erations in paralle	l: addit	ion and s	store.	
			e first operation performs an a dithe content of data memory					
			The operation is performed	on 40 bits in the D	-unit A	LU.		
			Input operands are sign exte	ended to 40 bits a	ccordin	g to SXM	1D.	
			☐ The shift operation is equivalent to the signed shift instruction.					
			Overflow detection and CAC54CM = 1, an intermediar locally set to 1 and no overflafter the shifting operation.	y shift operation i	s perf	ormed as	if M40 is	
			When an overflow is detecte SATD.	d, the accumulato	r is sat	urated ac	ccording to	
		sto is r	e second operation shifts the res ACy(31–16) to data mem not within –32 to +31, the shift formed with this value.	ory operand Ymei	n. If th	e 16-bit v	alue in T2	
			The input operand is shifted	in the D-unit shifte	er acco	ording to	SXMD.	
			After the shift, the high part of the memory location.	of the accumulator	, ACy(	31–16), i	s stored to	

## Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When this instruction is executed with C54CM = 1, the 6 LSBs of T2 are used to determine the shift quantity. The 6 LSBs of T2 define a shift quantity within -32 to +31. When the 16-bit value in T2 is between -32 to -17, a modulo 16 operation transforms the shift quantity to within -16 to -1.

☐ If the SST bit = 1 and the SXMD bit = 0, then the saturate and uns keywords are applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:

ACy = ACx + (Xmem << #16), Ymem = HI(saturate(uns(ACy << T2)))

☐ If the SST bit = 1 and the SXMD bit = 1, then only the saturate keyword is applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:

ACy = ACx + (Xmem << #16), Ymem = HI(saturate(ACy << T2))

Status Bits Affected by C54CM, M40, RDM, SATD, SST, SXMD

Affects ACOVy, CARRY

**Repeat** This instruction can be repeated.

**See Also** See the following other related instructions:

☐ Addition

☐ Store Accumulator Content to Memory

☐ Subtraction with Parallel Store Accumulator Content to Memory

Syntax	Description
AC0 = AC1 + (*AR3 << #16), *AR4 = HI(AC0 << T2)	Both instructions are performed in parallel. The content addressed by AR3 shifted left by 16 bits is added to the content of AC1 and the result is stored in AC0. The content of AC0 is shifted by the content of T2, and AC0(31–16) is stored at the address of AR4.

#### **ADDSUBCC**

#### Addition or Subtraction Conditionally

#### **Syntax Characteristics**

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = adsc(Smem, ACx, TC1)		No	3	1	Х
[2]	ACy = adsc(Smem, ACx, TC2)		No	3	1	X
Opcode	TC1	1101	1110 AA	AA A <i>I</i>	AAI SSI	D 0000

# **Operands**

#### ACx, ACy, Smem, TCx

TC2

#### **Description**

This instruction evaluates the selected TCx status bit and based on the result of the test, either an addition or a subtraction is performed. Evaluation of the condition on the TCx status bit is performed during the Execute phase of the instruction.

1101 1110 AAAA AAAI SSDD 0001

TC1 or TC2	Operation
0	ACy = ACx - (Smem << #16)
1	ACy = ACx + (Smem << #16)

 $\Box$  **TCx = 0**, then ACy = ACx – (Smem << #16):

This instruction subtracts the content of a memory (Smem) location shifted left by 16 bits from accumulator ACx and stores the result in accumulator ACy.

- The operation is performed on 40 bits in the D-unit ALU.
- Input operands are sign extended to 40 bits according to SXMD.
- The shift operation is equivalent to the signed shift instruction.
- Overflow detection and CARRY status bit depends on M40.
- When an overflow is detected, the accumulator is saturated according to SATD.
- $\Box$  **TCx = 1**, then ACy = ACx + (Smem << #16):

This instruction performs an addition operation between accumulator ACx and the content of a memory (Smem) location shifted left by 16 bits and stores the result in accumulator ACy.

- The operation is performed on 40 bits in the D-unit ALU.
- Input operands are sign extended to 40 bits according to SXMD.

- The shift operation is equivalent to the signed shift instruction.
- Overflow detection and CARRY status bit depends on M40.
- When an overflow is detected, the accumulator is saturated according to SATD.

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, an intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.

Status Bits Affected by C54CM, M40, SATD, SXMD, TCx

Affects ACOVy, CARRY

**Repeat** This instruction can be repeated.

**See Also** See the following other related instructions:

- ☐ Addition or Subtraction Conditionally with Shift
- ☐ Addition, Subtraction, or Move Accumulator Content Conditionally

#### **Example 1**

Syntax	Description
,	If TC1 = 1, the content addressed by AR3 shifted left by 16 bits is added to the content of AC1 and the result is stored in AC0. If TC1 = 0, the content addressed by AR3 shifted left by 16 bits is subtracted from the content of AC1 and the result is stored in AC0.

Syntax	Description			
AC1 = adsc(*AR1, AC0, TC2)	TC2 = 1, the content addressed by AR1 shifted left by 16 bits is added to the content of AC0 and the result is stored in AC1. The result generated an overflow and a carry.			

Before				After				
AC0	00	EC00	0000	AC0	00	EC00	0000	
AC1	00	0000	0000	AC1	01	1F00	0000	
AR1			0200	AR1			0200	
200			3300	200			3300	
TC2			1	TC2			1	
SXMD			0	SXMD			0	
M40			0	M4 0			0	
ACOV1			0	ACOV1			1	
CARRY			0	CARRY			1	

### ADDSUB2CC

#### Addition or Subtraction Conditionally with Shift

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = ads2c(Smem, ACx, Tx, TC1, TC2)	No	3	1	Х

#### **Opcode**

| 1101 | 1101 | AAAA | AAAI | SSDD | ss10

## **Operands**

ACx, ACy, Tx, Smem, TC1, TC2

#### Description

This instruction evaluates the TC1 status bit and based on the result of the test, either an addition or a subtraction is performed; this instruction evaluates the TC2 status bit and based on the result of the test, either a shift left by 16 bits or the content of Tx is performed. Evaluation of the condition on the TCx status bits is performed during the Execute phase of the instruction.

TC1	TC2	Operation
0	0	ACy = ACx - (Smem << Tx)
0	1	ACy = ACx - (Smem << #16)
1	0	ACy = ACx + (Smem << Tx)
1	1	ACy = ACx + (Smem << #16)

 $\Box$  TC1 = 0 and TC2 = 0, then ACy = ACx – (Smem << Tx):

This instruction subtracts the content of a memory (Smem) location shifted left by the content of Tx from an accumulator ACx and stores the result in accumulator ACy.

 $\Box$  TC1 = 0 and TC2 = 1, then ACy = ACx – (Smem << #16):

This instruction subtracts the content of a memory (Smem) location shifted left by 16 bits from an accumulator ACx and stores the result in accumulator ACy.

- The operation is performed on 40 bits in the D-unit shifter.
- Input operands are sign extended to 40 bits according to SXMD.
- The shift operation is equivalent to the signed shift instruction.
- Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.
- When an overflow is detected, the accumulator is saturated according to SATD.

$\Box$ TC1 = 1 and TC2 = 0, t	hen ACv = ACx + (	(Smem <<	: Tx):
-------------------------------	-------------------	----------	--------

This instruction performs an addition operation between an accumulator ACx and the content of a memory (Smem) location shifted left by the content of Tx and stores the result in accumulator ACy.

 $\Box$  **TC1 = 1 and TC2 = 1**, then ACy = ACx + (Smem << #16):

This instruction performs an addition operation between an accumulator ACx and the content of a memory (Smem) location shifted left by 16 bits and stores the result in accumulator ACy.

- The operation is performed on 40 bits in the D-unit shifter.
- Input operands are sign extended to 40 bits according to SXMD.
- The shift operation is equivalent to the signed shift instruction.
- Overflow detection and CARRY status bit depends on M40.
- When an overflow is detected, the accumulator is saturated according to SATD.

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1:

o 1 and
shifting

□ The 6 LSBs of Tx are used to determine the shift quantity. The 6 LSBs of Tx define a shift quantity within -32 to +31. When the value is between -32 to -17, a modulo 16 operation transforms the shift quantity to within -16 to -1.

**Status Bits** 

Affected by C54CM, M40, SATD, SXMD, TC1, TC2

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

See Also

See the following other related instructions:

- ☐ Addition or Subtraction Conditionally
- Addition, Subtraction, or Move Accumulator Content Conditionally

Syntax	Description
AC2 = ads2c(*AR2, AC0, T1, TC1, TC2)	TC1 = 1 and TC2 = 0, the content addressed by AR2 shifted left by the content of T1 is added to the content of AC0 and the result is stored in AC2. The result generated an overflow.

Before				After			
AC0	00	EC00	0000	AC0	00	EC00	0000
AC2	00	0000	0000	AC2	00	EC00	CC00
AR2			0201	AR2			0201
201			3300	201			3300
T1			0002	T1			0002
TC1			1	TC1			1
TC2			0	TC2			0
M40			0	M40			0
ACOV2			0	ACOV2			1
CARRY			0	CARRY			0

# **ADDSUBCC**

Addition, Subtraction, or Move Accumulator Content Conditionally

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = adsc(Smem, ACx, TC1, TC2)	No	3	1	X

#### Opcode

| 1101 | 1110 | AAAA | AAAI | SSDD | 0010

#### **Operands**

ACx, ACy, Smem, TC1, TC2

#### Description

This instruction evaluates the TCx status bits and based on the result of the test, an addition, a move, or a subtraction is performed. Evaluation of the condition on the TCx status bits is performed during the Execute phase of the instruction.

TC1	TC2	Operation
0	0	ACy = ACx - (Smem << #16)
0	1	ACy = ACx
1	0	ACy = ACx + (Smem << #16)
1	1	ACy = ACx

 $\Box$  TC2 = 1, then ACy = ACx:

This instruction moves the content of ACx to ACy.

- The 40-bit move operation is performed in the D-unit ALU.
- During the 40-bit move operation, an overflow is detected according to M40:
  - the destination accumulator overflow status bit (ACOVy) is set.
  - the destination register (ACy) is saturated according to SATD.
- $\Box$  **TC1 = 0 and TC2 = 0**, then ACy = ACx (Smem << #16):

This instruction subtracts the content of a memory (Smem) location shifted left by 16 bits from accumulator ACx and stores the result in accumulator ACy.

- The operation is performed on 40 bits in the D-unit ALU.
- Input operands are sign extended to 40 bits according to SXMD.
- The shift operation is equivalent to the signed shift instruction.
- Overflow detection and CARRY status bit depends on M40.
- When an overflow is detected, the accumulator is saturated according to SATD.

#### $\Box$ **TC1 = 1 and TC2 = 0**, then ACy = ACx + (Smem << #16):

This instruction performs an addition operation between accumulator ACx and the content of a memory (Smem) location shifted left by 16 bits and stores the result in accumulator ACy.

- The operation is performed on 40 bits in the D-unit ALU.
- Input operands are sign extended to 40 bits according to SXMD.
- The shift operation is equivalent to the signed shift instruction.
- Overflow detection and CARRY status bit depends on M40.
- When an overflow is detected, the accumulator is saturated according to SATD.

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, an intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.

Status Bits Affected by C54CM, M40, SATD, SXMD, TC1, TC2

Affects ACOVy, CARRY

**Repeat** This instruction can be repeated.

**See Also** See the following other related instructions:

- ☐ Addition or Subtraction Conditionally
- ☐ Addition or Subtraction Conditionally with Shift

Syntax	Description
AC0 = adsc(*AR3, AC1, TC1, TC2)	If TC2 = 1, the content of AC1 is stored in AC0. If TC2 = 0 and TC1 = 1, the content addressed by AR3 shifted left by 16 bits is added to the content of AC1 and the result is stored in AC0. If TC2 = 0 and TC1 = 0, the content addressed by AR3 shifted left by 16 bits is subtracted from the content of AC1 and the result is stored in AC0.

# **AND**

Bitwise AND

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = dst & src	Yes	2	1	Х
[2]	dst = src & k8	Yes	3	1	X
[3]	dst = src & k16	No	4	1	Х
[4]	dst = src & Smem	No	3	1	Х
[5]	ACy = ACy & (ACx <<< #SHIFTW)	Yes	3	1	Х
[6]	ACy = ACx & (k16 <<< #16)	No	4	1	Х
[7]	ACy = ACx & (k16 <<< #SHFT)	No	4	1	Х
[8]	Smem = Smem & k16	No	4	1	Х

Description	These instructions perform a bitwise AND operation:
	☐ In the D-unit, if the destination operand is an accumulator.
	☐ In the A-unit ALU, if the destination operand is an auxiliary or temporary register.
	☐ In the A-unit ALU, if the destination operand is the memory.
Status Bits	Affected by C54CM
	Affects none
See Also	See the following other related instructions:
	☐ Bitwise AND Memory with Immediate Value and Compare to Zero
	☐ Bitwise OR
	☐ Bitwise Exclusive OR (XOR)

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = dst & src	Yes	2	1	Χ

**Opcode** 

0010 100E FSSS FDDD

**Operands** 

dst, src

Description

This instruction performs a bitwise AND operation between two registers.

- ☐ When the destination (dst) operand is an accumulator:
  - The operation is performed on 40 bits in the D-unit ALU.
  - Input operands are zero extended to 40 bits.
  - If an auxiliary or temporary register is the source (src) operand of the instruction, the 16 LSBs of the auxiliary or temporary register are zero extended.
- ☐ When the destination (dst) operand is an auxiliary or temporary register:
  - The operation is performed on 16 bits in the A-unit ALU.
  - If an accumulator is the source (src) operand of the instruction, the 16 LSBs of the accumulator are used to perform the operation.

**Status Bits** Affected by none

> Affects none

Repeat This instruction can be repeated.

Syntax	Description
AC1 = AC1 & AC0	The content of AC0 is ANDed with the content of AC1 and the result is stored in AC1.

Before				After				
AC0	7E	2355	4FC0	AC0	7E	2355	4FC0	
AC1	0F	E340	5678	AC1	0E	2340	4640	

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	dst = src & k8	Yes	3	1	Х

## Opcode

0001 100E kkkk kkkk FDDD FSSS

#### **Operands**

dst, k8, src

#### Description

This instruction performs a bitwise AND operation between a source (src) register content and an 8-bit value, k8.

- ☐ When the destination (dst) operand is an accumulator:
  - The operation is performed on 40 bits in the D-unit ALU.
  - Input operands are zero extended to 40 bits.
  - If an auxiliary or temporary register is the source (src) operand of the instruction, the 16 LSBs of the auxiliary or temporary register are zero extended.
- ☐ When the destination (dst) operand is an auxiliary or temporary register:
  - The operation is performed on 16 bits in the A-unit ALU.
  - If an accumulator is the source (src) operand of the instruction, the 16 LSBs of the accumulator are used to perform the operation.

#### **Status Bits**

Affected by none

Affects none

## Repeat

This instruction can be repeated.

Syntax	Description
AC0 = AC1 & #FFh	The content of AC1 is ANDed with the unsigned 8-bit value (FFh) and the result is stored in AC0.

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline
[3]	dst = src & k16					No	4	1	Х
Opcod	e			0111	1101 kkkk	kkkk   kk	ck kk	kk   FDI	DD FSSS
Operar	nds	dst,	k16, src						
Description				•	ms a bitwise A 16-bit unsigned	•		veen a so	ource (src)
			When th	e destina	tion (dst) opera	and is an acc	umulat	or:	
			■ The	operation	is performed o	n 40 bits in tl	ne D-u	nit ALU.	
			■ Input	operand	s are zero exte	nded to 40 b	its.		
			instru	-	or temporary re and LSBs of the	-			
			When the	e destinat	ion (dst) operai	nd is an auxi	iary or	tempora	ry register:
			■ The	operation	is performed o	n 16 bits in tl	ne A-u	nit ALU.	
					ator is the sour e accumulator a	· , .			•
Status	Bits	Affe	ected by	none					
		Affe	ects	none					
Repeat 7			s instruction	on can be	repeated.				
Examp	le								
Syntax	,	De	scription						

Syntax	Description
AC0 = AC1 & #FFFFh	The content of AC1 is ANDed with the unsigned 16-bit value (FFFFh) and the result is stored in AC0.

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	dst = src & Smem	No	3	1	Х

## Opcode

1101 1001 AAAA AAAI FDDD FSSS

#### **Operands**

dst, Smem, src

#### Description

This instruction performs a bitwise AND operation between a source (src) register content and a memory (Smem) location.

- ☐ When the destination (dst) operand is an accumulator:
  - The operation is performed on 40 bits in the D-unit ALU.
  - Input operands are zero extended to 40 bits.
  - If an auxiliary or temporary register is the source (src) operand of the instruction, the 16 LSBs of the auxiliary or temporary register are zero extended.
- ☐ When the destination (dst) operand is an auxiliary or temporary register:
  - The operation is performed on 16 bits in the A-unit ALU.
  - If an accumulator is the source (src) operand of the instruction, the 16 LSBs of the accumulator are used to perform the operation.

#### **Status Bits**

Affected by none

Affects

none

## Repeat

This instruction can be repeated.

Syntax	Description
AC0 = AC1 & *AR3	The content of AC1 is ANDed with the content addressed by AR3 and the result is stored in AC0.

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline	
[5]	ACy = ACy & (A	4Cx <<	<b>#</b> SHIFTW	/)		Yes	3	1	Х	
Opcode					0001	000E DDS	SS 00	00 xxS	H IFTW	
<b>Operands</b> AC:			ACy, SHI	FTW						
Description		This instruction performs a bitwise AND operation between an accumulator (ACy) content and an accumulator (ACx) content shifted by the 6-bit value, SHIFTW.								
			☐ The shift and AND operations are performed in one cycle in the D-unit shifter.							
			When M40 = 0 and C54CM = 0, input operands ACx(31-0) are zero extended to 40 bits. Otherwise, ACx(39-0) is used as is.						) are zero	
		_	The input operand (ACx) is shifted by a 6-bit immediate value in the D-unit shifter.							
		_ T	The CARRY status bit is not affected by the logical shift operation.							
		Compatibility with C54x devices (C54CM = 1)								
			When C54CM = 1, the intermediary logical shift is performed as if M40 is loset to 1. The 8 upper bits of the 40-bit intermediary result are not cleared					•		
Status Bits Affec			ed by C54CM, M40							
		Affec	ts	none						
Repeat	<b>Deat</b> This instruction can be repeated.				d.					
Examp	le									
Syntax			Descripti	on						

Syntax	Description
	The content of AC0 is ANDed with the content of AC1 logically shifted left by 30 bits and the result is stored in AC0.

## **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	ACy = ACx & (k16 <<< #16)	No	4	1	Χ

0111 1010 kkkk kkkk kkkk kkkk SSDD 010x Opcode

**Operands** ACx, ACy, k16

This instruction performs a bitwise AND operation between an accumulator Description

(ACx) content and a 16-bit unsigned constant, k16, shifted left by 16 bits.

☐ The operation is performed on 40 bits in the D-unit ALU.

☐ Input operands are zero extended to 40 bits.

☐ The input operand (k16) is shifted 16 bits to the MSBs.

**Status Bits** Affected by none

> Affects none

Repeat This instruction can be repeated.

Syntax	Description				
,	The content of AC1 is ANDed with the unsigned 16-bit value (FFFFh) logically shifted left by 16 bits and the result is stored in AC0.				

# **Syntax Characteristics**

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline
[7]	ACy = ACx & (I	<16 <b>&lt;</b>	<< #SHFT)			No	4	1	Х
Opcod	e			0111	0010 kkkk	kkkk kk	ck kk	kk SSI	DD SHFT
Operar	nds	AC	x, ACy, k16	, SHFT					
Description		This instruction performs a bitwise AND operation between an accumulator (ACx) content and a 16-bit unsigned constant, k16, shifted left by the 4-bit value, SHFT.							
		☐ The shift and AND operations are performed in one cycle in the D-unit shifter.							
					nd C54CM = 0 s. Otherwise, A			,	) are zero
			The input operand (k16) is shifted by a 4-bit immediate value in the D-u shifter.					the D-unit	
			The CARR	Y status	s bit is not affe	cted by the lo	gical s	hift opera	ation.
Status	Bits	Aff	ected by	C54CI	M, M40				
		Aff	ects	none					

# Example

Repeat

Syntax	Description				
AC0 = AC1 & (#FFFFh <<< #15)	The content of AC1 is ANDed with the unsigned 16-bit value (FFFFh) logically shifted left by 15 bits and the result is stored in AC0.				

This instruction can be repeated.

#### Bitwise AND

## **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	Smem = Smem & k16	No	4	1	Х

| 1111 0100 | AAAA AAAI | kkkk kkkk kkkk kkkk Opcode

**Operands** k16, Smem

This instruction performs a bitwise AND operation between a memory (Smem) Description

location and a 16-bit unsigned constant, k16.

☐ The operation is performed on 16 bits in the A-unit ALU.

☐ The result is stored in memory.

**Status Bits** Affected by none

> Affects none

Repeat This instruction can be repeated.

## **Example**

Syntax	Description
	The content addressed by AR1 is ANDed with the unsigned 16-bit value (FC0h) and the result is stored in the location addressed by AR1.

Before After \*AR1 5678 \*AR1 0640

# **BAND**

## Bitwise AND Memory with Immediate Value and Compare to Zero

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	<b>TC1</b> = Smem & k16	No	4	1	Х
[2]	TC2 = Smem & k16	No	4	1	X

Opcode	TC1	1111	0010	AAAA	AAAI	kkkk	kkkk	kkkk	kkkk
	TC2	1111	0011	AAAA	AAAI	kkkk	kkkk	kkkk	kkkk

## Operands

k16, Smem, TCx

## **Description**

This instruction performs a bit field manipulation in the A-unit ALU. The 16-bit field mask, k16, is ANDed with the memory (Smem) operand and the result is compared to 0:

#### **Status Bits**

Affected by none

Affects TCx

#### Repeat

This instruction can be repeated.

#### See Also

See the following other related instructions:

☐ Bitwise AND

Syntax	Description
TC1 = *AR0 & #0060h	The unsigned 16-bit value (0060h) is ANDed with the content addressed by AR0. The result is 1, TC1 is set to 1.

Before		After	
*ARO	0040	*ARO	0040
TC1	0	TC1	1

# OR

Bitwise OR

# **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = dst   src	Yes	2	1	Х
[2]	dst = src   k8	Yes	3	1	X
[3]	dst = src   k16	No	4	1	X
[4]	dst = src   Smem	No	3	1	Х
[5]	ACy = ACy   (ACx <<< #SHIFTW)	Yes	3	1	Х
[6]	$ACy = ACx \mid (k16 <<< #16)$	No	4	1	Х
[7]	ACy = ACx   <b>(</b> k16 <b>&lt;&lt;&lt; #</b> SHFT <b>)</b>	No	4	1	Х
[8]	Smem = Smem   k16	No	4	1	Х

Description	These instructions perform a bitwise OR operation:
	☐ In the D-unit, if the destination operand is an accumulator.
	In the A-unit ALU, if the destination operand is an auxiliary or temporary register.
	☐ In the A-unit ALU, if the destination operand is the memory.
Status Bits	Affected by C54CM
	Affects none
See Also	See the following other related instructions:
	☐ Bitwise AND
	☐ Bitwise Exclusive OR (XOR)

# **Syntax Characteristics**

No. Syntax		Parallel Enable Bit	Size	Cycles	Pipeline
[1] dst = dst   src		Yes	2	1	Х
Opcode		001	.0 10	1E FSS	S FDDD
Operands	dst, src				
Description	This instruction performs a bitwise OR	operation b	etwee	n two reg	isters.
	☐ When the destination (dst) operand	d is an accu	mulato	or:	
	■ The operation is performed on	40 bits in th	ne D-u	nit ALU.	
	Input operands are zero exten	ded to 40 bi	ts.		
	If an auxiliary or temporary reg instruction, the 16 LSBs of the extended.			. , .	
	☐ When the destination (dst) operand	d is an auxil	iary or	tempora	ry register:
	The operation is performed on	16 bits in th	ne A-ui	nit ALU.	
	<ul><li>If an accumulator is the source</li><li>16 LSBs of the accumulator are</li></ul>	. , .			
Status Bits	Affected by none				
	Affects none				
Repeat	This instruction can be repeated.				
Example					
Syntax	Description				

The content of AC0 is ORed with the content of AC1 and the result is stored in AC0.

AC0 = AC0 | AC1

# **Syntax Characteristics**

No.	Syntax																																							P	_					_	_	it			S	ìi	Ζŧ	е		(	3	у	C	:l	e	39	s		ı	Ρi	ip	Œ	į	ir	ıe	•
[2]	dst = src   k8																																								,	Υ	Έ	Э:	s		_	_		_		3	3						1	I _								>	(		_	
Opcode	e																									l		C	0	C	) (	) (	С	) (	0	1	-		1	0	1	L]	E	2		ŀ	Σ.	k	k	:k	2		}	د]	ςk	ς.	k			]	F	ľ	Ο:	D]	D		]	F	S	S	SS	3
Operan	ds	dst	t, k8	8,	В,	3,	,	, ;	S	SI	rc	С																																																												
Descrip	otion		is ir giste																																					(	O	р	)(	е	ra	ai	ti	c	r	1	b	)€	∍t	W	е	•	er	1	;	а	ì	;	s	0	u	rc	С	Э	(	S	r	С
			WI	۷h	/h	h	16	ıe	е	er	n	tl	he	е	d	des	st	tir	na	ati	ioi	n	(0	ds	t)	)	(	С	0	P	ŗ	p	р	Э	er	ra	ar	n	d	is	; 6	а	ır	n	8	30	)(	С	u	n	าเ	u	la	at	or	۲:																
				ı					-	T	٦h	٦e	9 (	0	р	er	a	ti	io	n i	is	ŗ	е	rf	o	r	r	Υ	r	n	1	16	ıe	e	d	(	OI	n	4	0	k	b	١į	t	S	iı	n	1	th	١E	)	С	)-	·u	ni	it		Α	٠L	_'	ι	J										
				ı					I	lr	nį	рі	ut	t	O	ре	era	a	n	ds	8 6	ar	е	z	е	r	c	0	)	)	•	e	e	ex	Χİ	te	eı	n	de	ЭС	k	t	C	)	4	ŀ(	)	k	ic	ts	3.																					
									i	ir	กร	st	tru	u	ct		n	١,	-	/ o ne				•										-				_																	•							•										
			WI	۷h	/h	h	16	ıe	е	er	n	tl	he	е	d	des	st	tir	na	ati	OI	n	(0	sk	t)	)	C	0	)	P	ŗ	р	р	Э	er	ra	ar	n	d	is	6	а	ır	า	٤	aι	J.	X	il	ia	ır	У	(	OI	t	E	er	n	ıŗ	Э:	С	1(	ra	ar	у	r	е	C	jis	st	ie	۲
				I					-	Т	٦h	٦e	9 (	0	р	er	a	ıti	io	n i	is	ŗ	е	rf	o	r	r	Υ	r	n	1	16	ıe	e	d	(	OI	n	1	6	k	b	١į	t	S	iI	n	1	th	١E	)	Α	۱-	u	ni	it	1	Ą	١L	_!	L	J										
																				ıla he																				•				•																										1	th	16

Affected by **Status Bits** 

> Affects none

none

Repeat This instruction can be repeated.

Syntax	Description
AC0 = AC1   #FFh	The content of AC1 is ORed with the unsigned 8-bit value (FFh) and the result is stored in AC0.

# **Syntax Characteristics**

No. Syntax		Parallel Enable Bit	Size	Cycles	Pipeline
[3] dst = src   k16	3	No	4	1	Х
Opcode	0111   1110   kkkk	kkkk   kkł	ck kk	kk FDI	D FSSS
Operands	dst, k16, src				
Description	This instruction performs a bitwise or register content and a 16-bit unsigned	•		een a so	ource (src)
	☐ When the destination (dst) opera	nd is an acc	umulat	or:	
	■ The operation is performed o	n 40 bits in th	ne D-u	nit ALU.	
	Input operands are zero exte	nded to 40 b	its.		
	If an auxiliary or temporary re instruction, the 16 LSBs of the extended.	-		. , .	
	☐ When the destination (dst) operar	nd is an auxil	iary or	tempora	ry register:
	■ The operation is performed o	n 16 bits in th	ne A-u	nit ALU.	
	<ul><li>If an accumulator is the sour</li><li>16 LSBs of the accumulator a</li></ul>	` , .			-
Status Bits	Affected by none				
	Affects none				
Repeat	This instruction can be repeated.				
Example					
Syntax	Description				
AC0 = AC1   #FFFFh	The content of AC1 is ORed with the uns is stored in AC0.	signed 16-bit v	alue (F	FFFh) and	the result

## **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	dst = src   Smem	No	3	1	X

## Opcode

1101 1010 AAAA AAAI FDDD FSSS

#### **Operands**

dst, Smem, src

#### Description

This instruction performs a bitwise OR operation between a source (src) register content and a memory (Smem) location.

- ☐ When the destination (dst) operand is an accumulator:
  - The operation is performed on 40 bits in the D-unit ALU.
  - Input operands are zero extended to 40 bits.
  - If an auxiliary or temporary register is the source (src) operand of the instruction, the 16 LSBs of the auxiliary or temporary register are zero extended.
- ☐ When the destination (dst) operand is an auxiliary or temporary register:
  - The operation is performed on 16 bits in the A-unit ALU.
  - If an accumulator is the source (src) operand of the instruction, the 16 LSBs of the accumulator are used to perform the operation.

#### **Status Bits**

Affected by none

Affects none

## Repeat

This instruction can be repeated.

Syntax	Description
' '	The content of AC1 is ORed with the content addressed by AR3 and the result is stored in AC0.

# **Syntax Characteristics**

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline	
[5]	ACy = ACy   (A	Cx <	<< #SHIFTW)			Yes	3	1	Х	
Opcod	e				0001	000E DDS	SS 00	01 xxS	H IFTW	
Operar	nds	AC	ACx, ACy, SHIFTW							
Descri	ption	This instruction performs a bitwise OR operation between an accumulator (ACy) content and and an accumulator (ACx) content shifted by the 6-bit value, SHIFTW.								
			☐ The shift and OR operations are performed in one cycle in the D-unit shifter.						the D-unit	
			When M40 = 0 and C54CM = 0, input operands ACx(31-0) are zero extended to 40 bits. Otherwise, ACx(39-0) is used as is.						) are zero	
			The input operand (ACx) is shifted by a 6-bit immediate value in the D-un shifter.						the D-unit	
			The CARR	Y status bit is	not affec	ted by the lo	gical s	hift opera	ation.	
		Compatibility with C54x devices (C54CM = 1)								
				= 1, the interme upper bits of t	, ,				•	
Status	Bits	Affe	ected by	C54CM, M40	)					
		Affe	ects	none						
Repeat	t	Thi	s instruction	can be repea	ted.					

Syntax	Description
	The content of AC1 is ORed with the content of AC0 logically shifted left by 4 bits and the result is stored in AC1.

Before				After			
AC0	7E	2355	4FC0	AC0	7E	2355	4FC0
AC1	0F	E340	5678	AC1	0F	F754	FE78

## **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	ACy = ACx   <b>(</b> k16 <b>&lt;&lt;&lt; #16)</b>	No	4	1	Χ

0111 1010 kkkk kkkk kkkk kkkk SSDD 011x Opcode

**Operands** ACx, ACy, k16

This instruction performs a bitwise OR operation between an accumulator Description

(ACx) content and a 16-bit unsigned constant, k16, shifted left by 16 bits.

☐ The operation is performed on 40 bits in the D-unit ALU.

☐ Input operands are zero extended to 40 bits.

☐ The input operand (k16) is shifted 16 bits to the MSBs.

**Status Bits** Affected by none

> Affects none

Repeat This instruction can be repeated.

Syntax	Description
AC0 = AC1   (#FFFFh <<< #16)	The content of AC1 is ORed with the unsigned 16-bit value (FFFFh)
	logically shifted left by 16 bits and the result is stored in AC0.

# **Syntax Characteristics**

No. Syntax		Parallel Enable Bit	Size	Cycles	Pipeline
[7] ACy = ACx	(k16 <<< #SHFT)	No	4	1	Х
Opcode	0111 0011   kkkk	kkkk   kkk	k kk	kk SSI	D SHFT
Operands	ACx, ACy, k16, SHFT				
Description	This instruction performs a bitwise OI (ACx) content and a 16-bit unsigned value, SHFT.	-			
	The shift and OR operations are shifter.	performed i	n one	cycle in	the D-unit
	☐ When M40 = 0 and C54CM = 0, extended to 40 bits. Otherwise, AC			•	) are zero
	☐ The input operand (k16) is shifted be shifter.	oy a 4-bit imr	mediat	e value ir	the D-unit
	The CARRY status bit is not affect	ed by the lo	gical s	hift opera	ation
Status Bits	Affected by C54CM, M40				
	Affects none				
Repeat	This instruction can be repeated.				

Syntax	Description
	The content of AC1 is ORed with the unsigned 16-bit value (FFFh) logically shifted left by 15 bits and the result is stored in AC0.

# **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	Smem = Smem   k16	No	4	1	Х

| 1111 0101 | AAAA AAAI | kkkk kkkk kkkk kkkk Opcode

k16, Smem **Operands** 

This instruction performs a bitwise OR operation between a memory (Smem) Description

location and a 16-bit unsigned constant, k16.

☐ The operation is performed on 16 bits in the A-unit ALU.

☐ The result is stored in memory.

**Status Bits** Affected by none

> Affects none

Repeat This instruction can be repeated.

# **Example**

Syntax	Description
*AR1 = *AR1   #0FC0h	The content addressed by AR1 is ORed with the unsigned 16-bit value (FC0h) and the result is stored in the location addressed by AR1.

Before After \*AR1 5678 \*AR1 5FF8

# **XOR**

# Bitwise Exclusive OR (XOR)

# **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = dst ^ src	Yes	2	1	Х
[2]	dst = src ^ k8	Yes	3	1	X
[3]	dst = src ^ k16	No	4	1	X
[4]	dst = src ^ Smem	No	3	1	X
[5]	ACy = ACy ^ (ACx <<< #SHIFTW)	Yes	3	1	Х
[6]	$ACy = ACx ^ (k16 <<< #16)$	No	4	1	Х
[7]	$ACy = ACx ^ (k16 <<< \#SHFT)$	No	4	1	Х
[8]	Smem = Smem ^ k16	No	4	1	Х

Description	These instructions perform a bitwise exclusive-OR (XOR) operation:					
	☐ In the D-unit, if the destination operand is an accumulator.					
	In the A-unit ALU, if the destination operand is an auxiliary or temporary register.					
	☐ In the A-unit ALU, if the destination operand is the memory.					
Status Bits	Affected by C54CM					
	Affects none					
See Also	See the following other related instructions:					
	☐ Bitwise AND					
	☐ Bitwise OR					

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = dst ^ src	Yes	2	1	X

**Opcode** 

0010 110E FSSS FDDD

**Operands** 

dst, src

Description

This instruction performs a bitwise exclusive-OR (XOR) operation between two registers.

- ☐ When the destination (dst) operand is an accumulator:
  - The operation is performed on 40 bits in the D-unit ALU.
  - Input operands are zero extended to 40 bits.
  - If an auxiliary or temporary register is the source (src) operand of the instruction, the 16 LSBs of the auxiliary or temporary register are zero extended.
- ☐ When the destination (dst) operand is an auxiliary or temporary register:
  - The operation is performed on 16 bits in the A-unit ALU.
  - If an accumulator is the source (src) operand of the instruction, the 16 LSBs of the accumulator are used to perform the operation.

**Status Bits** 

Affected by none

Affects none

Repeat

This instruction can be repeated.

#### Example

Syntax	Description
AC1 = AC1 ^ AC0	The content of AC0 is XORed with the content of AC1 and the result is stored in AC1.

Before After

AC0 7E 2355 4FC0 AC0 7E 2355 4FC0 AC1 0F E340 5678 AC1 71 C015 19B8

# **Syntax Characteristics**

No. Syntax		Parallel Enable Bit	Size	Cycles	Pipeline
[2] dst = src ^ k8		Yes	3	1	Х
Opcode	0001	110E   kkk	k kk	kk FDD	D FSSS
Operands	dst, k8, src				
<b>Description</b> This instruction performs a bitwise exclusive-OR (XOR) op source (src) register content and an 8-bit value, k8.					between a
	☐ When the destination (dst) operar	nd is an accu	mulato	or:	
	■ The operation is performed or	n 40 bits in th	ne D-ui	nit ALU.	
	Input operands are zero exter	nded to 40 bi	ts.		
	If an auxiliary or temporary reginstruction, the 16 LSBs of the extended.	•		· , .	
	☐ When the destination (dst) operan	ıd is an auxili	iary or	tempora	ry register:
	■ The operation is performed or	n 16 bits in th	ne A-uı	nit ALU.	
	<ul><li>If an accumulator is the source</li><li>16 LSBs of the accumulator a</li></ul>				
Status Bits	Affected by none				
	Affects none				
Repeat	This instruction can be repeated.				
Example					
Syntax	Description				
AC0 = AC1 ^ #FFh	The content of AC1 is XORed with the unstored in AC0.	signed 8-bit v	alue (F	Fh) and th	ne result is

#### **Syntax Characteristics**

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[3]	dst = src ^ k16				No	4	1	Χ
Opcode Operan		dst, k16, src	0111	1111   kkkk	kkkk kkk	k kk	kk   FDD	D FSSS
Descrip	otion	This instruction performs a bitwise exclusive-OR (XOR) operation between source (src) register content and a 16-bit unsigned constant, k16.				between a		

- ☐ When the destination (dst) operand is an accumulator:
  - The operation is performed on 40 bits in the D-unit ALU.
  - Input operands are zero extended to 40 bits.
  - If an auxiliary or temporary register is the source (src) operand of the instruction, the 16 LSBs of the auxiliary or temporary register are zero extended.
- ☐ When the destination (dst) operand is an auxiliary or temporary register:
  - The operation is performed on 16 bits in the A-unit ALU.
  - If an accumulator is the source (src) operand of the instruction, the 16 LSBs of the accumulator are used to perform the operation.

**Status Bits** Affected by none

> Affects none

Repeat This instruction can be repeated.

Syntax	Description
AC0 = AC1 ^ #FFFFh	The content of AC1 is XORed with the unsigned 16-bit value (FFFFh) and the result is stored in AC0.

# **Syntax Characteristics**

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline
[4]	dst = src ^ Sme	m				No	3	1	X
Opcode	e				1101	1011 AAA	A AA	AI   FDD	D FSSS
Operan	nds	dst, Smem, src							
-			struction performs a bi		•		•	between a	
			Wh	en the destination (de	st) operar	nd is an accu	mulato	or:	
				The operation is per	formed o	n 40 bits in th	ne D-ui	nit ALU.	
				Input operands are z	ero exte	nded to 40 bi	ts.		
				If an auxiliary or tempostruction, the 16 LS extended.	•	_		. , .	
			Wh	en the destination (de	st) operar	nd is an auxil	iary or	tempora	ry register:
				The operation is per	formed o	n 16 bits in th	ne A-uı	nit ALU.	
				If an accumulator is 16 LSBs of the accu		. , .			

**Status Bits** 

Affected by none

Affects none

Repeat

This instruction can be repeated.

Syntax	Description
AC0 = AC1 ^ *AR3	The content of AC1 is XORed with the content addressed by AR3 and the result is stored in AC0.

# **Syntax Characteristics**

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline
[5]	$ACy = ACy ^ (A$	\Сх <b>&lt;</b> <	<< #SHIFTW	)		Yes	3	1	Х
Opcod	e				0001	000E DDS	SS 00	10 xxS	H IFTW
Operar	nds	ACx, ACy, SHIFTW							
Descri	ption	This instruction performs a bitwise exclusive-OR (XOR) operation betw accumulator (ACy) content and an accumulator (ACx) content shifted 6-bit value, SHIFTW.							
			The shift a shifter.	nd XOR opera	tions are	e performed	in one	cycle in	the D-unit
				0 = 0 and C54 0 40 bits. Othe				•	) are zero
			The input on shifter.	perand (ACx) is	s shifted	by a 6-bit im	mediat	e value ir	the D-unit
			The CARR	Y status bit is r	not affec	ted by the lo	gical s	hift opera	ition.
		Cor	mpatibility	with C54x dev	vices (C	54CM = 1)			
				= 1, the interme upper bits of th					•
Status	Bits	Affe	cted by	C54CM, M40					
		Affe	ects	none					

# Example

Repeat

Syntax	Description
AC0 = AC0 ^ (AC1 <<< #30)	The content of AC0 is XORed with the content of AC1 logically shifted left by 30 bits and the result is stored in AC0.

This instruction can be repeated.

## **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	ACy = ACx ^ <b>(</b> k16 <b>&lt;&lt;&lt; #16)</b>	No	4	1	Х

0111 1010 kkkk kkkk kkkk kkkk SSDD 100x **Opcode** 

**Operands** ACx, ACy, k16

**Description** This instruction performs a bitwise exclusive-OR (XOR) operation between an

accumulator (ACx) content and a 16-bit unsigned constant, k16, shifted left by

16 bits.

☐ The operation is performed on 40 bits in the D-unit ALU.

☐ Input operands are zero extended to 40 bits.

☐ The input operand (k16) is shifted 16 bits to the MSBs.

**Status Bits** Affected by none

> Affects none

Repeat This instruction can be repeated.

Syntax	Description
AC0 = AC1 ^ (#FFFFh <<< #16)	The content of AC1 is XORed with the unsigned 16-bit value (FFFFh)
	logically shifted left by 16 bits and the result is stored in AC0.

## **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	ACy = ACx ^ (k16 <<< #SHFT)	No	4	1	Х

0111 0100 kkkk kkkk kkkk kkkk SSDD SHFT Opcode

**Operands** ACx, ACy, k16, SHFT

Description This instruction performs a bitwise exclusive-OR (XOR) operation between an accumulator (ACx) content and a 16-bit unsigned constant, k16, shifted left by

the 4-bit value, SHFT.

☐ The shift and XOR operations are performed in one cycle in the D-unit shifter.

 $\square$  When M40 = 0 and C54CM = 0, input operands ACx(31-0) are zero extended to 40 bits. Otherwise, ACx(39-0) is used as is.

The input operand (k16) is shifted by a 4-bit immediate value in the D-unit shifter.

☐ The CARRY status bit is not affected by the logical shift operation.

**Status Bits** Affected by C54CM, M40

> Affects none

Repeat This instruction can be repeated.

Syntax	Description
AC0 = AC1 ^ (#FFFFh <<< #15)	The content of AC1 is XORed with the unsigned 16-bit value (FFFFh) logically shifted left by 15 bits and the result is stored in AC0.

## **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	Smem = Smem ^ k16	No	4	1	Х

| 1111 0110 | AAAA AAAI | kkkk kkkk kkkk kkkk Opcode

**Operands** k16, Smem

This instruction performs a bitwise exclusive-OR (XOR) operation between a Description

memory (Smem) location and a 16-bit unsigned constant, k16.

☐ The operation is performed on 16 bits in the A-unit ALU.

☐ The result is stored in memory.

**Status Bits** Affected by none

> Affects none

Repeat This instruction can be repeated.

Syntax	Description
*AR3 = *AR3 ^ #FFFFh	The content addressed by AR3 is XORed with the unsigned 16-bit value (FFFh) and the result is stored in the location addressed by AR3.

## BCC

#### Branch Conditionally

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles <sup>†</sup>	Pipeline
[1]	if (cond) goto  4	No	2	6/5	R
[2]	if (cond) goto L8	Yes	3	6/5	R
[3]	if (cond) goto L16	No	4	6/5	R
[4]	if (cond) goto P24	No	5	5/5	R

<sup>†</sup> x/y cycles: x cycles = condition true, y cycles = condition false

#### Description

These instructions evaluate a single condition defined by the cond field in the read phase of the pipeline. If the condition is true, a branch occurs to the program address label assembled into I4, Lx, or P24. There is a 1-cycle latency on the condition setting. A single condition can be tested as determined by the cond field of the instruction. See Table 1–3 for a list of conditions.

The instruction selection depends on the branch offset between the current PC value and the program branch address specified by the label.

These instructions cannot be repeated.

Status Bits Affected by ACOVx, CARRY, C54CM, M40, TCx

Affects ACOVx

**See Also** See the following other related instructions:

- Branch Unconditionally
- □ Branch on Auxiliary Register Not Zero
- Call Conditionally
- Compare and Branch

## Branch Conditionally

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles†	Pipeline
[1]	if (cond) goto 14	No	2	6/5	R

<sup>†</sup> x/y cycles: x cycles = condition true, y cycles = condition false

Opcode 0110 0111 1ccc cccc

Operands cond, I4

**Description** This instruction evaluates a single condition defined by the cond field in the

read phase of the pipeline. If the condition is true, a branch occurs to the program address label assembled into I4. There is a 1-cycle latency on the condition setting. A single condition can be tested as determined by the cond

field of the instruction. See Table 1–3 for a list of conditions.

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, the comparison of accumulators to 0 is performed as if M40

was set to 1.

Status Bits Affected by ACOVx, CARRY, C54CM, M40, TCx

Affects ACOVx

**Repeat** This instruction cannot be repeated.

Syntax	Description
if (AC0 != #0) goto branch	The content of AC0 is not equal to 0, control is passed to the program address
	label defined by branch.

	if (AC0 != #0) goto branch		
	addres	ss:	004057
branch			00405A
1:			

Before		After	
AC0	00 0000 3000	AC0	00 0000 3000
PC	004055	PC	00405A

#### Branch Conditionally

## **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles†	Pipeline
[2]	if (cond) goto L8	Yes	3	6/5	R
[3]	if (cond) goto L16	No	4	6/5	R

<sup>†</sup> x/y cycles: x cycles = condition true, y cycles = condition false

Opcode L8 0000 010E xCCC CCCC LLLL LLLL

L16 0110 1101 xCCC CCCC LLLL LLLL LLLL

Operands cond, Lx

**Description** This instruction evaluates a single condition defined by the cond field in the

read phase of the pipeline. If the condition is true, a branch occurs to the program address label assembled into Lx. There is a 1-cycle latency on the condition setting. A single condition can be tested as determined by the cond

field of the instruction. See Table 1-3 for a list of conditions.

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, the comparison of accumulators to 0 is performed as if M40

was set to 1.

Status Bits Affected by ACOVx, CARRY, C54CM, M40, TCx

Affects ACOVx

**Repeat** This instruction cannot be repeated.

#### **Example**

Syntax	Description
if (AC0 != #0) goto branch	The content of AC0 is not equal to 0, control is passed to the program address label defined by branch.

branch :			00305A
	if (AC0 != #0) goto branch		
		address:	004057

Before After

ACO 00 0000 3000 ACO 00 0000 3000 PC 004055 PC 00305A

## Branch Conditionally

## **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles <sup>†</sup>	Pipeline
[4]	if (cond) goto P24	No	5	5/5	R

<sup>†</sup> x/y cycles: x cycles = condition true, y cycles = condition false

Operands cond, P24

Description

This instruction evaluates a single condition defined by the cond field in the read phase of the pipeline. If the condition is true, a branch occurs to the program address label assembled into P24. There is a 1-cycle latency on the condition setting. A single condition can be tested as determined by the cond field of the instruction. See Table 1–3 for a list of conditions.

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, the comparison of accumulators to 0 is performed as if M40

was set to 1.

Status Bits Affected by ACOVx, CARRY, C54CM, M40, TCx

Affects ACOVx

**Repeat** This instruction cannot be repeated.

Syntax	Description
if (AC0 != #0) goto branch	The content of AC0 is not equal to 0, control is passed to the program address label defined by branch.

	.sect "code1"		
	if (AC0 != #0) goto branch		
		address:	004057
	.sect "code2"		
branch			00F05A
:			

Before		After	
AC0	00 0000 3000	AC0	00 0000 3000
PC	004055	PC	00F05A

# В

## Branch Unconditionally

## **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	goto ACx	No	2	10	Х
[2]	goto L7	Yes	2	6†	AD
[3]	goto L16	Yes	3	6 <sup>†</sup>	AD
[4]	goto P24	No	4	5	D

<sup>&</sup>lt;sup>†</sup> This instruction executes in 3 cycles if the addressed instruction is in the instruction buffer unit.

**Description** This instruction branches to a 24-bit program address defined by the content

of the 24 lowest bits of an accumulator (ACx), or to a program address defined

by the program address label assembled into Lx or P24.

These instructions cannot be repeated.

Status Bits Affected by none

Affects none

**See Also** See the following other related instructions:

- Branch Conditionally
- □ Branch on Auxiliary Register Not Zero
- Call Unconditionally
- Compare and Branch

# Branch Unconditionally

# **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	goto ACx	No	2	10	Х

1001 0001 xxxx xxSS Opcode

**Operands** ACx

**Description** This instruction branches to a 24-bit program address defined by the content

of the 24 lowest bits of an accumulator (ACx).

**Status Bits** Affected by none

> Affects none

Repeat This instruction cannot be repeated.

# **Example**

Syntax	Description
goto AC0	Program control is passed to the program address defined by the content of AC0(23–0).

Before After AC0 00 0000 403D AC0 00 0000 403D PC 001F0A PC 00403D

## Branch Unconditionally

## **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles†	Pipeline
[2]	goto L7	Yes	2	6	AD
[3]	goto L16	Yes	3	6	AD

<sup>†</sup> Executes in 3 cycles if the addressed instruction is in the instruction buffer unit.

0100 101E OLLL LLLL Opcode L7

> 0000 011E LLLL LLLL LLLL L16

**Operands** Lx

**Description** This instruction branches to a program address defined by a program address

label assembled into Lx.

**Status Bits** Affected by none

> Affects none

Repeat This instruction cannot be repeated.

#### **Example**

Syntax	Description
goto branch	Program control is passed to the absolute address defined by branch.

goto branch AC0 = #1address: 004044 006047 branch: ... ... AC0 = #0

Before After

004042 РC PC 006047 AC0 00 0000 0001 AC0 00 0000 0000

# Branch Unconditionally

# **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	goto P24	No	4	5	D

0110 1010 PPPP PPPP PPPP PPPP PPPP Opcode

**Operands** P24

**Description** This instruction branches to a program address defined by a program address

label assembled into P24.

**Status Bits** Affected by none

> Affects none

Repeat This instruction cannot be repeated.

## **Example**

Syntax	Description
goto branch	Program control is passed to the absolute address defined by branch.

goto branch AC0 = #1 address: 004044 branch: 006047 ... ... AC0 = #0

Before After

PC 004042 PC 006047 00 0000 0001 AC0 00 0000 0000 AC0

# BCC

#### Branch on Auxiliary Register Not Zero

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles†	Pipeline
[1]	<b>if (</b> ARn_mod <b>!= #0) goto</b> L16	No	4	6/5	AD

<sup>†</sup> x/y cycles: x cycles = condition true, y cycles = condition false

#### Opcode

| 1111 1100 | AAAA AAAI | LLLL | LLLL | LLLL

#### **Operands**

ARn\_mod, L16

#### Description

This instruction performs a conditional branch (selected auxiliary register content not equal to 0) of the program counter (PC). The program branch address is specified as a 16-bit signed offset, L16, relative to PC. Use this instruction to branch within a 64K-byte window centered on the current PC value.

The possible addressing operands can be grouped into three categories:

- ARx not modified (ARx as base pointer), some examples:
  - \*AR1; No modification or offset
  - \*AR1(#15); Use 16-bit immediate value (15) as offset
  - \*AR1(T0); Use content of T0 as offset
  - \*AR1(short(#4)); Use 3-bit immediate value (4) as offset
- ☐ ARx modified before being compared to 0, some examples:
  - \*-AR1; Decrement by 1 before comparison
  - \*+AR1(#20); Add 16-bit immediate value (20) before comparison
- ARx modified after being compared to 0, some examples:
  - \*AR1+; Increment by 1 after comparison
  - \*(AR1 T1); Subtract content of T1 after comparison
- 1) The content of the selected auxiliary register (ARn) is premodified in the address generation unit.
- 2) The (premodified) content of ARn is compared to 0 and sets the condition in the address phase of the pipeline.
- 3) If the condition is not true, a branch occurs. If the condition is true, the instructions are executed in sequence.
- 4) The content of ARn is postmodified in the address generation unit.

# Compatibility with C54x devices (C54CM = 1)

When C54CM = 1:

The premodifier \*ARn(T0) is not available; \*ARn(AR0) is available.

The postmodifiers \*(ARn + T0) and \*(ARn - T0) are not available; \*(ARn + AR0) and \*(ARn - AR0) are available.

The legality of the modifier usage is checked by the assembler when using the .c54cm\_on and .c54cm\_off assembler directives.

Status Bits Affected by C54CM

Affects none

**Repeat** This instruction cannot be repeated.

**See Also** See the following other related instructions:

Branch Conditionally

□ Branch Unconditionally

Compare and Branch

Syntax	Description
if (*AR1(#6) != #0) goto branch	The content of AR1 is compared to 0. The content is not 0, program control is passed to the program address label defined by branch.

	If (*AR1(#6) != #0) goto branch	address:	004004
		;	00400A
branch		;	00400C
:			

Before		After		
AR1	0005	AR1	0005	
PC.	004004	PC	004000	

Syntax	Description
if (*AR3- != #0) goto branch	The content of AR3 is compared to 0. The content is 0, program control is passed to the next instruction (the branch is not taken). AR3 is decremented
	by 1 after the comparison.

	If (*AR3-!=#0) goto branch	address:	00400F
		;	004013
branch		;	004015
branch :		;	004015

Before		After	
AR3	0000	AR3	FFFF
PC	00400F	PC	004013

#### CALLCC

#### Call Conditionally

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles <sup>†</sup>	Pipeline
[1]	if (cond) call L16	No	4	6/5	R
[2]	if (cond) call P24	No	5	5/5	R

 $<sup>^{\</sup>dagger}$  x/y cycles: x cycles = condition true, y cycles = condition false

#### Description

These instructions evaluate a single condition defined by the cond field in the read phase of the pipeline. If the condition is true, a subroutine call occurs to the program address defined by the program address label assembled into L16 or P24. There is a 1-cycle latency on the condition setting. A single condition can be tested as determined by the cond field of the instruction. See Table 1–3 for a list of conditions.

Before beginning a called subroutine, the CPU automatically saves the value of two internal registers: the program counter (PC) and a loop context register. The CPU can use these values to re-establish the context of the interrupted program sequence when the subroutine is done.

In the slow-return process (default), the return address (from the PC) and the loop context bits are stored to the stacks (in memory). When the CPU returns from a subroutine, the speed at which these values are restored is dependent on the speed of the memory accesses.

In the fast-return process, the return address (from the PC) and the loop context bits are saved to registers, so that these values can always be restored quickly. These special registers are the return address register (RETA) and the control-flow context register (CFCT). You can read from or write to RETA and CFCT as a pair with dedicated, 32-bit load and store instructions.

The instruction selection depends on the branch offset between the current PC value and program subroutine address specified by the label.

These instructions cannot be repeated.

Status Bits Affected by ACOVx, CARRY, C54CM, M40, TCx

Affects ACOVx

See Also	See the following other related instructions:
	☐ Branch Conditionally
	☐ Call Unconditionally
	☐ Return Conditionally
	☐ Return Unconditionally

#### Call Conditionally

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles <sup>†</sup>	Pipeline
[1]	if (cond) call L16	No	4	6/5	R

<sup>†</sup> x/y cycles: x cycles = condition true, y cycles = condition false

#### Opcode

0110 1110 xCCC CCCC LLLL LLLL LLLL

#### **Operands**

cond, L16

#### Description

This instruction evaluates a single condition defined by the cond field in the read phase of the pipeline. If the condition is true, a subroutine call occurs to the program address defined by the program address label assembled into L16. There is a 1-cycle latency on the condition setting. A single condition can be tested as determined by the cond field of the instruction. See Table 1–3 for a list of conditions.

When a subroutine call occurs in the slow-return process (default), the return address (from the PC) and the loop context bits are stored to the stacks. For fast-return mode operation, see the *TMS320C55x DSP CPU Reference Guide* (SPRU371).

- ☐ The data stack pointer (SP) is decremented by 1 word in the read phase of the pipeline. The 16 LSBs of the return address, from the program counter (PC), of the called subroutine are pushed to the top of SP.
- ☐ The system stack pointer (SSP) is decremented by 1 word in the read phase of the pipeline. The loop context bits concatenated with the 8 MSBs of the return address are pushed to the top of SSP.
- The PC is loaded with the subroutine program address. The active control flow execution context flags are cleared.

After Save 
$$\rightarrow$$
 SSP = x - 1 (Loop bits):PC(23-16)

Before Save  $\rightarrow$  SSP = x Previously saved data

After Save 
$$\rightarrow$$
 SP = y - 1  $\rightarrow$  PC(15-0)

Before Save  $\rightarrow$  SP = y  $\rightarrow$  Previously saved data

## Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, the comparison of accumulators to 0 is performed as if M40 was set to 1.

# CALLCC Call Conditionally (if call)

Affected by ACOVx, CARRY, C54CM, M40, TCx **Status Bits** 

> ACOVx Affects

This instruction cannot be repeated. Repeat

Syntax	Description
, , ,	The content of AC1 is equal to or greater than 2000h, control is passed to the program address label, subroutine. The program counter (PC) is loaded with the subroutine program address.

#### Call Conditionally

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles <sup>†</sup>	Pipeline
[2]	if (cond) call P24	No	5	5/5	R

<sup>†</sup> x/y cycles: x cycles = condition true, y cycles = condition false

#### **Opcode**

# Operands Description

cond, P24

oona, i

This instruction evaluates a single condition defined by the cond field in the read phase of the pipeline. If the condition is true, a subroutine call occurs to the program address defined by the program address label assembled into P24. There is a 1-cycle latency on the condition setting. A single condition can be tested as determined by the cond field of the instruction. See Table 1–3 for a list of conditions.

When a subroutine call occurs in the slow-return process (default), the return address (from the PC) and the loop context bits are stored to the stacks. For fast-return mode operation, see the *TMS320C55x DSP CPU Reference Guide* (SPRU371).

- ☐ The data stack pointer (SP) is decremented by 1 word in the read phase of the pipeline. The 16 LSBs of the return address, from the program counter (PC), of the called subroutine are pushed to the top of SP.
- ☐ The system stack pointer (SSP) is decremented by 1 word in the read phase of the pipeline. The loop context bits concatenated with the 8 MSBs of the return address are pushed to the top of SSP.
- The PC is loaded with the subroutine program address. The active control flow execution context flags are cleared.

After Save 
$$\rightarrow$$
 SSP = x - 1 (Loop bits):PC(23-16)

Before Save  $\rightarrow$  SSP = x Previously saved data

After Save 
$$\rightarrow$$
 SP = y - 1  $PC(15-0)$ 

Before Save  $\rightarrow$  SP = y Previously saved data

#### Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, the comparison of accumulators to 0 is performed as if M40 was set to 1.

# CALLCC Call Conditionally (if call)

Affected by ACOVx, CARRY, C54CM, M40, TCx **Status Bits** 

> ACOVx Affects

Repeat This instruction cannot be repeated.

Syntax	Description
if (TC1) call FOO	If TC1 is set to 1, control is passed to the program address label (FOO) assembled into an absolute address defined by the 24-bit value. If TC1 is cleared to 0, the program counter is incremented by 6 and the next instruction is executed.

#### CALL

#### Call Unconditionally

#### **Syntax Characteristics**

		Parallel			
No.	Syntax	Enable Bit	Size	Cycles	Pipeline
[1]	call ACx	No	2	10	Х
[2]	call L16	Yes	3	6	AD
[3]	call P24	No	4	5	D

#### Description

This instruction passes control to a specified subroutine program address defined by the content of the 24 lowest bits of the accumulator, ACx, or a program address label assembled into L16 or P24.

Before beginning a called subroutine, the CPU automatically saves the value of two internal registers: the program counter (PC) and a loop context register. The CPU can use these values to re-establish the context of the interrupted program sequence when the subroutine is done.

In the slow-return process (default), the return address (from the PC) and the loop context bits are stored to the stacks (in memory). When the CPU returns from a subroutine, the speed at which these values are restored is dependent on the speed of the memory accesses.

In the fast-return process, the return address (from the PC) and the loop context bits are saved to registers, so that these values can always be restored quickly. These special registers are the return address register (RETA) and the control-flow context register (CFCT). You can read from or write to RETA and CFCT as a pair with dedicated, 32-bit load and store instructions.

These instructions cannot be repeated.

Status Bits	Affected by	none
	Affects	none
See Also	See the following	ng other related instructions:
	☐ Branch Und	conditionally
	☐ Call Condit	ionally
	☐ Return Cor	nditionally
	☐ Return Und	conditionally

#### Call Unconditionally

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	call ACx	No	2	10	Χ

#### **Opcode**

1001 0010 xxxx xxSS

**Operands** 

ACx

**Description** 

This instruction passes control to a specified subroutine program address defined by the content of the 24 lowest bits of the accumulator, ACx.

In the slow-return process (default), the return address (from the PC) and the loop context bits are stored to the stacks. For fast-return mode operation, see the *TMS320C55x DSP CPU Reference Guide* (SPRU371).

- ☐ The data stack pointer (SP) is decremented by 1 word in the address phase of the pipeline. The 16 LSBs of the return address, from the program counter (PC), of the called subroutine are pushed to the top of SP.
- ☐ The system stack pointer (SSP) is decremented by 1 word in the address phase of the pipeline. The loop context bits concatenated with the 8 MSBs of the return address are pushed to the top of SSP.
- ☐ The PC is loaded with the subroutine program address. The active control flow execution context flags are cleared.

#### System Stack (SSP)

After Save 
$$\rightarrow$$
 SSP = x - 1 (Loop bits):PC(23-16)

Before Save  $\rightarrow$  SSP = x Previously saved data

# **After** Save → SP = y - 1 **Before** SP = y - 1

Data Stack (SP)

**Status Bits** 

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

Syntax	Description
call AC0	Program control is passed to the program address defined by the content of AC0(23–0).

#### Call Unconditionally

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	call L16	Yes	3	6	AD

#### **Opcode**

0000 100E LLLL LLLL LLLL

#### **Operands**

L16

#### Description

This instruction passes control to a specified subroutine program address defined by a program address label assembled into L16.

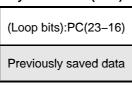
In the slow-return process (default), the return address (from the PC) and the loop context bits are stored to the stacks. For fast-return mode operation, see the TMS320C55x DSP CPU Reference Guide (SPRU371).

- ☐ The data stack pointer (SP) is decremented by 1 word in the address phase of the pipeline. The 16 LSBs of the return address, from the program counter (PC), of the called subroutine are pushed to the top of SP.
- ☐ The system stack pointer (SSP) is decremented by 1 word in the address phase of the pipeline. The loop context bits concatenated with the 8 MSBs of the return address are pushed to the top of SSP.
- The PC is loaded with the subroutine program address. The active control flow execution context flags are cleared.

Data Stack (SP)

After Save 
$$\rightarrow$$
 SSP = x - 1

Before Save  $\rightarrow$  SSP = x



After Save 
$$\rightarrow$$
 SP = y - 1  $\rightarrow$  PC(15-0)

Before Save  $\rightarrow$  SP = y  $\rightarrow$  Previously saved data

**Status Bits** 

Affected by none

**Affects** none

Repeat

This instruction cannot be repeated.

Syntax	Description
	Program control is passed to the program address label (FOO) assembled into the signed 16-bit offset value relative to the program counter register.

# Call Unconditionally

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	call P24	No	4	5	D

Opcode

0110 1100 PPPP PPPP PPPP PPPP PPPP

**Operands** 

P24

**Description** 

This instruction passes control to a specified subroutine program address defined by a program address label assembled into P24.

In the slow-return process (default), the return address (from the PC) and the loop context bits are stored to the stacks. For fast-return mode operation, see the *TMS320C55x DSP CPU Reference Guide* (SPRU371).

- ☐ The data stack pointer (SP) is decremented by 1 word in the address phase of the pipeline. The 16 LSBs of the return address, from the program counter (PC), of the called subroutine are pushed to the top of SP.
- ☐ The system stack pointer (SSP) is decremented by 1 word in the address phase of the pipeline. The loop context bits concatenated with the 8 MSBs of the return address are pushed to the top of SSP.
- ☐ The PC is loaded with the subroutine program address. The active control flow execution context flags are cleared.

#### System Stack (SSP)

$$\begin{array}{ccc} \textbf{After} & \rightarrow & \text{SSP} = x - 1 \\ \textbf{Save} & \rightarrow & \text{SSP} = x \end{array}$$

(Loop bits):PC(23–16)

Previously saved data

After Save → SP = y - 1

Before → SP = y

Data Stack (SP)

1 PC(15–0)

Previously saved data

**Status Bits** 

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

Syntax	Description
call FOO	Program control is passed to the program address label (FOO) assembled into an absolute address defined by the 24-bit value.

# .CR

### Circular Addressing Qualifier

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	circular()	No	1	1	AD

**Opcode** 1001 1101

**Operands** none

**Description** This instruction is an instruction qualifier that can be paralleled only with any

instruction making an indirect Smem, Xmem, Ymem, Lmem, Baddr, or Cmem addressing or mar instructions. This instruction cannot be executed in parallel with any other types of instructions and it cannot be executed as a stand-alone

instruction (assembler generates an error message).

When this instruction is used in parallel, all modifications of ARx and CDP pointer registers used in the indirect addressing mode are done circularly (as

if ST2\_55 register bits 0 to 8 were set to 1).

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction does not affect \*ARn, \*ARn+, \*ARn-, and \*ARn(DR0) addressing modes of dual memory access (Xmem/Ymem)

instruction.

Status Bits Affected by none

Affects none

**Repeat** This instruction can be repeated.

# **BCLR**

Clear Accumulator, Auxiliary, or Temporary Register Bit

#### **Syntax Characteristics**

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline
[1]	bit(src, Baddr) =	#0				No	3	1	Х
Opcod	e				1110	1100 AA	AA A	AAI   FSS	S 001x
Operar	nds	Bad	ddr, src						
Descri	ption	Thi	s instruction	performs a bit	manipu	ulation:			
			In the D-uni	t ALU, if the so	urce (s	rc) register o	peran	d is an ac	cumulator.
			In the A-un temporary r	it ALU, if the so	ource (	src) register	opera	nd is an a	auxiliary or
				clears to 0 a sin ource register.	gle bit,	as defined b	by the b	oit address	sing mode,
		The	generated	bit address mu	st be w	ithin:			
			bit address	accessing accu are used to de not within 0–39,	etermin	e the bit pos	sition).	If the ger	nerated bit
				accessing auxil	•		•	` •	
Status	Bits	Affe	ected by	none					
		Affe	ects	none					
Repeat	t	Thi	s instruction	can be repeate	ed.				
See Als	so	See	e the followir	ng other related	instruc	ctions:			
			Clear Memo	ory Bit					
			Clear Statu	s Register Bit					
			Complemen	nt Accumulator,	Auxilia	ary, or Temp	orary F	Register B	it
			Set Accumi	ulator, Auxiliary	or Ter	nporary Reg	ister B	it	
Examp	ole								

Syntax	Description
bit(AC0, AR3) = #0	The bit at the position defined by the content of AR3(4–0) in AC0 is cleared to 0.

**BCLR** 

Clear Memory Bit

☐ Set Memory Bit

# **Syntax Characteristics**

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[1]	bit(Smem, src)	= #0		No	3	1	Х
Opcod	e		1110	0011 AA	AA AZ	AAI FSS	SS 1101
Operar	nds	Smem, src					
Description		This instruction performs a bit manipulation in the A-unit ALU. The instruction clears to 0 a single bit, as defined by the content of the source (src) operand, of a memory (Smem) location.					
		The generated bit add are used to determine		•	ly the 4	LSBs of	the register
Status	Bits	Affected by none					
		Affects none					
Repeat	<u>:</u>	This instruction can be	e repeated.				
See Als	so	See the following other	er related instruc	ctions:			
		☐ Clear Accumulato	r, Auxiliary, or T	emporary R	egister	Bit	
		☐ Clear Status Regi	ster Bit				
		☐ Complement Men	nory Bit				

Syntax	Description
bit(*AR3, AC0) = #0	The bit at the position defined by AC0(3–0) in the content addressed by AR3 is cleared to 0.

#### **BCLR**

#### Clear Status Register Bit

#### **Syntax Characteristics**

		Parallel			
No.	Syntax	Enable Bit	Size	Cycles	Pipeline
[1]	bit(ST0, k4) = #0	Yes	2	1	Х
[2]	bit(ST1, k4) = #0	Yes	2	1	X
[3]	bit(ST2, k4) = #0	Yes	2	1	X
[4]	bit(ST3, k4) = #0	Yes	2	1†	Х

<sup>&</sup>lt;sup>†</sup> When this instruction is decoded to modify status bit CAFRZ (15), CAEN (14), or CACLR (13), the CPU pipeline is flushed and the instruction is executed in 5 cycles regardless of the instruction context.

Opcode	ST0	0100	011E kkkk	0000
	ST1	0100	011E   kkkk	0010
	ST2	0100	011E   kkkk	0100
	ST3	0100	011E kkkk	0110

#### Operands

k4, STx

#### **Description**

These instructions perform a bit manipulation in the A-unit ALU.

These instructions clear to 0 a single bit, as defined by a 4-bit immediate value, k4, in the selected status register (ST0, ST1, ST2, or ST3).

It is not allowed to access DP register mapped in ST0 register with bit(ST0, k4) = #0 instruction. Therefore, k4 cannot have a value of 0–8.

It is not allowed to access ASM bit field in ST1 with bit(ST1, k4) = #0 instruction. Therefore, k4 cannot have a value of 0–4.

#### Compatibility with C54x devices (C54CM = 1)

C55x DSP status registers bit mapping (Figure 5–1, page 5-92) does not correspond to C54x DSP status register bits.

#### **Status Bits**

Affected by none

Affects Selected status bits

#### Repeat

This instruction cannot be repeated.

See Also	See the following other related instructions:
	☐ Clear Accumulator, Auxiliary, or Temporary Register Bit
	☐ Clear Memory Bit
	☐ Set Status Register Bit

Syntax	Description
bit(ST2, #ST2_AR2LC) = #0; AR2LC = bit 2	The ST2 bit position defined by the label (ST2_AR2LC, bit 2)
	is cleared to 0.

Before	After		
ST2 55	0006	ST2 55	0002

Figure 5-1. Status Registers Bit Mapping

ST0	55
-----	----

15	14	13		12		11	10	9
ACOV2†	ACOV3†	TC1	t	TC2	C	ARRY	ACOV0	ACOV1
R/W-0	R/W-0	R/W-	-1	R/W-1	R	/W-1	R/W-0	R/W-0
8								0
				DP				
				R/W-0				
ST1_55								
15	14	13	1	2	11	10	9	8
BRAF	CPL	XF	Н	M I	NTM	M40 <sup>†</sup>	SATD	SXMD
R/W-0	R/W-0	R/W-1	R/V	V-0 R	2/W-1	R/W-0	R/W-0	R/W-1
7	6	5	4					0
C16	FRCT	C54CM <sup>†</sup>				ASM		
R/W-0	R/W-0	R/W-1				R/W-0		
ST2_55								
15	14	13	1	2	11	10	9	8
ARMS	Rese	erved	DB	GM EA	LLOW	RDM	Reserved	CDPLC
R/W-0			R/V	V–1 R	2/W-0	R/W-0		R/W-0
7	6	5		4	3	2	1	0
AR7LC	AR6LC	AR5LC	AR	4LC A	R3LC	AR2LC	AR1LC	AR0LC
R/W-0	R/W-0	R/W-0	R/V	V-0 R	2/W-0	R/W-0	R/W-0	R/W-0
ST3_55								
15	14	13	1	2 11				8
CAFRZ†	CAEN†	CACLR†	HII	NT‡	R	eserved (alw	ays write 1100	Ob)
R/W-0	R/W-0	R/W-0	R/V	V-1				
7	6	5	4		3	2	1	0
CBERR†	MPNMC§	SATA <sup>†</sup>		Reserved		CLKOFF	SMUL	SST
R/W-0	R/W-pins	R/W-0				R/W-0	R/W-0	R/W-0

**Legend:** R = Read; W = Write; -n = Value after reset

<sup>&</sup>lt;sup>†</sup> Highlighted bit: If you write to the protected address of the status register, a write to this bit has no effect, and the bit always appears as a 0 during read operations.

<sup>&</sup>lt;sup>‡</sup> The HINT bit is not used for all C55x host port interfaces (HPIs). Consult the documentation for the specific C55x DSP.

<sup>§</sup> The reset value of MPNMC may be dependent on the state of predefined pins at reset. To check this for a particular C55x DSP, see the boot loader section of its data sheet.

#### **CMP**

#### Compare Accumulator, Auxiliary, or Temporary Register Content

#### **Syntax Characteristics**

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline
[1]	TC1 = uns(src RELOP dst)		Yes	3	1	Х
[2]	TC2 = uns(src RELOP dst)		Yes	3	1	Х
Opcode	TC1	0001	001E FS	SS co	00 FDD	D xux0
	TC2	0001	001E FS	SS co	00 FDD	DD xux1

#### **Operands**

dst, RELOP, src, TCx

#### Description

This instruction performs a comparison in the D-unit ALU or in the A-unit ALU. Two accumulator, auxiliary registers, and temporary registers contents are compared. When an accumulator ACx is compared with an auxiliary or temporary register TAx, the 16 lowest bits of ACx are compared with TAx in the A-unit ALU. If the comparison is true, the TCx status bit is set to 1; otherwise, it is cleared to 0.

The comparison depends on the optional uns keyword and on M40 for accumulator comparisons. As the following table shows, the uns keyword specifies an unsigned comparison and M40 defines the comparison bit width for accumulator comparisons.

uns	src	dst	Comparison Type
no	TAx	TAy	16-bit signed comparison in A-unit ALU
no	TAx	ACy	16-bit signed comparison in A-unit ALU
no	ACx	TAy	16-bit signed comparison in A-unit ALU
no	ACx	ACy	if M40 = 0, 32-bit signed comparison in D-unit ALU if M40 = 1, 40-bit signed comparison in D-unit ALU
yes	TAx	TAy	16-bit unsigned comparison in A-unit ALU
yes	TAx	ACy	16-bit unsigned comparison in A-unit ALU
yes	ACx	TAy	16-bit unsigned comparison in A-unit ALU
yes	ACx	ACy	if $M40 = 0$ , 32-bit unsigned comparison in D-unit ALU if $M40 = 1$ , 40-bit unsigned comparison in D-unit ALU

#### Compatibility with C54x devices (C54CM = 1)

Contrary to the corresponding C54x instruction, the C55x register comparison instruction is performed in execute phase of the pipeline.

When C54CM = 1, the conditions testing the accumulators content are all performed as if M40 was set to 1.

Status Bits Affected by C54CM, M40

Affects TCx

**Repeat** This instruction can be repeated.

**See Also** See the following other related instructions:

- ☐ Compare Accumulator, Auxiliary, or Temporary Register Content with AND
- ☐ Compare Accumulator, Auxiliary, or Temporary Register Content with OR
- ☐ Compare Accumulator, Auxiliary, or Temporary Register Content Maximum
- ☐ Compare Accumulator, Auxiliary, or Temporary Register Content Minimum
- ☐ Compare Memory with Immediate Value

#### **Example 1**

Syntax	Description
TC1= AC1 = = T1	The signed content of AC1(15–0) is compared to the content of T1 and because
	they are equal, TC1 is set to 1.

Before		After	
AC1	00 0028 0400	AC1	00 0028 0400
T1	0400	T1	0400
TC1	0	TC1	1

Syntax	Description
TC1= T1 > = AC1	The content of T1 is compared to the signed content of AC1(15–0). The content of
	T1 is greater than the content of AC1, TC1 is set to 1.

Before				After			
T1			0500	T1			0500
AC1	80	0000	0400	AC1	80	0000	0400
TC1			0	TC1			1

# **CMPAND**

Compare Accumulator, Auxiliary, or Temporary Register Content with AND

☐ Compare Accumulator, Auxiliary, or Temporary Register Content Maximum

☐ Compare Accumulator, Auxiliary, or Temporary Register Content Minimum

#### **Syntax Characteristics**

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[1]	TCx = TCy & ur	ns(src RELOP dst	:)	Yes	3	1	Х
[2]	TCx = !TCy & u	i <mark>ns(</mark> src RELOP ds	st <mark>)</mark>	Yes	3	1	Х
Descri	ption	ALU. Two accare compared	ions perform a compa umulator, auxiliary reg . When an accumulato ister TAx, the 16 lowest	isters, and te or ACx is con	mporai npared	ry register with an a	rs contents auxiliary or
Status	Bits	Affected by	C54CM, M40, TCy				
		Affects	TCx				
See Als	so	See the follow	ing other related instru	uctions:			
		☐ Compare	Accumulator, Auxiliary	, or Temporar	y Regi	ster Cont	ent
		☐ Compare	Accumulator, Auxiliary,	or Temporary	/ Regis	ster Conte	ent with OR

☐ Compare Memory with Immediate Value

#### Compare Accumulator, Auxiliary, or Temporary Register Content with AND

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
	TCx = TCy & uns(src RELOP dst)				
[1a]	TC1 = TC2 & uns(src RELOP dst)	Yes	3	1	X
[1b]	TC2 = TC1 & uns(src RELOP dst)	Yes	3	1	Х

					_	
Opcode	0001	001E	FSSS	cc01	FDDD	0utt

**Operands** 

dst, RELOP, src, TC1, TC2

Description

This instruction performs a comparison in the D-unit ALU or in the A-unit ALU. Two accumulator, auxiliary registers, and temporary registers contents are compared. When an accumulator ACx is compared with an auxiliary or temporary register TAx, the 16 lowest bits of ACx are compared with TAx in the A-unit ALU. If the comparison is true, the TCx status bit is set to 1; otherwise, it is cleared to 0. The result of the comparison is ANDed with TCy; TCx is updated with this operation.

The comparison depends on the optional uns keyword and on M40 for accumulator comparisons. As the following table shows, the uns keyword specifies an unsigned comparison and M40 defines the comparison bit width for accumulator comparisons.

uns	src	dst	Comparison Type
no	TAx	TAy	16-bit signed comparison in A-unit ALU
no	TAx	ACy	16-bit signed comparison in A-unit ALU
no	ACx	TAy	16-bit signed comparison in A-unit ALU
no	ACx	ACy	If M40 = 0, 32-bit signed comparison in D-unit ALU if M40 = 1, 40-bit signed comparison in D-unit ALU
yes	TAx	TAy	16-bit unsigned comparison in A-unit ALU
yes	TAx	ACy	16-bit unsigned comparison in A-unit ALU
yes	ACx	TAy	16-bit unsigned comparison in A-unit ALU
yes	ACx	ACy	If M40 = 0, 32-bit unsigned comparison in D-unit ALU if M40 = 1, 40-bit unsigned comparison in D-unit ALU

#### Compatibility with C54x devices (C54CM = 1)

Contrary to the corresponding C54x instruction, the C55x register comparison instruction is performed in execute phase of the pipeline.

When C54CM = 1, the conditions testing the accumulators content are all performed as if M40 was set to 1.

Status Bits Affected by C54CM, M40, TCy

Affects TCx

**Repeat** This instruction can be repeated.

Syntax	Description
TC2 = TC1 & AC1 == AC2	The content of AC1(31–0) is compared to the content of AC2(31–0).
	The contents are equal (true), TC2 = TC1 & 1.

Before				After			
AC1	80	0028	0400	AC1	80	0028	0400
AC2	00	0028	0400	AC2	00	0028	0400
M40			0	M40			0
TC1			1	TC1			1
TC2			0	TC2			1

#### Compare Accumulator, Auxiliary, or Temporary Register Content with AND

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
	TCx = !TCy & uns(src RELOP dst)				
[2a]	TC1 = !TC2 & uns(src RELOP dst)	Yes	3	1	Χ
[2b]	TC2 = !TC1 & uns(src RELOP dst)	Yes	3	1	Х

					_	
Opcode	0001	001E	FSSS	cc01	FDDD	1utt

**Operands** 

dst, RELOP, src, TC1, TC2

Description

This instruction performs a comparison in the D-unit ALU or in the A-unit ALU. Two accumulator, auxiliary registers, and temporary registers contents are compared. When an accumulator ACx is compared with an auxiliary or temporary register TAx, the 16 lowest bits of ACx are compared with TAx in the A-unit ALU. If the comparison is true, the TCx status bit is set to 1; otherwise, it is cleared to 0. The result of the comparison is ANDed with the complement of TCy; TCx is updated with this operation.

The comparison depends on the optional uns keyword and on M40 for accumulator comparisons. As the following table shows, the uns keyword specifies an unsigned comparison and M40 defines the comparison bit width for accumulator comparisons.

uns	src	dst	Comparison Type
no	TAx	TAy	16-bit signed comparison in A-unit ALU
no	TAx	ACy	16-bit signed comparison in A-unit ALU
no	ACx	TAy	16-bit signed comparison in A-unit ALU
no	ACx	ACy	if M40 = 0, 32-bit signed comparison in D-unit ALU if M40 = 1, 40-bit signed comparison in D-unit ALU
yes	TAx	TAy	16-bit unsigned comparison in A-unit ALU
yes	TAx	ACy	16-bit unsigned comparison in A-unit ALU
yes	ACx	TAy	16-bit unsigned comparison in A-unit ALU
yes	ACx	ACy	if M40 = 0, 32-bit unsigned comparison in D-unit ALU if M40 = 1, 40-bit unsigned comparison in D-unit ALU

#### Compatibility with C54x devices (C54CM = 1)

Contrary to the corresponding C54x instruction, the C55x register comparison instruction is performed in execute phase of the pipeline.

When C54CM = 1, the conditions testing the accumulators content are all performed as if M40 was set to 1.

Status Bits Affected by C54CM, M40, TCy

Affects TCx

**Repeat** This instruction can be repeated.

Syntax	Description
TC2 = !TC1 & AC1 == AC2	The content of AC1(31–0) is compared to the content of AC2(31–0).
	The contents are equal (true), TC2 = !TC1 & 1.

Before				After			
AC1	80	0028	0400	AC1	80	0028	0400
AC2	00	0028	0400	AC2	00	0028	0400
M40			0	M40			0
TC1			1	TC1			1
TC2			0	<b>ፐ</b> ሮ2			0

# CMPOR

Compare Accumulator, Auxiliary, or Temporary Register Content with OR

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	TCx = TCy   uns(src RELOP dst)	Yes	3	1	Х
[2]	TCx = !TCy   uns(src RELOP dst)	Yes	3	1	X
Descri	ption These instructions perform a compa	arison in the [	D-unit A	ALU or in	the A-unit

Status Bits Affected by C54CM, M40, TCy

A-unit ALU.

Affects TCx

**See Also** See the following other related instructions:

- ☐ Compare Accumulator, Auxiliary, or Temporary Register Content
- ☐ Compare Accumulator, Auxiliary, or Temporary Register Content with AND

ALU. Two accumulator, auxiliary registers, and temporary registers contents are compared. When an accumulator ACx is compared with an auxiliary or temporary register TAx, the 16 lowest bits of ACx are compared with TAx in the

- ☐ Compare Accumulator, Auxiliary, or Temporary Register Content Maximum
- ☐ Compare Accumulator, Auxiliary, or Temporary Register Content Minimum
- ☐ Compare Memory with Immediate Value

#### Compare Accumulator, Auxiliary, or Temporary Register Content with OR

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
	TCx = TCy   uns(src RELOP dst)				
[1a]	TC1 = TC2   uns(src RELOP dst)	Yes	3	1	X
[1b]	TC2 = TC1   uns(src RELOP dst)	Yes	3	1	X

0001 001E FSSS cc10 FDDD Outt Opcode

**Operands** 

dst, RELOP, src, TC1, TC2

Description

This instruction performs a comparison in the D-unit ALU or in the A-unit ALU. Two accumulator, auxiliary registers, and temporary registers contents are compared. When an accumulator ACx is compared with an auxiliary or temporary register TAx, the 16 lowest bits of ACx are compared with TAx in the A-unit ALU. If the comparison is true, the TCx status bit is set to 1; otherwise, it is cleared to 0. The result of the comparison is ORed with TCy; TCx is updated with this operation.

The comparison depends on the optional uns keyword and on M40 for accumulator comparisons. As the following table shows, the uns keyword specifies an unsigned comparison and M40 defines the comparison bit width for accumulator comparisons.

uns	src	dst	Comparison Type
no	TAx	TAy	16-bit signed comparison in A-unit ALU
no	TAx	ACy	16-bit signed comparison in A-unit ALU
no	ACx	TAy	16-bit signed comparison in A-unit ALU
no	ACx	ACy	if M40 = 0, 32-bit signed comparison in D-unit ALU if M40 = 1, 40-bit signed comparison in D-unit ALU
yes	TAx	TAy	16-bit unsigned comparison in A-unit ALU
yes	TAx	ACy	16-bit unsigned comparison in A-unit ALU
yes	ACx	TAy	16-bit unsigned comparison in A-unit ALU
yes	ACx	ACy	if M40 = 0, 32-bit unsigned comparison in D-unit ALU if M40 = 1, 40-bit unsigned comparison in D-unit ALU

#### Compatibility with C54x devices (C54CM = 1)

Contrary to the corresponding C54x instruction, the C55x register comparison instruction is performed in execute phase of the pipeline.

When C54CM = 1, the conditions testing the accumulators content are all performed as if M40 was set to 1.

Status Bits Affected by C54CM, M40, TCy

Affects TCx

**Repeat** This instruction can be repeated.

Syntax	Description
TC2 = TC1   uns(AC1 != AR1)	The unsigned content of AC1(15–0) is compared to the unsigned content
	of AR1. The contents are equal (false), TC2 = TC1   0.

Before		After	
AC1	00 8028 0400	AC1	00 8028 0400
AR1	0400	AR1	0400
TC1	1	TC1	1
TC2	0	TC2	1

#### Compare Accumulator, Auxiliary, or Temporary Register Content with OR

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
	TCx = !TCy   uns(src RELOP dst)				
[2a]	TC1 = !TC2   uns(src RELOP dst)	Yes	3	1	X
[2b]	TC2 = !TC1   uns(src RELOP dst)	Yes	3	1	X

0001 001E FSSS cc10 FDDD 1utt Opcode

**Operands** 

dst, RELOP, src, TC1, TC2

Description

This instruction performs a comparison in the D-unit ALU or in the A-unit ALU. Two accumulator, auxiliary registers, and temporary registers contents are compared. When an accumulator ACx is compared with an auxiliary or temporary register TAx, the 16 lowest bits of ACx are compared with TAx in the A-unit ALU. If the comparison is true, the TCx status bit is set to 1; otherwise, it is cleared to 0. The result of the comparison is ORed with the complement of TCy; TCx is updated with this operation.

The comparison depends on the optional uns keyword and on M40 for accumulator comparisons. As the following table shows, the uns keyword specifies an unsigned comparison and M40 defines the comparison bit width for accumulator comparisons.

uns	src	dst	Comparison Type
no	TAx	TAy	16-bit signed comparison in A-unit ALU
no	TAx	ACy	16-bit signed comparison in A-unit ALU
no	ACx	TAy	16-bit signed comparison in A-unit ALU
no	ACx	ACy	if M40 = 0, 32-bit signed comparison in D-unit ALU if M40 = 1, 40-bit signed comparison in D-unit ALU
yes	TAx	TAy	16-bit unsigned comparison in A-unit ALU
yes	TAx	ACy	16-bit unsigned comparison in A-unit ALU
yes	ACx	TAy	16-bit unsigned comparison in A-unit ALU
yes	ACx	ACy	if M40 = 0, 32-bit unsigned comparison in D-unit ALU if M40 = 1, 40-bit unsigned comparison in D-unit ALU

#### Compatibility with C54x devices (C54CM = 1)

Contrary to the corresponding C54x instruction, the C55x register comparison instruction is performed in execute phase of the pipeline.

When C54CM = 1, the conditions testing the accumulators content are all performed as if M40 was set to 1.

Status Bits Affected by C54CM, M40, TCy

Affects TCx

**Repeat** This instruction can be repeated.

Syntax	Description
TC2 = !TC1   uns(AC1 != AR1)	The unsigned content of AC1(15–0) is compared to the unsigned content
	of AR1. The contents are equal (false), TC2 = !TC1   0.

Before				After			
AC1	00	8028	0400	AC1	00	8028	0400
AR1			0400	AR1			0400
TC1			1	TC1			1
TC2			1	TC2			0

MAX

Compare Accumulator, Auxiliary, or Temporary Register Content Maximum

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = max(src, dst)	Yes	2	1	Х

#### **Opcode**

0010 111E FSSS FDDD

#### **Operands**

dst, src

#### Description

This instruction performs a maximum comparison in the D-unit ALU or in the A-unit ALU. Two accumulator, auxiliary registers, and temporary registers contents are compared. When an accumulator ACx is compared with an auxiliary or temporary register TAx, the 16 lowest bits of ACx are compared with TAx in the A-unit ALU. If the comparison is true, the TCx status bit is set to 1; otherwise, it is cleared to 0.

- ☐ When the destination operand (dst) is an accumulator:
  - If an auxiliary or temporary register is the source operand (src) of the instruction, the 16 LSBs of the auxiliary or temporary register are sign extended to 40 bits according to SXMD.
  - The operation is performed on 40 bits in the D-unit ALU:

If M40 = 0, src(31-0) content is compared to dst(31-0) content. The extremum value is stored in dst. If the extremum value is the src content, the CARRY status bit is cleared to 0; otherwise, it is set to 1.

```
step1:if (src(31-0) > dst(31-0))
step2: { CARRY = 0; dst(39-0) = src(39-0) }
    else
step3: CARRY = 1
```

If M40 = 1, src(39-0) content is compared to dst(39-0) content. The extremum value is stored in dst. If the extremum value is the src content, the CARRY status bit is cleared to 0; otherwise, it is set to 1.

```
step1:if (src(39-0) > dst(39-0))
step2: { CARRY = 0; dst(39-0) = src(39-0) }
    else
step3: CARRY = 1
```

■ There is no overflow detection, overflow report, and saturation.

- ☐ When the destination operand (dst) is an auxiliary or temporary register:
  - If an accumulator is the source operand (src) of the instruction, the 16 LSBs of the accumulator are used to perform the operation.
  - The operation is performed on 16 bits in the A-unit ALU:

The src(15–0) content is compared to the dst(15–0) content. The extremum value is stored in dst.

```
step1: if (src(15-0) > dst(15-0))
step2: dst = src
```

■ There is no overflow detection and saturation.

#### Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if M40 status bit was locally set to 1. When the destination operand (dst) is an auxiliary or temporary register, the instruction execution is not impacted by the C54CM status bit. When the destination operand (dst) is an accumulator, this instruction always compares the source operand (src) with AC1 as follows:

- ☐ If an auxiliary or temporary register is the source operand (src) of the instruction, the 16 LSBs of the auxiliary or temporary register are sign extended to 40 bits according to SXMD
- ☐ The operation is performed on 40 bits in the D-unit ALU:

The src(39–0) content is compared to AC1(39–0) content. The extremum value is stored in dst. If the extremum value is the src content, the CARRY status bit is cleared to 0; otherwise, it is set to 1.

```
step1: if (src(39-0) > AC1(39-0))
step2: { CARRY = 0; dst(39-0) = src(39-0) }
    else
step3: { CARRY = 1; dst(39-0) = AC1(39-0) }
```

There is no overflow detection, overflow report, and saturation.

Status Bits

Affected by C54CM, M40, SXMD

Affects CARRY

Repeat

This instruction can be repeated.

See Also	See the following other related instructions:
	Compare Accumulator, Auxiliary, or Temporary Register Content
	Compare Accumulator, Auxiliary, or Temporary Register Content with AND
	Compare Accumulator, Auxiliary, or Temporary Register Content with OR
	☐ Compare Accumulator, Auxiliary, or Temporary Register Content Minimum
	Compare and Select Accumulator Content Maximum
	☐ Compare Memory with Immediate Value

# Example 1

Syntax	Description
AC1 = max(AC2, AC1)	The content of AC2 is less than the content of AC1, the content of AC1 remains
	the same and the CARRY status bit is set to 1.

Before				After			
AC2	00	0000	0000	AC2	00	0000	0000
AC1	00	8500	0000	AC1	00	8500	0000
SXMD			1	SXMD			1
M40			0	M40			0
CARRY			0	CARRY			1

# Example 2

Syntax	Description
` '	The content of AR1 is less than the content of AC1, the content of AC1 remains the same and the CARRY status bit is set to 1.

Before				After				
AR1			8020	AR1			8020	
AC1	00	0000	0040	AC1	00	0000	0040	
CARRY			0	CARRY			1	

Syntax	Description
T1 = max(AC1, T1)	The content of AC1(15–0) is greater than the content of T1, the content of
	AC1(15–0) is stored in T1 and the CARRY status bit is cleared to 0.

Before			After				
AC1	00 0000	8020	AC1	00	0000	8020	
T1		8010	T1			8020	
CARRY		0	CARRY			0	

MIN

Compare Accumulator, Auxiliary, or Temporary Register Content Minimum

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = min(src, dst)	Yes	2	1	Х

#### **Opcode**

0011 000E FSSS FDDD

#### **Operands**

dst, src

#### Description

This instruction performs a minimum comparison in the D-unit ALU or in the A-unit ALU. Two accumulator, auxiliary registers, and temporary registers contents are compared. When an accumulator ACx is compared with an auxiliary or temporary register TAx, the 16 lowest bits of ACx are compared with TAx in the A-unit ALU. If the comparison is true, the TCx status bit is set to 1; otherwise, it is cleared to 0.

- ☐ When the destination operand (dst) is an accumulator:
  - If an auxiliary or temporary register is the source operand (src) of the instruction, the 16 LSBs of the auxiliary or temporary register are sign extended to 40 bits according to SXMD.
  - The operation is performed on 40 bits in the D-unit ALU:

If M40 = 0, src(31-0) content is compared to dst(31-0) content. The extremum value is stored in dst. If the extremum value is the src content, the CARRY status bit is cleared to 0; otherwise, it is set to 1.

```
step1:if (src(31-0) < dst(31-0))
step2: { CARRY = 0; dst(39-0) = src(39-0) }
    else
step3:CARRY = 1</pre>
```

If M40 = 1, src(39-0) content is compared to dst(39-0) content. The extremum value is stored in dst. If the extremum value is the src content, the CARRY status bit is cleared to 0; otherwise, it is set to 1.

```
step1:if (src(39-0) < dst(39-0))
step2:{ CARRY = 0; dst(39-0) = src(39-0) }
    else
step3:CARRY = 1</pre>
```

■ There is no overflow detection, overflow report, and saturation.

- ☐ When the destination operand (dst) is an auxiliary or temporary register:
  - If an accumulator is the source operand (src) of the instruction, the 16 LSBs of the accumulator are used to perform the operation.
  - The operation is performed on 16 bits in the A-unit ALU:

The src(15–0) content is compared to the dst(15–0) content. The extremum value is stored in dst.

```
step1: if (src(15-0) < dst(15-0))
step2: dst = src</pre>
```

■ There is no overflow detection and saturation.

#### Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if M40 status bit was locally set to 1. When the destination operand (dst) is an auxiliary or temporary register, the instruction execution is not impacted by the C54CM status bit. When the destination operand (dst) is an accumulator, this instruction always compares the source operand (src) with AC1 as follows:

- ☐ If an auxiliary or temporary register is the source operand (src) of the instruction, the 16 LSBs of the auxiliary or temporary register are sign extended to 40 bits according to SXMD
- ☐ The operation is performed on 40 bits in the D-unit ALU:

The src(39–0) content is compared to AC1(39–0) content. The extremum value is stored in dst. If the extremum value is the src content, the CARRY status bit is cleared to 0; otherwise, it is set to 1.

```
step1: if (src(39-0) < AC1(39-0))
step2: { CARRY = 0; dst(39-0) = src(39-0) }
    else
step3: { CARRY = 1; dst(39-0) = AC1(39-0) }</pre>
```

There is no overflow detection, overflow report, and saturation.

Status Bits

Affected by C54CM, M40, SXMD

Affects CARRY

Repeat

This instruction can be repeated.

# See Also See the following other related instructions: Compare Accumulator, Auxiliary, or Temporary Register Content Compare Accumulator, Auxiliary, or Temporary Register Content with AND Compare Accumulator, Auxiliary, or Temporary Register Content with OR Compare Accumulator, Auxiliary, or Temporary Register Content Maximum Compare and Select Accumulator Content Minimum Compare Memory with Immediate Value

Syntax	Description
T1 = min(AC1, T1)	The content of AC1(15–0) is greater than the content of T1, the content of T1 remains the same and the CARRY status bit is set to 1.

Before				After			
AC1	00	8000	0000	AC1	00	8000	0000
T1			8020	T1			8020
CARRY			0	CARRY			1

# BCC

#### Compare and Branch

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	compare (uns(src RELOP K8)) goto L8	No	4	7/6	Х

<sup>†</sup> x/y cycles: x cycles = condition true, y cycles = condition false

#### **Opcode**

0110 1111 FSSS ccxu KKKK KKKK LLLL LLLL

**Operands** 

K8, L8, RELOP, src

#### Description

This instruction performs a comparison operation between a source (src) register content and an 8-bit signed value, K8. The instruction performs a comparison in the D-unit ALU or in the A-unit ALU. The comparison is performed in the execute phase of the pipeline. If the result of the comparison is true, a branch occurs.

The program branch address is specified as an 8-bit signed offset, L8, relative to the program counter (PC). Use this instruction to branch within a 256-byte window centered on the current PC value.

The comparison depends on the optional uns keyword and, for accumulator comparisons, on M40.

- ☐ In the case of an unsigned comparison, the 8-bit constant, K8, is zero extended to:
  - 16 bits, if the source (src) operand is an auxiliary or temporary register.
  - 40 bits, if the source (src) operand is an accumulator.
- ☐ In the case of a signed comparison, the 8-bit constant, K8, is sign extended to:
  - 16 bits, if the source (src) operand is an auxiliary or temporary register.
  - 40 bits, if the source (src) operand is an accumulator.

As the following table shows, the uns keyword specifies an unsigned comparison; M40 defines the comparison bit width of the accumulator.

uns	src	Comparison Type
no	TAx	16-bit signed comparison in A-unit ALU
no	ACx	if M40 = 0, 32-bit signed comparison in D-unit ALU if M40 = 1, 40-bit signed comparison in D-unit ALU
yes	TAx	16-bit unsigned comparison in A-unit ALU
yes	ACx	if M40 = 0, 32-bit unsigned comparison in D-unit ALU if M40 = 1, 40-bit unsigned comparison in D-unit ALU

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, the conditions testing the accumulator contents are all

performed as if M40 was set to 1.

**Status Bits** Affected by C54CM, M40

> Affects none

Repeat This instruction cannot be repeated.

See Also See the following other related instructions:

Branch Conditionally

□ Branch Unconditionally

□ Branch on Auxiliary Register Not Zero

Syntax	Description
compare (AC0 >= #12) goto branch	The signed content of AC0 is compared to the sign-extended 8-bit value (12). Because the content of AC0 is greater than or equal to 12, program control is passed to the program address label defined by branch (004078h).

	compare (AC0 >= #12) goto branch		
		address:	00 4075
branch			00 4078
:			

Before		After		
AC0	00 0000 3000	AC0	00 0000 3000	
PC	004071	PC	004078	

Syntax	Description
	The content of T1 is not equal to 1, program control is passed to the next instruction (the branch is not taken).

	compare (T1 != #1) goto branch		
		address:	00407D
branch :			004080

Before		After		
T1	0000	T1	0000	
PC	4079	PC	407D	

#### MAXDIFF

#### Compare and Select Accumulator Content Maximum

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	max_diff(ACx, ACy, ACz, ACw)	Yes	3	1	Х
[2]	max_diff_dbl(ACx, ACy, ACz, ACw, TRNx)	Yes	3	1	Х

**Description** Instruction [1] performs two paralleled 16-bit extremum selections in the D-unit

ALU. Instruction [2] performs a single 40-bit extremum selection in the D-unit

ALU.

Status Bits Affected by C54CM, M40, SATD

Affects ACOVw, CARRY

**See Also** See the following other related instructions:

☐ Compare Accumulator, Auxiliary, or Temporary Register Content

☐ Compare Accumulator, Auxiliary, or Temporary Register Content Maximum

☐ Compare and Select Accumulator Content Minimum

#### Compare and Select Accumulator Content Maximum

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	max_diff(ACx, ACy, ACz, ACw)	Yes	3	1	Х

# Opcode

0001 000E DDSS 1100 SSDD nnnn

#### **Operands**

ACw, ACx, ACy, ACz

#### Description

This instruction performs two paralleled 16-bit extremum selections in the D-unit ALU in one cycle. This instruction performs a dual maximum search.

The two operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulators are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit data path).

For each datapath (high and low):

- ☐ ACx and ACy are the source accumulators.
- ☐ The differences are stored in accumulator ACw.
- ☐ The subtraction computation is equivalent to the dual 16-bit subtractions instruction.
- □ For each of the two computations performed in the ALU, an overflow detection is made. If an overflow is detected on any of the data paths, the destination accumulator overflow status bit (ACOVw) is set.
  - For the operations performed in the ALU low part, overflow is detected at bit position 15.
  - For the operations performed in the ALU high part, overflow is detected at bit position 31.
- ☐ For all instructions, the carry of the operation performed in the ALU high part is reported in the CARRY status bit. The CARRY status bit is always extracted at bit position 31.
- ☐ Independently on each data path, if SATD = 1 when an overflow is detected on the data path, a saturation is performed:
  - For the operations performed in the ALU low part, saturation values are 7FFh (positive) and 8000h (negative).
  - For the operations performed in the ALU high part, saturation values are 00 7FFFh (positive) and FF 8000h (negative).

- ☐ The extremum is stored in accumulator ACz.
- ☐ The extremum is searched considering the selected bit width of the accumulators:
  - for the lower 16-bit data path, the sign bit is extracted at bit position 15
  - for the higher 24-bit data path, the sign bit is extracted at bit position 31
- According to the extremum found, a decision bit is shifted in TRNx from the MSBs to the LSBs:
  - TRN0 tracks the decision for the high part data path
  - TRN1 tracks the decision for the low part data path

If the extremum value is the ACx high or low part, the decision bit is cleared to 0; otherwise, it is set to 1:

```
TRN0 = TRN0 >> #1
TRN1 = TRN1 >> #1
ACw(39-16) = ACy(39-16) - ACx(39-16)
ACw(15-0) = ACy(15-0) - ACx(15-0)
If (ACx(31-16) > ACy(31-16))
   \{ bit(TRN0, 15) = \#0 ; ACz(39-16) = ACx(39-16) \}
else
   \{ bit(TRN0, 15) = #1 ; ACz(39-16) = ACy(39-16) \}
if (ACx(15-0) > ACy(15-0))
   \{ bit(TRN1, 15) = \#0 ; ACz(15-0) = ACx(15-0) \}
else
   \{ bit(TRN1, 15) = #1 ; ACz(15-0) = ACy(15-0) \}
```

#### Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if SATD is locally cleared to 0. Overflow is only detected and reported for the computation performed in the higher 24-bit data path (overflow is detected at bit position 31).

**Status Bits** 

Affected by C54CM, SATD

Affects ACOVw, CARRY

Repeat

This instruction can be repeated.

Syntax	Description
max_diff(AC0, AC1, AC2, AC1)	The difference is stored in AC1. The content of AC0(39–16) is subtracted from the content of AC1(39–16) and the result is stored in AC1(39–16). Since SATD = 1 and an overflow is detected, AC1(39–16) = FF 8000h (saturation). The content of AC0(15–0) is subtracted from the content of AC1(15–0) and the result is stored in AC1(15–0). The maximum is stored in AC2. The content of TRN0 and TRN1 is shifted right 1 bit. AC0(31–16) is greater than AC1(31–16), AC0(39–16) is stored in AC2(39–16) and TRN0(15) is cleared to 0. AC0(15–0) is greater than AC1(15–0), AC0(15–0) is stored in AC2(15–0) and TRN1(15) is cleared to 0.

Before			After			
AC0	10 2400	2222	AC0	10	2400	2222
AC1	90 0000	0000	AC1	FF	8000	DDDE
AC2	00 0000	0000	AC2	10	2400	2222
SATD		1	SATD			1
TRN0		1000	TRN0			0800
TRN1		0100	TRN1			0080
ACOV1		0	ACOV1			1
CARRY		1	CARRY			0

# Compare and Select Accumulator Content Maximum

### **Syntax Characteristics**

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[2a]	max_diff_dbl	(ACx,	ACy, ACz, ACw, TRN0)		Yes	3	1	Х
[2b]	max_diff_dbl	(ACx,	ACy, ACz, ACw <b>, TRN1)</b>		Yes	3	1	X
Opcode	e	TI	RNO	0001	000E DD	SS 11	101   SSD	D xxx0
		TI	RN1	0001	000E DD	SS 1	101 SSE	D xxx1
Operan	nds	AC	w, ACx, ACy, ACz, TRNx					
Descrip	otion		s instruction performs a sin s instruction performs a ma	_		selection	on in the D	D-unit ALU.
			ACx and ACy are the two	source	accumulato	S.		
			The difference between th ACw.	e source	e accumulato	ors is s	tored in a	ccumulator
			The subtraction computat	ion is ed	quivalent to t	he sub	traction in	nstruction.
			Overflow detection and subtraction borrow bit is re is the logical complement	eported	in the CARF	Y stati		
			When an overflow is dete SATD.	cted, the	e accumulato	or is sa	turated a	ccording to
			The extremum between the ACz.	e sourc	e accumulato	ors is s	tored in a	ccumulator
			The extremum computate maximum instruction. How the extremum search but	wever, th	ne CARRY s	tatus b	oit is not u	
			According to the extremu the MSBs to the LSBs. If cleared to 0; otherwise, it	the extr	emum value			

```
If M40 = 0:
    TRNx = TRNx >> #1
    ACw(39-0) = ACy(39-0) - ACx(39-0)
    If (ACx(31-0) > ACy(31-0))
        { bit(TRNx, 15) = #0 ; ACz(39-0) = ACx(39-0) }
    else
        { bit(TRNx, 15) = #1 ; ACz(39-0) = ACy(39-0) }

If M40 = 1:
    TRNx = TRNx >> #1
    ACw(39-0) = ACy(39-0) - ACx(39-0)
    If (ACx(39-0) > ACy(39-0))
        { bit(TRNx, 15) = #0 ; ACz(39-0) = ACx(39-0) }
    else
        { bit(TRNx, 15) = #1 ; ACz(39-0) = ACy(39-0) }
```

When C54CM = 1, this instruction is executed as if M40 status bit was locally set to 1. However to ensure compatibility versus overflow detection and saturation of the destination accumulator, this instruction must be executed with M40 = 0.

**Status Bits** 

Affected by C5

C54CM, M40, SATD

Affects

ACOVw, CARRY

#### Repeat

This instruction can be repeated.

Syntax	Description
max_diff_dbl(AC0, AC1, AC2, AC3, TRN1)	The difference is stored in AC3. The content of AC0 is subtracted from the content of AC1 and the result is stored in AC3. The maximum is stored in AC2. The content of TRN1 is shifted right 1 bit. AC0 is greater than AC1, AC0 is stored in AC2 and TRN1(15) is cleared to 0.

Before				After			
AC0	10	2400	2222	AC0	10	2400	2222
AC1	00	8000	DDDE	AC1	00	8000	DDDE
AC2	00	0000	0000	AC2	10	2400	2222
AC3	00	0000	0000	AC3	F0	5C00	BBBC
M40			1	M40			1
SATD			1	SATD			1
TRN1			0800	TRN1			0040
ACOV3			0	ACOV3			0
CARRY			0	CARRY			0

## MINDIFF

### Compare and Select Accumulator Content Minimum

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	min_diff(ACx, ACy, ACz, ACw)	Yes	3	1	Х
[2]	min_diff_dbl(ACx, ACy, ACz, ACw, TRNx)	Yes	3	1	Х

**Description** Instruction [1] performs two paralleled 16-bit extremum selections in the D-unit

ALU. Instruction [2] performs a single 40-bit extremum selection in the D-unit

ALU.

Status Bits Affected by C54CM, M40, SATD

Affects ACOVw, CARRY

**See Also** See the following other related instructions:

☐ Compare Accumulator, Auxiliary, or Temporary Register Content

☐ Compare Accumulator, Auxiliary, or Temporary Register Content Minimum

☐ Compare and Select Accumulator Content Maximum

### Compare and Select Accumulator Content Minimum

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	min_diff(ACx, ACy, ACz, ACw)	Yes	3	1	Χ

### Opcode

0001 000E DDSS 1110 SSDD xxxx

#### **Operands**

ACw, ACx, ACy, ACz

#### Description

This instruction performs two paralleled 16-bit extremum selections in the D-unit ALU in one cycle. This instruction performs a dual minimum search.

The two operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulators are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit data path).

For each datapath (high and low):

- ☐ ACx and ACy are the source accumulators.
- ☐ The differences are stored in accumulator ACw.
- ☐ The subtraction computation is equivalent to the dual 16-bit subtractions instruction.
- □ For each of the two computations performed in the ALU, an overflow detection is made. If an overflow is detected on any of the data paths, the destination accumulator overflow status bit (ACOVw) is set.
  - For the operations performed in the ALU low part, overflow is detected at bit position 15.
  - For the operations performed in the ALU high part, overflow is detected at bit position 31.
- ☐ For all instructions, the carry of the operation performed in the ALU high part is reported in the CARRY status bit. The CARRY status bit is always extracted at bit position 31.
- ☐ Independently on each data path, if SATD = 1 when an overflow is detected on the data path, a saturation is performed:
  - For the operations performed in the ALU low part, saturation values are 7FFh (positive) and 8000h (negative).
  - For the operations performed in the ALU high part, saturation values are 00 7FFFh (positive) and FF 8000h (negative).

- ☐ The extremum is stored in accumulator ACz.
- ☐ The extremum is searched considering the selected bit width of the accumulators:
  - for the lower 16-bit data path, the sign bit is extracted at bit position 15
  - for the higher 24-bit data path, the sign bit is extracted at bit position 31
- According to the extremum found, a decision bit is shifted in TRNx from the MSBs to the LSBs:
  - TRN0 tracks the decision for the high part data path
  - TRN1 tracks the decision for the low part data path

If the extremum value is the ACx high or low part, the decision bit is cleared to 0; otherwise, it is set to 1:

```
TRN0 = TRN0 >> #1

TRN1 = TRN1 >> #1

ACw(39-16) = ACy(39-16) - ACx(39-16)

ACw(15-0) = ACy(15-0) - ACx(15-0)

If (ACx(31-16) < ACy(31-16))
{ bit(TRN0, 15) = #0 ; ACz(39-16) = ACx(39-16) }

else
{ bit(TRN0, 15) = #1 ; ACz(39-16) = ACy(39-16) }

if (ACx(15-0) < ACy(15-0))
{ bit(TRN1, 15) = #0 ; ACz(15-0) = ACx(15-0) }

else
{ bit(TRN1, 15) = #1 ; ACz(15-0) = ACy(15-0) }
```

#### Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if SATD is locally cleared to 0. Overflow is only detected and reported for the computation performed in the higher 24-bit data path (overflow is detected at bit position 31).

**Status Bits** 

Affected by C54CM, SATD

Affects ACOVw, CARRY

Repeat

This instruction can be repeated.

Syntax	Description
min_diff(AC0, AC1, AC2, AC1)	The difference is stored in AC1. The content of AC0(39–16) is subtracted from the content of AC1(39–16) and the result is stored in AC1(39–16). Since SATD = 1 and an overflow is detected, AC1(39–16) = FF 8000h (saturation). The content of AC0(15–0) is subtracted from the content of AC1(15–0) and the result is stored in AC1(15–0). The minimum is stored in AC2 (sign bit extracted at bits 31 and 15). The content of TRN0 and TRN1 is shifted right 1 bit. AC0(31–16) is greater than or equal to AC1(31–16), AC1(39–16) is stored in AC2(39–16) and TRN0(15) is set to 1. AC0(15–0) is greater than or equal to AC1(15–0), AC1(15–0) is stored in AC2(15–0) and TRN1(15) is set to 1.

Before			After	
AC0	10 240	0 2222	AC0	10 2400 2222
AC1	00 800	0 DDDE	AC1	FF 8000 BBBC
AC2	10 240	0 2222	AC2	00 8000 DDDE
SATD		1	SATD	1
TRN0		0800	TRN0	8400
TRN1		0040	TRN1	8020
ACOV1		0	ACOV1	1
CARRY		0	CARRY	1

# Compare and Select Accumulator Content Minimum

# **Syntax Characteristics**

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[2a]	min_diff_dbl(A	Cx, A	ACy, ACz, ACw, TRN0)		Yes	3	1	Х
[2b]	min_diff_dbl(A	Cx, A	ACy, ACz, ACw, TRN1)		Yes	3	1	Х
Opcode	)	TF	RNO	0001	000E DD	SS 11	l11   SSD	D xxx0
		TF	RN1	0001	000E DD	SS 11	l11   SSD	D xxx1
Operan	ds	AC	w, ACx, ACy, ACz, TRNx					
Descrip	otion		s instruction performs a sing s instruction performs a mil	•		selectio	on in the D	)-unit ALU.
			ACx and ACy are the two	source	accumulator	S.		
			The difference between the ACw.	e source	e accumulato	ors is st	tored in a	ccumulator
			The subtraction computati	on is ed	quivalent to t	he sub	traction in	nstruction.
			Overflow detection and subtraction borrow bit is re is the logical complement	ported	in the CARR	Y statu		
			When an overflow is detected SATD.	ted, the	e accumulato	or is sa	turated ad	ccording to
			The extremum between the ACz.	e source	e accumulato	ors is s	tored in a	ccumulator
			The extremum computation maximum instruction. How the extremum search but I	vever, th	ne CARRY s	tatus b	oit is not u	
			According to the extremur the MSBs to the LSBs. If the cleared to 0; otherwise, it	he extr	emum value			

```
If M40 = 0:
```

```
TRNx = TRNx >> #1
ACw(39-0) = ACy(39-0) - ACx(39-0)
If (ACx(31-0) < ACy(31-0))
   \{ bit(TRNx, 15) = \#0 ; ACz(39-0) = ACx(39-0) \}
else
   \{ bit(TRNx, 15) = #1 ; ACz(39-0) = ACy(39-0) \}
```

#### If M40 = 1:

```
TRNx = TRNx >> #1
ACw(39-0) = ACy(39-0) - ACx(39-0)
If (ACx(39-0) < ACy(39-0))
   \{ bit(TRNx, 15) = \#0 ; ACz(39-0) = ACx(39-0) \}
else
   \{ bit(TRNx, 15) = #1 ; ACz(39-0) = ACy(39-0) \}
```

### Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if M40 status bit was locally set to 1. However to ensure compatibility versus overflow detection and saturation of the destination accumulator, this instruction must be executed with M40 = 0.

**Status Bits** Affected by C54CM, M40, SATD

> Affects ACOVw, CARRY

Repeat This instruction can be repeated.

Syntax	Description
min_diff_dbl(AC0, AC1, AC2, AC3, TRN0)	The difference is stored in AC3. The content of AC0 is subtracted from the content of AC1 and the result is stored in AC3. The minimum is stored in AC2. The content of TRN0 is shifted right 1 bit. If AC0 is less than AC1, AC0 is stored in AC2 and TRN0(15) is cleared to 0; otherwise, AC1 is stored in AC2 and TRN0(15) is set to 1.

### **CMP**

## Compare Memory with Immediate Value

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	TC1 = (Smem == K16)	No	4	1	Х
[2]	TC2 = (Smem == K16)	No	4	1	X

Opcode	TC1	1111	0000	AAAA	AAAI	KKKK	KKKK	KKKK	KKKK
	TC2	1111	0001	AAAA	AAAI	KKKK	KKKK	KKKK	KKKK

### **Operands**

K16, Smem, TCx

### **Description**

This instruction performs a comparison in the A-unit ALU. The data memory operand Smem is compared to the 16-bit signed constant, K16. If they are equal, the TCx status bit is set to 1; otherwise, it is cleared to 0.

if((Smem) == K16)  

$$TCx = 1$$
  
else  
 $TCx = 0$ 

### **Status Bits**

Affected by none

Affects TCx

#### Repeat

This instruction can be repeated.

#### See Also

See the following other related instructions:

☐ Compare Accumulator, Auxiliary, or Temporary Register Content

Syntax	Description
TC1 = (*AR1+ == #400h)	The content addressed by AR1 is compared to the signed 16-bit value
	(400h). Because they are equal, TC1 is set to 1. AR1 is incremented by 1.

Before		After	
AR1	0285	AR1	0286
0285	0400	0285	0400
TC1	0	TC1	1

Syntax	Description
TC2 = (*AR1 == #400h)	The content addressed by AR1 is compared to the signed 16-bit value (400h). Because they are not equal, TC2 is cleared to 0.

Before		After	
AR1	0285	AR1	0285
0285	0000	0285	0000
TC2	0	TC2	0

# **BNOT**

Complement Accumulator, Auxiliary, or Temporary Register Bit

### **Syntax Characteristics**

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline
[1]	cbit(src, Baddr)					No No	3	1	X
	obit(oro, badar)								
Opcod	e				1110	1100 AA	AA A	AAI FSS	SS 011x
Operar	nds	Ba	ddr, src						
Descri	ption	Thi	is instructio	n performs a bit	manipu	ılation:			
			In the D-u	nit ALU, if the so	ource (s	rc) register	operan	d is an ac	cumulator.
			In the A-u temporary	nit ALU, if the some $r$ register.	source (	src) register	opera	nd is an a	auxiliary or
			The instruction complements a single bit, as defined by the bit addressing mode, Baddr, of the source register.						
		The	e generated	d bit address mu	ıst be w	ithin:			
		□ 0–39 when accessing accumulator bits (only the 6 LSBs of the generated bit address are used to determine the bit position). If the generated bit address is not within 0–39, the selected register bit value does not change.							
				n accessing aux erated address	-		•		
Status	Bits	Aff	ected by	none					
		Aff	ects	none					
Repeat	t	Thi	is instructio	n can be repeat	ed.				
See Als	so	Se	e the follow	ring other related	d instru	ctions:			
			Clear Acc	umulator, Auxilia	ary, or T	emporary R	egister	Bit	
			Complem	ent Accumulator	r, Auxilia	ary, or Temp	orary F	Register C	ontent
			Complem	ent Memory Bit					
			Set Accur	nulator, Auxiliary	y, or Ter	nporary Reg	jister B	it	
Examp	le								
Syntax	<u> </u>	De	escription						

Syntax	Description
cbit(T0, AR1)	The bit at the position defined by the content of AR1(3–0) in T0 is complemented.

Before		After		
T0	E000	T0	F000	
AR1	000C	AR1	000C	

# NOT

# Complement Accumulator, Auxiliary, or Temporary Register Content

## **Syntax Characteristics**

No. Syntax	Parallel Enable Bit Size Cycles Pipeline						
[1] dst = ~src	Yes 2 1 X						
Opcode	0011 011E FSSS FDDD						
Operands	dst, src						
Description	This instruction computes the 1s complement (bitwise complement) of the content of the source register (src).						
	☐ When the destination (dst) operand is an accumulator:						
	■ The bit inversion is performed on 40 bits in the D-unit ALU and the result is stored in the destination accumulator.						
	If an auxiliary or temporary register is the source (src) operand of the instruction, the 16 LSBs of the auxiliary or temporary register are zero extended.						
	☐ When the destination (dst) operand is an auxiliary or temporary register:						
	■ The bit inversion is performed on 16 bits in the A-unit ALU and the result is stored in the destination auxiliary or temporary register.						
	■ If an accumulator is the source (src) operand of the instruction, the 16 LSBs of the accumulator are used to perform the operation.						
Status Bits	Affected by none						
	Affects none						
Repeat	This instruction can be repeated.						
See Also	See the following other related instructions:						
	☐ Complement Accumulator, Auxiliary, or Temporary Register Bit						
	☐ Negate Accumulator, Auxiliary, or Temporary Register Content						
Example							
_	Income.						

Syntax	Description
AC1 = ~AC0	The content of AC0 is complemented and the result is stored in AC1.

Beiore				After			
AC0	7E	2355	4FC0	AC0	7E	2355	4FC0
AC1	00	2300	5678	AC1	81	DCAA	B03F

### **BNOT**

### Complement Memory Bit

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	cbit(Smem, src)	No	3	1	Х
[1]	cbit(Smem, src)	No	3	1	

Opcode | 1110 0011 | AAAA AAAI | FSSS 111x

Operands Smem, src

**Description** This instruction performs a bit manipulation in the A-unit ALU. The instruction

complements a single bit, as defined by the content of the source (src)

operand, of a memory (Smem) location.

The generated bit address must be within 0-15 (only the 4 LSBs of the register

are used to determine the bit position).

Status Bits Affected by none

Affects none

**Repeat** This instruction can be repeated.

**See Also** See the following other related instructions:

☐ Clear Memory Bit

☐ Complement Accumulator, Auxiliary, or Temporary Register Bit

☐ Complement Accumulator, Auxiliary, or Temporary Register Content

☐ Set Memory Bit

Syntax	Description
cbit(*AR3, AC0)	The bit at the position defined by AC0(3–0) in the content addressed by AR3 is complemented.

### EXP

### Compute Exponent of Accumulator Content

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	$Tx = \mathbf{exp}(ACx)$	Yes	3	1	Χ

**Opcode** | 0001 000E | xxSS 1000 | xxdd xxxx

Operands ACx, Tx

**Description**This instruction computes the exponent of the source accumulator ACx in the D-unit shifter. The result of the operation is stored in the temporary register Tx.

The A-unit ALU is used to make the move operation.

This exponent is a signed 2s-complement value in the -8 to 31 range. The exponent is computed by calculating the number of leading bits in ACx and subtracting 8 from this value. The number of leading bits is the number of shifts to the MSBs needed to align the accumulator content on a signed 40-bit representation.

ACx is not modified after the execution of this instruction. If ACx is equal to 0, Tx is loaded with 0.

This instruction produces in Tx the opposite result than computed by the Compute Mantissa and Exponent of Accumulator Content instruction (page 5-132).

Status Bits Affected by none

Affects none

**Repeat** This instruction can be repeated.

**See Also** See the following other related instructions:

Compute Mantissa and Exponent of Accumulator Content

Syntax	Description
$T1 = \exp(AC0)$	The exponent is computed by subtracting 8 from the number of leading bits in the content of AC0. The exponent value is a signed 2s-complement value in the –8 to 31 range and is stored in T1.

Before		After	
AC0	FF FFFF FFCB	AC0	FF FFFF FFCB
T1	0000	T1	0019

## MANT::NEXP

### Compute Mantissa and Exponent of Accumulator Content

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = mant(ACx), Tx = -exp(ACx)	Yes	3	1	X2

**Opcode** 

0001 000E DDSS 1001 xxdd xxxx

**Operands** 

ACx, ACy, Tx

**Description** 

This instruction computes the exponent and mantissa of the source accumulator ACx. The computation of the exponent and the mantissa is executed in the D-unit shifter. The exponent is computed and stored in the temporary register Tx. The A-unit is used to make the move operation. The mantissa is stored in the accumulator ACy.

The exponent is a signed 2s-complement value in the -31 to 8 range. The exponent is computed by calculating the number of leading bits in ACx and subtracting this value from 8. The number of leading bits is the number of shifts to the MSBs needed to align the accumulator content on a signed 40-bit representation.

The mantissa is obtained by aligning the ACx content on a signed 32-bit representation. The mantissa is computed and stored in ACy.

- ☐ The shift operation is performed on 40 bits.
  - When shifting to the LSBs, bit 39 of ACx is extended to bit 31.
  - When shifting to the MSBs, 0 is inserted at bit position 0.
- ☐ If ACx is equal to 0, Tx is loaded with 8000h.

This instruction produces in Tx the opposite result than computed by the Compute Exponent of Accumulator Content instruction (page 5-131).

Status Bits

Affected by none

Affects

none

Repeat

This instruction can be repeated.

See Also

See the following other related instructions:

☐ Compute Exponent of Accumulator Content

# Example 1

Syntax	Description
AC1 = mant(AC0), T1 = -exp(AC0)	The exponent is computed by subtracting the number of leading bits in the content of AC0 from 8. The exponent value is a signed 2s-complement value in the –31 to 8 range and is stored in T1. The mantissa is computed by aligning the content of AC0 on a signed 32-bit representation. The mantissa value is stored in AC1.

Before				After			
AC0	21	0A0A	0A0A	AC0	21	0A0A	0A0A
AC1	FF	FFFF	F001	AC1	00	4214	1414
T1			0000	T1			0007

Syntax	Description
AC1 = mant(AC0), T1 = -exp(AC0)	The exponent is computed by subtracting the number of leading bits in the content of AC0 from 8. The exponent value is a signed 2s-complement value in the –31 to 8 range and is stored in T1. The mantissa is computed by aligning the content of AC0 on a signed 32-bit representation. The mantissa value is stored in AC1.

Beiore				After			
AC0	00	E804	0000	AC0	00	E804	0000
AC1	FF	FFFF	F001	AC1	00	7402	0000
T1			0000	T1			0001

## **BCNT**

#### Count Accumulator Bits

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	Tx = count(ACx, ACy, TC1)	Yes	3	1	X
[2]	Tx = count(ACx, ACy, TC2)	Yes	3	1	X

Opcode	TC1	0001	000E	xxSS	1010	ssdd	xxx0
	TC2	0001	000E	XXSS	1010	ssdd	xxx1

### **Operands** ACx, ACy, Tx, TCx

## **Description** This instruction performs bit field manipulation in the D-unit shifter. The result

is stored in the selected temporary register (Tx). The A-unit ALU is used to

make the move operation.

Accumulator ACx is ANDed with accumulator ACy. The number of bits set to 1 in the intermediary result is evaluated and stored in the selected temporary register (Tx). If the number of bits is even, the selected TCx status bit is cleared to 0. If the number of bits is odd, the selected TCx status bit is set to 1.

Status Bits Affected by none

Affects TCx

**Repeat** This instruction can be repeated.

Syntax	Description
	The content of AC1 is ANDed with the content of AC2, the number of bits set to 1 in the result is evaluated and stored in T1. The number of bits set to 1 is odd, TC1 is set to 1.

Before				After			
AC1	7E	2355	4FC0	AC1	7E	2355	4FC0
AC2	0F	E340	5678	AC2	0 F	E340	5678
T1			0000	T1			000B
TC1			0	TC1			1

ADD

#### **Dual 16-Bit Additions**

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	HI(ACy) = HI(Lmem) + HI(ACx), LO(ACy) = LO(Lmem) + LO(ACx)	No	3	1	Х
[2]	HI(ACx) = HI(Lmem) + Tx, LO(ACx) = LO(Lmem) + Tx	No	3	1	Х

**Description** These instructions perform two paralleled addition operations in one cycle.

> The operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulator are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit datapath).

**Status Bits** Affected by C54CM, SATD, SXMD Affects ACOVx, ACOVy, CARRY

See Also See the following other related instructions:

- Addition
- ☐ Addition or Subtraction Conditionally
- ☐ Addition or Subtraction Conditionally with Shift
- ☐ Addition with Parallel Store Accumulator Content to Memory
- Addition, Subtraction, or Move Accumulator Content Conditionally
- Dual 16-Bit Addition and Subtraction
- Dual 16-Bit Subtraction and Addition

#### **Dual 16-Bit Additions**

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	HI(ACy) = HI(Lmem) + HI(ACx), LO(ACy) = LO(Lmem) + LO(ACx)	No	3	1	Х

#### **Opcode**

| 1110 | 1110 | AAAA | AAAI | SSDD | 000x

#### **Operands**

ACx, ACy, Lmem

#### **Description**

This instruction performs two paralleled addition operations in one cycle. The operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulator are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit datapath).

- ☐ The data memory operand dbl(Lmem) is divided into two 16-bit parts:
  - the lower part is used as one of the 16-bit operands of the ALU low part
  - the higher part is sign extended to 24 bits according to SXMD and is used in the ALU high part
- ☐ The data memory operand dbl(Lmem) addresses are aligned:
  - if Lmem address is even: most significant word = Lmem, least significant word = Lmem + 1
  - if Lmem address is odd: most significant word = Lmem, least significant word = Lmem 1
- ☐ For each of the two computations performed in the ALU, an overflow detection is made. If an overflow is detected on any of the data paths, the destination accumulator overflow status bit (ACOVy) is set.
  - For the operations performed in the ALU low part, overflow is detected at bit position 15.
  - For the operations performed in the ALU high part, overflow is detected at bit position 31.
- ☐ For all instructions, the carry of the operation performed in the ALU high part is reported in the CARRY status bit. The CARRY status bit is always extracted at bit position 31.

- ☐ Independently on each data path, if SATD = 1 when an overflow is detected on the data path, a saturation is performed:
  - For the operations performed in the ALU low part, saturation values are 7FFFh and 8000h.
  - For the operations performed in the ALU high part, saturation values are 00 7FFFh and FF 8000h.

When C54CM = 1, this instruction is executed as if SATD is locally cleared to 0. Overflow is only detected and reported for the computation performed in the higher 24-bit datapath (overflow is detected at bit position 31).

- ☐ When C54CM = 1 and C16 = 1, the instruction behaves like a dual 16-bit instruction and the carry is not propagated at bit 15 in the D-unit ALU.
- ☐ When C54CM = 1 and C16 = 0, the instruction behaves like a single arithmetic instruction and the carry is propagated at bit 15 in the D-unit ALU.

**Status Bits** Affected by C16, C54CM, SATD, SXMD

> Affects ACOVy, CARRY

Repeat This instruction can be repeated.

Syntax	Description
HI(AC0) = HI(*AR3) + HI(AC1), LO(AC0) = LO(*AR3) + LO(AC1)	Both instructions are performed in parallel. When the Lmem address is even (AR3 = even): The content of AC1(39–16) is added to the content addressed by AR3 and the result is stored in AC0(39–16). The content of AC1(15–0) is added to the content addressed by AR3 + 1 and the result is stored in AC0(15–0).

#### **Dual 16-Bit Additions**

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	HI(ACx) = HI(Lmem) + Tx, LO(ACx) = LO(Lmem) + Tx	No	3	1	Х

#### Opcode

1110 1110 AAAA AAAI ssDD 100x

#### **Operands**

ACx, Lmem, Tx

#### Description

This instruction performs two paralleled addition operations in one cycle. The operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulator are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit datapath).

- ☐ The temporary register Tx:
  - is used as one of the 16-bit operands of the ALU low part
  - is duplicated and, according to SXMD, sign extended to 24 bits to be used in the ALU high part
- ☐ The data memory operand dbl(Lmem) is divided into two 16-bit parts:
  - the lower part is used as one of the 16-bit operands of the ALU low part
  - the higher part is sign extended to 24 bits according to SXMD and is used in the ALU high part
- ☐ The data memory operand dbl(Lmem) addresses are aligned:
  - if Lmem address is even: most significant word = Lmem, least significant word = Lmem + 1
  - if Lmem address is odd: most significant word = Lmem, least significant word = Lmem 1
- ☐ For each of the two computations performed in the ALU, an overflow detection is made. If an overflow is detected on any of the data paths, the destination accumulator overflow status bit (ACOVx) is set.
  - For the operations performed in the ALU low part, overflow is detected at bit position 15.
  - For the operations performed in the ALU high part, overflow is detected at bit position 31.

- For all instructions, the carry of the operation performed in the ALU high part is reported in the CARRY status bit. The CARRY status bit is always extracted at bit position 31.
- ☐ Independently on each data path, if SATD = 1 when an overflow is detected on the data path, a saturation is performed:
  - For the operations performed in the ALU low part, saturation values are 7FFFh and 8000h.
  - For the operations performed in the ALU high part, saturation values are 00 7FFFh and FF 8000h.

When C54CM = 1, this instruction is executed as if SATD is locally cleared to 0. Overflow is only detected and reported for the computation performed in the higher 24-bit datapath (overflow is detected at bit position 31).

**Status Bits** Affected by C54CM, SATD, SXMD

> Affects ACOVx, CARRY

Repeat This instruction can be repeated.

Syntax	Description
	Both instructions are performed in parallel. When the Lmem address is even (AR3 = even): The content of T0 is added to the content addressed by AR3 and the result is stored in AC0(39–16). The duplicated content of T0 is added to the content addressed by AR3 + 1 and the result is stored in AC0(15–0).

### **ADDSUB**

#### Dual 16-Bit Addition and Subtraction

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	HI(ACx) = Smem + Tx, LO(ACx) = Smem - Tx	No	3	1	Х
[2]	HI(ACx) = HI(Lmem) + Tx, LO(ACx) = LO(Lmem) - Tx	No	3	1	Х

#### **Description** These instructions perform two paralleled addition and subtraction operations

in one cycle.

The operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulator are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit datapath).

#### **Status Bits** Affected by C54CM, SATD, SXMD

Affects ACOVx, ACOVy, CARRY

#### See Also See the following other related instructions:

- ☐ Addition
- □ Dual 16-Bit Additions
- ☐ Dual 16-Bit Subtractions
- ☐ Dual 16-Bit Subtraction and Addition
- ☐ Subtraction

#### Dual 16-Bit Addition and Subtraction

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	HI(ACx) = Smem + Tx, LO(ACx) = Smem - Tx	No	3	1	Х

#### **Opcode**

1101 1110 AAAA AAAI ssDD 1000

### **Operands**

ACx, Smem, Tx

#### Description

This instruction performs two paralleled arithmetical operations in one cycle: an addition and subtraction. The operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulator are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit datapath).

- ☐ The data memory operand Smem:
  - is used as one of the 16-bit operands of the ALU low part
  - is duplicated and, according to SXMD, sign extended to 24 bits to be used in the ALU high part
- ☐ The temporary register Tx:
  - is used as one of the 16-bit operands of the ALU low part
  - is duplicated and, according to SXMD, sign extended to 24 bits to be used in the ALU high part
- ☐ For each of the two computations performed in the ALU, an overflow detection is made. If an overflow is detected on any of the data paths, the destination accumulator overflow status bit (ACOVx) is set.
  - For the operations performed in the ALU low part, overflow is detected at bit position 15.
  - For the operations performed in the ALU high part, overflow is detected at bit position 31.
- For all instructions, the carry of the operation performed in the ALU high part is reported in the CARRY status bit. The CARRY status bit is always extracted at bit position 31.

- ☐ Independently on each data path, if SATD = 1 when an overflow is detected on the data path, a saturation is performed:
  - For the operations performed in the ALU low part, saturation values are 7FFh and 8000h.
  - For the operations performed in the ALU high part, saturation values are 00 7FFFh and FF 8000h.

When C54CM = 1, this instruction is executed as if SATD is locally cleared to 0. Overflow is only detected and reported for the computation performed in the higher 24-bit datapath (overflow is detected at bit position 31).

**Status Bits** 

Affected by C54CM, SATD, SXMD

Affects ACOVx, CARRY

Repeat

This instruction can be repeated.

Syntax	Description
HI(AC1) = *AR1 + T1,	Both instructions are performed in parallel. The content addressed by AR1 is added
LO(AC1) = *AR1 – T1	to the content of T1 and the result is stored in AC1(39–16). The duplicated content of T1 is subtracted from the duplicated content addressed by AR1 and the result is
	stored in AC1(15–0).

Before		After	
AC1	00 2300 0000	AC1	00 2300 A300
T1	4000	T1	4000
AR1	0201	AR1	0201
201	E300	201	E300
SXMD	1	SXMD	1
M40	1	M40	1
ACOV0	0	ACOV0	0
CARRY	0	CARRY	1

#### Dual 16-Bit Addition and Subtraction

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	HI(ACx) = HI(Lmem) + Tx, LO(ACx) = LO(Lmem) - Tx	No	3	1	Х

#### **Opcode**

| 1110 | 1110 | AAAA | AAAI | ssDD | 110x

#### **Operands**

ACx, Lmem, Tx

#### Description

This instruction performs two paralleled arithmetical operations in one cycle: an addition and subtraction. The operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulator are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit datapath).

- ☐ The temporary register Tx:
  - is used as one of the 16-bit operands of the ALU low part
  - is duplicated and, according to SXMD, sign extended to 24 bits to be used in the ALU high part
- ☐ The data memory operand dbl(Lmem) is divided into two 16-bit parts:
  - the lower part is used as one of the 16-bit operands of the ALU low part
  - the higher part is sign extended to 24 bits according to SXMD and is used in the ALU high part
- ☐ The data memory operand dbl(Lmem) addresses are aligned:
  - if Lmem address is even: most significant word = Lmem, least significant word = Lmem + 1
  - if Lmem address is odd: most significant word = Lmem, least significant word = Lmem 1
- ☐ For each of the two computations performed in the ALU, an overflow detection is made. If an overflow is detected on any of the data paths, the destination accumulator overflow status bit (ACOVx) is set.
  - For the operations performed in the ALU low part, overflow is detected at bit position 15.
  - For the operations performed in the ALU high part, overflow is detected at bit position 31.

- ☐ For all instructions, the carry of the operation performed in the ALU high part is reported in the CARRY status bit. The CARRY status bit is always extracted at bit position 31.
- ☐ Independently on each data path, if SATD = 1 when an overflow is detected on the data path, a saturation is performed:
  - For the operations performed in the ALU low part, saturation values are 7FFFh and 8000h.
  - For the operations performed in the ALU high part, saturation values are 00 7FFFh and FF 8000h.

When C54CM = 1, this instruction is executed as if SATD is locally cleared to 0. Overflow is only detected and reported for the computation performed in the higher 24-bit datapath (overflow is detected at bit position 31).

- ☐ When C54CM = 1 and C16 = 1, the instruction behaves like a dual 16-bit instruction and the carry is not propagated at bit 15 in the D-unit ALU.

Status Bits Affected by C16, C54CM, SATD, SXMD

Affects ACOVx, CARRY

**Repeat** This instruction can be repeated.

Syntax	Description
HI(AC0) = HI(*AR3) + T0, LO(AC0) = LO(*AR3) - T0	Both instructions are performed in parallel. When the Lmem address is even (AR3 = even): The content of T0 is added to the content addressed by AR3 and the result is stored in AC0(39–16). The duplicated content of T0 is subtracted from the content addressed by AR3 + 1 and the result is stored in AC0(15–0).

### SUB

### **Dual 16-Bit Subtractions**

### **Syntax Characteristics**

No.	Syntax	Parallel Enable bit	Size	Cycles	Pipeline
[1]	HI(ACy) = HI(ACx) - HI(Lmem), LO(ACy) = LO(ACx) - LO(Lmem)	No	3	1	Х
[2]	HI(ACy) = HI(Lmem) - HI(ACx), LO(ACy) = LO(Lmem) - LO(ACx)	No	3	1	X
[3]	HI(ACx) = Tx - HI(Lmem), LO(ACx) = Tx - LO(Lmem)	No	3	1	Х
[4]	HI(ACx) = HI(Lmem) - Tx, LO(ACx) = LO(Lmem) - Tx	No	3	1	Х

**Description** These instructions perform two paralleled subtraction operations in one cycle.

The operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulator are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit datapath).

Status Bits Affected by C54CM, SATD, SXMD

Affects ACOVx, ACOVy, CARRY

**See Also** See the following other related instructions:

- ☐ Addition or Subtraction Conditionally
- ☐ Addition or Subtraction Conditionally with Shift
- Addition, Subtraction, or Move Accumulator Content Conditionally
- □ Dual 16-Bit Addition and Subtraction
- ☐ Dual 16-Bit Subtraction and Addition
- Subtract Conditionally
- Subtraction
- ☐ Subtraction with Parallel Store Accumulator Content to Memory

#### **Dual 16-Bit Subtractions**

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	HI(ACy) = HI(ACx) - HI(Lmem), LO(ACy) = LO(ACx) - LO(Lmem)	No	3	1	Х

#### **Opcode**

| 1110 | 1110 | AAAA | AAAI | SSDD | 001x

### **Operands**

ACx, ACy, Lmem

#### Description

This instruction performs two paralleled subtraction operations in one cycle. The operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulator are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit data path).

- ☐ The data memory operand dbl(Lmem) is divided into two 16-bit parts:
  - the lower part is used as one of the 16-bit operands of the ALU low part
  - the higher part is sign extended to 24 bits according to SXMD and is used in the ALU high part
- ☐ The data memory operand dbl(Lmem) addresses are aligned:
  - if Lmem address is even: most significant word = Lmem, least significant word = Lmem + 1
  - if Lmem address is odd: most significant word = Lmem, least significant word = Lmem - 1
- ☐ For each of the two computations performed in the ALU, an overflow detection is made. If an overflow is detected on any of the data paths, the destination accumulator overflow status bit (ACOVy) is set.
  - For the operations performed in the ALU low part, overflow is detected at bit position 15.
  - For the operations performed in the ALU high part, overflow is detected at bit position 31.
- ☐ For all instructions, the carry of the operation performed in the ALU high part is reported in the CARRY status bit. The CARRY status bit is always extracted at bit position 31.

- ☐ Independently on each data path, if SATD = 1 when an overflow is detected on the data path, a saturation is performed:
  - For the operations performed in the ALU low part, saturation values are 7FFFh and 8000h.
  - For the operations performed in the ALU high part, saturation values are 00 7FFFh and FF 8000h.

When C54CM = 1, this instruction is executed as if SATD is locally cleared to 0. Overflow is only detected and reported for the computation performed in the higher 24-bit datapath (overflow is detected at bit position 31).

- ☐ When C54CM = 1 and C16 = 1, the instruction behaves like a dual 16-bit instruction and the carry is not propagated at bit 15 in the D-unit ALU.
- ☐ When C54CM = 1 and C16 = 0, the instruction behaves like a single arithmetic instruction and the carry is propagated at bit 15 in the D-unit ALU.

**Status Bits** Affected by C16, C54CM, SATD, SXMD

> Affects ACOVy, CARRY

Repeat This instruction can be repeated.

Syntax	Description
HI(AC0) = HI(AC1) - HI(*AR3), $LO(AC0) = LO(AC1) - LO(*AR3)$	Both instructions are performed in parallel. When the Lmem address is even (AR3 = even): The content addressed by AR3 (sign extended to 24 bits) is subtracted from the content of AC1(39–16) and the result is stored in AC0(39–16). The content addressed by AR3 + 1 is subtracted from the content of AC1(15–0) and the result is stored in AC0(15–0).

#### **Dual 16-Bit Subtractions**

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable bit	Size	Cycles	Pipeline
[2]	HI(ACy) = HI(Lmem) - HI(ACx), LO(ACy) = LO(Lmem) - LO(ACx)	No	3	1	Х

#### **Opcode**

| 1110 | 1110 | AAAA | AAAI | SSDD | 010x

### **Operands**

ACx, ACy, Lmem

#### Description

This instruction performs two paralleled subtraction operations in one cycle. The operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulator are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit datapath).

- ☐ The data memory operand dbl(Lmem) is divided into two 16-bit parts:
  - the lower part is used as one of the 16-bit operands of the ALU low part
  - the higher part is sign extended to 24 bits according to SXMD and is used in the ALU high part
- ☐ The data memory operand dbl(Lmem) addresses are aligned:
  - if Lmem address is even: most significant word = Lmem, least significant word = Lmem + 1
  - if Lmem address is odd: most significant word = Lmem, least significant word = Lmem 1
- ☐ For each of the two computations performed in the ALU, an overflow detection is made. If an overflow is detected on any of the data paths, the destination accumulator overflow status bit (ACOVy) is set.
  - For the operations performed in the ALU low part, overflow is detected at bit position 15.
  - For the operations performed in the ALU high part, overflow is detected at bit position 31.
- ☐ For all instructions, the carry of the operation performed in the ALU high part is reported in the CARRY status bit. The CARRY status bit is always extracted at bit position 31.

- ☐ Independently on each data path, if SATD = 1 when an overflow is detected on the data path, a saturation is performed:
  - For the operations performed in the ALU low part, saturation values are 7FFFh and 8000h.
  - For the operations performed in the ALU high part, saturation values are 00 7FFFh and FF 8000h.

When C54CM = 1, this instruction is executed as if SATD is locally cleared to 0. Overflow is only detected and reported for the computation performed in the higher 24-bit datapath (overflow is detected at bit position 31).

- ☐ When C54CM = 1 and C16 = 1, the instruction behaves like a dual 16-bit instruction and the carry is not propagated at bit 15 in the D-unit ALU.
- ☐ When C54CM = 1 and C16 = 0, the instruction behaves like a single arithmetic instruction and the carry is propagated at bit 15 in the D-unit ALU.

**Status Bits** Affected by C 16, C54CM, SATD, SXMD

> Affects ACOVy, CARRY

Repeat This instruction can be repeated.

Syntax	Description
HI(AC0) = HI(*AR3) - HI(AC1), LO(AC0) = LO(*AR3) - LO(AC1)	Both instructions are performed in parallel. When the Lmem address is even (AR3 = even): The content of AC1(39–16) is subtracted from the content addressed by AR3 and the result is stored in AC0(39–16). The content of AC1(15–0) is subtracted from the content addressed by AR3 + 1 and the result is stored in AC0(15–0).

#### **Dual 16-Bit Subtractions**

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable bit	Size	Cycles	Pipeline
[3]	HI(ACx) = Tx - HI(Lmem), LO(ACx) = Tx - LO(Lmem)	No	3	1	Х

#### Opcode

#### **Operands**

ACx, Lmem, Tx

#### Description

This instruction performs two paralleled subtraction operations in one cycle. The operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulator are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit datapath).

- ☐ The temporary register Tx:
  - is used as one of the 16-bit operands of the ALU low part
  - is duplicated and, according to SXMD, sign extended to 24 bits to be used in the ALU high part
- ☐ The data memory operand dbl(Lmem) is divided into two 16-bit parts:
  - the lower part is used as one of the 16-bit operands of the ALU low part
  - the higher part is sign extended to 24 bits according to SXMD and is used in the ALU high part
- ☐ The data memory operand dbl(Lmem) addresses are aligned:
  - if Lmem address is even: most significant word = Lmem, least significant word = Lmem + 1
  - if Lmem address is odd: most significant word = Lmem, least significant word = Lmem 1
- ☐ For each of the two computations performed in the ALU, an overflow detection is made. If an overflow is detected on any of the data paths, the destination accumulator overflow status bit (ACOVx) is set.
  - For the operations performed in the ALU low part, overflow is detected at bit position 15.
  - For the operations performed in the ALU high part, overflow is detected at bit position 31.

- For all instructions, the carry of the operation performed in the ALU high part is reported in the CARRY status bit. The CARRY status bit is always extracted at bit position 31.
- ☐ Independently on each data path, if SATD = 1 when an overflow is detected on the data path, a saturation is performed:
  - For the operations performed in the ALU low part, saturation values are 7FFFh and 8000h.
  - For the operations performed in the ALU high part, saturation values are 00 7FFFh and FF 8000h.

When C54CM = 1, this instruction is executed as if SATD is locally cleared to 0. Overflow is only detected and reported for the computation performed in the higher 24-bit datapath (overflow is detected at bit position 31).

**Status Bits** Affected by C54CM, SATD, SXMD

> Affects ACOVx, CARRY

Repeat This instruction can be repeated.

Syntax	Description
HI(AC0) = T0 - HI(*AR3), LO(AC0) = T0 - LO(*AR3)	Both instructions are performed in parallel. When the Lmem address is even (AR3 = even): The content addressed by AR3 is subtracted from the content of T0 and the result is stored in AC0(39–16). The content addressed by AR3 + 1 is subtracted from the duplicated content of T0 and the result is stored in AC0(15–0).

#### **Dual 16-Bit Subtractions**

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable bit	Size	Cycles	Pipeline
[4]	HI(ACx) = HI(Lmem) - Tx, LO(ACx) = LO(Lmem) - Tx	No	3	1	Х

#### Opcode

#### **Operands**

ACx, Tx, Lmem

#### **Description**

This instruction performs two paralleled subtraction operations in one cycle. The operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulator are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit datapath).

- ☐ The temporary register Tx:
  - is used as one of the 16-bit operands of the ALU low part
  - is duplicated and, according to SXMD, sign extended to 24 bits to be used in the ALU high part
- ☐ The data memory operand dbl(Lmem) is divided into two 16-bit parts:
  - the lower part is used as one of the 16-bit operands of the ALU low part
  - the higher part is sign extended to 24 bits according to SXMD and is used in the ALU high part
- ☐ The data memory operand dbl(Lmem) addresses are aligned:
  - if Lmem address is even: most significant word = Lmem, least significant word = Lmem + 1
  - if Lmem address is odd: most significant word = Lmem, least significant word = Lmem 1
- ☐ For each of the two computations performed in the ALU, an overflow detection is made. If an overflow is detected on any of the data paths, the destination accumulator overflow status bit (ACOVx) is set.
  - For the operations performed in the ALU low part, overflow is detected at bit position 15.
  - For the operations performed in the ALU high part, overflow is detected at bit position 31.

- For all instructions, the carry of the operation performed in the ALU high part is reported in the CARRY status bit. The CARRY status bit is always extracted at bit position 31.
- ☐ Independently on each data path, if SATD = 1 when an overflow is detected on the data path, a saturation is performed:
  - For the operations performed in the ALU low part, saturation values are 7FFFh and 8000h.
  - For the operations performed in the ALU high part, saturation values are 00 7FFFh and FF 8000h.

When C54CM = 1, this instruction is executed as if SATD is locally cleared to 0. Overflow is only detected and reported for the computation performed in the higher 24-bit datapath (overflow is detected at bit position 31).

- ☐ When C54CM = 1 and C16 = 1, the instruction behaves like a dual 16-bit instruction and the carry is not propagated at bit 15 in the D-unit ALU.
- $\square$  When C54CM = 1 and C16 = 0, the instruction behaves like a single arithmetic instruction and the carry is propagated at bit 15 in the D-unit ALU.

**Status Bits** Affected by C16, C54CM, SATD, SXMD

> Affects ACOVx, CARRY

Repeat This instruction can be repeated.

Syntax	Description
HI(AC0) = HI(*AR3) – T0, LO(AC0) = LO(*AR3) – T0	Both instructions are performed in parallel. When the Lmem address is even (AR3 = even): The content of T0 is subtracted from the content addressed by AR3 and the result is stored in AC0(39–16). The duplicated content of T0 is subtracted from the content addressed by AR3 + 1 and the result is stored in AC0(15–0).

# **SUBADD**

### Dual 16-Bit Subtraction and Addition

# **Syntax Characteristics**

No.	Syntax	Parallel Enable bit	Size	Cycles	Pipeline
[1]	HI(ACx) = Smem - Tx, LO(ACx) = Smem + Tx	No	3	1	Х
[2]	HI(ACx) = HI(Lmem) - Tx, LO(ACx) = LO(Lmem) + Tx	No	3	1	Х

### **Description**

These instructions perform two paralleled subtraction and addition operations in one cycle.

The operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulator are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit datapath).

#### **Status Bits**

Affected by C54CM, SATD, SXMD

Affects ACOVx, ACOVy, CARRY

#### See Also

See the following other related instructions:

- ☐ Addition
- □ Dual 16-Bit Additions
- ☐ Dual 16-Bit Addition and Subtraction
- □ Dual 16-Bit Subtractions
- ☐ Subtraction

#### Dual 16-Bit Subtraction and Addition

### **Syntax Characteristics**

No.	Syntax	Parallel Enable bit	Size	Cycles	Pipeline
[1]	HI(ACx) = Smem - Tx, LO(ACx) = Smem + Tx	No	3	1	Х

#### Opcode

| 1101 1110 | AAAA AAAI | ssDD 1001

#### **Operands**

ACx, Smem, Tx

#### Description

This instruction performs two paralleled arithmetical operations in one cycle: a subtraction and addition. The operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulator are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit datapath).

- ☐ The data memory operand Smem:
  - is used as one of the 16-bit operands of the ALU low part
  - is duplicated and, according to SXMD, sign extended to 24 bits to be used in the ALU high part
- ☐ The temporary register Tx:
  - is used as one of the 16-bit operands of the ALU low part
  - is duplicated and, according to SXMD, sign extended to 24 bits to be used in the ALU high part
- ☐ For each of the two computations performed in the ALU, an overflow detection is made. If an overflow is detected on any of the data paths, the destination accumulator overflow status bit (ACOVx) is set.
  - For the operations performed in the ALU low part, overflow is detected at bit position 15.
  - For the operations performed in the ALU high part, overflow is detected at bit position 31.
- For all instructions, the carry of the operation performed in the ALU high part is reported in the CARRY status bit. The CARRY status bit is always extracted at bit position 31.

- ☐ Independently on each data path, if SATD = 1 when an overflow is detected on the data path, a saturation is performed:
  - For the operations performed in the ALU low part, saturation values are 7FFh and 8000h.
  - For the operations performed in the ALU high part, saturation values are 00 7FFFh and FF 8000h.

# Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if SATD is locally cleared to 0. Overflow is only detected and reported for the computation performed in the higher 24-bit datapath (overflow is detected at bit position 31).

Status Bits Affected by C54CM, SATD, SXMD

Affects ACOVx, CARRY

**Repeat** This instruction can be repeated.

Syntax	Description
HI(AC0) = *AR3 - T0, LO(AC0) = *AR3 + T0	Both instructions are performed in parallel. The content of T0 is subtracted from the content addressed by AR3 and the result is stored in AC0(39–16). The duplicated content of T0 is added to the duplicated content addressed by AR3 and the result is stored in AC0(15–0).

#### Dual 16-Bit Subtraction and Addition

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable bit	Size	Cycles	Pipeline
[2]	HI(ACx) = HI(Lmem) - Tx, LO(ACx) = LO(Lmem) + Tx	No	3	1	Х

### Opcode

| 1110 | 1110 | AAAA | AAAI | ssDD | 111x

#### **Operands**

ACx, Lmem, Tx

#### Description

This instruction performs two paralleled arithmetical operations in one cycle: a subtraction and addition. The operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulator are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit datapath).

- ☐ The temporary register Tx:
  - is used as one of the 16-bit operands of the ALU low part
  - is duplicated and, according to SXMD, sign extended to 24 bits to be used in the ALU high part
- ☐ The data memory operand dbl(Lmem) is divided into two 16-bit parts:
  - the lower part is used as one of the 16-bit operands of the ALU low part
  - the higher part is sign extended to 24 bits according to SXMD and is used in the ALU high part
- ☐ The data memory operand dbl(Lmem) addresses are aligned:
  - if Lmem address is even: most significant word = Lmem, least significant word = Lmem + 1
  - if Lmem address is odd: most significant word = Lmem, least significant word = Lmem 1
- ☐ For each of the two computations performed in the ALU, an overflow detection is made. If an overflow is detected on any of the data paths, the destination accumulator overflow status bit (ACOVx) is set.
  - For the operations performed in the ALU low part, overflow is detected at bit position 15.
  - For the operations performed in the ALU high part, overflow is detected at bit position 31.

- ☐ For all instructions, the carry of the operation performed in the ALU high part is reported in the CARRY status bit. The CARRY status bit is always extracted at bit position 31.
- ☐ Independently on each data path, if SATD = 1 when an overflow is detected on the data path, a saturation is performed:
  - For the operations performed in the ALU low part, saturation values are 7FFFh and 8000h.
  - For the operations performed in the ALU high part, saturation values are 00 7FFFh and FF 8000h.

# Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if SATD is locally cleared to 0. Overflow is only detected and reported for the computation performed in the higher 24-bit datapath (overflow is detected at bit position 31).

- ☐ When C54CM = 1 and C16 = 1, the instruction behaves like a dual 16-bit instruction and the carry is not propagated at bit 15 in the D-unit ALU.
- When C54CM = 1 and C16 = 0, the instruction behaves like a single arithmetic instruction and the carry is propagated at bit 15 in the D-unit ALU.

Status Bits Affected by C16, C54CM, SATD, SXMD

Affects ACOVx, CARRY

**Repeat** This instruction can be repeated.

Syntax	Description
HI(AC0) = HI(*AR3) - T0, LO(AC0) = LO(*AR3) + T0	Both instructions are performed in parallel. When the Lmem address is even (AR3 = even): The content of T0 is subtracted from the content addressed by AR3 and the result is stored in AC0(39–16). The duplicated content of T0 is added to the content addressed by AR3 + 1 and the result is stored in AC0(15–0).

# XCC

# Execute Conditionally

# **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	if (cond) execute(AD_Unit)	No	2	1	AD
[2]	if (cond) execute(D_Unit)	No	2	1	Х

# **Description**

These instructions evaluate a single condition defined by the cond field and allow you to control execution of all operations implied by the instruction or part of the instruction. See Table 1–3 for a list of conditions.

Instruction [1] allows you to control the entire execution flow from the address phase to the execute phase of the pipeline. Instruction [2] allows you to only control the execution flow from the execute phase of the pipeline. The use of a label, where control of the execute conditionally instruction ends, is optional.

These instructions may	v be executed alone.

☐ These instructions may be executed with two paralleled instr
--

These instructions may b	e executed	with the	instruction	with	which	it is
paralleled.						

٦.	These instruction	e may ha	executed w	ith the r	rovious	inetruction
	rnese instruction	is may be	executed w	nın me ı	nevious	mstruction.

These instructions may be executed with the previous instruction and two
paralleled instructions.

These instructions cannot be repeated.
--

These instructions cannot be used as the last instruction in a repeat loop
structure.

These instructions cannot control the execution of the following program
control instructions:

goto	(cond) goto	intr	blockrepeat	return_int
call	(cond) call	idle	(cond) execute(AD_unit)	
return	(cond) return	reset	(cond) execute(D_unit)	
trap	localrepeat	repeat	while (cond) repeat	

### **Status Bits**

Affected by ACOVx, CARRY, C54CM, M40, TCx

Affects **ACOV**x

# Execute Conditionally

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	if (cond) execute(AD_Unit)	No	2	1	AD
Opcode	e	'		1	c cccc
		<b>"</b>		į	C CCCC

The assembler selects the opcode depending on the instruction position in a paralleled pair.

#### **Operands**

cond

# **Description**

This instruction evaluates a single condition defined by the cond field and allows you to control the execution flow of an instruction, or instructions, from the address phase to the execute phase of the pipeline. See Table 1–3 for a list of conditions.

When this instruction moves into the address phase of the pipeline, the condition specified in the cond field is evaluated. If the tested condition is true, the conditional instruction(s) is read and executed; if the tested condition is false, the conditional instruction(s) is not read and program control is passed to the instruction following the conditional instruction(s) or to the program address defined by label. There is a 3-cycle latency for the condition testing.

☐ This instruction may be executed alone:

```
if(cond) execute(AD_unit)
  instruction_executes_conditionally
label:
```

This instruction may be executed with two paralleled instructions:

```
if(cond) execute(AD_unit)
  instruction_1_executes_conditionally
  || instruction_2_executes_conditionally
abel:
```

☐ This instruction may be executed with the instruction with which it is paralleled:

```
if(cond) execute(AD_unit)
    || instruction_executes_conditionally
label:
```

☐ This instruction may be executed with a previous instruction:

```
previous_instruction
    || if(cond) execute(AD_unit)
    instruction_executes_conditionally
label:
```

☐ This instruction may be executed with a previous instruction and two paralleled instructions:

This instruction cannot be used as the last instruction in a repeat loop structure.

This instruction cannot control the execution of the following program control instructions:

goto	(cond) goto	intr	blockrepeat	return_int
call	(cond) call	idle	(cond) execute(AD_unit)	
return (cond) return reset (cond) execute(D_unit)				
trap	localrepeat	repeat	while (cond) repeat	

### Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, the comparison of accumulators to 0 is performed as if M40 was set to 1.

### **Status Bits**

Affected by ACOVx, CARRY, C54CM, M40, TCx

Affects ACOVx

#### Repeat

This instruction cannot be repeated.

Syntax	Description
if (TC1) execute(AD_unit)	TC1 is equal to 1, the next instruction is executed (AR1 is incremented by 1).
mar(*AR1+)	The content of AC1 is added to the content addressed by AR1 + 1 (2021h) and the result is stored in AC1.
AC1 = AC1 + *AR1	and result to stored in 7.6 ii

Before		After	
AC1	00 0000 4300	AC1	00 0000 6321
TC1	1	TC1	1
CARRY	1	CARRY	0
AR1	0200	AR1	0201
200	2020	200	2020
201	2021	201	2021

Syntax	Description
if (TC1) execute(AD_unit)	TC1 is not equal to 1, the next instruction is not executed (AR1 is not
mar(*AR1+)	incremented). The content of AC1 is added to the content addressed by AR1 (2020h) and the result is stored in AC1.
AC1 = AC1 + *AR1	(

Before		After	
AC1	00 0000 4300	AC1	00 0000 6320
TC1	0	TC1	0
CARRY	1	CARRY	0
AR1	0200	AR1	0200
200	2020	200	2020
201	2021	201	2021

### Execute Conditionally

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size (	Cycles	Pipeline
[2]	if (cond) execute(D_Unit)	No	2	1	Х
Opcod	e	1003	1 011	0   100	c cccc
		1003	1 111	0   100	C CCCC
		1003	1 111	1   100	C CCCC

The assembler selects the opcode depending on the instruction position in a paralleled pair.

### **Operands**

cond

### **Description**

This instruction evaluates a single condition defined by the cond field and allows you to control the execution flow of an instruction, or instructions, from the execute phase of the pipeline. This instruction differs from instruction [1] because in this instruction operations performed in the address phase are always executed. See Table 1–3 for a list of conditions.

When this instruction moves into the execute phase of the pipeline, the condition specified in the cond field is evaluated. If the tested condition is true, the conditional instruction(s) is read and executed; if the tested condition is false, the conditional instruction(s) is not read and program control is passed to the instruction following the conditional instruction(s) or to the program address defined by label. There is a 0-cycle latency for the condition testing.

☐ This instruction may be executed alone:

```
if(cond) execute(D_unit)
  instruction_executes_conditionally
label:
```

This instruction may be executed with two paralleled instructions:

```
if(cond) execute(D_unit)
   instruction_1_executes_conditionally
   || instruction_2_executes_conditionally
label:
```

☐ This instruction may be executed with the instruction with which it is paralleled. When this instruction syntax is used and the instruction to be executed conditionally is a store-to-memory instruction, there is a 1-cycle latency for the condition setting.

```
if(cond) execute(D_unit)
    || instruction_executes_conditionally
label:
```

☐ This instruction may be executed with a previous instruction:

```
previous_instruction
    || if(cond) execute(D_unit)
    instruction_executes_conditionally
label:
```

☐ This instruction may be executed with a previous instruction and two paralleled instructions:

```
previous_instruction
    || if(cond) execute(D_unit)
    instruction_1_executes_conditionally
    || instruction_2_executes_conditionally
label:
```

This instruction cannot be used as the last instruction in a repeat loop structure.

When the instruction to be executed conditionally is an instruction to read data from memory, the data read operation is performed regardless of the condition and the read data is discarded at the execute phase if the condition is false.

This instruction cannot control the execution of the following program control instructions:

goto	(cond) goto	intr	blockrepeat	return_int
call	(cond) call	idle	(cond) execute(AD_unit)	
return (cond) return reset		(cond) execute(D_unit)		
trap localrepeat repeat		while (cond) repeat		

and an instruction to read data from I/O space.

### Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, the comparison of accumulators to 0 is performed as if M40 was set to 1.

Status Bits Affected by ACOVx, CARRY, C54CM, M40, TCx

Affects ACOVx

**Repeat** This instruction cannot be repeated.

# Example 1

Syntax	Description
if (TC1) execute(D_unit)	TC1 is equal to 1, the next instruction is executed (AR1 is incremented by 1).
mar(*AR1+)	The content of AC1 is added to the content addressed by AR1 + 1 (2021h) and the result is stored in AC1.
AC1 = AC1 + *AR1	and research steriou in 7 to 11

Before		After	
AC1	00 0000 4300	AC1	00 0000 6321
TC1	1	TC1	1
CARRY	1	CARRY	0
AR1	0200	AR1	0201
200	2020	200	2020
201	2021	201	2021

Syntax	Description
if (TC1) execute(D_unit)	TC1 is not equal to 1, the next instruction would not be executed; however,
mar(*AR1+)	since the next instruction is a pointer modification, AR1 is incremented by 1 in the address phase. The content of AC1 is added to the content addressed
AC1 = AC1 + *AR1	by AR1 + 1 (2021h) and the result is stored in AC1.

Before		After	
AC1	00 0000 4300	AC1	00 0000 6321
TC1	0	TC1	0
CARRY	1	CARRY	0
AR1	0200	AR1	0201
200	2020	200	2020
201	2021	201	2021

# BFXPA

### Expand Accumulator Bit Field

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = field_expand(ACx, k16)	No	4	1	Х

Opcode 0111 0110 kkkk kkkk kkkk FDDD 01SS

Operands ACx, dst, k16

**Description** This instruction performs a bit field manipulation in the D-unit shifter. When the

destination register (dst) is an A-unit register (ARx or Tx), a dedicated bus

carries the output of the D-unit shifter directly into dst.

The 16-bit field mask, k16, is scanned from the least significant bits (LSBs) to the most significant bits (MSBs). According to the bit set to 1 in the bit field mask, the 16 LSBs of the source accumulator (ACx) bits are extracted and

separated with 0 toward the MSBs. The result is stored in the dst.

Status Bits Affected by none

Affects none

**Repeat** This instruction can be repeated.

**See Also** See the following other related instructions:

□ Extract Accumulator Bit Field

#### **Example**

Syntax	Description
T2 = field_expand(AC0,#8024h)	Each bit of the unsigned 16-bit value (8024h) is scanned from the LSB to the MSB to test for a 1. If the bit is set to 1, the bit in AC0 is extracted and separated with 0 toward the MSB in T2; otherwise, the corresponding bit in AC0 is not extracted. The result is stored in T2.

#### Execution

#k16 (8024h) 1000 0000 0010 0100
ACO(15-0) 0010 1011 0110 0101
T2 1000 0000 0000 0100

Before After

AC0 00 2300 2B65 AC0 00 2300 2B65 T2 0000 T2 8004

## **BFXTR**

#### Extract Accumulator Bit Field

#### **Syntax Characteristics**

No.	Syntax				Paral Enable		Size	Cycles	Pipeline
[1]	dst = field_extract(ACx, k16)				No		4	1	Х
Opcod	le	0111	0110	kkkk	kkkk	kkkk	kk]	kk   FDI	DD 00SS

Operands ACx, dst, k16

**Description** This instruction performs a bit field manipulation in the D-unit shifter. When the

destination register (dst) is an A-unit register (ARx or Tx), a dedicated bus

carries the output of the D-unit shifter directly into dst.

The 16-bit field mask, k16, is scanned from the least significant bits (LSBs) to the most significant bits (MSBs). According to the bit set to 1 in the bit field mask, the corresponding 16 LSBs of the source accumulator (ACx) bits are extracted and packed toward the LSBs. The result is stored in the dst.

Status Bits Affected by none

Affects none

**Repeat** This instruction can be repeated.

**See Also** See the following other related instructions:

Expand Accumulator Bit Field

#### **Example**

Syntax	Description
T2 = field_extract(AC0,#8024h)	Each bit of the unsigned 16-bit value (8024h) is scanned from the LSB to the MSB to test for a 1. If the bit is set to 1, the corresponding bit in AC0 is extracted and packed toward the LSB in T2; otherwise, the corresponding bit in AC0 is not extracted. The result is stored in T2.

#### Execution

Before After

AC0 00 2300 55AA AC0 00 2300 55AA T2 0000 T2 0002

### **FIRSSUB**

Finite Impulse Response Filter, Antisymmetrical

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	firsn(Xmem, Ymem, coef(Cmem), ACx, ACy)	No	4	1	Х

#### Opcode

1000 0101 XXXM MMYY YMMM 11mm DDx1 DDU%

#### **Operands**

ACx, ACy, Cmem, Xmem, Ymem

### **Description**

This instruction performs two parallel operations: multiply and accumulate (MAC), and subtraction. The firsn() operation is executed:

$$ACy = ACy + (ACx * Cmem),$$
  
 $ACx = (Xmem << #16) - (Ymem << #16)$ 

The first operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of ACx(32–16) and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

- ☐ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.
- ☐ Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy.
- ☐ Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- ☐ When an addition overflow is detected, the accumulator is saturated according to SATD.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

The second operation subtracts the content of data memory operand Ymem, shifted left 16 bits, from the content of data memory operand Xmem, shifted left 16 bits.

- ☐ The operation is performed on 40 bits in the D-unit ALU.
- ☐ Input operands are sign extended to 40 bits according to SXMD.

	The shift o	peration is equivalent to the signed shift instruction.
	subtraction	detection and CARRY status bit depends on M40. The borrow bit is reported in the CARRY status bit; the borrow bit al complement of the CARRY status bit.
	When an o	verflow is detected, the accumulator is saturated according to
Со	mpatibility	with C54x devices (C54CM = 1)
C5		uction is executed with $M40 = 0$ , compatibility is ensured. When o overflow detection, report, and saturation is done after the on.
Aff	ected by	C54CM, FRCT, M40, SATD, SMUL, SXMD
Aff	ects	ACOVx, ACOVy, CARRY

# **Example**

Repeat

See Also

**Status Bits** 

Syntax	Description
firsn(*AR0, *AR1, coef(*CDP), AC0, AC1)	The content of AC0(32–16) multiplied by the content addressed by the coefficient data pointer register (CDP) is added to the content of AC1 and the result is stored in AC1. The content addressed by AR1 shifted left by 16 bits is subtracted from the content addressed by AR0 shifted left by 16 bits and the result is stored in AC0.

This instruction can be repeated.

See the following other related instructions:

☐ Finite Impulse Response Filter, Symmetrical

Before				After				
AC0	00	6900	0000	AC0	00	4500	0000	
AC1	00	0023	0000	AC1	FF	D8ED	3F00	
*AR0			3400	*AR0			3400	
*AR1			EF00	*AR1			EF00	
*CDP			A067	*CDP			A067	
ACOV0			0	ACOV0			0	
ACOV1			0	ACOV1			0	
CARRY			0	CARRY			0	
FRCT			0	FRCT			0	
SXMD			0	SXMD			0	

# **FIRSADD**

Finite Impulse Response Filter, Symmetrical

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	firs(Xmem, Ymem, coef(Cmem), ACx, ACy)	No	4	1	Х

#### Opcode

1000 0101 XXXM MMYY YMMM 11mm DDx0 DDU%

#### **Operands**

ACx, ACy, Cmem, Xmem, Ymem

### Description

This instruction performs two parallel operations: multiply and accumulate (MAC), and addition. The firs() operation is executed:

$$ACy = ACy + (ACx * Cmem),$$
  
 $ACx = (Xmem << #16) + (Ymem << #16)$ 

The first operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of ACx(32–16) and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

- ☐ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.
- ☐ Multiplication overflow detection depends on SMUL.
- ☐ The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy.
- ☐ Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- ☐ When an addition overflow is detected, the accumulator is saturated according to SATD.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

The second operation performs an addition operation between the content of data memory operand Xmem, shifted left 16 bits, and the content of data memory operand Ymem, shifted left 16 bits.

- ☐ The operation is performed on 40 bits in the D-unit ALU.
- ☐ Input operands are sign extended to 40 bits according to SXMD.

	The shift	operation	is equiva	lent to the	signed s	hift instruction.
--	-----------	-----------	-----------	-------------	----------	-------------------

Overflow detection and CARRY status bit depends on M40.

☐ When an overflow is detected, the accumulator is saturated according to SATD.

### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, no overflow detection, report, and saturation is done after the shifting operation.

Status Bits Affected by C54CM, FRCT, M40, SATD, SMUL, SXMD

Affects ACOVx, ACOVy, CARRY

**Repeat** This instruction can be repeated.

**See Also** See the following other related instructions:

☐ Finite Impulse Response Filter, Antisymmetrical

Syntax	Description
firs(*AR0, *AR1, coef(*CDP), AC0, AC1)	The content of AC0(32–16) multiplied by the content addressed by the coefficient data pointer register (CDP) is added to the content of AC1 and the result is stored in AC1. The content addressed by AR0 shifted left by 16 bits is added to the content addressed by AR1 shifted left by 16 bits and the result is stored in AC0.

Before				After			
AC0	00	6900	0000	AC0	00	2300	0000
AC1	00	0023	0000	AC1	FF	D8ED	3F00
*AR0			3400	*ARO			3400
*AR1			EF00	*AR1			EF00
*CDP			A067	*CDP			A067
ACOV0			0	ACOV0			0
ACOV1			0	ACOV1			0
CARRY			0	CARRY			1
FRCT			0	FRCT			0
SXMD			0	SXMD			0

**IDLE** 

Idle

# **Syntax Characteristics**

No. Syntax		Parallel Enable Bit	Size	Cycles	Pipeline
[1] idle		No	4	?	D
Opcode	0111 1010 xxxx	xxxx xxx	xx xx	xx x	x 110x
Operands	none				
Description	This instruction forces the program be a reset occurs. The power-down mode on a configuration register acces mechanism.	e that the proc	essor	operates i	n depends
Status Bits	Affected by INTM				
	Affects none				
Repeat	This instruction cannot be repeated.				

LMS

# Least Mean Square

# **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	Ims(Xmem, Ymem, ACx, ACy)	No	4	1	Χ

#### **Opcode**

1000 0110 XXXM MMYY YMMM DDDD 110x xxx%

### **Operands**

ACx, ACy, Xmem, Ymem

#### Description

This instruction performs two parallel operations in one cycle: multiply and accumulate (MAC), and addition. The instruction is executed:

The first operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Xmem, sign extended to 17 bits, and the content of data memory operand Ymem, sign extended to 17 bits.

- ☐ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.
- ☐ Multiplication overflow detection depends on SMUL.
- ☐ The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- ☐ When an addition overflow is detected, the accumulator is saturated according to SATD.

The second operation performs an addition between an accumulator content and the content of data memory operand Xmem shifted left by 16 bits.

- ☐ The operation is performed on 40 bits in the D-unit ALU.
- ☐ Input operands are sign extended to 40 bits according to SXMD.
- ☐ The shift operation is equivalent to the signed shift instruction.
- Overflow detection and CARRY status bit depends on M40.
- Rounding is performed according to RDM.

☐ When an overflow is detected on the result of the rounding, the accumulator is saturated according to SATD. Note that no overflow detection is performed on the intermediate result after the addition but before the rounding.

# Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, the rounding is performed without clearing the 16 lowest bits of ACx. The addition operation has no overflow detection, report, and saturation after the shifting operation.

**Status Bits** Affected by C54CM, FRCT, M40, RDM, SATD, SMUL, SXMD

> Affects ACOVx, ACOVy, CARRY

Repeat This instruction can be repeated.

Syntax	Description
Ims(*AR0, *AR1, AC0, AC1)	The content addressed by AR0 multiplied by the content addressed by AR1 is added to the content of AC1 and the result is stored in AC1. The content addressed by AR0 shifted left by 16 bits is added to the content of AC0. The result is rounded and stored in AC0.

Before				After			
AC0	00	1111	2222	AC0	00	2111	0000
AC1	00	1000	0000	AC1	00	1200	0000
*AR0			1000	*ARO			1000
*AR1			2000	*AR1			2000
ACOV0			0	ACOV0			0
ACOV1			0	ACOV1			0
CARRY			0	CARRY			0
FRCT			0	FRCT			0

**LMSF** 

# Least Mean Square (LMSF)

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	Imsf(Xmem, Ymem, ACx, ACy)	No	4	1	Χ

#### Opcode

1000 0111 XXXM MMYY YMMM SSDD 0110 0001

#### **Operands**

ACx, ACy, Xmem, Ymem, T3

#### Description

This instruction performs three parallel operations in one cycle. The operations are executed in the D-unit MAC and D-unit ALU. The instruction is executed:

```
ACx = T3 * (Ymem)

ACy = ACy + (Xmem) * (Ymem)

Xmem = HI(rnd(ACx + (Xmem) << #16))
```

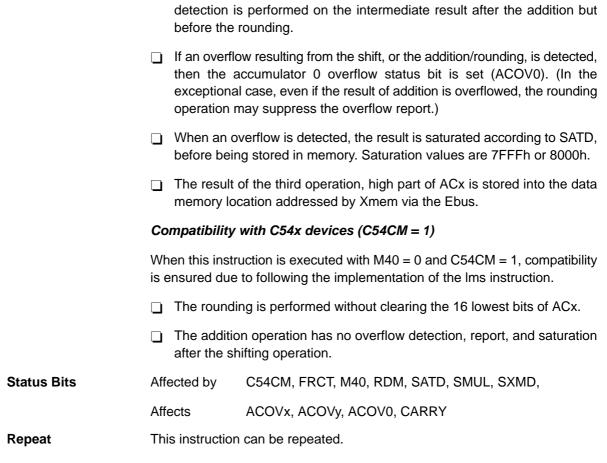
The first operation performs a multiplication in D-unit MAC1. The input operands of the multiplier are the content of data register T3 and the content of data memory operand Ymem. The implied T3 operand is sign extended to 17 bits in the MAC1. The data memory operand Ymem is addressed by DAGEN path Y by using Ymem addressing mode, driven on the CDB bus, and sign extended to 17 bits in the MAC1.

- ☐ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.
- ☐ Multiplication overflow detection depends on SMUL.
- ☐ The 32-bit result of the multiplication is sign extended to 40 bits.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ☐ When an overflow is detected, the accumulator is saturated according to SATD.

The second operation performs a multiplication and an addition in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand Xmem and the content of data memory operand Ymem. The data memory operand Xmem is addressed by DAGEN path X by using Xmem addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC2. The other data memory operand Ymem is addressed by DAGEN path Y by using the Ymem addressing mode, driven on data bus CDB, and sign extended to 17 bits in the MAC2.

	If FRC1 = 1, the output of the multiplier is shifted left by 1 bit.
	Multiplication overflow detection depends on SMUL.
	The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy.
	Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
	When an overflow is detected, the accumulator is saturated according to SATD. $ \label{eq:saturated} % \begin{center} \end{center} % \begin{center} ce$
the me	e third operation performs an addition between an accumulator content and content of data memory operand Xmem in the D-unit ALU. The data mory operand Xmem is driven on the DDB bus as described in the above cond operation, sign extended to 40 bits according to SXMD, shifted to the by 16 bits, and supplied to the D-unit ALU.
	The shift operation is identical to the arithmetic shift instruction. Therefore, an overflow detection, report, and saturation is done after the shifting operation.
	Overflow and CARRY detection are operated as M40 is locally set to 0.
	Addition overflow is always detected at bit position 31.
	Addition carry report in CARRY status bit is always extracted at bit position 31.
	A rounding is always performed on the result of the addition. The rounding operation depends on the RDM status value.
	When RDM is 0, the biased rounding to the infinite is performed. 2^15 is added to the 40-bit result of the accumulation.
	When RDM is 1, the unbiased rounding to the nearest is performed. According to the value of the 17 LSBs of the 40-bit result of accumulation, 2^15 is added as the following pseudo code description.
	$ if (2^15 < bit(15-0) < 2^16) \\ add 2^15 to the 40-bit result of the accumulation \\ else if (bit(15-0) == 2^15) \\ if (bit(16) == 1) \\ add 2^15 to the 40-bit result of the \\ accumulation $
П	When an overflow is detected on the result of the rounding, the

accumulator is saturated according to SATD. Note that no overflow



#### **Example**

	Syntax	Description
Ī	msf(*AR2-,*AR3+,AC0,AC1);	The product of the content addressed by AR2 and the content addressed by
- 1	SXM=1, FRCT=1;	AR3 is added to the content of AC1 and the result is stored in AC1. The
-   ;	assuming 4KW bank DARAM	content addressed by AR2, shifted to the left by 16 bits, is added to the con-
		tent of AC0. The result is rounded and stored in AC0.

#### Execution

```
T3[16:0] * ((Ymem)[16:0])) -> ACx[39:0]
ACy[39:0]+(Xmem)[16:0]*(Ymem))[16:0])) -> ACy[39:0]
HI(rnd(ACx[39:0]+((Xmem)<<#16))) -> Xmem
```

# LMSF Least Mean Square (Imsf)

Before				After			
AC0	00	3FFF	8000	AC0	00	0200	0000
AC1	00	0000	8000	AC1	00	0004	8000
Т3			8000	Т3			8000
XAR2		00	30FF	XAR2		00	30FE
XAR3		00	2000	XAR3		00	2001
Data memory							
2000h			FE00	2000h			FE00
30FFh			FF00	30FFh			3F00

5-178 Instruction Set Descriptions

# .LR

# Linear Addressing Qualifier

This instruction can be repeated.

# **Syntax Characteristics**

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline		
[1]	linear()			No	1	1	AD		
Opcode	e					100	1100		
Operar	nds	none							
Description		This instruction is an instruction qualifier that can be paralleled only with any instruction making an indirect Smem, Xmem, Ymem, Lmem, Baddr, or Cmem addressing or mar instructions. This instruction cannot be executed in parallel with any other types of instructions and it cannot be executed as a stand-alone instruction (assembler generates an error message).							
		pointer registe	uction is used in pa s used in the indirec r bits 0 to 8 were cle	t addressing m					
Status	Bits	Affected by	none						
		Affects	none						

Repeat

## MOV

### Load Accumulator from Memory

# **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = rnd(Smem << Tx)	No	3	1	Х
[2]	ACx = low_byte(Smem) << #SHIFTW	No	3	1	X
[3]	ACx = high_byte(Smem) << #SHIFTW	No	3	1	X
[4]	ACx = Smem <b>&lt;&lt; #16</b>	No	2	1	X
[5]	ACx = uns(Smem)	No	3	1	X
[6]	ACx = uns(Smem) << #SHIFTW	No	4	1	X
[7]	ACx = M40(dbl(Lmem))	No	3	1	X
[8]	LO(ACx) = Xmem, HI(ACx) = Ymem	No	3	1	Х

**Description** These instructions load a 16-bit signed constant, K16, the content of a memory

(Smem) location, the content of a data memory operand (Lmem), or the content of dual data memory operands (Xmem and Ymem) to a selected

accumulator (ACx).

Status Bits Affected by C54CM, M40, RDM, SATD, SXMD

Affects ACOVx

**See Also** See the following other related instructions:

- □ Load Accumulator from Memory with Parallel Store Accumulator Content to Memory
- □ Load Accumulator Pair from Memory
- ☐ Load Accumulator with Immediate Value
- ☐ Load Accumulator, Auxiliary, or Temporary Register from Memory
- ☐ Load Accumulator, Auxiliary, or Temporary Register with Immediate Value
- □ Load Auxiliary or Temporary Register Pair from Memory
- ☐ Multiply and Accumulate with Parallel Load Accumulator from Memory
- ☐ Multiply and Subtract with Parallel Load Accumulator from Memory

#### **Syntax Characteristics**

No.	Syntax		Para Enable		Size	Cycles	Pipeline
[1]	ACx = rnd(Smem << Tx)		No	١	3	1	Х
Opcod	e	110	1 1101	AAAA	. AA	AI   x%D	DD ss11

**Operands** ACx, Smem, Tx

Description This instruction loads the content of a memory (Smem) location shifted by the content of Tx to the accumulator (ACx):

☐ The input operand is sign extended to 40 bits according to SXMD.

The input operand is shifted by the 4-bit value in the D-unit shifter. The shift operation is equivalent to the signed shift instruction.

Rounding is performed in the D-unit shifter according to RDM, if the optional rnd keyword is applied to the input operand.

### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, no overflow detection, report, and saturation is done after the shifting operation. The 6 LSBs of Tx are used to determine the shift quantity. The 6 LSBs of Tx define a shift quantity within –32 to +31. When the value is between -32 to -17, a modulo 16 operation transforms the shift quantity to within -16 to -1.

**Status Bits** Affected by C54CM, M40, RDM, SATD, SXMD

> Affects **ACOV**x

Repeat This instruction can be repeated.

Syntax	Description
AC0 = *AR3 << T0	AC0 is loaded with the content addressed by AR3 shifted by the content of T0.

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACx = low_byte(Smem) << #SHIFTW	No	3	1	Х

Opcode | 1110 0001 AAAA AAAI | DDSH IFTW

Operands ACx, SHIFTW, Smem

**Description**This instruction loads the low-byte content of a memory (Smem) location shifted by the 6-bit value, SHIFTW, to the accumulator (ACx):

- ☐ The content of the memory location is sign extended to 40 bits according to SXMD.
- The input operand is shifted by the 6-bit value in the D-unit shifter. The shift operation is equivalent to the signed shift instruction.
- ☐ In this instruction, Smem **cannot** reference to a memory-mapped register (MMR). This instruction cannot access a byte within an MMR. If Smem is an MMR, the DSP sends a hardware bus-error interrupt (BERRINT) request to the CPU.

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, no overflow detection, report, and saturation is done after the shifting operation.

Status Bits Affected by C54CM, M40, SATD, SXMD

Affects ACOVx

**Repeat** This instruction can be repeated.

Syntax	Description
AC0 = low_byte(*AR3) << #31	The low-byte content addressed by AR3 is shifted left by 31 bits and loaded into AC0.

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ACx = high_byte(Smem) << #SHIFTW	No	3	1	Χ

Opcode | 1110 0010 AAAA AAAI | DDSH IFTW

Operands ACx, SHIFTW, Smem

**Description**This instruction loads the high-byte content of a memory (Smem) location shifted by the 6-bit value, SHIFTW, to the accumulator (ACx):

- ☐ The content of the memory location is sign extended to 40 bits according to SXMD.
- The input operand is shifted by the 6-bit value in the D-unit shifter. The shift operation is equivalent to the signed shift instruction.
- ☐ In this instruction, Smem **cannot** reference to a memory-mapped register (MMR). This instruction cannot access a byte within an MMR. If Smem is an MMR, the DSP sends a hardware bus-error interrupt (BERRINT) request to the CPU.

### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, no overflow detection, report, and saturation is done after the shifting operation.

Status Bits Affected by C54CM, M40, SATD, SXMD

Affects ACOVx

**Repeat** This instruction can be repeated.

Syntax	Description
AC0 = high_byte(*AR3) << #31	The high-byte content addressed by AR3 is shifted left by 31 bits and loaded into AC0.

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	ACx = Smem <b>&lt;&lt; #16</b>	No	2	1	Х

Opcode | 1011 00DD | AAAA AAAI

Operands ACx, Smem

**Description** This instruction loads the content of a memory (Smem) location shifted left by 16 bits to the accumulator (ACx):

☐ The input operand is sign extended to 40 bits according to SXMD.

☐ The shift operation is equivalent to the signed shift instruction.

☐ The input operand is shifted left by 16 bits according to M40.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, overflow detection, report, and saturation is done after the shifting operation

Status Bits Affected by C54CM, M40, SATD, SXMD

Affects ACOVx

**Repeat** This instruction can be repeated.

Syntax	Description
AC1 = *AR3+ << #16	The content addressed by AR3 shifted left by 16 bits is loaded into AC1. AR3 is
	incremented by 1.

Before				After			
AC1	00	0200	FC00	AC1	00	3400	0000
AR3			0200	AR3			0201
200			3400	200			3400

# **Syntax Characteristics**

No. Syntax		Parallel Enable Bit	Size	Cycles	Pipeline
[5] $ACx = uns(Sm)$	em <mark>)</mark>	No	3	1	Х
Opcode	1:	01 1111 AA	AA AA	AAI   xxI	DD 010u
Operands	ACx, Smem				
Description	This instruction loads the conteaccumulator (ACx):	nt of a memory	/ (Sme	em) loca	tion to the
	☐ The memory operand is exter	ided to 40 bits a	ccordir	ng to uns	
	If the optional uns keyword of the memory location is	• •		•	the content
	If the optional uns keywo	rd is not applied	I to the	input op	erand, the

- content of the memory location is sign extended to 40 bits according to SXMD.
- ☐ The load operation in the accumulator uses a dedicated path independent of the D-unit ALU, the D-unit shifter, and the D-unit MACs.

# Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by SXMD

Affects none

**Repeat** This instruction can be repeated.

Syntax	Description
AC0 = uns(*AR3)	The content addressed by AR3 is zero extended to 40 bits and loaded into AC0.

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	ACx = uns(Smem) << #SHIFTW	No	4	1	Х

Opcode

| 1111 1001 | AAAA AAAI | uxSH IFTW | xxDD 10xx

**Operands** 

ACx, SHIFTW, Smem

**Description** 

This instruction loads the content of a memory (Smem) location, shifted by the 6-bit value, SHIFTW, to the accumulator (ACx):

- ☐ The memory operand is extended to 40 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 40 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 40 bits according to SXMD.
- ☐ The input operand is shifted by the 6-bit value in the D-unit shifter. The shift operation is equivalent to the signed shift instruction.

### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, no overflow detection, report, and saturation is done after the shifting operation.

**Status Bits** 

Affected by

C54CM, M40, SATD, SXMD

Affects

**ACOV**x

Repeat

This instruction can be repeated.

Syntax	Description					
` ,	The content addressed by AR3 is zero extended to 40 bits, shifted left by 31 bits, and loaded into AC0.					

# **Syntax Characteristics**

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline	
[7]	ACx = M40(dbl)	<b>(</b> Lme	em <b>)</b> )			No	3	1	Х	
Opcod	e				1110	1101 AAA	A AA	AI   xxD	D 100g	
Operar	nds	AC	x, Lmem							
Description		This instruction loads the content of data memory operand (Lmem) to the accumulator (ACx):								
			The input of	perand is sign	extende	ed to 40 bits	accord	ing to SX	MD.	
		☐ The load operation in the accumulator uses a dedicated path independent of the D-unit ALU, the D-unit shifter, and the D-unit MACs.								
			☐ Status bit M40 is locally set to 1, if the optional M40 keyword is applied to the input operand.							
		Co	mpatibility	npatibility with C54x devices (C54CM = 1)						
		Wł	nen this instr	ruction is execu	uted with	M40 = 0, co	mpatib	ility is en	sured.	
Status	Bits	Affected by M40, SATD, SXMD								
		Aff	ects	ACOVx						
Repeat This instruction can be repeated.					ted.					

Syntax	Description
	The content (long word) addressed by AR3 and AR3 + 1 is loaded into AC0. Because this instruction is a long–operand instruction, AR3 is decremented by 2 after the execution.

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	LO(ACx) = Xmem, HI(ACx) = Ymem	No	3	1	Х

#### Opcode

1000 0001 XXXM MMYY YMMM 10DD

### **Operands**

ACx, Xmem, Ymem

#### **Description**

This instruction performs a dual 16-bit load of accumulator high and low parts. The operation is executed in dual 16-bit mode; however, it is independent of the 40-bit D-unit ALU. The 16 lower bits of the accumulator are separated from the higher 24 bits and the 8 guard bits are attached to the higher 16-bit datapath.

- ☐ The data memory operand Xmem is loaded as a 16-bit operand to the destination accumulator (ACx) low part. And, according to SXMD the data memory operand Ymem is sign extended to 24 bits and is loaded to the destination accumulator (ACx) high part.
- For the load operations in higher accumulator bits, overflow detection is performed at bit position 31. If an overflow is detected, the destination accumulator overflow status bit (ACOVx) is set.
- ☐ If SATD is 1 when an overflow is detected on the higher data path, a saturation is performed with saturation value of 00 7FFh.

### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, this instruction is executed as if SATD was locally cleared to 0.

#### **Status Bits**

Affected by

C54CM, M40, SATD, SXMD

Affects

**ACOV**x

### Repeat

This instruction can be repeated.

Syntax	Description
LO(AC0) = *AR3, HI(AC0) = *AR4	The content at the location addressed by AR4, sign extended to 24 bits, is loaded into AC0(39–16) and the content at the location addressed by AR3 is loaded into AC0(15–0).

# MOV::MOV

Load Accumulator from Memory with Parallel Store Accumulator Content to Memory

# **Syntax Characteristics**

No.	Syntax						Parallel Enable Bit	Size	Cycles	Pipeline		
[1]	ACy = Xmem << Ymem = <b>HI(</b> AC						No	4	1	Х		
Opcod	e			100	0 0111 X	MXX	MMYY YM	MM SS	SDD   110	x xxxx		
Operar	nds	ACx, ACy, T2, Xmem, Ymem										
Description		This instruction performs two operations in parallel: load and store.										
		The first operation loads the content of data memory operand Xmem shifted left by 16 bits to the accumulator ACy.										
			☐ The input operand is sign extended to 40 bits according to SXMD.									
		☐ The shift operation is equivalent to the signed shift instruction.										
		☐ The input operand is shifted left by 16 bits according to M40.										
	The second operation shifts the accumulator ACx by the content of T2 and stores ACx(31–16) to data memory operand Ymem. If the 16-bit value in T2 is not within $-32$ to $+31$ , the shift is saturated to $-32$ or $+31$ and the shift is performed with this value.											
	☐ The input operand is shifted in the D-unit shifter according to SXMD.											
		☐ After the shift, the high part of the accumulator, ACx(31–16), is stored to the memory location.										
		Compatibility with C54x devices (C54CM = 1)										
		this de to	s instru termine +31. V	uction is exe the shift qu Vhen the 10	is executed vecuted with (user the Grantity. The 66-bit value in the shift qua	C540 LSB 1 T2	CM = 1, the s of T2 defin is between	6 LSB ie a shi -32 to	s of T2 a ft quantity o –17, a	re used to within –32		
			are a	pplied to the e user, with ACy = Xme	and the SXMI e instruction r the following em << #16, (saturate(un	egar g syn	dless of the tax:			•		

☐ If the SST bit = 1 and the SXMD bit = 1, then only the saturate keyword is applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:  $ACy = Xmem \ll #16$ , Ymem = HI(saturate(ACx << T2)) **Status Bits** Affected by C54CM, M40, RDM, SATD, SST, SXMD Affects **ACOVy** Repeat This instruction can be repeated. See Also See the following other related instructions: □ Load Accumulator from Memory ■ Load Accumulator Pair from Memory □ Load Accumulator with Immediate Value ☐ Load Accumulator, Auxiliary, or Temporary Register from Memory ☐ Load Accumulator, Auxiliary, or Temporary Register with Immediate Value

Syntax	Description
AC0 = *AR3 << #16, *AR4 = HI(AC1 << T2)	Both instructions are performed in parallel. The content addressed by AR3 shifted left by 16 bits is stored in AC0. The content of AC1 is shifted by the content of T2, and AC1(31–16) is stored at the address of AR4.

Cycles Pipeline

Size

**Parallel** 

☐ Load Accumulator, Auxiliary, or Temporary Register with Immediate Value

☐ Multiply and Accumulate with Parallel Load Accumulator from Memory

☐ Multiply and Subtract with Parallel Load Accumulator from Memory

☐ Load Auxiliary or Temporary Register Pair from Memory

**Enable Bit** 

# MOV

No.

# Load Accumulator Pair from Memory

# **Syntax Characteristics**

**Syntax** 

[1]	pair(HI(ACx)) =	: Lme	em		No	3	1	Х
[2]	pair(LO(ACx))	= Lm	iem		No	3	1	Χ
Descri	ption			ons load the content of nulator pair, ACx and A		ory opei	rand (Lm	em) to the
Status	Bits	Aff	ected by	C54CM, M40, SATD,	SXMD			
		Aff	ects	ACOVx, ACOV(x + 1	)			
See Als	so	Se	e the followi	ng other related instruc	ctions:			
			Load Accu	mulator from Memory				
			Load Accu	mulator from Memory v	vith Parallel	Store Ac	cumulato	or Content
			Load Accu	mulator with Immediate	e Value			
			Load Accu	mulator, Auxiliary, or Te	emporary Re	gister fr	om Mem	ory

# Load Accumulator Pair from Memory

# **Syntax Characteristics**

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline	
[1]	pair(HI(ACx)) =	Lme	m			No	3	1	X	
Opcod	e				1110	1101 AA	AA AA	AI   00I	DD 1010	
Operar	nds	AC	x, Lmem							
Descri	ption	This instruction loads the 16 MSBs of data memory operand (Lmem) 24 MSBs of the destination accumulator (ACx) and loads the 16 LSBs data memory operand (Lmem) to the 24 MSBs of the destination accum AC(x+1).						SBs of the		
			sign extend	Bs and 16 LS ded to 24 bits or ACx and AC	and load	ed into the 2	4 MSB	s of the		
			performed	nd operation in at bit position or overflow sta	31. If ar	n overflow is				
			The valid AC2/AC3.	combination	of sour	ce accumul	ators	are AC0	/AC1 and	
		Со	mpatibility	with C54x de	vices (C	54CM = 1)				
		C5		uction is execu verflow detect			•	-		
Status	Bits	Affe	ected by	C54CM, M40	), SATD,	SXMD				
		Affe	ects	ACOVx, ACO	OV(x + 1)	)				
Repeat	t	Thi	This instruction can be repeated.							

Syntax	Description
	The 16 highest bits of the content at the location addressed by AR3 are loaded into AC2(31–16). The 16 lowest bits of the content at the location addressed by AR3 + 1 when the value in AR3 is even or AR3 – 1 when AR3 is odd are loaded into AC3(31–16). AR3 is incremented by 1.

## Execution

(Lmem[31:16]) -> ACx[39:16],  $(Lmem[15:0]) \rightarrow AC(x+1)[39:16]$ 

Before				After			
AC1	FF	FFFF	8000	AC1	00	1234	8000
AC2	00	1234	1234	AC2	FF	ABCD	8000
XAR3		00	2000	XAR3		00	2002
Data memory							
2000h			1234	2000h			1234
2001h			ABCD	2001h			ABCD

# Load Accumulator Pair from Memory

# **Syntax Characteristics**

No. Syntax		Parallel Enable Bit	Size	Cycles	Pipeline
[2] pair(LO(ACx)) =	= Lmem	No	3	1	X
Opcode	1110	1101 AAA	AA AA	AI 00D	D 1100
Operands	ACx, Lmem				
Description	This instruction loads the 16 MSBs of 16 LSBs of the destination accumulate data memory operand (Lmem) to the 1 AC(x+1).	or (ACx) and	loads	the 16 L	SBs of the
	☐ The 16 LSBs of the source accumand loaded into the 24 MSBs of according to the SXMD.		_		
	☐ For the load operation in higher a performed at bit position 31. If an accumulator overflow status bit is	overflow is			
	☐ When an overflow is detected of ACx[15]=1, a saturation is performed	•	-		
	☐ The valid combination of source AC2/AC3.	e accumula	ators a	are AC0	/AC1 and
	Compatibility with C54x devices (C5	54CM = 1)			
	When this instruction is executed with	M40 = 0, co	mpatib	ility is en	sured.
Status Bits	Affected by C54CM, SXMD				
	Affects none				
Repeat	This instruction can be repeated.				

Syntax	Description
. , , , ,	The 16 highest bits of the content at the location addressed by AR3 are loaded into AC0(15–0). The 16 lowest bits of the content at the location addressed by AR3 + 1 when the value in AR3 is even or AR3 – 1 when AR3 is odd are loaded into AC1(15–0).

## Execution

(Lmem[31:16]) -> ACx[15:0],  $(Lmem[15:0]) \rightarrow AC(x+1)[15:0]$ 

Before				After			
AC3	00	1234	5678	AC3	00	1234	ABCD
AC0	FF	FFFF	FFFF	AC0	FF	FFFF	1234
XAR5		00	2001	XAR5		00	1FFF
Data memory							
2000h			1234	2000h			1234
2001h			ABCD	2001h			ABCD

#### Load Accumulator with Immediate Value

## **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = K16 <b>&lt;&lt; #16</b>	No	4	1	Х
[2]	ACx = K16 <b>&lt;&lt; #</b> SHFT	No	4	1	X

**Description** These instructions load a 16-bit signed constant, K16, to a selected accumulator (ACx). **Status Bits** Affected by C54CM, M40, SATD, SXMD **ACOV**x Affects See Also See the following other related instructions: □ Load Accumulator from Memory ☐ Load Accumulator from Memory with Parallel Store Accumulator Content to Memory Load Accumulator Pair from Memory ☐ Load Accumulator, Auxiliary, or Temporary Register from Memory ☐ Load Accumulator, Auxiliary, or Temporary Register with Immediate Value ■ Load Auxiliary or Temporary Register Pair from Memory ☐ Multiply and Accumulate with Parallel Load Accumulator from Memory ☐ Multiply and Subtract with Parallel Load Accumulator from Memory

# Load Accumulator with Immediate Value

# **Syntax Characteristics**

No. Syntax			Parallel Enable Bit	Size	Cycles	Pipeline		
[1] ACx = K16 <<	: #16		No	4	1	X		
Opcode		0111 1010 KKKK	KKKK KK	KK KK	KKK   xxD	D 101x		
Operands	ACx, K16							
Description	This instruction loads the 16-bit signed constant, K16, shifted left by 16 bits to the accumulator (ACx):							
	☐ The 16-bi	t constant, K16, is sign e	extended to 4	0 bits	according	to SXMD.		
	☐ The shift of	operation is equivalent t	o the signed	shift ir	struction	-		
	☐ The input	operand is shifted left b	y 16 bits acc	ording	to M40.			
	Compatibility	y with C54x devices (C	54CM = 1)					
	When this inst	truction is executed with verflow detection, report,	M40 = 0, com	•	•			
Status Bits	Affected by	C54CM, M40, SATD,	SXMD					
	Affects	ACOVx						
Repeat	This instruction can be repeated.							
Example								
Syntax	Description							
AC0 = #-2 << #16	AC0 is loaded	with the signed 16-bit value	e (-2) shifted	left by	16 bits.			

#### Load Accumulator with Immediate Value

#### **Syntax Characteristics**

No.	Syntax				Paral Enable		Size	Cycles	Pipeline
[2]	ACx = K16 << #SHFT				No		4	1	Х
Opcod	e	0111	0101	KKKK	KKKK	KKKK	KK	KK xxI	DD SHFT

Operands ACx, K16, SHFT

**Description**This instruction loads the 16-bit signed constant, K16, shifted left by the 4-bit value, SHFT, to the accumulator (ACx):

☐ The 16-bit constant, K16, is sign extended to 40 bits according to SXMD.

☐ The input operand is shifted by the 4-bit value in the D-unit shifter. The shift operation is equivalent to the signed shift instruction.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, no overflow detection, report, and saturation is done after the shifting operation.

Status Bits Affected by C54CM, M40, SXMD

Affects none

**Repeat** This instruction can be repeated.

Syntax	Description
AC0 = #-2 << #15	AC0 is loaded with the signed 16-bit value (-2) shifted left by 15 bits.

# Load Accumulator, Auxiliary, or Temporary Register from Memory

## **Syntax Characteristics**

		Parallel			
No.	Syntax	Enable Bit	Size	Cycles	Pipeline
[1]	dst = Smem	No	2	1	Х
[2]	dst = uns(high_byte(Smem))	No	3	1	Χ
[3]	dst = uns(low_byte(Smem))	No	3	1	Х

**Description** These instructions load the content of a memory (Smem) location to a selected destination (dst) register. **Status Bits** Affected by M40, SXMD Affects none See the following other related instructions: See Also □ Load Accumulator from Memory ☐ Load Accumulator from Memory with Parallel Store Accumulator Content to Memory ■ Load Accumulator Pair from Memory ■ Load Accumulator with Immediate Value ☐ Load Accumulator, Auxiliary, or Temporary Register with Immediate Value ☐ Load Auxiliary or Temporary Register Pair from Memory Multiply and Accumulate with Parallel Load Accumulator from Memory ☐ Multiply and Subtract with Parallel Load Accumulator from Memory ☐ Store Accumulator, Auxiliary, or Temporary Register Content to Memory

# Load Accumulator, Auxiliary, or Temporary Register from Memory

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	
[1]	dst = Smem	No	2	1	Χ	

# Opcode

1010 FDDD AAAA AAAI

# Operands

dst, Smem

#### Description

This instruction loads the content of a memory (Smem) location to the destination (dst) register.

- ☐ When the destination register is an accumulator:
  - The content of the memory location is sign extended to 40 bits according to SXMD.
  - The load operation in the destination register uses a dedicated path independent of the D-unit ALU, the D-unit shifter, and the D-unit MACs.
- ☐ When the destination register is an auxiliary or temporary register:
  - The content of the memory location is sign extended to 16 bits.
  - The load operation in the destination register uses a dedicated path independent of the A-unit ALU.

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

#### **Status Bits**

Affected by M40, SXMD

Affects

## Repeat

This instruction can be repeated.

none

Syntax	Description
AR1 = *AR3+	AR1 is loaded with the content addressed by AR3. AR3 is incremented by 1.

Before		After	
AR1	FC00	AR1	3400
AR3	0200	AR3	0201
200	3400	200	3400

## Load Accumulator, Auxiliary, or Temporary Register from Memory

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	dst = uns(high_byte(Smem))	No	3	1	Х

#### **Opcode**

| 1101 | 1111 | AAAA | AAAI | FDDD | 000u

**Operands** 

dst, Smem

Description

This instruction loads the high-byte content of a memory (Smem) location to the destination (dst) register.

- ☐ When the destination register is an accumulator:
  - The memory operand is extended to 40 bits according to uns.
    - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 40 bits.
    - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 40 bits according to SXMD.
  - The load operation in the destination register uses a dedicated path independent of the D-unit ALU, the D-unit shifter, and the D-unit MACs.
- ☐ When the destination register is an auxiliary or temporary register:
  - The memory operand is extended to 16 bits according to uns.
    - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 16 bits.
    - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 16 bits regardless of SXMD.
  - The load operation in the destination register uses a dedicated path independent of the A-unit ALU.
- ☐ In this instruction, Smem **cannot** reference to a memory-mapped register (MMR). This instruction cannot access a byte within an MMR. If Smem is an MMR, the DSP sends a hardware bus-error interrupt (BERRINT) request to the CPU.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Affected by **Status Bits** M40, SXMD

> Affects none

Repeat This instruction can be repeated.

Syntax	Description
AC0 = uns(high_byte(*AR3))	The high-byte content addressed by AR3 is zero extended to 40 bits and loaded into AC0.

## Load Accumulator, Auxiliary, or Temporary Register from Memory

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	dst = uns(low_byte(Smem))	No	3	1	Х

#### **Opcode**

| 1101 | 1111 | AAAA | AAAI | FDDD | 001u

**Operands** 

dst, Smem

Description

This instruction loads the low-byte content of a memory (Smem) location to the destination (dst) register.

- When the destination register is an accumulator:
  - The memory operand is extended to 40 bits according to uns.
    - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 40 bits.
    - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 40 bits according to SXMD.
  - The load operation in the destination register uses a dedicated path independent of the D-unit ALU, the D-unit shifter, and the D-unit MACs.
- ☐ When the destination register is an auxiliary or temporary register:
  - The memory operand is extended to 16 bits according to uns.
    - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 16 bits.
    - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 16 bits regardless of SXMD.
  - The load operation in the destination register uses a dedicated path independent of the A-unit ALU.
- ☐ In this instruction, Smem **cannot** reference to a memory-mapped register (MMR). This instruction cannot access a byte within an MMR. If Smem is an MMR, the DSP sends a hardware bus-error interrupt (BERRINT) request to the CPU.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Affected by **Status Bits** M40, SXMD

> Affects none

Repeat This instruction can be repeated.

Syntax	Description
AC0 = uns(low_byte(*AR3))	The low-byte content addressed by AR3 is zero extended to 40 bits and loaded into AC0.

Load Accumulator, Auxiliary, or Temporary Register with Immediate Value

# **Syntax Characteristics**

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline					
[1]	dst = k4		Yes	2	1	X					
[2]	dst = -k4		Yes	2	1	X					
[3]	dst = K16		No	4	1	Х					
Descrip	otion	These instructions load a 4-bit ur representation of the 4-bit unsigned to a selected destination (dst) regi	d constant; or a 1			•					
Status	Bits	Affected by M40, SXMD									
		Affects none									
See Als	80	See the following other related ins	tructions:								
		☐ Load Accumulator from Memo	ory								
		<ul><li>Load Accumulator from Memo to Memory</li></ul>	ry with Parallel S	Store A	ccumulat	or Content					
		☐ Load Accumulator Pair from M	lemory								
		☐ Load Accumulator with Immed	liate Value								
		Load Accumulator, Auxiliary, o	r Temporary Re	gister f	rom Men	nory					
		☐ Load Auxiliary or Temporary R	Register Pair fror	n Mem	ory						
		☐ Multiply and Accumulate with	Parallel Load Ac	cumul	ator from	Memory					

☐ Multiply and Subtract with Parallel Load Accumulator from Memory

# Load Accumulator, Auxiliary, or Temporary Register with Immediate Value

# Syntax Characteristics

No.	Syntax																-		rall ble		+	S:	ze	_	`\/C	les		Din	eline
[1]	dst = k4																		es/				20	_	<b>ر</b> 1		'		X
Opcod	е																			0	01	1	1:	101	Ξ	kk	kk	: I	DDI
Operar	nds	dst	, k4	4																									
Descri	ption		is in: giste			uctic	on l	loa	ads	th	e 4	4-b	oit (	un	sig	gne	ed o	cor	nsta	an	t, ł	κ4,	to	the	e d	est	ina	ıtioı	n (ds
			Wł	/he	en	the	e de	lest	tina	atic	on	re	gis	te	r is	aı	n a	CC	um	nul	ato	or:							
				1	Th	he 4	4-b	oit c	con	nsta	ant	t, k	۲4,	is	ze	ero	ex	ter	nde	ed	to	40	bi	ts.					
				i	inc		pen		•												-								d pa D-ur
			Wł	/he	en	the	e de	lest	tina	atic	on	re	gis	te	r is	aı	n a	ux	ilia	ry	or	te	mp	ora	ary	re	gis	ter:	
					Th	he 4	4-b	oit c	con	nsta	ant	t, k	۲4,	is	ze	ero	ex	ter	nde	ed	to	16	bi	ts.					
						he l dep			•									tic	n ı	reç	gis	ter	us	es	а	dec	lica	ate	d pa
		Co	mp	oat	tib	bilit	ty v	wit	th C	C54	4x	de	evi	ce	es	(C	540	CM	1 =	1)									

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by M40

Affects none

**Repeat** This instruction can be repeated.

Syntax	Description
AC0 = #2	AC0 is loaded with the unsigned 4-bit value (2).

#### Load Accumulator, Auxiliary, or Temporary Register with Immediate Value

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	dst = -k4	Yes	2	1	Χ

Opcode 0011 111E kkkk FDDD

Operands

dst, k4

**Description** 

This instruction loads the 2s complement representation of the 4-bit unsigned constant, k4, to the destination (dst) register.

- ☐ When the destination register is an accumulator:
  - The 4-bit constant, k4, is negated in the I-unit, loaded into the accumulator, and sign extended to 40 bits before being processed by the D-unit as a signed constant.
  - The load operation in the destination register uses a dedicated path independent of the D-unit ALU, the D-unit shifter, and the D-unit MACs.
- ☐ When the destination register is an auxiliary or temporary register:
  - The 4-bit constant, k4, is zero extended to 16 bits and negated in the I-unit before being processed by the A-unit as a signed K16 constant.
  - The load operation in the destination register uses a dedicated path independent of the A-unit ALU.

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by M40

Affects none

**Repeat** This instruction can be repeated.

Syntax	Description
AC0 = #-2	AC0 is loaded with a 2s complement representation of the unsigned 4-bit value (2).

# Load Accumulator, Auxiliary, or Temporary Register with Immediate Value

# **Syntax Characteristics**

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[3]	dst = K16			No	4	1	Х
Opcod	e		0111 0110 KKK	K KKKK KKI	KK KF	KKK   FDI	D 10xx
Operar	nds	dst, K16					
<b>Description</b> This instruction loads the 16-bit signed constant, K16, to the register.				the destir	nation (dst)		
		_	destination register is ended to 40 bits acco		•	16-bit con	stant, K16,
			destination register is in the destination regnit ALU.	•	•	, ,	-
		Compatibility	with C54x devices	(C54CM = 1)			
		When this inst	ruction is executed w	rith M40 = 0, cc	mpatik	oility is er	sured.
Status	Bits	Affected by	M40, SXMD				
		Affects	none				
Repeat	1	This instructio	n can be repeated.				
Examp	le						

Syntax	Description
AC1 = #248	AC1 is loaded with the signed 16-bit value (248).

Before After

AC1 00 0200 FC00 AC1 00 0000 00F8

| 1110 1101 | AAAA AAAI | FDDD 111x

## MOV

**Opcode** 

Description

# Load Auxiliary or Temporary Register Pair from Memory

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	pair(TAx) = Lmem	No	3	1	Х

Operands Lmem, TAx

•

This instruction loads the 16 highest bits of data memory operand (Lmem) to the temporary or auxiliary register (TAx) and loads the 16 lowest bits of data memory operand (Lmem) to temporary or auxiliary register TA(x + 1):

- The load operation in the temporary or auxiliary register uses a dedicated path independent of the A-unit ALU.
- ☐ Valid auxiliary registers are AR0, AR2, AR4, and AR6.
- ☐ Valid temporary registers are T0 and T2.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by M40

Affects none

**Repeat** This instruction can be repeated.

**See Also** See the following other related instructions:

- ☐ Load Accumulator, Auxiliary, or Temporary Register from Memory
- ☐ Load Accumulator, Auxiliary, or Temporary Register with Immediate Value
- Modify Auxiliary or Temporary Register Content

Syntax	Description
pair(T0) = *AR2	The 16 highest bits of the content at the location addressed by AR2 are loaded into T0 and the 16 lowest bits of the content at the location addressed by AR2 + 1 are loaded into T1.

# Load CPU Register from Memory

# **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	BK03 = Smem	No	3	1	Х
[2]	BK47 = Smem	No	3	1	X
[3]	BKC = Smem	No	3	1	X
[4]	BSA01 = Smem	No	3	1	X
[5]	BSA23 = Smem	No	3	1	X
[6]	BSA45 = Smem	No	3	1	X
[7]	BSA67 = Smem	No	3	1	X
[8]	BSAC = Smem	No	3	1	X
[9]	BRC0 = Smem	No	3	1	X
[10]	BRC1 = Smem	No	3	1	X
[11]	CDP = Smem	No	3	1	X
[12]	CSR = Smem	No	3	1	X
[13]	<b>DP</b> = Smem	No	3	1	X
[14]	<b>DPH</b> = Smem	No	3	1	X
[15]	PDP = Smem	No	3	1	X
[16]	SP = Smem	No	3	1	X
[17]	SSP = Smem	No	3	1	X
[18]	TRN0 = Smem	No	3	1	X
[19]	TRN1 = Smem	No	3	1	X
[20]	RETA = dbl(Lmem)	No	3	5	X

Opcode See Table 5-1 (page 5-212).

Lmem, Smem Operands

#### **Description**

Instructions [1] through [19] load the content of a memory (Smem) location to the destination CPU register. This instruction uses a dedicated datapath independent of the A-unit ALU and the D-unit operators to perform the operation. The content of the memory location is zero extended to the bitwidth of the destination CPU register.

The operation is performed in the execute phase of the pipeline. There is a 3-cycle latency between PDP, DP, SP, SSP, CDP, BSAx, BKx, BRCx, and CSR loads and their use in the address phase by the A-unit address generator units or by the P-unit loop control management.

For instruction [10], when BRC1 is loaded, the block repeat save register (BRS1) is also loaded with the same value.

Instruction [20] loads the content of data memory operand (Lmem) to the 24-bit RETA register (the return address of the calling subroutine) and to the 8-bit CFCT register (active control flow execution context flags of the calling subroutine):

	the 8 highest bits of the RETA register.
	☐ The 16 lowest bits of Lmem are loaded into the 16 lowest bits of the RETA register.
	When instruction [20] is decoded, the CPU pipeline is flushed and the instruction is executed in 5 cycles, regardless of the instruction context.
Status Bits	Affected by none
	Affects none
Repeat	Instructions [13] and [20] cannot be repeated; all other instructions can be repeated.
See Also	See the following other related instructions:
	☐ Load CPU Register with Immediate Value

Table 5–1. Opcodes for Load CPU Register from Memory Instruction

No.	Syntax	Opcode
[1]	BK03 = Smem	1101 1100 AAAA AAAI 1001 xx10
[2]	BK47 = Smem	1101 1100 AAAA AAAI 1010 xx10
[3]	BKC = Smem	1101 1100 AAAA AAAI 1011 xx10
[4]	BSA01 = Smem	1101 1100 AAAA AAAI 0010 xx10
[5]	BSA23 = Smem	1101 1100 AAAA AAAI 0011 xx10
[6]	BSA45 = Smem	1101 1100 AAAA AAAI 0100 xx10
[7]	BSA67 = Smem	1101 1100 AAAA AAAI 0101 xx10
[8]	BSAC = Smem	1101 1100 AAAA AAAI 0110 xx10
[9]	BRC0 = Smem	1101 1100 AAAA AAAI x001 xx11
[10]	BRC1 = Smem	1101 1100 AAAA AAAI x010 xx11
[11]	CDP = Smem	1101 1100 AAAA AAAI 0001 xx10
[12]	CSR = Smem	1101 1100 AAAA AAAI x000 xx11
[13]	<b>DP</b> = Smem	1101 1100 AAAA AAAI 0000 xx10
[14]	<b>DPH</b> = Smem	1101 1100 AAAA AAAI 1100 xx10
[15]	PDP = Smem	1101 1100 AAAA AAAI 1111 xx10
[16]	SP = Smem	1101 1100 AAAA AAAI 0111 xx10
[17]	SSP = Smem	1101 1100 AAAA AAAI 1000 xx10
[18]	TRN0 = Smem	1101 1100 AAAA AAAI x011 xx11
[19]	TRN1 = Smem	1101 1100 AAAA AAAI x100 xx11
[20]	RETA = dbl(Lmem)	1110 1101 AAAA AAAI xxxx 011x

SWPU068E 5-212 Instruction Set Descriptions

# Load CPU Register with Immediate Value

## **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	<b>BK03</b> = k12	Yes	3	1	AD
[2]	<b>BK47</b> = k12	Yes	3	1	AD
[3]	<b>BKC</b> = k12	Yes	3	1	AD
[4]	<b>BRC0</b> = k12	Yes	3	1	AD
[5]	<b>BRC1</b> = k12	Yes	3	1	AD
[6]	<b>CSR</b> = k12	Yes	3	1	AD
[7]	<b>DPH</b> = k7	Yes	3	1	AD
[8]	<b>PDP</b> = k9	Yes	3	1	AD
[9]	<b>BSA01</b> = k16	No	4	1	AD
[10]	<b>BSA23</b> = k16	No	4	1	AD
[11]	<b>BSA45</b> = k16	No	4	1	AD
[12]	<b>BSA67</b> = k16	No	4	1	AD
[13]	<b>BSAC</b> = k16	No	4	1	AD
[14]	<b>CDP</b> = k16	No	4	1	AD
[15]	<b>DP</b> = k16	No	4	1	AD
[16]	<b>SP</b> = k16	No	4	1	AD
[17]	<b>SSP</b> = k16	No	4	1	AD

Opcode

See Table 5-2 (page 5-214).

**Operands** 

kx

**Description** 

These instructions load the unsigned constant, kx, to the destination CPU register. This instruction uses a dedicated datapath independent of the A-unit ALU and the D-unit operators to perform the operation. The constant is zero extended to the bitwidth of the destination CPU register.

For instruction [5], when BRC1 is loaded, the block repeat save register (BRS1) is also loaded with the same value.

The operation is performed in the address phase of the pipeline.

Status Bits

Affected by none

Affects none

Repeat

Instruction [15] cannot be repeated; all other instructions can be repeated.

See Also

See the following other related instructions:

Load CPU Register from Memory

Table 5–2. Opcodes for Load CPU Register with Immediate Value Instruction

No.	Syntax				Opc	ode			
[1]	<b>BK03</b> = k12		0001	011E	kkkk	kkkk	kkkk	0100	
[2]	<b>BK47</b> = k12		0001	011E	kkkk	kkkk	kkkk	0101	
[3]	<b>BKC</b> = k12		0001	011E	kkkk	kkkk	kkkk	0110	
[4]	<b>BRC0</b> = k12		0001	011E	kkkk	kkkk	kkkk	1001	
[5]	<b>BRC1</b> = k12		0001	011E	kkkk	kkkk	kkkk	1010	
[6]	<b>CSR</b> = k12		0001	011E	kkkk	kkkk	kkkk	1000	
[7]	<b>DPH</b> = k7		0001	011E	xxxx	xkkk	kkkk	0000	
[8]	<b>PDP</b> = k9		0001	011E	xxxk	kkkk	kkkk	0011	
[9]	<b>BSA01</b> = k16	0111	1000	kkkk	kkkk	kkkk	kkkk	xxx0	011x
[10]	<b>BSA23</b> = k16	0111	1000	kkkk	kkkk	kkkk	kkkk	xxx0	100x
[11]	<b>BSA45</b> = k16	0111	1000	kkkk	kkkk	kkkk	kkkk	xxx0	101x
[12]	<b>BSA67</b> = k16	0111	1000	kkkk	kkkk	kkkk	kkkk	xxx0	110x
[13]	<b>BSAC</b> = k16	0111	1000	kkkk	kkkk	kkkk	kkkk	xxx0	111x
[14]	<b>CDP</b> = k16	0111	1000	kkkk	kkkk	kkkk	kkkk	xxx0	010x
[15]	<b>DP</b> = k16	0111	1000	kkkk	kkkk	kkkk	kkkk	xxx0	000x
[16]	<b>SP</b> = k16	0111	1000	kkkk	kkkk	kkkk	kkkk	xxx1	000x
[17]	<b>SSP</b> = k16	0111	1000	kkkk	kkkk	kkkk	kkkk	xxx0	001x

5-214 Instruction Set Descriptions SWPU068E

# Load Extended Auxiliary Register from Memory

## **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	XAdst = <b>dbl(</b> Lmem <b>)</b>	No	3	1	Χ
Opcode Operan	'	1101 AAA	AA AA	AI XDD	D 1111

**Description**This instruction loads the lower 23 bits of the data addressed by data memory operand (Lmem) to the 23-bit destination register (XARx, XSP, XSSP, XDP, or

XCDP).

Status Bits Affected by none

Affects none

**Repeat** This instruction can be repeated.

**See Also** See the following other related instructions:

☐ Load Extended Auxiliary Register with Immediate Value

■ Modify Extended Auxiliary Register Content

■ Move Extended Auxiliary Register Content

☐ Store Extended Auxiliary Register Content to Memory

Syntax	Description
XAR1 = dbl(*AR3)	The 7 lowest bits of the content at the location addressed by AR3 and the 16 bits of
	the content at the location addressed by AR3 + 1 are loaded into XAR1.

Before		After	
XAR1	00 0000	XAR1	12 OFD3
AR3	0200	AR3	0200
200	3492	200	3492
201	0FD3	201	0FD3

# **AMOV**

#### Load Extended Auxiliary Register with Immediate Value

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	XAdst = k23	No	6	1	AD

**Opcode** | 1110 1100 | AAAA AAAI | 0DDD 1110

Operands k23, XAdst

**Description**This instruction loads a 23-bit unsigned constant (k23) into the 23-bit

destination register (XARx, XSP, XSSP, XDP, or XCDP). This operation is completed in the address phase of the pipeline by the A-unit address

generator. Data memory is not accessed.

The premodification or postmodification of the auxiliary register (ARx), the use of \*port(#K), and the use of the readport() or writeport() qualifier is not supported for this instruction. The use of auxiliary register offset operations is supported. If the corresponding bit (ARnLC) in status register ST2\_55 is set to 1, the circular buffer management also controls the result stored in XAdst.

Status Bits Affected by ST2\_55

Affects none

**Repeat** This instruction can be repeated.

**See Also** See the following other related instructions:

- Load Extended Auxiliary Register from Memory

- ☐ Store Extended Auxiliary Register Content to Memory

Syntax	Description
XAR0 = #7FFFFFh	The 23-bit value (7FFFFFh) is loaded into XAR0.

# Load Memory with Immediate Value

# **Syntax Characteristics**

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[1]	Smem = K8				No	3	1	Х
[2]	Smem = K16				No	4	1	Х
Opcode	e	K8			0 0110 A			
		K16	1111	1011 AAA	A AAAI K	KKK K	KKK KKK	K KKKK
Operar	nds	Kx, Smem						
Descri	ption	These instruct an 8-bit signed (Smem) locati	d consta	nt, K8, or a 1	6-bit signed o	constan	t, K16, to	a memory
		For instruction before being s			lue is always	signed	l extended	d to 16 bits
Status	Bits	Affected by	none					
		Affects	none					
Repeat	t .	Both instruction	ns [1] ar	nd [2] can be	repeated.			
See Als	so	See the follow	ring othe	r related instr	uctions:			

# Example

Syntax	Description
*(#0501h) = #248	The signed 16-bit value (248) is loaded to address 501h.

☐ Move Memory to Memory

Before		After	
0501	FC00	0501	F800

# .LK

#### Lock Access Qualifier

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	lock()	No	2	1	D
Opcod	le	01	00 01	101   111	.1 0010

# **Operands**

none

#### **Description**

This is an operand qualifier that can be paralleled with any of 13 instructions (listed below) which execute a read-modify-write operation to a specific memory operand. If the lock() qualifier is applied to any of 13 instructions, the lock signal is activated at the same cycle with the read request and the corresponding write request follows this read request. This means any memory request issued by other instructions cannot be located between this locked read and write request due to stall generation. This also provides a suitable interface with the OCP.

This operand qualifier cannot be executed:

□ Alone

☐ In parallel with instructions except the 13 lock instructions

Any of the 13 instructions using the lock() qualifier cannot be combined with any other user-defined parallelism instruction.

The 13 lock instructions which can be paralleled with the lock() qualifier are listed in the table below.

Number	Algebraic	Mnemonic
1	TC1 = bit(Smem, k4), bit(Smem, k4) = #1	BTSTSET k4, Smem, TC1
2	TC2 = bit(Smem, k4), bit(Smem, k4) = #1	BTSTSET k4, Smem, TC2
3	TC1 = bit(Smem, k4), bit(Smem, k4) = #0	BTSTCLR k4, Smem, TC1
4	TC2 = bit(Smem, k4), bit(Smem, k4) = #0	BTSTCLR k4, Smem, TC2
5	TC1 = bit(Smem, k4), cbit(Smem, k4)	BTSTNOT k4, Smem, TC1
6	TC2 = bit(Smem, k4), cbit(Smem, k4)	BTSTNOT k4, Smem, TC2
7	bit(Smem, src) = #1	BSET src, Smem
8	bit(Smem, src) = #0	BCLR src, Smem
9	cbit(Smem, src)	BNOT src, Smem
10	Smem = Smem & k16	AND k16, Smem
11	Smem = Smem   k16	OR k16, Smem
12	Smem = Smem ^ k16	XOR k16, Smem
13	Smem = Smem + k16	ADD k16, Smem

Any of the 13 instructions with the lock() qualifier is not allowed in the conditional execution context which is applied by "if(cond) execute(D\_unit)" instruction due to OCP compliance. The cases below are illegal and rejected by the code-gen tools:

```
if(cond execute(D unit)
TC1=bit(*ar2+, #2), bit(*ar2+, #2)=#1 | lock()
instruction || if(cond) execute(D_unit)
TC1=bit(*ar2+, #2), bit(*ar2+, #2)=#1 | lock()
```

## Compatibility with C54x devices (C54CM = 1)

None.

**Status Bits** Affected by none

> Affects none

Repeat This instruction can be repeated.

**Example** TC1=bit(\*ar2+, #2), bit(\*ar2+, #2)=#1 | lock()

Before			After		
XAR2	00	1780	XAR2	00	1781
Data memory					
1780h		FE00	1780h		FE04
1781h		3800	1781h		3800
TC1		х			0

# DELAY

#### Memory Delay

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	delay(Smem)	No	2	1	Х

# Opcode | 1011 0110 | AAAA AAAI

#### **Operands**

#### Smem

#### **Description**

This instruction copies the content of the memory (Smem) location into the next higher address (Smem + 1). When the data is copied, the content of the addressed location remains the same. A dedicated datapath is used to make this memory move.

When this instruction is executed, the two address register arithmetic units ARAU X and Y, of the A-unit data address generator unit, are used to compute the two addresses Smem and Smem + 1. The address generation is not affected by circular addressing; if Smem points to the end of a circular buffer, Smem + 1 will point to an address outside the circular buffer.

The soft dual memory addressing mode mechanism cannot be applied to this instruction. This instruction cannot use the \*port(#k16) addressing mode or be paralleled with the readport() or writeport() operand qualifier.

This instruction cannot be used for accesses to I/O space. Any illegal access to I/O space generates a hardware bus-error interrupt (BERRINT) to be handled by the CPU.

#### **Status Bits**

Affected by none

Affects none

#### Repeat

This instruction can be repeated.

Syntax	Description
delay(*AR1+)	The content addressed by AR1 is copied to the next higher address, AR1 + 1. AR1 is incremented by 1.

Before		After	
AR1	0200	AR1	0201
200	3400	200	3400
201	0D80	201	3400
202	2030	202	2030

#### mmap

#### Memory-Mapped Register Access Qualifier

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	mmap()	No	1	1	D

#### **Opcode**

1001 1000

# Operands Description

none

This is an operand qualifier that can be paralleled with any instruction making a Smem or Lmem direct memory access (dma). This operand qualifier allows you to locally prevent the dma access from being relative to the data stack pointer (SP) or the local data page register (DP). It forces the dma access to be relative to the memory-mapped register (MMR) data page start address, 00 0000h.

This operand qualifier cannot be executed:

- as a stand-alone instruction (assembler generates an error message)
- in parallel with instructions not embedding an Smem or Lmem data memory operand
- in parallel with instructions loading or storing a byte to a register (see Load Accumulator, Auxiliary, or Temporary Register from Memory instructions [2] and [3]; Load Accumulator from Memory instructions [2] and [3]; and Store Accumulator, Auxiliary, or Temporary Register Content to Memory instructions [2] and [3])

The MMRs are mapped as 16-bit data entities between addresses 0h and 5Fh. The scratch-pad memory that is mapped between addresses 60h and 7Fh of each main data pages of 64K words cannot be accessed through this mechanism.

Any instruction using the mmap() modifier cannot be combined with any other user-defined parallelism instruction.

#### **Status Bits**

Affected by none

Affects none

#### Repeat

This instruction can be repeated.

Syntax	Description
	AC0_L is a keyword representing AC0(15–0). The content of AC0(15–0) is copied into T2.

## AMAR

#### Modify Auxiliary Register Content

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	mar(Smem)	No	2	1	AD

Opcode | 1011 0100 AAAA AAAI

Operands Smem

**Description**This instruction performs, in the A-unit address generation units, the auxiliary

register modification specified by Smem as if a word single data memory operand access was made. The operation is performed in the address phase

of the pipeline; however, data memory is not accessed.

If the destination register is an auxiliary register and the corresponding bit (ARnLC) in status register ST2\_55 is set to 1, the circular buffer management

controls the result stored in the destination register.

Compatibility with C54x devices (C54CM = 1)

In the translated code section, the mar() instruction must be executed with

C54CM set to 1.

When circular modification is selected for the destination auxiliary register, this instruction modifies the selected destination auxiliary register by using BK03

as the circular buffer size register; BK47 is not used.

Status Bits Affected by ST2\_55

Affects none

**Repeat** This instruction can be repeated.

mar(*AR3+)	The content of AR3 is incremented by 1.
Syntax	Description
Example	
	☐ Parallel Modify Auxiliary Register Contents
	☐ Modify Extended Auxiliary Register Content by Subtraction
	☐ Modify Extended Auxiliary Register Content by Addition
	☐ Modify Extended Auxiliary Register Content
	☐ Modify Auxiliary Register Content with Parallel Multiply and Subtract
	Modify Auxiliary Register Content with Parallel Multiply and Accumulate
	☐ Modify Auxiliary Register Content with Parallel Multiply
	☐ Modify Auxiliary or Temporary Register Content by Subtraction
	☐ Modify Auxiliary or Temporary Register Content by Addition
	☐ Modify Auxiliary or Temporary Register Content
See Also	See the following other related instructions:

# AMAR::MPY

Modify Auxiliary Register Content with Parallel Multiply

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	<pre>mar(Xmem), ACx = M40(rnd(uns(Ymem) * uns(coef(Cmem))))</pre>	No	4	1	Х

#### Opcode

| 1000 0010 | XXXM MMYY | YMMM 11mm | uuxx DDg%

#### **Operands**

ACx, Cmem, Xmem, Ymem

#### **Description**

This instruction performs two parallel operations in one cycle: modify auxiliary register (MAR) and multiply. The operations are executed in the two D-unit MACs.

The first operation performs an auxiliary register modification. The auxiliary register modification is specified by the content of data memory operand Xmem.

The second operation performs a multiplication in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Ymem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, extended to 17 bits.

- ☐ Input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- ☐ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- ☐ The 32-bit result of the multiplication is sign extended to 40 bits.
- □ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVx) is set.
- ☐ When an overflow is detected, the accumulator is saturated according to SATD.

	_	struction provides the option to locally set M40 to 1 for the execution instruction, if the optional M40 keyword is applied to the instruction.
	on som	s instruction, the Cmem operand is accessed through the BB bus; ne C55x-based devices, the BB bus is only connected to internal ry and not to external memory. To prevent the generation of a bus he Cmem operand must not be mapped on external memory.
	while allow	flow can also disable the usage of the corresponding MAC unit, ving the modification of auxiliary registers in the three address units through the following instructions:
	■ ma	ar(Xmem)
	■ ma	ar(Ymem)
	<b>■</b> ma	ar(Cmem)
Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
	Affects	ACOVx
Repeat	This instruc	ction can be repeated.
See Also	See the fol	llowing other related instructions:
	☐ Modify	Auxiliary Register Content
	☐ Modify	Auxiliary Register Content with Parallel Multiply and Accumulate
	☐ Modify	Auxiliary Register Content with Parallel Multiply and Subtract
	☐ Multiply	у
Example		
Syntax		Description
mar(*AR3+), AC0 = uns(*AR4) * uns(coef(*CDP))		Both instructions are performed in parallel. AR3 is incremented by 1. The unsigned content addressed by AR4 is multiplied by the unsigned content addressed by the coefficient data pointer register (CDP) and the

result is stored in ACO.

### AMAR::MAC

Modify Auxiliary Register Content with Parallel Multiply and Accumulate

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	mar(Xmem), ACx = M40(rnd(ACx + (uns(Ymem) * uns(coef(Cmem)))))	No	4	1	Х
[2]	mar(Xmem), ACx = M40(rnd((ACx >> #16) + (uns(Ymem) * uns(coef(Cmem)))))	No	4	1	Х

**Description** These instructions perform two parallel operations in one cycle: modify

auxiliary register (MAR), and multiply and accumulate (MAC). The operations

are executed in the two D-unit MACs.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVx, ACOVy

**See Also** See the following other related instructions:

■ Modify Auxiliary Register Content

☐ Modify Auxiliary Register Content with Parallel Multiply

☐ Modify Auxiliary Register Content with Parallel Multiply and Subtract

■ Multiply and Accumulate

### Modify Auxiliary Register Content with Parallel Multiply and Accumulate

### Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	mar(Xmem), ACx = M40(rnd(ACx + (uns(Ymem) * uns(coef(Cmem)))))	No	4	1	Х

### **Opcode**

1000 0011 XXXM MMYY YMMM 11mm uuxx DDg%

### **Operands**

ACx, Cmem, Xmem, Ymem

### Description

This instruction performs two parallel operations in one cycle: modify auxiliary register (MAR), and multiply and accumulate (MAC). The operations are executed in the two D-unit MACs.

The first operation performs an auxiliary register modification. The auxiliary register modification is specified by the content of data memory operand Xmem.

The second operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Ymem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, extended to 17 bits.

- Input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- ☐ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- ☐ The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVx) is set.

SATD.
 This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.
 For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal

☐ When an overflow is detected, the accumulator is saturated according to

Each data flow can also disable the usage of the corresponding MAC unit, while allowing the modification of auxiliary registers in the three address generation units through the following instructions:

memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

- mar(Xmem)
- mar(Ymem)
- mar(Cmem)

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVx

**Repeat** This instruction can be repeated.

Syntax	Description
mar(*AR3+), AC0 = AC0 + (uns(*AR4) * uns(coef(*CDP)))	Both instructions are performed in parallel. AR3 is incremented by 1. The unsigned content addressed by AR4 multiplied by the unsigned content addressed by the coefficient data pointer register (CDP) is added to the content of AC0 and the result is stored in AC0.

### Modify Auxiliary Register Content with Parallel Multiply and Accumulate

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	mar(Xmem), ACx = M40(rnd((ACx >> #16) + (uns(Ymem) * uns(coef(Cmem)))))	No	4	1	Х

### **Opcode**

1000 0100 XXXM MMYY YMMM 01mm uuxx DDg%

### **Operands**

ACx, Cmem, Xmem, Ymem

### Description

This instruction performs two parallel operations in one cycle: modify auxiliary register (MAR), and multiply and accumulate (MAC). The operations are executed in the two D-unit MACs.

The first operation performs an auxiliary register modification. The auxiliary register modification is specified by the content of data memory operand Xmem.

The second operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Ymem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, extended to 17 bits.

- ☐ Input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- ☐ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.
- ☐ Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx shifted right by 16 bits. The shifting operation is performed with a sign extension of source accumulator ACx(39).
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.

- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVx) is set.
- ☐ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Each data flow can also disable the usage of the corresponding MAC unit, while allowing the modification of auxiliary registers in the three address generation units through the following instructions:

- mar(Xmem)
- mar(Ymem)
- mar(Cmem)

**Status Bits** Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

> Affects **ACOV**x

Repeat This instruction can be repeated.

Syntax	Description
mar(*AR2+), AC0 = ((AC0 >> #16) + (uns(*AR1) * uns(coef(*CDP))))	Both instructions are performed in parallel. AR2 is incremented by 1. The unsigned content addressed by AR1 multiplied by the unsigned content addressed by the coefficient data pointer register (CDP) is added to the content of AC0 shifted right by 16 bits and the result is stored in AC0. An overflow is detected in AC0.

Before				Aft	ter			
AC0	00	6900	0000	AC	0	00	95C0	9200
AC1	00	0023	0000	AC:	1	00	0023	0000
*AR1			EF00	*AI	R1			EF00
AR2			0201	AR2	2			0202
*CDP			A067	*CI	OP			A067
ACOV0			0	ACC	0VC			1
ACOV1			0	ACC	OV1			0
CARRY			0	CAI	RRY			0
M40			0	M4 (	0			0
FRCT			0	FRO	CT			0
SATD			0	SAT	ΓD			0

### AMAR::MAS

Modify Auxiliary Register Content with Parallel Multiply and Subtract

### **Syntax Characteristics**

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
mar(Xmem),	No	4	1	Х
	•	Syntax Enable Bit mar(Xmem), No	SyntaxEnable BitSizemar(Xmem),No4	SyntaxEnable BitSizeCyclesmar(Xmem),No41

### **Opcode**

1000 0101 XXXM MMYY YMMM 00mm uuxx DDg%

### **Operands**

ACx, Cmem, Xmem, Ymem

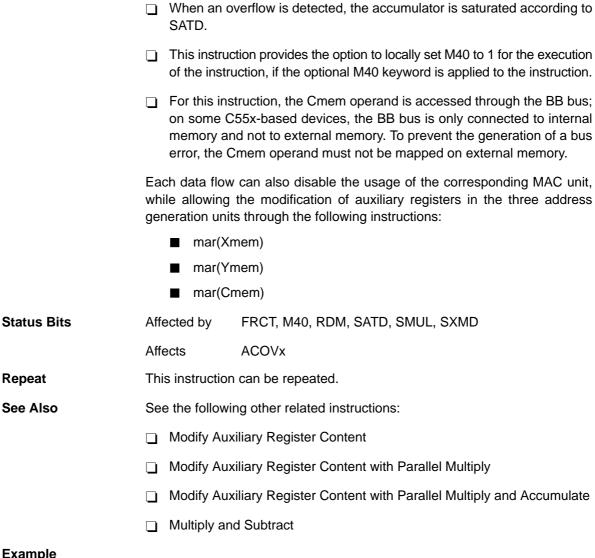
### Description

This instruction performs two parallel operations in one cycle: modify auxiliary register (MAR), and multiply and subtract (MAS). The operations are executed in the two D-unit MACs.

The first operation performs an auxiliary register modification. The auxiliary register modification is specified by the content of data memory operand Xmem.

The second operation performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Ymem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, extended to 17 bits.

- Input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- ☐ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- ☐ The 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACx.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVx) is set.



Syntax	Description
mar(*AR3+), AC0 = AC0 - (uns(*AR4) * uns(coef(*CDP)))	Both instructions are performed in parallel. AR3 is incremented by 1. The unsigned content addressed by AR4 multiplied by the unsigned content addressed by the coefficient data pointer register (CDP) is subtracted from the content of AC0 and the result is stored in AC0.

# **AMOV**

# Modify Auxiliary or Temporary Register Content

# **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	mar(TAy = TAx)	No	3	1	AD
[2]	mar(TAx = P8)	No	3	1	AD
[3]	mar(TAx = D16)	No	4	1	AD

Description	These instructions perform, in the A-unit address generation units:
	a move from auxiliary or temporary register TAx to auxiliary or temporary register TAy
	a load in the auxiliary or temporary registers TAx of a program address defined by a program address label assembled into P8
	a load in the auxiliary or temporary registers TAx of the absolute data address signed constant D16
	The operation is performed in the address phase of the pipeline, however data memory is not accessed.
Status Bits	Affected by none
	Affects none
See Also	See the following other related instructions:
	☐ Load Auxiliary or Temporary Register from Memory
	☐ Modify Auxiliary Register Content
	☐ Modify Auxiliary or Temporary Register Content by Addition
	☐ Modify Auxiliary or Temporary Register Content by Subtraction
	☐ Modify Extended Auxiliary Register Content
	☐ Modify Extended Auxiliary Register Content by Addition
	☐ Modify Extended Auxiliary Register Content by Subtraction

### Modify Auxiliary or Temporary Register Content

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	mar(TAy = TAx)	No	3	1	AD

 Opcode
 0001
 010E
 FSSS
 xxxxx
 FDDD
 0001

 0001
 010E
 FSSS
 xxxxx
 FDDD
 1001

The assembler selects the opcode depending on the instruction position in a

paralleled pair.

**Operands** TAx, TAy

**Description** This instruction performs, in the A-unit address generation units, a move from

the auxiliary or temporary register TAx to auxiliary or temporary register TAy. The operation is performed in the address phase of the pipeline; however, data

memory is not accessed.

Status Bits Affected by none

Affects none

**Repeat** This instruction can be repeated.

Example 1

Syntax	Description
mar(AR0 = AR1)	The content of AR1 is copied to AR0.

Syntax	Description
mar(T0 = T1)	The content of T1 is copied to T0.

### Modify Auxiliary or Temporary Register Content

### **Syntax Characteristics**

No.	Syntax		Par Enab		Size	Cycl	es l	Pipeline
[2]	mar(TAx = P8)		N	0	3	1		AD
Opcod	e	000	1 010E	PPF	P PI	PPP E	DDD	0101

The assembler selects the opcode depending on the instruction position in a paralleled pair.

0001 010E PPPP PPPP FDDD 1101

Operands TAx, P8

**Description** This instruction performs, in the A-unit address generation units, a load in the

auxiliary or temporary registers TAx of a program address defined by a program address label assembled into P8. The operation is performed in the address phase of the pipeline; however, data memory is not accessed.

Status Bits Affected by none

Affects none

**Repeat** This instruction can be repeated.

### Example 1

Syntax	Description
mar(AR0 = #255)	The unsigned 8-bit value (255) is copied to AR0.

Syntax	Description
mar(T0 = #255)	The unsigned 8-bit value (255) is copied to T0.

### Modify Auxiliary or Temporary Register Content

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	<b>mar(</b> TAx = D16 <b>)</b>	No	4	1	AD

0111 0111 DDDD DDDD DDDD DDDD FDDD xxxx Opcode

**Operands** TAx, D16

**Description** This instruction performs, in the A-unit address generation units, a load in the

> auxiliary or temporary registers TAx of the absolute data address signed constant D16. The operation is performed in the address phase of the pipeline;

however, data memory is not accessed.

**Status Bits** Affected by none

> Affects none

Repeat This instruction can be repeated.

Syntax	Description
mar(T1 = #FFFFh)	The address FFFFh is copied to T1.

### **AADD**

### Modify Auxiliary or Temporary Register Content by Addition

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	mar(TAy + TAx)	No	3	1	AD
[2]	mar(TAx + P8)	No	3	1	AD

Description	These instructions

These instructions perform, in the A-unit address generation units:

- an addition between two auxiliary or temporary registers, TAx and TAy, and stores the result in TAy
- an addition between the auxiliary or temporary registers TAx and a program address defined by a program address label assembled into unsigned P8, and stores the result in TAx

The operation is performed in the address phase of the pipeline, however data memory is not accessed.

If the destination register is an auxiliary register and the corresponding bit (ARnLC) in status register ST2\_55 is set to 1 or the circular addressing qualifier is in paralleled, the circular buffer management controls the result stored in the destination register.

### **Status Bits**

Affected by ST2 55

Affects none

### See Also

See the following other related instructions:

- Modify Auxiliary Register Content
- Modify Auxiliary or Temporary Register Content
- ☐ Modify Auxiliary or Temporary Register Content by Subtraction
- Modify Extended Auxiliary Register Content
- ☐ Modify Extended Auxiliary Register Content by Addition
- ☐ Modify Extended Auxiliary Register Content by Subtraction

### Modify Auxiliary or Temporary Register Content by Addition

### **Syntax Characteristics**

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline
[1]	mar(TAy + TAx)		No	3	1	AD
Opcod	e	0001	010E FSS	S xx	xx   FDD	D 0000
		0001	010E FSS	S xx	xx FDD	D 1000

The assembler selects the opcode depending on the instruction position in a paralleled pair.

### **Operands**

TAx, TAy

### **Description**

This instruction performs, in the A-unit address generation units, an addition between two auxiliary or temporary registers, TAy and TAx, and stores the result in TAy. The content of TAx is considered signed. The operation is performed in the address phase of the pipeline; however, data memory is not accessed.

If the destination register is an auxiliary register and the corresponding bit (ARnLC) in status register ST2\_55 is set to 1 or the circular addressing qualifier is in paralleled, the circular buffer management controls the result stored in the destination register.

### Compatibility with C54x devices (C54CM = 1)

In the translated code section, the mar() instruction must be executed with C54CM set to 1.

When circular modification is selected for the destination auxiliary register, this instruction modifies the selected destination auxiliary register by using BK03 as the circular buffer size register; BK47 is not used.

**Status Bits** 

Affected by ST2\_55

Affects none

Repeat

This instruction can be repeated.

# Example 1

Syntax	Description
mar(AR0 + T0)	The content of AR0 is added to the signed content of T0 and the result is stored in AR0.

Before			After			
XAR0	01	0000	XAR0	00	8000	
T0		8000	T0		8000	

Syntax	Description
mar(T0 + T1)	The content of T0 is added to the content of T1 and the result is stored in T0.

### Modify Auxiliary or Temporary Register Content by Addition

### **Syntax Characteristics**

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline
[2]	mar(TAx + P8)		No	3	1	AD
Opcod	е	0001	010E PPP	P PP	PP FDD	D 0100
		0001	010E PPP	P PP	PP FDD	D 1100

The assembler selects the opcode depending on the instruction position in a paralleled pair.

### **Operands**

TAx, P8

### Description

This instruction performs, in the A-unit address generation units, an addition between the auxiliary or temporary register TAx and a program address defined by a program address label assembled into unsigned P8, and stores the result in TAx. The operation is performed in the address phase of the pipeline; however, data memory is not accessed.

If the destination register is an auxiliary register and the corresponding bit (ARnLC) in status register ST2\_55 is set to 1 or the circular addressing qualifier is in paralleled, the circular buffer management controls the result stored in the destination register.

### Compatibility with C54x devices (C54CM = 1)

In the translated code section, the mar() instruction must be executed with C54CM set to 1.

When circular modification is selected for the destination auxiliary register, this instruction modifies the selected destination auxiliary register by using BK03 as the circular buffer size register; BK47 is not used.

### **Status Bits**

Affected by ST2\_55

Affects

none

### Repeat

This instruction can be repeated.

Syntax	Description
mar(T0 + #255)	The unsigned 8-bit value (255) is added to the content of T0 and the result is stored in T0.

### **ASUB**

### Modify Auxiliary or Temporary Register Content by Subtraction

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	mar(TAy – TAx)	No	3	1	AD
[2]	mar(TAx – P8)	No	3	1	AD

Desc	rij	oti	on

These instructions perform, in the A-unit address generation units:

- a subtraction between two auxiliary or temporary registers, TAy and TAx, and stores the result in TAy
- a subtraction between the auxiliary or temporary registers TAx and a program address defined by a program address label assembled into unsigned P8, and stores the result in TAx

The operation is performed in the address phase of the pipeline, however data memory is not accessed.

If the destination register is an auxiliary register and the corresponding bit (ARnLC) in status register ST2\_55 is set to 1 or the circular addressing qualifier is in paralleled, the circular buffer management controls the result stored in the destination register.

### **Status Bits**

Affected by ST2 55

Affects none

### See Also

See the following other related instructions:

- Modify Auxiliary Register Content
- Modify Auxiliary or Temporary Register Content
- ☐ Modify Auxiliary or Temporary Register Content by Addition
- Modify Extended Auxiliary Register Content
- ☐ Modify Extended Auxiliary Register Content by Addition
- ☐ Modify Extended Auxiliary Register Content by Subtraction

### Modify Auxiliary or Temporary Register Content by Subtraction

### **Syntax Characteristics**

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline
[1]	mar(TAy – TAx)		No	3	1	AD
Opcod	е	0001	010E FSS	S xx	xx FDD	D 0010
		0001	010E FSS	S xx	xx FDD	D 1010

The assembler selects the opcode depending on the instruction position in a paralleled pair.

Operands

TAx, TAy

**Description** 

This instruction performs, in the A-unit address generation units, a subtraction between two auxiliary or temporary registers, TAy and TAx, and stores the result in TAy. The content of TAx is considered signed. The operation is performed in the address phase of the pipeline; however, data memory is not accessed.

If the destination register is an auxiliary register and the corresponding bit (ARnLC) in status register ST2\_55 is set to 1 or the circular addressing qualifier is in paralleled, the circular buffer management controls the result stored in the destination register.

### Compatibility with C54x devices (C54CM = 1)

In the translated code section, the mar() instruction must be executed with C54CM set to 1.

When circular modification is selected for the destination auxiliary register, this instruction modifies the selected destination auxiliary register by using BK03 as the circular buffer size register; BK47 is not used.

Status Bits Affected by ST2\_55

Affects none

**Repeat** This instruction can be repeated.

# Example 1

Syntax	Description
mar(AR0 - T0)	The signed content of T0 is subtracted from the content of AR0 and the result is stored in AR0.

Before			After		
XAR0	01	8000	XAR0	01	0000
ΤO		8000	ΤO		8000

Syntax	Description
mar(T0 - T1)	The content of T1 is subtracted from the content of T0 and the result is stored in T0.

### Modify Auxiliary or Temporary Register Content by Subtraction

### **Syntax Characteristics**

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline
[2]	mar(TAx – P8)		No	3	1	AD
Opcod	е	0001	010E PPP	P PP	PP FDD	D 0110
		0001	010E PPP	P PP	PP FDD	D 1110

The assembler selects the opcode depending on the instruction position in a paralleled pair.

### **Operands**

TAx, P8

### Description

This instruction performs, in the A-unit address generation units, a subtraction between the auxiliary or temporary register TAx and a program address defined by a program address label assembled into unsigned P8, and stores the result in TAx. The operation is performed in the address phase of the pipeline; however, data memory is not accessed.

If the destination register is an auxiliary register and the corresponding bit (ARnLC) in status register ST2\_55 is set to 1 or the circular addressing qualifier is in paralleled, the circular buffer management controls the result stored in the destination register.

### Compatibility with C54x devices (C54CM = 1)

In the translated code section, the mar() instruction must be executed with C54CM set to 1.

When circular modification is selected for the destination auxiliary register, this instruction modifies the selected destination auxiliary register by using BK03 as the circular buffer size register; BK47 is not used.

### Status Bits

Affected by ST2\_55

Affects

none

### Repeat

This instruction can be repeated.

Syntax	Description
,	The unsigned 8-bit value (255) is subtracted from the signed content of AR0 and the result is stored in AR0.

# AADD

## Modify Data Stack Pointer

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	SP = SP + K8	Yes	2	1	AD

Opcode 0100 111E KKKK KKKK

Operands K8

**Description** This instruction performs an addition in the A-unit data-address generation

unit (DAGEN) in the address phase of the pipeline. The 8-bit signed constant, K8, is sign extended to 16 bits and added to the data stack pointer (SP). When in 32-bit stack configuration, the system stack pointer (SSP) is also modified. Updates of the SP and SSP (depending on the stack configuration) should not

be executed in parallel with this instruction.

Status Bits Affected by none

Affects none

**Repeat** This instruction can be repeated.

Syntax	Description
SP = SP + #127	The 8-bit value (127) is sign extended to 16 bits and added to the stack pointer (SP).

# **AMAR**

# Modify Extended Auxiliary Register Content

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	XAdst = mar(Smem)	No	3	1	AD
[2]	mar(XACdst = XACsrc)	Yes	3	1	AD

[2] Illar (70 (003) = 7	710310				103	0	'	/\D
Description	These instr	uctions	perform, in th	e A-unit a	ddress ge	eneratio	n units:	
	the 23-l	oit des	address specifi tination registe : accessed.	•	•			
	register operation	f, from on is pe	nove from one XACsrc to XA erformed in the accessed.	Cdst, and	stores th	e resul	t in XAC	dst. The
Status Bits	Affected by	S	T2_55					
	Affects	no	one					
See Also	See the foll	owing	other related ir	nstructions	S:			
	Load E	xtende	d Auxiliary Re	gister fron	n Memory			
	☐ Load E	xtende	d Auxiliary Re	gister with	Immedia	te Value	)	
	☐ Modify	Auxilia	ry Register Co	ontent				
	☐ Move E	xtende	ed Auxiliary Re	egister Cor	ntent			
	☐ Store E	xtende	ed Auxiliary Re	gister Cor	ntent to M	emory		
	☐ Modify	Extend	led Auxiliary R	egister Co	ontent by	Additior	1	

☐ Modify Extended Auxiliary Register Content by Subtraction

# Modify Extended Auxiliary Register Content

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	XAdst = mar(Smem)	No	3	1	AD

**Opcode** | 1110 1100 | AAAA AAAI | XDDD 1110

Operands Smem, XAdst

**Description** This instruction computes the effective address specified by the Smem

operand field and modifies the 23-bit destination register (XARx, XSP, XSSP, XDP, or XCDP). This operation is completed in the address phase of the pipeline by the A-unit address generator. Data memory is not accessed.

The premodification or postmodification of the auxiliary register (ARx), the use of \*port(#K), and the use of the readport() or writeport() qualifier is not supported for this instruction. The use of auxiliary register offset operations is supported. If the corresponding bit (ARnLC) in status register ST2\_55 is set to 1, the circular buffer management also controls the result stored in XAdst.

Status Bits Affected by ST2\_55

Affects none

**Repeat** This instruction can be repeated.

Syntax	Description
XAR0 = mar(*AR1)	The content of AR1 is loaded into XAR0.

### Modify Extended Auxiliary Register Content

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	mar(XACdst = XACsrc)	Yes	3	1	AD
Opcode	DAG_X:	010E XAC	S 00	01 XAC	D 0001
	DAG_Y: 0000	010E XAC	S 00	01 XAC	D 1001
Operan	ds XARx, XARy, XCDP				
Descrip	move from one addressing registe XACsrc to XACdst, and stores th performed in the address phase of th	r to another a e result in X <i>i</i>	addres ACdst.	sing reg	ister, from eration is

Compatibility with C54x devices (C54CM = 1)

None.

accessed.

**Status Bits** Affected by

Affects

Repeat This instruction can be repeated.

### Example 1

Syntax	Description
mar(XAR1 = XAR0)	The content of XAR0 is copied to XAR1.

Before		After	
XAR0	12 3456	XAR0	12 3456
XAR1	43 5634	XAR1	12 3456

### Example 2

Syntax	Description
mar(XCDP = XAR7)	The content of XAR7 is copied to XCDP.

Before		After	
XCDP	00 8000	XCDP	01 4000
XAR7	01 4000	XAR7	01 4000

Execution

(XACsrc) -> XACdst

# **AADD**

# Modify Extended Auxiliary Register Content by Addition

### **Syntax Characteristics**

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline	
[1]	mar(XACdst	+ XACsrc)		Yes	3	1	AD	
Opcod	e	DAG_X:	0001	010E XAC	CS 00	001 XAC	D 0000	
		DAG_Y:	0001	010E XAC	CS 00	01 XAC	D 1000	
Opera	nds	XARx, XARy, XCDP						
Description		This instruction performs, in the A-unit address generation units, a full 23-bit unsigned addition between two auxiliary or other addressing registers, XACdst and XACsrc, and stores the result in XACdst. The operation is performed in the address phase of the pipeline; however, data memory is not accessed.						
		Since the operation register is an auxiliary mode; that is, the res register ST2_55 to 1 operating in parallel is	register, it is not sult of setting the is undefined and	t allowed to u	ise the	circular a t (ARnLC	addressing 5) in status	
		Compatibility with C	54x devices (C	54CM = 1)				

Compatibility with C54x devices (C54CM = 1)

None.

**Status Bits** Affected by

Affects

Repeat This instruction can be repeated.

# See Also See the following other related instructions: Modify Auxiliary or Temporary Register Content Modify Auxiliary or Temporary Register Content by Addition Modify Auxiliary or Temporary Register Content by Subtraction Modify Auxiliary Register Content with Parallel Multiply Modify Auxiliary Register Content with Parallel Multiply and Accumulate Modify Auxiliary Register Content with Parallel Multiply and Subtract Modify Extended Auxiliary Register Content Modify Extended Auxiliary Register Content by Subtraction

### Example 1

Syntax	Description
mar(XAR1 + XAR0)	The content of XAR0 is added to XAR1 and stored in XAR1.

Parallel Modify Auxiliary Register Contents

Before		After	
XAR0	12 3456	XAR0	12 3456
XAR1	43 5634	XAR1	55 8A8A

### **Example 2**

Syntax	Description	
mar(XCDP + XAR7)	The content of XAR7 is added to XCDP and stored in XCDP.	

Before		After	
XCDP	00 8000	XCDP	01 0080
XAR7	00 8080	XAR7	00 8080

### Execution

(XACdst) + (XACsrc) -> XACdst

# **ASUB**

# Modify Extended Auxiliary Register Content by Subtraction

### **Syntax Characteristics**

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[1]	mar(XACdst -	XACsrc)		Yes	3	1	AD
Opcod	e	DAG_X:	0001	010E   XA	CS 00	01 XAC	D 0010
		DAG_Y:	0001	010E X	CS 00	01 XAC	D 1010
Operar	nds	XARx, XARy, XCDP					
Descri	ption	This instruction performs, in the A-unit address generation units, a full 23-bit subtraction between two auxiliary or other addressing registers, XACdst and XACsrc, and stores the result in XACdst. The operation is performed in the address phase of the pipeline; however, data memory is not accessed.  If the destination register is an auxiliary register and the corresponding bit			ACdst and med in the ssed.		
		(ARnLC) in status register significant control the result stored in the	allel, the	circular but	fer mar		_
		Compatibility with C54x de	vices (C	54CM = 1)			
		None.					
Status	Bits	Affected by					
		Affects					
Repeat	t	This instruction can be repea	ited.				

See Also	See the following other related instructions:		
		Modify Auxiliary or Temporary Register Content	
		Modify Auxiliary or Temporary Register Content by Addition	
		Modify Auxiliary or Temporary Register Content by Subtraction	
		Modify Auxiliary Register Content with Parallel Multiply	
		Modify Auxiliary Register Content with Parallel Multiply and Accumulate	
		Modify Auxiliary Register Content with Parallel Multiply and Subtract	
		Modify Extended Auxiliary Register Content	
		Modify Extended Auxiliary Register Content by Addition	
		Parallel Modify Auxiliary Register Contents	

# Example 1

Syntax	Description	
mar(XAR1 - XAR0)	The content of XAR0 is subtracted from XAR1 and stored in XAR1.	

Before		After	
XAR0	12 3456	XAR0	12 3456
XAR1	43 5634	XAR1	31 21DE

### Example 2

Syntax	Description	
mar(XCDP - XAR7)	The content of XAR7 is subtracted from XCDP and stored in XCDP.	

Before		After	
XCDP	00 8000	XCDP	00 7000
XAR7	00 1000	XAR7	00 1000

### Execution

(XACdst) - (XACsrc) -> XACdst

# MOV

## Move Accumulator Content to Auxiliary or Temporary Register

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	TAx = HI(ACx)	Yes	2	1	Х
Opcod	e	010	0 01	0E 00S	S FDDD

**Operands** ACx, TAx

**Description** This instruction moves the high part of the accumulator, ACx(31-16), to the

destination auxiliary or temporary register (TAx). The 16-bit move operation

is performed in the A-unit ALU.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

**Status Bits** Affected by M40

> Affects none

Repeat This instruction can be repeated.

See Also See the following other related instructions:

☐ Move Accumulator, Auxiliary, or Temporary Register Content

☐ Move Auxiliary or Temporary Register Content to Accumulator

### **Example**

Syntax	Description	
AR2 = HI(AC0)	The content of AC0(31–16) is copied to AR2.	

Before			After			
AC0	01 E500	0030	AC0	01	E500	0030
AR2		0200	AR2			E500

# MOV

# Move Accumulator, Auxiliary, or Temporary Register Content

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit Size Cycles Pipeline
[1]	dst = src	Yes 2 1 X
Opcod	le	0010 001E FSSS FDDD
Opera	nds	dst, src
Descri	ption	This instruction moves the content of the source (src) register to the destination (dst) register:
		When the destination (dst) register is an accumulator:
		■ The 40-bit move operation is performed in the D-unit ALU.
		During the 40-bit move operation, an overflow is detected according to M40:
		the destination accumulator overflow status bit (ACOVx) is set.
		the destination register (ACx) is saturated according to SATD.
		If the source (src) register is an auxiliary or temporary register, the 16 LSBs of the source register are sign extended to 40 bits according to SXMD.
		☐ When the destination (dst) register is an auxiliary or temporary register:
		■ The 16-bit move operation is performed in the A-unit ALU.
		If the source (src) register is an accumulator, the 16 LSBs of the accumulator are used to perform the operation.
		Compatibility with C54x devices (C54CM = 1)
		When this instruction is executed with M40 = 0, compatibility is ensured.
Status	Bits	Affected by M40, SATD, SXMD
		Affects ACOVx
Repea	t	This instruction can be repeated.
See Al	so	See the following other related instructions:
		☐ Move Accumulator Content to Auxiliary or Temporary Register
		☐ Move Auxiliary or Temporary Register Content to Accumulator
		☐ Move Auxiliary or Temporary Register Content to CPU Register

Move Extended Auxiliary Register Content

Syntax	Description
AC1 = AC0	The content of AC0 is copied to AC1. Because an overflow occurred, ACOV1 is set to 1.

Before		After	
AC0	01 E500 0030	AC0	01 E500 0030
AC1	00 2800 0200	AC1	01 E500 0030
M40	0	M40	0
SATD	0	SATD	0
ACOV1	0	ACOV1	1

# MOV

# Move Auxiliary or Temporary Register Content to Accumulator

### **Syntax Characteristics**

No.	Syntax							Parallel Enable Bit	Size	Cycles	Pipeline
[1]	HI(ACx) = TAx							Yes	2	1	Х
Opcod	le							01	01 00	)1E FSS	SS 00DD
Operands			x, TAx								
Descri	ption							ne auxiliary (x(31–16):	or temp	oorary req	gister (TAx)
			The 16-	bit r	nove op	peration is	perfo	ormed in the	D-uni	t ALU.	
			During t	the	16-bit n	nove oper	ration,	an overflo	w is de	etected a	ccording to
			■ the destination accumulator overflow status bit (ACOVx) is set.								
			■ the	des	tination	register (A	ACx)	is saturated	accor	ding to S	ATD.
		☐ If the source (src) register is an auxiliary or temporary register, the 16 LSBs of the source register are sign extended to 40 bits according to SXMD.									
		Compatibility with C54x devices (C54CM = 1)									
		Wh	nen this ir	nstru	uction is	executed	d with	M40 = 0, co	ompatil	oility is er	nsured.
Status	Bits	Aff	ected by		M40, S	SATD, SXN	MD				
		Aff	ects		ACOV	x					
Repeat	t	Thi	is instruc	tion	can be	repeated.					
See Al	so	Se	e the follo	owin	g other	related in	nstruc	tions:			
			Move A	ccui	mulator	Content to	to Aux	iliary or Ter	nporar	y Registe	r
			Move A	ccui	mulator	, Auxiliary,	, or Te	emporary Re	egister	Content	
			Move A	uxili	ary or T	emporary	/ Regi	ster Conten	t to CF	U Regist	er
			Move E	xten	nded Au	xiliary Re	gister	Content			
Examp	ole										

Syntax	Description
HI(AC0) = T0	The content of T0 is copied to AC0(31–16).

### MOV

### Move Auxiliary or Temporary Register Content to CPU Register

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	BRC0 = TAx	Yes	2	1	Х
[2]	BRC1 = TAx	Yes	2	1	X
[3]	<b>CDP</b> = TAx	Yes	2	1	X
[4]	CSR = TAx	Yes	2	1	X
[5]	SP = TAx	Yes	2	1	X
[6]	SSP = TAx	Yes	2	1	Χ

**Opcode** 

See Table 5-3 (page 5-258).

**Operands** 

TAx

Description

This instruction moves the content of the auxiliary or temporary register (TAx) to the selected CPU register. All the move operations are performed in the execute phase of the pipeline and the A-unit ALU is used to transfer the content of the registers.

There is a 3-cycle latency between SP, SSP, CDP, TAx, CSR, and BRCx update and their use in the address phase by the A-unit address generator units or by the P-unit loop control management.

For instruction [2] when BRC1 is loaded with the content of TAx, the block repeat save register (BRS1) is also loaded with the same value.

**Status Bits** 

Affected by none

Affects none

Repeat

This instruction can be repeated.

See Also

See the following other related instructions:

- ☐ Move Accumulator Content to Auxiliary or Temporary Register
- ☐ Move Accumulator, Auxiliary, or Temporary Register Content
- ☐ Move Auxiliary or Temporary Register Content to Accumulator
- ☐ Move CPU Register Content to Auxiliary or Temporary Register
- Move Extended Auxiliary Register Content

# Example

Syntax	Description
BRC1 = T1	The content of T1 is copied to the block repeat register (BRC1) and to the block
	repeat save register (BRS1).

Before		After	
T1	0034	T1	0034
BRC1	00EA	BRC1	0034
BRS1	00EA	BRS1	0034

Table 5–3. Opcodes for Move Auxiliary or Temporary Register Content to CPU Register Instruction

No.	Syntax	Opcode
[1]	<b>BRC0</b> = TAx	0101 001E FSSS 1110
[2]	BRC1 = TAx	0101 001E FSSS 1101
[3]	CDP = TAx	0101 001E FSSS 1010
[4]	CSR = TAx	0101 001E FSSS 1100
[5]	<b>SP</b> = TAx	0101 001E FSSS 1000
[6]	SSP = TAx	0101 001E FSSS 1001

5-258 Instruction Set Descriptions SWPU068E

### MOV

### Move CPU Register Content to Auxiliary or Temporary Register

### **Syntax Characteristics**

		Parallel			
No.	Syntax	Enable Bit	Size	Cycles	Pipeline
[1]	TAx = BRC0	Yes	2	1	Х
[2]	TAx = BRC1	Yes	2	1	X
[3]	TAx = CDP	Yes	2	1	X
[4]	TAx = SP	Yes	2	1	X
[5]	TAx = SSP	Yes	2	1	X
[6]	TAx = RPTC	Yes	2	1	Χ

**Opcode** See Table 5–4 (page 5-260).

**Operands** TAx

Description

This instruction moves the content of the selected CPU register to the auxiliary or temporary register (TAx). All the move operations are performed in the execute phase of the pipeline and the A-unit ALU is used to transfer the content of the registers.

For instructions [1] and [2], BRCx is decremented in the address phase of the last instruction of a loop. These instructions have a 3-cycle latency requirement versus the last instruction of a loop.

For instructions [3], [4], and [5], there is a 3-cycle latency between SP, SSP, CDP, and TAx update and their use in the address phase by the A-unit address generator units or by the P-unit loop control management.

Status Bits Affected by none

Affects none

**Repeat** Instruction [6] cannot be repeated; all other instructions can be repeated.

**See Also** See the following other related instructions:

- Move Accumulator Content to Auxiliary or Temporary Register
- ☐ Move Auxiliary or Temporary Register Content to CPU Register
- ☐ Store CPU Register Content to Memory

# Example

Syntax	Description	
T1 = BRC1	The content of block repeat register (BRC1) is copied to T1.	

Before		After		
T1	0034	T1	OOEA	
BRC1	00EA	BRC1	00EA	

Table 5–4. Opcodes for Move CPU Register Content to Auxiliary or Temporary Register Instruction

No.	Syntax	Opcode
[1]	TAx = <b>BRC0</b>	0100 010E 1100 FDDD
[2]	TAx = <b>BRC1</b>	0100 010E 1101 FDDD
[3]	TAx = CDP	0100 010E 1010 FDDD
[4]	TAx = SP	0100 010E 1000 FDDD
[5]	TAx = <b>SSP</b>	0100 010E 1001 FDDD
[6]	TAx = <b>RPTC</b>	0100 010E 1110 FDDD

5-260 Instruction Set Descriptions SWPU068E

# MOV

# Move Extended Auxiliary Register Content

# **Syntax Characteristics**

No. Synt	ax				Parallel Enable Bit	Size	Cycles	Pipeline
[1] xdst	= xsrc				No	2	1	Х
Opcode					100	01 00	00 XSS	S XDDD
Operands		st, xsrc			•		•	
Description		This instruction moves the content of the source register (xsrc) to the destination register (xdst):						
				register (xo is a 23-bit	•		•	•
		■ The 23	3-bit move o	peration is p	performed in	the D	unit ALU	
		■ The up	oper bits of A	ACx are fille	d with 0.			
				gister (xsrc) dst) is a 23-b			•	•
		■ The 23	3-bit move o	peration is p	performed in	the A	unit ALU	
		■ The lo	wer 23 bits	of ACx are l	oaded into x	dst.		
			ors, the Mo	register (xsrove Accumul			_	
Status Bits	Aff	ected by	none					
	Aff	ects	none					
Repeat	Th	This instruction can be repeated.		eated.				
See Also	Se	e the followi	ng other rel	ated instruc	tions:			
		Load Exter	nded Auxilia	ary Register	from Memo	ry		
		Load Exter	nded Auxilia	ary Register	with Immed	iate Va	lue	
		Modify Ext	ended Auxil	liary Registe	er Content			
		Store Exte	nded Auxilia	ary Register	Content to	Memor	у	
Example								

Syntax	Description			
XAR1 = AC0	The lower 23 bits of AC0 are loaded into XAR1.			

# MOV

# Move Memory to Memory

# **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	Smem = coef(Cmem)	No	3	1	Х
[2]	coef(Cmem) = Smem	No	3	1	X
[3]	Lmem = dbl(coef(Cmem))	No	3	1	X
[4]	dbl(coef(Cmem)) = Lmem	No	3	1	X
[5]	dbl(Ymem) = dbl(Xmem)	No	3	1	Х
[6]	Ymem = Xmem	No	3	1	Х

**Description**These instructions store the content of a memory location to a memory location. They use a dedicated datapath to perform the operation.

Status Bits Affected by none

Affects none

**See Also** See the following other related instructions:

- ☐ Store Accumulator Content to Memory
- ☐ Store Accumulator, Auxiliary, or Temporary Register Content to Memory
- ☐ Store Auxiliary or Temporary Register Pair Content to Memory
- ☐ Store CPU Register Content to Memory
- ☐ Store Extended Auxiliary Register Content to Memory

# **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	Smem = coef(Cmem)	No	3	1	Х

| 1110 | 1111 | AAAA AAAI | xxxx 00mm Opcode

Cmem, Smem **Operands** 

Description This instruction stores the content of a data memory operand Cmem,

addressed using the coefficient addressing mode, to a memory (Smem)

location.

For this instruction, the Cmem operand is not accessed through the BB bus. On all C55x-based devices, the Cmem operand may be mapped in external

or internal memory space.

**Status Bits** Affected by none

> Affects none

Repeat This instruction can be repeated.

Syntax	Description
*(#0500h) = coef(*CDP)	The content addressed by the coefficient data pointer register (CDP) is copied to address 0500h.

Before		After	
*CDP	3400	*CDP	3400
500	0000	500	3400

# **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	coef(Cmem) = Smem	No	3	1	Χ

Opcode 1110 1111 AAAA AAAI xxxx 01mm

Operands Cmem, Smem

**Description** This instruction stores the content of a memory (Smem) location to a data

memory (Cmem) location addressed using the coefficient addressing mode.

For this instruction, the Cmem operand is not accessed through the BB bus. On all C55x-based devices, the Cmem operand may be mapped in external

or internal memory space.

Status Bits Affected by none

Affects none

**Repeat** This instruction can be repeated.

Syntax	Description
coef(*CDP) = *AR3	The content addressed by AR3 is copied in the location addressed by the coefficient
	data pointer register (CDP).

# **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	Lmem = dbl(coef(Cmem))	No	3	1	Χ

Opcode | 1110 1111 | AAAA AAAI | xxxx 10mm

Operands Cmem, Lmem

**Description** This instruction stores the content of two consecutive data memory (Cmem)

locations, addressed using the coefficient addressing mode, to two

consecutive data memory (Lmem) locations.

For this instruction, the Cmem operand is not accessed through the BB bus. On all C55x-based devices, the Cmem operand may be mapped in external

or internal memory space.

Status Bits Affected by none

Affects none

**Repeat** This instruction can be repeated.

Syntax	Description
, , , , , , , , , , , , , , , , , , , ,	The content (long word) addressed by the coefficient data pointer register (CDP) and CDP + 1 is copied in the location addressed by AR1 and AR1 + 1, respectively. After the memory store, CDP is incremented by the content of T0 (5).

Before		After	
TO	0005	TO	0005
CDP	0200	CDP	0205
AR1	0300	AR1	0300
200	3400	200	3400
201	0FD3	201	0FD3
300	0000	300	3400
301	0000	301	0FD3

# **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	dbl(coef(Cmem)) = Lmem	No	3	1	Х

1110 1111 AAAA AAAI xxxx 11mm Opcode

**Operands** Cmem, Lmem

**Description** This instruction stores the content of two consecutive data memory (Lmem)

locations to two consecutive data memory (Cmem) locations addressed using

the coefficient addressing mode.

For this instruction, the Cmem operand is not accessed through the BB bus. On all C55x-based devices, the Cmem operand may be mapped in external

or internal memory space.

**Status Bits** Affected by none

> Affects none

Repeat This instruction can be repeated.

Syntax	Description
dbl(coef(*CDP)) = *AR3+	The content (long word) addressed by AR3 and AR3 + 1 is copied in the location addressed by the coefficient data pointer register (CDP) and CDP + 1, respectively. Because this instruction is a long-operand instruction, AR3 is incremented by 2 after the execution.

# **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	dbl(Ymem) = dbl(Xmem)	No	3	1	Х

**Opcode** | 1000 0000 | XXXM MMYY | YMMM 00xx

Operands Xmem, Ymem

**Description** This instruction stores the content of two consecutive data memory (Xmem)

locations, addressed using the dual addressing mode, to two consecutive data

memory (Ymem) locations.

Status Bits Affected by none

Affects none

**Repeat** This instruction can be repeated.

Syntax	Description
dbl(*AR1) = dbl(*AR0)	The content addressed by AR0 is copied in the location addressed by AR1 and the
	content addressed by AR0 + 1 is copied in the location addressed by AR1 + 1.

Before		After	
AR0	0300	AR0	0300
AR1	0400	AR1	0400
300	3400	300	3400
301	0FD3	301	0FD3
400	0000	400	3400
401	0000	401	0FD3

# **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	Ymem = Xmem	No	3	1	Χ

**Opcode** | 1000 0000 | XXXM MMYY | YMMM 01xx

Operands Xmem, Ymem

**Description** This instruction stores the content of data memory (Xmem) location,

addressed using the dual addressing mode, to data memory (Ymem) location.

Status Bits Affected by none

Affects none

**Repeat** This instruction can be repeated.

Syntax	Description
*AR3 = *AR5	The content addressed by AR5 is copied in the location addressed by AR3.

MPY

Multiply

# **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(ACy * ACx)	Yes	2	1	Х
[2]	ACy = rnd(ACx * Tx)	Yes	2	1	X
[3]	ACy = rnd(ACx * K8)	Yes	3	1	X
[4]	ACy = rnd(ACx * K16)	No	4	1	X
[5]	ACx = rnd(Smem * coef(Cmem))[, T3 = Smem]	No	3	1	X
[6]	ACy = rnd(Smem * ACx)[, T3 = Smem]	No	3	1	X
[7]	ACx = rnd(Smem * K8)[, T3 = Smem]	No	4	1	X
[8]	ACx = M40(rnd(uns(Xmem) * uns(Ymem)))[, T3 = Xmem]	No	4	1	X
[9]	ACx = rnd(uns(Tx * Smem))[, T3 = Smem]	No	3	1	Х
[10]	ACx = rnd(Smem * uns(coef(Cmem)))	No	3	1	Х

Description	These instruction	ons perform a multiplication in the D-unit MAC.
Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
	Affects	ACOVx, ACOVy
See Also	See the following	ng other related instructions:
	☐ Modify Auxi	liary Register Content with Parallel Multiply
	☐ Multiply and	d Accumulate
	☐ Multiply and	d Accumulate with Parallel Multiply
	☐ Multiply and	d Subtract
	☐ Multiply and	Subtract with Parallel Multiply
	☐ Multiply with	n Parallel Multiply and Accumulate
	☐ Multiply with	n Parallel Store Accumulator Content to Memory
	Parallel Mul	Itiplies
	☐ Square	

# **Syntax Characteristics**

No. Syntax					Parallel Enable Bit	Size	Cycles	Pipeline	
[1] $ACy = rnd(ACy)$	* AC	Ex)			Yes	2	1	Х	
Opcode					010	01 01	0E DDS	S 011%	
Operands	AC	x, ACy							
Description	This instruction performs a multiplication in the D-unit MAC. The input operands of the multiplier are ACx(32–16) and ACy(32–16).								
		If FRCT =	1, the output	of the mul	tiplier is shift	ed to t	he left by	1 bit.	
		Multiplication	on overflow o	detection d	epends on S	SMUL.			
	☐ The 32-bit result of the multiplication is sign extended to 40 bits.								
		☐ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.							
	<ul> <li>Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.</li> </ul>								
		When an o	verflow is de	tected, the	accumulato	r is sat	urated ad	ccording to	
	Co	mpatibility	with C54x o	levices (C	54CM = 1)				
	Wh	nen this instr	uction is exe	cuted with	M40 = 0, co	mpatib	ility is en	sured.	
Status Bits	Aff	ected by	FRCT, M40	, RDM, SA	ATD, SMUL				
	Aff	ects	ACOVy						
Repeat	Thi	is instruction	can be repe	eated.					

Syntax	Description
AC1 = AC1 * AC0	The product of the content of AC1 and the content of AC0 is stored in AC1.

Before				After			
AC0	02	6000	3400	AC0	02	6000	3400
AC1	00	C000	0000	AC1	00	4800	0000
M40			1	M40			1
FRCT			0	FRCT			0
ACOV1			0	ACOV1			0

Syntax	Characteristic	s							
No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline	
[2]	ACy = rnd(ACx)	* Tx)			Yes	2	1	X	
Opcode	9				010	01 10	0E DDS	SS ss0%	
Operan	ds	AC	Sx, ACy, Tx						
Descrip	otion	ope		n performs a multipli ne multiplier are ACx bits.				•	
			If FRCT =	1, the output of the mu	ıltiplier is shif	ted to t	he left by	1 bit.	
			Multiplication	on overflow detection	depends on S	SMUL.			
			☐ The 32-bit result of the multiplication is sign extended to 40 bits.						
		Rounding is performed according to RDM, if the applied to the instruction.					ne optional rnd keyword is		
				detection depends on accumulator overflow		n overflow is detected, the (ACOVy) is set.			
			When an o	verflow is detected, the	e accumulato	r is sat	urated a	ccording to	
		Со	mpatibility	with C54x devices (C	C54CM = 1)				
		Wh	nen this instr	ruction is executed with	n M40 = 0, co	mpatik	oility is er	nsured.	
Status	Bits	Aff	ected by	FRCT, M40, RDM, S	ATD, SMUL				
		Aff	ects	ACOVy					
Repeat		Thi	is instruction	can be repeated.					

Syntax	Description
AC0 = AC1 * T0	The product of the content of AC1 and the content of T0 is stored in AC0.

# **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ACy = rnd(ACx * K8)	Yes	3	1	Х

Opcode | 0001 111E | KKKK KKKK | SSDD xx0%

Operands ACx, ACy, K8

**Description**This instruction performs a multiplication in the D-unit MAC. The input operands of the multiplier are ACx(32–16) and the 8-bit signed constant, K8,

sign extended to 17 bits.

☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.

☐ The 32-bit result of the multiplication is sign extended to 40 bits.

Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by FRCT, M40, RDM

Affects none

**Repeat** This instruction can be repeated.

Syntax	Description
AC0 = AC1 * #-2	The product of the content of AC1 and a signed 8-bit value (-2) is stored in AC0.

Syntax Characteri	Parallel Parallel							
No. Syntax	Enable Bit Size Cycles Pipelin							
[4] $ACy = rnd(A$	ACx * K16) No 4 1 X							
Opcode	0111 1001 KKKK KKKK KKKK KKKK SSDD xx0							
Operands	ACx, ACy, K16							
Description	This instruction performs a multiplication in the D-unit MAC. The inproperands of the multiplier are ACx(32–16) and the 16-bit signed consta K16, sign extended to 17 bits.							
	☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.							
	☐ Multiplication overflow detection depends on SMUL.							
	☐ The 32-bit result of the multiplication is sign extended to 40 bits.							
	☐ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.							
	☐ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.							
	☐ When an overflow is detected, the accumulator is saturated according to SATD.							
	Compatibility with C54x devices (C54CM = 1)							
	When this instruction is executed with M40 = 0, compatibility is ensured.							
Status Bits	Affected by FRCT, M40, RDM, SATD, SMUL							
	Affects ACOVy							
Repeat This instruction can be repeated.								
Example								
Syntax	Description							
AC0 = AC1 * #-64	The product of the content of AC1 and a signed 16-bit value (-64) is stored in AC							

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	ACx = rnd(Smem * coef(Cmem))[, T3 = Smem]	No	3	1	Х

### Opcode

1101 0001 AAAA AAAI U%DD 00mm

#### **Operands**

ACx, Cmem, Smem

### Description

This instruction performs a multiplication in the D-unit MAC. The input operands of the multiplier are the content of a memory location (Smem), sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and sign extended to 17 bits.

- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- ☐ The 32-bit result of the multiplication is sign extended to 40 bits.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVx) is set.
- ☐ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to store the 16-bit data memory operand Smem in temporary register T3.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Affected by FRCT, M40, RDM, SATD, SMUL **Status Bits** 

> Affects ACOVx

Repeat This instruction can be repeated.

Syntax	Description
AC0 = *AR3 * coef(*CDP)	The product of the content addressed by AR3 and the content addressed by the coefficient data pointer register (CDP) is stored in AC0.

### **Syntax Characteristics**

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline
[6]	ACy = rnd(Smem * ACx) [,T3 = Smem]		No	3	1	Х
Opcod	e	1101	0011 AAA	A AA	AI U%D	D 00SS
Opera	ACx, ACy, Smem					

# **Description**

This instruction performs a multiplication in the D-unit MAC. The input operands of the multiplier are ACx(32-16) and the content of a memory location (Smem), sign extended to 17 bits.

- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- ☐ Multiplication overflow detection depends on SMUL.
- ☐ The 32-bit result of the multiplication is sign extended to 40 bits.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- ☐ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to store the 16-bit data memory operand Smem in temporary register T3.

### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

### Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVy

#### **Repeat** This instruction can be repeated.

Syntax	Description
AC0 = *AR3 * AC1	The product of the content addressed by AR3 and the content of AC1 is stored in AC0.

### **Syntax Characteristics**

No.	Syntax			Paral Enable		Size	Cycles	Pipeline
[7]	ACx = rnd(Smem * K8) [,T3 = Smem]	No		4	1	Х		
Opcod	<b>e</b> 11:	L1 1000	AAAA	AAAI	KKKK	KK	KK xxD	D x0U%

# **Operands**

ACx, K8, Smem

### **Description**

This instruction performs a multiplication in the D-unit MAC. The input operands of the multiplier are the content of a memory location (Smem), sign extended to 17 bits, and the 8-bit signed constant, K8, sign extended to 17 bits.

- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- ☐ The 32-bit result of the multiplication is sign extended to 40 bits.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.

This instruction provides the option to store the 16-bit data memory operand Smem in temporary register T3.

# Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

**Status Bits** FRCT, M40, RDM Affected by

> Affects none

Repeat This instruction can be repeated.

Syntax	Description
AC0 = *AR3 * #-2	The product of the content addressed by AR3 and a signed 8-bit value (–2) is stored in AC0.

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline		
[8]	ACx = M40(rr	nd(uns(	Xmem) * uns(Ymem	)))[, T3 = Xmem]	No	4	1	Х		
Opcod	e		1000	0110 XXXM	MMYY YMM	MM xx	DD 000	g uuU%		
Operar	nds	AC	x, Xmem, Ymem							
Descri	ption	ope ext	This instruction performs a multiplication in the D-unit MAC. The inpoperands of the multiplier are the content of data memory operand Xme extended to 17 bits, and the content of data memory operand Yme extended to 17 bits.							
			Input operands ar	e extended to 1	7 bits accord	ling to	uns.			
			•	uns keyword is a y location is zero			•	the content		
			•	I uns keyword is memory location						
			If FRCT = 1, the c	output of the mul	tiplier is shift	ted to t	he left by	1 bit.		
			Multiplication ove	rflow detection d	depends on SMUL.					
			The 32-bit result of	of the multiplicat	ation is sign extended to 40 bits.					
			Rounding is perfo	_	to RDM, if the	ne opti	onal rnd	keyword is		
			Overflow detection destination accum	•				ected, the		
			When an overflow SATD.	v is detected, the	accumulato	r is sat	turated ad	ccording to		
			This instruction provides the option to locally set M40 to 1 for the executi the instruction, if the optional M40 keyword is applied to the instruction.							

This instruction provides the option to store the 16-bit data memory operand Xmem in temporary register T3.

# Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Affected by FRCT, M40, RDM, SATD, SMUL, SXMD **Status Bits** 

> Affects ACOVx

Repeat This instruction can be repeated.

Syntax	Description
AC0 = uns(*AR3) * uns(*AR4)	The product of the unsigned content addressed by AR3 and the unsigned content addressed by AR4 is stored in AC0.

# **Syntax Characteristics**

Syntax	Characteristi	CS								
No.	Syntax						Parallel Enable Bit	Size	Cycles	Pipeline
[9]	ACx = rnd(uns	(Tx *	Smem)	) [,T3 =	: Smem]		No	3	1	Х
Opcod	e					1101	0011 AA	AA AA	AI U%E	DD ulss
Operar	nds	AC	x, Sm	em, Tx	(	·			•	
Descri	ption	op	This instruction performs a multiplication in the D-unit MAC. The input operands of the multiplier are the content of Tx, sign extended to 17 bits, and the content of a memory location (Smem), sign extended to 17 bits.							
			☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.							
			Multi	plicatio	n overflow	detection of	depends on S	SMUL.		
			☐ The 32-bit result of the multiplication is extended to 40 bits according to uns							
					•	s keyword i nded to 40	s applied to bits.	the ins	struction,	the 32-bit
				-		keyword is nded to 40 l	not applied to	o the ir	struction	, the 32-bit
			☐ Rounding is performed according to RDM, if the optional rnd keyword applied to the instruction.						keyword is	
			Overflow detection depends on M40. If an overflow is detected, destination accumulator overflow status bit (ACOVx) is set.						ected, the	
			Whei		verflow is d	letected, the	e accumulato	r is sat	urated a	cording to
			This instruction provides the option to store the 16-bit data mer Smem in temporary register T3.						ta memo	ry operand
		Co	mpati	ibility	with C54x	devices (C	54CM = 1)			
		Wł	nen thi	s instr	uction is ex	ecuted with	M40 = 0, cc	mpatik	oility is er	sured.
Status	Bits	Aff	ected	by	FRCT, M <sup>2</sup>	10, RDM, S	ATD, SMUL			
		Aff	ects		ACOVx					
Repeat	t	Th	is instr	uction	can be rep	peated.				

Syntax	Description
,	The unsigned product of the content addressed by AR3 and the content of T0 is stored in AC0.

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles Pipeline		
[10]	ACx = rnd(Smem * uns(coef(Cmem)))	No	3	1	X	

#### **Opcode**

1101 0000 AAAA AAAI 0%DD 01mm

### **Operands**

ACx, Cmem, Smem

### Description

This instruction performs a multiplication in the D-unit MAC1. The input operands of the multiplier are the content of a data memory location (Smem) and the content of a data memory operand (Cmem).

#### Note:

The uns keyword is mandatory for this instruction.

The data memory operand Smem is addressed by DAGEN path X by using the Smem addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC1. The other data memory operand Cmem is addressed by DAGEN path C by using the coefficient addressing mode, driven on data bus BDB, and sign extended to 17 bits with filling zeros in the MAC1.

- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- ☐ Multiplication overflow detection depends on SMUL.
- ☐ The 32-bit result of the multiplication is sign extended to 40 bits.
- ☐ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ☐ When an overflow is detected, the accumulator is saturated according to SATD.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

This instruction can be applied to compute the intermediate multiplication result of a double precision multiplication and to free up one DAGEN operator (DAGEN path Y) for storing an instruction with enabling parallelism.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVx

**Repeat** This instruction can be repeated.

# **Example**

Syntax	Description
AC0 = *AR3 - * uns(coef(*CDP+))	The product of the content addressed by AR3 and the unsigned content addressed by the coefficient data pointer register (CDP) is stored in AC0.
	AR3 is decremented by 1 and CDP is incremented by 1.

#### Execution

rnd((Smem) [16:0] \*uns(Cmem) [16:0]) -> ACx

Before				After			
AC0	FF	8000	0000	AC0	FF	FF00	0000
XAR3		00	1001	XAR3		00	1000
Data memory							
1001h			FE00	1001h			FE00
XCDP		00	2000	XCDP		00	2000
Coeff memor	У						
2000h			8000	2000h			8000

# MPY::MAC

# Multiply with Parallel Multiply and Accumulate

# **Syntax Characteristics**

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipe line
[1]	*	rnd(uns(Xmem) * uns(coef(Cmem)))), rnd((ACy >> #16) + (uns(Ymem) * nem)))))	No	4	1	Х
[2]	ACy = M40(1) $ACx = M40(1)$	No	4	1	X	
[3]	ACy = M40(I ACx = M40(I uns(LO(coe	No	4	1	Х	
[4]	$ACy = \frac{M40(1)}{ACx}$ $ACx = \frac{M40(1)}{ACx}$	No	5	1	Χ	
Description		These instructions perform two parallel of multiply and accumulate (MAC). The open MACs.	-	•		

multiply and accumulate (MAC). The operations are execut MACs.

Status Bits

Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVx, ACOVy

See Also

See the following other related instructions:

Multiply

Multiply and Accumulate

Parallel Multiply and Accumulates

### Multiply With Parallel Multiply and Accumulate

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = M40(rnd(uns(Xmem) * uns(coef(Cmem)))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(coef(Cmem)))))	No	4	1	Х

### **Opcode**

1000 0100 XXXM MMYY YMMM 10mm uuDD DDg%

### **Operands**

ACx, ACy, Cmem, Xmem, Ymem

### Description

This instruction performs two parallel operations in one cycle: multiply, and multiply and accumulate (MAC). The operations are executed in the two D-unit MACs.

The first operation performs a multiplication in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Xmem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

The second operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Ymem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

- ☐ Input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- ☐ Multiplication overflow detection depends on SMUL.
- For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits.
- ☐ For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy, which has been shifted to the right by 16 bits. The shifting operation is performed with a sign extension of source accumulator ACy(39).

applied to	the instru	ction.					
Overflow destination	detection n accumul	•			is	detected,	the

Rounding is performed according to RDM, if the optional rnd keyword is

☐ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Each data flow can also disable the usage of the corresponding MAC unit, while allowing the modification of auxiliary registers in the three address generation units through the following instructions:

- mar(Xmem)
- mar(Ymem)
- mar(Cmem)

**Status Bits** 

Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVx, ACOVy

Repeat

This instruction can be repeated.

Syntax	Description
AC0 = uns(*AR3) * uns(coef(*CDP)), AC1 = (AC1 >> #16) + (uns(*AR4) * uns(coef(*CDP)))	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the coefficient data pointer register (CDP) is stored in AC0. The product of the unsigned content addressed by AR4 and the unsigned content addressed by CDP is added to the content of AC1, which has been shifted to the right by 16 bits. The result is stored in AC1.

# Multiply with Parallel Multiply and Accumulate

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACy = M40(rnd(uns(Smem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx + (uns(Smem) * uns(LO(coef(Cmem)))))))	No	4	1	Х

#### **Opcode**

| 1111 | 1101 | AAAA | AAAI | 0000 | 01mm | DDDD | uug%

#### **Operands**

ACx, ACy, Cmem, Smem

### Description

This instruction performs two parallel operations in one cycle: multiply, and multiply and accumulate (MAC). The operations are executed in the two D-unit MACs.

The first operation performs a multiplication in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand HI(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC2 (this data is shared to MAC1 and MAC2). The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA (effective address); the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and an accumulation in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand LO(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

The content of the memory location is zero extended to 17 bits, if the
optional uns keyword is applied to the input operand.

- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.

L	extended to 40 bits.
	For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.
	Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
	Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
	When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

# Compatibility with C54x devices (C54CM = 1)

None.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVx, ACOVy

**Repeat** This instruction can be repeated.

### **Example**

Syntax	Description
AC1 = uns(*AR3-) * uns(HI(coef(*CDP+))), AC0 = AC0 + (uns(*AR3-) * uns(LO(coef(*CDP+))))	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is stored in AC1. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the lower part of the CDP is added to the content of AC0. The result is stored in AC0. AR3 is decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

#### Execution

M40(rnd(uns(Smem)[16:0]\*uns(HI(coef(Cmem)))[16:0])) -> ACy
ACx+M40(rnd(uns(Smem)[16:0]\*uns(LO(coef(Cmem)))[16:0])) -> ACx

# MPY::MAC Multiply with Parallel Multiply and Accumulate

Before				After			
AC0	00	0000	8000	AC0	00	3F80	8000
XAR3		00	10FF	XAR3		00	10FE
Data memory							
10FFh			FE00	10FFh			FE00
XCDP		00	2000	XCDP		00	2002
Coeff memory	У						
2001h			4000	2001h			4000
AC1		8000	0000	AC1	00	7F00	0000
Coeff memory	У						
2000h			8000	2000h			8000

5-288 Instruction Set Descriptions SWPU068E

### Multiply with Parallel Multiply and Accumulate

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipe- line
[3]	ACy = M40(rnd(uns(HI(Lmem)) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx + (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))	No	4	1	Х

**Opcode** 

| 1111 | 1101 | AAAA | AAAI | 0100 | 01mm | DDDD | uug%

**Operands** 

ACx, ACy, Cmem, Lmem

Description

This instruction performs two parallel operations in one cycle: multiply, and multiply and accumulate (MAC). The operations are executed in the two D-unit MACs.

The first operation performs a multiplication in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand HI(Lmem) and the content of data memory operand HI(coef(Cmem)). The data memory operand HI(Lmem) is addressed by DAGEN path X with the EA (effective address); the data, which can be assumed to be the higher part of long word memory data, is driven on data bus CDB and sign extended to 17 bits in the MAC2. The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and an accumulation in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand LO(Lmem) and the content of data memory operand LO(coef(Cmem)). The data memory operand LO(Lmem) is addressed by DAGEN path X with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word memory data, is driven on data bus DDB and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

☐ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.

	If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
	Multiplication overflow detection depends on SMUL.
	For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits.
	For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.
	Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
	Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
	When an overflow is detected, the accumulator is saturated according to SATD.
	s instruction provides the option to locally set M40 to 1 for the execution o instruction, if the optional M40 keyword is applied to the instruction.
and bus pre	this instruction, the Cmem operand is accessed through the BAB, BDB B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DE ses are only connected to internal memory and not to external memory. To vent the generation of a bus error, the Cmem operand must not be mapped external memory.
Со	mpatibility with C54x devices (C54CM = 1)

None.

**Status Bits** Affected by FRCT, M40, RDM, SATD, SMUL

> Affects ACOVx, ACOVy

Repeat This instruction can be repeated.

Syntax	Description
AC1 = uns(HI(*AR3-)) * uns(HI(coef(*CDP+))), AC0 = AC0 + (uns(LO(*AR3-)) * uns(LO(coef(*CDP+))))	Both instructions are performed in parallel. The product of the unsigned content addressed by the higher part of AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is stored in AC1. The product of the unsigned content addressed by the lower part of AR3 and the unsigned content addressed by the lower part of the CDP is added to the content of AC0. The result is stored in AC0. When AR3– is used with HI/LO, AR3 is decremented by 2. When CDP+ is used with HI/LO, CDP is incremented by 2.

#### Execution

M40(rnd(uns(HI(Lmem))[16:0]\*uns(HI(coef(Cmem)))[16:0])) -> ACy ACx+M40(rnd(uns(LO(Lmem))[16:0]\*uns(LO(coef(Cmem)))[16:0])) -> ACx

Before				After			
AC0	00	0000	8000	AC0	00	3F80	8000
XAR3		00	10FE	XAR3		00	10FC
Data memory							
10FFh			FE00	10FFh			FE00
XCDP		00	2000	XCDP		00	2002
Coeff memory	Y						
2001h			4000	2001h			4000
AC1	FF	8000	0000	AC1	00	7F80	0000
Data memory							
10FEh			FF00	10FEh			FF00
Coeff memory	Y						
2000h			8000	2000h			8000

# Multiply With Parallel Multiply and Accumulate

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	ACy = M40(rnd(uns(Ymem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx + uns(Xmem) * uns(LO(coef(Cmem)))))	No	5 (*)	1	Х

(\*) 1 LSB is allocated to instruction slot #2.

Opcode

1001 0010 XXXM MMYY YMMM 01mm uuDD DDg%

**Operands** 

ACx, ACy, Cmem, Xmem, Ymem

Description

This instruction performs two parallel operations in one cycle: multiply, and multiply and accumulate (MAC). The operations are executed in the two D-unit MACs.

The first operation performs a multiplication in the D-unit MAC2. The input operands of the multiplier are the contents of data memory operand Ymem, extended to 17 bits, and the content of data memory operand HI(coef(Cmem)) which is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and an accumulation in the D-unit MAC1. The input operands of the multiplier are the contents of data memory operand Xmem, extended to 17 bits, and the content of data memory operand LO(coef(Cmem)) which is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- ☐ The input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.

	The 32-bit result of the multiplication is sign extended to 40 bits.
_	Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
	Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
	When an overflow is detected, the accumulator is saturated according to SATD. $ \label{eq:saturated} % \begin{center} \end{center} % \begin{center} ce$
	Because this instruction occupies both instruction slots $\#1$ and $\#2$ , this can not be executed in parallel with other instructions.
	The Xmem operand can access the MMRs but the Ymem operand can not. $ \\$
	s instruction provides the option to locally set M40 to 1 for the execution of instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

# Compatibility with C54x devices (C54CM = 1)

None.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVx, ACOVy

**Repeat** This instruction can be repeated.

# **Example**

Syntax	Description
AC1 = uns(*AR3-) * uns(HI(coef(*CDP+))), AC0 = AC0 + (uns(*AR2-) * uns(LO(coef(*CDP+))))	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is stored in AC1. The product of the unsigned content addressed by AR2 and the unsigned content addressed by the lower part of the CDP is added to the content of AC0. The result is stored in AC0. AR3 and AR2 are decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

#### Execution

# MPY::MAC Multiply with Parallel Multiply and Accumulate

Before				After			
AC0	00	0000	8000	AC0	00	3F80	8000
XAR2		00	10FE	XAR2		00	10FD
XAR3		00	20FE	XAR3		00	20FD
Data memory							
10FEh			FE00	10FEh			FE00
XCDP		00	2000	XCDP		00	2002
Coeff memory	У						
2001h			4000	2001h			4000
AC1	FF	8000	0000	AC1	00	7F80	0000
Data memory							
20FEh			FF00	20FFh			FF00
Coeff memory	У						
2000h			8000	2000h			8000

5-294 Instruction Set Descriptions SWPU068E

# MPY::MAS

# Multiply With Parallel Multiply and Subtract

# **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipe- line
[1]	ACy = M40(rnd(uns(Smem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx - (uns(Smem) * uns(LO(coef(Cmem))))))	No	4	1	Х
[2]	ACy = M40(rnd(uns(HI(Lmem)) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx - (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))	No	4	1	X
[3]	ACy = M40(rnd(uns(Ymem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx - uns(Xmem) * uns(LO(coef(Cmem)))))	No	5	1	X

Description		ions perform two parallel operations in one cycle: multiply, and ubtract (MAS). The operations are executed in the two D-unit
Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL
	Affects	ACOVx, ACOVy
See Also	See the follow	ring other related instructions:
	Multiply	
	Multiply ar	nd Subtract
	□ Parallel M	ultiply and Subtract

# Multiply with Parallel Multiply and Subtract

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = M40(rnd(uns(Smem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx - (uns(Smem) * uns(LO(coef(Cmem))))))	No	4	1	Х

#### **Opcode**

| 1111 | 1101 | AAAA | AAAI | 0000 | 11mm | DDDD | uug%

#### **Operands**

ACx, ACy, Cmem, Smem

#### **Description**

This instruction performs two parallel operations in one cycle: multiply, and multiply and subtract (MAS). The operations are executed in the two D-unit MACs.

The first operation performs a multiplication in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand HI(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC2 (this data is shared to MAC1 and MAC2). The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA (effective address); the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and a subtraction in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand LO(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

The content of the memory location is zero extended to 17 bits, if the
optional uns keyword is applied to the input operand.

- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.

For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits.
For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACx.
Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

# Compatibility with C54x devices (C54CM = 1)

None.

**Status Bits** Affected by FRCT, M40, RDM, SATD, SMUL

> Affects ACOVx, ACOVy

Repeat This instruction can be repeated.

### **Example**

Syntax	Description
AC1 = uns(*AR3-) * uns(HI(coef(*CDP+))), AC0 = AC0 - (uns(*AR3-) * uns(LO(coef(*CDP+))))	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is stored in AC1. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the lower part of the CDP is subtracted from the content of AC0. The result is stored in AC0. AR3 is decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

M40(rnd(uns(Smem) [16:0] \*uns(HI(coef(Cmem))) [16:0])) -> ACy ACx-M40(rnd(uns(Smem)[16:0]\*uns(LO(coef(Cmem)))[16:0])) -> ACx

# MPY::MAS Multiply With Parallel Multiply and Subtract

Before				After			
AC0	00	0000	8000	AC0	FF	C080	8000
XAR3		00	10FF	XAR3		00	10FE
Data memory							
10FFh			FE00	10FFh			FE00
XCDP		00	2000	XCDP		00	2002
Coeff memor	У						
2001h			4000	2001h			4000
AC1	FF	8000	0000	AC1	00	7F00	0000
Coeff memor	У						
2000h			8000	2000h			8000

5-298 Instruction Set Descriptions SWPU068E

# Multiply with Parallel Multiply and Subtract

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipe- line
[2]	ACy = M40(rnd(uns(HI(Lmem)) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx - (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))	No	4	1	Х

**Opcode** 

| 1111 | 1101 | AAAA | AAAI | 0100 | 11mm | DDDD | uug%

**Operands** 

ACx, ACy, Cmem, Lmem

Description

This instruction performs two parallel operations in one cycle: multiply, and multiply and subtract (MAS). The operations are executed in the two D-unit MACs.

The first operation performs a multiplication in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand HI(Lmem) and the content of data memory operand HI(coef(Cmem)). The data memory operand HI(Lmem) is addressed by DAGEN path X with the EA (effective address); the data, which can be assumed to be the higher part of long word memory data, is driven on data bus CDB and sign extended to 17 bits in the MAC2. The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and a subtraction in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand LO(Lmem) and the content of data memory operand LO(coef(Cmem)). The data memory operand LO(Lmem) is addressed by DAGEN path X with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word memory data, is driven on data bus DDB and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

☐ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.

	If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.						
	Multiplication overflow detection depends on SMUL.						
	For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits.						
	For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACx.						
	Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.						
	Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.						
	When an overflow is detected, the accumulator is saturated according to SATD.						
	s instruction provides the option to locally set M40 to 1 for the execution o instruction, if the optional M40 keyword is applied to the instruction.						
and	For this instruction, the Cmem operand is accessed through the BAB, BDB and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DE buses are only connected to internal memory and not to external memory. To						

prevent the generation of a bus error, the Cmem operand must not be mapped

Compatibility with C54x devices (C54CM = 1)

None.

**Status Bits** Affected by FRCT, M40, RDM, SATD, SMUL

on external memory.

ACOVx, ACOVy Affects

Repeat This instruction can be repeated.

Syntax	Description
AC1 = uns(HI(*AR3-)) * uns(HI(coef(*CDP+))), AC0 = AC0 - (uns(LO(*AR3-)) * uns(LO(coef(*CDP+))))	Both instructions are performed in parallel. The product of the unsigned content addressed by the higher part of AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is stored in AC1. The product of the unsigned content addressed by the lower part of AR3 and the unsigned content addressed by the lower part of the CDP is subtracted from the content of AC0. The result is stored in AC0. When AR3– is used with HI/LO, AR3 is decremented by 2. When CDP+ is used with HI/LO, CDP is incremented by 2.

#### Execution

M40(rnd(uns(HI(Lmem))[16:0]\*uns(HI(coef(Cmem)))[16:0])) -> ACy ACx-M40(rnd(uns(LO(Lmem))[16:0]\*uns(LO(coef(Cmem)))[16:0])) -> ACx

Before				After			
AC0	00	0000	8000	AC0	FF	C080	8000
XAR3		00	10FE	XAR3		00	10FC
Data memory							
10FFh			FE00	10FFh			FE00
XCDP		00	2000	XCDP		00	2002
Coeff memory	У						
2001h			4000	2001h			4000
AC1	FF	8000	0000	AC1	00	7F80	0000
Data memory							
10FEh			FF00	10FEh			FF00
Coeff memory	У						
2000h			8000	2000h			8000

# Multiply with Parallel Multiply and Subtract

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ACy = M40(rnd(uns(Ymem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx - uns(Xmem) * uns(LO(coef(Cmem)))))	No	5 (*)	1	Х

(\*) 1 LSB is allocated to instruction slot #2.

Opcode

1001 0010 XXXM MMYY YMMM 10mm uuDD DDg%

**Operands** 

ACx, ACy, Cmem, Xmem, Ymem

Description

This instruction performs two parallel operations in one cycle: multiply, and multiply and subtract (MAS). The operations are executed in the two D-unit MACs.

The first operation performs a multiplication in the D-unit MAC2. The input operands of the multiplier are the contents of data memory operand Ymem, extended to 17 bits, and the contents of data memory operand HI(coef(Cmem)) which is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and a subtraction in the D-unit MAC1. The input operands of the multiplier are the contents of data memory operand Xmem, extended to 17 bits, and the content of data memory operand LO(coef(Cmem)) which is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC 1.

- ☐ The input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.

The 32-bit result of the multiplication is sign extended to 40 bits.
Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
When an overflow is detected, the accumulator is saturated according to SATD.
Because this instruction occupies both instruction slots #1 and #2, this can not be executed in parallel with other instructions.
The Xmem operand can access the MMRs but the Ymem operand can not.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 key word is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

# Compatibility with C54x devices (C54CM = 1)

None.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVx, ACOVy

**Repeat** This instruction can be repeated.

# **Example**

Syntax	Description
AC1 = uns(*AR3-) * uns(HI(coef(*CDP+))), AC0 = AC0 - (uns(*AR2-) * uns(LO(coef(*CDP+))))	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is stored in AC1. The product of the unsigned content addressed by AR2 and the unsigned content addressed by the lower part of the CDP is subtracted from the content of AC0. The result is stored in AC0. AR3 and AR2 are decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

#### Execution

# MPY::MAS Multiply with Parallel Multiply and Subtract

Before				After			
AC0	00	0000	8000	AC0	FF	C080	8000
XAR2		00	10FE	XAR2		00	10FD
XAR3		00	20FE	XAR3		00	20FD
Data memory							
10FEh			FE00	10FEh			FE00
XCDP		00	2000	XCDP		00	2002
Coeff memory	Y						
2001h			4000	2001h			4000
AC1	FF	8000	0000	AC1	00	7F80	0000
Data memory							
20FEh			FF00	20FFh			FF00
Coeff memory	Y						
2000h			8000	2000h			8000

5-304 Instruction Set Descriptions SWPU068E

# MPYM::MOV

#### Multiply with Parallel Store Accumulator Content to Memory

## Syntax Characteristics

No.	Syntax				Paral Enable		Size	Cycles	Pipeline
[1]	ACy = rnd(Tx * Xmem), Ymem = HI(ACx << T2) [,T3 = Xm	nem]			No		4	1	Х
Opcod	e	1000	0111	MXXX	MMYY	YMMN	n ssi	DD 00	0x ssU%

# Operands

ACx, ACy, Tx, Xmem, Ymem

#### Description

This instruction performs two operations in parallel: multiply and store.

The first operation performs a multiplication in the D-unit MAC. The input operands of the multiplier are the content of Tx, sign extended to 17 bits, and the content of data memory operand Xmem, sign extended to 17 bits.

- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- ☐ The 32-bit result of the multiplication is sign extended to 40 bits.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- When an overflow is detected, the accumulator is saturated according to SATD.
- ☐ This instruction provides the option to store the 16-bit data memory operand Xmem in temporary register T3.

The second operation shifts the accumulator ACx by the content of T2 and stores ACx(31-16) to data memory operand Ymem. If the 16-bit value in T2 is not within -32 to +31, the shift is saturated to -32 or +31 and the shift is performed with this value.

- ☐ The input operand is shifted in the D-unit shifter according to SXMD.
- ☐ After the shift, the high part of the accumulator, ACx(31–16), is stored to the memory location.

# Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When this instruction is executed with C54CM = 1, the 6 LSBs of T2 determine the

**Status Bits** 

Repeat

See Also

the 16-bit value in T2 is between -32 to -17, a modulo 16 operation transforms the shift quantity to within -16 to -1. ☐ If the SST bit = 1 and the SXMD bit = 0, then the saturate and uns keywords are applied to the instruction regardless of the optional keywords selected by the user, with the following syntax: ACy = rnd(Tx \* Xmem),Ymem = HI(saturate(uns(ACx << T2))) [,T3 = Xmem] ☐ If the SST bit = 1 and the SXMD bit = 1, then only the saturate keyword is applied to the instruction regardless of the optional keywords selected by the user, with the following syntax: ACy = rnd(Tx \* Xmem),Ymem = HI(saturate(ACx << T2)) [,T3 = Xmem] Affected by C54CM, FRCT, M40, RDM, SATD, SMUL, SST, SXMD Affects **ACOVy** This instruction can be repeated. See the following other related instructions: ☐ Addition with Parallel Store Accumulator Content to Memory ☐ Multiply ☐ Multiply and Accumulate with Parallel Store Accumulator Content to Memory Multiply and Subtract with Parallel Store Accumulator Content to Memory

☐ Subtraction with Parallel Store Accumulator Content to Memory

shift quantity. The 6 LSBs of T2 define a shift quantity within -32 to +31. When

### **Example**

Syntax	Description
AC1 = rnd(T0 * *AR0+), *AR1+ = HI(AC0 << T2)	Both instructions are performed in parallel. The content addressed by AR0 is multiplied by the content of T0. Since FRCT = 1, the result is multiplied by 2, rounded, and stored in AC1. The content of AC0 is shifted by the content of T2, and AC0(31–16) is stored at the address of AR1. AR0 and AR1 are both incremented by 1.

Store Accumulator Content to Memory

Before				After			
AC0	FF	8421	1234	AC0	FF	8421	1234
AC1	00	0000	0000	AC1	00	2000	0000
AR0			0200	AR0			0201
AR1			0300	AR1			0301
TO			4000	T0			4000
T2			0004	T2			0004
200			4000	200			4000
300			1111	300			4211
FRCT			1	FRCT			1
ACOV1			0	ACOV1			0
CARRY			0	CARRY			0

# MAC

# Multiply and Accumulate (MAC)

# **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(ACy + (ACx * Tx))	Yes	2	1	Х
[2]	ACy = rnd((ACy * Tx) + ACx)	Yes	2	1	X
[3]	ACy = rnd(ACx + (Tx * K8))	Yes	3	1	X
[4]	ACy = rnd(ACx + (Tx * K16))	No	4	1	X
[5]	ACx = rnd(ACx + (Smem * coef(Cmem)))[, T3 = Smem]	No	3	1	X
[6]	ACy = rnd(ACy + (Smem * ACx))[, T3 = Smem]	No	3	1	X
[7]	ACy = rnd(ACx + (Tx * Smem))[, T3 = Smem]	No	3	1	X
[8]	ACy = rnd(ACx + (Smem * K8))[, T3 = Smem]	No	4	1	X
[9]	ACy = M40(rnd(ACx + (uns(Xmem) * uns(Ymem)))) [, T3 = Xmem]	No	4	1	Х
[10]	ACy = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(Ymem)))) [, T3 = Xmem]	No	4	1	Х
[11]	ACx = rnd(ACx + (Smem * uns(coef(Cmem))))	No	3	1	X

**Description** These instructions perform a multiplication and an accumulation in the D-unit MAC. **Status Bits** Affected by FRCT, M40, RDM, SATD, SMUL, SXMD Affects ACOVx, ACOVy See Also See the following other related instructions: ☐ Modify Auxiliary Register Content with Parallel Multiply and Accumulate Multiply and Accumulate with Parallel Delay ☐ Multiply and Accumulate with Parallel Load Accumulator from Memory Multiply and Accumulate with Parallel Multiply ☐ Multiply and Accumulate with Parallel Store Accumulator Content to Memory Multiply and Subtract

Multiply and Subtract with Parallel Multiply and Accumulate
Multiply with Parallel Multiply and Accumulate
Parallel Multiply and Accumulates

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	$ACy = \frac{\text{rnd}(ACy + (ACx * Tx))}{ACy}$	Yes	2	1	X

**Opcode** 0101 011E DDSS ss0%

Operands ACx, ACy, Tx

**Description**This instruction performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are ACx(32–16) and the content of Tx, sign extended to 17 bits.

- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- ☐ Multiplication overflow detection depends on SMUL.
- ☐ The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy.
- ☐ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ☐ Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- ☐ When an addition overflow is detected, the accumulator is saturated according to SATD.

# Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVy

**Repeat** This instruction can be repeated.

Syntax	Description
AC0 = AC0 + (AC1 * T0)	The product of the content of AC1 and the content of T0 is added to the content of AC0. The result is stored in AC0.

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACy = rnd((ACy * Tx) + ACx)	Yes	2	1	Х

**Opcode** 0101 100E DDSS ss1%

Operands ACx, ACy, Tx

**Description**This instruction performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are ACy(32–16) and the content of Tx, sign extended to 17 bits.

- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by bit.
- ☐ Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.
- ☐ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- When an addition overflow is detected, the accumulator is saturated according to SATD.

# Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVy

**Repeat** This instruction can be repeated.

Syntax Description	
AC1 = rnd((AC1 * T1) + AC0)	The product of the content of AC1 and the content of T1 is added to the content of AC0. The result is rounded and stored in AC1.

#### Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ACy = rnd(ACx + (Tx * K8))	Yes	3	1	Х

**Opcode** 

0001 111E KKKK KKKK SSDD ss1%

**Operands** 

ACx, ACy, K8, Tx

**Description** 

This instruction performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of Tx, sign extended to 17 bits, and the 8-bit signed constant, K8, sign extended to 17 bits.

- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- ☐ When an addition overflow is detected, the accumulator is saturated according to SATD.

## Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

**Status Bits** 

Affected by

FRCT, M40, RDM, SATD

Affects

ACOVy

Repeat

This instruction can be repeated.

Syntax	Description
AC0 = AC1 + (T0 * K8)	The product of the content of T0 and a signed 8-bit value is added to the content of AC1. The result is stored in AC0.

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	ACy = rnd(ACx + (Tx * K16))	No	4	1	Х

Opcode

0111 1001 KKKK KKKK KKKK KKKK SSDD ss1%

**Operands** 

ACx, ACy, K16, Tx

Description

This instruction performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of Tx, sign extended to 17 bits, and the 16-bit signed constant, K16, sign extended to 17 bits.

- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- ☐ Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.
- ☐ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- When an addition overflow is detected, the accumulator is saturated according to SATD.

# Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVy

**Repeat** This instruction can be repeated.

Syntax Description	
AC0 = AC1 + (T0 * #FFFFh)	The product of the content of T0 and a signed 16-bit value (FFFFh) is added to the content of AC1. The result is stored in AC0.

## **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	ACx = rnd(ACx + (Smem * coef(Cmem)))[, T3 = Smem]	No	3	1	Х
Opcod	<b>e</b>   1101	0001 AAA	A AA	AI U%D	D 01mm

**Operands** 

ACx, Cmem, Smem

**Description** 

This instruction performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of a memory location (Smem), sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and sign extended to 17 bits.

- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- ☐ Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVx) is set.
- ☐ When an addition overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to store the 16-bit data memory operand Smem in temporary register T3.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVx

**Repeat** This instruction can be repeated.

Syntax	Description
AC2 = rnd(AC2 + (*AR1 * coef(*CDP)))	The product of the content addressed by AR1 and the content addressed by the coefficient data pointer register (CDP) is added to the content of AC2. The result is rounded and stored in AC2. The result generated an overflow.

Before		After		
AC2	00 EC00 000	0 AC2	00 EC00	0000
AR1	030	2 AR2		0302
CDP	020	2 CDP		0202
302	FE0	0 302		FE00
202	004	0 202		0040
ACOV2		0 ACOV2		1

# **Syntax Characteristics**

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline	
[6]	ACy = rnd(ACy)	+ <b>(</b> Sı	mem * ACx <b>)</b> )[	, T3 = Smem]		No	3	1	Х	
Opcod	e				1101	0010 AAA	AA AA	AI U%D	D 00SS	
Operar	nds	AC	x, ACy, Sme	em						
			This instruction performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are ACx(32–16) and the content of a memory location (Smem), sign extended to 17 bits.							
			If FRCT =	1, the output o	f the mul	tiplier is shift	ed to t	he left by	1 bit.	
☐ Multiplication over				on overflow de	tection d	epends on S	SMUL.			
		☐ The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy.						and added		
<del>-</del>			Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.							
Addition overflow detection the destination accumulation			•				•			
	When an addition overflow is detected, the accumulator is according to SATD.				saturated					
				n provides the o	-	store the 16	-bit da	ta memoi	ry operand	
		Со	mpatibility	with C54x de	vices (C	54CM = 1)				
		Wh	nen this instr	uction is exec	uted with	M40 = 0, co	mpatib	oility is en	sured.	
Status	Bits	Aff	ected by	FRCT, M40,	RDM, SA	ATD, SMUL				
		Aff	ects	ACOVy						
Repeat	<u>:</u>	This instruction can be repeated.								
Examp	le									

Syntax	Description
AC1 = AC1 + (*AR3 * AC0)	The product of the content addressed by AR3 and the content of AC0 is added to the content of AC1. The result is stored in AC1.

# **Syntax Characteristics**

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline
[7]	ACy = rnd(ACx)	+ <b>(</b> T)	( * Smem <b>)</b> )[,	T3 = Smem]		No	3	1	Х
Opcode				1101	0100 AA	AA AA	AI U%D	D ssSS	
Operands ACx, ACy, Smem, Tx			em, Tx						
Description  This instruction performs a multi- MAC. The input operands of the r to 17 bits, and the content of a r 17 bits.				ne multip	lier are the co	ontent o	of Tx, sign	n extended	
☐ If FRCT = 1, the output			1, the output of	the mul	tiplier is shift	ed to t	he left by	1 bit.	
			Multiplication overflow detection depends on SMUL.						
		☐ The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.							
		Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.							
		Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.							
When an addition overflow is detected, the accumulator according to SATD.				ulator is	saturated				
This instruction provides the option to store the 16-bit data memory Smem in temporary register T3.				y operand					
		Со	mpatibility	with C54x dev	vices (C	54CM = 1)			
		Wh	en this instr	ruction is execu	ited with	M40 = 0, co	mpatib	ility is en	sured.
Status	Bits	Affe	ected by	FRCT, M40, F	RDM, SA	TD, SMUL			
		Affe	ects	ACOVy					
Repeat	t	This instruction can be repeated.							
Examp	le								

Syntax	Description
,	The product of the content addressed by AR3 and the content of T0 is added to the content of AC1. The result is stored in AC0.

#### **Syntax Characteristics**

				Parallel			
No.	Syntax			Enable Bit	Size	Cycles	Pipeline
[8]	ACy = rnd(ACx + (S)	mem * K8 <b>))</b> [, T3 = Sm	em]	No	4	1	Χ
Opcod	e	1111	1000 AAAA	AAAI KKK	K KK	KK SSD	D x1U%
Operar	nds AC	Cx, ACy, K8, Smem					
Descri	iption This instruction performs a multiplication and an accumulation in the D-uni						

Description

This instruction performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of a memory location (Smem), sign extended to 17 bits, and the 8-bit signed constant, K8, sign extended to 17 bits.

- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- ☐ The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.
- □ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- ☐ When an addition overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to store the 16-bit data memory operand Smem in temporary register T3.

### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by FRCT, M40, RDM, SATD

Affects ACOVy

**Repeat** This instruction can be repeated.

Syntax	Description
AC0 = AC1 + (*AR3 * #FFh)	The product of the content addressed by AR3 and a signed 8-bit value (FFh) is added to the content of AC1. The result is stored in AC0.

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[9]	ACy = M40(rnd(ACx + (uns(Xmem) * uns(Ymem)))) [, T3 = Xmem]	No	4	1	Х

## Opcode

| 1000 0110 | XXXM MMYY | YMMM SSDD | 001g uuU%

# **Operands**

ACx, ACy, Xmem, Ymem

#### Description

This instruction performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Xmem, extended to 17 bits, and the content of data memory operand Ymem, extended to 17 bits.

- ☐ Input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- ☐ When an addition overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

This instruction provides the option to store the 16-bit data memory operand Xmem in temporary register T3.

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Affected by FRCT, M40, RDM, SATD, SMUL, SXMD **Status Bits** 

> ACOVy Affects

This instruction can be repeated. Repeat

# Example

Syntax	Description
AC3 = rnd(AC3 + (uns(*AR2+) * uns(*AR3+)))	The product of the unsigned content addressed by AR2 and the unsigned content addressed by AR3 is added to the content of AC3. The result is rounded and stored in AC3. The result generated an overflow. AR2 and AR3 are both incremented by 1.

Before		After	
AC3	00 2300 EC00	AC3	00 9221 0000
AR2	302	AR2	303
AR3	202	AR3	203
ACOV3	0	ACOV3	1
302	FE00	302	FE00
202	7000	202	7000
M40	0	M4 0	0
SATD	0	SATD	0
FRCT	0	FRCT	0

Instruction Set Descriptions

## **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[10]	ACy = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(Ymem)))) [, T3 = Xmem]	No	4	1	Х

# Opcode

| 1000 0110 | XXXM MMYY | YMMM SSDD | 010g uuU%

#### **Operands**

ACx, ACy, Xmem, Ymem

# Description

This instruction performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Xmem, extended to 17 bits, and the content of data memory operand Ymem, extended to 17 bits.

- ☐ Input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- ☐ Multiplication overflow detection depends on SMUL.
- ☐ The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx, which has been shifted to the right by 16 bits. The shifting operation is performed with a sign extension of source accumulator ACx(39).
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- ☐ When an addition overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

This instruction provides the option to store the 16-bit data memory operand Xmem in temporary register T3.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Affected by FRCT, M40, RDM, SATD, SMUL, SXMD **Status Bits** 

> Affects ACOVy

This instruction can be repeated. Repeat

# **Example**

Syntax	Description
AC0 = (AC1 >> #16) + (uns(*AR3) * uns(*AR4))	The product of the unsigned content addressed by AR3 and the unsigned content addressed by AR4 is added to the content of AC1, which has been shifted to the right by 16 bits. The result is stored in AC0.

5-322 Instruction Set Descriptions

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[11]	ACx = rnd(ACx + (Smem * uns(coef(Cmem))))	No	3	1	Χ

### Opcode

| 1101 0000 | AAAA AAAI | 0%DD 10mm

### **Operands**

ACx, Cmem, Smem

#### Description

This instruction performs a multiplication and an accumulation in the D-unit MAC1. The input operands of the multiplier are the content of a data memory location (Smem) and the content of a data memory operand (Cmem).

#### Note:

The uns keyword is mandatory for this instruction.

The data memory operand Smem is addressed by DAGEN path X by using the Smem addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC1. The other data memory operand Cmem is addressed by DAGEN path C by using the coefficient addressing mode, driven on data bus BDB, and sign extended to 17 bits with filling zeros in the MAC1.

- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ☐ When an overflow is detected, the accumulator is saturated according to SATD.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To

prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

This instruction can be applied to compute the intermediate multiplication result and accumulation to the other partial result of double precision multiplication, and to free up one DAGEN operator (DAGEN path Y) for storing an instruction with enabling parallelism.

# Compatibility with C54x devices (C54CM = 1)

None.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVx

**Repeat** This instruction can be repeated.

# **Example**

Syntax	Description
	The product of the content addressed by AR3 and the content addressed by the coefficient data pointer register (CDP) is added to the content of AC0. The result is stored in AC0. AR3 is decremented by 1 and CDP in incremented by 1.

#### Execution

rnd(ACx+(Smem)[16:0]\*uns(Cmem)[16:0]) -> ACx

Before			After		
AC0	00 000	8000	AC0	FF FF00	8000
XAR3	0.0	1001	XAR3	00	1000
Data memory					
1001h		FE00	1001h		FE00
XCDP	0.0	2000	XCDP	00	2001
Coeff memory	7				
2000h		8000	2000h		8000

# **MACMZ**

#### Multiply and Accumulate with Parallel Delay

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = rnd(ACx + (Smem * coef(Cmem)))[, T3 = Smem], delay(Smem)	No	3	1	Х

#### Opcode

1101 0000 AAAA AAAI U%DD xxmm

### **Operands**

ACx, Cmem, Smem

#### Description

This instruction performs a multiplication and an accumulation in the D-unit MAC in parallel with the delay memory instruction. The input operands of the multiplier are the content of a memory location (Smem), sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and sign extended to 17 bits.

- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- ☐ Multiplication overflow detection depends on SMUL.
- ☐ The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.
- ☐ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVx) is set.
- When an addition overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to store the 16-bit data memory operand Smem in temporary register T3.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

The soft dual memory addressing mode mechanism cannot be applied to this instruction. This instruction cannot use the \*port(#k16) addressing mode or be paralleled with the readport() or writeport() operand qualifier.

This instruction cannot be used for accesses to I/O space. Any illegal access to I/O space generates a hardware bus-error interrupt (BERRINT) to be handled by the CPU.

	Со	mpatibility	vith C54x devices (C54CM = 1)					
	Wh	When this instruction is executed with M40 set to 0, compatibility						
Status Bits	Affe	ected by	FRCT, M40, RDM, SATD, SMUL					
	Affe	ects	ACOVx					
Repeat	Thi	s instruction	can be repeated.					
See Also	Sec	e the followir	g other related instructions:					
		Modify Aux	liary Register Content with Parallel Multip	ly and Accumulate				
		Multiply and	Accumulate					
		Multiply and	Accumulate with Parallel Load Accumulate	ator from Memory				
		Multiply and	Accumulate with Parallel Multiply					
		Multiply an Memory	d Accumulate with Parallel Store Accur	mulator Content to				
		Multiply and	Subtract with Parallel Multiply and Accur	mulate				
		Multiply wit	Parallel Multiply and Accumulate					
		Parallel Mu	tiply and Accumulates					

Syntax	Description
AC0 = AC0 + (*AR3 * coef(*CDP)), delay(*AR3)	The product of the content addressed by AR3 and the content addressed by the coefficient data pointer register (CDP) is added to the content of AC0. The result is stored in AC0. The content addressed by AR3 is copied into the next higher address.

# MACM::MOV

Multiply and Accumulate with Parallel Load Accumulator from Memory

# **Syntax Characteristics**

No.	Syntax		-	Parallel Enable Bit	Size	Cycles	Pipeline			
[1]	ACx = rnd(ACx ACy = Ymem <			No	4	1	X			
Opcod	e		1000 0110   XXXM	MMYY   YMM	MM DE	DD   101	.x ssU%			
Opera	nds	AC	x, ACy, Tx, Xmem, Ymem							
Descri	ption		is instruction performs two operati AC) and load.	ions in paralle	l: multi	ply and a	ccumulate			
		The first operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of Tx, sign extended to 17 bits, and the content of data memory operand Xmem, sign extended to 17 bits.								
			☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.							
			☐ Multiplication overflow detection depends on SMUL.							
			The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.							
			Rounding is performed accordinapplied to the instruction.	iding is performed according to RDM, if the optional rnd keyword ed to the instruction.						
			•	tion depends on M40. If an overflow is detect ator overflow status bit (ACOVx) is set.						
			When an addition overflow is detected, the accumulator is saturat according to SATD.							
		☐ This instruction provides the option to store the 16-bit data memo operand Xmem in temporary register T3.								
		The second operation loads the content of data memory operand Ymwhich has been shifted to the left by 16 bits, into accumulator ACy.								
		☐ The input operand is sign extended to 40 bits according to SXMD.								
		☐ The shift operation is equivalent to the signed shift instruction.								

☐ The input operand is shifted to the left by 16 bits according to M40.

	Compatibility	with C54x devices (C54CM = 1)
	When this instr	uction is executed with M40 = 0, compatibility is ensured.
Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
	Affects	ACOVx, ACOVy
Repeat	This instruction	can be repeated.
See Also	See the following	ng other related instructions:
	☐ Modify Aux	ciliary Register Content with Parallel Multiply and Accumulate
	☐ Multiply and	d Accumulate
	☐ Multiply and	d Accumulate with Parallel Delay
	☐ Multiply and	d Accumulate with Parallel Multiply
	☐ Multiply an Memory	nd Accumulate with Parallel Store Accumulator Content to
	☐ Multiply and	d Subtract with Parallel Load Accumulator from Memory
	☐ Multiply wit	h Parallel Multiply and Accumulate
	☐ Parallel Mu	ultiply and Accumulates

Syntax	Description
AC0 = AC0 + (T0 * *AR3), AC1 = *AR4 << #16	Both instructions are performed in parallel. The product of the content addressed by AR3 and the content of T0 is added to the content of AC0. The result is stored in AC0. The content addressed by AR4, which has been shifted to the left by 16 bits, is stored in AC1.

# MAC::MPY

# Multiply and Accumulate with Parallel Multiply

# **Syntax Characteristics**

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline		
[1]	$ACx = \frac{M40(rnd(AC))}{ACy} = \frac{M40(rnd(un))}{M40(rnd(un))}$			* uns(coef(Cmem))))), oef(Cmem))))	No	4	1	Х		
[2]	$ACy = \frac{M40(rnd(AC))}{ACx} = \frac{M40(rnd(un))}{M40(rnd(un))}$			* uns(HI(coef(Cmem)))))), O(coef(Cmem)))))	No	4	1	X		
[3]	ACy = M40(rnd((A uns(HI(coef(Cmer ACx = M40(rnd(un	n <b>))))</b> )	<b>)</b> ,		No	4	1	Х		
[4]	<pre>ACy = M40(rnd(ACy + (uns(HI(Lmem)) *     uns(HI(coef(Cmem)))))), ACx = M40(rnd(uns(LO(Lmem)) * uns(LO(coef(Cmem)))))</pre>					4	1	Х		
[5]	5] ACy = M40(rnd((ACy >> #16) + (uns(HI(Lmem)) * uns(HI(coef(Cmem))))), ACx = M40(rnd(uns(LO(Lmem)) * uns(LO(coef(Cmem)))))					4	1	Х		
[6]	<pre>[6] ACy = M40(rnd((ACy &gt;&gt; #16) + (uns(Ymem) *     uns(HI(coef(Cmem))))),     ACx = M40(rnd(uns(Xmem) * uns(LO(coef(Cmem)))))</pre>					5	1	Х		
Desc	ription	acc	These instructions perform two parallel operations in one cycle: multiply accumulate (MAC) and multiply. The operations are executed in the two IMACs.							
Statu	s Bits	Affe	ected by	FRCT, M40, RDM, SAT	D, SMUL, S	XMD				
		Affe	fects ACOVx, ACOVy							
See A	Also	See	See the following other related instructions:							
			☐ Modify Auxiliary Register Content with Parallel Multiply and Accumulate							
			☐ Multiply and Accumulate							
			Multiply and Accumulate with Parallel Delay							
			☐ Multiply and Accumulate with Parallel Load Accumulator from Memory							
			Multiply and Accumulate with Parallel Store Accumulator Content to Memory							
			☐ Multiply and Subtract with Parallel Multiply							
			Multiply wi	th Parallel Multiply and A	ccumulate					
			Parallel M	ultiply and Accumulates						

# Multiply and Accumulate With Parallel Multiply

## **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = M40(rnd(ACx + (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd(uns(Ymem) * uns(coef(Cmem))))	No	4	1	Х

#### Opcode

1000 0010 XXXM MMYY YMMM 01mm uuDD DDg%

### **Operands**

ACx, ACy, Cmem, Xmem, Ymem

### Description

This instruction performs two parallel operations in one cycle: multiply and accumulate (MAC) and multiply. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Xmem, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

This second operation performs a multiplication in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Ymem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

- ☐ Input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.
- For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits.
- □ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.

Overflow	detection	depends	on	M40.	lf	an	overflow	is	detected,	the
destination accumulator overflow status bit is set.										

☐ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Each data flow can also disable the usage of the corresponding MAC unit, while allowing the modification of auxiliary registers in the three address generation units through the following instructions:

- mar(Xmem)
- mar(Ymem)
- mar(Cmem)

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVx, ACOVy

**Repeat** This instruction can be repeated.

Syntax	Description
AC0 = AC0 + (uns(*AR3) * uns(coef(*CDP))), AC1 = uns(*AR4) * uns(coef(*CDP))	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the coefficient data pointer register (CDP) is added to the content of AC0. The result is stored in AC0. The product of the unsigned content addressed by AR4 and the unsigned content addressed by CDP is stored in AC1.

# Multiply and Accumulate With Parallel Multiply

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACy = M40(rnd(ACy + (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(uns(Smem) * uns(LO(coef(Cmem)))))	No	4	1	Х

#### **Opcode**

| 1111 | 1101 | AAAA | AAAI | 0000 | 10mm | DDDD | uug%

#### **Operands**

ACx, ACy, Cmem, Smem

### Description

This instruction performs two parallel operations in one cycle: multiply and accumulate (MAC) and multiply. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand HI(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC2 (this data is shared to MAC1 and MAC2). The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA (effective address); the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand LO(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

The content of the memory location is zero extended to 17 bits, if the	
optional uns keyword is applied to the input operand.	

- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.

	For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy.
	For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits.
	Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
	Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
	When an overflow is detected, the accumulator is saturated according to SATD.
<b>-</b>	

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

# Compatibility with C54x devices (C54CM = 1)

None.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVx, ACOVy

**Repeat** This instruction can be repeated.

### **Example**

Syntax	Description
AC1 = AC1 + (uns(*AR3-) * uns(HI(coef(*CDP+)))), AC0 = uns(*AR3-) * uns(LO(coef(*CDP+)))	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the lower part of CDP is stored in AC0. AR3 is decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

#### Execution

ACy+M40(rnd(uns(Smem) [16:0] \*uns(HI(coef(Cmem))) [16:0])) -> ACy
M40(rnd(uns(Smem) [16:0] \*uns(LO(coef(Cmem))) [16:0])) -> ACx

# MAC::MPY Multiply and Accumulate with Parallel Multiply

Before				After			
AC0	FF 8	3000	0000	AC0	00	3F80	0000
XAR3		00	10FF	XAR3		00	10FE
Data memory							
10FFh			FE00	10FFh			FE00
XCDP		00	2000	XCDP		00	2002
Coeff memor	У						
2001h			4000	2001h			4000
AC1	00 0	000	8000	AC1	00	7F00	8000
Coeff memor	У						
2000h			8000	2000h			8000

5-334 Instruction Set Descriptions SWPU068E

# Multiply and Accumulate With Parallel Multiply

## **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ACy = M40(rnd((ACy >> #16) + (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(uns(Smem) * uns(LO(coef(Cmem)))))	No	4	1	Х

## **Opcode**

| 1111 | 1101 | AAAA | AAAI | 0010 | 10mm | DDDD | uug%

**Operands** 

ACx, ACy, Cmem, Smem

## **Description**

This instruction performs two parallel operations in one cycle: multiply and accumulate (MAC) and multiply. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand HI(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC2 (this data is shared to MAC1 and MAC2). The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA (effective address); the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand LO(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

	The content of the memory location is zero extended to 17 bits, if the
	optional uns keyword is applied to the input operand.
$\Box$	If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.

Multiplication overflow detection depends on SMUL.

_	extended to 40 bits and added to the source accumulator ACy, which has been shifted to the right by 16 bits. The shifting operation is performed with a sign extension of source accumulator ACy(39).
	For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits.
	Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
	Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
	When an overflow is detected, the accumulator is saturated according to SATD.

For the first operation, the 32-bit result of the multiplication is sign

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

## Compatibility with C54x devices (C54CM = 1)

None.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVx, ACOVy

**Repeat** This instruction can be repeated.

## Example

Syntax	Description
AC1 = (AC1 >> #16) + (uns(*AR3-) * uns(HI(coef(*CDP+)))), AC0 = uns(*AR3-) * uns(LO(coef(*CDP+)))	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1, which has been shifted to the right by 16 bits. The result is stored in AC1. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the lower part of CDP is stored in AC0. AR3 is decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

## Execution

(ACy>>#16)+M40(rnd(uns(Smem)[16:0]\*uns(HI(coef(Cmem)))[16:0])) -> ACy M40(rnd(uns(Smem) [16:0] \*uns(LO(coef(Cmem))) [16:0])) -> ACx

Before				After			
AC0	FF	8000	0000	AC0	00	3F80	0000
XAR3		00	10FF	XAR3		00	10FE
Data memory							
10FFh			FE00	10FFh			FE00
XCDP		00	2000	XCDP		00	2002
Coeff memor	У						
2001h			4000	2001h			4000
AC1	00	0800	0000	AC1	00	7F00	0800
Coeff memor	У						
2000h			8000	2000h			8000

# Multiply and Accumulate With Parallel Multiply

## **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	ACy = M40(rnd(ACy + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(uns(LO(Lmem)) * uns(LO(coef(Cmem)))))	No	4	1	Х

#### **Opcode**

| 1111 | 1101 | AAAA | AAAI | 0100 | 10mm | DDDD | uug%

## **Operands**

ACx, ACy, Cmem, Lmem

## Description

This instruction performs two parallel operations in one cycle: multiply and accumulate (MAC) and multiply. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand HI(Lmem) and the content of data memory operand HI(coef(Cmem)). The data memory operand HI(Lmem) is addressed by DAGEN path X with the EA (effective address); the data, which can be assumed to be the higher part of long word memory data, is driven on data bus CDB and sign extended to 17 bits in the MAC2. The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand LO(Lmem) and the content of data memory operand LO(coef(Cmem)). The data memory operand LO(Lmem) is addressed by DAGEN path X with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word memory data, is driven on data bus DDB and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

The content of the memory location is zero extended to 17 bits, if the
optional uns keyword is applied to the input operand.

<sup>☐</sup> If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.

		Multiplication	n overflow dete	ection depe	nds on SN	ИUL.		
			st operation, the 40 bits and ad					n
		For the sec	ond operation, 40 bits.	the 32-bit	result of	the mult	iplication is sig	n
		•	s performed acone instruction.	cording to F	RDM, if the	e optiona	ıl rnd keyword i	is
			etection depen accumulator ov				is detected, th	е
		When an ov	verflow is detec	ted, the acc	cumulator	is satura	ted according t	:0
	This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.							
	and bus pre	B2DB buse es are only	ion, the Cmem s; on some C5: connected to in eration of a bus mory.	5xx-based ternal mem	devices, the	ne BAB, ot to exte	BDB, and B2Dernal memory. T	В
	Co	mpatibility	with C54x devi	ices (C54C	CM = 1)			
	Nor	ne.						
Status Bits	Affe	ected by	FRCT, M40, R	DM, SATD	, SMUL			
	Affe	ects	ACOVx, ACO	/y				
Repeat	This	s instruction	can be repeate	ed.				

# Example

Syntax	Description
AC1 = AC1 + (uns(HI(*AR3-)) * uns(HI(coef(*CDP+)))), AC0 = uns(LO(*AR3-)) * uns(LO(coef(*CDP+)))	Both instructions are performed in parallel. The product of the unsigned content addressed by the higher part of AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by the lower part of AR3 and the unsigned content addressed by the lower part of CDP is stored in AC0. When AR3– is used with HI/LO, AR3 is decremented by 2. When CDP+ is used with HI/LO, CDP is incremented by 2.

## Execution

ACy+M40(rnd(uns(HI(Lmem))[16:0]\*uns(HI(coef(Cmem)))[16:0])) -> ACy
M40(rnd(uns(LO(Lmem))[16:0]\*uns(LO(coef(Cmem)))[16:0])) -> ACx

Before				After			
AC0	FF	8000	0000	AC0	00	3F80	0000
XAR3		00	10FE	XAR3		00	10FC
Data memory							
10FFh			FE00	10FFh			FE00
XCDP		00	2000	XCDP		00	2002
Coeff memor	У						
2001h			4000	2001h			4000
AC1	00	0000	8000	AC1	00	7F80	8000
Data memory							
10FEh			FF00	10FEh			FF00
Coeff memor	У						
2000h			8000	2000h			8000

5-340 Instruction Set Descriptions

# Multiply and Accumulate With Parallel Multiply

# **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	ACy = M40(rnd((ACy>>#16) + (uns(HI(Lmem)) * uns(HI(coef(Cmem))))), ACx = M40(rnd(uns(LO(Lmem)) * uns(LO(coef(Cmem)))))	No	4	1	Х

#### Opcode

| 1111 | 1101 | AAAA | AAAI | 0110 | 10mm | DDDD | uug%

## **Operands**

ACx, ACy, Cmem, Lmem

## Description

This instruction performs two parallel operations in one cycle: multiply and accumulate (MAC) and multiply. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand HI(Lmem) and the content of data memory operand HI(coef(Cmem)). The data memory operand HI(Lmem) is addressed by DAGEN path X with the EA (effective address); the data, which can be assumed to be the higher part of long word memory data, is driven on data bus CDB and sign extended to 17 bits in the MAC2. The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand LO(Lmem) and the content of data memory operand LO(coef(Cmem)). The data memory operand LO(Lmem) is addressed by DAGEN path X with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word memory data, is driven on data bus DDB and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.
- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.

		Multiplication	on overnow detection depends on SiviOL.
		extended to been shifted	st operation, the 32-bit result of the multiplication is sign o 40 bits and added to the source accumulator ACy, which has d to the right by 16 bits. The shifting operation is performed with nsion of source accumulator ACy(39).
		For the sec	cond operation, the 32-bit result of the multiplication is sign a 40 bits.
		_	s performed according to RDM, if the optional rnd keyword is he instruction.
			etection depends on M40. If an overflow is detected, the accumulator overflow status bit is set.
		When an or SATD.	verflow is detected, the accumulator is saturated according to
			provides the option to locally set M40 to 1 for the execution of if the optional M40 keyword is applied to the instruction.
	and bus pre	d B2DB buse ses are only	tion, the Cmem operand is accessed through the BAB, BDB, es; on some C55xx-based devices, the BAB, BDB, and B2DB connected to internal memory and not to external memory. To teration of a bus error, the Cmem operand must not be mapped mory.
	Co	mpatibility	with C54x devices (C54CM = 1)
	Noi	ne.	
Status Bits	Affe	ected by	FRCT, M40, RDM, SATD, SMUL
	Affe	ects	ACOVx, ACOVy
Repeat	Thi	s instruction	can be repeated.

# Example

Syntax	Description
AC1 = (AC1 >> #16) + (uns(HI(*AR3-)) * uns(HI(coef(*CDP+)))), AC0 = uns(LO(*AR3-)) * uns(LO(coef(*CDP+)))	Both instructions are performed in parallel. The product of the unsigned content addressed by the higher part of AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1, which has been shifted to the right by 16 bits. The result is stored in AC1. The product of the unsigned content addressed by the lower part of AR3 and the unsigned content addressed by the lower part of CDP is stored in AC0. When AR3—is used with HI/LO, AR3 is decremented by 2. When CDP+ is used with HI/LO, CDP is incremented by 2.

#### Execution

(ACy>>#16) +M40(rnd(uns(HI(Lmem))[16:0]\*uns(HI(coef(Cmem)))[16:0])) -> ACy M40(rnd(uns(LO(Lmem))[16:0]\*uns(LO(coef(Cmem)))[16:0])) -> ACx

Before				After			
AC0	FF	8000	0000	AC0	00	3F80	0000
XAR3		00	10FE	XAR3		00	10FC
Data memory							
10FFh			FE00	10FFh			FE00
XCDP		00	2000	XCDP		00	2002
Coeff memor	У						
2001h			4000	2001h			4000
AC1	00	0800	0000	AC1	00	7F80	0800
Data memory							
10FEh			FF00	10FEh			FF00
Coeff memor	У						
2000h			8000	2000h			8000

# Multiply and Accumulate With Parallel Multiply

## **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(uns(Xmem) * uns(LO(coef(Cmem)))))	No	5 (*)	1	Х

(\*) 1 LSB is allocated to instruction slot #2.

Opcode

1001 0100 XXXM MMYY YMMM 10mm uuDD DDg%

**Operands** 

ACx, ACy, Cmem, Xmem, Ymem

Description

This instruction performs two parallel operations in one cycle: multiply and accumulate (MAC) and multiply. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the contents of data memory operand Ymem, extended to 17 bits, and the content of data memory operand HI(coef(Cmem)) which is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication in the D-unit MAC1. The input operands of the multiplier are the contents of data memory operand Xmem, extended to 17 bits, and the content of data memory operand LO(coef(Cmem)) which is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and extended to 17 bits in the MAC1.

- ☐ The input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.

	The 32-bit result of the multiplication is sign extended to 40 bits.
	Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
	Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
	When an overflow is detected, the accumulator is saturated according to SATD.
	Because this instruction occupies both instruction slots #1 and #2, this can not be executed in parallel with other instructions.
	The Xmem operand can access the MMRs but the Ymem operand can not.
Thi	s instruction provides the option to locally set M40 to 1 for the execution of

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

# Compatibility with C54x devices (C54CM = 1)

None.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVx, ACOVy

**Repeat** This instruction can be repeated.

## Example

Syntax	Description
AC1 = (AC1 >> #16) + (uns(*AR3-) * uns(Hl(coef(*CDP+)))), AC0 = uns(*AR2-) * uns(LO(coef(*CDP+)))	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1, which has been shifted to the right by 16 bits. The result is stored in AC1. The product of the unsigned content addressed by AR2 and the unsigned content addressed by the lower part of the CDP is stored in AC0. AR3 and AR2 are decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

## Execution

M40(rnd(uns(Xmem)[16:0] \* uns(LO(coef(Cmem)))[16:0])) -> ACx
M40(rnd((ACy >> #16) + uns(Ymem)[16:0] \* uns(HI(coef(Cmem)))[16:0])) -> ACy

Before				After			
AC0	FF 8	3000	0000	AC0	00	3F80	0000
XAR2		00	10FE	XAR2		00	10FD
XAR3		00	20FE	XAR3		00	20FD
Data memory							
10FEh			FE00	10FEh			FE00
XCDP		00	2000	XCDP		00	2002
Coeff memory	7						
2001h			4000	2001h			4000
AC1	00 0	0800	0000	AC1	00	7F80	8000
Data memory							
20FEh			FF00	20FFh			FF00
Coeff memory	7						
2000h			8000	2000h			8000

5-346 Instruction Set Descriptions

# MAC::MAS

# Multiply and Accumulate With Parallel Multiply and Subtract

# **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = M40(rnd(ACy + (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx - (uns(Smem) * uns(LO(coef(Cmem))))))	No	4	1	Х
[2]	ACy = M40(rnd((ACy >> #16) + (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx - (uns(Smem) * uns(LO(coef(Cmem)))))))	No	4	1	Х
[3]	ACy = M40(rnd(ACy + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx - (uns(LO(Lmem)) * uns(LO(coef(Cmem)))))))	No	4	1	Х
[4]	ACy = M40(rnd((ACy >> #16) + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))),  ACx = M40(rnd(ACx - (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))	No	4	1	Х
[5]	ACy = M40(rnd(ACy + uns(Ymem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx - uns(Xmem) * uns(LO(coef(Cmem)))))	No	5	1	X
[6]	ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx - (uns(Xmem) * uns(LO(coef(Cmem))))))	No	5	1	Х

**Description** These instructions perform two parallel operations in one cycle: multiply and

accumulate (MAC) and multiply and subtract (MAS). The operations are

executed in the two D-unit MACs.

**Status Bits** Affected by FRCT, M40, RDM, SATD, SMUL

> Affects ACOVx, ACOVy

See Also	See the following other related instructions:
	☐ Modify Auxiliary Register Content with Parallel Multiply and Subtract
	☐ Multiply and Subtract
	☐ Multiply and Subtract with Parallel Load Accumulator from Memory
	☐ Multiply and Subtract with Parallel Store Accumulator Content to Memory
	☐ Multiply and Subtract with Parallel Multiply
	☐ Parallel Multiply and Subtracts

# Multiply and Accumulate With Parallel Multiply and Subtract

# **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = M40(rnd(ACy + (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx - (uns(Smem) * uns(LO(coef(Cmem))))))	No	4	1	Х

**Opcode** 

| 1111 | 1101 | AAAA | AAAI | 0001 | 10mm | DDDD | uug%

**Operands** 

ACx, ACv, Cmem, Smem

Description

This instruction performs two parallel operations in one cycle: multiply and accumulate (MAC) and multiply and subtract (MAS). The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand HI(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC2 (this data is shared to MAC1 and MAC2). The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA (effective address); the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and a subtraction in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand LO(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.
- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.

	Multiplication overflow detection depends on SMUL.
	For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy.
	For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACx.
	Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
	Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
	When an overflow is detected, the accumulator is saturated according to SATD.
	s instruction provides the option to locally set M40 to 1 for the execution o instruction, if the optional M40 keyword is applied to the instruction.
and bus pre	this instruction, the Cmem operand is accessed through the BAB, BDB B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DE ses are only connected to internal memory and not to external memory. To vent the generation of a bus error, the Cmem operand must not be mapped external memory.
Co	mpatibility with C54x devices (C54CM = 1)
Nor	ne.

**Status Bits** Affected by FRCT, M40, RDM, SATD, SMUL

> ACOVx, ACOVy Affects

Repeat This instruction can be repeated.

# **Example**

Syntax	Description
AC1 = AC1 + (uns(*AR3-) * uns(HI(coef(*CDP+)))), AC0 = AC0 - (uns(*AR3-) * uns(LO(coef(*CDP+))))	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the lower part of CDP is subtracted from the content of AC0. The result is stored in AC0. AR3 is decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

#### Execution

ACy+M40(rnd(uns(Smem)[16:0]\*uns(HI(coef(Cmem)))[16:0])) -> ACy 

I	Before				After			
Z	AC0	00	0000	8000	AC0	FF	C080	8000
2	KAR3		00	10FF	XAR3		00	10FE
Ι	Data memory							
1	L0FFh			FE00	10FFh			FE00
2	KCDP		00	2000	XCDP		00	2002
(	Coeff memor	У						
2	2001h			4000	2001h			4000
F	AC1	00	0000	8000	AC1	00	7F00	8000
(	Coeff memory							
2	2000h			8000	2000h			8000

# Multiply and Accumulate With Parallel Multiply and Subtract

# **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACy = M40(rnd((ACy >> #16) + (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx - (uns(Smem) * uns(LO(coef(Cmem))))))	No	4	1	Х

**Opcode** 

| 1111 | 1101 | AAAA | AAAI | 0010 | 01mm | DDDD | uug%

**Operands** 

ACx, ACy, Cmem, Smem

Description

This instruction performs two parallel operations in one cycle: multiply and accumulate (MAC) and multiply and subtract (MAS). The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand HI(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC2 (this data is shared to MAC1 and MAC2). The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA (effective address); the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and a subtraction in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand LO(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

The content of the memory location is zero extended to 17 bits, if the
optional uns keyword is applied to the input operand.

<sup>☐</sup> If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.

		Multiplication	on overflow detection depends on SMUL.		
		extended to been shifted	st operation, the 32-bit result of the multiplication is sign o 40 bits and added to the source accumulator ACy, which has d to the right by 16 bits. The shifting operation is performed with nation of source accumulator ACy(39).		
			cond operation, the 32-bit result of the multiplication is sign a 40 bits and subtracted from the source accumulator ACx.		
		_	s performed according to RDM, if the optional rnd keyword is he instruction.		
			etection depends on M40. If an overflow is detected, the accumulator overflow status bit is set.		
		When an ov	verflow is detected, the accumulator is saturated according to		
			provides the option to locally set M40 to 1 for the execution of if the optional M40 keyword is applied to the instruction.		
	and bus pre	For this instruction, the Cmem operand is accessed through the B and B2DB buses; on some C55xx-based devices, the BAB, BDB, a buses are only connected to internal memory and not to external m prevent the generation of a bus error, the Cmem operand must not b on external memory.			
	Co	mpatibility	with C54x devices (C54CM = 1)		
	Nor	ne.			
Status Bits	Affe	ected by	FRCT, M40, RDM, SATD, SMUL		
	Affe	ects	ACOVx, ACOVy		
Repeat	Thi	s instruction	can be repeated.		

# **Example**

Syntax	Description
AC1 = (AC1 >> #16) + (uns(*AR3-) * uns(HI(coef(*CDP+)))), AC0 = AC0 - (uns(*AR3-) * uns(LO(coef(*CDP+))))	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1, which has been shifted to the right by 16 bits. The result is stored in AC1. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the lower part of CDP is subtracted from the content of AC0. The result is stored in AC0. AR3 is decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

#### Execution

(ACy>>#16) +M40 (rnd (uns (Smem) [16:0] \*uns (HI (coef (Cmem))) [16:0])) -> ACy ACx-M40(rnd(uns(Smem)[16:0]\*uns(LO(coef(Cmem)))[16:0])) -> ACx

Before				After			
AC0	00	0000	8000	AC0	FF	C080	8000
XAR3		00	10FF	XAR3		00	10FE
Data memory							
10FFh			FE00	10FFh			FE00
XCDP		00	2000	XCDP		00	2002
Coeff memor	У						
2001h			4000	2001h			4000
AC1	00	0800	0000	AC1	00	7F00	0800
Coeff memory							
2000h			8000	2000h			8000

# Multiply and Accumulate With Parallel Multiply and Subtract

## **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ACy = M40(rnd(ACy + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx - (uns(LO(Lmem)) * uns(LO(coef(Cmem)))))))	No	4	1	Х

## Opcode

| 1111 | 1101 | AAAA | AAAI | 0101 | 10mm | DDDD | uug%

## **Operands**

ACx, ACy, Cmem, Lmem

#### Description

This instruction performs two parallel operations in one cycle: multiply and accumulate (MAC) and multiply and subtract (MAS). The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand HI(Lmem) and the content of data memory operand HI(coef(Cmem)). The data memory operand HI(Lmem) is addressed by DAGEN path X with the EA (effective address); the data, which can be assumed to be the higher part of long word memory data, is driven on data bus CDB and sign extended to 17 bits in the MAC2. The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and a subtraction in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand LO(Lmem) and the content of data memory operand LO(coef(Cmem)). The data memory operand LO(Lmem) is addressed by DAGEN path X with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word memory data, is driven on data bus DDB and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.

If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
Multiplication overflow detection depends on SMUL.
For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy.
For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACx.
Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

## Compatibility with C54x devices (C54CM = 1)

None.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVx, ACOVy

**Repeat** This instruction can be repeated.

## Example

Syntax	Description
AC1 = AC1 + (uns(HI(*AR3-)) * uns(HI(coef(*CDP+)))), AC0 = AC0 - (uns(LO(*AR3-)) * uns(LO(coef(*CDP+))))	Both instructions are performed in parallel. The product of the unsigned content addressed by the higher part of AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by lower part of AR3 and the unsigned content addressed by the lower part of CDP is subtracted from the content of AC0. The result is stored in AC0. When AR3—is used with HI/LO, AR3 is decremented by 2. When CDP+ is used with HI/LO, CDP is incremented by 2.

## Execution

ACy+M40(rnd(uns(HI(Lmem))[16:0]\*uns(HI(coef(Cmem)))[16:0])) -> ACy ACx-M40(rnd(uns(LO(Lmem))[16:0]\*uns(LO(coef(Cmem)))[16:0])) -> ACx

Before				After			
AC0	00	0000	8000	AC0	FF	C080	8000
XAR3		00	10FE	XAR3		00	10FC
Data memory							
10FFh			FE00	10FFh			FE00
XCDP		00	2000	XCDP		00	2002
Coeff memory	Y						
2001h			4000	2001h			4000
AC1	00	0000	8000	AC1	00	7F80	0000
Data memory							
10FEh			FF00	10FEh			FF00
Coeff memory	Y						
2000h			8000	2000h			8000

# Multiply and Accumulate With Parallel Multiply and Subtract

## **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	ACy = M40(rnd((ACy >> #16) + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx - (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))	No	4	1	Х

#### Opcode

| 1111 | 1101 | AAAA | AAAI | 0110 | 01mm | DDDD | uug%

## **Operands**

ACx, ACy, Cmem, Lmem

## Description

This instruction performs two parallel operations in one cycle: multiply and accumulate (MAC) and multiply and subtract (MAS). The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand HI(Lmem) and the content of data memory operand HI(coef(Cmem)). The data memory operand HI(Lmem) is addressed by DAGEN path X with the EA (effective address); the data, which can be assumed to be the higher part of long word memory data, is driven on data bus CDB and sign extended to 17 bits in the MAC2. The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and a subtraction in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand LO(Lmem) and the content of data memory operand LO(coef(Cmem)). The data memory operand LO(Lmem) is addressed by DAGEN path X with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word memory data, is driven on data bus DDB and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

☐ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.

		If FRCT = 1	1, the output of the multiplier is shifted to the left by 1 bit.	
		Multiplication	on overflow detection depends on SMUL.	
		extended to been shifted	est operation, the 32-bit result of the multiplication is sign o 40 bits and added to the source accumulator ACy, which has d to the right by 16 bits. The shifting operation is performed with ension of source accumulator ACy(39).	
			cond operation, the 32-bit result of the multiplication is sign o 40 bits and subtracted from the source accumulator ACx.	
		_	is performed according to RDM, if the optional rnd keyword is the instruction.	
		detection depends on M40. If an overflow is detected, the accumulator overflow status bit is set.		
		verflow is detected, the accumulator is saturated according to		
		his instruction provides the option to locally set M40 to 1 for the execute instruction, if the optional M40 keyword is applied to the instruction		
	and bus pre	or this instruction, the Cmem operand is accessed through the BAB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and uses are only connected to internal memory and not to external memorevent the generation of a bus error, the Cmem operand must not be man external memory.		
	Co	mpatibility	with C54x devices (C54CM = 1)	
	Nor	ne.		
Status Bits	Affe	ected by	FRCT, M40, RDM, SATD, SMUL	
	Affe	ects	ACOVx, ACOVy	
Repeat	This	s instruction	can be repeated.	

# Example

Syntax	Description
AC1 = (AC1>>#16) + (uns(HI(*AR3-)) * uns(HI(coef(*CDP+)))), AC0 = AC0 - (uns(LO(*AR3-)) * uns(LO(coef(*CDP+))))	Both instructions are performed in parallel. The product of the unsigned content addressed by the higher part of AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1, which has been shifted to the right by 16 bits. The result is stored in AC1. The product of the unsigned content addressed by lower part of AR3 and the unsigned content addressed by the lower part of CDP is subtracted from the content of AC0. The result is stored in AC0. When AR3– is used with HI/LO, AR3 is decremented by 2. When CDP+ is used with HI/LO, CDP is incremented by 2.

## Execution

(ACy>>#16) +M40(rnd(uns(HI(Lmem))[16:0]\*uns(HI(coef(Cmem)))[16:0])) -> ACy 

Before				After			
AC0	00	0000	8000	AC0	FF	C080	8000
XAR3		00	10FE	XAR3		00	10FC
Data memory							
10FFh			FE00	10FFh			FE00
XCDP		00	2000	XCDP		00	2002
Coeff memor	У						
2001h			4000	2001h			4000
AC1	00	0800	0000	AC1	00	7F80	0800
Data memory							
10FEh			FF00	10FEh			FF00
Coeff memor	У						
2000h			8000	2000h			8000

# Multiply and Accumulate With Parallel Multiply and Subtract

## **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	ACy = M40(rnd(ACy + uns(Ymem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx - uns(Xmem) * uns(LO(coef(Cmem)))))	No	5 (*)	1	Х

(\*) 1 LSB is allocated to instruction slot #2.

## **Opcode**

1001 0011 XXXM MMYY YMMM 01mm uuDD DDg%

#### **Operands**

ACx, ACy, Cmem, Xmem, Ymem

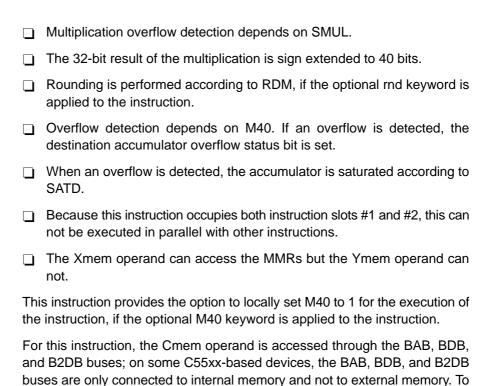
## Description

This instruction performs two parallel operations in one cycle: multiply and accumulate (MAC) and multiply and subtract (MAS). The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the contents of data memory operand Ymem, extended to 17 bits, and the content of data memory operand HI(coef(Cmem)) which is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and a subtraction in the D-unit MAC1. The input operands of the multiplier are the contents of data memory operand Xmem, extended to 17 bits, and the content of data memory operand LO(coef(Cmem)) which is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- ☐ The input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.



prevent the generation of a bus error, the Cmem operand must not be mapped

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

on external memory.

Affects ACOVx, ACOVy

**Repeat** This instruction can be repeated.

#### **Example**

Syntax	Description
AC1 = AC1 + (uns(*AR3-) * uns(HI(coef(*CDP+)))), AC0 = AC0 - (uns(*AR2-) * uns(LO(coef(*CDP+))))	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by AR2 and the unsigned content addressed by the lower part of the CDP is subtracted from the content of AC0. The result is stored in AC0. AR3 and AR2 are decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

## Execution

M40(rnd(ACx - uns(Xmem)[16:0] \* uns(LO(coef(Cmem)))[16:0])) -> ACx M40(rnd(ACy + uns(Ymem)[16:0] \* uns(HI(coef(Cmem)))[16:0])) -> ACy

Before				After			
AC0	00	0000	8000	AC0	FF	C080	8000
XAR2		00	10FE	XAR2		00	10FD
XAR3		00	20FE	XAR3		00	20FD
Data memory							
10FEh			FE00	10FEh			FE00
XCDP		00	2000	XCDP		00	2002
Coeff memor	У						
2001h			4000	2001h			4000
AC1	00	0000	8000	AC1	00	7F80	8000
Data memory							
20FEh			FF00	20FFh			FF00
Coeff memor	У						
2000h			8000	2000h			8000

# Multiply and Accumulate With Parallel Multiply and Subtract

# **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx - (uns(Xmem) * uns(LO(coef(Cmem))))))	No	5 (*)	1	Х

(\*) 1 LSB is allocated to instruction slot #2.

**Opcode** 

1001 0100 XXXM MMYY YMMM 00mm uuDD DDg%

**Operands** 

ACx, ACy, Cmem, Xmem, Ymem

**Description** 

This instruction performs two parallel operations in one cycle: multiply and accumulate (MAC) and multiply and subtract (MAS). The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the contents of data memory operand Ymem, extended to 17 bits, and the content of data memory operand HI(coef(Cmem)) which is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and a subtraction in the D-unit MAC1. The input operands of the multiplier are the contents of data memory operand Xmem, extended to 17 bits, and the content of data memory operand LO(coef(Cmem)) which is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- ☐ The input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.

		Multiplication	ion overflow detection depends on SMUL.
		The 32-bit	result of the multiplication is sign extended to 40 bits.
		•	is performed according to RDM, if the optional rnd keyword is the instruction.
			detection depends on M40. If an overflow is detected, the accumulator overflow status bit is set.
		When an o SATD.	overflow is detected, the accumulator is saturated according to
			his instruction occupies both instruction slots #1 and #2, this can ecuted in parallel with other instructions.
		The Xmem not.	n operand can access the MMRs but the Ymem operand can
			n provides the option to locally set M40 to 1 for the execution of if the optional M40 keyword is applied to the instruction.
	and bus pre	B2DB buse are only co	ction, the Cmem operand is accessed through the BAB, BDB, les; on some C55xx-based devices, the BAB, BDB, and B2DB connected to internal memory and not to external memory. To neration of a bus error, the Cmem operand must not be mapped emory.
	Co	mpatibility	with C54x devices (C54CM = 1)
	Noi	ne.	
Status Bits	Affe	ected by	FRCT, M40, RDM, SATD, SMUL, SXMD

# Example

Repeat

Syntax	Description
AC1 = (AC1 >> #16) + (uns(*AR3-) * uns(HI(coef(*CDP+)))), AC0 = AC0 - (uns(*AR2-) * uns(LO(coef(*CDP+))))	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1, which has been shifted to the right by 16 bits. The result is stored in AC1. The product of the unsigned content addressed by AR2 and the unsigned content addressed by the lower part of the CDP is added to the content of AC0. The result is stored in AC0. AR3 and AR2 are decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

ACOVx, ACOVy

This instruction can be repeated.

Affects

## Execution

M40(rnd((ACy >> #16) + uns(Ymem)[16:0] \* uns(HI(coef(Cmem)))[16:0])) -> ACy

Before				After			
AC0	00	0000	8000	AC0	FF	C080	8000
XAR2		00	10FE	XAR2		00	10FD
XAR3		00	20FE	XAR3		00	20FD
Data memory							
10FEh			FE00	10FEh			FE00
XCDP		00	2000	XCDP		00	2002
Coeff memor	У						
2001h			4000	2001h			4000
AC1	00	0800	0000	AC1	00	7F80	8000
Data memory							
20FEh			FF00	20FFh			FF00
Coeff memor	У						
2000h			8000	2000h			8000

Instruction Set Descriptions

# MACM::MOV

Multiply and Accumulate with Parallel Store Accumulator Content to Memory

## **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(ACy + (Tx * Xmem)), Ymem = HI(ACx << T2) [,T3 = Xmem]	No No	4		Х
Opcod	e   1000 0111   XX	XM MMYY YMM	MM SS	SDD 001	.x ssU%
Operai	ACx, ACy, Tx, Xmem, Ymem				
Descri	This instruction performs two oper	ations in paralle	l: multi	ply and a	ccumulate

(MAC) and store.

The first operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of Tx, sign extended to 17 bits, and the content of data memory operand Xmem, sign extended to 17 bits.

If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
Multiplication overflow detection depends on SMUL.
The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy.
Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
When an addition overflow is detected, the accumulator is saturated according to SATD.
This instruction provides the option to store the 16-bit data memory operand Xmem in temporary register T3.

The second operation shifts the accumulator ACx by the content of T2 and stores ACx(31–16) to data memory operand Ymem. If the 16-bit value in T2 is not within -32 to +31, the shift is saturated to -32 or +31 and the shift is performed with this value.

- ☐ The input operand is shifted in the D-unit shifter according to SXMD.
- After the shift, the high part of the accumulator, ACx(31–16), is stored to the memory location.

## Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When this instruction is executed with C54CM = 1, the 6 LSBs of T2 are used to determine the shift quantity. The 6 LSBs of T2 define a shift quantity within -32 to +31. When the 16-bit value in T2 is between -32 to -17, a modulo 16 operation transforms the shift quantity to within -16 to -1.

☐ If the SST bit = 1 and the SXMD bit = 0, then the saturate and uns keywords are applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:

ACy = rnd(ACy + (Tx \* Xmem)), Ymem = HI(saturate(uns(ACx << T2))) [,T3 = Xmem]

☐ If the SST bit = 1 and the SXMD bit = 1, then only the saturate keyword is applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:

ACy = rnd(ACy + (Tx \* Xmem)), Ymem = HI(saturate(ACx << T2)) [,T3 = Xmem]

**Status Bits** 

Affected by

C54CM, FRCT, M40, RDM, SATD, SMUL, SST, SXMD

Affects ACOVy

Repeat

This instruction can be repeated.

See Also

See the following other related instructions:

- Modify Auxiliary Register Content with Parallel Multiply and Accumulate
- Multiply and Accumulate
- Multiply and Accumulate with Parallel Delay
- Multiply and Accumulate with Parallel Load Accumulator from Memory
- Multiply and Accumulate with Parallel Multiply
- ☐ Multiply and Subtract with Parallel Store Accumulator Content to Memory
- Multiply with Parallel Multiply and Accumulate
- Parallel Multiply and Accumulates

#### Example

Syntax	Description
AC0 = AC0 + (T0 * *AR3), *AR4 = HI(AC1 << T2)	Both instructions are performed in parallel. The product of the content addressed by AR3 and the content of T0 is added to the content of AC0. The result is stored in AC0. The content of AC1 is shifted by the content of T2, and AC1(31–16) is stored at the address of AR4.

# MAS

# Multiply and Subtract

# **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(ACy - (ACx * Tx))	Yes	2	1	Х
[2]	ACx = rnd(ACx - (Smem * coef(Cmem)))[, T3 = Smem]	No	3	1	X
[3]	ACy = rnd(ACy - (Smem * ACx))[, T3 = Smem]	No	3	1	X
[4]	ACy = rnd(ACx - (Tx * Smem))[, T3 = Smem]	No	3	1	X
[5]	ACy = M40(rnd(ACx - (uns(Xmem) * uns(Ymem)))) [, T3 = Xmem]	No	4	1	Х
[6]	ACx = rnd(ACx - (Smem * uns(coef(Cmem))))	No	3	1	X

**Description** These instructions perform a multiplication and a subtraction in the D-unit MAC. **Status Bits** Affected by FRCT, M40, RDM, SATD, SMUL, SXMD Affects ACOVx, ACOVy See Also See the following other related instructions: ☐ Modify Auxiliary Register Content with Parallel Multiply and Subtract Multiply and Accumulate ☐ Multiply and Subtract with Parallel Load Accumulator from Memory Multiply and Subtract with Parallel Multiply ☐ Multiply and Subtract with Parallel Multiply and Accumulate ☐ Multiply and Subtract with Parallel Store Accumulator Content to Memory Parallel Multiply and Subtracts

# **Syntax Characteristics**

No. Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[1] $ACy = rnd(ACy)$	– <b>(</b> ACx * Tx <b>)</b> )		Yes	2	1	Х
Opcode			010	01 01	.1E DDS	SS ss1%
Operands	ACx, ACy, T	x	·		·	
Description		ion performs a multiplicati erands of the multiplier are 17 bits.				
	☐ If FRCT	= 1, the output of the mul	ltiplier is shif	ted to t	he left by	1 bit.
	☐ Multiplic	ation overflow detection o	depends on S	SMUL.		
	<del></del>	☐ The 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACy.				
		Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.				
		v detection depends on ion accumulator overflow				ected, the
	☐ When a SATD.					
	Compatibil	ity with C54x devices (C	54CM = 1)			
	When this in	struction is executed with	M40 = 0, cc	mpatik	oility is er	sured.
Status Bits	Affected by	FRCT, M40, RDM, SA	ATD, SMUL			
	Affects	ACOVy				
Repeat	This instruct	ion can be repeated.				

### **Example**

Syntax		Descriptio	n			
AC1 = rn	d(AC1 - (AC0 * T1))		The product of the content of AC0 and the content of T1 is subtracted from the content of AC1. The result is rounded and stored in AC1.			
Before		After				
AC0	00 EC00 0000	AC0	00 EC00 0000			
AC1	00 3400 0000	AC1	00 1680 0000			
T1	2000	T1	2000			

0

0

0

M40

ACOV1

FRCT

0

0

M40

ACOV1

FRCT

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACx = rnd(ACx - (Smem * coef(Cmem)))[, T3 = Smem]	No	3	1	Х

#### **Opcode**

| 1101 0001 | AAAA AAAI | U%DD 10mm

#### **Operands**

ACx, Cmem, Smem

#### Description

This instruction performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are the content of a memory location (Smem), sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and sign extended to 17 bits.

- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACx.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVx) is set.
- ☐ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to store the 16-bit data memory operand Smem in temporary register T3.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Affected by FRCT, M40, RDM, SATD, SMUL **Status Bits** 

> Affects ACOVx

Repeat This instruction can be repeated.

Syntax	Description
AC2 = rnd(AC2 - (*AR1 * coef(*CDP)))	The product of the content addressed by AR1 and the content addressed by the coefficient data pointer register (CDP) is subtracted
	from the content of AC2. The result is rounded and stored in AC2.

Before		After	
AC2	00 EC00 0000	AC2	00 EC01 0000
AR1	0302	AR2	0302
CDP	0202	CDP	0202
302	FE00	302	FE00
202	0040	202	0040
ACOV2	0	ACOV2	1
SATD	0	SATD	0
RDM	0	RDM	0
FRCT	0	FRCT	0

# **Syntax Characteristics**

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ACy = rnd(ACy	– <b>(</b> Sn	nem * ACx)	)[, T3 = Smem]		No	3	1	Х
Opcod	le				1101	0010 AA	AA AA	AI U%I	DD 01SS
Opera	nds	AC	x, ACy, Sn	nem					
Descri	ption	The	This instruction performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are ACx(32–16) and the content of a memory location (Smem), sign extended to 17 bits.						
			If FRCT =	1, the output	of the mul	tiplier is shift	ed to t	he left by	1 bit.
			Multiplica	tion overflow d	letection d	epends on S	SMUL.		
The 32-bit result of the multiplication subtracted from the source accumulat					_	exten	ded to 4	0 bits and	
	Rounding is performed according to RDM, if the option applied to the instruction.					otional rnd keyword is			
			Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.						
			When an SATD.	overflow is det	ected, the	accumulato	r is sat	urated a	ccording to
				on provides the porary register	•	store the 16	-bit da	ta memo	ry operand
Compatibility with C54x devices (C54CM = 1)									
		When this instruction is executed with M40 = 0, compatibility is ensure						sured.	
Status	Bits	Affected by FRCT, M40, RDM, SATD, SMUL							
		Affe	ects	ACOVy					
Repea	t	This instruction can be repeated.							
Examp	ole								
Syntax	K		Descrip	tion					

# AC0 = AC0 - (\*AR3 \* AC1)

SWPU068E

The product of the content addressed by AR3 and the content of AC1 is subtracted from the content of AC0. The result is stored in AC0.

# **Syntax Characteristics**

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline	
[4]	ACy = rnd(ACx	– <b>(</b> T	x * Smem <b>))</b> [,	T3 = Smem]		No	3	1	Х	
Opcod	e				1101	0101 AA	AA AA	AI U%I	DD ssSS	
Operar	nds	AC	x, ACy, Sm	em, Tx						
Description			This instruction performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are the content of Tx, sign extended to 17 bits, and the content of a memory location (Smem), sign extended to 17 bits.							
			If FRCT =	1, the output of	the mul	Itiplier is shift	ted to t	he left by	1 bit.	
			Multiplicati	on overflow det	rflow detection depends on SMUL.					
		☐ The 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACx.								
			☐ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.							
			<ul> <li>Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.</li> </ul>							
			☐ When an overflow is detected, the accumulator is saturated according SATD.					ccording to		
	This instruction provides the option to store the 1 Smem in temporary register T3.					store the 16	-bit da	ta memo	ry operand	
		Co	mpatibility	with C54x dev	rices (C	54CM = 1)				
		Wł	nen this insti	ruction is execu	ted with	M40 = 0, co	mpatik	oility is er	sured.	
Status	Bits	Aff	ected by	FRCT, M40, F	RDM, SA	ATD, SMUL				
		Aff	ects	ACOVy						
Repeat	i	Th	is instructior	n can be repeate	ed.					

Syntax	Description
AC0 = AC1 - (T0 * *AR3)	The product of the content addressed by AR3 and the content of T0 is subtracted from the content of AC1. The result is stored in AC0.

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	ACy = M40(rnd(ACx - (uns(Xmem) * uns(Ymem)))) [, T3 = Xmem]	No	4	1	Х

### Opcode

| 1000 0110 | XXXM MMYY | YMMM SSDD | 011g uuU%

### **Operands**

ACx, ACy, Xmem, Ymem

#### Description

This instruction performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Xmem, extended to 17 bits, and the content of data memory operand Ymem, extended to 17 bits.

- ☐ Input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- ☐ The 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACx.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- ☐ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

This instruction provides the option to store the 16-bit data memory operand Xmem in temporary register T3.

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Affected by FRCT, M40, RDM, SATD, SMUL, SXMD **Status Bits** 

> ACOVy Affects

Repeat This instruction can be repeated.

Syntax	Description
AC3 = AC3 - (uns(*AR2+) * uns(*AR3+))	The product of the unsigned content addressed by AR2 and the
	unsigned content addressed by AR3 is subtracted from the content
	of AC3. The result is stored in AC3. AR2 and AR3 are both incremented by 1.

Before		After	
AC3	00 2300 EC00	AC3 FF B3E0 EC	00
AR2	302	AR2 3	03
AR3	202	AR3 2	03
ACOV3	0	ACOV3	0
302	FE00	302 FE	00
202	7000	202 70	00
FRCT	0	FRCT	0

#### Syntax Characteristics

No.	No. Syntax		Size	Cycles	Pipeline
[6]	ACx = rnd(ACx - (Smem * uns(coef(Cmem))))	No	3	1	Χ

#### **Opcode**

| 1101 0000 | AAAA AAAI | 0%DD 11mm

#### **Operands**

ACx, Cmem, Smem

#### **Description**

This instruction performs a multiplication and a subtraction in the D-unit MAC1. The input operands of the multiplier are the content of a data memory location (Smem) and the content of a data memory operand (Cmem).

#### Note:

The uns keyword is mandatory for this instruction.

The data memory operand Smem is addressed by DAGEN path X by using the Smem addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC1. The another data memory operand Cmem is addressed by DAGEN path C by using the coefficient addressing mode, driven on data bus BDB, and sign extended to 17 bits with filling zeros in the MAC1.

- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- ☐ The 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACx.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ☐ When an overflow is detected, the accumulator is saturated according to SATD.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

This instruction can be applied to compute the intermediate multiplication result and subtraction from the other partial result of double precision arithmetic, and to free up one DAGEN operator (DAGEN path Y) for storing an instruction with enabling parallelism.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVx

**Repeat** This instruction can be repeated.

### **Example**

Syntax	Description
	The product of the content addressed by AR3 and the unsigned content addressed by the coefficient data pointer register (CDP) is subtracted from the content of AC0. The result is stored in AC0. AR3 is decremented by 1 and CDP is incremented by 1.

#### Execution

rnd(ACx+(Smem)[16:0]\*uns(Cmem)[16:0]) -> ACx

Before				After			
AC0	00	0000	8000	AC0	00	0100	8000
XAR3		00	1001	XAR3		00	1000
Data memory							
1001h			FE00	1001h			FE00
XCDP		00	2000	XCDP		00	2001
Coeff memor	У						
2000h			8000	2000h			8000

### MASM::MOV

#### Multiply and Subtract with Parallel Load Accumulator from Memory

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = rnd(ACx - (Tx * Xmem)), ACy = Ymem << #16 [,T3 = Xmem]	No	4	1	Х

#### Opcode

1000 0110 XXXM MMYY YMMM DDDD 100x ssU%

### **Operands**

ACx, ACy, Tx, Xmem, Ymem

#### Description

This instruction performs two operations in parallel: multiply and subtract (MAS), and load.

The first operation performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are the content of Tx, sign extended to 17 bits, and the content of data memory operand Xmem, sign extended to 17 bits.

- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- ☐ Multiplication overflow detection depends on SMUL.
- ☐ The 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACx.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVx) is set.
- ☐ When an overflow is detected, the accumulator is saturated according to SATD.
- This instruction provides the option to store the 16-bit data memory operand Xmem in temporary register T3.

The second operation loads the content of data memory operand Ymem, which has been shifted to the left by 16 bits, into accumulator ACy.

- ☐ The input operand is sign extended to 40 bits according to SXMD.
- ☐ The shift operation is equivalent to the signed shift instruction.
- ☐ The input operand is shifted to the left by 16 bits according to M40.

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

**Status Bits** Affected by FRCT, M40, RDM, SATD, SMUL, SXMD ACOVx, ACOVy Affects Repeat This instruction can be repeated. See Also See the following other related instructions: ☐ Modify Auxiliary Register Content with Parallel Multiply and Subtract Multiply and Accumulate with Parallel Load Accumulator from Memory Multiply and Subtract Multiply and Subtract with Parallel Multiply Multiply and Subtract with Parallel Multiply and Accumulate ☐ Multiply and Subtract with Parallel Store Accumulator Content to Memory Parallel Multiply and Subtracts

Syntax	Description
AC0 = AC0 - (T0 * *AR3), AC1 = *AR4 << #16	Both instructions are performed in parallel. The product of the content addressed by AR3 and the content of T0 is subtracted from the content of AC0. The result is stored in AC0. The content addressed by AR4, which has been shifted to the left by 16 bits, is stored in AC1.

Parallel

# MAS::MPY

# Multiply and Subtract with Parallel Multiply

## **Syntax Characteristics**

No.	Syntax		Enable Bit	Size	Cycles	Pipeline				
[1]	ACx = M40(rnd(A ACy = M40(rnd(u)ACy))			uns(coef(Cmem))))), ef(Cmem))))	No	4	1	X		
[2]	ACy = M40(rnd(ACy - (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(uns(Smem) * uns(LO(coef(Cmem)))))				No	4	1	X		
[3]	ACy = M40(rnd(ACy - (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(uns(LO(Lmem)) * uns(LO(coef(Cmem)))))				No	4	1	X		
Descri	ption	These instructions perform two parallel operations in one cycle: multiply as subtract (MAS) and multiply. The operations are executed in the two D-us MACs.								
Status Bits		Affected by FRCT, M40, RDM, S			ATD, SMUL, SXMD					
		Affe	ects A	ACOVx, ACOVy						
See Al	so	See the following other related instructions:								
		☐ Modify Auxiliary Register Content with Parallel Multiply and Subtract								
		☐ Multiply and Accumulate with Parallel Multiply								
		☐ Multiply and Subtract								
		☐ Multiply and Subtract with Parallel Load Accumulator from Memory								
		☐ Multiply and Subtract with Parallel Multiply and Accumulate								
		☐ Multiply and Subtract with Parallel Store Accumulator Content to Memory								
		☐ Parallel Multiply and Subtracts								

### Multiply and Subtract With Parallel Multiply

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = M40(rnd(ACx - (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd(uns(Ymem) * uns(coef(Cmem))))	No	4	1	Х

#### Opcode

1000 0010 XXXM MMYY YMMM 10mm uuDD DDg%

#### **Operands**

ACx, ACy, Cmem, Xmem, Ymem

#### Description

This instruction performs two parallel operations in one cycle: multiply and subtract (MAS) and multiply. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Xmem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

The second operation performs a multiplication in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Ymem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

- ☐ Input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACx.
- For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.

- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ☐ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Each data flow can also disable the usage of the corresponding MAC unit, while allowing the modification of auxiliary registers in the three address generation units through the following instructions:

- mar(Xmem)
- mar(Ymem)
- mar(Cmem)

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVx, ACOVy

**Repeat** This instruction can be repeated.

Syntax	Description
AC0 = AC0 - (uns(*AR3) * uns(coef(*CDP))), AC1 = uns(*AR4) * uns(coef(*CDP))	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the coefficient data pointer register (CDP) is subtracted from the content of AC0. The result is stored in AC0. The product of the unsigned content addressed by AR4 and the unsigned content addressed by CDP is stored in AC1.

### Multiply and Subtract With Parallel Multiply

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACy = M40(rnd(ACy - (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(uns(Smem) * uns(LO(coef(Cmem)))))	No	4	1	Х

#### **Opcode**

| 1111 | 1101 | AAAA | AAAI | 0001 | 00mm | DDDD | uug%

#### **Operands**

ACx, ACy, Cmem, Smem

#### Description

This instruction performs two parallel operations in one cycle: multiply and subtract (MAS) and multiply. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and a subtraction in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand HI(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC2 (this data is shared to MAC1 and MAC2). The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA (effective address); the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand LO(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

The content of the memory location is zero extended to 17 bits, if the
optional uns keyword is applied to the input operand.
If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.

Multiplication overflow detection depends on SMUL.

L	extended to 40 bits and subtracted from the source accumulator ACy.
	For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits.
	Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
	Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
	When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

### Compatibility with C54x devices (C54CM = 1)

None.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVx, ACOVy

**Repeat** This instruction can be repeated.

#### **Example**

Syntax	Description
AC1 = AC1 - (uns(*AR3-) * uns(HI(coef(*CDP+)))), AC0 = uns(*AR3-) * uns(LO(coef(*CDP+)))	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is subtracted from the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the lower part of CDP is stored in AC0. AR3 is decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

#### Execution

ACy-M40(rnd(uns(Smem) [16:0] \*uns(HI(coef(Cmem))) [16:0])) -> ACy
M40(rnd(uns(Smem) [16:0] \*uns(LO(coef(Cmem))) [16:0])) -> ACx

# MAS::MPY Multiply and Subtract With Parallel Multiply

Before				After			
AC0	FF 8	000	0000	AC0	00	3F80	0000
XAR3		00	10FF	XAR3		00	10FE
Data memory							
10FFh			FE00	10FFh			FE00
XCDP		00	2000	XCDP		00	2002
Coeff memor	У						
2001h			4000	2001h			4000
AC1	00 0	000	8000	AC1	FF	8100	8000
Coeff memor	У						
2000h			8000	2000h			8000

5-386 Instruction Set Descriptions SWPU068E

### Multiply and Subtract With Parallel Multiply

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ACy = M40(rnd(ACy - (uns(HI(Lmem)) * uns(HI(coef(Cmem))))), ACx = M40(rnd(uns(LO(Lmem)) * uns(LO(coef(Cmem)))))	No	4	1	Х

#### **Opcode**

| 1111 | 1101 | AAAA | AAAI | 0101 | 00mm | DDDD | uug%

#### **Operands**

ACx, ACy, Cmem, Lmem

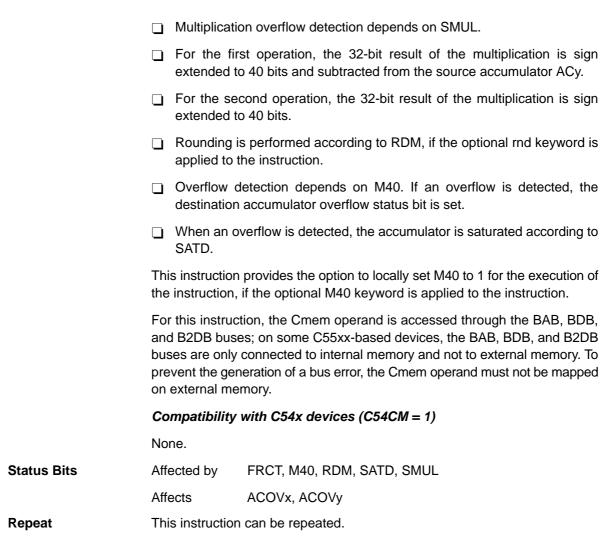
#### Description

This instruction performs two parallel operations in one cycle: multiply and subtract (MAS) and multiply. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and a subtraction in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand HI(Lmem) and the content of data memory operand HI(coef(Cmem)). The data memory operand HI(Lmem) is addressed by DAGEN path X with the EA (effective address); the data, which can be assumed to be the higher part of long word memory data, is driven on data bus CDB and sign extended to 17 bits in the MAC2. The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand LO(Lmem) and the content of data memory operand LO(coef(Cmem)). The data memory operand LO(Lmem) is addressed by DAGEN path X with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word memory data, is driven on data bus DDB and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.
- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.



Syntax	Description
AC1 = AC1 - (uns(HI(*AR3-)) * uns(HI(coef(*CDP+)))), AC0 = uns(LO(*AR3-)) * uns(LO(coef(*CDP+)))	Both instructions are performed in parallel. The product of the unsigned content addressed by the higher part of AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is subtracted from the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by the lower part of AR3 and the unsigned content addressed by the lower part of CDP is stored in AC0. When AR3– is used with HI/LO, AR3 is decremented by 2. When CDP+ is used with HI/LO, CDP is incremented by 2.

#### Execution

ACy-M40(rnd(uns(HI(Lmem))[16:0]\*uns(HI(coef(Cmem)))[16:0])) -> ACy  $\label{eq:M40(rnd(uns(LO(Lmem))[16:0]*uns(LO(coef(Cmem)))[16:0])) -> ACx }$ 

Before				After			
AC0	FF	8000	0000	AC0	00	3F80	0000
XAR3		00	10FE	XAR3		00	10FC
Data memory							
10FFh			FE00	10FFh			FE00
XCDP		00	2000	XCDP		00	2002
Coeff memory	У						
2001h			4000	2001h			4000
AC1	00	0000	8000	AC1	FF	8080	8000
Data memory							
10FEh			FF00	10FEh			FF00
Coeff memory	У						
2000h			8000	2000h			8000

### MAS::MAC

### Multiply and Subtract with Parallel Multiply and Accumulate

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	$ \begin{aligned} &ACx = M40(rnd(ACx - \textbf{(uns}(Xmem) * uns(\textbf{coef}(Cmem))))), \\ &ACy = M40(rnd(ACy + \textbf{(uns}(Ymem) * uns(\textbf{coef}(Cmem))))) \end{aligned} $	No	4	1	Х
[2]	$ACx = \frac{M40(rnd(ACx - (uns(Xmem) * uns(coef(Cmem)))))}{ACy = \frac{M40(rnd((ACy >> #16) + (uns(Ymem) * uns(coef(Cmem)))))}{ACy = \frac{M40(rnd(ACy >> #16) + (uns(Ymem) * uns(coef(Cmem))))}{ACy = \frac{M40(rnd(ACy - (uns(Xmem) * uns(coef(Cmem)))))}{ACy = \frac{M40(rnd(ACy - (uns(Xmem) * uns(coef(Cmem))))}{ACy = \frac{M40(rnd(ACy - (uns(Xmem) * uns(coef(Cmem)))}{ACy = \frac{M40(rnd(ACy - (uns(Cmem) * uns(coef(Cmem)))}{ACy = M40(rnd(ACy - (uns($	No	4	1	X
[3]	ACy = M40(rnd(ACy - (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx + (uns(Smem) * uns(LO(coef(Cmem))))))	No	4	1	X
[4]	ACy = M40(rnd(ACy - (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx + (uns(LO(Lmem)) * uns(LO(coef(Cmem)))))))	No	4	1	Х

These instructions perform two parallel operations in one cycle: multiply and subtract (MAS) and multiply and accumulate (MAC). The operations are executed in the two D-unit MACs. **Status Bits** Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

> Affects ACOVx, ACOVy

See Also See the following other related instructions:

- ☐ Modify Auxiliary Register Content with Parallel Multiply and Subtract
- Multiply and Subtract
- Multiply and Subtract with Parallel Load Accumulator from Memory
- Multiply and Subtract with Parallel Multiply
- Multiply and Subtract with Parallel Store Accumulator Content to Memory
- Parallel Multiply and Subtracts

### Multiply and Subtract with Parallel Multiply and Accumulate

#### Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	$ ACx = \frac{M40(rnd(ACx - (uns(Xmem) * uns(coef(Cmem)))))}{ACy = \frac{M40(rnd(ACy + (uns(Ymem) * uns(coef(Cmem)))))}{ACy = \frac{M40(rnd(ACy + (uns(Ymem) * uns(coef(Cmem)))))}{ACy = \frac{M40(rnd(ACx - (uns(Xmem) * uns(coef(Cmem)))))}{ACy = \frac{M40(rnd(ACy + (uns(Xmem) * uns(coef(Cmem))))}{ACy = \frac{M40(rnd(ACy + (uns(Xmem) * uns(coef(Cmem))))}{ACy = \frac{M40(rnd(ACx + (uns(Xmem) * uns(coef(Cmem)))}{ACy = \frac{M40(rnd(ACx + (uns(Xmem) * uns(coef(Cmem))))}{ACy = \frac{M40(rnd(ACx + (uns(Xmem) * uns(coef(Cmem)))}{ACy = \frac{M40(rnd(ACx + (uns(Xmem) * uns(coef(Cmem))))}{ACy = \frac{M40(rnd(ACx + (uns(Xmem) * uns(coef(Cmem)))}{ACy = M40(rnd(ACx + (uns(Xmem) * u$	No	4	1	Х

### Opcode

1000 0011 XXXM MMYY YMMM 01mm uuDD DDg%

### **Operands**

ACx, ACy, Cmem, Xmem, Ymem

#### Description

This instruction performs two parallel operations in one cycle: multiply and subtract (MAS) and multiply and accumulate (MAC). The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Xmem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

The second operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Ymem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

- ☐ Input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACx.
- For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.

- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ☐ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Each data flow can also disable the usage of the corresponding MAC unit, while allowing the modification of auxiliary registers in the three address generation units through the following instructions:

- mar(Xmem)
- mar(Ymem)
- mar(Cmem)

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVx, ACOVy

Repeat This instruction can be repeated.

Syntax	Description
AC0 = M40(rnd(AC0 - (uns(*AR0) * uns(coef(*CDP))))), AC1 = M40(rnd(AC1 + (uns(*AR1) * uns(coef(*CDP)))))	Both instructions are performed in parallel. The product of the unsigned content addressed by ARO and the unsigned content addressed by the coefficient data pointer register (CDP) is subtracted from the content of ACO. The result is rounded and stored in ACO. The product of the unsigned content addressed by AR1 and the unsigned content addressed by CDP is added to the content of AC1. The result is rounded and stored in AC1.

Before				After				
AC0	00	6900	0000	AC0	00	486B	0000	
AC1	00	0023	0000	AC1	00	95E3	0000	
*AR0			3400	*AR0			3400	
*AR1			EF00	*AR1			EF00	
*CDP			A067	*CDP			A067	
ACOV0			0	ACOV0			0	
ACOV1			0	ACOV1			0	
CARRY			0	CARRY			0	
FRCT			0	FRCT			0	

#### Multiply and Subtract with Parallel Multiply and Accumulate

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACx = M40(rnd(ACx - (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(coef(Cmem)))))	No	4	1	Х

### Opcode

1000 0100 XXXM MMYY YMMM 00mm uuDD DDg%

### **Operands**

ACx, ACy, Cmem, Xmem, Ymem

#### **Description**

This instruction performs two parallel operations in one cycle: multiply and subtract (MAS) and multiply and accumulate (MAC). The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Xmem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

The second operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Ymem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

- Input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACx.
- For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy, which has been shifted to the right by 16 bits. The shifting operation is performed with a sign extension of source accumulator ACy(39).

applied to the instruction.

Overflow detection depends on M40. If an overflow is detected, the

Rounding is performed according to RDM, if the optional rnd keyword is

destination accumulator overflow status bit is set.

When an overflow is detected, the accumulator is saturated according to

SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Each data flow can also disable the usage of the corresponding MAC unit, while allowing the modification of auxiliary registers in the three address generation units through the following instructions:

- mar(Xmem)
- mar(Ymem)
- mar(Cmem)

**Status Bits** 

Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVx, ACOVy

Repeat

This instruction can be repeated.

Syntax	Description
AC0 = AC0 - (uns(*AR3) * uns(coef(*CDP))), AC1 = (AC1 >> #16) + (uns(*AR4) * uns(coef(*CDP)))	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the coefficient data pointer register (CDP) is subtracted from the content of AC0. The result is stored in AC0. The product of the unsigned content addressed by AR4 and the unsigned content addressed by CDP is added to the content of AC1, which has been shifted to the right by 16 bits. The result is stored in AC1.

### Multiply and Subtract with Parallel Multiply and Accumulate

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ACy = M40(rnd(ACy - (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx + (uns(Smem) * uns(LO(coef(Cmem))))))	No	4	1	Х

**Opcode** 

| 1111 | 1101 | AAAA | AAAI | 0001 | 11mm | DDDD | uug%

**Operands** 

ACx, ACv, Cmem, Smem

Description

This instruction performs two parallel operations in one cycle: multiply and subtract (MAS) and multiply and accumulate (MAC). The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and a subtraction in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand HI(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC2 (this data is shared to MAC1 and MAC2). The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA (effective address); the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and an accumulation in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand LO(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.
- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.

		Multiplication	on overflow detection depends on SMUL.
			rst operation, the 32-bit result of the multiplication is sign to 40 bits and subtracted from the source accumulator ACy.
			econd operation, the 32-bit result of the multiplication is sign to 40 bits and added to the source accumulator ACx.
		•	is performed according to RDM, if the optional rnd keyword is the instruction.
			detection depends on M40. If an overflow is detected, the accumulator overflow status bit is set.
		When an ov SATD.	overflow is detected, the accumulator is saturated according to
			n provides the option to locally set M40 to 1 for the execution o , if the optional M40 keyword is applied to the instruction.
	and bus pre	B2DB buse es are only	ction, the Cmem operand is accessed through the BAB, BDB es; on some C55xx-based devices, the BAB, BDB, and B2DE connected to internal memory and not to external memory. To neration of a bus error, the Cmem operand must not be mapped emory.
	Co	mpatibility	with C54x devices (C54CM = 1)
	Nor	ne.	
Status Bits	Affe	ected by	FRCT, M40, RDM, SATD, SMUL
	Affe	ects	ACOVx, ACOVy

# Repeat **Example**

Syntax	Description
AC1 = AC1 - (uns(*AR3-) * uns(HI(coef(*CDP+)))), AC0 = AC0 + (uns(*AR3-) * uns(LO(coef(*CDP+))))	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is subtracted from the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the lower part of CDP is added to the content of AC0. The result is stored in AC0. AR3 is decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

This instruction can be repeated.

```
ACy-M40(rnd(uns(Smem)[16:0]*uns(HI(coef(Cmem)))[16:0])) -> ACy
```

Before				After			
AC0	00	0000	8000	AC0	00	3F80	8000
XAR3		00	10FF	XAR3		00	10FE
Data memory							
10FFh			FE00	10FFh			FE00
XCDP		00	2000	XCDP		00	2002
Coeff memory	У						
2001h			4000	2001h			4000
AC1	00	0000	8000	AC1	FF	8100	8000
Coeff memor	У						
2000h			8000	2000h			8000

### Multiply and Subtract with Parallel Multiply and Accumulate

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	ACy = M40(rnd(ACy - (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx + (uns(LO(Lmem)) * uns(LO(coef(Cmem)))))))	No	4	1	Х

#### Opcode

| 1111 | 1101 | AAAA | AAAI | 0101 | 11mm | DDDD | uug%

#### **Operands**

ACx, ACy, Cmem, Lmem

#### Description

This instruction performs two parallel operations in one cycle: multiply and subtract (MAS) and multiply and accumulate (MAC). The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and a subtraction in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand HI(Lmem) and the content of data memory operand HI(coef(Cmem)). The data memory operand HI(Lmem) is addressed by DAGEN path X with the EA (effective address); the data, which can be assumed to be the higher part of long word memory data, is driven on data bus CDB and sign extended to 17 bits in the MAC2. The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and an accumulation in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand LO(Lmem) and the content of data memory operand LO(coef(Cmem)). The data memory operand LO(Lmem) is addressed by DAGEN path X with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word memory data, is driven on data bus DDB and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

☐ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.

	If FRCT = 1	, the output of the multiplier is shifted to the left by 1 bit.
	Multiplication	n overflow detection depends on SMUL.
		st operation, the 32-bit result of the multiplication is sign 40 bits and subtracted from the source accumulator ACy.
		cond operation, the 32-bit result of the multiplication is sign 40 bits and added to the source accumulator ACx.
	_	s performed according to RDM, if the optional rnd keyword is ne instruction.
		etection depends on M40. If an overflow is detected, the accumulator overflow status bit is set.
	When an over SATD.	verflow is detected, the accumulator is saturated according to
		provides the option to locally set M40 to 1 for the execution of if the optional M40 keyword is applied to the instruction.
and bus pre	d B2DB buse ses are only o	ion, the Cmem operand is accessed through the BAB, BDB, s; on some C55xx-based devices, the BAB, BDB, and B2DB connected to internal memory and not to external memory. To eration of a bus error, the Cmem operand must not be mapped mory.
Co	mpatibility v	with C54x devices (C54CM = 1)
Noi	ne.	
Affe	ected by	FRCT, M40, RDM, SATD, SMUL
Affe	ects	ACOVx, ACOVy
Thi	s instruction	can be repeated.

# Repeat **Example**

**Status Bits** 

Syntax	Description
AC1 = AC1 - (uns(HI(*AR3-)) * uns(HI(coef(*CDP+)))), AC0 = AC0 + (uns(LO(*AR3-)) * uns(LO(coef(*CDP+))))	Both instructions are performed in parallel. The product of the unsigned content addressed by the higher part of AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is subtracted from the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by the lower part of AR3 and the unsigned content addressed by the lower part of CDP is added to the content of AC0. The result is stored in AC0. When AR3– is used with HI/LO, AR3 is decremented by 2. When CDP+ is used with HI/LO, CDP is incremented by 2.

#### Execution

ACy-M40(rnd(uns(HI(Lmem))[16:0]\*uns(HI(coef(Cmem)))[16:0])) -> ACy
ACx+M40(rnd(uns(LO(Lmem))[16:0]\*uns(LO(coef(Cmem)))[16:0])) -> ACx

Before				After			
AC0	00	0000	8000	AC0	00	3F80	8000
XAR3		00	10FE	XAR3		00	10FC
Data memory							
10FFh			FE00	10FFh			FE00
XCDP		00	2000	XCDP		00	2002
Coeff memor	У						
2001h			4000	2001h			4000
AC1	00	0000	8000	AC1	FF	8080	8000
Data memory							
10FEh			FF00	10FEh			FF00
Coeff memory	У						
2000h			8000	2000h			8000

5-400 Instruction Set Descriptions SWPU068E

# MASM:MOV

Multiply and Subtract with Parallel Store Accumulator Content to Memory

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(ACy - (Tx * Xmem)), Ymem = HI(ACx << T2) [,T3 = Xmem]	No	4	1	Х

# Opcode

1000 0111 XXXM MMYY YMMM SSDD 010x ssU%

#### **Operands**

ACx, ACy, Tx, Xmem, Ymem

### **Description**

This instruction performs two operations in parallel: multiply and subtract (MAS) and store.

The first operation performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are the content of Tx, sign extended to 17 bits, and the content of data memory operand Xmem, sign extended to 17 bits.

- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACy.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- ☐ When an overflow is detected, the accumulator is saturated according to SATD.
- ☐ This instruction provides the option to store the 16-bit data memory operand Xmem in temporary register T3.

The second operation shifts the accumulator ACx by the content of T2 and stores ACx(31-16) to data memory operand Ymem. If the 16-bit value in T2 is not within -32 to +31, the shift is saturated to -32 or +31 and the shift is performed with this value.

- ☐ The input operand is shifted in the D-unit shifter according to SXMD.
- After the shift, the high part of the accumulator, ACx(31–16), is stored to the memory location.

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When this instruction is executed with C54CM = 1, the 6 LSBs of T2 determine the shift quantity. The 6 LSBs of T2 define a shift quantity within -32 to +31. When the 16-bit value in T2 is between -32 to -17, a modulo 16 operation transforms the shift quantity to within -16 to -1.

☐ If the SST bit = 1 and the SXMD bit = 0, then the saturate and uns keywords are applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:

ACy = rnd(ACy - (Tx \* Xmem)), Ymem = HI(saturate(uns(ACx << T2))) [,T3 = Xmem]

☐ If the SST bit = 1 and the SXMD bit = 1, then only the saturate keyword is applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:

ACy = rnd(ACy - (Tx \* Xmem)), Ymem = HI(saturate(ACx << T2)) [,T3 = Xmem]

Status Bits Affected by C54CM, FRCT, M40, RDM, SATD, SMUL, SST, SXMD

Affects ACOVy

**Repeat** This instruction can be repeated.

**See Also** See the following other related instructions:

- ☐ Modify Auxiliary Register Content with Parallel Multiply and Subtract
- Multiply and Accumulate with Parallel Store Accumulator Content to Memory
- Multiply and Subtract
- ☐ Multiply and Subtract with Parallel Load Accumulator from Memory
- Multiply and Subtract with Parallel Multiply
- ☐ Multiply and Subtract with Parallel Multiply and Accumulate
- Parallel Multiply and Subtracts

Syntax	Description
AC0 = AC0 - (T0 * *AR3), *AR4 = HI(AC1 << T2)	Both instructions are performed in parallel. The product of the content addressed by AR3 and the content of T0 is subtracted from the content of AC0. The result is stored in AC0. The content of AC1 is shifted by the content of T2, and AC1(31–16) is stored at the address of AR4.

### NEG

### Negate Accumulator, Auxiliary, or Temporary Register Content

#### **Syntax Characteristics**

[1] dst = - src Yes 2 1 X	No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
	[1]	dst = -src	Yes	2	1	Х

### Opcode

0011 010E FSSS FDDD

#### **Operands**

dst, src

#### Description

This instruction computes the 2s complement of the content of the source register (src). This instruction clears the CARRY status bit to 0 for all nonzero values of src. If src equals 0, the CARRY status bit is set to 1.

- ☐ When the destination operand (dst) is an accumulator:
  - The operation is performed on 40 bits in the D-unit ALU.
  - Input operands are sign extended to 40 bits according to SXMD.
  - If an auxiliary or temporary register is the source operand (src) of the instruction, the 16 LSBs of the auxiliary or temporary register are sign extended according to SXMD.
  - Overflow detection and CARRY status bit depends on M40.
  - When an overflow is detected, the accumulator is saturated according to SATD.
- ☐ When the destination operand (dst) is an auxiliary or temporary register:
  - The operation is performed on 16 bits in the A-unit ALU.
  - If an accumulator is the source operand (src) of the instruction, the 16 LSBs of the accumulator are used to perform the operation.
  - Overflow detection is done at bit position 15.
  - When an overflow is detected, the destination register is saturated according to SATA.

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

**Status Bits** Affected by M40, SATA, SATD, SXMD

> Affects ACOVx, CARRY

Repeat This instruction can be repeated.

See Also See the following other related instructions:

☐ Complement Accumulator, Auxiliary, or Temporary Register Bit

☐ Complement Accumulator, Auxiliary, or Temporary Register Content

### **Example**

Syntax	Description
AC0 = -AC1	The 2s complement of the content of AC1 is stored in AC0.

5-404 Instruction Set Descriptions



### No Operation (nop)

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	nop	Yes	1	1	D
[2]	nop_16	Yes	2	1	D

0010 000E Opcode

**Operands** none

Instruction [1] increments the program counter register (PC) by 1 byte. Description

Instruction [2] increments the PC by 2 bytes.

**Status Bits** Affected by none

> Affects none

Repeat This instruction can be repeated.

Syntax	Description
nop	The program counter (PC) is incremented by 1 byte.

# **AMAR**

# Parallel Modify Auxiliary Register Contents

# **Syntax Characteristics**

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline			
[1]	mar(Xmem), m	ar(Ymem), mar(coef(Cmem))	No	4	1	Х			
Opcode	е	1000 0101   XXXI	MMYY YMM	ИМ 10	mm xxx	x xxxx			
Operan	nds	Cmem, Xmem, Ymem							
Description		This instruction performs three parallel modify auxiliary register (MAR) operations in one cycle. The auxiliary register modification is specified by:							
		☐ The content of data memory ope	erand Xmem						
		☐ The content of data memory ope	erand Ymem						
		☐ The content of a data memory coefficient addressing mode	operand Cm	em, ad	ddressed	using the			
Status	Bits	Affected by none							
		Affects none							
Repeat		This instruction can be repeated.							
See Als	50	See the following other related instru	uctions:						
		☐ Modify Auxiliary Register Conter	nt						
		☐ Modify Extended Auxiliary Regis	ster Content						

Syntax	Description
mar(*AR3+), mar(*AR4-), mar(coef(*CDP))	AR3 is incremented by 1. AR4 is decremented by 1. CDP is not modified.

# MPY::MPY

# Parallel Multiplies

# **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = M40(rnd(uns(Xmem) * uns(coef(Cmem)))), ACy = M40(rnd(uns(Ymem) * uns(coef(Cmem))))	No	4	1	Х
[2]	ACy = M40(rnd(uns(Smem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(uns(Smem) * uns(LO(coef(Cmem)))))	No	4	1	Х
[3]	$ \begin{aligned} &ACy = \textcolor{red}{M40(rnd(uns(HI(Lmem)) \ ^* \ uns(HI(coef(Cmem)))))}, \\ &ACx = \textcolor{red}{M40(rnd(uns(LO(Lmem)) \ ^* \ uns(LO(coef(Cmem)))))} \end{aligned} $	No	4	1	Х
[4]	ACy = M40(rnd(uns(Ymem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(uns(Xmem) * uns(LO(coef(Cmem)))))	No	5	1	Х

Description		ons perform two parallel multiply operations in one cycle. The executed in the two D-unit MACs.
Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
	Affects	ACOVx, ACOVy
See Also	See the followi	ng other related instructions:
	☐ Modify Aux	kiliary Register Content with Parallel Multiply
	☐ Multiply	
	☐ Multiply an	d Accumulate with Parallel Multiply
	☐ Multiply an	d Subtract with Parallel Multiply
	☐ Parallel Mu	ultiply and Accumulates
	☐ Parallel Mu	ultiply and Subtracts

# Parallel Multiplies

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = M40(rnd(uns(Xmem) * uns(coef(Cmem)))), ACy = M40(rnd(uns(Ymem) * uns(coef(Cmem))))	No	4	1	Х

#### Opcode

1000 0010 XXXM MMYY YMMM 00mm uuDD DDg%

### **Operands**

ACx, ACy, Cmem, Xmem, Ymem

#### **Description**

This instruction performs two parallel multiply operations in one cycle. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Xmem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

This second operation performs a multiplication in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Ymem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

- ☐ Input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- ☐ The 32-bit result of the multiplication is sign extended to 40 bits.
- □ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ☐ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Each data flow can also disable the usage of the corresponding MAC unit, while allowing the modification of auxiliary registers in the three address generation units through the following instructions:

- mar(Xmem)
- mar(Ymem)
- mar(Cmem)

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVx, ACOVy

**Repeat** This instruction can be repeated.

Syntax	Description
AC0 = uns(*AR3) * uns(coef(*CDP)), AC1 = uns(*AR4) * uns(coef(*CDP))	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the coefficient data pointer register (CDP) is stored in AC0. The product of the unsigned content addressed by AR4 and the unsigned content addressed by CDP is stored in AC1.

#### Parallel Multiplies

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACy = M40(rnd(uns(Smem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(uns(Smem) * uns(LO(coef(Cmem)))))	No	4	1	Х

#### Opcode

| 1111 | 1101 | AAAA | AAAI | 0000 | 00mm | DDDD | uug%

#### **Operands**

ACx, ACy, Cmem, Smem

#### Description

This instruction performs two parallel multiply operations in one cycle. The operations are executed in the D-unit MACs.

The first operation performs a multiplication in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand HI(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC2 (this data is shared to MAC1 and MAC2). The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA (effective address); the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand LO(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- ☐ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.
- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- ☐ The 32-bit result of the multiplication is sign extended to 40 bits.

Rounding is performed according to RDM, if the optional rnd keyword is
applied to the instruction.

- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ☐ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

# Compatibility with C54x devices (C54CM = 1)

None.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVx, ACOVy

**Repeat** This instruction can be repeated.

#### **Example**

Syntax	Description
AC1 = uns(*AR3-) * uns(HI(coef(*CDP+))), AC0 = uns(*AR3-) * uns(LO(coef(*CDP+)))	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is stored in AC1. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the lower part of CDP is stored in AC0. AR3 is decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

#### Execution

```
M40(rnd(uns(Smem) [16:0] *uns(HI(coef(Cmem))) [16:0])) -> ACy
M40(rnd(uns(Smem) [16:0] *uns(LO(coef(Cmem))) [16:0])) -> ACx
```

# MPY::MPY Parallel Multiplies

Before				After			
AC0	FF	8000	0000	AC0	00	3F80	0000
XAR3		00	10FF	XAR3		00	10FE
Data memory							
10FFh			FE00	10FFh			FE00
XCDP		00	2000	XCDP		00	2002
Coeff memor	У						
2001h			4000	2001h			4000
AC1	FF	8000	0000	AC1	00	7F00	0000
Coeff memor	У						
2000h			8000	2000h			8000

5-412 Instruction Set Descriptions SWPU068E

#### Parallel Multiplies

# **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	$ ACy = \frac{M40(rnd(uns(HI(Lmem)) * uns(HI(coef(Cmem)))))}{ACx = \frac{M40(rnd(uns(LO(Lmem)) * uns(LO(coef(Cmem)))))}{ACx} $	No	4	1	Х

#### Opcode

| 1111 | 1101 | AAAA | AAAI | 0100 | 00mm | DDDD | uug%

#### **Operands**

ACx, ACy, Cmem, Lmem

#### Description

This instruction performs two parallel multiply operations in one cycle. The operations are executed in the D-unit MACs.

The first operation performs a multiplication in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand HI(Lmem) and the content of data memory operand HI(coef(Cmem)). The data memory operand HI(Lmem) is addressed by DAGEN path X with the EA (effective address); the data, which can be assumed to be the higher part of long word memory data, is driven on data bus CDB and sign extended to 17 bits in the MAC2 (this data is shared to MAC1 and MAC2). The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand LO(Lmem) and the content of data memory operand LO(coef(Cmem)). The data memory operand LO(Lmem) is addressed by DAGEN path X with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word memory data, is driven on data bus DDB and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.
- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.

The 32-bit result of the multiplication is sign extended to 40 bits.
Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

# Compatibility with C54x devices (C54CM = 1)

None.

**Status Bits** Affected by FRCT, M40, RDM, SATD, SMUL

> Affects ACOVx, ACOVy

Repeat This instruction can be repeated.

# **Example**

Syntax	Description
AC1 = uns(HI(*AR3-)) * uns(HI(coef(*CDP+))), AC0 = uns(LO(*AR3-)) * uns(LO(coef(*CDP+)))	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is stored in AC1. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the lower part of CDP is stored in AC0. When AR3– is used with HI/LO, AR3 is decremented by 2. When CDP+ is used with HI/LO, CDP is incremented by 2.

#### Execution

```
M40(rnd(uns(HI(Lmem))[16:0]*uns(HI(coef(Cmem)))[16:0])) -> ACy
M40(rnd(uns(LO(Lmem))[16:0]*uns(LO(coef(Cmem)))[16:0])) -> ACx
```

Before				After			
AC0	FF	8000	0000	AC0	00	3F80	0000
XAR3		00	10FE	XAR3		00	10FC
Data memory							
10FFh			FE00	10FFh			FE00
XCDP		00	2000	XCDP		00	2002
Coeff memory	Y						
2001h			4000	2001h			4000
AC1	FF	8000	0000	AC1	00	7F80	0000
Data memory							
10FEh			FF00	10FEh			FF00
Coeff memory	Y						
2000h			8000	2000h			8000

# Parallel Multiplies

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	ACy = M40(rnd(uns(Ymem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(uns(Xmem) * uns(LO(coef(Cmem)))))	No	5 (*)	1	Х

(\*) 1 LSB is allocated to instruction slot #2.

#### Opcode

1001 0010 XXXM MMYY YMMM 00mm uuDD DDg%

### **Operands**

ACx, ACy, Cmem, Xmem, Ymem

# Description

This instruction performs two parallel multiply operations in one cycle. The operations are executed in the D-unit MACs.

The first operation performs a multiplication in the D-unit MAC2. The input operands of the multiplier are the contents of data memory operand Ymem, extended to 17 bits, and the content of data memory operand HI(coef(Cmem)) which is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication in the D-unit MAC1. The input operands of the multiplier are the contents of data memory operand Xmem, extended to 17 bits, and the content of data memory operand LO(coef(Cmem)) which is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- ☐ The input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- ☐ Multiplication overflow detection depends on SMUL.
- ☐ The 32-bit result of the multiplication is sign extended to 40 bits.

Rounding is performed according to RDM, if the optional rnd keyword is
applied to the instruction.

- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ☐ When an overflow is detected, the accumulator is saturated according to SATD.
- ☐ Because this instruction occupies both instruction slots #1 and #2, this can not be executed in parallel with other instructions.
- ☐ The Xmem operand can access the MMRs but the Ymem operand can not.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

### Compatibility with C54x devices (C54CM = 1)

None.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVx, ACOVy

**Repeat** This instruction can be repeated.

### **Example**

Syntax	Description
AC1 = uns(*AR3-) * uns(HI(coef(*CDP+))), AC0 = uns(*AR2-) * uns(LO(coef(*CDP+)))	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is stored in AC1. The product of the unsigned content addressed by AR2 and the unsigned content addressed by the lower part of the CDP is stored in AC0. AR3 and AR2 are decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

#### Execution

```
M40(rnd(uns(Xmem)[16:0] * uns(LO(coef(Cmem)))[16:0])) -> ACx
M40(rnd(uns(Ymem)[16:0] * uns(HI(coef(Cmem)))[16:0])) -> ACy
```

# MPY::MPY Parallel Multiplies

Before				After			
AC0	FF	8000	0000	AC0	00	3F80	0000
XAR2		00	10FE	XAR2		00	10FD
XAR3		00	20FE	XAR3		00	20FD
Data memory							
10FEh			FE00	10FEh			FE00
XCDP		00	2000	XCDP		00	2002
Coeff memory	У						
2001h			4000	2001h			4000
AC1	FF	8000	0000	AC1	00	7F80	0000
Data memory							
20FEh			FF00	20FFh			FF00
Coeff memory	У						
2000h			8000	2000h			8000

5-418 Instruction Set Descriptions SWPU068E



# **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = M40(rnd(ACx + (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd(ACy + (uns(Ymem) * uns(coef(Cmem)))))	No	4	1	Х
[2]	ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(coef(Cmem))))), ACy = M4(rnd(ACy + (uns(Ymem) * uns(coef(Cmem)))))	No	4	1	X
[3]	ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(coef(Cmem)))))	No	4	1	Х
[4]	$ \begin{aligned} &ACy = \frac{M40(rnd(ACy + \mathbf{(uns(Smem) * uns(Hl(coef(Cmem)))))})}{ACx = \frac{M40(rnd(ACx + \mathbf{(uns(Smem) * uns(LO(coef(Cmem)))))})}{MCx = \frac{M40(NCx + \mathbf{(uns(Smem) * uns(LO(coef(Cmem))))})}{MCx = \frac{M40(MCx + \mathbf{(uns(Smem) * uns(LO(coef(Cmem))))})}{MCx = \frac{M40(MCx + \mathbf{(uns(Smem) * uns(LO(coef(Cmem)))})}{MCx = M40(MCx + \mathbf{(uns(Smem) * uns(LO(coef(Cmem))$	No	4	1	X
[5]	$ ACy = \frac{M40(rnd(ACy + (uns(Smem) * uns(HI(coef(Cmem))))))}{ACx = \frac{M40(rnd((ACx >> #16) + (uns(Smem) * uns(LO(coef(Cmem))))))}{} $	No	4	1	X
[6]	ACy = M40(rnd((ACy >> #16) + (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd((ACx >> #16) + (uns(Smem) * uns(LO(coef(Cmem))))))	No	4	1	X
[7]	ACy = M40(rnd(ACy + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx + (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))	No	4	1	X
[8]	ACy = M40(rnd(ACy + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd((ACx >> #16) + (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))	No	4	1	Х
[9]	ACy = M40(rnd((ACy >> #16) + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd((ACx >> #16) + (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))	No	4	1	Х
[10]	$ ACy = \frac{M40(rnd(ACy + uns(Ymem) * uns(HI(coef(Cmem)))))}{ACx = \frac{M40(rnd(ACx + uns(Xmem) * uns(LO(coef(Cmem)))))}{ACx = \frac{M40(rnd(ACx + uns(Xmem) * uns(LO(coef(Cmem))))}{ACx = \frac{M40(rnd(ACx + uns(LO(coef(Cmem))))}{ACx = \frac{M40(rnd(ACx + uns(LO(coef(Cmem)))}{ACx = \frac{M40(rnd(ACx + uns(LO(coef(Cmem))))}{ACx = \frac{M40(rnd(ACx + uns(LO(coef(Cmem))))}{ACx = \frac{M40(rnd(ACx + uns(LO(coef(Cmem))))}{ACx = \frac{M40(rnd(ACx + uns(LO(coef(Cmem)))}{ACx = \frac{M40(rnd(ACx + uns(LO(coef(Cmem)))}{ACx = \frac{M40(rnd(ACx + uns(LO(coef(Cmem))))}{ACx = \frac{M40(rnd(ACx + uns(LO(coef(Cmem))))}{ACx = \frac{M40(rnd(ACx + uns(LO(coef(Cmem)))}{ACx = \frac{M40(rnd(ACx + uns(LO(coef(Cmem)))}{ACx = \frac{M40(rnd(ACx + uns(LO(coef(Cmem)))}{ACx = \frac{M40(rnd(ACx + uns(LO(coef(Cmem)))}{$	No	5	1	Х

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[11]	ACy = M40(rnd(AC) ACx = M40(rnd((AC)) uns(LO(coef(C))	Cx >:	> #16) + (uns(	uns(HI(coef(Cmem)))))) (Xmem) *	, No	5	1	Х
[12]	uns(HI(coef(Cmen ACx = M40(rnd((A	Cy = M40(rnd((ACy >> #16) + (uns(Ymem) * ns(HI(coef(Cmem)))))), Cx = M40(rnd((ACx >> #16) + (uns(Xmem) * ns(LO(coef(Cmem))))))					1	X
<b>Description</b> These instructions perform two paral operations in one cycle. The operations							. ,	
Statu	s Bits	Aff	ected by	FRCT, M40, RDM, SA	ATD, SMUL, S	XMD		
		Aff	ects	ACOVx, ACOVy				
See A	Also	Se	See the following other related instructions:					
			Modify Aux	iliary Register Content	with Parallel	Multipl	ly and Ad	cumulate
			Multiply and	d Accumulate				
			Multiply and	d Accumulate with Par	allel Delay			
			Multiply and Accumulate with Parallel Load Accumulator from Memory					Memory
			Multiply and	d Accumulate with Par	allel Multiply			
			Multiply and	Accumulate with Parall	el Store Accun	nulator	Content	to Memory
	Multiply and Subtract with Paralle			l Multiply and	Accun	nulate		
			Multiply wit	h Parallel Multiply and	Accumulate			
			Parallel Mu	Itiplies				

Parallel Multiply and Subtracts

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = M40(rnd(ACx + (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd(ACy + (uns(Ymem) * uns(coef(Cmem)))))	No	4	1	Х

### Opcode

1000 0011 XXXM MMYY YMMM 00mm uuDD DDg%

# Operands

ACx, ACy, Cmem, Xmem, Ymem

**Description**This instruction performs two parallel multiply and accumulate (MAC) operations in one cycle. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Xmem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

The second operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Ymem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

- ☐ Input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- ☐ The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Each data flow can also disable the usage of the corresponding MAC unit, while allowing the modification of auxiliary registers in the three address generation units through the following instructions:

- mar(Xmem)
- mar(Ymem)
- mar(Cmem)

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVx, ACOVy

**Repeat** This instruction can be repeated.

Syntax	Description
AC0 = AC0 + (uns(*AR3) * uns(coef(*CDP))), AC1 = AC1 + (uns(*AR4) * uns(coef(*CDP)))	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the coefficient data pointer register (CDP) is added to the content of AC0. The result is stored in AC0. The product of the unsigned content addressed by AR4 and the unsigned content addressed by CDP is added to the content of AC1. The result is stored in AC1.

#### Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(coef(Cmem))))), ACy = M4(rnd(ACy + (uns(Ymem) * uns(coef(Cmem)))))	No	4	1	Х

#### **Opcode**

1000 0011 XXXM MMYY YMMM 10mm uuDD DDg%

### **Operands**

ACx, ACy, Cmem, Xmem, Ymem

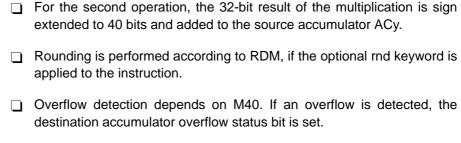
# **Description**

This instruction performs two parallel multiply and accumulate (MAC) operations in one cycle. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Xmem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

The second operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Ymem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

- ☐ Input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx, which has been shifted to the right by 16 bits. The shifting operation is performed with a sign extension of source accumulator ACx(39).



☐ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Each data flow can also disable the usage of the corresponding MAC unit, while allowing the modification of auxiliary registers in the three address generation units through the following instructions:

- mar(Xmem)
- mar(Ymem)
- mar(Cmem)

Status Bits Affected by

FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVx, ACOVy

Repeat

This instruction can be repeated.

Syntax	Description
AC0 = (AC0 >> #16) + (uns(*AR3) * uns(coef(*CDP))), AC1 = AC1 + (uns(*AR4) * uns(coef(*CDP)))	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the coefficient data pointer register (CDP) is added to the content of AC0, which has been shifted to the right by 16 bits. The result is stored in AC0. The product of the unsigned content addressed by AR4 and the unsigned content addressed by CDP is added to the content of AC1. The result is stored in AC1.

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(coef(Cmem)))))	No	4	1	Х

#### **Opcode**

1000 0100 XXXM MMYY YMMM 11mm uuDD DDg%

#### **Operands**

ACx, ACy, Cmem, Xmem, Ymem

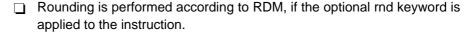
### **Description**

This instruction performs two parallel multiply and accumulate (MAC) operations in one cycle. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Xmem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

The second operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Ymem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

- Input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- ☐ The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator, which has been shifted to the right by 16 bits. The shifting operation is performed with a sign extension of source accumulator bit 39.



- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ☐ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Each data flow can also disable the usage of the corresponding MAC unit, while allowing the modification of auxiliary registers in the three address generation units through the following instructions:

- mar(Xmem)
- mar(Ymem)
- mar(Cmem)

**Status Bits** 

Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVx, ACOVy

Repeat

This instruction can be repeated.

Syntax	Description
AC0 = (AC0 >> #16) + (uns(*AR3) * uns(coef(*CDP))), AC1 = (AC1 >> #16) + (uns(*AR4) * uns(coef(*CDP)))	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the coefficient data pointer register (CDP) is added to the content of AC0, which has been shifted to the right by 16 bits. The result is stored in AC0. The product of the unsigned content addressed by AR4 and the unsigned content addressed by CDP is added to the content of AC1, which has been shifted to the right by 16 bits. The result is stored in AC1.

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	ACy = M40(rnd(ACy + (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx + (uns(Smem) * uns(LO(coef(Cmem)))))))	No	4	1	Х

#### **Opcode**

| 1111 | 1101 | AAAA | AAAI | 0001 | 01mm | DDDD | uug%

#### **Operands**

ACx, ACy, Cmem, Smem

#### Description

This instruction performs two parallel multiply and accumulate (MAC) operations in one cycle. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand HI(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC2 (this data is shared to MAC1 and MAC2). The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA (effective address); the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and an accumulation in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand LO(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

The content of the memory location is zero extended to 17 bits, if the
optional uns keyword is applied to the input operand.

- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.

to the source accumulator.
Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

# Compatibility with C54x devices (C54CM = 1)

None.

**Status Bits** Affected by FRCT, M40, RDM, SATD, SMUL

> ACOVx, ACOVy Affects

Repeat This instruction can be repeated.

### **Example**

Syntax	Description
AC1 = AC1 + (uns(*AR3-) * uns(HI(coef(*CDP+)))), AC0 = AC0 + (uns(*AR3-) * uns(LO(coef(*CDP+))))	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the lower part of CDP is added to the content of AC0. The result is stored in AC0. AR3 is decremented by 1 and CDP. When CDP+ is used with HI/LO, CDP is incremented by 2.

#### Execution

ACy+M40 (rnd (uns (Smem) [16:0] \*uns (HI (coef (Cmem))) [16:0])) -> ACy ACx+M40(rnd(uns(Smem)[16:0]\*uns(LO(coef(Cmem)))[16:0])) -> ACx

Before				After			
AC0	00	0000	8000	AC0	00	3F80	8000
XAR3		00	10FF	XAR3		00	10FE
Data memory							
10FFh			FE00	10FFh			FE00
XCDP		00	2000	XCDP		00	2002
Coeff memory	Y						
2001h			4000	2001h			4000
AC1	00	0000	8000	AC1	00	7F00	8000
Coeff memory	Y						
2000h			8000	2000h			8000

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	ACy = M40(rnd(ACy + (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd((ACx >> #16) + (uns(Smem) * uns(LO(coef(Cmem))))))	No	4	1	Х

### Opcode

| 1111 | 1101 | AAAA | AAAI | 0010 | 00mm | DDDD | uug%

#### **Operands**

ACx, ACy, Cmem, Smem

# Description

This instruction performs two parallel multiply and accumulate (MAC) operations in one cycle. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand HI(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC2 (this data is shared to MAC1 and MAC2). The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA (effective address); the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and an accumulation in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand LO(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

The content of the memory location is zero extended to 17 bits, if the
optional uns keyword is applied to the input operand.

- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.

extended to 40 bits and added to the source accumulator ACy.
For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx, which has been shifted to the right by 16 bits. The shifting operation is performed with a sign extension of source accumulator ACx(39).
Rounding is performed according to RDM, if the optional rnd keyword is

- applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ☐ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

#### Compatibility with C54x devices (C54CM = 1)

None.

**Status Bits** Affected by FRCT, M40, RDM, SATD, SMUL

> ACOVx, ACOVy Affects

Repeat This instruction can be repeated.

Syntax	Description
AC1 = AC1 + (uns(*AR3-) * uns(HI(coef(*CDP+)))),	Both instructions are performed in parallel.
AC0 = (AC0 >> #16) + (uns(*AR3-) * uns(LO(coef(*CDP+))))	The product of the unsigned content ad-
	dressed by AR3 and the unsigned content
	addressed by the higher part of the coefficient
	data pointer register (CDP) is added to the
	content of AC1. The result is stored in AC1.
	The product of the unsigned content ad-
	dressed by AR3 and the unsigned content
	addressed by the lower part of CDP is added
	to the content of AC0, which has been shifted
	to the right by 16 bits. The result is stored in
	AC0. AR3 is decremented by 1. When CDP+
	is used with HI/LO, CDP is incremented by 2.

# MAC::MAC Parallel Multiply and Accumulates

### Execution

ACy+M40(rnd(uns(Smem) [16:0] \*uns(HI(coef(Cmem))) [16:0])) -> ACy
(ACx>>#16)+M40(rnd(uns(Smem) [16:0] \*uns(LO(coef(Cmem))) [16:0])) -> ACx

Before				After			
AC0	00	0800	0000	AC0	00	3F80	0800
XAR3		00	10FF	XAR3		00	10FE
Data memory							
10FFh			FE00	10FFh			FE00
XCDP		00	2000	XCDP		00	2002
Coeff memor	У						
2001h			4000	2001h			4000
AC1	00	0000	8000	AC1	00	7F80	8000
Coeff memory							
2000h			8000	2000h			8000

5-432 Instruction Set Descriptions

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	ACy = M40(rnd((ACy >> #16) + (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd((ACx >> #16) + (uns(Smem) * uns(LO(coef(Cmem))))))	No	4	1	Х

#### Opcode

| 1111 | 1101 | AAAA | AAAI | 0010 | 11mm | DDDD | uug%

#### **Operands**

ACx, ACy, Cmem, Smem

# Description

This instruction performs two parallel multiply and accumulate (MAC) operations in one cycle. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand HI(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC2 (this data is shared to MAC1 and MAC2). The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA (effective address); the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and an accumulation in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand LO(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- ☐ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.
- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.

- ☐ The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator, which has been shifted to the right by 16 bits. The shifting operation is performed with a sign extension of source accumulator bit 39. Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ☐ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

### Compatibility with C54x devices (C54CM = 1)

None.

**Status Bits** Affected by FRCT, M40, RDM, SATD, SMUL

> ACOVx, ACOVy Affects

Repeat This instruction can be repeated.

# **Example**

Syntax	Description
AC1 = (AC1 >> #16) + (uns(*AR3-) * uns(HI(coef(*CDP+)))), AC0 = (AC0 >> #16) + (uns(*AR3-) * uns(LO(coef(*CDP+))))	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1, which has been shifted to the right by 16 bits. The result is stored in AC1. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the lower part of CDP is added to the content of AC0, which has been shifted to the right by 16 bits. The result is stored in AC0. AR3 is decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

#### Execution

(ACy>>#16) +M40 (rnd (uns (Smem) [16:0] \*uns (HI (coef (Cmem))) [16:0])) -> ACy (ACx>>#16)+M40(rnd(uns(Smem)[16:0]\*uns(LO(coef(Cmem)))[16:0])) -> ACx

Before				After			
AC0	00	0800	0000	AC0	00	3F80	0800
XAR3		00	10FF	XAR3		00	10FE
Data memory							
10FFh			FE00	10FFh			FE00
XCDP		00	2000	XCDP		00	2002
Coeff memory	Y						
2001h			4000	2001h			4000
AC1	00	0800	0000	AC1	00	7F00	0800
Coeff memory	Y						
2000h			8000	2000h			8000

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	ACy = M40(rnd(ACy + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx + (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))	No	4	1	Х

#### **Opcode**

| 1111 | 1101 | AAAA | AAAI | 0101 | 01mm | DDDD | uug%

### **Operands**

ACx, ACy, Cmem, Lmem

### **Description**

This instruction performs two parallel multiply and accumulate (MAC) operations in one cycle. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand HI(Lmem) and the content of data memory operand HI(coef(Cmem)). The data memory operand HI(Lmem) is addressed by DAGEN path X with the EA (effective address); the data, which can be assumed to be the higher part of long word memory data, is driven on data bus CDB, and sign extended to 17 bits in the MAC2. The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and an accumulation in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand LO(Lmem) and the content of data memory operand LO(coef(Cmem)). The data memory operand LO(Lmem) is addressed by DAGEN path X with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word memory data, is driven on data bus DDB, and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- ☐ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.
- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.

		Multiplication	on overflow detection depends on SMUL.			
			result of the multiplication is sign extended to 40 bits and added ce accumulator.			
		•	s performed according to RDM, if the optional rnd keyword is he instruction.			
			letection depends on M40. If an overflow is detected, the accumulator overflow status bit is set.			
		When an ov SATD.	verflow is detected, the accumulator is saturated according to			
	This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.					
	and bus pre	d B2DB buse ses are only	tion, the Cmem operand is accessed through the BAB, BDB, es; on some C55xx-based devices, the BAB, BDB, and B2DB connected to internal memory and not to external memory. To the leration of a bus error, the Cmem operand must not be mapped mory.			
	Co	mpatibility	with C54x devices (C54CM = 1)			
	No	ne.				
Status Bits	Affe	ected by	FRCT, M40, RDM, SATD, SMUL			
	Affe	ects	ACOVx, ACOVy			

# **Example**

Repeat

Syntax	Description
AC1 = AC1 + (uns(HI(*AR3-)) * uns(HI(coef(*CDP+)))), AC0 = AC0 + (uns(LO(*AR3-)) * uns(LO(coef(*CDP+))))	Both instructions are performed in parallel. The product of the unsigned content addressed by the higher part of AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by the lower part of AR3 and the unsigned content addressed by the lower part of CDP is added to the content of AC0. The result is stored in AC0. When AR3– is used with HI/LO, AR3 is decremented by 2. When CDP+ is used with HI/LO, CDP is incremented by 2.

This instruction can be repeated.

# MAC::MAC Parallel Multiply and Accumulates

Before				After			
AC0	00	0000	8000	AC0	00	3F80	8000
XAR3		00	10FE	XAR3		00	10FC
Data memory							
10FFh			FE00	10FFh			FE00
XCDP		00	2000	XCDP		00	2002
Coeff memory	Y						
2001h			4000	2001h			4000
AC1	00	0000	8000	AC1	00	7F80	8000
Data memory							
10FEh			FF00	10FEh			FF00
Coeff memory	Y						
2000h			8000	2000h			8000

5-438 Instruction Set Descriptions SWPU068E

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	ACy = M40(rnd(ACy + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd((ACx >> #16) + (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))	No	4	1	Х

#### **Opcode**

| 1111 | 1101 | AAAA | AAAI | 0110 | 00mm | DDDD | uug%

#### **Operands**

ACx, ACy, Cmem, Lmem

# **Description**

This instruction performs two parallel multiply and accumulate (MAC) operations in one cycle. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand HI(Lmem) and the content of data memory operand HI(coef(Cmem)). The data memory operand HI(Lmem) is addressed by DAGEN path X with the EA (effective address); the data, which can be assumed to be the higher part of long word memory data, is driven on data bus CDB, and sign extended to 17 bits in the MAC2. The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and an accumulation in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand LO(Lmem) and the content of data memory operand LO(coef(Cmem)). The data memory operand LO(Lmem) is addressed by DAGEN path X with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word memory data, is driven on data bus DDB and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- ☐ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.
- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.

		Multiplication	on overflow detection depends on SMUL.				
			st operation, the 32-bit result of the multiplication is sign a 40 bits and added to the source accumulator ACy.				
		extended to been shifted	cond operation, the 32-bit result of the multiplication is sign of 40 bits and added to the source accumulator ACx, which has do to the right by 16 bits. The shifting operation is performed with ansion of source accumulator ACx(39).				
		_	s performed according to RDM, if the optional rnd keyword is he instruction.				
			etection depends on M40. If an overflow is detected, the accumulator overflow status bit is set.				
		When an o	verflow is detected, the accumulator is saturated according to				
	This instruction provides the option to locally set M40 to 1 for the executhe instruction, if the optional M40 keyword is applied to the instruction						
	and bus pre	d B2DB buse ses are only	tion, the Cmem operand is accessed through the BAB, BDB, es; on some C55xx-based devices, the BAB, BDB, and B2DB connected to internal memory and not to external memory. To the learning of a bus error, the Cmem operand must not be mapped mory.				
	Со	mpatibility	with C54x devices (C54CM = 1)				
	No	ne.					
Status Bits	Affe	ected by	FRCT, M40, RDM, SATD, SMUL				
	Affe	ects	ACOVx, ACOVy				
Repeat	Thi	s instruction	can be repeated.				
Example							

Syntax	Description
AC1 = AC1 + (uns(HI(*AR3-)) * uns(HI(coef(*CDP+)))), AC0 = (AC0 >> #16) + (uns(LO(*AR3-)) * uns(LO(coef(*CDP+))))	Both instructions are performed in parallel. The product of the unsigned content addressed by the higher part of AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by the lower part of AR3 and the unsigned content addressed by the lower part of CDP is added to the content of AC0, which has been shifted to the right by 16 bits. The result is stored in AC0. When AR3– is used with HI/LO, AR3 is decremented by 2. When CDP+ is used with HI/LO, CDP is incremented by 2.

### Execution

ACy+M40(rnd(uns(HI(Lmem))[16:0]\*uns(HI(coef(Cmem)))[16:0])) -> ACy (ACx>>#16)+M40(rnd(uns(LO(Lmem))[16:0]\*uns(LO(coef(Cmem)))[16:0])) -> ACx

Before				After			
AC0	00	0800	8000	AC0	00	3F80	0800
XAR3		00	10FE	XAR3		00	10FC
Data memory							
10FFh			FE00	10FFh			FE00
XCDP		00	2000	XCDP		00	2002
Coeff memory	Y						
2001h			4000	2001h			4000
AC1	00	0000	8000	AC1	00	7F80	8000
Data memory							
10FEh			FF00	10FEh			FF00
Coeff memory	Y						
2000h			8000	2000h			8000

### Parallel Multiply and Accumulates

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[9]	ACy = M40(rnd((ACy >> #16) + (uns(HI(Lmem))* uns(HI(coef(Cmem)))))), ACx = M40(rnd((ACx >> #16) + (uns(LO(Lmem))* uns(LO(coef(Cmem))))))	No	4	1	Х

#### **Opcode**

| 1111 | 1101 | AAAA | AAAI | 0110 | 11mm | DDDD | uug%

#### **Operands**

ACx, ACy, Cmem, Lmem

### **Description**

This instruction performs two parallel multiply and accumulate (MAC) operations in one cycle. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand HI(Lmem) and the content of data memory operand HI(coef(Cmem)). The data memory operand HI(Lmem) is addressed by DAGEN path X with the EA (effective address); the data, which can be assumed to be the higher part of long word memory data, is driven on data bus CDB and sign extended to 17 bits in the MAC2. The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and an accumulation in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand LO(Lmem) and the content of data memory operand LO(coef(Cmem)). The data memory operand LO(Lmem) is addressed by DAGEN path X with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word memory data, is driven on data bus DDB and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- ☐ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.
- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.

	Multiplication overflow detection depends on SMUL.
	The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator, which has been shifted to the right by 16 bits. The shifting operation is performed with a sign extension of source accumulator bit 39.
	Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
	Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
	When an overflow is detected, the accumulator is saturated according to SATD.
Thi	s instruction provides the option to locally set M40 to 1 for the execution of

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

### Compatibility with C54x devices (C54CM = 1)

None.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVx, ACOVy

**Repeat** This instruction can be repeated.

Syntax	Description
AC1 = (AC1 >> #16) + (uns(HI(*AR3-)) * uns(HI(coef(*CDP+)))), AC0 = (AC0 >> #16) + (uns(LO(*AR3-)) * uns(LO(coef(*CDP+))))	Both instructions are performed in parallel. The product of the unsigned content addressed by the higher part of AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1, which has been shifted to the right by 16 bits. The result is stored in AC1. The product of the unsigned content addressed by lower part of AR3 and the unsigned content addressed by the lower part of CDP is added to the content of AC0, which has been shifted to the right by 16 bits. The result is stored in AC0. When AR3– is used with HI/LO, AR3 is decremented by 2. When CDP+ is used with HI/LO, CDP is incremented by 2.

### Execution

(ACy>>#16) +M40(rnd(uns(HI(Lmem))[16:0]\*uns(HI(coef(Cmem)))[16:0])) -> ACy
(ACx>>#16) +M40(rnd(uns(LO(Lmem))[16:0]\*uns(LO(coef(Cmem)))[16:0])) -> ACx

Before				After			
AC0	00	0800	0000	AC0	00	3F80	0800
XAR3		00	10FE	XAR3		00	10FC
Data memory							
10FFh			FE00	10FFh			FE00
XCDP		00	2000	XCDP		00	2002
Coeff memory	7						
2001h			4000	2001h			4000
AC1	00	0800	0000	AC1	00	7F80	0800
Data memory							
10FEh			FF00	10FEh			FF00
Coeff memory	7						
2000h			8000	2000h			8000

5-444 Instruction Set Descriptions

### Parallel Multiply and Accumulates

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[10]	ACy = M40(rnd(ACy + uns(Ymem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx + uns(Xmem) * uns(LO(coef(Cmem)))))	No	5 (*)	1	X

(\*) 1 LSB is allocated to instruction slot #2.

#### Opcode

1001 0011 XXXM MMYY YMMM 00mm uuDD DDg%

### **Operands**

ACx, ACy, Cmem, Xmem, Ymem

### **Description**

This instruction performs two parallel multiply and accumulate (MAC) operations in one cycle. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the contents of data memory operand Ymem, extended to 17 bits, and the content of data memory operand HI(coef(Cmem)) which is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and an accumulation in the D-unit MAC1. The input operands of the multiplier are the contents of data memory operand Xmem, extended to 17 bits, and the content of data memory operand LO(coef(Cmem)) which is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- ☐ The input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.

	The 32-bit result of the multiplication is sign extended to 40 bits.						
	Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.						
	Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.						
	When an overflow is detected, the accumulator is saturated according to SATD.						
	Because this instruction occupies both instruction slots #1 and #2, this can not be executed in parallel with other instructions.						
	The Xmem operand can access the MMRs but the Ymem operand can not.						
Thi	This instruction provides the option to locally set M40 to 1 for the execution of						

the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

### Compatibility with C54x devices (C54CM = 1)

None.

**Status Bits** Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

> Affects ACOVx, ACOVy

Repeat This instruction can be repeated.

Syntax	Description
AC1 = AC1 + (uns(*AR3-) * uns(HI(coef(*CDP+)))), AC0 = AC0 + (uns(*AR2-) * uns(LO(coef(*CDP+))))	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by AR2 and the unsigned content addressed by the lower part of the CDP is added to the content of AC0. The result is stored in AC0. AR3 and AR2 are decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

### Execution

M40(rnd(ACx + uns(Xmem)[16:0] \* uns(LO(coef(Cmem)))[16:0])) -> ACx M40(rnd(ACy + uns(Ymem)[16:0] \* uns(HI(coef(Cmem)))[16:0])) -> ACy

Before				After				
AC0	00	0000	8000	AC0	00	3F80	8000	
XAR2		00	10FE	XAR2		00	10FD	
XAR3		00	20FE	XAR3		00	20FD	
Data memory	Data memory							
10FEh			FE00	10FEh			FE00	
XCDP		00	2000	XCDP		00	2002	
Coeff memor	У							
2001h			4000	2001h			4000	
AC1	00	0000	8000	AC1	00	7F80	8000	
Data memory								
20FEh			FF00	20FFh			FF00	
Coeff memor	У							
2000h			8000	2000h			8000	

### Parallel Multiply and Accumulates

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[11]	ACy = M40(rnd(ACy + (uns(Ymem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(LO(coef(Cmem))))))	No	5 (*)	1	Х

(\*) 1 LSB is allocated to instruction slot #2.

Opcode

1001 0011 XXXM MMYY YMMM 10mm uuDD DDg%

**Operands** 

ACx, ACy, Cmem, Xmem, Ymem

**Description** 

This instruction performs two parallel multiply and accumulate (MAC) operations in one cycle. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the contents of data memory operand Ymem, extended to 17 bits, and the contents of data memory operand HI(coef(Cmem)) which is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and an accumulation in the D-unit MAC1. The input operands of the multiplier are the contents of data memory operand Xmem, extended to 17 bits, and the contents of data memory operand LO(coef(Cmem)) which is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- ☐ The input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.

	The 32-bit result of the multiplication is sign extended to 40 bits.
	Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
	Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
	When an overflow is detected, the accumulator is saturated according to SATD.
	Because this instruction occupies both instruction slots #1 and #2, this can not be executed in parallel with other instructions.
	The Xmem operand can access the MMRs but the Ymem operand can not.
<b>-</b>	

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

### Compatibility with C54x devices (C54CM = 1)

None.

**Status Bits** Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

> Affects ACOVx, ACOVy

Repeat This instruction can be repeated.

Syntax	Description
AC1 = AC1 + (uns(*AR3-) * uns(HI(coef(*CDP+)))),	Both instructions are performed in parallel. The prod-
AC0 = (AC0 >> #16) + (uns(*AR2-) *	uct of the unsigned content addressed by AR3 and the
uns(LO(coef(*CDP+))))	unsigned content addressed by the higher part of the
	coefficient data pointer register (CDP) is added to the
	content of AC1. The result is stored in AC1. The prod-
	uct of the unsigned content addressed by AR2 and the
	unsigned content addressed by the lower part of the
	CDP is added to the content of AC0, which has been
	shifted to the right by 16 bits. The result is stored in
	AC0. AR3 and AR2 are decremented by 1. When
	CDP+ is used with HI/LO, CDP is incremented by 2.

### Execution

M40(rnd((ACx >> #16) + uns(Xmem)[16:0] \* uns(LO(coef(Cmem)))[16:0])) -> ACx  $\texttt{M40(rnd(ACy + uns(Ymem)[16:0] * uns(HI(coef(Cmem)))[16:0]))} \ \ -> \ \texttt{ACy}$ 

Before				After			
AC0	00	0800	8000	AC0	00	3F80	0800
XAR2		00	10FE	XAR2		00	10FD
XAR3		00	20FE	XAR3		00	20FD
Data memory							
10FEh			FE00	10FEh			FE00
XCDP		00	2000	XCDP		00	2002
Coeff memor	У						
2001h			4000	2001h			4000
AC1	00	0000	8000	AC1	00	7F80	8000
Data memory							
20FEh			FF00	20FFh			FF00
Coeff memor	У						
2000h			8000	2000h			8000

### Parallel Multiply and Accumulates

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[12]	ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(LO(coef(Cmem))))))	No	5 (*)	1	Х

(\*) 1 LSB is allocated to instruction slot #2.

#### Opcode

1001 0011 XXXM MMYY YMMM 11mm uuDD DDg%

### **Operands**

ACx, ACy, Cmem, Xmem, Ymem

### **Description**

This instruction performs two parallel multiply and accumulate (MAC) operations in one cycle. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the contents of data memory operand Ymem, extended to 17 bits, and the content of data memory operand HI(coef(Cmem)) which is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and an addition in the D-unit MAC1. The input operands of the multiplier are the contents of data memory operand Xmem, extended to 17 bits, and the content of data memory operand LO(coef(Cmem)) which is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- ☐ The input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.

The 32-bit result of the multiplication is sign extended to 40 bits.					
Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.					
Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.					
When an overflow is detected, the accumulator is saturated according to SATD.					
Because this instruction occupies both instruction slots #1 and #2, this can not be executed in parallel with other instructions.					
The Xmem operand can access the MMRs but the Ymem operand can not.					
This instruction provides the option to locally set M40 to 1 for the execution of					

the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

### Compatibility with C54x devices (C54CM = 1)

None.

**Status Bits** Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

> Affects ACOVx, ACOVy

Repeat This instruction can be repeated.

Syntax	Description
Syntax  AC1 = AC1 + (uns(*AR3-) * uns(HI(coef(*CDP+)))),  AC0 = (AC0 >> #16) + (uns(*AR2-) *  uns(LO(coef(*CDP+))))	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by AR2 and the unsigned content addressed by the lower part of the CDP is added to the content of AC0, which has been shifted to the right by 16 bits. The result is stored in
	AC0. AR3 and AR2 are decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

### Execution

```
M40(rnd((ACx >> #16) + uns(Xmem)[16:0] * uns(LO(coef(Cmem)))[16:0])) -> ACx
M40(rnd((ACy >> #16) + uns(Ymem)[16:0] * uns(HI(coef(Cmem)))[16:0])) -> ACy
```

Before				After			
AC0	00	0800	8000	AC0	00	3F80	0800
XAR2		00	10FE	XAR2		00	10FD
XAR3		00	20FE	XAR3		00	20FD
Data memory							
10FEh			FE00	10FEh			FE00
XCDP		00	2000	XCDP		00	2002
Coeff memory	Y						
2001h			4000	2001h			4000
AC1	00	8000	0000	AC1	00	7F80	8000
Data memory							
20FEh			FF00	20FFh			FF00
Coeff memory	Y						
2000h			8000	2000h			8000

### MAS::MAS

### Parallel Multiply and Subtracts

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	$ \begin{aligned} &ACx = M40(rnd(ACx - \mathbf{(uns}(Xmem) * uns(\mathbf{coef}(Cmem))))), \\ &ACy = M40(rnd(ACy - \mathbf{(uns}(Ymem) * uns(\mathbf{coef}(Cmem))))) \end{aligned} $	No	4	1	Х
[2]	ACy = M40(rnd(ACy - (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx - (uns(Smem) * uns(LO(coef(Cmem))))))	No	4	1	Х
[3]	ACy = M40(rnd(ACy - (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx - (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))	No	4	1	Х
[4]	ACy = M40(rnd(ACy - (uns(Ymem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx - (uns(Xmem) * uns(LO(coef(Cmem))))))	No	5	1	Х

**Description** These instructions perform two parallel multiply and subtract (MAS) operations in one cycle. The operations are executed in the two D-unit MACs. **Status Bits** Affected by FRCT, M40, RDM, SATD, SMUL, SXMD Affects ACOVx, ACOVy See Also See the following other related instructions: Modify Auxiliary Register Content with Parallel Multiply and Subtract Multiply and Subtract Multiply and Subtract with Parallel Load Accumulator from Memory Multiply and Subtract with Parallel Multiply Multiply and Subtract with Parallel Multiply and Accumulate Multiply and Subtract with Parallel Store Accumulator Content to Memory Parallel Multiplies Parallel Multiply and Accumulates

### Parallel Multiply and Subtracts

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = M40(rnd(ACx - (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd(ACy - (uns(Ymem) * uns(coef(Cmem)))))	No	4	1	Х

### Opcode

1000 0101 XXXM MMYY YMMM 01mm uuDD DDg%

### **Operands**

ACx, ACy, Cmem, Xmem, Ymem

### **Description**

This instruction performs two parallel multiply and subtract (MAS) operations in one cycle. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Xmem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

The second operation performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Ymem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

- ☐ Input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- ☐ The 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Each data flow can also disable the usage of the corresponding MAC unit, while allowing the modification of auxiliary registers in the three address generation units through the following instructions:

- mar(Xmem)
- mar(Ymem)
- mar(Cmem)

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVx, ACOVy

**Repeat** This instruction can be repeated.

Syntax	Description
AC0 = AC0 - (uns(*AR3) * uns(coef(*CDP))), AC1 = AC1 - (uns(*AR4) * uns(coef(*CDP)))	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the coefficient data pointer register (CDP) is subtracted from the content of AC0. The result is stored in AC0. The product of the unsigned content addressed by AR4 and the unsigned content addressed by CDP is subtracted from the content of AC1. The result is stored in AC1.

### Parallel Multiply and Subtracts

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACy = M40(rnd(ACy - (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx - (uns(Smem) * uns(LO(coef(Cmem)))))))	No	4	1	Х

#### Opcode

| 1111 | 1101 | AAAA | AAAI | 0011 | 00mm | DDDD | uug%

#### **Operands**

ACx, ACy, Cmem, Smem

### Description

This instruction performs two parallel multiply and subtract (MAS) operations in one cycle. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and a subtraction in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand HI(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC2 (this data is shared to MAC1 and MAC2). The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA (effective address); the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and a subtraction in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand LO(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- ☐ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.
- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.

The 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator.
 Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
 Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
 When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

### Compatibility with C54x devices (C54CM = 1)

None.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVx, ACOVy

**Repeat** This instruction can be repeated.

#### **Example**

Syntax	Description
AC1 = AC1 - (uns(*AR3-) * uns(HI(coef(*CDP+)))), AC0 = AC0 - (uns(*AR3-) * uns(LO(coef(*CDP+))))	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is subtracted from the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the lower part of CDP is subtracted from the content of AC0. The result is stored in AC0. AR3 is decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

#### Execution

ACy-M40(rnd(uns(Smem)[16:0]\*uns(HI(coef(Cmem)))[16:0])) -> ACy
ACx-M40(rnd(uns(Smem)[16:0]\*uns(LO(coef(Cmem)))[16:0])) -> ACx

Before				After			
AC0	00	0000	8000	AC0	FF	C080	8000
XAR3		00	10FF	XAR3		00	10FE
Data memory							
10FFh			FE00	10FFh			FE00
XCDP		00	2000	XCDP		00	2002
Coeff memory	У						
2001h			4000	2001h			4000
AC1	00	0000	8000	AC1	FF	8100	8000
Coeff memory	У						
2000h			8000	2000h			8000

### Parallel Multiply and Subtracts

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ACy = M40(rnd(ACy - (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx - (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))	No	4	1	Х

#### **Opcode**

| 1111 | 1101 | AAAA | AAAI | 0111 | 00mm | DDDD | uug%

#### **Operands**

ACx, ACy, Cmem, Lmem

### **Description**

This instruction performs two parallel multiply and subtract (MAS) operations in one cycle. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and a subtraction in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand HI(Lmem) and the content of data memory operand HI(coef(Cmem)). The data memory operand HI(Lmem) is addressed by DAGEN path X with the EA (effective address); the data, which can be assumed to be the higher part of long word memory data, is driven on data bus CDB and sign extended to 17 bits in the MAC2. The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and a subtraction in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand LO(Lmem) and the content of data memory operand LO(coef(Cmem)). The data memory operand LO(Lmem) is addressed by DAGEN path X with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word memory data, is driven on data bus DDB and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- ☐ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.
- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.

		Multiplication	on overflow detection depends on SMUL.	
			result of the multiplication is sign extended to 40 bits and from the source accumulator.	
		_	s performed according to RDM, if the optional rnd keyword is the instruction.	
			detection depends on M40. If an overflow is detected, the accumulator overflow status bit is set.	
		When an o	verflow is detected, the accumulator is saturated according to	
			provides the option to locally set M40 to 1 for the execution of if the optional M40 keyword is applied to the instruction.	
	and bus pre	d B2DB buse ses are only	tion, the Cmem operand is accessed through the BAB, BDB, es; on some C55xx-based devices, the BAB, BDB, and B2DB connected to internal memory and not to external memory. To neration of a bus error, the Cmem operand must not be mapped mory.	
	Co	mpatibility	with C54x devices (C54CM = 1)	
	No	ne.		
Status Bits	Aff	ected by	FRCT, M40, RDM, SATD, SMUL	
	Aff	ects	ACOVx, ACOVy	
Repeat	Thi	is instruction can be repeated.		

Syntax	Description
AC1 = AC1 - (uns(HI(*AR3-)) * uns(HI(coef(*CDP+)))), AC0 = AC0 - (uns(LO(*AR3-)) * uns(LO(coef(*CDP+))))	Both instructions are performed in parallel. The product of the unsigned content addressed by the higher part of AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is subtracted from the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by the lower part of AR3 and the unsigned content addressed by the lower part of CDP is subtracted from the content of AC0. The result is stored in AC0. When AR3— is used with HI/LO, AR3 is decremented by 2. When CDP+ is used with HI/LO, CDP is incremented by 2.

# MAS::MAS Parallel Multiply and Subtracts

Before				After			
AC0	00	0000	8000	AC0	FF	C080	8000
XAR3		00	10FE	XAR3		00	10FC
Data memory							
10FFh			FE00	10FFh			FE00
XCDP		00	2000	XCDP		00	2002
Coeff memor	У						
2001h			4000	2001h			4000
AC1	00	0000	8000	AC1	FF	8080	8000
Data memory							
10FEh			FF00	10FEh			FF00
Coeff memor	У						
2000h			8000	2000h			8000

5-462 Instruction Set Descriptions SWPU068E

### Parallel Multiply and Subtracts

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	ACy = M40(rnd(ACy - (uns(Ymem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx - (uns(Xmem) * uns(LO(coef(Cmem)))))))	No	5 (*)	1	Х

(\*) 1 LSB is allocated to instruction slot #2.

#### Opcode

| 1001 0101 | XXXM MMYY | YMMM 01mm | uuDD DDg%

### **Operands**

ACx, ACy, Cmem, Xmem, Ymem

### **Description**

This instruction performs two parallel multiply and subtraction (MAS) operations in one cycle. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and a subtraction in the D-unit MAC2. The input operands of the multiplier are the contents of data memory operand Ymem, extended to 17 bits, and the content of data memory operand HI(coef(Cmem)) which is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and a subtraction in the D-unit MAC1. The input operands of the multiplier are the contents of data memory operand Xmem, extended to 17 bits, and the content of data memory operand LO(coef(Cmem)) which is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA-1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- ☐ The input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- ☐ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.

	The 32-bit result of the multiplication is sign extended to 40 bits.
	Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
	Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
	When an overflow is detected, the accumulator is saturated according to SATD.
	Because this instruction occupies both instruction slots #1 and #2, this can not be executed in parallel with other instructions.
	The Xmem operand can access the MMRs but the Ymem operand can not.
Th	is instruction provides the entire to levelly set M40 to 4 for the execution of

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

### Compatibility with C54x devices (C54CM = 1)

None.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVx, ACOVy

**Repeat** This instruction can be repeated.

Syntax	Description
AC1 = AC1 - (uns(*AR3-) * uns(HI(coef(*CDP+)))), AC0 = AC0 - (uns(*AR2-) * uns(LO(coef(*CDP+))))	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is subtracted from the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by AR2 and the unsigned content addressed by the lower part of the CDP is subtracted from the content of AC0. The result is stored in AC0. AR3 and AR2 are decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

### Execution

M40(rnd(ACx - uns(Xmem)[16:0] \* uns(LO(coef(Cmem)))[16:0])) -> ACx M40(rnd(ACy - uns(Ymem)[16:0] \* uns(HI(coef(Cmem)))[16:0])) -> ACy

Before				After			
AC0	00	0000	8000	AC0	FF	C080	8000
XAR2		00	10FE	XAR2		00	10FD
XAR3		00	20FE	XAR3		00	20FD
Data memory							
10FEh			FE00	10FEh			FE00
XCDP		00	2000	XCDP		00	2002
Coeff memor	У						
2001h			4000	2001h			4000
AC1	00	0000	8000	AC1	FF	8080	8000
Data memory							
20FEh			FF00	20FFh			FF00
Coeff memor	У						
2000h			8000	2000h			8000

### port

### Peripheral Port Register Access Qualifiers

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	readport()	No	1	1	D
[2]	writeport()	No	1	1	D
Opcode	-			100	1 1001
	writeport			100	1 1010

#### **Operands**

#### none

### **Description**

These operand qualifiers allow you to locally disable access toward the data memory and enable access to the 64K-word I/O space. The I/O data location is specified by the Smem, Xmem, or Ymem fields.

- ☐ A readport() operand qualifier may be included in any instruction making a word single data memory access Smem or Xmem that is used in a read operation, except instructions using delay().
- □ A readport() operand qualifier cannot be used in any instruction making a dual memory access Xmem or Ymem that is used in read operation. There is an exception for the instructions making a dual read/write memory access of the type Ymem = Xmem, or Smem = coeff, where readport() qualifier can be used.
- ☐ A writeport() operand qualifier may be included in any instruction making a word single data memory access Smem or Ymem that is used in a write operation, except instructions using the delay().
- ☐ A writeport() operand qualifier cannot be used in any instruction making a dual memory access Xmem or Ymem that is used in write operation. There is an exception for the instructions making a dual read/write memory access of the type Ymem = Xmem, or coeff = Smem, where writeport() qualifier can be used.
- A readport() or writeport() operand qualifier cannot be used as a stand-alone instruction (the assembler generates an error message).

Any instruction making a word single data memory access Smem (except those listed above) can use the \*port(#k16) addressing mode to access the 64K-word I/O space with an immediate address. When an instruction uses \*port(#k16), the 16-bit unsigned constant, k16, is encoded in a 2-byte extension to the instruction. Because of the extension, an instruction using \*port(#k16) cannot be executed in parallel with another instruction.

The following indirect operands cannot be used for accesses to I/O space. An instruction using one of these operands requires a 2-byte extension to the instruction. Because of the extension, an instruction using one of the following indirect operands cannot be executed with these operand qualifiers.

	(#K16	)
--	-------	---

\*+ARn(#K16)

□ \*CDP(#K16)

→ \*+CDP(#K16)

**Status Bits** Affected by none

> Affects none

Repeat An instruction using this operand qualifier can be repeated.

### **Example 1**

Syntax	Description
T2 = *AR3    readport()	The content addressed by AR3 (I/O address) is loaded into T2.

Syntax	Description
*AR3 = T2	The content of T2 is written to the location addressed by AR3 (I/O address).
writeport()	

# POPBOTH

Pop Accumulator or Extended Auxiliary Register Content from Stack Pointers

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	xdst = popboth()	Yes	2	1	Х
Opcod		010	01 00	OE XDD	D 0100

**Operands** xdst

Description

This instruction moves the content of two 16-bit data memory locations addressed by the data stack pointer (SP) and system stack pointer (SSP) to accumulator ACx or to the 23-bit destination register (XARx, XSP, XSSP, XDP, or XCDP).

The content of xdst(15–0) is loaded from the location addressed by SP and the content of xdst(31–16) is loaded from the location addressed by SSP. The return address register (RETA) and the control-flow context register (CFCT) are not accessed by this instruction even in the fast-return process.

When xdst is a 23-bit register, the upper 9 bits of the data memory addressed by SSP are discarded and only the 7 lower bits of the data memory are loaded into the high part of xdst(22–16).

When xdst is an accumulator, the guard bits, ACx(39–32), are reloaded (unchanged) with the current value and are not modified by this instruction.

Status Bits Affected by none

Affects none

**Repeat** This instruction can be repeated.

**See Also** See the following other related instructions:

□ Pop Top of Stack

Push to Top of Stack

Push Accumulator or Extended Auxiliary Register Content to Stack Pointers

POP

Pop Top of Stack

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst1, dst2 = <b>pop()</b>	Yes	2	1	Х
[2]	dst = pop()	Yes	2	1	X
[3]	dst, Smem = <b>pop()</b>	No	3	1	X
[4]	$ACx = \mathbf{dbl(pop())}$	Yes	2	1	X
[5]	Smem = pop()	No	2	1	X
[6]	dbl(Lmem) = pop()	No	2	1	Х

These instructions move the content of the data memory location addressed by the data stack pointer (SP) to:

- an accumulator, auxiliary, or temporary register
- a data memory location

The return address register (RETA) and the control-flow context register (CFCT) are not accessed by this instruction even in the fast-return process.

When the destination register is an accumulator, the guard bits and the 16 higher bits of the accumulator, ACx(39–16), are reloaded (unchanged) with the current value and are not modified by these instructions.

The increment operation performed on SP is done by the A-unit address generator dedicated to the stack addressing management.

#### **Status Bits**

Affected by none

Affects none

#### See Also

See the following other related instructions:

- Pop Accumulator or Extended Auxiliary Register Content from Stack Pointers
- ☐ Push to Top of Stack
- ☐ Push Accumulator or Extended Auxiliary Register Content to Stack Pointers

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst1, dst2 = <b>pop()</b>	Yes	2	1	Х

Opcode 0011 101E FSSS FDDD

Note: FSSS = dst1, FDDD = dst2

Operands dst1, dst2

**Description** This instruction moves the content of the 16-bit data memory location pointed

by SP to destination register dst1 and moves the content of the 16-bit data

memory location pointed by SP + 1 to destination register dst2.

When the destination register, dst1 or dst2, is an accumulator, the content of the 16-bit data memory operand is moved to the destination accumulator low part, ACx(15–0). The guard bits and the 16 higher bits of the accumulator, ACx(39–16), are reloaded (unchanged) with the current value and are not

modified by this instruction. SP is incremented by 2.

Status Bits Affected by none

Affects none

**Repeat** This instruction can be repeated.

Syntax	Description
AC0, AC1 = pop()	The content of the memory location pointed by the data stack pointer (SP) is copied to AC0(15–0) and the content of the memory location pointed by SP + 1 is copied to AC1(15–0). Bits 39–16 of the accumulators are unchanged. The SP is incremented by 2.

Before				After				
AC0	00	4500	0000	AC0	00	4500	4890	
AC1	F7	5678	9432	AC1	F7	5678	2300	
SP			0300	SP			0302	
300			4890	300			4890	
301			2300	301			2300	

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	dst = pop()	Yes	2	1	Χ

0101 000E FDDD x010 **Opcode** 

**Operands** dst

**Description** This instruction moves the content of the 16-bit data memory location pointed

by SP to destination register dst.

When the destination register, dst, is an accumulator, the content of the 16-bit data memory operand is moved to the destination accumulator low part, ACx(15-0). The guard bits and the 16 higher bits of the accumulator, ACx(39-16), are reloaded (unchanged) with the current value and are not

modified by this instruction. SP is incremented by 1.

**Status Bits** Affected by none

> Affects none

Repeat This instruction cannot be repeated with a single conditional or unconditional

repeat instruction. It can be repeated in other repeat instructions.

Syntax	Description
AC0 = pop()	The content of the memory location pointed by the data stack pointer (SP) is copied to AC0(15–0). Bits 39–16 of AC0 are unchanged. The SP is incremented by 1.

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	dst, Smem = <b>pop()</b>	No	3	1	Х

1110 0100 AAAA AAAI FDDD x1xx Opcode

**Operands** dst, Smem

**Description** This instruction moves the content of the 16-bit data memory location pointed

by SP to destination register dst and moves the content of the 16-bit data

memory location pointed by SP + 1 to data memory (Smem) location.

When the destination register, dst, is an accumulator, the content of the 16-bit data memory operand is moved to the destination accumulator low part, ACx(15-0). The guard bits and the 16 higher bits of the accumulator, ACx(39-16), are reloaded (unchanged) with the current value and are not

modified by this instruction. SP is incremented by 2.

**Status Bits** Affected by none

> Affects none

Repeat This instruction can be repeated.

Syntax	Description
AC0, *AR3 = pop()	The content of the memory location pointed by the data stack pointer (SP) is copied to AC0(15–0) and the content of the memory location pointed by SP + 1 is copied to the location addressed by AR3. Bits 39–16 of AC0 are unchanged. The SP is incremented by 2.

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	ACx = dbl(pop())	Yes	2	1	X

**Opcode** 0101 000E | xxDD x011

**Operands** ACx

**Description** This instruction moves the content of the 16-bit data memory location pointed

by SP to the accumulator high part ACx(31–16) and moves the content of the 16-bit data memory location pointed by SP + 1 to the accumulator low part

ACx(15-0).

The guard bits of the accumulator, ACx(39-32), are reloaded (unchanged) with the current value and are not modified by this instruction. SP is

incremented by 2.

Status Bits Affected by none

Affects none

**Repeat** This instruction cannot be repeated with a single conditional or unconditional

repeat instruction. It can be repeated in other repeat instructions.

Syntax	Description
AC1 = dbl(pop())	The content of the memory location pointed by the data stack pointer (SP) is copied to AC1(31–16) and the content of the memory location pointed by SP + 1 is copied
	to AC1(15–0). Bits 39–32 of AC1 are unchanged. The SP is incremented by 2.

Before				After			
AC1	03	3800	FC00	AC1	03	5644	F800
SP			0304	SP			0306
304			5644	304			5644
305			F800	305			F800

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	Smem = pop()	No	2	1	Х

| 1011 | 1011 | AAAA AAAI Opcode

**Operands** Smem

Description This instruction moves the content of the 16-bit data memory location pointed

by SP to data memory (Smem) location. SP is incremented by 1.

**Status Bits** Affected by none

> Affects none

Repeat This instruction can be repeated.

Syntax	Description
*AR1 = pop()	The content of the memory location pointed by the data stack pointer (SP) is copied to the location addressed by AR1. The SP is incremented by 1.

Before		After	
AR1	0200	AR1	0200
SP	0300	SP	0301
200	3400	200	6903
300	6903	300	6903

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	dbl(Lmem) = pop()	No	2	1	Х

#### 1011 1000 AAAA AAAI **Opcode**

**Operands** Lmem

**Description** This instruction moves the content of the 16-bit data memory location pointed

by SP to the 16 highest bits of data memory location Lmem and moves the content of the 16-bit data memory location pointed by SP + 1 to the 16 lowest

bits of data memory location Lmem.

When Lmem is at an even address, the two 16-bit values popped from the stack are stored at memory location Lmem in the same order. When Lmem is at an odd address, the two 16-bit values popped from the stack are stored at

memory location Lmem in the reverse order.

SP is incremented by 2.

**Status Bits** Affected by none

> Affects none

Repeat This instruction can be repeated.

Syntax	Description
dbl(*AR3-) = pop()	The content of the memory location pointed by the data stack pointer (SP) is copied to the 16 highest bits of the location addressed by AR3 and the content of the memory location pointed by SP + 1 is copied to the 16 lowest bits of the location addressed by AR3. Because this instruction is a long-operand instruction, AR3 is decremented by 2 after the execution. The SP is incremented by 2.

# **PSHBOTH**

Push Accumulator or Extended Auxiliary Register Content to Stack Pointers

### **Syntax Characteristics**

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[1]	pshboth(xsrc)			Yes	2	1	Х
Opcod	e			010	01 00	0E XSS	S 0101
Operands		xsrc					
Description  This instruction moves the lower 32 bits of a source register (XARx, XSP, XSSP, XDP, or a locations addressed by the data stack pointe (SSP). The return address register (RETA register (CFCT) are not accessed by this in process.		(DP, or XCDP k pointer (SP r (RETA) and	) to the ) and s I the c	two 16-k ystem sta control-flo	oit memory ack pointer ow context		
			xsrc(15–0) is moved to the (31–16) is moved to the			•	
		When xsrc is a SSP are filled	a 23-bit register, the $u_{\parallel}$ with 0.	oper 9 bits of	the loc	ation add	dressed by
Status	Bits	Affected by	none				
		Affects	none				
Repeat	t	This instruction	n can be repeated.				
See Also		See the follow	ng other related instru	ıctions:			
		□ Pop Accum	nulator or Extended Aux f Stack	iliary Register	Content	from Sta	ck Pointers

Push to Top of Stack

# **PSH**

# Push to Top of Stack

## **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	push(src1, src2)	Yes	2	1	Х
[2]	push(src)	Yes	2	1	X
[3]	push(src, Smem)	No	3	1	X
[4]	dbl(push(ACx))	Yes	2	1	X
[5]	push(Smem)	No	2	1	X
[6]	push(dbl(Lmem))	No	2	1	X

Description	These instructions move one or two operands to the data memory location addressed by the data stack pointer (SP). The return address register (RETA) and the control-flow context register (CFCT) are not accessed by this instruction even in the fast-return process. The operands may be:			
	<ul><li>an accumulator, auxiliary, or temporary register</li><li>a data memory location</li></ul>			
	The decrement operation performed on SP is done by the A-unit address generator dedicated to the stack addressing management.			
Status Bits	Affected by none			
	Affects none			
See Also	See the following other related instructions:			
	□ Pop Top of Stack			
	☐ Pop Accumulator or Extended Auxiliary Register Content from Stack Pointers			
	☐ Push Accumulator or Extended Auxiliary Register Content to Stack Pointers			

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	push(src1, src2)	Yes	2	1	Χ

0011 100E FSSS FDDD Opcode

Note: FSSS = src1, FDDD = src2

**Operands** src1, src2

**Description** This instruction decrements SP by 2, then moves the content of the source

> register src1 to the 16-bit data memory location pointed by SP and moves the content of the source register src2 to the 16-bit data memory location pointed

by SP + 1.

When the source register, src1 or src2, is an accumulator, the source accumulator low part, ACx(15-0), is moved to the 16-bit data memory

operand.

**Status Bits** Affected by none

> Affects none

Repeat This instruction can be repeated.

Syntax	Description
	The data stack pointer (SP) is decremented by 2. The content of AR0 is copied to the memory location pointed by SP and the content of AC1(15–0) is copied to the memory location pointed by SP + 1.

Before				After			
AR0			0300	AR0			0300
AC1	03	5644	F800	AC1	03	5644	F800
SP			0300	SP			02FE
2FE			0000	2FE			0300
2FF			0000	2FF			F800
300			5890	300			5890

### **Syntax Characteristics**

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[2]	push(src)			Yes	2	1	Х
Opcod	e			010	01 00	0E FSS	SS x110
Operar	nds	src					
Descri	ption	register (src) source regist	n decrements SP by to the 16-bit data me er is an accumula moved to the 16-bit o	emory location tor, the source	pointe	d by SP. umulator	When the
Status	Bits	Affected by	none				
		Affects	none				
Repeat	t		n cannot be repeated tion. It can be repeate				conditional
Examp	le						

Syntax	Description
push(AC0)	The data stack pointer (SP) is decremented by 1. The content of AC0(15–0) is copied
	to the memory location pointed by SP.

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	push(src, Smem)	No	3	1	Χ

1110 0100 AAAA AAAI FSSS x0xx Opcode

**Operands** Smem, src

**Description** This instruction decrements SP by 2, then moves the content of the source

> register (src) to the 16-bit data memory location pointed by SP and moves the content of the data memory (Smem) location to the 16-bit data memory

location pointed by SP + 1.

When the source register is an accumulator, the source accumulator low part,

ACx(15–0), is moved to the 16-bit data memory operand.

**Status Bits** Affected by none

> Affects none

Repeat This instruction can be repeated.

Syntax	Description
push(AC0, *AR3)	The data stack pointer (SP) is decremented by 2. The content of AC0(15–0) is copied to the memory location pointed by SP and the content addressed by AR3 is copied to the memory location pointed by SP + 1.

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	dbl(push(ACx))	Yes	2	1	Х

0101 000E xxSS x111 Opcode

**Operands AC**x

This instruction decrements SP by 2, then moves the content of the **Description** 

> accumulator high part ACx(31-16) to the 16-bit data memory location pointed by SP and moves the content of the accumulator low part ACx(15-0) to the

16-bit data memory location pointed by SP + 1.

**Status Bits** Affected by none

> Affects none

Repeat This instruction cannot be repeated with a single conditional or unconditional

repeat instruction. It can be repeated in other repeat instructions.

Syntax	Description
dbl(push(AC0))	The data stack pointer (SP) is decremented by 2. The content of AC0(31–16) is copied to the memory location pointed by SP and the content of AC0(15–0) is copied to the memory location pointed by SP + 1.

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	push(Smem)	No	2	1	Х

| 1011 | 0101 | AAAA AAAI Opcode

**Operands** Smem

Description This instruction decrements SP by 1, then moves the content of the data

memory (Smem) location to the 16-bit data memory location pointed by SP.

**Status Bits** Affected by none

> Affects none

Repeat This instruction can be repeated.

Syntax	Description
push(*AR1)	The data stack pointer (SP) decremented by 1. The content addressed by AR1 is copied to the memory location pointed by SP.

Before		After	
*AR1	6903	*AR1	6903
SP	0305	SP	0304
304	0000	304	6903
305	0300	305	0300

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	push(dbl(Lmem))	No	2	1	Х

1011 0111 AAAA AAAI **Opcode** 

**Operands** Lmem

**Description** This instruction decrements SP by 2, then moves the 16 highest bits of data

memory location Lmem to the 16-bit data memory location pointed by SP and moves the 16 lowest bits of data memory location Lmem to the 16-bit data

memory location pointed by SP + 1.

When Lmem is at an even address, the two 16-bit values pushed onto the stack are stored at memory location Lmem in the same order. When Lmem is at an odd address, the two 16-bit values pushed onto the stack are stored at

memory location Lmem in the reverse order.

**Status Bits** Affected by none

> Affects none

Repeat This instruction can be repeated.

Syntax	Description
push(dbl(*AR3-))	The data stack pointer (SP) is decremented by 2. The 16 highest bits of the content at the location addressed by AR3 are copied to the memory location pointed by SP and the 16 lowest bits of the content at the location addressed by AR3 are copied to the memory location pointed by SP + 1. Because this instruction is a long-operand instruction, AR3 is decremented by 2 after the execution.

### **RPTB**

### Repeat Block of Instructions Unconditionally

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	localrepeat{}	Yes	2	1	AD
[2]	blockrepeat{}	Yes	3	1	AD

1,1	local epearty					103	_	ı	710
[2]	blockrepeat{}					Yes	3	1	AD
Description		These instructions repeat a block of instructions the number of times specified by:							
			The content	of BRC0 +	1, if no loop	has alrea	ıdy beer	n detecte	d.
			The conten detected.	t of BRS1 -	+ 1, if one	level of the	he loop	has alre	eady been
			op structures aracteristics:	s defined b	y these in	structions	must h	nave the	following
			The minimur	m number of	instructions	executed	within or	ne loop ite	eration is 2.
			The minimu	m number o	of cycles exe	ecuted with	nin one	loop itera	ation is 2.
			Since the reinstruction of same iteration is	cycles from to on or the ne	the end of t	the loop is	uncerta	ain (effec	tive in the
			The block-r			-		-	ching to a
			C54CM bit i	n ST1_55 c	annot be m	odified wit	hin a blo	ock-repea	at loop.
		Th	ese instructio	ns cannot b	e repeated.				
			e section 1.5 echanism.	for a list of i	nstructions	that canno	ot be use	ed in a re	peat block
Status	Bits	Aff	ected by	none					
		Aff	ects	none					
See Als	80	Se	e the followin	g other relat	ted instructi	ons:			
			Repeat Sing	gle Instructio	on Condition	nally			
			Repeat Sing	gle Instructio	on Uncondit	ionally			
			Repeat Sing	gle Instruction	n Uncondition	onally and	Decrem	ent CSR	
			Repeat Sing	gle Instructio	on Uncondit	ionally and	d Increm	ent CSR	!

### Repeat Block of Instructions Unconditionally

### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	localrepeat{}	Yes	2	1	AD

#### **Opcode**

0100 101E 1111 1111

### **Operands**

none

#### Description

This instruction repeats a block of instructions the number of times specified by:

- ☐ The content of BRC0 + 1, if no loop has already been detected. In this case:
  - In the address phase of the pipeline, RSA0 is loaded with the program address of the first instruction of the loop.
  - The program address of the last instruction of the loop (that may be two parallel instructions) is computed in the address phase of the pipeline and stored in REA0.
  - BRC0 is decremented at the decode phase of the last instruction of the loop when its content is not equal to 0.
  - BRC0 contains 0 after the block-repeat operation has ended.
- ☐ The content of BRS1 + 1, if one level of the loop has already been detected. In this case:
  - BRC1 is loaded with the content of BRS1 in the address phase of the repeat block instruction.
  - In the address phase of the pipeline, RSA1 is loaded with the program address of the first instruction of the loop.
  - The program address of the last instruction of the loop (that may be two parallel instructions) is computed in the address phase of the pipeline and stored in REA1.
  - BRC1 is decremented at the decode phase of the last instruction of the loop when its content is not equal to 0.
  - BRC1 contains 0 after the block-repeat operation has ended.
  - BRS1 content is not impacted by the block-repeat operation.

	op structures d aracteristics:	lefined	by	this	instruction	must	have	the	following
	The minimum no	umber o	f ins	tructio	ons executed	within	one lo	op ite	ration is 2.
	The minimum number of cycles executed within one loop iteration is 2.								
	The maximum loop size is 128 bytes.								
	Since the result of updating BRCx (and BRAF in C54CM = 1) within 3 instruction cycles from the end of the loop is uncertain (effective in the same iteration or the next iteration depending on the pipeline state), this operation is prohibited.								
	C54CM bit in ST1_55 cannot be modified within a block-repeat loop.								
	The following instructions cannot be used as the last instruction in the loop structure:							in the loop	
wh	ile (cond && (RPT	TC < k8))	rep	eat	repeat(k8)	repe	eat(CSF	₹), CS	SR += k4
if (	cond) execute(AD	_Unit)			repeat(k16)	repe	repeat(CSR), CSR += TA		
if (	cond) execute(D_	Unit)			repeat(CSR	) repe	eat(CSF	₹), CS	SR −= k4
Ins us wit	ete:  Structions if (conced as the last instruction the the instruction struction_execute	struction with wh	in th	he loo it is p	op structure i	f the in	struction	on is	executed
	ocal loop is define m within the instr					loop is	s repea	itedly	executed
	All the code of the local loop must fit within the 128-byte, 4-byte-aligned IBQ; therefore, local repeat blocks are limited to 128 bytes minus the 0 to 3 bytes of first-instruction misalignment. The 128th byte of the IBQ can only occur in a paralleled instruction. See Figure 5–2 for legal uses of the local repeat instruction.								
	The following in	struction	ns ca	annot	be used in a	ny forr	m in a l	ocal l	oop code:
	blockrepeat	call			goto				
	idle	intr			reset				
	return	trap							
	Nested local repeat block instructions are allowed.								

	branch con address po at a higher a conditionall	ranch instructions allowed in a localrepeat structure are the aditionally instructions (if (cond) goto) with a target branch inting to an instruction included within the loop code and being address than the branching instruction. In this case, the branch y instruction is executed in 3 cycles and the condition is a the read phase of the pipeline (there is a 1-cycle latency or setting).		
Co	mpatibility	with C54x devices (C54CM = 1)		
Wh	en C54CM =	=1:		
	This instruction only uses block-repeat level 0; block-repeat level 1 is disabled.			
	The block-repeat active flag (BRAF) is set to 1. BRAF is cleared to 0 at the end of the block-repeat operation when BRC0 contains 0.			
	You can sto	op an active block-repeat operation by clearing BRAF to 0.		
	block-repear context savunlike C54x the stack. Y	at control registers for level 1 are not used. Nested at operations are supported using the C54x convention with ve/restore and BRAF. When an interrupt is acknowledged, a device, BRAF is captured into CFCT register, and saved to ou can use a block/local loop instruction in an interrupt without BRAF (while preserving BRC0, RSA0 and REA0).		
		utomatically cleared to 0 when a far branch (FB) or far call struction is executed.		
Affe	ected by	none		
Affe	ects	none		

This instruction cannot be repeated.

**Status Bits** 

Repeat

### **Example**

Syntax	Description
localrepeat	A block of instructions is repeated as defined by the content of BRC0 + 1.

	Address	BRC0	RSA0	REA0	BRS1
BRC0 = #3		0003	0000	0000	0000
localrepeat {	004003	?*	4005	400D	?
	004005	?	?	?	?
	00400D	DTZ**	?	?	?
}		0000	4005	400D	0000
*?: Unchanged  **DTZ: Decrease till zero					

Figure 5–2. Legal Uses of Repeat Block of Instructions Unconditionally (localrepeat) Instruction

### (a) 128-Byte Unaligned Loop-Legal Use

```
; no alignment directive
localrepeat {
                1st instruction
                                               } 128-byte loop body
                Last instruction
          }
next instruction
```

The entire localrepeat block and the next instruction reside in the IBQ, this code is accepted by the assembler.

# Figure 5–2. Legal Uses of Repeat Block of Instructions Unconditionally (localrepeat) Instruction (Continued)

(b) 129-Byte Unaligned Loop with Single Instruction at End of Loop—Illegal Use

```
interpretation incomparation in the control in the control incomparation ```

The localrepeat instruction is not aligned; the *next instruction* may not be fetched in the IBQ. Because the last instruction of the localrepeat block is a nonparalleled (single) instruction, the CPU must confirm that the *next instruction* does not have a parallel enable bit; therefore, this code is rejected by the assembler.

(c) 129-Byte Unaligned Loop with Paralleled Instruction at End of Loop—Legal Use

The localrepeat instruction is not aligned; the *next instruction* may not be fetched in the IBQ. Because the last instruction of the localrepeat block is a paralleled instruction, the CPU does not need to confirm that the *next instruction* does not have a parallel enable bit; therefore, this code is accepted by the assembler.

# Figure 5–2. Legal Uses of Repeat Block of Instructions Unconditionally (localrepeat) Instruction (Continued)

(d) 129-Byte Aligned Loop with Single Instruction at End of Loop-Legal Use

The localrepeat instruction is aligned, so the entire localrepeat block and the *next instruction* reside in the IBQ. Because the *next instruction* is in the IBQ, the CPU can confirm that the *next instruction* does not have a parallel enable bit; therefore, this code is accepted by the assembler.

### (e) 130-Byte Unaligned Loop—Illegal Use

The localrepeat instruction is not aligned; the entire localrepeat block may not reside in the IBQ. Because the last instruction of the localrepeat block may not reside in the IBQ, this code is rejected by the assembler.

# Figure 5–2. Legal Uses of Repeat Block of Instructions Unconditionally (localrepeat) Instruction (Continued)

(f) 130-Byte Aligned Loop with Single Instruction at End of Loop—Legal Use

```
align 4 ; alignment directive

nop_16||nop ; 3-byte instruction

localrepeat {

1st instruction

...... } 130-byte loop body

Last instruction

(nonparalleled = single)

}

next instruction
......
```

The nop instructions are aligned so the localrepeat instruction, the entire localrepeat block, and the *next instruction* reside in the IBQ. Because the *next instruction* is in the IBQ, the CPU can confirm that the *next instruction* does not have a parallel enable bit; therefore, this code is accepted by the assembler.

(g) 132-Byte Aligned Loop with Paralleled Instruction at End of Loop-Legal Use

The nop instruction is aligned, so the localrepeat instruction and the entire localrepeat block reside in the IBQ; the *next instruction* is not fetched in the IBQ. Because the last instruction of the localrepeat block is a paralleled instruction, the CPU does not need to confirm that the *next instruction* does not have a parallel enable bit; therefore, this code is accepted by the assembler.

### Repeat Block of Instructions Unconditionally

### **Syntax Characteristics**

| No. | Syntax        | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|---------------|------------------------|------|--------|----------|
| [2] | blockrepeat{} | Yes                    | 3    | 1      | AD       |

#### **Opcode**

0000 111E | 1111 1111 | 1111 1111

#### **Operands**

none

#### **Description**

This instruction repeats a block of instructions the number of times specified by:

- the content of BRC0 + 1, if no loop has already been detected. In this case:
  - In the address phase of the pipeline, RSA0 is loaded with the program address of the first instruction of the loop.
  - The program address of the last instruction of the loop (that may be two parallel instructions) is computed in the address phase of the pipeline and stored in REA0.
  - BRC0 is decremented at the decode phase of the last instruction of the loop when its content is not equal to 0.
  - BRC0 contains 0 after the block-repeat operation has ended.
- the content of BRS1 + 1, if one level of the loop has already been detected. In this case:
  - BRC1 is loaded with the content of BRS1 in the address phase of the repeat block instruction.
  - In the address phase of the pipeline, RSA1 is loaded with the program address of the first instruction of the loop.
  - The program address of the last instruction of the loop (that may be two parallel instructions) is computed in the address phase of the pipeline and stored in REA1.
  - BRC1 is decremented at the decode phase of the last instruction of the loop when its content is not equal to 0.
  - BRC1 contains 0 after the block-repeat operation has ended.
  - BRS1 content is not impacted by the block-repeat operation.

|      | op structures defined by these tracteristics:                                                                                                                                                                                                      | instructions   | must have the following         |  |  |  |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|---------------------------------|--|--|--|
|      | The minimum number of instruction                                                                                                                                                                                                                  | ns executed w  | rithin one loop iteration is 2. |  |  |  |
|      | The minimum number of cycles of                                                                                                                                                                                                                    | executed with  | in one loop iteration is 2.     |  |  |  |
|      | The maximum loop size is 64K bytes.                                                                                                                                                                                                                |                |                                 |  |  |  |
|      | The block-repeat operation can only be cleared by branching to a destination address outside the active block-repeat loop.                                                                                                                         |                |                                 |  |  |  |
|      | Since the result of updating BRCx (and BRAF in C54CM = 1) within 3 instruction cycles from the end of the loop is uncertain (effective in the same iteration or the next iteration depending on the pipeline state), this operation is prohibited. |                |                                 |  |  |  |
|      | C54CM bit in ST1_55 cannot be modified within a block-repeat loop.                                                                                                                                                                                 |                |                                 |  |  |  |
|      | The following instructions cannot be used as the last instruction in the loop structure:                                                                                                                                                           |                |                                 |  |  |  |
| wh   | while (cond && (RPTC < k8)) repeat repeat(k8) repeat(CSR), CSR += k4                                                                                                                                                                               |                |                                 |  |  |  |
| if ( | cond) execute(AD_Unit)                                                                                                                                                                                                                             | repeat(k16)    | repeat(CSR), CSR += TAx         |  |  |  |
| if ( | cond) execute(D_Unit)                                                                                                                                                                                                                              | repeat(CSR)    | repeat(CSR), CSR -= k4          |  |  |  |
|      | See section 1.5 for a list of insblock-repeat loop code.                                                                                                                                                                                           | structions tha | t cannot be used in the         |  |  |  |
| Со   | mpatibility with C54x devices (                                                                                                                                                                                                                    | C54CM = 1)     |                                 |  |  |  |
| Wh   | en C54CM =1:                                                                                                                                                                                                                                       |                |                                 |  |  |  |
|      | This instruction only uses block-repeat level 0; block-repeat level 1 is disabled.                                                                                                                                                                 |                |                                 |  |  |  |
|      | The block-repeat active flag (BRAF) is set to 1. BRAF is cleared to 0 at the end of the block-repeat operation when BRC0 contains 0.                                                                                                               |                |                                 |  |  |  |
|      | You can stop an active block-rep                                                                                                                                                                                                                   | eat operation  | by clearing BRAF to 0.          |  |  |  |
|      | Block-repeat control registers for level 1 are not used. Nested block-repeat operations are supported using the C54x convention with context save/restore and BRAF. The control-flow context register (CFCT) values are not used.                  |                |                                 |  |  |  |
|      | BRAF is automatically cleared to 0 when a far branch (FB) or far call (FCALL) instruction is executed.                                                                                                                                             |                |                                 |  |  |  |

Affected by **Status Bits** none

> Affects none

Repeat This instruction cannot be repeated.

| Syntax      | Description                                                                                                                                                                                           |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| blockrepeat | A block of instructions is repeated as defined by the content of BRC0 + 1. A second loop of instructions is repeated as defined by the content of BRS1 + 1 (BRC1 is loaded with the content of BRS1). |

|                 | Address   | BRC0  | RSA0 | REA0 | BRS1 | BRC1   | RSA1 | REA1 |
|-----------------|-----------|-------|------|------|------|--------|------|------|
| BRC0 = #3       |           | 0003  | 0000 | 0000 | 0000 | 0000   | 0000 | 0000 |
| BRC1 = #1       |           | ?*    | ?    | ?    | 0001 | 0001   | ?    | ?    |
| blockrepeat {   | 004006    | ?     | 4009 | 4017 | ?    | ?      | ?    | ?    |
|                 | 004009    | ?     | ?    | ?    | ?    | ?      | ?    | ?    |
| localrepeat {   | 00400B    | ?     | ?    | ?    | ?    | (BRS1) | 400D | 4015 |
|                 | 00400D    | ?     | ?    | ?    | ?    | ?      | ?    | ?    |
|                 | 004015    | ?     | ?    | ?    | ?    | DTZ**  | ?    | ?    |
| }               |           |       |      |      |      |        |      |      |
|                 | 004017    | DTZ** | ?    | ?    | ?    | ?      | ?    | ?    |
| }               |           | 0000  | 4009 | 4017 | 0001 | 0000   | 400D | 4015 |
| *?: Unchanged   |           |       |      |      |      |        |      |      |
| **DTZ: Decrease | till zero |       |      |      |      |        |      |      |

### **RPTCC**

### Repeat Single Instruction Conditionally

### **Syntax Characteristics**

| No.    | Syntax         |                                                                                                                                 |                                                                                                                                                                                                     | Parallel<br>Enable Bit                             | Size                          | Cycles                              | Pipeline                                |  |  |
|--------|----------------|---------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------|-------------------------------|-------------------------------------|-----------------------------------------|--|--|
| [1]    | while (cond && | (RP                                                                                                                             | TC < k8)) repeat                                                                                                                                                                                    | Yes                                                | 3                             | 1                                   | AD                                      |  |  |
| Opcod  | e              |                                                                                                                                 | 0000                                                                                                                                                                                                | 0 000E xC                                          | CC C                          | CCC kkk                             | k kkkk                                  |  |  |
| Opera  | nds            | cor                                                                                                                             | nd, k8                                                                                                                                                                                              | ·                                                  |                               | ·                                   |                                         |  |  |
| Descri | ption          | lon<br>ins<br>val                                                                                                               | s instruction evaluates a single or<br>g as the condition is true, the ne<br>tructions is repeated the number<br>ue, k8 + 1. The maximum number<br>calleled instructions is 2 <sup>8</sup> –1 (255) | ext instruction<br>of times spec<br>er of executio | or the<br>ified by<br>ns of a | next two<br>y an 8-bit<br>given ins | paralleled<br>immediate<br>struction or |  |  |
|        |                | The 8 LSBs of the repeat counter register (RPTC):                                                                               |                                                                                                                                                                                                     |                                                    |                               |                                     |                                         |  |  |
|        |                | ☐ Are loaded with the immediate value at the address phase of the pipeline                                                      |                                                                                                                                                                                                     |                                                    |                               |                                     |                                         |  |  |
|        |                |                                                                                                                                 | Are decremented by 1 in the decode phase of the repeated instruction.                                                                                                                               |                                                    |                               |                                     |                                         |  |  |
|        |                | Must not be accessed when it is being decremented in the repeat<br>mechanism or in parallel with the repeat instruction itself. |                                                                                                                                                                                                     |                                                    |                               |                                     |                                         |  |  |
|        |                | The                                                                                                                             | e 8 MSBs of RPTC:                                                                                                                                                                                   |                                                    |                               |                                     |                                         |  |  |
|        |                |                                                                                                                                 | Are loaded with the cond code a                                                                                                                                                                     | at the address                                     | phase                         | e of the pip                        | peline.                                 |  |  |
|        |                |                                                                                                                                 | Are untouched during the while/                                                                                                                                                                     | repeat structu                                     | ıre exe                       | ecution.                            |                                         |  |  |
|        |                | in t                                                                                                                            | each step of the iteration, the cor<br>he execute phase of the pipeline<br>truction repetition stops.                                                                                               |                                                    | •                             |                                     |                                         |  |  |
|        |                |                                                                                                                                 | If the condition becomes false at                                                                                                                                                                   | any execution                                      | of the                        | repeated                            | instruction                             |  |  |

were not performed.

the 8 LSBs of RPTC are corrected to indicate exactly how many iterations

□ Since the condition is evaluated in the execute phase of the repeated instruction, when the condition is tested false, some of the succeeding iterations of that repeated instruction may have gone through the address, access, and read phases of the pipeline. Therefore, they may have modified the pointer registers used in the DAGEN units to generate data

memory operands addresses in the address phase.

When the while/repeat structure is exited, reading the computed single-repeat register (CSR) content enables you to determine how many instructions have gone through the address phase of the pipeline. You may then use the Repeat Single Instruction Unconditionally instruction [3] to rewind the pointer registers. Note that this must only be performed when a false condition has been met inside the while/repeat structure.

☐ The following table provides the 8 LSBs of RPTC and CSR once the while/repeat structure is exited.

| If the condition is not met  | RPTC[7:0] content after exiting loop | CSR content after<br>exiting loop |
|------------------------------|--------------------------------------|-----------------------------------|
| At 1 <sup>st</sup> iteration | RPTCinit + 1                         | 4                                 |
| At 2 <sup>nd</sup> iteration | RPTCinit                             | 4                                 |
| At 3 <sup>rd</sup> iteration | RPTC – 1                             | 4                                 |
|                              |                                      |                                   |
| At RPTCinit – 2 iteration    | 4                                    | 3                                 |
| At RPTCinit – 1 iteration    | 3                                    | 2                                 |
| At RPTCinit iteration        | 2                                    | 1                                 |
| At RPTCinit + 1 iteration    | 1                                    | 0                                 |
| Never                        | 0                                    | 0                                 |

RPTCinit is the number of requested iterations minus 1.

The repeat single mechanism triggered by this instruction is interruptible. Saving and restoring the RPTC content in ISRs enables you to preserve the while/repeat structure context.

Instead of programming a number of iterations (minus 1) equal to 0, it is recommended that you use the conditional execute() structure.

This instruction cannot be used as the last or the second to last instruction in a repeat loop structure.

See section 1.5 for a list of instructions that cannot be used in a repeat single mechanism.

In addition, any store-to-memory instruction including push instructions cannot be used in a conditional repeat single mechanism.

### Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, the comparison of accumulators to 0 is performed as if M40 was set to 1.

Status Bits

Affected by ACOVx, CARRY, C54CM, M40, TCx

Affects ACOVx

Repeat This instruction cannot be repeated.

See Also See the following other related instructions:

Repeat Block of Instructions Unconditionally

Repeat Single Instruction Unconditionally and Decrement CSR

Repeat Single Instruction Unconditionally and Increment CSR

| Syntax                                 | Description                                                                                                                                                                                                                                                                                                    |
|----------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| while (AC1 > #0 && (RPTC < #7)) repeat | As long as the content of AC1 is greater than 0 and the repeat counter is not equal to 0, the next single instruction is repeated as defined by the unsigned 8-bit value (7) + 1. At the address phase of the pipeline, RPTC is automatically initialized to 4107h and then is immediately decreased to 4106h. |

| while (AC1 > #0 && (RPTC < #7)) repeat | address: | 004004 |
|----------------------------------------|----------|--------|
| AC1 = AC1 - (T0 * *AR1)                |          | 004008 |
|                                        |          | 00400B |

| Before |              | After          |     |
|--------|--------------|----------------|-----|
| AC1    | 00 2359 0340 | AC1 00 1FC2 7B | 340 |
| TO     | 0340         | T0 03          | 340 |
| *AR1   | 2354         | *AR1 23        | 54  |
| RPTC   | 4106†        | RPTC 00        | 000 |

<sup>&</sup>lt;sup>†</sup> At the address phase of the pipeline, RPTC is automatically initialized to 4107h and then is immediately decreased to 4106h.

### **RPT**

### Repeat Single Instruction Unconditionally

### **Syntax Characteristics**

| No. | Syntax      | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|-------------|------------------------|------|--------|----------|
| [1] | repeat(k8)  | Yes                    | 2    | 1      | AD       |
| [2] | repeat(k16) | Yes                    | 3    | 1      | AD       |
| [3] | repeat(CSR) | Yes                    | 2    | 1      | AD       |

### Description

This instruction repeats the next instruction or the next two paralleled instructions the number of times specified by the content of the computed single repeat register (CSR) + 1 or an immediate value, kx + 1. This value is loaded into the repeat counter register (RPTC). The maximum number of executions of a given instruction or paralleled instructions is  $2^{16} - 1$  (65535).

The repeat single mechanism triggered by these instructions is interruptible.

These instructions cannot be repeated.

These instructions cannot be used as the last instruction in a repeat loop structure.

Two paralleled instructions can be repeated when following the parallelism general rules.

See section 1.5 for a list of instructions that cannot be used in a repeat single mechanism.

### **Status Bits**

Affected by none

Affects none

#### See Also

See the following other related instructions:

- ☐ Repeat Block of Instructions Unconditionally
- ☐ Repeat Single Instruction Conditionally
- ☐ Repeat Single Instruction Unconditionally and Decrement CSR
- Repeat Single Instruction Unconditionally and Increment CSR

## Repeat Single Instruction Unconditionally

### **Syntax Characteristics**

| No.     | Syntax                                                                                                                                                        |                                                                                      |                                                | Parallel<br>Enable Bit | Size     | Cycles      | Pipeline    |
|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|------------------------------------------------|------------------------|----------|-------------|-------------|
| [1]     | repeat(k8)                                                                                                                                                    |                                                                                      |                                                | Yes                    | 2        | 1           | AD          |
| [2]     | repeat(k16)                                                                                                                                                   |                                                                                      |                                                | Yes                    | 3        | 1           | AD          |
| Opcod   | e                                                                                                                                                             | k8                                                                                   |                                                | 01                     | 00 1     | 10E   kkk   | k kkkk      |
|         |                                                                                                                                                               | k16                                                                                  | 000                                            | 00 110E kk             | kk kl    | kkk   kkk   | k kkkk      |
| Operar  | nds                                                                                                                                                           | kx                                                                                   |                                                |                        |          |             |             |
| Descri  | This instruction repeats the next instruction or the next to instructions the number of times specified by an immediate value repeat counter register (RPTC): |                                                                                      |                                                |                        |          |             | -           |
|         |                                                                                                                                                               | ☐ Is loaded \                                                                        | with the immediate v                           | alue in the add        | lress pl | hase of th  | e pipeline. |
|         |                                                                                                                                                               | ☐ Is decreme                                                                         | ented by 1 in the ded                          | code phase of          | the rep  | eated inst  | ruction.    |
|         |                                                                                                                                                               | ☐ Contains (                                                                         | at the end of the re                           | peat single me         | chanis   | m.          |             |
|         |                                                                                                                                                               | _                                                                                    | pe accessed when it<br>m or in parallel with t | •                      |          |             | peat single |
|         |                                                                                                                                                               | The repeat sin                                                                       | gle mechanism trigg                            | ered by this in        | structio | n is interr | uptible.    |
|         |                                                                                                                                                               | Two paralleled general rules.                                                        | d instructions can be                          | repeated whe           | en follo | wing the p  | oarallelism |
|         |                                                                                                                                                               | This instruction structure.                                                          | on cannot be used                              | as the last in         | structic | on in a re  | epeat loop  |
|         |                                                                                                                                                               | See section 1.5 for a list of instructions that cannot be used in a repearmechanism. |                                                |                        |          |             |             |
| Status  | Bits                                                                                                                                                          | Affected by                                                                          | none                                           |                        |          |             |             |
|         |                                                                                                                                                               | Affects                                                                              | none                                           |                        |          |             |             |
| Repeat  | t                                                                                                                                                             | This instruction                                                                     | n cannot be repeated                           | d.                     |          |             |             |
| SW/DU/O | 2005                                                                                                                                                          |                                                                                      |                                                | lantu e Com            | C++ D-   |             | F 400       |

### Example 1

| Syntax                    | Description                                                                     |  |  |  |  |  |
|---------------------------|---------------------------------------------------------------------------------|--|--|--|--|--|
| repeat(#3)                | The single instruction following the repeat instruction is repeated four times. |  |  |  |  |  |
| AC1 = AC1 + *AR3+ * *AR4+ |                                                                                 |  |  |  |  |  |

| Before |    |      |      | After |    |      |      |
|--------|----|------|------|-------|----|------|------|
| AC1    | 00 | 0000 | 0000 | AC1   | 00 | 3376 | AD10 |
| AR3    |    |      | 0200 | AR3   |    |      | 0204 |
| AR4    |    |      | 0400 | AR4   |    |      | 0404 |
| 200    |    |      | AC03 | 200   |    |      | AC03 |
| 201    |    |      | 3468 | 201   |    |      | 3468 |
| 202    |    |      | FE00 | 202   |    |      | FE00 |
| 203    |    |      | 23DC | 203   |    |      | 23DC |
| 400    |    |      | D768 | 400   |    |      | D768 |
| 401    |    |      | 6987 | 401   |    |      | 6987 |
| 402    |    |      | 3400 | 402   |    |      | 3400 |
| 403    |    |      | 7900 | 403   |    |      | 7900 |

| Syntax       | Description                                                                             |
|--------------|-----------------------------------------------------------------------------------------|
| repeat(#513) | A single instruction is repeated as defined by the unsigned 16-bit value + 1 (513 + 1). |

## Repeat Single Instruction Unconditionally

### **Syntax Characteristics**

| No.    | Syntax      |                                                                                                                                                                                                                          |         |             |                | Parallel<br>Enable Bi                          | t Size   | Cycles      | Pipeline    |  |
|--------|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|-------------|----------------|------------------------------------------------|----------|-------------|-------------|--|
| [3]    | repeat(CSR) |                                                                                                                                                                                                                          |         |             |                | Yes                                            | 2        | 1           | AD          |  |
| Opcod  | e           |                                                                                                                                                                                                                          |         |             |                | 0                                              | 100 1    | 00E   xxx   | xx x000     |  |
| Operar | nds         | none                                                                                                                                                                                                                     |         |             |                |                                                |          |             |             |  |
| Descri | ption       | This instruction repeats the next instruction or the next two paralleled instructions the number of times specified by the content of the computed single repeat register (CSR) + 1. The repeat counter register (RPTC): |         |             |                |                                                |          |             |             |  |
|        |             | ☐ Is loa                                                                                                                                                                                                                 | ded w   | vith CSR    | content in     | the address p                                  | hase of  | the pipeli  | ne.         |  |
|        |             | ☐ Is decremented by 1 in the decode phase of the repeated instruction.                                                                                                                                                   |         |             |                |                                                |          |             |             |  |
|        |             | ☐ Contains 0 at the end of the repeat single mechanism.                                                                                                                                                                  |         |             |                |                                                |          |             |             |  |
|        |             | Must not be accessed when it is being decremented in the repeat single<br>mechanism or in parallel with the repeat instruction itself.                                                                                   |         |             |                |                                                |          |             |             |  |
|        |             | The repea                                                                                                                                                                                                                | at sing | gle mech    | nanism trigg   | riggered by this instruction is interruptible. |          |             |             |  |
|        |             | Two paralleled instructions can be repeated when following the parallelism general rules.                                                                                                                                |         |             |                |                                                |          |             |             |  |
|        |             | This instruction cannot be used as the last instruction in a repeat loop structure.                                                                                                                                      |         |             |                |                                                |          |             |             |  |
|        |             | See section                                                                                                                                                                                                              |         | ofor a list | t of instructi | ions that canr                                 | ot be us | sed in a re | peat single |  |
| Status | Bits        | Affected b                                                                                                                                                                                                               | ру      | none        |                |                                                |          |             |             |  |
|        |             | Affects                                                                                                                                                                                                                  |         | none        |                |                                                |          |             |             |  |
| Repeat | t           | This instr                                                                                                                                                                                                               | uction  | cannot      | be repeate     | d.                                             |          |             |             |  |

| Syntax                    | Description                                                                    |
|---------------------------|--------------------------------------------------------------------------------|
| repeat(CSR)               | The single instruction following the repeat instruction is repeated as defined |
| AC1 = AC1 + *AR3+ * *AR4+ | by the content of CSR + 1.                                                     |

| Before |    |      |      | After |    |      |      |
|--------|----|------|------|-------|----|------|------|
| AC1    | 00 | 0000 | 0000 | AC1   | 00 | 3376 | AD10 |
| CSR    |    |      | 0003 | CSR   |    |      | 0003 |
| AR3    |    |      | 0200 | AR3   |    |      | 0204 |
| AR4    |    |      | 0400 | AR4   |    |      | 0404 |
| 200    |    |      | AC03 | 200   |    |      | AC03 |
| 201    |    |      | 3468 | 201   |    |      | 3468 |
| 202    |    |      | FE00 | 202   |    |      | FE00 |
| 203    |    |      | 23DC | 203   |    |      | 23DC |
| 400    |    |      | D768 | 400   |    |      | D768 |
| 401    |    |      | 6987 | 401   |    |      | 6987 |
| 402    |    |      | 3400 | 402   |    |      | 3400 |
| 403    |    |      | 7900 | 403   |    |      | 7900 |

### RPTSUB

### Repeat Single Instruction Unconditionally and Decrement CSR

### **Syntax Characteristics**

| No.         | Syntax         |                                 |                                                                                              | Parallel<br>Enable Bit | Size     | Cycles      | Pipeline    |
|-------------|----------------|---------------------------------|----------------------------------------------------------------------------------------------|------------------------|----------|-------------|-------------|
| [1]         | repeat(CSR), ( | CSR -= k4                       |                                                                                              | Yes                    | 2        | 1           | Х           |
| Opcod       | le             |                                 |                                                                                              | 01                     | 00 1     | 00E   kkk   | k x011      |
| Opera       | nds            | k4                              |                                                                                              |                        |          |             |             |
| Description |                | instructions the                | on repeats the next<br>e number of times sp<br>egister (CSR) + 1. Th                         | pecified by the        | e conte  | ent of the  | computed    |
|             |                | ☐ Is loaded \                   | with CSR content in th                                                                       | ne address ph          | ase of   | the pipeli  | ne.         |
|             |                | ☐ Is decreme                    | ented by 1 in the deco                                                                       | ode phase of t         | he rep   | eated inst  | truction.   |
|             |                | ☐ Contains 0                    | at the end of the rep                                                                        | eat single me          | chanis   | m.          |             |
|             |                |                                 | e accessed when it is<br>m or in parallel with th                                            | _                      |          |             | peat single |
|             |                | decremented by of the pipeline: | nit ALU, this instruct<br>by k4. The CSR modif<br>there is a 3-cycle lat<br>a address phase. | ication is perfo       | rmed     | in the exe  | cute phase  |
|             |                | The repeat sin                  | gle mechanism trigge                                                                         | ered by this ins       | structio | n is interr | uptible.    |
|             |                | Two paralleled general rules.   | I instructions can be                                                                        | repeated whe           | n follo  | wing the    | parallelism |
|             |                | This instruction structure.     | n cannot be used a                                                                           | as the last in         | structio | on in a re  | epeat loop  |
|             |                | See section 1.8 mechanism.      | 5 for a list of instructio                                                                   | ons that canno         | t be us  | ed in a re  | peat single |
| Status      | Bits           | Affected by                     | none                                                                                         |                        |          |             |             |
|             |                | Affects                         | none                                                                                         |                        |          |             |             |
| _           |                |                                 |                                                                                              |                        |          |             |             |

This instruction cannot be repeated.

Repeat

### RPTSUB Repeat Single Instruction Unconditionally and Decrement CSR (repeat)

| See Also | See the following other related instructions:                 |
|----------|---------------------------------------------------------------|
|          | ☐ Repeat Block of Instructions Unconditionally                |
|          | ☐ Repeat Single Instruction Conditionally                     |
|          | ☐ Repeat Single Instruction Unconditionally                   |
|          | ☐ Repeat Single Instruction Unconditionally and Increment CSR |

### Example

| Syntax                 | Description                                                                                                                               |
|------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| repeat(CSR), CSR -= #2 | A single instruction is repeated as defined by the content of CSR + 1. The content of CSR is decremented by the unsigned 4-bit value (2). |

Instruction Set Descriptions

### RPTADD

### Repeat Single Instruction Unconditionally and Increment CSR

### **Syntax Characteristics**

| No. | Syntax                  | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|-------------------------|------------------------|------|--------|----------|
| [1] | repeat(CSR), CSR += TAx | Yes                    | 2    | 1      | Х        |
| [2] | repeat(CSR), CSR -= k4  | Yes                    | 2    | 1      | Х        |

### **Description**

These instructions repeat the next instruction or the next two paralleled instructions the number of times specified by the content of the computed single repeat register (CSR) + 1. This value is loaded into the repeat counter register (RPTC). The maximum number of executions of a given instruction or paralleled instructions is  $2^{16} - 1$  (65535).

With the A-unit ALU, these instructions allow the content of CSR to be incremented. The CSR modification is performed in the execute phase of the pipeline; there is a 3-cycle latency between the CSR modification and its usage in the address phase.

The repeat single mechanism triggered by these instructions is interruptible.

Two paralleled instructions can be repeated when following the parallelism general rules.

These instructions cannot be repeated.

These instructions cannot be used as the last instruction in a repeat loop structure.

See section 1.5 for a list of instructions that cannot be used in a repeat single mechanism.

### **Status Bits**

Affected by none

Affects

#### See Also

See the following other related instructions:

none

- ☐ Repeat Block of Instructions Unconditionally
- ☐ Repeat Single Instruction Conditionally
- Repeat Single Instruction Unconditionally
- ☐ Repeat Single Instruction Unconditionally and Decrement CSR

### Repeat Single Instruction Unconditionally and Increment CSR

### **Syntax Characteristics**

| No. | Syntax                  | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|-------------------------|------------------------|------|--------|----------|
| [1] | repeat(CSR), CSR += TAx | Yes                    | 2    | 1      | Х        |

**Opcode** 

0100 100E FSSS x001

**Operands** 

TAx

**Description** 

This instruction repeats the next instruction or the next two paralleled instructions the number of times specified by the content of the computed single repeat register (CSR) + 1. The repeat counter register (RPTC):

- ☐ Is loaded with CSR content in the address phase of the pipeline.
- ☐ Is decremented by 1 in the decode phase of the repeated instruction.
- ☐ Contains 0 at the end of the repeat single mechanism.
- ☐ Must not be accessed when it is being decremented in the repeat single mechanism or in parallel with the repeat instruction itself.

With the A-unit ALU, this instruction allows the content of CSR to be incremented by the content of TAx. The CSR modification is performed in the execute phase of the pipeline; there is a 3-cycle latency between the CSR modification and its usage in the address phase.

The repeat single mechanism triggered by this instruction is interruptible.

Two paralleled instructions can be repeated when following the parallelism general rules.

This instruction cannot be used as the last instruction in a repeat loop structure.

See section 1.5 for a list of instructions that cannot be used in a repeat single mechanism.

**Status Bits** 

Affected by none

Affects

none

Repeat

This instruction cannot be repeated.

| Syntax | Description                                                                                                                                       |
|--------|---------------------------------------------------------------------------------------------------------------------------------------------------|
|        | A single instruction is repeated as defined by the content of CSR + 1. The content of CSR is incremented by the content of temporary register T1. |

### Repeat Single Instruction Unconditionally and Increment CSR

### **Syntax Characteristics**

| No. | Syntax                 | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|------------------------|------------------------|------|--------|----------|
| [2] | repeat(CSR), CSR += k4 | Yes                    | 2    | 1      | Χ        |

### Opcode

0100 100E kkkk x010

### **Operands**

k4

### Description

This instruction repeats the next instruction or the next two paralleled instructions the number of times specified by the content of the computed single repeat register (CSR) + 1. The repeat counter register (RPTC):

- ☐ Is loaded with CSR content in the address phase of the pipeline.
- ☐ Is decremented by 1 in the decode phase of the repeated instruction.
- Contains 0 at the end of the repeat single mechanism.
- ☐ Must not be accessed when it is being decremented in the repeat single mechanism or in parallel with the repeat instruction itself.

With the A-unit ALU, this instruction allows the content of CSR to be incremented by k4. The CSR modification is performed in the execute phase of the pipeline; there is a 3-cycle latency between the CSR modification and its usage in the address phase.

The repeat single mechanism triggered by this instruction is interruptible.

Two paralleled instructions can be repeated when following the parallelism general rules.

This instruction cannot be used as the last instruction in a repeat loop structure.

See section 1.5 for a list of instructions that cannot be used in a repeat single mechanism.

### **Status Bits**

Affected by none

Affects none

### Repeat

This instruction cannot be repeated.

| Syntax                 | Description                                                                                                                               |
|------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| repeat(CSR), CSR += #2 | A single instruction is repeated as defined by the content of CSR + 1. The content of CSR is incremented by the unsigned 4-bit value (2). |

### **RETCC**

### Return Conditionally

### **Syntax Characteristics**

| No. | Syntax           | Parallel<br>Enable Bit | Size | Cycles <sup>†</sup> | Pipeline |
|-----|------------------|------------------------|------|---------------------|----------|
| [1] | if (cond) return | Yes                    | 3    | 5/5                 | R        |

<sup>†</sup> x/y cycles: x cycles = condition true, y cycles = condition false

#### **Opcode**

0000 001E xCCC CCCC xxxx xxxx

#### **Operands**

cond

#### **Description**

This instructions evaluates a single condition defined by the cond field in the read phase of the pipeline. If the condition is true, a return occurs to the return address of the calling subroutine. There is a 1-cycle latency on the condition setting. A single condition can be tested as determined by the cond field of the instruction. See Table 1–3 for a list of conditions.

After returning from a called subroutine, the CPU restores the value of two internal registers: the program counter (PC) and a loop context register. The CPU uses these values to re-establish the context of the program sequence.

In the slow-return process (default), the return address (from the PC) and the loop context bits are restored from the stacks (in memory). When the CPU returns from a subroutine, the speed at which these values are restored is dependent on the speed of the memory accesses.

In the fast-return process, the return address (from the PC) and the loop context bits are restored from the return address register (RETA) and the control-flow context register (CFCT). You can read from or write to RETA and CFCT as a pair with dedicated, 32-bit load and store instructions. For fast-return mode operation, see the *TMS320C55x DSP CPU Reference Guide* (SPRU371).

When a return from a subroutine occurs:

- ☐ The loop context bits concatenated with the 8 MSBs of the return address are popped from the top of the system stack pointer (SSP). The SSP is incremented by 1 word in the read phase of the pipeline.
- □ The 16 LSBs of the return address are popped from the top of the data stack pointer (SP). The SP is incremented by 1 word in the read phase of the pipeline.

#### 

### Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, the comparison of accumulators to 0 is performed as if M40 was set to 1.

Status Bits Affected by ACOVx, CARRY, C54CM, M40, TCx

Affects ACOVx

**Repeat** This instruction cannot be repeated.

**See Also** See the following other related instructions:

- Call Conditionally
- Call Unconditionally
- ☐ Return from Interrupt
- ☐ Return Unconditionally

| Syntax                 | Description                                                                                                               |
|------------------------|---------------------------------------------------------------------------------------------------------------------------|
| if (ACOV0 = #0) return | The AC0 overflow bit is equal to 0, the program counter (PC) is loaded with the return address of the calling subroutine. |

| Before |   | After |                  |
|--------|---|-------|------------------|
| ACOV0  | 0 | ACOV0 | 0                |
| PC     |   | PC    | (return address) |
| SP     |   | SP    |                  |

**RET** 

Return Unconditionally

### **Syntax Characteristics**

| No. | Syntax | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|--------|------------------------|------|--------|----------|
| [1] | return | Yes                    | 2    | 5      | D        |

**Opcode** 

0100 100E xxxx x100

**Operands** 

none

Description

This instruction passes control back to the calling subroutine.

After returning from a called subroutine, the CPU restores the value of two internal registers: the program counter (PC) and a loop context register. The CPU uses these values to re-establish the context of the program sequence.

In the slow-return process (default), the return address (from the PC) and the loop context bits are restored from the stacks (in memory). When the CPU returns from a subroutine, the speed at which these values are restored is dependent on the speed of the memory accesses.

In the fast-return process, the return address (from the PC) and the loop context bits are restored from the return address register (RETA) and the control-flow context register (CFCT). You can read from or write to RETA and CFCT as a pair with dedicated, 32-bit load and store instructions. For fast-return mode operation, see the *TMS320C55x DSP CPU Reference Guide* (SPRU371).

- ☐ The loop context bits concatenated with the 8 MSBs of the return address are popped from the top of the system stack pointer (SSP). The SSP is incremented by 1 word in the address phase of the pipeline.
- ☐ The 16 LSBs of the return address are popped from the top of the data stack pointer (SP). The SP is incremented by 1 word in the address phase of the pipeline.

After  $\rightarrow$  SSP = x + 1 Return

(Loop bits):PC(23–16)

Previously stored data

 $\begin{array}{ccc} \textbf{Before} & \rightarrow & & \text{SP = y} \\ \textbf{Return} & & & \end{array}$ 

After  $\rightarrow$  SP = y + 1 Return PC(15-0)

Previously stored data

Return

| Status Bits | Affected by       | none                                                               |
|-------------|-------------------|--------------------------------------------------------------------|
|             | Affects           | none                                                               |
| Repeat      | This instruction  | cannot be repeated.                                                |
| See Also    | See the following | ng other related instructions:                                     |
|             | ☐ Call Conditi    | onally                                                             |
|             | ☐ Call Uncon      | ditionally                                                         |
|             | ☐ Return Con      | ditionally                                                         |
|             | ☐ Return from     | n Interrupt                                                        |
| Example     |                   |                                                                    |
| Syntax      | Description       |                                                                    |
| return      | The program co    | unter is loaded with the return address of the calling subroutine. |

### RETI

### Return from Interrupt

### **Syntax Characteristics**

| No. | Syntax     | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|------------|------------------------|------|--------|----------|
| [1] | return_int | No                     | 2    | 5      | D        |

#### **Opcode**

none

# Operands Description

This instruction passes control back to the interrupted task.

After returning from an interrupt service routine (ISR), the CPU automatically restores the value of some CPU registers and two internal registers: the program counter (PC) and a loop context register. The CPU uses these values to re-establish the context of the program sequence.

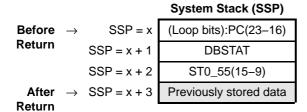
0100 100E xxxx x101

In the slow-return process (default), the return address (from the PC), the loop context bits, and some CPU registers are restored from the stacks (in memory). When the CPU returns from an ISR, the speed at which these values are restored is dependent on the speed of the memory accesses.

In the fast-return process, the return address (from the PC) and the loop context bits are restored from the return address register (RETA) and the control-flow context register (CFCT). You can read from or write to RETA and CFCT as a pair with dedicated, 32-bit load and store instructions. Some CPU registers are restored from the stacks (in memory). For fast-return mode operation, see the *TMS320C55x DSP CPU Reference Guide* (SPRU371).

- ☐ The loop context bits concatenated with the 8 MSBs of the return address are popped from the top of the system stack pointer (SSP). The SSP is incremented by 1 word in the address phase of the pipeline.
- ☐ The 16 LSBs of the return address are popped from the top of the data stack pointer (SP). The SP is incremented by 1 word in the address phase of the pipeline.
- ☐ The debug status register (DBSTAT) content is popped from the top of SSP. The SSP is incremented by 1 word in the access phase of the pipeline.
- ☐ The status register 1 (ST1\_55) content is popped from the top of SP. The SP is incremented by 1 word in the access phase of the pipeline.
- ☐ The 7 higher bits of status register 0 (ST0\_55) concatenated with 9 zeroes are popped from the top of SSP. The SSP is incremented by 1 word in the read phase of the pipeline.

☐ The status register 2 (ST2\_55) content is popped from the top of SP. The SP is incremented by 1 word in the read phase of the pipeline.



|        |                          | Data Stack (SP)        |
|--------|--------------------------|------------------------|
| Before | $\rightarrow$ SP = y     | PC(15-0)               |
| Return | SP = y + 1               | ST1_55                 |
|        | SP = y + 2               | ST2_55                 |
|        | $\rightarrow$ SP = y + 3 | Previously stored data |
| Return |                          | <u> </u>               |

Status Bits Affected by none
Affects none

**Repeat** This instruction cannot be repeated.

**See Also** See the following other related instructions:

- ☐ Return Conditionally
- □ Return Unconditionally
- ☐ Software Interrupt
- ☐ Software Trap

| Syntax     | Description                                                                         |  |
|------------|-------------------------------------------------------------------------------------|--|
| return_int | The program counter (PC) is loaded with the return address of the interrupted task. |  |

## ROL

## Rotate Left Accumulator, Auxiliary, or Temporary Register Content

### **Syntax Characteristics**

| No. | Syntax                                | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|---------------------------------------|------------------------|------|--------|----------|
|     | dst = BitOut \\ src \\ BitIn          |                        |      |        |          |
| [1] | dst = <b>TC2 \\</b> src <b>\\ TC2</b> | Yes                    | 3    | 1      | X        |
| [2] | dst = TC2 \\ src \\ CARRY             | Yes                    | 3    | 1      | Х        |
| [3] | dst = CARRY \\ src \\ TC2             | Yes                    | 3    | 1      | Х        |
| [4] | dst = CARRY \\ src \\ CARRY           | Yes                    | 3    | 1      | Х        |

#### **Opcode**

0001 001E FSSS xx11 FDDD 0xvv

### **Operands**

dst, src

#### Description

This instruction performs a bitwise rotation to the MSBs. Both TC2 and CARRY can be used to shift in one bit (BitIn) or to store the shifted out bit (BitOut). The one bit in BitIn is shifted into the source (src) operand and the shifted out bit is stored to BitOut.

- ☐ When the destination (dst) operand is an accumulator:
  - if an auxiliary or temporary register is the source (src) operand of the instruction, the 16 LSBs of the register are zero extended to 40 bits
  - the operation is performed on 40 bits in the D-unit shifter
  - BitIn is inserted at bit position 0
  - BitOut is extracted at a bit position according to M40
- ☐ When the destination (dst) operand is an auxiliary or temporary register:
  - if an accumulator is the source (src) operand of the instruction, the
     16 LSBs of the accumulator are used to perform the operation
  - the operation is performed on 16 bits in the A-unit ALU
  - BitIn is inserted at bit position 0
  - BitOut is extracted at bit position 15

### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

#### **Status Bits**

Affected by

CARRY, M40, TC2

Affects

CARRY, TC2

| Repeat   | This instruction can be repeated.                                    |
|----------|----------------------------------------------------------------------|
| See Also | See the following other related instructions:                        |
|          | ☐ Rotate Right Accumulator, Auxiliary, or Temporary Register Content |

| Syntax                    | Description                                                                                                                                                                                                                                             |
|---------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AC1 = CARRY \\ AC1 \\ TC2 | The value of TC2 (1) before the execution of the instruction is shifted into the LSB of AC1 and bit 31 shifted out from AC1 is stored in the CARRY status bit. The rotated value is stored in AC1. Because M40 = 0, the guard bits (39–32) are cleared. |

| Before |              | After |              |
|--------|--------------|-------|--------------|
| AC1    | OF E340 5678 | AC1   | 00 C680 ACF1 |
| TC2    | 1            | TC2   | 1            |
| CARRY  | 1            | CARRY | 1            |
| M40    | 0            | M40   | 0            |

### ROR

### Rotate Right Accumulator, Auxiliary, or Temporary Register Content

### **Syntax Characteristics**

| No. | Syntax                                | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|---------------------------------------|------------------------|------|--------|----------|
|     | dst = BitIn // src // BitOut          |                        |      |        |          |
| [1] | dst = <b>TC2</b> // src // <b>TC2</b> | Yes                    | 3    | 1      | X        |
| [2] | dst = TC2 // src // CARRY             | Yes                    | 3    | 1      | X        |
| [3] | dst = CARRY // src // TC2             | Yes                    | 3    | 1      | Х        |
| [4] | dst = CARRY // src // CARRY           | Yes                    | 3    | 1      | Х        |

#### **Opcode**

0001 001E FSSS xx11 FDDD 1xvv

## **Operands**

dst, src

#### Description

This instruction performs a bitwise rotation to the LSBs. Both TC2 and CARRY can be used to shift in one bit (BitIn) or to store the shifted out bit (BitOut). The one bit in BitIn is shifted into the source (src) operand and the shifted out bit is stored to BitOut.

- ☐ When the destination (dst) operand is an accumulator:
  - if an auxiliary or temporary register is the source (src) operand of the instruction, the 16 LSBs of the register are zero extended to 40 bits
  - the operation is performed on 40 bits in the D-unit shifter
  - BitIn is inserted at a bit position according to M40
  - BitOut is extracted at bit position 0
- ☐ When the destination (dst) operand is an auxiliary or temporary register:
  - if an accumulator is the source (src) operand of the instruction, the
     16 LSBs of the accumulator are used to perform the operation
  - the operation is performed on 16 bits in the A-unit ALU
  - BitIn is inserted at bit position 15
  - BitOut is extracted at bit position 0

### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

#### **Status Bits**

Affected by

CARRY, M40, TC2

Affects

CARRY, TC2

Repeat This instruction can be repeated. See Also See the following other related instructions: ☐ Rotate Left Accumulator, Auxiliary, or Temporary Register Content

| Syntax                  | Description                                                                                                                                                                                                                            |
|-------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AC1 = TC2 // AC0 // TC2 | The value of TC2 (1) before the execution of the instruction is shifted into bit 31 of AC0 and the LSB shifted out from AC0 is stored in TC2. The rotated value is stored in AC1. Because M40 = 0, the guard bits (39–32) are cleared. |

| Before |         |      | After |    |      |      |
|--------|---------|------|-------|----|------|------|
| AC0    | 5F B000 | 1234 | AC0   | 5F | B000 | 1234 |
| AC1    | 00 C680 | ACF1 | AC1   | 00 | D800 | 091A |
| TC2    |         | 1    | TC2   |    |      | 0    |
| M40    |         | 0    | M40   |    |      | 0    |

## ROUND

#### Round Accumulator Content

### **Syntax Characteristics**

| No. | Syntax         | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|----------------|------------------------|------|--------|----------|
| [1] | ACy = rnd(ACx) | Yes                    | 2    | 1      | Χ        |

#### **Opcode**

0101 010E DDSS 101%

### **Operands**

ACx, ACy

#### **Description**

This instruction performs a rounding of the source accumulator ACx in the D-unit ALU.

- ☐ The rounding operation depends on RDM:
  - $\blacksquare$  When RDM = 0, the biased rounding to the infinite is performed. 8000h (2<sup>15</sup>) is added to the 40-bit source accumulator ACx.
  - When RDM = 1, the unbiased rounding to the nearest is performed. According to the value of the 17 LSBs of the 40-bit source accumulator ACx, 8000h (215) is added:

```
if (8000h < bit(15-0) < 10000h)
   add 8000h to the 40-bit source accumulator ACx
else if ( bit (15-0) == 8000h)
   if(bit(16) == 1)
   add 8000h to the 40-bit source accumulator ACx
```

If a rounding has been performed, the 16 lowest bits of the result are cleared to 0.

- ☐ Addition overflow detection depends on M40.
- □ No addition carry report is stored in CARRY status bit.
- ☐ If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- When an overflow is detected, the accumulator is saturated according to SATD.

### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, the rounding is performed without clearing the LSBs of accumulator ACx.

**Status Bits** Affected by C54CM, M40, RDM, SATD

> Affects ACOVy

Repeat This instruction cannot be repeated.

| Syntax         | Description                                                                                                                                                                  |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AC1 = rnd(AC0) | The content of AC0 is added to 8000h, the 16 LSBs are cleared to 0, and the result is stored in AC1. M40 is cleared to 0, so overflow is detected at bit 31; SATD is cleared |
|                | to 0, so AC1 is not saturated.                                                                                                                                               |

| Before |    |      |      | After |    |      |      |
|--------|----|------|------|-------|----|------|------|
| AC0    | EF | 0FF0 | 8023 | AC0   | EF | 0FF0 | 8023 |
| AC1    | 00 | 0000 | 0000 | AC1   | EF | 0FF1 | 0000 |
| RDM    |    |      | 1    | RDM   |    |      | 1    |
| M40    |    |      | 0    | M40   |    |      | 0    |
| SATD   |    |      | 0    | SATD  |    |      | 0    |
| ACOV1  |    |      | 0    | ACOV1 |    |      | 1    |

### SAT

#### Saturate Accumulator Content

### **Syntax Characteristics**

| No. | Syntax                   | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|--------------------------|------------------------|------|--------|----------|
| [1] | ACy = saturate(rnd(ACx)) | Yes                    | 2    | 1      | Х        |

### **Opcode**

0101 010E DDSS 110%

#### **Operands**

ACx, ACy

## Description

This instruction performs a saturation of the source accumulator ACx to the 32-bit width frame in the D-unit ALU.

- A rounding is performed if the optional rnd keyword is applied to the instruction. The rounding operation depends on RDM:
  - When RDM = 0, the biased rounding to the infinite is performed. 8000h (2<sup>15</sup>) is added to the 40-bit source accumulator ACx.
  - When RDM = 1, the unbiased rounding to the nearest is performed. According to the value of the 17 LSBs of the 40-bit source accumulator ACx, 8000h (215) is added:

```
if( 8000h < bit(15-0) < 10000h)
   add 8000h to the 40-bit source accumulator ACx
else if ( bit (15-0) == 8000h)
   if(bit(16) == 1)
   add 8000h to the 40-bit source accumulator ACx
```

If a rounding has been performed, the 16 lowest bits of the result are cleared to 0.

- ☐ An overflow is detected at bit position 31.
- □ No addition carry report is stored in CARRY status bit.
- If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- ☐ When an overflow is detected, the destination register is saturated. Saturation values are 00 7FFF FFFFh (positive overflow) or FF 8000 0000h (negative overflow).

### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, the rounding is performed without clearing the LSBs of accumulator ACx.

**Status Bits** Affected by C54CM, RDM

> ACOVy Affects

Repeat This instruction can be repeated.

## Example 1

| Syntax              | Description                                                                                           |
|---------------------|-------------------------------------------------------------------------------------------------------|
| AC1 = saturate(AC0) | The 32-bit width content of AC0 is saturated and the saturated value, FF 8000 0000, is stored in AC1. |

| Before |       |         | After |      |           |
|--------|-------|---------|-------|------|-----------|
| AC0    | EF OF | F0 8023 | AC0   | EF 0 | FF0 8023  |
| AC1    | 00 00 | 0000    | AC1   | FF 8 | 8000 0000 |
| ACOV1  |       | 0       | ACOV1 |      | 1         |

| Syntax         | Description                                                                       |
|----------------|-----------------------------------------------------------------------------------|
| AC1 = satu-    | The 32-bit width content of AC0 is saturated. The saturated value, 00 7FFF FFFFh, |
| rate(rnd(AC0)) | is rounded, 16 LSBs are cleared, and stored in AC1.                               |

| Before |              | After |              |
|--------|--------------|-------|--------------|
| AC0    | 00 7FFF 8000 | AC0   | 00 7FFF 8000 |
| AC1    | 00 0000 0000 | AC1   | 00 7FFF 0000 |
| RDM    | (            | ) RDM | 0            |
| ACOV1  | (            | ACOV1 | 1            |

# **BSET**

Set Accumulator, Auxiliary, or Temporary Register Bit

## **Syntax Characteristics**

| No.                                    | Syntax         |                           |                                                                                                                                                                                                                                 | Parallel<br>Enable Bit | Size    | Cycles     | Pipeline     |  |
|----------------------------------------|----------------|---------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|---------|------------|--------------|--|
| [1]                                    | bit(src, Baddr | ) = #1                    |                                                                                                                                                                                                                                 | No                     | 3       | 1          | Χ            |  |
| Opcod                                  | le             |                           | 1110                                                                                                                                                                                                                            | 1100 AA                | AA A    | AAI FSS    | SS 000x      |  |
| Opera                                  | nds            | Baddr, src                |                                                                                                                                                                                                                                 |                        |         |            |              |  |
| Descri                                 | iption         | This instruction          | on performs a bit manipu                                                                                                                                                                                                        | ulation:               |         |            |              |  |
|                                        |                | In the D-υ                | ınit ALU, if the source (s                                                                                                                                                                                                      | src) register (        | operan  | d is an ac | cumulator.   |  |
|                                        |                | ☐ In the A-u<br>temporary | unit ALU, if the source (<br>y register.                                                                                                                                                                                        | src) register          | opera   | nd is an a | auxiliary or |  |
|                                        |                |                           | n sets to 1 a single bit, a source register.                                                                                                                                                                                    | as defined b           | y the b | it address | sing mode,   |  |
| The generated bit address must be with |                |                           |                                                                                                                                                                                                                                 | vithin:                |         |            |              |  |
|                                        |                | bit addres                | □ 0–39 when accessing accumulator bits (only the 6 LSBs of the generated bit address are used to determine the bit position). If the generated bit address is not within 0–39, the selected register bit value does not change. |                        |         |            |              |  |
|                                        |                |                           | en accessing auxiliary or nerated address are use                                                                                                                                                                               |                        | -       |            |              |  |
| Status                                 | Bits           | Affected by               | none                                                                                                                                                                                                                            |                        |         |            |              |  |
|                                        |                | Affects                   | none                                                                                                                                                                                                                            |                        |         |            |              |  |
| Repea                                  | t              | This instruction          | on can be repeated.                                                                                                                                                                                                             |                        |         |            |              |  |
| See Al                                 | lso            | See the follow            | ving other related instruc                                                                                                                                                                                                      | ctions:                |         |            |              |  |
|                                        |                | ☐ Clear Acc               | cumulator, Auxiliary, or T                                                                                                                                                                                                      | emporary R             | egister | Bit        |              |  |
|                                        |                | ☐ Complem                 | ent Accumulator, Auxilia                                                                                                                                                                                                        | ary, or Temp           | orary F | Register B | it           |  |
|                                        |                | ☐ Set Memo                | ory Bit                                                                                                                                                                                                                         |                        |         |            |              |  |
|                                        |                | ☐ Set Statu               | s Register Bit                                                                                                                                                                                                                  |                        |         |            |              |  |
| Examp                                  | ole            |                           |                                                                                                                                                                                                                                 |                        |         |            |              |  |

| Syntax             | Description                                                                    |
|--------------------|--------------------------------------------------------------------------------|
| bit(AC0, AR3) = #1 | The bit at the position defined by the content of AR3(4–0) in AC0 is set to 1. |

**BSET** 

Set Memory Bit

## **Syntax Characteristics**

| No.     | Syntax         |                  |                                                                  | Parallel<br>Enable Bit | Size     | Cycles    | Pipeline     |
|---------|----------------|------------------|------------------------------------------------------------------|------------------------|----------|-----------|--------------|
| [1]     | bit(Smem, src) | = #1             |                                                                  | No                     | 3        | 1         | Х            |
| Opcode  | e              |                  | 1110                                                             | 0011 AA                | AA A     | AAI   FSS | SS 1100      |
| Operar  | nds            | Smem, src        |                                                                  |                        |          |           |              |
| Descri  | ption          |                  | n performs a bit manipugle bit, as defined by the nem) location. |                        |          |           |              |
|         |                | _                | bit address must be wit termine the bit position                 | ,                      | ly the 4 | LSBs of   | the register |
| Status  | Bits           | Affected by      | none                                                             |                        |          |           |              |
|         |                | Affects          | none                                                             |                        |          |           |              |
| Repeat  | t              | This instruction | n can be repeated.                                               |                        |          |           |              |
| See Als | so             | See the follow   | ing other related instruc                                        | ctions:                |          |           |              |
|         |                | ☐ Clear Mem      | nory Bit                                                         |                        |          |           |              |
|         |                | ☐ Compleme       | ent Memory Bit                                                   |                        |          |           |              |
|         |                | ☐ Set Accum      | nulator, Auxiliary, or Ter                                       | mporary Reg            | jister B | it        |              |

## Example

| Syntax              | Description                                                                              |
|---------------------|------------------------------------------------------------------------------------------|
| bit(*AR3, AC0) = #1 | The bit at the position defined by AC0(3–0) in the content addressed by AR3 is set to 1. |

☐ Set Status Register Bit

### **BSET**

### Set Status Register Bit

### **Syntax Characteristics**

| No. | Syntax            | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|-------------------|------------------------|------|--------|----------|
| [1] | bit(ST0, k4) = #1 | Yes                    | 2    | 1      | Х        |
| [2] | bit(ST1, k4) = #1 | Yes                    | 2    | 1      | Х        |
| [3] | bit(ST2, k4) = #1 | Yes                    | 2    | 1      | X        |
| [4] | bit(ST3, k4) = #1 | Yes                    | 2    | 1†     | X        |

<sup>&</sup>lt;sup>†</sup> When this instruction is decoded to modify status bit CAFRZ (15), CAEN (14), or CACLR (13), the CPU pipeline is flushed and the instruction is executed in 5 cycles regardless of the instruction context.

| Opcode | ST0 | 0100 | 011E kkkk | 0001 |
|--------|-----|------|-----------|------|
|        | ST1 | 0100 | 011E kkkk | 0011 |
|        | ST2 | 0100 | 011E kkkk | 0101 |
|        | ST3 | 0100 | 011E kkkk | 0111 |

### Operands k4, STx

### **Description**

These instructions perform a bit manipulation in the A-unit ALU.

These instructions set to 1 a single bit, as defined by a 4-bit immediate value, k4, in the selected status register (ST0, ST1, ST2, or ST3).

It is not allowed to access DP register mapped in ST0 register with bit(ST0, k4) = #1 instruction. Therefore, k4 cannot have a value of 0–8.

It is not allowed to access ASM bit field in ST1 with bit(ST1, k4) = #1 instruction. Therefore, k4 cannot have a value of 0-4.

## Compatibility with C54x devices (C54CM = 1)

C55x DSP status registers bit mapping (Figure 5–3, page 5-526) does not correspond to C54x DSP status register bits.

Status Bits Affected by none

Affects Selected status bits

**Repeat** This instruction cannot be repeated.

**See Also** See the following other related instructions:

Clear Status Register Bit

| h | Set Accumulator, | Auxiliarv. | or Tem | porary | Register | Bit |
|---|------------------|------------|--------|--------|----------|-----|
|   |                  |            |        |        |          |     |

☐ Set Memory Bit

| Syntax                                       | Description                                                                |
|----------------------------------------------|----------------------------------------------------------------------------|
| bit(ST0, ST0_CARRY) = #1; ST0_CARRY = bit 11 | The ST0 bit position defined by the label (ST0_CARRY, bit 11) is set to 1. |

| Before |      | After |      |
|--------|------|-------|------|
| ST0    | 0000 | ST0   | 0800 |

Figure 5-3. Status Registers Bit Mapping

| ST0 | 55 |
|-----|----|
|-----|----|

| 15                                              | 14                            | 13                             |                                                     | 12                        | 11                                              | 10                                          | 9                            |  |  |  |  |
|-------------------------------------------------|-------------------------------|--------------------------------|-----------------------------------------------------|---------------------------|-------------------------------------------------|---------------------------------------------|------------------------------|--|--|--|--|
| ACOV2†                                          | ACOV3†                        | TC1                            | t                                                   | TC2                       | CARRY                                           | ACOV0                                       | ACOV1                        |  |  |  |  |
| R/W-0                                           | R/W-0                         | R/W-                           | -1 F                                                | /W-1                      | R/W-1                                           | R/W-0                                       | R/W-0                        |  |  |  |  |
| 8                                               |                               |                                |                                                     |                           |                                                 |                                             | 0                            |  |  |  |  |
|                                                 | DP                            |                                |                                                     |                           |                                                 |                                             |                              |  |  |  |  |
|                                                 |                               |                                | F                                                   | :/W-0                     |                                                 |                                             |                              |  |  |  |  |
| ST1_55                                          | ST1_55                        |                                |                                                     |                           |                                                 |                                             |                              |  |  |  |  |
| 15                                              | 14                            | 13                             | 12                                                  | 11                        | 10                                              | 9                                           | 8                            |  |  |  |  |
| BRAF                                            | CPL                           | XF                             | НМ                                                  | INTI                      | <b>М40</b> †                                    | SATD                                        | SXMD                         |  |  |  |  |
| R/W-0                                           | R/W-0                         | R/W-1                          | R/W-0                                               | R/W-                      | -1 R/W-0                                        | R/W-0                                       | R/W-1                        |  |  |  |  |
| 7                                               | 6                             | 5                              | 4                                                   |                           |                                                 |                                             | 0                            |  |  |  |  |
| C16                                             | FRCT                          | C54CM <sup>†</sup>             |                                                     |                           | ASM                                             |                                             |                              |  |  |  |  |
| R/W-0                                           | R/W-0                         | R/W-1                          |                                                     |                           | R/W-0                                           | )                                           |                              |  |  |  |  |
| ST2_55                                          |                               |                                |                                                     |                           |                                                 |                                             |                              |  |  |  |  |
|                                                 |                               |                                |                                                     |                           |                                                 |                                             |                              |  |  |  |  |
| 15                                              | 14                            | 13                             | 12                                                  | 11                        | 10                                              | 9                                           | 8                            |  |  |  |  |
| 15<br>ARMS                                      | 14<br>Rese                    |                                | 12<br>DBGM                                          | 11<br>EALLO               | •                                               | 9<br>Reserved                               | 8<br>CDPLC                   |  |  |  |  |
|                                                 |                               |                                |                                                     |                           | DW RDM                                          | Reserved                                    |                              |  |  |  |  |
| ARMS                                            |                               |                                | DBGM                                                | EALLO                     | DW RDM                                          | Reserved                                    | CDPLC                        |  |  |  |  |
| ARMS<br>R/W-0                                   | Rese                          | erved                          | DBGM<br>R/W-1                                       | EALLO<br>R/W-             | OW RDM<br>-0 R/W-0<br>2                         | Reserved )                                  | CDPLC<br>R/W-0               |  |  |  |  |
| ARMS<br>R/W-0<br>7                              | Rese                          | erved<br>5                     | DBGM<br>R/W-1<br>4                                  | EALLO<br>R/W-             | OW RDM -0 R/W-0 2 C AR2L0                       | Reserved  1  AR1LC                          | CDPLC<br>R/W-0               |  |  |  |  |
| ARMS R/W-0 7 AR7LC                              | Rese<br>6<br>AR6LC            | 5<br>AR5LC                     | DBGM<br>R/W-1<br>4<br>AR4LC                         | R/W-<br>3<br>AR3L         | OW RDM -0 R/W-0 2 C AR2L0                       | Reserved  1  AR1LC                          | CDPLC<br>R/W-0<br>0<br>AR0LC |  |  |  |  |
| ARMS R/W-0 7 AR7LC R/W-0                        | Rese<br>6<br>AR6LC            | 5<br>AR5LC                     | DBGM<br>R/W-1<br>4<br>AR4LC                         | R/W-<br>3<br>AR3L         | OW RDM -0 R/W-0 2 C AR2L0                       | Reserved  1  AR1LC                          | CDPLC<br>R/W-0<br>0<br>AR0LC |  |  |  |  |
| ARMS R/W-0 7 AR7LC R/W-0 ST3_55                 | 6 AR6LC R/W-0                 | 5<br>AR5LC<br>R/W-0            | DBGM<br>R/W-1<br>4<br>AR4LC<br>R/W-0                | R/W-<br>3<br>AR3L<br>R/W- | OW RDM -0 R/W-0 2 -C AR2L0 -0 R/W-0             | Reserved  1  AR1LC                          | CDPLC R/W-0 0 AROLC R/W-0    |  |  |  |  |
| ARMS R/W-0 7 AR7LC R/W-0 ST3_55 15              | 6 AR6LC R/W-0                 | 5<br>AR5LC<br>R/W-0            | DBGM<br>R/W-1<br>4<br>AR4LC<br>R/W-0                | R/W-<br>3<br>AR3L<br>R/W- | OW RDM -0 R/W-0 2 -C AR2L0 -0 R/W-0             | Reserved  1 C AR1LC R/W-0                   | CDPLC R/W-0 0 AROLC R/W-0    |  |  |  |  |
| ARMS R/W-0 7 AR7LC R/W-0 ST3_55 15 CAFRZ†       | 6 AR6LC R/W-0  14 CAEN†       | 5<br>AR5LC<br>R/W-0            | DBGM R/W-1 4 AR4LC R/W-0 12 HINT‡                   | R/W-<br>3<br>AR3L<br>R/W- | OW RDM -0 R/W-0 2 -C AR2L0 -0 R/W-0             | Reserved  1 C AR1LC R/W-0                   | CDPLC R/W-0 0 AROLC R/W-0    |  |  |  |  |
| ARMS R/W-0 7 AR7LC R/W-0 ST3_55 15 CAFRZ† R/W-0 | 6 AR6LC R/W-0  14 CAEN† R/W-0 | 5 AR5LC R/W-0  13 CACLR† R/W-0 | DBGM R/W-1 4 AR4LC R/W-0 12 HINT <sup>‡</sup> R/W-1 | R/W-<br>3<br>AR3L<br>R/W- | OW RDM -0 R/W-0 2 -C AR2L0 -0 R/W-0 Reserved (a | Reserved  1 C AR1LC R/W-0  always write 110 | CDPLC R/W-0 0 AROLC R/W-0 8  |  |  |  |  |

**Legend:** R = Read; W = Write; -n = Value after reset

<sup>&</sup>lt;sup>†</sup> Highlighted bit: If you write to the protected address of the status register, a write to this bit has no effect, and the bit always appears as a 0 during read operations.

<sup>&</sup>lt;sup>‡</sup> The HINT bit is not used for all C55x host port interfaces (HPIs). Consult the documentation for the specific C55x DSP.

<sup>§</sup> The reset value of MPNMC may be dependent on the state of predefined pins at reset. To check this for a particular C55x DSP, see the boot loader section of its data sheet.

# **SFTCC**

# Shift Accumulator Content Conditionally

## **Syntax Characteristics**

| No.     | Syntax         |                                                                         | Parallel<br>Enable Bit | Size     | Cycles      | Pipeline    |  |  |
|---------|----------------|-------------------------------------------------------------------------|------------------------|----------|-------------|-------------|--|--|
| [1]     | ACx = sftc(AC) | c, TC1)                                                                 | Yes                    | 2        | 1           | Х           |  |  |
| [2]     | ACx = sftc(AC) | c, TC2)                                                                 | Yes                    | 2        | 1           | Х           |  |  |
| Opcod   | e              | TC1                                                                     | 01                     | 01 10    | D1E DDx     | xx xx10     |  |  |
|         |                | TC2                                                                     | 01                     | 01 10    | D1E DDx     | x xx11      |  |  |
| Operar  | nds            | ACx, TCx                                                                |                        |          |             |             |  |  |
| Descri  | ption          | If the source accumulator ACx(39 status bit to 1.                       | 9–0) is equal to 0,    | this ins | truction se | ets the TCx |  |  |
|         |                | If the source accumulator ACx(3                                         | 31–0) has two sig      | n bits:  |             |             |  |  |
|         |                | ☐ this instruction shifts left the 32-bit accumulator ACx by 1 bit      |                        |          |             |             |  |  |
|         |                | ☐ the TCx status bit is cleared                                         | to 0                   |          |             |             |  |  |
|         |                | If the source accumulator ACs instruction sets the TCx status b         | •                      | t have   | two sigr    | bits, this  |  |  |
|         |                | The sign bits are extracted at bit                                      | positions 31 and       | 30.      |             |             |  |  |
| Status  | Bits           | Affected by none                                                        |                        |          |             |             |  |  |
|         |                | Affects TCx                                                             |                        |          |             |             |  |  |
| Repeat  | t              | This instruction can be repeated                                        |                        |          |             |             |  |  |
| See Als | so             | See the following other related in                                      | nstructions:           |          |             |             |  |  |
|         |                | ☐ Shift Accumulator Content Logically                                   |                        |          |             |             |  |  |
|         |                | ☐ Shift Accumulator, Auxiliary, or Temporary Register Content Logically |                        |          |             |             |  |  |
|         |                | ☐ Signed Shift of Accumulator                                           | Content                |          |             |             |  |  |
|         |                | ☐ Signed Shift of Accumulator                                           | , Auxiliary, or Tem    | nporary  | Register    | Content     |  |  |

# Example 1

| Syntax               | Description                                                                                              |
|----------------------|----------------------------------------------------------------------------------------------------------|
| AC0 = sftc(AC0, TC1) | Because AC0(31) XORed with AC0(30) equals 1, the content of AC0 is not shifted left and TC1 is set to 1. |

| Before |              | After |              |
|--------|--------------|-------|--------------|
| AC0    | FF 8765 0055 | AC0   | FF 8765 0055 |
| TC1    | 0            | TC1   | 1            |

| Syntax               | Description                                                                     |
|----------------------|---------------------------------------------------------------------------------|
| AC0 = sftc(AC0, TC2) | Because AC0(31) XORed with AC0(30) equals 0, the content of AC0 is shifted left |
|                      | by 1 bit and TC2 is cleared to 0.                                               |

| Before |        |        | After |    |      |      |
|--------|--------|--------|-------|----|------|------|
| AC0    | 00 123 | 4 0000 | AC0   | 00 | 2468 | 0000 |
| TC2    |        | 0      | TC2   |    |      | 0    |

## **SFTL**

## Shift Accumulator Content Logically

## **Syntax Characteristics**

| No. | Syntax                | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|-----------------------|------------------------|------|--------|----------|
| [1] | ACy = ACx <<< Tx      | Yes                    | 2    | 1      | Х        |
| [2] | ACy = ACx <<< #SHIFTW | Yes                    | 3    | 1      | Х        |

**Description** These instructions perform an unsigned shift by an immediate value, SHIFTW, or the content of a temporary register (Tx) in the D-unit shifter. **Status Bits** Affected by C54CM, M40 Affects **CARRY** See Also See the following other related instructions: ☐ Shift Accumulator Content Conditionally ☐ Shift Accumulator, Auxiliary, or Temporary Register Content Logically ☐ Signed Shift of Accumulator Content ☐ Signed Shift of Accumulator, Auxiliary, or Temporary Register Content

### Shift Accumulator Content Logically

### **Syntax Characteristics**

| No. | Syntax             | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|--------------------|------------------------|------|--------|----------|
| [1] | $ACy = ACx \ll Tx$ | Yes                    | 2    | 1      | Х        |

Opcode

0101 110E DDSS ss00

**Operands** 

ACx, ACy, Tx

Description

This instruction shifts by the temporary register (Tx) content the accumulator (ACx) content and stores the shifted-out bit in the CARRY status bit. If the 16-bit value contained in Tx is out of the -32 to +31 range, the shift is saturated to -32 or +31 and the shift operation is performed with this value. However, no overflow is reported when such saturation occurs.

- ☐ The operation is performed on 40 bits in the D-unit shifter.
- ☐ The shift operation is performed according to M40.
- $\Box$  The CARRY status bit contains the shifted-out bit. When the shift count is zero, Tx = 0, the CARRY status bit is cleared to 0.

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, the 6 LSBs of Tx define the shift quantity within -32 to +31. When the value is between -32 to -17, a modulo 16 operation transforms the shift quantity to within -16 to -1.

**Status Bits** 

Affected by C54CM, M40

Affects CARRY

Repeat

This instruction can be repeated.

| Syntax           | Description                                                                                                                                                                                                                      |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AC1 = AC0 >>> T0 | The content of AC0 is logically shifted right by the content of T0 and the result is stored in AC1. There is a right shift because the content of T0 is negative $(-6)$ . Because M40 = 0, the guard bits $(39-32)$ are cleared. |

| Before |    |      |      | After |    |      |      |
|--------|----|------|------|-------|----|------|------|
| AC0    | 5F | B000 | 1234 | AC0   | 5F | B000 | 1234 |
| AC1    | 00 | C680 | ACF0 | AC1   | 00 | 02C0 | 0048 |
| T0     |    |      | FFFA | TO    |    |      | FFFA |
| M40    |    |      | 0    | M40   |    |      | 0    |

# Shift Accumulator Content Logically

## **Syntax Characteristics**

| No.    | Syntax        |       |                                                                |                                    |           | Parallel<br>Enable Bit | Size   | Cycles       | Pipeline      |
|--------|---------------|-------|----------------------------------------------------------------|------------------------------------|-----------|------------------------|--------|--------------|---------------|
| [2]    | ACy = ACx <<< | #SHIF | TW                                                             |                                    |           | Yes                    | 3      | 1            | X             |
| Opcod  | e             |       |                                                                |                                    | 0001      | 000E DDS               | SS 01  | 11 xxS       | H IFTW        |
| Operar | nds           | ACx,  | ACy, SHI                                                       | FTW                                |           |                        |        |              |               |
| Descri | ption         |       |                                                                | shifts by a 6-b<br>shifted-out bit |           |                        |        | ulator (AC   | Cx) content   |
|        |               | _ T   | ☐ The operation is performed on 40 bits in the D-unit shifter. |                                    |           |                        |        |              |               |
|        |               | _ T   | he shift o                                                     | peration is pe                     | rformed a | ccording to I          | M40.   |              |               |
|        |               | _     |                                                                | RY status bit co<br>TTW = 0, the C |           |                        |        |              | nift count is |
|        |               | Com   | patibility                                                     | with C54x de                       | evices (C | 54CM = 1)              |        |              |               |
|        |               | Whe   | n this inst                                                    | ruction is exec                    | uted with | M40 = 0, co            | mpatib | oility is en | sured.        |
| Status | Bits          | Affec | Affected by M40                                                |                                    |           |                        |        |              |               |
|        |               | Affec | ts                                                             | CARRY                              |           |                        |        |              |               |
| Repeat | t             | This  | instructior                                                    | n can be repea                     | ated.     |                        |        |              |               |
| Examp  | ile           |       |                                                                |                                    |           |                        |        |              |               |

| Syntax            | Description                                                                              |
|-------------------|------------------------------------------------------------------------------------------|
| AC0 = AC1 <<< #31 | The content of AC1 is logically shifted left by 31 bits and the result is stored in AC0. |

# **SFTL**

Shift Accumulator, Auxiliary, or Temporary Register Content Logically

## **Syntax Characteristics**

| No. | Syntax                  | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|-------------------------|------------------------|------|--------|----------|
| [1] | dst = dst <<< #1        | Yes                    | 2    | 1      | X        |
| [2] | dst = dst >>> <b>#1</b> | Yes                    | 2    | 1      | X        |

| Description | These instructi          | ons perform an unsigned shift by 1 bit:                                  |
|-------------|--------------------------|--------------------------------------------------------------------------|
|             | ☐ In the D-ur            | nit shifter, if the destination operand is an accumulator (ACx).         |
|             | In the A-un register (TA | hit ALU, if the destination operand is an auxiliary or temporary $Ax$ ). |
| Status Bits | Affected by              | C54CM, M40                                                               |
|             | Affects                  | CARRY                                                                    |
| See Also    | See the followi          | ng other related instructions:                                           |
|             | ☐ Shift Accur            | mulator Content Conditionally                                            |
|             | ☐ Shift Accur            | mulator Content Logically                                                |
|             | ☐ Signed Shi             | ift of Accumulator Content                                               |
|             | ☐ Signed Shi             | ift of Accumulator, Auxiliary, or Temporary Register Content             |

## Shift Accumulator, Auxiliary, or Temporary Register Content Logically

## **Syntax Characteristics**

| No. | Syntax           | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|------------------|------------------------|------|--------|----------|
| [1] | dst = dst <<< #1 | Yes                    | 2    | 1      | Х        |
|     |                  | _                      |      |        |          |

## **Opcode**

Description

0101 000E FDDD x000

## **Operands**

dst

This instruction shifts left by 1 bit the input operand (dst). The CARRY status bit contains the shifted-out bit.

- ☐ When the destination operand (dst) is an accumulator:
  - The operation is performed on 40 bits in the D-unit shifter.
  - 0 is inserted at bit position 0.
  - The shifted-out bit is extracted at a bit position according to M40.
- ☐ When the destination operand (dst) is an auxiliary or temporary register:
  - The operation is performed on 16 bits in the A-unit ALU.
    - 0 is inserted at bit position 0.
    - The shifted-out bit is extracted at bit position 15 and stored in the CARRY status bit.

## Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

**Status Bits** Affected by M40

> Affects **CARRY**

Repeat This instruction can be repeated.

| Syntax           | Description                                                                                                                                                                                 |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AC1 = AC1 <<< #1 | The content of AC1 is logically shifted left by 1 bit and the result is stored in AC1. Because M40 = 0, the CARRY status bit is extracted at bit 31 and the guard bits (39–32) are cleared. |

| Before |              | After |              |
|--------|--------------|-------|--------------|
| AC1    | 8F E340 5678 | AC1   | 00 C680 ACF0 |
| CARRY  | 0            | CARRY | 1            |
| M40    | 0            | M4 0  | 0            |

## Shift Accumulator, Auxiliary, or Temporary Register Content Logically

### **Syntax Characteristics**

| No. | Syntax                  | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|-------------------------|------------------------|------|--------|----------|
| [2] | dst = dst >>> <b>#1</b> | Yes                    | 2    | 1      | Х        |

### Opcode

0101 000E FDDD x001

## Operands

dst

### Description

This instruction shifts right by 1 bit the input operand (dst). The CARRY status bit contains the shifted-out bit.

- ☐ When the destination operand (dst) is an accumulator:
  - The operation is performed on 40 bits in the D-unit shifter.
  - 0 is inserted at a bit position according to M40.
  - The shifted-out bit is extracted at bit position 0 and stored in the CARRY status bit.
- ☐ When the destination operand (dst) is an auxiliary or temporary register:
  - The operation is performed on 16 bits in the A-unit ALU.
  - 0 is inserted at bit position 15.
  - The shifted-out bit is extracted at bit position 0 and stored in the CARRY status bit.

## Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

## **Status Bits**

Affected by M40

Affects CARRY

#### Repeat

This instruction can be repeated.

| Syntax           | Description                                                                             |
|------------------|-----------------------------------------------------------------------------------------|
| AC0 = AC0 >>> #1 | The content of AC0 is logically shifted right by 1 bit and the result is stored in AC0. |

## **SFTS**

## Signed Shift of Accumulator Content

## **Syntax Characteristics**

| No. | Syntax                                 | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|----------------------------------------|------------------------|------|--------|----------|
| [1] | ACy = ACx << Tx                        | Yes                    | 2    | 1      | Х        |
| [2] | ACy = ACx <b>&lt;<c< b=""> Tx</c<></b> | Yes                    | 2    | 1      | X        |
| [3] | ACy = ACx << #SHIFTW                   | Yes                    | 3    | 1      | Х        |
| [4] | ACy = ACx << <b>C</b> #SHIFTW          | Yes                    | 3    | 1      | Х        |

**Description** These instructions perform a signed shift by an immediate value, SHIFTW, or by the content of a temporary register (Tx) in the D-unit shifter. **Status Bits** Affected by C54CM, M40, SATA, SATD, SXMD Affects ACOVx, ACOVy, CARRY See Also See the following other related instructions: ☐ Shift Accumulator Content Conditionally ☐ Shift Accumulator Content Logically ☐ Shift Accumulator, Auxiliary, or Temporary Register Content Logically ☐ Signed Shift of Accumulator, Auxiliary, or Temporary Register Content

## Signed Shift of Accumulator Content

### **Syntax Characteristics**

| No. | Syntax           | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|------------------|------------------------|------|--------|----------|
| [1] | ACy = ACx  << Tx | Yes                    | 2    | 1      | Х        |

### **Opcode**

0101 110E DDSS ss01

### **Operands**

ACx, ACy, Tx

#### **Description**

This instruction shifts by the temporary register (Tx) content the accumulator (ACx) content. If the 16-bit value contained in Tx is out of the -32 to +31 range, the shift is saturated to -32 or +31 and the shift operation is performed with this value; a destination accumulator overflow is reported when such saturation occurs.

- ☐ The operation is performed on 40 bits in the D-unit shifter.
- When M40 = 0, the input to the shifter is modified according to SXMD and then the modified input is shifted by the Tx content:
  - if SXMD = 0, 0 is substituted for the guard bits (39–32) as the input, instead of ACx(39–32), to the shifter
  - if SXMD = 1, bit 31 of the source operand is substituted for the guard bits (39–32) as the input, instead of ACx(39–32), to the shifter
- ☐ The sign position of the source operand is compared to the shift quantity. This comparison depends on M40:
  - if M40 =0, comparison is performed versus bit 31
  - if M40 =1, comparison is performed versus bit 39
- 0 is inserted at bit position 0.
- ☐ The shifted-out bit is extracted according to M40.
- $\square$  After shifting, unless otherwise noted, when M40 = 0:
  - overflow is detected at bit position 31 (if an overflow is detected, the destination ACOVy bit is set)
  - if SATD = 1, when an overflow is detected, the destination accumulator saturation values are 00 7FFF FFFh (positive overflow) or FF 8000 0000h (negative overflow)

- $\square$  After shifting, unless otherwise noted, when M40 = 1:
  - overflow is detected at bit position 39 (if an overflow is detected, the destination ACOVy bit is set)
  - if SATD = 1, when an overflow is detected, the destination accumulator saturation values are 7F FFFF FFFFh (positive overflow) or 80 0000 0000h (negative overflow)

### Compatibility with C54x devices (C54CM = 1)

When C54CM = 1:

- ☐ These instructions are executed as if M40 status bit was locally set to 1.
- ☐ There is no overflow detection, overflow report, and saturation performed by the D-unit shifter.
- ☐ The 6 LSBs of Tx are used to determine the shift quantity. The 6 LSBs of Tx define a shift quantity within –32 to +31. When the value is between –32 to -17, a modulo 16 operation transforms the shift quantity to within -16 to −1.

**Status Bits** Affected by C54CM, M40, SATD, SXMD

> Affects **ACOVy**

Repeat This instruction can be repeated.

| Syntax          | Description                                                                         |
|-----------------|-------------------------------------------------------------------------------------|
| AC0 = AC1 << T0 | The content of AC1 is shifted by the content of T0 and the result is stored in AC0. |

## Signed Shift of Accumulator Content

### **Syntax Characteristics**

| No.   | Syntax                                 | Parallel<br>Enable Bit | Size  | Cycles  | Pipeline |
|-------|----------------------------------------|------------------------|-------|---------|----------|
| [2]   | ACy = ACx <b>&lt;<c< b=""> Tx</c<></b> | Yes                    | 2     | 1       | Х        |
| Opcod | e                                      | 01                     | 01 13 | 10E DDS | SS ss10  |

## Operands

ACx, ACy, Tx

### Description

This instruction shifts by the temporary register (Tx) content the accumulator (ACx) content and stores the shifted-out bit in the CARRY status bit. If the 16-bit value contained in Tx is out of the -32 to +31 range, the shift is saturated to -32 or +31 and the shift operation is performed with this value; a destination accumulator overflow is reported when such saturation occurs.

- ☐ The operation is performed on 40 bits in the D-unit shifter.
- ☐ When M40 = 0, the input to the shifter is modified according to SXMD and then the modified input is shifted by the Tx content:
  - if SXMD = 0, 0 is substituted for the guard bits (39–32) as the input, instead of ACx(39–32), to the shifter
  - if SXMD = 1, bit 31 of the source operand is substituted for the guard bits (39–32) as the input, instead of ACx(39–32), to the shifter
- ☐ The sign position of the source operand is compared to the shift quantity. This comparison depends on M40:
  - if M40 =0, comparison is performed versus bit 31
  - if M40 =1, comparison is performed versus bit 39
- □ 0 is inserted at bit position 0.
- The shifted-out bit is extracted according to M40 and stored in the CARRY status bit. When the shift count is zero, Tx = 0, the CARRY status bit is cleared to 0.
- $\square$  After shifting, unless otherwise noted, when M40 = 0:
  - overflow is detected at bit position 31 (if an overflow is detected, the destination ACOVy bit is set)
  - if SATD = 1, when an overflow is detected, the destination accumulator saturation values are 00 7FFF FFFh (positive overflow) or FF 8000 0000h (negative overflow)

- $\square$  After shifting, unless otherwise noted, when M40 = 1:
  - overflow is detected at bit position 39 (if an overflow is detected, the destination ACOVy bit is set)
  - if SATD = 1, when an overflow is detected, the destination accumulator saturation values are 7F FFFF FFFFh (positive overflow) or 80 0000 0000h (negative overflow)

### Compatibility with C54x devices (C54CM = 1)

When C54CM = 1:

- ☐ These instructions are executed as if M40 status bit was locally set to 1.
- ☐ There is no overflow detection, overflow report, and saturation performed by the D-unit shifter.
- ☐ The 6 LSBs of Tx are used to determine the shift quantity. The 6 LSBs of Tx define a shift quantity within –32 to +31. When the value is between –32 to -17, a modulo 16 operation transforms the shift quantity to within -16 to −1.

**Status Bits** 

Affected by C54CM, M40, SATD, SXMD

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

| Syntax                                                                                                                                                                                                                                                                       | Description                                                                                                                                                                                                               |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AC2 = AC2 < <c t1<="" td=""><td>The content of AC2 is shifted left by the content of T1 and the saturated result is stored in AC2. The shifted out bit is stored in the CARRY status bit. Since SATD = 1 and M40 = <math>0</math>, AC2 = FF 8000 0000 (saturation).</td></c> | The content of AC2 is shifted left by the content of T1 and the saturated result is stored in AC2. The shifted out bit is stored in the CARRY status bit. Since SATD = 1 and M40 = $0$ , AC2 = FF 8000 0000 (saturation). |

| Before |              | After |              |
|--------|--------------|-------|--------------|
| AC2    | 80 AA00 1234 | AC2   | FF 8000 0000 |
| T1     | 0005         | T1    | 0005         |
| CARRY  | 0            | CARRY | 1            |
| M40    | 0            | M40   | 0            |
| ACOV2  | 0            | ACOV2 | 1            |
| SXMD   | 1            | SXMD  | 1            |
| SATD   | 1            | SATD  | 1            |

## Signed Shift of Accumulator Content

### **Syntax Characteristics**

| No.    | Syntax         |                                                                                  |                                           |           | Parallel<br>Enable Bit | Size     | Cycles     | Pipeline     |
|--------|----------------|----------------------------------------------------------------------------------|-------------------------------------------|-----------|------------------------|----------|------------|--------------|
| [3]    | ACy = ACx << i | #SHII                                                                            | FTW                                       |           | Yes                    | 3        | 1          | Х            |
| Opcode | e              |                                                                                  |                                           | 0001      | 000E DD                | SS 0     | 101 xxS    | SH IFTW      |
| Operar | nds            | AC                                                                               | x, ACy, SHIFTW                            |           |                        |          |            |              |
| Descri | otion          | This instruction shifts by a 6-bit value, SHIFTW, the accumulator (ACx) content. |                                           |           |                        |          | ator (ACx) |              |
|        |                |                                                                                  | The operation is perfor                   | med on 40 | bits in the D          | D-unit s | shifter.   |              |
|        |                |                                                                                  | When M40 = 0, the input                   |           |                        |          | •          | SXMD and     |
|        |                |                                                                                  | ■ if SXMD = 0, 0 is s instead of ACx(39-  |           | _                      | d bits ( | 39–32) as  | s the input, |
|        |                |                                                                                  | ■ if SXMD = 1, bit 31 bits (39–32) as the |           | •                      |          |            | •            |

- if M40 =0, comparison is performed versus bit 31
- if M40 =1, comparison is performed versus bit 39
- 0 is inserted at bit position 0.
- ☐ The shifted-out bit is extracted according to M40.

This comparison depends on M40:

- $\square$  After shifting, unless otherwise noted, when M40 = 0:
  - overflow is detected at bit position 31 (if an overflow is detected, the destination ACOVy bit is set)

☐ The sign position of the source operand is compared to the shift quantity.

- if SATD = 1, when an overflow is detected, the destination accumulator saturation values are 00 7FFF FFFFh (positive overflow) or FF 8000 0000h (negative overflow)
- ☐ After shifting, unless otherwise noted, when M40 = 1:
  - overflow is detected at bit position 39 (if an overflow is detected, the destination ACOVy bit is set)
  - if SATD = 1, when an overflow is detected, the destination accumulator saturation values are 7F FFFF FFFFh (positive overflow) or 80 0000 0000h (negative overflow)

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, these instructions are executed as if M40 status bit was locally set to 1. There is no overflow detection, overflow report, and saturation

performed by the D-unit shifter.

**Status Bits** Affected by C54CM, M40, SATD, SXMD

> Affects **ACOVy**

Repeat This instruction can be repeated.

## **Example 1**

| Syntax           | Description                                                                    |
|------------------|--------------------------------------------------------------------------------|
| AC0 = AC1 << #31 | The content of AC1 is shifted left by 31 bits and the result is stored in AC0. |

| Syntax            | Description                                                                     |
|-------------------|---------------------------------------------------------------------------------|
| AC0 = AC1 << #-32 | The content of AC1 is shifted right by 32 bits and the result is stored in AC0. |

## Signed Shift of Accumulator Content

## **Syntax Characteristics**

| No.    | Syntax       |                |                                                                                                                                         |                                                    |                | Parallel<br>Enable Bit | Size     | Cycles      | Pipeline      |
|--------|--------------|----------------|-----------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------|----------------|------------------------|----------|-------------|---------------|
| [4]    | ACy = ACx << | : <b>C</b> #S⊦ | IIFTV                                                                                                                                   | I                                                  |                | Yes                    | 3        | 1           | Х             |
| Opcod  | е            |                |                                                                                                                                         |                                                    | 0001           | 000E DD                | SS 0     | 110   xxs   | SH IFTW       |
| Operar | nds          | AC             | x, A(                                                                                                                                   | Cy, SHIFTW                                         |                |                        |          |             |               |
| Descri | ption        |                | This instruction shifts by a 6-bit value, SHIFTW, the accumulator (ACx) content and stores the shifted-out bit in the CARRY status bit. |                                                    |                |                        |          |             |               |
|        |              |                | The                                                                                                                                     | operation is perf                                  | ormed on 40    | bits in the [          | D-unit s | shifter.    |               |
|        |              |                |                                                                                                                                         | en M40 = 0, the ir<br>n the modified inp           | -              |                        |          | _           | SXMD and      |
|        |              |                |                                                                                                                                         | if SXMD = 0, 0 is instead of ACx(3)                |                | -                      | d bits ( | 39–32) as   | s the input,  |
|        |              |                | •                                                                                                                                       | if SXMD = 1, bit 3<br>bits (39–32) as the          |                | •                      |          |             | •             |
|        |              |                |                                                                                                                                         | sign position of t<br>comparison dep               | •              |                        | npared   | I to the sh | ift quantity. |
|        |              |                |                                                                                                                                         | if M40 =0, compa                                   | arison is perf | ormed versu            | ıs bit 3 | 1           |               |
|        |              |                | •                                                                                                                                       | if M40 =1, compa                                   | arison is perf | ormed versu            | ıs bit 3 | 9           |               |
|        |              |                | 0 is                                                                                                                                    | inserted at bit po                                 | sition 0.      |                        |          |             |               |
|        |              |                | stat                                                                                                                                    | shifted-out bit is us bit. When the scleared to 0. |                | -                      |          |             |               |

- $\square$  After shifting, unless otherwise noted, when M40 = 0:
  - overflow is detected at bit position 31 (if an overflow is detected, the destination ACOVy bit is set)
  - if SATD = 1, when an overflow is detected, the destination accumulator saturation values are 00 7FFF FFFh (positive overflow) or FF 8000 0000h (negative overflow)

- $\square$  After shifting, unless otherwise noted, when M40 = 1:
  - overflow is detected at bit position 39 (if an overflow is detected, the destination ACOVy bit is set)
  - if SATD = 1, when an overflow is detected, the destination accumulator saturation values are 7F FFFF FFFFh (positive overflow) or 80 0000 0000h (negative overflow)

## Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, these instructions are executed as if M40 status bit was locally set to 1. There is no overflow detection, overflow report, and saturation performed by the D-unit shifter.

**Status Bits** Affected by C54CM, M40, SATD, SXMD

> Affects ACOVy, CARRY

Repeat This instruction can be repeated.

| Syntax                                                                                                                               | Description                                                                                |
|--------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------|
| AC1 = AC0 < <c #-5<="" td=""><td>The content of AC0 is shifted right by 5 bits and the result is stored in AC1. The shifted</td></c> | The content of AC0 is shifted right by 5 bits and the result is stored in AC1. The shifted |
|                                                                                                                                      | out bit is stored in the CARRY status bit.                                                 |

| Before |    |      |      | After |    |      |      |
|--------|----|------|------|-------|----|------|------|
| AC0    | FF | 8765 | 0055 | AC0   | FF | 8765 | 0055 |
| AC1    | 00 | 4321 | 1234 | AC1   | FF | FC3B | 2802 |
| CARRY  |    |      | 0    | CARRY |    |      | 1    |
| SXMD   |    |      | 1    | SXMD  |    |      | 1    |

## **SFTS**

Signed Shift of Accumulator, Auxiliary, or Temporary Register Content

## **Syntax Characteristics**

| No. | Syntax          | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|-----------------|------------------------|------|--------|----------|
| [1] | dst = dst >> #1 | Yes                    | 2    | 1      | Х        |
| [2] | dst = dst << #1 | Yes                    | 2    | 1      | X        |

Description These instructions perform a shift of 1 bit: ☐ In the D-unit shifter, if the destination operand is an accumulator (ACx). ☐ In the A-unit ALU, if the destination operand is an auxiliary or temporary register (TAx). **Status Bits** Affected by C54CM, M40, SATA, SATD, SXMD Affects ACOVx, ACOVy, CARRY See Also See the following other related instructions: Shift Accumulator Content Conditionally ☐ Shift Accumulator Content Logically ☐ Shift Accumulator, Auxiliary, or Temporary Register Content Logically ☐ Signed Shift of Accumulator Content

# Signed Shift of Accumulator, Auxiliary, or Temporary Register Content

## **Syntax Characteristics**

| No.    | Syntax          |                                                                                                                                          |                |                                                                  | Parallel<br>Enable Bit | Size     | Cycles      | Pipeline     |  |  |
|--------|-----------------|------------------------------------------------------------------------------------------------------------------------------------------|----------------|------------------------------------------------------------------|------------------------|----------|-------------|--------------|--|--|
| [1]    | dst = dst >> #1 |                                                                                                                                          |                |                                                                  | Yes                    | 2        | 1           | Х            |  |  |
| Opcod  | e               |                                                                                                                                          |                |                                                                  | 01                     | .00 0    | 10E   01×   | 0 FDDD       |  |  |
| Opera  | nds             | dst                                                                                                                                      | t              |                                                                  |                        |          |             |              |  |  |
| Descri | ption           | This instruction shifts right by 1 bit the content of the destination register (dst).                                                    |                |                                                                  |                        |          |             |              |  |  |
|        |                 | lf tl                                                                                                                                    | he destinati   | on operand (dst) is a                                            | ın accumulator         | :        |             |              |  |  |
|        |                 |                                                                                                                                          | The opera      | tion is performed on                                             | 40 bits in the l       | D-unit s | shifter.    |              |  |  |
|        |                 |                                                                                                                                          |                | 0 = 0, the input to the nodified input is shiften                |                        |          | ording to   | SXMD and     |  |  |
|        |                 |                                                                                                                                          |                | ID = 0, 0 is substituted of ACx(39–32), to                       | _                      | d bits ( | 39–32) as   | s the input, |  |  |
|        |                 | ■ if SXMD = 1, bit 31 of the source operand is substituted for the gual bits (39–32) as the input, instead of ACx(39–32), to the shifter |                |                                                                  |                        |          | -           |              |  |  |
|        |                 | ☐ Bit 39 is extended according to SXMD                                                                                                   |                |                                                                  |                        |          |             |              |  |  |
|        |                 | ☐ The shifted-out bit is extracted at bit position 0.                                                                                    |                |                                                                  |                        |          |             |              |  |  |
|        |                 | lf tl                                                                                                                                    | he destinati   | on operand (dst) is a                                            | n auxiliary or t       | empora   | ary registe | er:          |  |  |
|        |                 |                                                                                                                                          | The opera      | tion is performed on                                             | 16 bits in the         | ۹-unit A | ALU.        |              |  |  |
|        |                 |                                                                                                                                          | Bit 15 is si   | gn extended.                                                     |                        |          |             |              |  |  |
|        |                 | Co                                                                                                                                       | mpatibility    | with C54x devices                                                | (C54CM=1)              |          |             |              |  |  |
|        |                 | loc                                                                                                                                      | ally set to 1  | = 1, these instruction. There is no overflow the D-unit shifter. |                        |          |             |              |  |  |
| Status | Bits            | Aff                                                                                                                                      | ected by       | C54CM, M40, SXN                                                  | MD                     |          |             |              |  |  |
|        |                 | Aff                                                                                                                                      | ects           | none                                                             |                        |          |             |              |  |  |
| Repea  | t               | Thi                                                                                                                                      | is instruction | n can be repeated.                                               |                        |          |             |              |  |  |
|        |                 |                                                                                                                                          |                |                                                                  |                        |          |             |              |  |  |

# Example

| Syntax          | Description                                                                   |
|-----------------|-------------------------------------------------------------------------------|
| AC0 = AC0 >> #1 | The content of AC0 is shifted right by 1 bit and the result is stored in AC0. |

5-546 Instruction Set Descriptions

## Signed Shift of Accumulator, Auxiliary, or Temporary Register Content

## **Syntax Characteristics**

| No.    | Syntax          |       |                                                                              | Parallel<br>Enable Bit | Size     | Cycles     | Pipeline      |
|--------|-----------------|-------|------------------------------------------------------------------------------|------------------------|----------|------------|---------------|
| [2]    | dst = dst << #1 |       |                                                                              | Yes                    | 2        | 1          | Х             |
| Opcod  | e               |       |                                                                              | 01                     | 00 0     | 10E 012    | k1 FDDD       |
| Operar | nds             | dst   |                                                                              | ,                      |          | ı          |               |
| Descri | ption           | Thi   | is instruction shifts left by 1 bit th                                       | ne content of th       | e desti  | ination re | gister (dst). |
|        |                 | lf th | ne destination operand (dst) is a                                            | an accumulator         | :        |            |               |
|        |                 |       | The operation is performed on                                                | 40 bits in the I       | D-unit s | shifter.   |               |
|        |                 |       | When M40 = 0, the input to the then the modified input is shift              |                        | ied acc  | cording to | SXMD and      |
|        |                 |       | ■ if SXMD = 0, 0 is substitut instead of ACx(39–32), to                      | _                      | d bits ( | 39–32) a   | s the input   |
|        |                 |       | ■ if SXMD = 1, bit 31 of the s<br>bits (39–32) as the input, i               | •                      |          |            | •             |
|        |                 |       | The sign position of the source This comparison depends on I                 | •                      | npared   | to the sh  | ift quantity  |
|        |                 |       | ■ if M40 =0, comparison is p                                                 | erformed versu         | us bit 3 | 31         |               |
|        |                 |       | ■ if M40 =1, comparison is p                                                 | erformed versu         | us bit 3 | 9          |               |
|        |                 |       | 0 is inserted at bit position 0.                                             |                        |          |            |               |
|        |                 |       | The shifted-out bit is extracted                                             | according to N         | 140.     |            |               |
|        |                 |       | After shifting, unless otherwise                                             | e noted, when N        | Л40 = (  | O:         |               |
|        |                 |       | <ul><li>overflow is detected at bit<br/>destination ACOVx bit is s</li></ul> | •                      | an over  | flow is de | etected, the  |
|        |                 |       | ■ if SATD = 1, when an                                                       | overflow is            | detect   | ed, the    | destinatior   |

 $\square$  After shifting, unless otherwise noted, when M40 = 1:

overflow) or FF 8000 0000h (negative overflow)

overflow is detected at bit position 39 (if an overflow is detected, the destination ACOVx bit is set)

accumulator saturation values are 00 7FFF FFFFh (positive

■ if SATD = 1, when an overflow is detected, the destination accumulator saturation values are 7F FFFF FFFFh (positive overflow) or 80 0000 0000h (negative overflow)

If the destination operand (dst) is an auxiliary or temporary register:

- ☐ The operation is performed on 16 bits in the A-unit ALU.
- 0 is inserted at bit position 0.
- ☐ After shifting, unless otherwise noted:
  - overflow is detected at bit position 15 (if an overflow is detected, the destination ACOVx bit is set)
  - if SATA = 1, when an overflow is detected, the destination register saturation values are 7FFFh (positive overflow) or 8000h (negative overflow)

## Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, these instructions are executed as if M40 status bit was locally set to 1. There is no overflow detection, overflow report, and saturation performed by the D-unit shifter.

**Status Bits** 

Affected by

C54CM, M40, SATA, SATD, SXMD

Affects

ACOVx

Repeat

This instruction can be repeated.

| Syntax        | Description                                                                |  |
|---------------|----------------------------------------------------------------------------|--|
| T2 = T2 << #1 | The content of T2 is shifted left by 1 bit and the result is stored in T2. |  |

| Before |      | After |      |
|--------|------|-------|------|
| T2     | EF27 | T2    | DE4E |
| SATA   | 1    | SATA  | 1    |

INTR

### Software Interrupt

### **Syntax Characteristics**

| No.   | Syntax   | Parallel<br>Enable Bit | Size | Cycles    | Pipeline |
|-------|----------|------------------------|------|-----------|----------|
| [1]   | intr(k5) | No                     | 2    | 3         | D        |
| Opcod | e        | 100                    | 01 0 | 101   0xx | k kkkk   |

#### **Opcode**

**Operands** 

k5

Description

This instruction passes control to a specified interrupt service routine (ISR) and interrupts are globally disabled (INTM bit is set to 1 after ST1\_55 content is pushed onto the data stack pointer). The ISR address is stored at the interrupt vector address defined by the content of an interrupt vector pointer (IVPD or IVPH) combined with the 5-bit constant, k5. This instruction is executed regardless of the value of INTM bit.

#### Note:

DBSTAT (the debug status register) holds debug context information used during emulation. Make sure the ISR does not modify the value that will be returned to DBSTAT.

Before beginning an ISR, the CPU automatically saves the value of some CPU registers and two internal registers: the program counter (PC) and a loop context register. The CPU can use these values to re-establish the context of the interrupted program sequence when the ISR is done.

In the slow-return process (default), the return address (from the PC), the loop context bits, and some CPU registers are stored to the stacks (in memory). When the CPU returns from an ISR, the speed at which these values are restored is dependent on the speed of the memory accesses.

In the fast-return process, the return address (from the PC) and the loop context bits are saved to registers, so that these values can always be restored quickly. These special registers are the return address register (RETA) and the control-flow context register (CFCT). You can read from or write to RETA and CFCT as a pair with dedicated, 32-bit load and store instructions. Some CPU registers are saved to the stacks (in memory). For fast-return mode operation, see the TMS320C55x DSP CPU Reference Guide (SPRU371).

When control is passed to the ISR:

| The data stack pointer (SP) is decremented by 1 word in the address     |
|-------------------------------------------------------------------------|
| phase of the pipeline. The status register 2 (ST2_55) content is pushed |
| to the top of SP.                                                       |

- ☐ The system stack pointer (SSP) is decremented by 1 word in the address phase of the pipeline. The 7 higher bits of status register 0 (ST0\_55) concatenated with 9 zeroes are pushed to the top of SSP.
- ☐ The SP is decremented by 1 word in the access phase of the pipeline. The status register 1 (ST1\_55) content is pushed to the top of SP.
- ☐ The SSP is decremented by 1 word in the access phase of the pipeline. The debug status register (DBSTAT) content is pushed to the top of SSP.
- ☐ The SP is decremented by 1 word in the read phase of the pipeline. The 16 LSBs of the return address, from the program counter (PC), of the called subroutine are pushed to the top of SP.
- ☐ The SSP is decremented by 1 word in the read phase of the pipeline. The loop context bits concatenated with the 8 MSBs of the return address are pushed to the top of SSP.
- ☐ The PC is loaded with the ISR program address. The active control flow execution context flags are cleared.

When the software interrupt is acknowledged, the corresponding bits in IFR0 and IFR1 are cleared.

#### System Stack (SSP)

After 
$$\rightarrow$$
 SSP = x - 3 (Loop bits):PC(23-16)

SSP = x - 2 DBSTAT

SSP = x - 1 ST0\_55(15-9)

Before  $\rightarrow$  SSP = x

Save

#### Data Stack (SP)

After 
$$\rightarrow$$
 SP = y - 3 PC(15-0)
Save SP = y - 2 ST1\_55
SP = y - 1 ST2\_55
Before  $\rightarrow$  SP = y
Save Previously saved data

Status Bits Affected by none

Affects INTM, IFR0, IFR1

**Repeat** This instruction cannot be repeated.

**See Also** See the following other related instructions:

Return from Interrupt

Software Trap

| Syntax     | Description                                                                                           |
|------------|-------------------------------------------------------------------------------------------------------|
| intr(#3)   | Program control is passed to the specified interrupt service routine. The interrupt vector address is |
| 11111 (#3) | defined by the content of an interrupt vector pointer (IVPD) combined with the unsigned 5-bit value   |

RESET

Software Reset

## **Syntax Characteristics**

| No.         | Syntax |                                                                                   |                                                                                                                                                                                       | Parallel<br>Enable bit                                        | Size                                | Cycles                                                         | Pipeline                                                 |  |  |
|-------------|--------|-----------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------|-------------------------------------|----------------------------------------------------------------|----------------------------------------------------------|--|--|
| [1]         | reset  |                                                                                   |                                                                                                                                                                                       | No                                                            | 2                                   | ?                                                              | D                                                        |  |  |
| Opcod       | е      |                                                                                   |                                                                                                                                                                                       | 10                                                            | 01 01                               | 100   xxx                                                      | x xxxx                                                   |  |  |
| Operar      | nds    | none                                                                              |                                                                                                                                                                                       |                                                               |                                     |                                                                |                                                          |  |  |
| Description |        |                                                                                   | This instruction performs a nonmaskable software reset that can be used any time to put the device in a known state.                                                                  |                                                               |                                     |                                                                |                                                          |  |  |
|             |        | (Table 5–5 ar<br>pointer registe<br>is acknowled<br>pending inter<br>system contr | truction affects ST0_55, and Figure 5–4); status refers (IVPD and IVPH) are reged, the INTM is set to rupts in IFR0 and IFR1 of register, the interrupt of the from the initializate. | egister ST3 not affected. 1 to disable are cleare t vectors p | 5_55 all When le mas d. The ointer, | nd interru<br>the reset<br>kable inte<br>initializa<br>and the | instruction<br>errupts. All<br>tion of the<br>peripheral |  |  |
| Status      | Bits   | Affected by                                                                       | none                                                                                                                                                                                  |                                                               |                                     |                                                                |                                                          |  |  |
|             |        | Affects                                                                           | IFR0, IFR1, ST0_55,                                                                                                                                                                   | ST1_55, ST                                                    | 2_55                                |                                                                |                                                          |  |  |

This instruction cannot be repeated.

Repeat

Table 5-5. Effects of a Software Reset on DSP Registers

| Register | Bit   | Reset<br>Value | Comment                                                                                                                                                       |
|----------|-------|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| T2       | All   | 0              | All bits are cleared. To ensure TMS320C54x DSP compatibility, instructions affected by ASM bit will use a shift count of 0 (no shift).                        |
| IFR0     | All   | 0              | All pending interrupt flags are cleared.                                                                                                                      |
| IFR1     | All   | 0              | All pending interrupt flags are cleared.                                                                                                                      |
| ST0_55   | ACOV2 | 0              | AC2 overflow flag is cleared.                                                                                                                                 |
|          | ACOV3 | 0              | AC3 overflow flag is cleared.                                                                                                                                 |
|          | TC1   | 1              | Test control flag 1 is cleared.                                                                                                                               |
|          | TC2   | 1              | Test control flag 2 is cleared.                                                                                                                               |
|          | CARRY | 1              | CARRY bit is cleared.                                                                                                                                         |
|          | ACOV0 | 0              | AC0 overflow flag is cleared.                                                                                                                                 |
|          | ACOV1 | 0              | AC1 overflow flag is cleared.                                                                                                                                 |
|          | DP    | 0              | All bits are cleared, data page 0 is selected.                                                                                                                |
| ST1_55   | BRAF  | 0              | This flag is cleared.                                                                                                                                         |
|          | CPL   | 0              | The DP (rather than SP) direct addressing mode is selected. Direct accesses to data space are made relative to the data page register (DP).                   |
|          | XF    | 1              | External flag is set.                                                                                                                                         |
|          | НМ    | 0              | When an active HOLD signal forces the DSP to place its external interface in the high-impedance state, the DSP continues executing code from internal memory. |
|          | INTM  | 1              | Maskable interrupts are globally disabled.                                                                                                                    |
|          | M40   | 0              | 32-bit (rather than 40-bit) computation mode is selected for the D unit.                                                                                      |
|          | SATD  | 0              | CPU will not saturate overflow results in the D unit.                                                                                                         |
|          | SXMD  | 1              | Sign-extension mode is on.                                                                                                                                    |
|          | C16   | 0              | Dual 16-bit mode is off. For an instruction that is affected by C16, the D-unit ALU performs one 32-bit operation rather than two parallel 16-bit operations. |
|          | FRCT  | 0              | Results of multiply operations are not shifted.                                                                                                               |
|          | C54CM | 1              | TMS320C54x-compatibility mode is on.                                                                                                                          |
|          | ASM   | 0              | Instructions affected by ASM will use a shift count of 0 (no shift).                                                                                          |

5-552 Instruction Set Descriptions

Table 5–5. Effects of a Software Reset on DSP Registers (Continued)

| Register | Bit    | Reset<br>Value | Comment                                                                                                                                       |
|----------|--------|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| ST2_55   | ARMS   | 0              | When you use the AR indirect addressing mode, the DSP mode (rather than control mode) operands are available.                                 |
|          | DBGM   | 1              | Debug events are disabled.                                                                                                                    |
|          | EALLOW | 0              | A program cannot write to the non-CPU emulation registers.                                                                                    |
|          | RDM    | 0              | When an instruction specifies that an operand should be rounded, the CPU uses rounding to the infinite (rather than rounding to the nearest). |
|          | CDPLC  | 0              | CDP is used for linear addressing (rather than circular addressing).                                                                          |
|          | AR7LC  | 0              | AR7 is used for linear addressing.                                                                                                            |
|          | AR6LC  | 0              | AR6 is used for linear addressing.                                                                                                            |
|          | AR5LC  | 0              | AR5 is used for linear addressing.                                                                                                            |
|          | AR4LC  | 0              | AR4 is used for linear addressing.                                                                                                            |
|          | AR3LC  | 0              | AR3 is used for linear addressing.                                                                                                            |
|          | AR2LC  | 0              | AR2 is used for linear addressing.                                                                                                            |
|          | AR1LC  | 0              | AR1 is used for linear addressing.                                                                                                            |
|          | AR0LC  | 0              | AR0 is used for linear addressing.                                                                                                            |

Figure 5-4. Effects of a Software Reset on Status Registers

## ST0\_55

| 15    | 14    | 13  | 12  | 11    | 10    | 9     |
|-------|-------|-----|-----|-------|-------|-------|
| ACOV2 | ACOV3 | TC1 | TC2 | CARRY | ACOV0 | ACOV1 |
| 0     | 0     | 1   | 1   | 1     | 0     | 0     |

| 8 |    | 0 |
|---|----|---|
|   | DP |   |
|   | 0  | _ |

### ST1\_55

|   | 15   | 14   | 13    | 12 | 11   | 10  | 9    | 8    |
|---|------|------|-------|----|------|-----|------|------|
|   | BRAF | CPL  | XF    | НМ | INTM | M40 | SATD | SXMD |
|   | 0    | 0    | 1     | 0  | 1    | 0   | 0    | 1    |
|   |      |      |       |    |      |     |      |      |
| _ | 7    | 6    | 5     | 4  |      |     |      | 0    |
|   | C16  | FRCT | C54CM |    |      | ASM |      |      |
|   | 0    | 0    | 1     |    |      | 0   | _    |      |

#### ST2\_55

| 15    | 14    | 13    | 12    | 11     | 10    | 9        | 8     |
|-------|-------|-------|-------|--------|-------|----------|-------|
| ARMS  | Rese  | erved | DBGM  | EALLOW | RDM   | Reserved | CDPLC |
| 0     |       |       | 1     | 0      | 0     |          | 0     |
|       |       |       |       |        |       |          |       |
| 7     | 6     | 5     | 4     | 3      | 2     | 1        | 0     |
| AR7LC | AR6LC | AR5LC | AR4LC | AR3LC  | AR2LC | AR1LC    | AR0LC |
| 0     | 0     | 0     | 0     | 0      | 0     | 0        | 0     |

**TRAP** 

Software Trap

#### **Syntax Characteristics**

| No. | Syntax   | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|----------|------------------------|------|--------|----------|
| [1] | trap(k5) | No                     | 2    | ?      | D        |

Opcode

1001 0101 1xxk kkkk

**Operands** 

k5

**Description** 

This instruction passes control to a specified interrupt service routine (ISR) and this instruction does not affect INTM bit in ST1\_55 and DBGM bit in ST2\_55. The ISR address is stored at the interrupt vector address defined by the content of an interrupt vector pointer (IVPD or IVPH) combined with the 5-bit constant, k5. This instruction is executed regardless of the value of INTM bit . This instruction is not maskable.

#### Note:

DBSTAT (the debug status register) holds debug context information used during emulation. Make sure the ISR does not modify the value that will be returned to DBSTAT.

Before beginning an ISR, the CPU automatically saves the value of some CPU registers and two internal registers: the program counter (PC) and a loop context register. The CPU can use these values to re-establish the context of the interrupted program sequence when the ISR is done.

In the slow-return process (default), the return address (from the PC), the loop context bits, and some CPU registers are stored to the stacks (in memory). When the CPU returns from an ISR, the speed at which these values are restored is dependent on the speed of the memory accesses.

In the fast-return process, the return address (from the PC) and the loop context bits are saved to registers, so that these values can always be restored quickly. These special registers are the return address register (RETA) and the control-flow context register (CFCT). You can read from or write to RETA and CFCT as a pair with dedicated, 32-bit load and store instructions. Some CPU registers are saved to the stacks (in memory). For fast-return mode operation, see the *TMS320C55x DSP CPU Reference Guide* (SPRU371).

When control is passed to the ISR:

- ☐ The data stack pointer (SP) is decremented by 1 word in the address phase of the pipeline. The status register 2 (ST2\_55) content is pushed to the top of SP.
- ☐ The system stack pointer (SSP) is decremented by 1 word in the address phase of the pipeline. The 7 higher bits of status register 0 (ST0\_55) concatenated with 9 zeroes are pushed to the top of SSP.
- The SP is decremented by 1 word in the access phase of the pipeline. The status register 1 (ST1 55) content is pushed to the top of SP.
- The SSP is decremented by 1 word in the access phase of the pipeline. The debug status register (DBSTAT) content is pushed to the top of SSP.
- The SP is decremented by 1 word in the read phase of the pipeline. The 16 LSBs of the return address, from the program counter (PC), of the called subroutine are pushed to the top of SP.
- The SSP is decremented by 1 word in the read phase of the pipeline. The loop context bits concatenated with the 8 MSBs of the return address are pushed to the top of SSP.
- ☐ The PC is loaded with the ISR program address. The active control flow execution context flags are cleared.

#### System Stack (SSP)

#### After $\rightarrow$ SSP = x - 3(Loop bits):PC(23-16) Save SSP = x - 2**DBSTAT** SSP = x - 1ST0\_55(15-9) SSP = xPreviously saved data Before $\rightarrow$ Save

#### Data Stack (SP)

After 
$$\rightarrow$$
 SP = y - 3 PC(15-0)

Save SP = y - 2 ST1\_55

SP = y - 1 ST2\_55

Before  $\rightarrow$  SP = y

Save Previously saved data

Affected by **Status Bits** none

> Affects none

Repeat This instruction cannot be repeated.

See Also See the following other related instructions:

- ☐ Return from Interrupt
- Software Interrupt

| Syntax  | Description                                                                                              |
|---------|----------------------------------------------------------------------------------------------------------|
| trap(5) | Program control is passed to the specified interrupt service routine. The interrupt vector address is    |
|         | defined by the content of an interrupt vector pointer (IVPD) combined with the unsigned 5-bit value (5). |

SQR

Square

## **Syntax Characteristics**

| No. | Syntax                              | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|-------------------------------------|------------------------|------|--------|----------|
| [1] | ACy = rnd(ACx * ACx)                | Yes                    | 2    | 1      | Х        |
| [2] | ACx = rnd(Smem * Smem)[, T3 = Smem] | No                     | 3    | 1      | X        |

| Description | This instruction performs a multiplication in the D-unit MAC. The input operands of the multiplier are:    |
|-------------|------------------------------------------------------------------------------------------------------------|
|             | <ul><li>□ ACx(32–16)</li><li>□ the content of a memory (Smem) location, sign extended to 17 bits</li></ul> |
| Status Bits | Affected by FRCT, M40, RDM, SATD, SMUL                                                                     |
|             | Affects ACOVx, ACOVy                                                                                       |
| See Also    | See the following other related instructions:                                                              |
|             | ☐ Multiply                                                                                                 |
|             | ☐ Square and Accumulate                                                                                    |
|             | ☐ Square and Subtract                                                                                      |
|             | ☐ Square Distance                                                                                          |

## Square

## **Syntax Characteristics**

| No.         | Syntax         |                                                                                                                    |               |                                          | Parallel<br>Enable Bit | Size     | Cycles       | Pipeline    |
|-------------|----------------|--------------------------------------------------------------------------------------------------------------------|---------------|------------------------------------------|------------------------|----------|--------------|-------------|
| [1]         | ACy = rnd(ACx) | * AC                                                                                                               | x)            |                                          | Yes                    | 2        | 1            | Χ           |
| Opcod       | e              |                                                                                                                    |               |                                          | 010                    | 01 01    | .0E DDS      | S 100%      |
| Opera       | nds            | AC                                                                                                                 | x, ACy        |                                          |                        |          |              |             |
| Description |                | This instruction performs a multiplication in the D-unit MAC. The input operands of the multiplier are ACx(32–16). |               |                                          |                        |          |              |             |
|             |                |                                                                                                                    | If FRCT =     | 1, the output of the r                   | multiplier is shift    | ed left  | by 1 bit.    |             |
|             |                |                                                                                                                    | Multiplicat   | on overflow detectio                     | n depends on S         | SMUL.    |              |             |
|             |                |                                                                                                                    | The 32-bit    | result of the multiplic                  | cation is sign ex      | ktende   | d to 40 bi   | ts.         |
|             |                |                                                                                                                    | •             | is performed accordithe instruction.     | ing to RDM, if the     | ne opti  | onal rnd     | keyword is  |
|             |                |                                                                                                                    |               | detection depends of accumulator overflo |                        |          |              | ected, the  |
|             |                |                                                                                                                    | When an o     | overflow is detected,                    | the accumulato         | r is sat | turated a    | ccording to |
|             |                | Со                                                                                                                 | mpatibility   | with C54x devices                        | (C54CM=1)              |          |              |             |
|             |                | Wh                                                                                                                 | nen this inst | ruction is executed v                    | vith M40 = 0, co       | mpatik   | oility is er | sured.      |
| Status      | Bits           | Aff                                                                                                                | ected by      | FRCT, M40, RDM,                          | SATD, SMUL             |          |              |             |
|             |                | Aff                                                                                                                | ects          | ACOVy                                    |                        |          |              |             |
| Repea       | t              | Thi                                                                                                                | s instruction | n can be repeated.                       |                        |          |              |             |
| Examp       | ole            |                                                                                                                    |               |                                          |                        |          |              |             |

| Syntax          | Description                                                    |
|-----------------|----------------------------------------------------------------|
| AC0 = AC1 * AC1 | The content of AC1 is squared and the result is stored in AC0. |

## Square

#### **Syntax Characteristics**

| -      | Characteristic |                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | Parallel           |          |              |            |
|--------|----------------|------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|----------|--------------|------------|
| No.    | Syntax         |                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | Enable Bit         | Size     | Cycles       | Pipeline   |
| [2]    | ACx = rnd(Sme  | em * Smem)[, T3 =                                                            | = Smem]                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | No                 | 3        | 1            | Х          |
| Opcod  | le             |                                                                              | 110                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | 01 0011 AA         | AA AZ    | AAI   U%I    | DD 10xx    |
| Opera  | nds            | ACx, Smem                                                                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                    |          |              |            |
| Descri | ption          |                                                                              | on performs a multiple multiplier are the confideration of the confidera |                    |          |              |            |
|        |                | ☐ If FRCT =                                                                  | 1, the output of the n                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | nultiplier is shif | ted left | by 1 bit.    |            |
|        |                | Multiplicat                                                                  | tion overflow detectio                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | n depends on S     | SMUL.    |              |            |
|        |                | ☐ The 32-bi                                                                  | t result of the multiplic                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | cation is sign e   | xtende   | d to 40 b    | ts.        |
|        |                |                                                                              | is performed accordi                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | ng to RDM, if t    | he opti  | onal rnd     | keyword is |
|        |                |                                                                              | detection depends on accumulator overflo                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                    |          |              | ected, the |
|        |                | ☐ When an overflow is detected, the accumulator is saturated according SATD. |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                    |          |              |            |
|        |                |                                                                              | n provides the option porary register T3.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | to store the 16    | 6-bit da | ta memo      | ry operand |
|        |                | Compatibility                                                                | / with C54x devices                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | (C54CM = 1)        |          |              |            |
|        |                | When this ins                                                                | truction is executed w                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | vith M40 = 0, co   | ompatil  | oility is er | nsured.    |
| Status | Bits           | Affected by                                                                  | FRCT, M40, RDM,                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | SATD, SMUL         |          |              |            |
|        |                | Affects                                                                      | ACOVx                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                    |          |              |            |
| Repea  | t              | This instructio                                                              | n can be repeated.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                    |          |              |            |
| Examp  | ole            |                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                    |          |              |            |
|        |                | 1                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                    |          |              |            |

| Syntax            | Description                                                              |
|-------------------|--------------------------------------------------------------------------|
| AC0 = *AR3 * *AR3 | The content addressed by AR3 is squared and the result is stored in AC0. |

## **SQA**

## Square and Accumulate

☐ Square and Subtract

## **Syntax Characteristics**

| No. | Syntax                                      | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|---------------------------------------------|------------------------|------|--------|----------|
| [1] | ACy = rnd(ACy + (ACx * ACx))                | Yes                    | 2    | 1      | X        |
| [2] | ACy = rnd(ACx + (Smem * Smem)) [,T3 = Smem] | No                     | 3    | 1      | X        |

| Description |                   | performs a multiplication and an accumulation in the D-unit operands of the multiplier are: |
|-------------|-------------------|---------------------------------------------------------------------------------------------|
|             | ☐ ACx(32–16       | s)<br>t of a memory (Smem) location, sign extended to 17 bits                               |
| Status Bits | Affected by       | FRCT, M40, RDM, SATD, SMUL                                                                  |
|             | Affects           | ACOVx, ACOVy                                                                                |
| See Also    | See the following | ng other related instructions:                                                              |
|             | ☐ Multiply an     | d Accumulate                                                                                |
|             | ☐ Square          |                                                                                             |
|             | ☐ Square Dis      | stance                                                                                      |

0101 010E DDSS 001%

#### Square and Accumulate

#### **Syntax Characteristics**

| No. | Syntax                       | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|------------------------------|------------------------|------|--------|----------|
| [1] | ACy = rnd(ACy + (ACx * ACx)) | Yes                    | 2    | 1      | Х        |

**Opcode** 

Description

**Operands** ACx, ACy

> This instruction performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are ACx(32–16).

- ☐ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- ☐ The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- ☐ When an addition overflow is detected, the accumulator is saturated according to SATD.

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

**Status Bits** Affected by FRCT, M40, RDM, SATD, SMUL

> Affects **ACOVy**

Repeat This instruction can be repeated.

| Syntax                  | Description                                                                                |
|-------------------------|--------------------------------------------------------------------------------------------|
| AC0 = AC0 + (AC1 * AC1) | The content of AC1 squared is added to the content of AC0 and the result is stored in AC0. |

## Square and Accumulate

## **Syntax Characteristics**

| No.         | Syntax         |                                                                                                                                                                                               |                |                                        | E        | Parallel<br>Enable Bit | Size    | Cycles       | Pipeline   |
|-------------|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|----------------------------------------|----------|------------------------|---------|--------------|------------|
| [2]         | ACy = rnd(ACx) | + <b>(</b> Sı                                                                                                                                                                                 | mem * Smem     | ) [,T3 = Smem]                         |          | No                     | 3       | 1            | Х          |
| Opcod       | e              |                                                                                                                                                                                               |                | 1:                                     | 101 0    | 0010   AAA             | A AA    | AI U%D       | D 10SS     |
| Operar      | nds            | AC                                                                                                                                                                                            | ACx, ACy, Smem |                                        |          |                        |         |              |            |
| Description |                | This instruction performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of a memory (Smem) location, sign extended to 17 bits. |                |                                        |          |                        |         |              |            |
|             |                |                                                                                                                                                                                               | If FRCT = 1    | I, the output of the                   | multip   | lier is shift          | ed left | by 1 bit.    |            |
|             |                |                                                                                                                                                                                               | Multiplication | on overflow detect                     | ion dep  | oends on S             | SMUL.   |              |            |
|             |                |                                                                                                                                                                                               |                | esult of the multiplice accumulator AC |          | is sign exte           | ended t | o 40 bits    | and added  |
|             |                |                                                                                                                                                                                               | -              | s performed accor<br>he instruction.   | ding to  | RDM, if th             | ne opti | onal rnd l   | keyword is |
|             |                |                                                                                                                                                                                               |                | erflow detection d                     | •        |                        |         |              |            |
|             |                |                                                                                                                                                                                               | When an a      | addition overflow o SATD.              | is dete  | ected, the             | accum   | nulator is   | saturated  |
|             |                |                                                                                                                                                                                               |                | provides the option                    | on to st | ore the 16             | -bit da | ta memoi     | y operand  |
|             |                | Со                                                                                                                                                                                            | mpatibility    | with C54x device                       | es (C54  | ICM = 1)               |         |              |            |
|             |                | Wr                                                                                                                                                                                            | nen this instr | uction is executed                     | with M   | 140 = 0, co            | mpatik  | oility is en | sured.     |
| Status      | Bits           | Aff                                                                                                                                                                                           | ected by       | FRCT, M40, RDN                         | И, SAT   | D, SMUL                |         |              |            |
|             |                | Aff                                                                                                                                                                                           | ects           | ACOVy                                  |          |                        |         |              |            |
| Repeat      | :              | Thi                                                                                                                                                                                           | is instruction | can be repeated.                       |          |                        |         |              |            |
| Examp       | le             |                                                                                                                                                                                               |                |                                        |          |                        |         |              |            |

| Syntax                    | Description                                                                                          |
|---------------------------|------------------------------------------------------------------------------------------------------|
| AC0 = AC1 + (*AR3 * *AR3) | The content addressed by AR3 squared is added to the content of AC1 and the result is stored in AC0. |



# Square and Subtract

## **Syntax Characteristics**

| No. | Syntax                                      | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|---------------------------------------------|------------------------|------|--------|----------|
| [1] | ACy = rnd(ACy - (ACx * ACx))                | Yes                    | 2    | 1      | X        |
| [2] | ACy = rnd(ACx - (Smem * Smem))[, T3 = Smem] | No                     | 3    | 1      | Х        |

| Description | This instruction performs a multiplication and a subtraction in the D-unit MAC The input operands of the multiplier are: |  |  |  |  |
|-------------|--------------------------------------------------------------------------------------------------------------------------|--|--|--|--|
|             | <ul><li>☐ ACx(32–16)</li><li>☐ the content of a memory (Smem) location, sign extended to 17 bits</li></ul>               |  |  |  |  |
| Status Bits | Affected by FRCT, M40, RDM, SATD, SMUL                                                                                   |  |  |  |  |
|             | Affects ACOVx, ACOVy                                                                                                     |  |  |  |  |
| See Also    | See the following other related instructions:                                                                            |  |  |  |  |
|             | ☐ Multiply and Subtract                                                                                                  |  |  |  |  |
|             | ☐ Square                                                                                                                 |  |  |  |  |
|             | ☐ Square and Accumulate                                                                                                  |  |  |  |  |
|             | ☐ Square Distance                                                                                                        |  |  |  |  |

### Square and Subtract

#### **Syntax Characteristics**

| No. | Syntax                       | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|------------------------------|------------------------|------|--------|----------|
| [1] | ACy = rnd(ACy - (ACx * ACx)) | Yes                    | 2    | 1      | Χ        |

**Opcode** 

0101 010E DDSS 010%

**Operands** 

ACx, ACy

**Description** 

This instruction performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are ACx(32–16).

- ☐ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- ☐ The 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACy.
- ☐ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- ☐ When an overflow is detected, the accumulator is saturated according to SATD.

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

**Status Bits** 

Affected by

FRCT, M40, RDM, SATD, SMUL

Affects

ACOVy

Repeat

This instruction can be repeated.

| Syntax                  | Description                                                                                       |
|-------------------------|---------------------------------------------------------------------------------------------------|
| AC1 = AC1 - (AC0 * AC0) | The content of AC0 squared is subtracted from the content of AC1 and the result is stored in AC1. |

# Square and Subtract

## **Syntax Characteristics**

| No.    | Syntax        |                  |                                                                               | Parallel<br>Enable Bit | Size     | Cycles       | Pipeline    |
|--------|---------------|------------------|-------------------------------------------------------------------------------|------------------------|----------|--------------|-------------|
| [2]    | ACy = rnd(ACx | - (Smem * Sme    | m <b>)</b> )[, T3 = Smem]                                                     | No                     | 3        | 1            | Χ           |
| Opcod  | le            |                  | 1101                                                                          | 0010 AA                | AA AA    | AAI U%I      | DD 11SS     |
| Opera  | nds           | ACx, ACy, Sn     | nem                                                                           |                        |          |              |             |
| Descri | iption        | The input op     | on performs a multiplicat<br>erands of the multiplier<br>extended to 17 bits. |                        |          |              |             |
|        |               | ☐ If FRCT =      | = 1, the output of the mu                                                     | ultiplier is shift     | ted left | by 1 bit.    |             |
|        |               | Multiplica       | tion overflow detection                                                       | depends on S           | SMUL.    |              |             |
|        |               |                  | oit result of the multiplic<br>and from the source accur                      | _                      | exten    | ded to 4     | 0 bits and  |
|        |               | _                | g is performed according the instruction.                                     | g to RDM, if the       | he opti  | onal rnd     | keyword is  |
|        |               | <del>_</del>     | detection depends on accumulator overflow                                     |                        |          |              | ected, the  |
|        |               | ☐ When an SATD.  | overflow is detected, th                                                      | e accumulato           | or is sa | turated a    | ccording to |
|        |               |                  | on provides the option to porary register T3.                                 | o store the 16         | -bit da  | ta memo      | ry operand  |
|        |               | Compatibilit     | y with C54x devices (C                                                        | C54CM = 1)             |          |              |             |
|        |               | When this ins    | struction is executed with                                                    | h M40 = 0, cc          | mpatil   | oility is er | sured.      |
| Status | Bits          | Affected by      | FRCT, M40, RDM, S                                                             | ATD, SMUL              |          |              |             |
|        |               | Affects          | ACOVy                                                                         |                        |          |              |             |
| Repea  | t             | This instruction | on can be repeated.                                                           |                        |          |              |             |
| Examp  | ole           |                  |                                                                               |                        |          |              |             |
| Syntax | x             | Descrip          | otion                                                                         |                        |          |              |             |

| Syntax                                | Description                                                                                                 |
|---------------------------------------|-------------------------------------------------------------------------------------------------------------|
| · · · · · · · · · · · · · · · · · · · | The content addressed by AR3 squared is subtracted from the content of AC1 and the result is stored in AC0. |

#### **SQDST**

#### Square Distance

#### **Syntax Characteristics**

| No. | Syntax                      | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|-----------------------------|------------------------|------|--------|----------|
| [1] | sqdst(Xmem, Ymem, ACx, ACy) | No                     | 4    | 1      | Χ        |

**Opcode** 

1000 0110 XXXM MMYY YMMM DDDD 1110 xxn%

**Operands** 

ACx, ACy, Xmem, Ymem

**Description** 

This instruction performs two parallel operations: multiply and accumulate (MAC), and subtract:

$$ACy = ACy + (ACx * ACx),$$
  
 $ACx = (Xmem << #16) - (Ymem << #16)$ 

The first operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are ACx(32–16).

- ☐ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.
- ☐ Multiplication overflow detection depends on SMUL.
- ☐ The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- ☐ When an addition overflow is detected, the accumulator is saturated according to SATD.

The second operation subtracts the content of data memory operand Ymem, shifted left 16 bits, from the content of data memory operand Xmem, shifted left 16 bits.

- ☐ The operation is performed on 40 bits in the D-unit ALU.
- ☐ Input operands are sign extended to 40 bits according to SXMD.
- ☐ The shift operation is equivalent to the signed shift instruction.
- Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.
- ☐ When an overflow is detected, the accumulator is saturated according to SATD.

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, during the subtraction an intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.

Status Bits Affected by C54CM, FRCT, M40, SATD, SMUL, SXMD

Affects ACOVx, ACOVy, CARRY

**Repeat** This instruction can be repeated.

**See Also** See the following other related instructions:

☐ Absolute Distance

☐ Square

Square and Accumulate

☐ Square and Subtract

| Syntax                      | Description                                                                                                                                                                                                                                              |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| sqdst(*AR0, *AR1, AC0, AC1) | The content of AC0 squared is added to the content of AC1 and the result is stored in AC1. The content addressed by AR1 shifted left by 16 bits is subtracted from the content addressed by AR0 shifted left by 16 bits and the result is stored in AC0. |

| Before |              | After |              |
|--------|--------------|-------|--------------|
| AC0    | FF ABCD 0000 | AC0   | FF FFAB 0000 |
| AC1    | 00 0000 0000 | AC1   | 00 1BB1 8229 |
| *AR0   | 0055         | *ARO  | 0055         |
| *AR1   | AA00         | *AR1  | 00AA         |
| ACOV0  | 0            | ACOV0 | 0            |
| ACOV1  | 0            | ACOV1 | 0            |
| CARRY  | 0            | CARRY | 0            |
| FRCT   | 0            | FRCT  | 0            |

## MOV

### Store Accumulator Content to Memory

### **Syntax Characteristics**

| No.  | Syntax                                                             | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|------|--------------------------------------------------------------------|------------------------|------|--------|----------|
| [1]  | Smem = <b>HI(</b> ACx <b>)</b>                                     | No                     | 2    | 1      | Х        |
| [2]  | Smem = HI(rnd(ACx))                                                | No                     | 3    | 1      | X        |
| [3]  | Smem = LO(ACx << Tx)                                               | No                     | 3    | 1      | X        |
| [4]  | Smem = HI(rnd(ACx << Tx))                                          | No                     | 3    | 1      | X        |
| [5]  | Smem = LO(ACx << #SHIFTW)                                          | No                     | 3    | 1      | X        |
| [6]  | Smem = <b>HI(</b> ACx << <b>#</b> SHIFTW <b>)</b>                  | No                     | 3    | 1      | X        |
| [7]  | Smem = HI(rnd(ACx << #SHIFTW))                                     | No                     | 4    | 1      | X        |
| [8]  | Smem = HI(saturate(uns(rnd(ACx))))                                 | No                     | 3    | 1      | X        |
| [9]  | Smem = HI(saturate(uns(rnd(ACx << Tx))))                           | No                     | 3    | 1      | X        |
| [10] | Smem = HI(saturate(uns(rnd(ACx << #SHIFTW))))                      | No                     | 4    | 1      | X        |
| [11] | dbl(Lmem) = ACx                                                    | No                     | 3    | 1      | X        |
| [12] | dbl(Lmem) = saturate(uns(ACx))                                     | No                     | 3    | 1      | X        |
| [13] | HI(Lmem) = HI(ACx) >> #1,<br>LO(Lmem) = LO(ACx) >> #1              | No                     | 3    | 1      | Х        |
| [14] | Xmem = <b>LO(</b> ACx <b>)</b> ,<br>Ymem = <b>HI(</b> ACx <b>)</b> | No                     | 3    | 1      | Х        |

**Description** This instruction stores the content of the selected accumulator (ACx) to a

memory (Smem) location, to a data memory operand (Lmem), or to dual data

memory operands (Xmem and Ymem).

Status Bits Affected by C54CM, RDM, SXMD

Affects none

| See Also | Se | e the following other related instructions:                                    |
|----------|----|--------------------------------------------------------------------------------|
|          |    | Addition with Parallel Store Accumulator Content to Memory                     |
|          |    | Load Accumulator from Memory with Parallel Store Accumulator Content to Memory |
|          |    | Load Accumulator, Auxiliary, or Temporary Register from Memory                 |
|          |    | Multiply and Accumulate with Parallel Store Accumulator Content to Memory      |
|          |    | Multiply and Subtract with Parallel Store Accumulator Content to Memory        |
|          |    | Multiply with Parallel Store Accumulator Content to Memory                     |
|          |    | Store Accumulator Pair Content to Memory                                       |
|          |    | Store Accumulator, Auxiliary, or Temporary Register Content to Memory          |
|          |    | Store Auxiliary or Temporary Register Pair Content to Memory                   |
|          |    | Subtraction with Parallel Store Accumulator Content to Memory                  |

#### **Syntax Characteristics**

| [1] Smem = <b>HI(</b> ACx <b>)</b> No 2 1 X | No. | Syntax         | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|---------------------------------------------|-----|----------------|------------------------|------|--------|----------|
|                                             | [1] | Smem = HI(ACx) | No                     | 2    | 1      | Х        |

Opcode 1011 11SS AAAA AAAI

Operands ACx, Smem

**Description** This instruction stores the high part of the accumulator, ACx(31-16), to the

memory (Smem) location. The store operation to the memory location uses a dedicated path independent of the D-unit ALU, the D-unit shifter, and the D-unit

MACs.

Status Bits Affected by none

Affects none

**Repeat** This instruction can be repeated.

| Syntax         | Description                                                           |
|----------------|-----------------------------------------------------------------------|
| *AR3 = HI(AC0) | The content of AC0(31–16) is stored at the location addressed by AR3. |

#### **Syntax Characteristics**

| No. | Syntax              | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|---------------------|------------------------|------|--------|----------|
| [2] | Smem = HI(rnd(ACx)) | No                     | 3    | 1      | Х        |

## Opcode

| 1110 1000 | AAAA AAAI | SSxx x0x%

## Operands

ACx, Smem

#### Description

This instruction stores the high part of the accumulator, ACx(31-16), to the memory (Smem) location. Rounding is performed in the D-unit shifter according to RDM, if the optional rnd keyword is applied to the input operand.

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with C54CM = 1, overflow detection at the output of the shifter consists of checking if the sign of the input operand is identical to the most-significant bits of the 40-bit result of the shift and round operation:

☐ If the SST bit = 1 and the SXMD bit = 0, then the saturate, rnd, and uns keywords are applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:

Smem = HI(saturate(uns(rnd(ACx))))

☐ If the SST bit = 1 and the SXMD bit = 1, then only the saturate and rnd keywords are applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:

Smem = HI(saturate(rnd(ACx)))

- ☐ If the optional uns keyword is applied to the input operand, then bits 39–32 of the result are compared to 0.
- If the optional uns keyword is not applied to the input operand, then bits 39–31 of the result are compared to bit 39 of the input operand and SXMD.

#### **Status Bits**

Affected by

C54CM, RDM, SST, SXMD

Affects

none

#### Repeat

This instruction can be repeated.

| Syntax              | Description                                                                       |
|---------------------|-----------------------------------------------------------------------------------|
| *AR3 = HI(rnd(AC0)) | The content of AC0(31–16) is rounded and stored at the location addressed by AR3. |

#### **Syntax Characteristics**

| No. | Syntax               | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|----------------------|------------------------|------|--------|----------|
| [3] | Smem = LO(ACx << Tx) | No                     | 3    | 1      | X        |

Opcode

1110 0111 AAAA AAAI SSss 00xx

**Operands** 

ACx, Smem, Tx

#### **Description**

This instruction shifts the accumulator, ACx, by the content of Tx and stores the low part of the accumulator, ACx(15–0), to the memory (Smem) location. If the 16-bit value in Tx is not within -32 to +31, the shift is saturated to -32 or +31 and the shift is performed with this value. The input operand is shifted in the D-unit shifter according to SXMD.

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with C54CM = 1, the 6 LSBs of Tx determine the shift quantity. The 6 LSBs of Tx define a shift quantity within -32 to +31. When the 16-bit value in Tx is between -32 to -17, a modulo 16 operation transforms the shift quantity to within -16 to -1.

☐ If the SST bit = 1 and the SXMD bit = 0, then the saturate and uns keywords are applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:

Smem = LO(saturate(uns(ACx << Tx)))

☐ If the SST bit = 1 and the SXMD bit = 1, then only the saturate keyword is applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:

Smem = LO(saturate(ACx << Tx))

**Status Bits** 

Affected by

C54CM, RDM, SST, SXMD

Affects

none

#### Repeat

This instruction can be repeated.

| Syntax | Description                                                                                                  |
|--------|--------------------------------------------------------------------------------------------------------------|
| '      | The content of AC0 is shifted by the content of T0 and AC0(15–0) is stored at the location addressed by AR3. |

1110 0111 AAAA AAAI SSss 10x%

#### Store Accumulator Content to Memory

#### **Syntax Characteristics**

| No. | Syntax                    | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|---------------------------|------------------------|------|--------|----------|
| [4] | Smem = HI(rnd(ACx << Tx)) | No                     | 3    | 1      | Х        |

# Opcode Operands

ACx, Smem, Tx

#### Description

This instruction shifts the accumulator, ACx, by the content of Tx and stores high part of the accumulator, ACx(31–16), to the memory (Smem) location. If the 16-bit value in Tx is not within -32 to +31, the shift is saturated to -32 or +31 and the shift is performed with this value. The input operand is shifted in the D-unit shifter according to SXMD. Rounding is performed in the D-unit shifter according to RDM, if the optional rnd keyword is applied to the input operand.

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with C54CM = 1, the 6 LSBs of Tx determine the shift quantity. The 6 LSBs of Tx define a shift quantity within -32 to +31. When the 16-bit value in Tx is between -32 to -17, a modulo 16 operation transforms the shift quantity to within -16 to -1.

☐ If the SST bit = 1 and the SXMD bit = 0, then the saturate, rnd, and uns keywords are applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:

Smem = HI(saturate(uns(rnd(ACx << Tx))))

☐ If the SST bit = 1 and the SXMD bit = 1, then only the saturate and rnd keywords are applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:

Smem = HI(saturate(rnd(ACx << Tx)))

Status Bits

Affected by

C54CM, RDM, SST, SXMD

Affects

none

#### Repeat

This instruction can be repeated.

| Syntax | Description                                                                                                                |
|--------|----------------------------------------------------------------------------------------------------------------------------|
| , ,    | The content of AC0 is shifted by the content of T0, is rounded, and AC0(31–16) is stored at the location addressed by AR3. |

#### **Syntax Characteristics**

| No.   | Syntax                    |      | Parallel<br>Enable B |       | e Cy | cles | Pipeline |
|-------|---------------------------|------|----------------------|-------|------|------|----------|
| [5]   | Smem = LO(ACx << #SHIFTW) |      | No                   | 3     |      | 1    | Х        |
| Opcod | e                         | 1110 | 1001 A               | AAA . | AAAI | SSSH | I IFTW   |

#### **Operands**

ACx, SHIFTW, Smem

#### Description

This instruction shifts the accumulator, ACx, by the 6-bit value, SHIFTW, and stores the low part of the accumulator, ACx(15–0), to the memory (Smem) location. The input operand is shifted by the 6-bit value in the D-unit shifter according to SXMD.

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with C54CM = 1, overflow detection at the output of the shifter consists of checking if the sign of the input operand is identical to the most-significant bits of the 40-bit result of the shift and round operation:

☐ If the SST bit = 1 and the SXMD bit = 0, then the saturate and uns keywords are applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:

Smem = LO(saturate(uns(ACx << #SHIFTW)))

☐ If the SST bit = 1 and the SXMD bit = 1, then only the saturate keyword is applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:

Smem = LO(saturate(ACx << #SHIFTW))

- ☐ If the optional uns keyword is applied to the input operand, then bits 39–32 of the result are compared to 0.
- ☐ If the optional uns keyword is not applied to the input operand, then bits 39–31 of the result are compared to bit 39 of the input operand and SXMD.

Status Bits Affected by C54CM, RDM, SST, SXMD

Affects none

**Repeat** This instruction can be repeated.

| Syntax | Description                                                                                             |
|--------|---------------------------------------------------------------------------------------------------------|
| ,      | The content of AC0 is shifted left by 31 bits and AC0(15–0) is stored at the location addressed by AR3. |

#### **Syntax Characteristics**

| No.   | Syntax                                            |      | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-------|---------------------------------------------------|------|------------------------|------|--------|----------|
| [6]   | Smem = <b>HI(</b> ACx << <b>#</b> SHIFTW <b>)</b> |      | No                     | 3    | 1      | Х        |
| Opcod | e                                                 | 1110 | 1010 AAA               | A AA | AI SSS | H IFTW   |

#### **Operands**

ACx, SHIFTW, Smem

#### **Description**

This instruction shifts the accumulator, ACx, by the 6-bit value, SHIFTW, and stores the high part of the accumulator, ACx(31–16), to the memory (Smem) location. The input operand is shifted by the 6-bit value in the D-unit shifter according to SXMD.

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with C54CM = 1, overflow detection at the output of the shifter consists of checking if the sign of the input operand is identical to the most-significant bits of the 40-bit result of the shift and round operation:

☐ If the SST bit = 1 and the SXMD bit = 0, then the saturate and uns keywords are applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:

Smem = HI(saturate(uns(ACx << #SHIFTW)))

☐ If the SST bit = 1 and the SXMD bit = 1, then only the saturate keyword is applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:

Smem = HI(saturate(ACx << #SHIFTW))

- ☐ If the optional uns keyword is applied to the input operand, then bits 39–32 of the result are compared to 0.
- ☐ If the optional uns keyword is not applied to the input operand, then bits 39–31 of the result are compared to bit 39 of the input operand and SXMD.

Status Bits Affected by

C54CM, RDM, SST, SXMD

Affects none

Repeat

This instruction can be repeated.

| Syntax                | Description                                                                                              |
|-----------------------|----------------------------------------------------------------------------------------------------------|
| *AR3 = HI(AC0 << #31) | The content of AC0 is shifted left by 31 bits and AC0(31–16) is stored at the location addressed by AR3. |

## **Syntax Characteristics**

| No.    | Syntax        |                                                                                                                                                                                                                                                          |                                                                                                                           | Parallel<br>Enable Bit                       | Size                         | Cycles                             | Pipeline                  |
|--------|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------|----------------------------------------------|------------------------------|------------------------------------|---------------------------|
| [7]    | Smem = HI(rne | d(ACx << #SHIFTW                                                                                                                                                                                                                                         | <b>())</b>                                                                                                                | No                                           | 4                            | 1                                  | Х                         |
| Opcod  | е             |                                                                                                                                                                                                                                                          | 1111 1010 AAAA                                                                                                            | AAAI xxs                                     | SH IF                        | TW SSx                             | x x0x%                    |
| Operar | nds           | ACx, SHIFTW,                                                                                                                                                                                                                                             | Smem                                                                                                                      |                                              |                              |                                    |                           |
| Descri | ption         | stores the high location. The ir according to S                                                                                                                                                                                                          | shifts the accumulator, part of the accumulato<br>part operand is shifted<br>KMD. Rounding is performal rnd keyword is ap | r, ACx(31–10<br>by the 6-bit<br>ormed in the | 6), to tl<br>value<br>D-unit | ne memo<br>in the D-<br>shifter ac | ry (Smem)<br>unit shifter |
|        |               | Compatibility                                                                                                                                                                                                                                            | with C54x devices (C                                                                                                      | 54CM = 1)                                    |                              |                                    |                           |
|        |               | When this instruction is executed with C54CM = 1, overflow detection at the output of the shifter consists of checking if the sign of the input operand is identical to the most-significant bits of the 40-bit result of the shift and round operation: |                                                                                                                           |                                              |                              |                                    |                           |
|        |               | keywords<br>keywords s                                                                                                                                                                                                                                   | bit = 1 and the SXMD bare applied to the inselected by the user, win = HI(saturate(uns(rnd                                | struction re<br>th the follow                | gardle<br>ing syr            | ss of the                          |                           |
|        |               | keywords<br>keywords s                                                                                                                                                                                                                                   | bit = 1 and the SXMD<br>are applied to the in<br>selected by the user, wi<br>n = HI(saturate(rnd(AC                       | struction re<br>th the follow                | gardle<br>ing syr            | ss of the                          |                           |
|        |               | ☐ If the optional uns keyword is applied to the input operand, then bits 39–32 of the result are compared to 0.                                                                                                                                          |                                                                                                                           |                                              |                              |                                    |                           |
|        |               |                                                                                                                                                                                                                                                          | nal uns keyword is not<br>e result are compared to                                                                        |                                              | -                            |                                    |                           |
| Status | Bits          | Affected by                                                                                                                                                                                                                                              | C54CM, RDM, SST, S                                                                                                        | SXMD                                         |                              |                                    |                           |
|        |               | ∧ ffo oto                                                                                                                                                                                                                                                | 222                                                                                                                       |                                              |                              |                                    |                           |

Affects

none

## Repeat

This instruction can be repeated.

| Syntax                     | Description                                                                                                           |
|----------------------------|-----------------------------------------------------------------------------------------------------------------------|
| *AR3 = HI(rnd(AC0 << #31)) | The content of AC0 is shifted left by 31 bits, is rounded, and AC0(31–16) is stored at the location addressed by AR3. |

#### Syntax Characteristics

| No.    | Syntax                                  |                | Parallel<br>Enable Bit | Size    | Cycles  | Pipeline    |
|--------|-----------------------------------------|----------------|------------------------|---------|---------|-------------|
| [8]    | Smem = HI(saturate(uns(rnd(ACx))))      |                | No                     | 3       | 1       | Х           |
| Opcod  | e                                       | 1110           | 1000 AAA               | AA AA   | AI SSx  | x x1u%      |
| Opera  | nds ACx, Smem                           |                |                        |         |         |             |
| Descri | <b>ption</b> This instruction stores th | e high part of | f the accumi           | ulator, | ACx(31- | 16), to the |

- memory (Smem) location.
- $\square$  When the C54CM bit = 0 or the SST bit = 0, the saturate and uns keywords are optional and can be applied or not.
- ☐ Input operands are considered signed or unsigned according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is considered unsigned.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is considered signed.
- ☐ If the optional rnd keyword is applied to the input operand, rounding is performed in the D-unit shifter according to RDM.
- ☐ When a rounding overflow is detected and if the optional saturate keyword is applied to the input operand, the 40-bit output of the operation is saturated:
  - If the optional uns keyword is applied to the input operand, saturation value is 00 FFFF FFFFh.
  - If the optional uns keyword is not applied, saturation values are 00 7FFF FFFFh (positive overflow) or FF 8000 0000h (negative overflow).

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with C54CM = 1, overflow detection at the output of the shifter consists of checking if the sign of the input operand is identical to the most-significant bits of the 40-bit result of the round operation:

 $\Box$  If the SST bit = 1 and the SXMD bit = 0, then the saturate, rnd, and uns keywords are applied to the instruction regardless of the optional keywords selected by the user.

| If the SST bit = 1 and the SXMD bit = 1, then only the saturate and rnd |
|-------------------------------------------------------------------------|
| keywords are applied to the instruction regardless of the optional      |
| keywords selected by the user, with the following syntax:               |
| Smem = HI(saturate(rnd(ACx)))                                           |

☐ If the optional uns keyword is applied to the input operand, then bits 39–32 of the result are compared to 0.

☐ If the optional uns keyword is not applied to the input operand, then bits 39-31 of the result are compared to bit 39 of the input operand and SXMD.

**Status Bits** Affected by C54CM, RDM, SST, SXMD

> Affects none

Repeat This instruction can be repeated.

| Syntax | Description                                               |
|--------|-----------------------------------------------------------|
|        | The unsigned content of AC0 is rounded, is saturated, and |
|        | AC0(31–16) is stored at the location addressed by AR3.    |

#### **Syntax Characteristics**

| No.   | Syntax                                   |      | Parallel<br>Enable Bit | Size  | Cycles  | Pipeline |
|-------|------------------------------------------|------|------------------------|-------|---------|----------|
| [9]   | Smem = HI(saturate(uns(rnd(ACx << Tx)))) |      | No                     | 3     | 1       | Х        |
| Opcod | e                                        | 1110 | 0111 AA                | AA AA | AAI SSs | ss 11u%  |

#### **Operands**

ACx, Smem, Tx

#### **Description**

This instruction shifts the accumulator, ACx, by the content of Tx and stores the high part of the accumulator, ACx(31–16), to the memory (Smem) location. If the 16-bit value in Tx is not within -32 to +31, the shift is saturated to -32 or +31 and the shift is performed with this value.

- ☐ When the C54CM bit = 0 or the SST bit = 0, the saturate and uns keywords are optional and can be applied or not.
- ☐ Input operands are considered signed or unsigned according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is considered unsigned.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is considered signed.
- ☐ The input operand is shifted in the D-unit shifter according to SXMD.
- ☐ When shifting, the sign position of the input operand is compared to the shift quantity.
  - If the optional uns keyword is applied to the input operand, this comparison is performed against bit 32 of the shifted operand.
  - If the optional uns keyword is not applied, this comparison is performed against bit 31 of the shifted operand that is considered signed (the sign is defined by bit 39 of the input operand and SXMD).
  - An overflow is generated accordingly.
- ☐ If the optional rnd keyword is applied to the input operand, rounding is performed in the D-unit shifter according to RDM.
- ☐ When a shift or rounding overflow is detected and if the optional saturate keyword is applied to the input operand, the 40-bit output of the operation is saturated:
  - If the optional uns keyword is applied to the input operand, saturation value is 00 FFFF FFFFh.

■ If the optional uns keyword is not applied, saturation values are 00 7FFF FFFFh (positive overflow) or FF 8000 0000h (negative overflow).

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with C54CM = 1:

- ☐ If the SST bit = 1 and the SXMD bit = 0, then the saturate, rnd, and uns keywords are applied to the instruction regardless of the optional keywords selected by the user.
- ☐ If the SST bit = 1 and the SXMD bit = 1, then only the saturate and rnd keywords are applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:

Smem = HI(saturate(rnd(ACx << Tx)))

- Overflow detection at the output of the shifter consists of checking if the sign of the input operand is identical to the most-significant bits of the 40-bit result of the shift and round operation.
  - If the optional uns keyword is applied to the input operand, then bits 39–32 of the result are compared to 0.
  - If the optional uns keyword is not applied to the input operand, then bits 39–31 of the result are compared to bit 39 of the input operand and SXMD.
- □ The 6 LSBs of Tx determine the shift quantity. The 6 LSBs of Tx define a shift quantity within -32 to +31. When the 16-bit value in Tx is between -32 to -17, a modulo 16 operation transforms the shift quantity to within -16 to -1.

Status Bits Affected by C54CM, RDM, SST, SXMD

Affects none

**Repeat** This instruction can be repeated.

| Syntax                                   | Description                                                                                                                                       |
|------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| *AR3 = HI(saturate(uns(rnd(AC0 << T0)))) | The unsigned content of AC0 is shifted by the content of T0, is rounded, is saturated, and AC0(31–16) is stored at the location addressed by AR3. |

## **Syntax Characteristics**

| No.      | Syntax |              |                        |           |                                      | Parallel<br>Enable Bit                                | Size    | Cycles     | Pipeline    |
|----------|--------|--------------|------------------------|-----------|--------------------------------------|-------------------------------------------------------|---------|------------|-------------|
| [10]     | Smem = | HI(saturate( | uns(rnd(ACx            | << #SHII  | FTW <b>)</b> )))                     | No                                                    | 4       | 1          | Х           |
| Opcod    | le     |              |                        | 1111      | 1010 AAAA                            | AAAI uxSH                                             | I IFT   | TW SSxx    | x x1x%      |
| Operands |        |              | x, SHIFTW              | Smem      |                                      |                                                       |         |            |             |
| Descri   | ption  | sto          |                        |           |                                      | r, ACx, by the 6<br>or, ACx(31–16)                    |         |            |             |
|          |        |              |                        |           | oit = 0 or the SS<br>an be applied o | ST bit = 0, the sa<br>or not.                         | aturate | and uns    | keywords    |
|          |        |              | Input oper             | ands are  | considered si                        | gned or unsigr                                        | ned ac  | cording t  | o uns.      |
|          |        |              |                        |           | •                                    | applied to the irnsidered unsigr                      |         | oerand, th | ne content  |
|          |        |              |                        | •         | •                                    | s not applied to<br>on is considere                   |         |            | erand, the  |
|          |        |              | The input according    | •         |                                      | y the 6-bit val                                       | ue in   | the D-u    | nit shifter |
|          |        |              | When shift shift quant | -         | sign position of                     | of the input ope                                      | erand i | s compa    | red to the  |
|          |        |              |                        | •         | •                                    | is applied to<br>ainst bit 32 of th                   |         |            |             |
|          |        |              | perfor                 | med aga   | ainst bit 31 of                      | d is not appli<br>the shifted ope<br>bit 39 of the in | erand   | that is co | onsidered   |
|          |        |              | ■ An ove               | erflow is | generated acc                        | ordingly.                                             |         |            |             |
|          |        |              | •                      |           |                                      | plied to the inp                                      | -       | erand, ro  | ounding is  |
|          |        |              |                        | applied   | -                                    | is detected ar erand, the 40-b                        |         | •          |             |

value is 00 FFFF FFFFh.

■ If the optional uns keyword is applied to the input operand, saturation

■ If the optional uns keyword is not applied, saturation values are 00 7FFF FFFFh (positive overflow) or FF 8000 0000h (negative overflow).

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with C54CM = 1, overflow detection at the output of the shifter consists of checking if the sign of the input operand is identical to the most-significant bits of the 40-bit result of the shift and round operation.

| If the SST | bit = | : 1 and th | ne S | SXML  | D bit = $0$ , the | en the satura | ate, | rnd, | and uns  |
|------------|-------|------------|------|-------|-------------------|---------------|------|------|----------|
| keywords   | are   | applied    | to   | the   | instruction       | regardless    | of   | the  | optional |
| keywords   | seled | cted by th | ne u | ıser. |                   |               |      |      |          |

☐ If the SST bit = 1 and the SXMD bit = 1, then only the saturate and rnd keywords are applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:

Smem = HI(saturate(rnd(ACx << #SHIFTW)))

☐ If the optional uns keyword is applied to the input operand, then bits 39–32 of the result are compared to 0.

☐ If the optional uns keyword is not applied to the input operand, then bits 39–31 of the result are compared to bit 39 of the input operand and SXMD.

**Status Bits** 

Affected by

C54CM, RDM, SST, SXMD

Affects

none

Repeat

This instruction can be repeated.

| Syntax | Description                                                                                                                                  |
|--------|----------------------------------------------------------------------------------------------------------------------------------------------|
|        | The unsigned content of AC0 is shifted left by 31 bits, is rounded, is saturated, and AC0(31–16) is stored at the location addressed by AR3. |

### **Syntax Characteristics**

| No.      | Syntax                        |                                                                                                                | Parallel<br>Enable Bit | Size   | Cycles    | Pipeline   |
|----------|-------------------------------|----------------------------------------------------------------------------------------------------------------|------------------------|--------|-----------|------------|
| [11]     | <b>dbl(</b> Lmem <b>)</b> = A | Сх                                                                                                             | No                     | 3      | 1         | Х          |
| Opcode   | •                             | 11                                                                                                             | .10 1011 AAA           | A AA   | AI xxS    | S 10x0     |
| Operan   | ds                            | ACx, Lmem                                                                                                      |                        |        |           |            |
| Descrip  | tion                          | This instruction stores the content memory operand (Lmem). The st a dedicated path independent of D-unit MACs. | ore operation to       | the me | mory loca | ation uses |
| Status I | Bits                          | Affected by none                                                                                               |                        |        |           |            |

Status Bits Affected by none

Affects none

**Repeat** This instruction can be repeated.

| Syntax          | Description                                                                 |
|-----------------|-----------------------------------------------------------------------------|
| dbl(*AR3) = AC0 | The content of AC0 is stored at the locations addressed by AR3 and AR3 + 1. |

#### **Syntax Characteristics**

| No.    | Syntax                         |      | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|--------|--------------------------------|------|------------------------|------|--------|----------|
| [12]   | dbl(Lmem) = saturate(uns(ACx)) |      | No                     | 3    | 1      | Х        |
| Opcode | e                              | 1110 | 1011 AAA               | A AA | AI xxS | S 10u1   |

#### **Operands**

ACx, Lmem

#### Description

This instruction stores the content of the accumulator, ACx(31–0), to the data memory operand (Lmem).

- ☐ When the C54CM bit = 0 or the SST bit = 0, the saturate and uns keywords are optional and can be applied or not.
- ☐ Input operands are considered signed or unsigned according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is considered unsigned.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is considered signed.
- ☐ The 40-bit output of the operation is saturated:
  - If the optional uns keyword is applied to the input operand, saturation value is 00 FFFF FFFFh.
  - If the optional uns keyword is not applied, saturation values are 00 7FFF FFFFh (positive overflow) or FF 8000 0000h (negative overflow).
- ☐ The store operation to the memory location uses the D-unit shifter.

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with C54CM = 1, overflow detection at the output of the shifter consists of checking if the sign of the input operand is identical to the most-significant bits of the 40-bit result of the shift and round operation.

- ☐ If the SST bit = 1 and the SXMD bit = 0, then the saturate and uns keywords are applied to the instruction regardless of the optional keywords selected by the user.
- ☐ If the SST bit = 1 and the SXMD bit = 1, then only the saturate keyword is applied to the instruction regardless of the optional keywords selected by the user.

|             |      | •             | al uns keyword is applied to the input operand, then bits 39–32 tare compared to 0.                                           |
|-------------|------|---------------|-------------------------------------------------------------------------------------------------------------------------------|
|             |      | •             | nal uns keyword is not applied to the input operand, then bits e result are compared to bit 39 of the input operand and SXMD. |
| Status Bits | Affe | ected by      | C54CM, RDM, SST, SXMD                                                                                                         |
|             | Affe | ects          | none                                                                                                                          |
| Repeat      | Thi  | s instruction | can be repeated.                                                                                                              |
|             |      |               |                                                                                                                               |

| Syntax                         | Description                                                                                        |
|--------------------------------|----------------------------------------------------------------------------------------------------|
| dbl(*AR3) = saturate(uns(AC0)) | The unsigned content of AC0 is saturated and stored at the locations addressed by AR3 and AR3 + 1. |

### Store Accumulator Content to Memory

### **Syntax Characteristics**

| No.  | Syntax                                                | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|------|-------------------------------------------------------|------------------------|------|--------|----------|
| [13] | HI(Lmem) = HI(ACx) >> #1,<br>LO(Lmem) = LO(ACx) >> #1 | No                     | 3    | 1      | Х        |

**Opcode** | 1110 1011 | AAAA AAAI | xxSS 1101

Operands ACx, Lmem

**Description**This instruction performs two store operations in parallel and is executed in the D-unit shifter:

☐ The 16 highest bits of the accumulator, ACx(31–16), shifted right by 1 bit (bit 31 is sign extended according to SXMD), are stored to the 16 highest bits of the data memory operand (Lmem).

☐ The 16 lowest bits, ACx(15–0), shifted right by 1 bit (bit 15 is sign extended according to SXMD), are stored to the 16 lowest bits of the data memory

operand (Lmem).

Affects none

**SXMD** 

**Repeat** This instruction can be repeated.

Affected by

### **Example**

**Status Bits** 

| Syntax                   | Description                                                                                                                                                                                         |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| LO(*AR1) = LO(AC0) >> #1 | The content of AC0(31–16), shifted right by 1 bit, is stored at the location addressed by AR1 and the content of AC0(15–0), shifted right by 1 bit, is stored at the location addressed by AR1 + 1. |

| 1000 0000 | XXXM MMYY | YMMM 10SS

### Store Accumulator Content to Memory

### **Syntax Characteristics**

| No.  | Syntax                                            | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|------|---------------------------------------------------|------------------------|------|--------|----------|
| [14] | Xmem = <b>LO</b> (ACx),<br>Ymem = <b>HI</b> (ACx) | No                     | 3    | 1      | Х        |

Operands ACx, Xmem, Ymem

**Description** This instruction performs two store operations in parallel:

☐ The 16 lowest bits of the accumulator, ACx(15–0), are stored to data memory operand Xmem.

☐ The 16 highest bits, ACx(31–16), are stored to data memory operand Ymem.

Status Bits Affected by none

Affects none

**Repeat** This instruction can be repeated.

### **Example**

Opcode

| Syntax             | Description                                                                 |
|--------------------|-----------------------------------------------------------------------------|
| *AR1 = $LO(AC0)$ , | The content of AC0(15–0) is stored at the location addressed by AR1 and the |
| *AR2 = HI(AC0)     | content of AC0(31–16) is stored at the location addressed by AR2.           |

| Before |    |      |      | After |    |      |      |
|--------|----|------|------|-------|----|------|------|
| AC0    | 01 | 4500 | 0030 | AC0   | 01 | 4500 | 0030 |
| AR1    |    |      | 0200 | AR1   |    |      | 0200 |
| AR2    |    |      | 0201 | AR2   |    |      | 0201 |
| 200    |    |      | 3400 | 200   |    |      | 0030 |
| 201    |    |      | 0FD3 | 201   |    |      | 4500 |

### MOV

### Store Accumulator Pair Content to Memory

### **Syntax Characteristics**

| No.         | Syntax          |                                                                           |                                                                                  |                                           | Parallel<br>Enable Bit | Size     | Cycles     | Pipeline  |  |
|-------------|-----------------|---------------------------------------------------------------------------|----------------------------------------------------------------------------------|-------------------------------------------|------------------------|----------|------------|-----------|--|
| [1]         | Lmem = pair(HI( | ACx)                                                                      | )                                                                                |                                           | No                     | 3        | 1          | Х         |  |
| [2]         | Lmem = pair(LO  | (ACx                                                                      | <b>())</b>                                                                       |                                           | No                     | 3        | 1          | Х         |  |
| Description |                 |                                                                           |                                                                                  | n stores the content of data memory opera |                        | ccumu    | lator pair | ; ACx and |  |
| Status      | s Bits          | Aff                                                                       | ected by                                                                         | none                                      |                        |          |            |           |  |
|             |                 | Aff                                                                       | ects                                                                             | none                                      |                        |          |            |           |  |
| See Also    |                 | See the following other related instructions:                             |                                                                                  |                                           |                        |          |            |           |  |
|             |                 | ☐ Addition with Parallel Store Accumulator Content to Memory              |                                                                                  |                                           |                        |          |            |           |  |
|             |                 |                                                                           | ☐ Load Accumulator from Memory with Parallel Store Accumulator Content to Memory |                                           |                        |          |            |           |  |
|             |                 |                                                                           | ☐ Load Accumulator, Auxiliary, or Temporary Register from Memory                 |                                           |                        |          |            |           |  |
|             |                 | ☐ Multiply and Accumulate with Parallel Store Accumulator Content to Mer  |                                                                                  |                                           |                        |          |            | to Memory |  |
|             |                 | ☐ Multiply and Subtract with Parallel Store Accumulator Content to Memory |                                                                                  |                                           |                        |          |            |           |  |
|             |                 | ☐ Multiply with Parallel Store Accumulator Content to Memory              |                                                                                  |                                           |                        |          |            |           |  |
|             |                 | ☐ Store Accumulator Content to Memory                                     |                                                                                  |                                           |                        |          |            |           |  |
|             |                 |                                                                           | Store Accu                                                                       | ımulator, Auxiliary, o                    | r Temporary Re         | gister ( | Content to | o Memory  |  |
|             |                 |                                                                           | Store Auxi                                                                       | liary or Temporary R                      | egister Pair Con       | tent to  | Memory     | ,         |  |

☐ Subtraction with Parallel Store Accumulator Content to Memory

### Store Accumulator Pair Content to Memory

### **Syntax Characteristics**

| No.     | Syntax      |                                 |                                                                                                                                                                                                                                            | Parallel<br>Enable Bit | Size  | Cycles | Pipeline |  |  |
|---------|-------------|---------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|-------|--------|----------|--|--|
| [1]     | Lmem = pair | (HI(ACx))                       |                                                                                                                                                                                                                                            | No                     | 3     | 1      | Х        |  |  |
| Opcod   | е           |                                 | 1110                                                                                                                                                                                                                                       | 1011 AA                | AA AA | AI xxS | SS 1110  |  |  |
| Operar  | nds         | ACx, Lmem                       |                                                                                                                                                                                                                                            |                        |       |        |          |  |  |
| Descrip | otion       | the 16 highes                   | This instruction stores the 16 highest bits of the accumulator, $ACx(31-16)$ , to the 16 highest bits of the data memory operand (Lmem) and stores the 16 highest bits of $AC(x + 1)$ to the 16 lowest bits of data memory operand (Lmem): |                        |       |        |          |  |  |
|         |             | mory location<br>ne D-unit shif |                                                                                                                                                                                                                                            |                        | •     |        |          |  |  |
|         |             | Valid accu                      | mulators are AC0/AC1                                                                                                                                                                                                                       | and AC2/AC             | 3.    |        |          |  |  |
| Status  | Bits        | Affected by                     | none                                                                                                                                                                                                                                       |                        |       |        |          |  |  |
|         |             | Affects                         | none                                                                                                                                                                                                                                       |                        |       |        |          |  |  |

### Example

Repeat

| Syntax                | Description                                                                                                                                                                 |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| *AR1+ = pair(HI(AC0)) | The content of AC0(31–16) is stored at the location addressed by AR1 and the content of AC1(31–16) is stored at the location addressed by AR1 + 1. AR1 is incremented by 2. |

This instruction can be repeated.

| Before |    |      |      | After |    |      |      |
|--------|----|------|------|-------|----|------|------|
| AC0    | 01 | 4500 | 0030 | AC0   | 01 | 4500 | 0030 |
| AC1    | 03 | 5644 | F800 | AC1   | 03 | 5644 | F800 |
| AR1    |    |      | 0200 | AR1   |    |      | 0202 |
| 200    |    |      | 3400 | 200   |    |      | 4500 |
| 201    |    |      | 0FD3 | 201   |    |      | 5644 |

### Store Accumulator Pair Content to Memory

### **Syntax Characteristics**

| No. | Syntax               | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|----------------------|------------------------|------|--------|----------|
| [2] | Lmem = pair(LO(ACx)) | No                     | 3    | 1      | Х        |

**Opcode** | 1110 1011 | AAAA AAAI | xxSS 1111

Operands ACx, Lmem

**Description**This instruction stores the 16 lowest bits of the accumulator, ACx(15–0), to the 16 highest bits of the data memory operand (Lmem) and stores the 16 lowest

bits of AC(x + 1) to the 16 lowest bits of data memory operand (Lmem):

☐ The store operation to the memory location uses a dedicated path independent of the D-unit ALU, the D-unit shifter, and the D-unit MACs.

☐ Valid accumulators are AC0/AC1 and AC2/AC3.

Status Bits Affected by none

Affects none

**Repeat** This instruction can be repeated.

| Syntax               | Description                                                                 |
|----------------------|-----------------------------------------------------------------------------|
| *AR3 = pair(LO(AC0)) | The content of AC0(15–0) is stored at the location addressed by AR3 and the |
|                      | content of AC1(15–0) is stored at the location addressed by AR3 + 1.        |



Store Accumulator, Auxiliary, or Temporary Register Content to Memory

**Parallel** 

### **Syntax Characteristics**

| No.         | Syntax        |                       |                                                                                |                                      | Enable Bit     | Size     | Cycles     | Pipeline    |  |
|-------------|---------------|-----------------------|--------------------------------------------------------------------------------|--------------------------------------|----------------|----------|------------|-------------|--|
| [1]         | Smem = src    | Smem = src            |                                                                                |                                      |                | 2        | 1          | Х           |  |
| [2]         | high_byte(Sme | high_byte(Smem) = src |                                                                                |                                      |                | 3        | 1          | X           |  |
| [3]         | low_byte(Sme  | m <b>)</b> =          | src                                                                            |                                      | No             | 3        | 1          | Х           |  |
| Description |               |                       | is instructior<br>emory (Smer                                                  | n stores the content of m) location. | f the selected | source   | e (src) re | gister to a |  |
| Status      | Bits          | Aff                   | ected by                                                                       | none                                 |                |          |            |             |  |
|             |               | Aff                   | ects none                                                                      |                                      |                |          |            |             |  |
| See Also    |               | Se                    | ee the following other related instructions:                                   |                                      |                |          |            |             |  |
|             |               |                       | Addition with Parallel Store Accumulator Content to Memory                     |                                      |                |          |            |             |  |
|             |               |                       | Load Accumulator from Memory with Parallel Store Accumulator Content to Memory |                                      |                |          |            |             |  |
|             |               |                       | Load Accumulator, Auxiliary, or Temporary Register from Memory                 |                                      |                |          |            |             |  |
|             |               |                       | Multiply and Accumulate with Parallel Store Accumulator Content to Memory      |                                      |                |          |            |             |  |
|             |               |                       | Multiply and Subtract with Parallel Store Accumulator Content to Memory        |                                      |                |          |            |             |  |
|             |               |                       | Multiply with Parallel Store Accumulator Content to Memory                     |                                      |                |          |            |             |  |
|             |               |                       | Store Accumulator Content to Memory                                            |                                      |                |          |            |             |  |
|             |               |                       | Store Accu                                                                     | mulator Pair Content                 | to Memory      |          |            |             |  |
|             |               |                       | Store Auxil                                                                    | liary or Temporary Reg               | gister Pair Co | ntent to | Memory     | 1           |  |
|             |               |                       |                                                                                |                                      |                |          |            |             |  |

☐ Subtraction with Parallel Store Accumulator Content to Memory

### Store Accumulator, Auxiliary, or Temporary Register Content to Memory

### **Syntax Characteristics**

| No. | Syntax     | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|------------|------------------------|------|--------|----------|
| [1] | Smem = src | No                     | 2    | 1      | Х        |

Opcode

1100 FSSS AAAA AAAI

**Operands** 

Smem, src

Description

This instruction stores the content of the source (src) register to a memory (Smem) location.

- ☐ When the source register is an accumulator:
  - The low part of the accumulator, ACx(15–0), is stored to the memory location.
  - The store operation to the memory location uses a dedicated path independent of the D-unit ALU, the D-unit shifter, and the D-unit MACs.
- ☐ When the source register is an auxiliary or temporary register:
  - The content of the auxiliary or temporary register is stored to the memory location.
  - The store operation to the memory location uses a dedicated path independent of the A-unit ALU.

**Status Bits** 

Affected by none

Affects none

Repeat

This instruction can be repeated.

| Syntax          | Description                                          |
|-----------------|------------------------------------------------------|
| *(#0E10h) = AC0 | The content of AC0(15–0) is stored at location E10h. |

| Before |    |      |      | After |    |      |      |
|--------|----|------|------|-------|----|------|------|
| AC0    | 23 | 0400 | 6500 | AC0   | 23 | 0400 | 6500 |
| 0E10   |    |      | 0000 | 0E10  |    |      | 6500 |

### Store Accumulator, Auxiliary, or Temporary Register Content to Memory

### **Syntax Characteristics**

| No.     | Syntax        |               |                  |                            |                                                                    | Parallel<br>Enable Bit | Size     | Cycles      | Pipeline    |
|---------|---------------|---------------|------------------|----------------------------|--------------------------------------------------------------------|------------------------|----------|-------------|-------------|
| [2]     | high_byte(Sme | em <b>)</b> = | src              |                            |                                                                    | No                     | 3        | 1           | X           |
| Opcode  | е             |               |                  |                            | 1110                                                               | 0101 AAA               | A AA     | AI FSS      | S 01x0      |
| Operar  | nds           | Sm            | nem, src         |                            | ·                                                                  | ·                      |          | ·           |             |
| Descrip | otion         | hig           | h byte (bi       |                            | ne low byte (bits<br>the memory (S                                 | •                      |          |             |             |
|         |               |               | When th          | e source re                | gister is an ac                                                    | cumulator:             |          |             |             |
|         |               |               |                  | low part of the nemory loc | the accumulato<br>ation.                                           | or, ACx(7–0),          | is store | ed to the h | igh byte of |
|         |               |               |                  | •                          | ation to the mothe the D-unit ALU,                                 | •                      |          |             | •           |
|         |               |               | When th          | e source re                | gister is an au                                                    | xiliary or tem         | porary   | register:   |             |
|         |               |               |                  | . ,                        | ts 7–0) content<br>gh byte of the                                  |                        | •        | emporary    | register is |
|         |               |               |                  | •                          | ation to the methe the A-unit ALL                                  | •                      | on use   | s a dedid   | cated path  |
|         |               |               | (MMR).<br>an MMF | This instruc               | mem <b>cannot</b> retion cannot action cannot action sends a hard. | cess a byte v          | vithin a | n MMR. I    | f Smem is   |
| Status  | Bits          | Aff           | ected by         | none                       |                                                                    |                        |          |             |             |
|         |               | Aff           | ects             | none                       |                                                                    |                        |          |             |             |
| Repeat  |               | Thi           | s instruct       | ion can be                 | repeated.                                                          |                        |          |             |             |
| Examp   | le            |               |                  |                            |                                                                    |                        |          |             |             |

| Example |
|---------|
|---------|

| Syntax                | Description                                                                                      |
|-----------------------|--------------------------------------------------------------------------------------------------|
| high_byte(*AR1) = AC1 | The content of AC1(7–0) is stored in the high byte (bits 15–8) at the location addressed by AR1. |

| Before |              | After |              |
|--------|--------------|-------|--------------|
| AC1    | 20 FC00 6788 | AC1   | 20 FC00 6788 |
| AR1    | 0200         | AR1   | 0200         |
| 200    | 6903         | 200   | 8803         |

### Store Accumulator, Auxiliary, or Temporary Register Content to Memory

### **Syntax Characteristics**

| No. | Syntax               | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|----------------------|------------------------|------|--------|----------|
| [3] | low_byte(Smem) = src | No                     | 3    | 1      | Х        |
|     |                      |                        |      |        |          |

### Opcode

1110 0101 AAAA AAAI FSSS 01x1

#### **Operands**

Smem, src

### Description

This instruction stores the low byte (bits 7-0) of the source (src) register to the low byte (bits 7-0) of the memory (Smem) location. The high byte (bits 15-8) of Smem is unchanged.

- ☐ When the source register is an accumulator:
  - The low part of the accumulator, ACx(7–0), is stored to the low byte of the memory location.
  - The store operation to the memory location uses a dedicated path independent of the D-unit ALU, the D-unit shifter, and the D-unit MACs.
- ☐ When the source register is an auxiliary or temporary register:
  - The low part (bits 7–0) content of the auxiliary or temporary register is stored to the low byte of the memory location.
  - The store operation to the memory location uses a dedicated path independent of the A-unit ALU.
- ☐ In this instruction, Smem **cannot** reference to a memory-mapped register (MMR). This instruction cannot access a byte within an MMR. If Smem is an MMR, the DSP sends a hardware bus-error interrupt (BERRINT) request to the CPU.

Affects none

Affected by

Repeat

**Status Bits** 

This instruction can be repeated.

none

| Syntax               | Description                                                                                    |
|----------------------|------------------------------------------------------------------------------------------------|
| low_byte(*AR3) = AC0 | The content of AC0(7–0) is stored in the low byte (bits 7–0) at the location addressed by AR3. |

### MOV

### Store Auxiliary or Temporary Register Pair Content to Memory

### **Syntax Characteristics**

| No. Syntax      | Parallel<br>Enable Bit Size Cycles Pipeline                                                                                                                                                                                          |  |  |  |  |  |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|--|--|--|--|
| [1] Lmem = pai  | r(TAx) No 3 1 X                                                                                                                                                                                                                      |  |  |  |  |  |
| Opcode          | 1110   1011   AAAA   AAAI   FSSS   1100                                                                                                                                                                                              |  |  |  |  |  |
| Operands        | TAx, Lmem                                                                                                                                                                                                                            |  |  |  |  |  |
| Description     | This instruction stores the content of the temporary or auxiliary register (TAx) to the 16 highest bits of the data memory operand (Lmem) and stores the content of $TA(x + 1)$ to the 16 lowest bits of data memory operand (Lmem): |  |  |  |  |  |
|                 | ☐ The store operation to the memory location uses a dedicated path independent of the A-unit ALU.                                                                                                                                    |  |  |  |  |  |
|                 | ☐ Valid auxiliary registers are AR0, AR2, AR4, and AR6.                                                                                                                                                                              |  |  |  |  |  |
|                 | ☐ Valid temporary registers are T0 and T2.                                                                                                                                                                                           |  |  |  |  |  |
| Status Bits     | Affected by none                                                                                                                                                                                                                     |  |  |  |  |  |
|                 | Affects none                                                                                                                                                                                                                         |  |  |  |  |  |
| Repeat          | This instruction can be repeated.                                                                                                                                                                                                    |  |  |  |  |  |
| See Also        | See the following other related instructions:                                                                                                                                                                                        |  |  |  |  |  |
|                 | ☐ Load Accumulator, Auxiliary, or Temporary Register from Memory                                                                                                                                                                     |  |  |  |  |  |
|                 | ☐ Store Accumulator, Auxiliary, or Temporary Register Content to Memory                                                                                                                                                              |  |  |  |  |  |
| Example         |                                                                                                                                                                                                                                      |  |  |  |  |  |
| Syntax          | Description                                                                                                                                                                                                                          |  |  |  |  |  |
| *AR2 = pair(T0) | The content of T0 is stored at the location addressed by AR2 and the content of T1 is stored at the location addressed by AR2 + 1.                                                                                                   |  |  |  |  |  |

### MOV

### Store CPU Register Content to Memory

### **Syntax Characteristics**

| No.  | Syntax             | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|------|--------------------|------------------------|------|--------|----------|
| [1]  | Smem = <b>BK03</b> | No                     | 3    | 1      | Х        |
| [2]  | Smem = <b>BK47</b> | No                     | 3    | 1      | X        |
| [3]  | Smem = <b>BKC</b>  | No                     | 3    | 1      | X        |
| [4]  | Smem = BSA01       | No                     | 3    | 1      | X        |
| [5]  | Smem = BSA23       | No                     | 3    | 1      | X        |
| [6]  | Smem = BSA45       | No                     | 3    | 1      | X        |
| [7]  | Smem = BSA67       | No                     | 3    | 1      | X        |
| [8]  | Smem = <b>BSAC</b> | No                     | 3    | 1      | X        |
| [9]  | Smem = BRC0        | No                     | 3    | 1      | X        |
| [10] | Smem = BRC1        | No                     | 3    | 1      | X        |
| [11] | Smem = CDP         | No                     | 3    | 1      | X        |
| [12] | Smem = CSR         | No                     | 3    | 1      | X        |
| [13] | Smem = <b>DP</b>   | No                     | 3    | 1      | X        |
| [14] | Smem = <b>DPH</b>  | No                     | 3    | 1      | X        |
| [15] | Smem = PDP         | No                     | 3    | 1      | X        |
| [16] | Smem = <b>SP</b>   | No                     | 3    | 1      | X        |
| [17] | Smem = <b>SSP</b>  | No                     | 3    | 1      | X        |
| [18] | Smem = TRN0        | No                     | 3    | 1      | X        |
| [19] | Smem = TRN1        | No                     | 3    | 1      | X        |
| [20] | dbl(Lmem) = RETA   | No                     | 3    | 5      | X        |

Opcode See Table 5-6 (page 5-599).

Operands Lmem, Smem

#### Description

These instructions store the content of the selected source CPU register to a memory (Smem) location or a data memory operand (Lmem).

For instructions [9] and [10], the block repeat register (BRCx) is decremented in the address phase of the last instruction of the loop. These instructions have a 3-cycle latency requirement versus the last instruction of the loop.

For instruction [20], the content of the 24-bit RETA register (the return address of the calling subroutine) and the 8-bit CFCT register (active control flow execution context flags of the calling subroutine) are stored to the data memory operand (Lmem):

- The content of the CFCT register and the 8 highest bits of the RETA register are stored in the 16 highest bits of Lmem.
- ☐ The 16 lowest bits of the RETA register are stored in the 16 lowest bits of Lmem.

When instruction [20] is decoded, the CPU pipeline is flushed and the instruction is executed in 5 cycles, regardless of the instruction context.

#### **Status Bits**

Affected by none

Affects none

#### Repeat

Instruction [20] cannot be repeated; all other instructions can be repeated.

### See Also

See the following other related instructions:

- Load CPU Register from Memory
- Load CPU Register with Immediate Value
- Move CPU Register Content to Auxiliary or Temporary Register
- ☐ Store Accumulator Content to Memory
- ☐ Store Accumulator Pair Content to Memory
- ☐ Store Accumulator, Auxiliary, or Temporary Register Content to Memory
- ☐ Store Auxiliary or Temporary Register Pair Content to Memory

| Syntax | Description                                                                                                     |
|--------|-----------------------------------------------------------------------------------------------------------------|
|        | The content of the data stack pointer (SP) is stored in the location addressed by AR1. AR1 is incremented by 1. |

| Before |      | After |      |
|--------|------|-------|------|
| AR1    | 0200 | AR1   | 0201 |
| SP     | 0200 | SP    | 0200 |
| 200    | 0000 | 200   | 0200 |

### Example 2

| Syntax | Description                                                                                                        |
|--------|--------------------------------------------------------------------------------------------------------------------|
|        | The content of the system stack pointer (SSP) is stored in the location addressed by AR1. AR1 is incremented by 1. |

| Before |      | After |      |
|--------|------|-------|------|
| AR1    | 0201 | AR1   | 0202 |
| SSP    | 0000 | SSP   | 0000 |
| 201    | 00FF | 201   | 0000 |

### Example 3

| Syntax       | Description                                                                               |
|--------------|-------------------------------------------------------------------------------------------|
| *AR1+ = TRN0 | The content of the transition register (TRN0) is stored in the location addressed by AR1. |
|              | AR1 is incremented by 1.                                                                  |

| Before |      | After |      |
|--------|------|-------|------|
| AR1    | 0202 | AR1   | 0203 |
| TRN0   | 3490 | TRN0  | 3490 |
| 202    | 0000 | 202   | 3490 |

### Example 4

| Syntax       | Description                                                                                                        |
|--------------|--------------------------------------------------------------------------------------------------------------------|
| *AR1+ = TRN1 | The content of the transition register (TRN1) is stored in the location addressed by AR1. AR1 is incremented by 1. |

| Before |      | After |      |
|--------|------|-------|------|
| AR1    | 0203 | AR1   | 0204 |
| TRN1   | 0020 | TRN1  | 0020 |
| 203    | 0000 | 203   | 0020 |

| Syntax           | Description                                                                                |
|------------------|--------------------------------------------------------------------------------------------|
| dbl(*AR3) = RETA | The contents of the RETA and CFCT are stored in the location addressed by AR3 and AR3 + 1. |

Table 5-6. Opcodes for Store CPU Register Content to Memory Instruction

| No.  | Syntax              |      |      | Орс  | ode  |      |      |
|------|---------------------|------|------|------|------|------|------|
| [1]  | Smem = <b>BK03</b>  | 1110 | 0101 | AAAA | AAAI | 1001 | 10xx |
| [2]  | Smem = <b>BK47</b>  | 1110 | 0101 | AAAA | AAAI | 1010 | 10xx |
| [3]  | Smem = <b>BKC</b>   | 1110 | 0101 | AAAA | AAAI | 1011 | 10xx |
| [4]  | Smem = <b>BSA01</b> | 1110 | 0101 | AAAA | AAAI | 0010 | 10xx |
| [5]  | Smem = <b>BSA23</b> | 1110 | 0101 | AAAA | AAAI | 0011 | 10xx |
| [6]  | Smem = <b>BSA45</b> | 1110 | 0101 | AAAA | AAAI | 0100 | 10xx |
| [7]  | Smem = <b>BSA67</b> | 1110 | 0101 | AAAA | AAAI | 0101 | 10xx |
| [8]  | Smem = <b>BSAC</b>  | 1110 | 0101 | AAAA | AAAI | 0110 | 10xx |
| [9]  | Smem = BRC0         | 1110 | 0101 | AAAA | AAAI | x001 | 11xx |
| [10] | Smem = BRC1         | 1110 | 0101 | AAAA | AAAI | x010 | 11xx |
| [11] | Smem = CDP          | 1110 | 0101 | AAAA | AAAI | 0001 | 10xx |
| [12] | Smem = CSR          | 1110 | 0101 | AAAA | AAAI | x000 | 11xx |
| [13] | Smem = <b>DP</b>    | 1110 | 0101 | AAAA | AAAI | 0000 | 10xx |
| [14] | Smem = <b>DPH</b>   | 1110 | 0101 | AAAA | AAAI | 1100 | 10xx |
| [15] | Smem = <b>PDP</b>   | 1110 | 0101 | AAAA | AAAI | 1111 | 10xx |
| [16] | Smem = <b>SP</b>    | 1110 | 0101 | AAAA | AAAI | 0111 | 10xx |
| [17] | Smem = SSP          | 1110 | 0101 | AAAA | AAAI | 1000 | 10xx |
| [18] | Smem = TRN0         | 1110 | 0101 | AAAA | AAAI | x011 | 11xx |
| [19] | Smem = TRN1         | 1110 | 0101 | AAAA | AAAI | x100 | 11xx |
| [20] | dbl(Lmem) = RETA    | 1110 | 1011 | AAAA | AAAI | xxxx | 01xx |

### MOV

### Store Extended Auxiliary Register Content to Memory

### **Syntax Characteristics**

| No. | Syntax            | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|-------------------|------------------------|------|--------|----------|
| [1] | dbl(Lmem) = XAsrc | No                     | 3    | 1      | Х        |
|     |                   |                        |      |        |          |

**Opcode** | 1110 1101 | AAAA AAAI | XSSS 0101

Operands Lmem, XAsrc

**Description** This instruction moves the content of the 23-bit source register (XARx, XSP,

XSSP, XDP, or XCDP) to the 32-bit data memory location addressed by data memory operand (Lmem). The upper 9 bits of the data memory are filled with 0:

Status Bits Affected by none

Affects none

**Repeat** This instruction can be repeated.

**See Also** See the following other related instructions:

■ Load Extended Auxiliary Register from Memory

☐ Load Extended Auxiliary Register with Immediate Value

| Syntax           | Description                                                                                                                                                                                                      |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| dbl(*AR3) = XAR1 | The 7 highest bits of XAR1 are moved to the 7 lowest bits of the location addressed by AR3, the 9 highest bits are filled with 0, and the 16 lowest bits of XAR1 are moved to the location addressed by AR3 + 1. |

| Before |         | After |         |
|--------|---------|-------|---------|
| XAR1   | 7F 3492 | XAR1  | 7F 3492 |
| AR3    | 0200    | AR3   | 0200    |
| 200    | 3765    | 200   | 007F    |
| 201    | 0FD3    | 201   | 3492    |

### **SUBC**

### Subtract Conditionally

### **Syntax Characteristics**

| No. | Syntax               | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|----------------------|------------------------|------|--------|----------|
| [1] | subc(Smem, ACx, ACy) | No                     | 3    | 1      | X        |

### Opcode

1101 1110 AAAA AAAI SSDD 0011

# Operands Description

ACx, ACy, Smem

This instruction performs a conditional subtraction in the D-unit ALU. The D-unit shifter is not used to perform the memory operand shift.

- ☐ The 16-bit data memory operand Smem is sign extended to 40 bits according to SXMD, shifted left by 15 bits, and subtracted from the content of the source accumulator ACx.
  - The shift operation is equivalent to the signed shift instruction.
  - Overflow and carry bit is always detected at bit position 31. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.
  - If an overflow is detected and reported in accumulator overflow bit ACOVy, no saturation is performed on the result of the operation.
- ☐ If the result of the subtraction is greater than 0 (bit 39 = 0), the result is shifted left by 1 bit, added to 1, and stored in the destination accumulator ACy.
- ☐ If the result of the subtraction is less than 0 (bit 39 = 1), the source accumulator ACx is shifted left by 1 bit and stored in the destination accumulator ACy.

```
if ((ACx - (Smem << #15)) >= 0)
    ACy = (ACx - (Smem << #15)) << #1 + 1
else
    ACy = ACx << #1</pre>
```

This instruction is used to make a 16 step 16-bit by 16-bit division. The divisor and the dividend are both assumed to be positive in this instruction. SXMD affects this operation:

- ☐ If SXMD = 1, the divisor must have a 0 value in the most significant bit
- ☐ If SXMD = 0, any 16-bit divisor value produces the expected result

The dividend, which is in the source accumulator ACx, must be positive (bit 31 = 0) during the computation.

| Status Bits | Affected by       | SXMD                                                  |
|-------------|-------------------|-------------------------------------------------------|
|             | Affects           | ACOVy, CARRY                                          |
| Repeat      | This instruction  | can be repeated.                                      |
| See Also    | See the following | ng other related instructions:                        |
|             | ☐ Addition or     | Subtraction Conditionally                             |
|             | ☐ Addition or     | Subtraction Conditionally with Shift                  |
|             | ☐ Addition, S     | ubtraction, or Move Accumulator Content Conditionally |
|             | Dual 16-Bit       | Subtraction and Addition                              |
|             | Subtraction       | n                                                     |
|             | Subtraction       | with Parallel Store Accumulator Content to Memory     |

### Example 1

| Syntax | Description                                                                                                                                                                                                                                                           |
|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|        | The content addressed by AR1 shifted left by 15 bits is subtracted from the content of AC0. The result is greater than 0; therefore, the result is shifted left by 1 bit, added to 1, and the new result stored in AC1. The result generated an overflow and a carry. |

| Before |         |      | After |    |      |      |
|--------|---------|------|-------|----|------|------|
| AC0    | 23 4300 | 0000 | AC0   | 23 | 4300 | 0000 |
| AC1    | 00 0000 | 0000 | AC1   | 46 | 8400 | 0001 |
| AR1    |         | 300  | AR1   |    |      | 300  |
| 300    |         | 200  | 300   |    |      | 200  |
| SXMD   |         | 0    | SXMD  |    |      | 0    |
| ACOV1  |         | 0    | ACOV1 |    |      | 1    |
| CARRY  |         | 0    | CARRY |    |      | 1    |

| Syntax               | Description                                                                                                                                                                                                                                                                                                                                                                                       |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| repeat (CSR)         | The content addressed by AR1 shifted left by 15 bits is subtracted from the                                                                                                                                                                                                                                                                                                                       |
| subc(*AR1, AC1, AC1) | content of AC1. The result is greater than 0; therefore, the result is shifted left by 1 bit, added to 1, and the new result stored in AC1. The content addressed by AR1 shifted left by 15 bits is subtracted from the content of AC1. The result is greater than 0; therefore, the result is shifted left by 1 bit, added to 1, and the new result stored in AC1. The result generated a carry. |

| Before |       |         | After |         |      |
|--------|-------|---------|-------|---------|------|
| AC1    | 00 07 | 46 0000 | AC1   | 00 1A18 | 0007 |
| AR1    |       | 200     | AR1   |         | 200  |
| 200    |       | 0100    | 200   |         | 0100 |
| CSR    |       | 1       | CSR   |         | 0    |
| ACOV1  |       | 0       | ACOV1 |         | 0    |
| CARRY  |       | 0       | CARRY |         | 1    |

**SUB** 

### Subtraction

### **Syntax Characteristics**

| No.  | Syntax                              | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|------|-------------------------------------|------------------------|------|--------|----------|
| [1]  | dst = dst - src                     | Yes                    | 2    | 1      | Х        |
| [2]  | dst = dst - k4                      | Yes                    | 2    | 1      | X        |
| [3]  | dst = src - K16                     | No                     | 4    | 1      | X        |
| [4]  | dst = src - Smem                    | No                     | 3    | 1      | X        |
| [5]  | dst = Smem - src                    | No                     | 3    | 1      | X        |
| [6]  | ACy = ACy - (ACx << Tx)             | Yes                    | 2    | 1      | X        |
| [7]  | ACy = ACy - (ACx << #SHIFTW)        | Yes                    | 3    | 1      | X        |
| [8]  | ACy = ACx - (K16 << #16)            | No                     | 4    | 1      | X        |
| [9]  | ACy = ACx - (K16 << #SHFT)          | No                     | 4    | 1      | X        |
| [10] | ACy = ACx - (Smem << Tx)            | No                     | 3    | 1      | X        |
| [11] | ACy = ACx - (Smem << #16)           | No                     | 3    | 1      | X        |
| [12] | ACy = (Smem << #16) - ACx           | No                     | 3    | 1      | X        |
| [13] | ACy = ACx - uns(Smem) - BORROW      | No                     | 3    | 1      | X        |
| [14] | ACy = ACx - uns(Smem)               | No                     | 3    | 1      | X        |
| [15] | ACy = ACx - (uns(Smem) << #SHIFTW)  | No                     | 4    | 1      | X        |
| [16] | ACy = ACx - dbl(Lmem)               | No                     | 3    | 1      | X        |
| [17] | ACy = dbl(Lmem) - ACx               | No                     | 3    | 1      | Х        |
| [18] | ACx = (Xmem << #16) - (Ymem << #16) | No                     | 3    | 1      | X        |

These instructions perform a subtraction operation. **Description** 

Affected by CARRY, C54CM, M40, SATA, SATD, SXMD **Status Bits** 

> ACOVx, ACOVy, CARRY Affects

| See Also | See the following other related instructions:                    |
|----------|------------------------------------------------------------------|
|          | ☐ Addition                                                       |
|          | Addition or Subtraction Conditionally                            |
|          | Addition or Subtraction Conditionally with Shift                 |
|          | Addition, Subtraction, or Move Accumulator Content Conditionally |
|          | Dual 16-Bit Addition and Subtraction                             |
|          | Dual 16-Bit Subtractions                                         |
|          | Dual 16-Bit Subtraction and Addition                             |
|          | ☐ Subtract Conditionally                                         |
|          | Subtraction with Parallel Store Accumulator Content to Memory    |

### **Syntax Characteristics**

| No.    | Syntax          |      |            |       |                                                    |            | Parallel<br>Enable Bit | Size      | Cycles       | Pipeline     |
|--------|-----------------|------|------------|-------|----------------------------------------------------|------------|------------------------|-----------|--------------|--------------|
| [1]    | dst = dst - src |      |            |       |                                                    |            | Yes                    | 2         | 1            | Х            |
| Opcod  | e               |      |            |       |                                                    |            | 00                     | 10 0:     | 11E FSS      | SS FDDD      |
| Opera  | nds             | dst, | src        |       |                                                    |            | 1                      |           | ı            |              |
| Descri | ption           | This | s instruc  | ction | performs a s                                       | ubtractio  | n operation            | betwee    | en two reg   | gisters.     |
|        |                 |      | When th    | he d  | destination op                                     | erand (d   | st) is an acc          | umulat    | or:          |              |
|        |                 |      | ■ The      | е ор  | eration is per                                     | formed o   | n 40 bits in           | the D-u   | ınit ALU.    |              |
|        |                 |      | ■ Inpu     | ut o  | perands are                                        | sign exte  | nded to 40 b           | its acc   | ording to    | SXMD.        |
|        |                 |      | inst       | truc  | uxiliary or tem<br>tion, the 16 LS<br>ed according | SBs of the | e auxiliary or         |           | -            |              |
|        |                 |      | sub        | otrac | ow detection<br>ction borrow b<br>ne logical com   | it is repo | rted in the C          | ARRY      | status bit;  |              |
|        |                 |      | ■ Who      |       | an overflow is<br>D.                               | detected   | , the accumu           | ulator is | saturated    | d according  |
|        |                 |      |            |       | destination operation is per                       | •          | •                      | -         | •            | ry register: |
|        |                 |      |            |       | ccumulator is<br>Bs of the accu                    |            | •                      | ` ,       |              | •            |
|        |                 |      | ■ Ove      | erflo | w detection i                                      | s done at  | bit position           | 15.       |              |              |
|        |                 |      |            |       | an overflow ing to SATA.                           | s detecte  | d, the desti           | nation    | register is  | saturated    |
|        |                 | Col  | mpatibil   | lity  | with C54x de                                       | evices (C  | 54CM = 1)              |           |              |              |
|        |                 | Wh   | en this ir | nstr  | uction is exec                                     | cuted with | n M40 = 0, c           | ompati    | bility is er | nsured.      |
| Status | Bits            | Affe | ected by   | ,     | M40, SATA,                                         | SATD, S    | SXMD                   |           |              |              |
|        |                 |      |            |       |                                                    |            |                        |           |              |              |

### Repeat Example

| Syntax          | Description                                                                               |
|-----------------|-------------------------------------------------------------------------------------------|
| AC0 = AC0 - AC1 | The content of AC1 is subtracted from the content of AC0 and the result is stored in AC0. |

ACOVx, CARRY

This instruction can be repeated.

Affects

#### **Syntax Characteristics**

| No. | Syntax         | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|----------------|------------------------|------|--------|----------|
| [2] | dst = dst - k4 | Yes                    | 2    | 1      | Х        |

#### Opcode

0100 011E kkkk FDDD

**Operands** 

dst, k4

Description

This instruction subtracts a 4-bit unsigned constant, k4, from a register.

- ☐ When the destination operand (dst) is an accumulator:
  - The operation is performed on 40 bits in the D-unit ALU.
  - Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.
  - When an overflow is detected, the accumulator is saturated according to SATD.
- ☐ When the destination operand (dst) is an auxiliary or temporary register:
  - The operation is performed on 16 bits in the A-unit ALU.
  - Overflow detection is done at bit position 15.
  - When an overflow is detected, the destination register is saturated according to SATA.

### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

**Status Bits** Affected by M40, SATA, SATD

Affects

ACOVx, CARRY

Repeat

This instruction can be repeated.

| Syntax          | Description                                                                                         |
|-----------------|-----------------------------------------------------------------------------------------------------|
| AC0 = AC0 - #15 | An unsigned 4-bit value (15) is subtracted from the content of AC0 and the result is stored in AC0. |

### **Syntax Characteristics**

| No. | Syntax          | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|-----------------|------------------------|------|--------|----------|
| [3] | dst = src - K16 | No                     | 4    | 1      | Х        |

#### Opcode

0111 1100 KKKK KKKK KKKK KKKK FDDD FSSS

#### **Operands**

dst, K16, src

### Description

This instruction subtracts a 16-bit signed constant, K16, from a register.

- ☐ When the destination operand (dst) is an accumulator:
  - The operation is performed on 40 bits in the D-unit ALU.
  - If an auxiliary or temporary register is the source operand (src) of the instruction, the 16 LSBs of the auxiliary or temporary register are sign extended according to SXMD.
  - The 16-bit constant, K16, is sign extended to 40 bits according to SXMD.
  - Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.
  - When an overflow is detected, the accumulator is saturated according to SATD.
- ☐ When the destination operand (dst) is an auxiliary or temporary register:
  - The operation is performed on 16 bits in the A-unit ALU.
  - If an accumulator is the source operand (src) of the instruction, the 16 LSBs of the accumulator are used to perform the operation.
  - Overflow detection is done at bit position 15.
  - When an overflow is detected, the destination register is saturated according to SATA.

### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

**SUB** Subtraction

Status Bits Affected by M40, SATA, SATD, SXMD

Affects ACOVx, CARRY

**Repeat** This instruction can be repeated.

| Syntax            | Description                                                                                          |
|-------------------|------------------------------------------------------------------------------------------------------|
| AC0 = AC1 – FFFFh | A signed 16-bit value (FFFFh) is subtracted from the content of AC1 and the result is stored in AC0. |

### **Syntax Characteristics**

| No. | Syntax           | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|------------------|------------------------|------|--------|----------|
| [4] | dst = src - Smem | No                     | 3    | 1      | Х        |

### **Opcode**

| 1101 0111 | AAAA AAAI | FDDD FSSS

#### **Operands**

dst, Smem, src

#### Description

This instruction subtracts the content of a memory (Smem) location from a register content.

- ☐ When the destination operand (dst) is an accumulator:
  - The operation is performed on 40 bits in the D-unit ALU.
  - If an auxiliary or temporary register is the source operand (src) of the instruction, the 16 LSBs of the auxiliary or temporary register are sign extended according to SXMD.
  - The content of the memory location is sign extended to 40 bits according to SXMD.
  - Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.
  - When an overflow is detected, the accumulator is saturated according to SATD.
- ☐ When the destination operand (dst) is an auxiliary or temporary register:
  - The operation is performed on 16 bits in the A-unit ALU.
  - If an accumulator is the source operand (src) of the instruction, the 16 LSBs of the accumulator are used to perform the operation.
  - Overflow detection is done at bit position 15.
  - When an overflow is detected, the destination register is saturated according to SATA.

### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

### **SUB** Subtraction

Affected by M40, SATA, SATD, SXMD **Status Bits** 

> Affects ACOVx, CARRY

Repeat This instruction can be repeated.

| Syntax           | Description                                                                                         |
|------------------|-----------------------------------------------------------------------------------------------------|
| AC0 = AC1 - *AR3 | The content addressed by AR3 is subtracted from the content of AC1 and the result is stored in AC0. |

### **Syntax Characteristics**

| No. | Syntax           | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|------------------|------------------------|------|--------|----------|
| [5] | dst = Smem - src | No                     | 3    | 1      | Х        |

### **Opcode**

| 1101 1000 | AAAA AAAI | FDDD FSSS

#### **Operands**

dst, Smem, src

#### Description

This instruction subtracts a register content from the content of a memory (Smem) location.

- ☐ When the destination operand (dst) is an accumulator:
  - The operation is performed on 40 bits in the D-unit ALU.
  - If an auxiliary or temporary register is the source operand (src) of the instruction, the 16 LSBs of the auxiliary or temporary register are sign extended according to SXMD.
  - The content of the memory location is sign extended to 40 bits according to SXMD.
  - Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.
  - When an overflow is detected, the accumulator is saturated according to SATD.
- ☐ When the destination operand (dst) is an auxiliary or temporary register:
  - The operation is performed on 16 bits in the A-unit ALU.
    - If an accumulator is the source operand (src) of the instruction, the 16 LSBs of the accumulator are used to perform the operation.
    - Overflow detection is done at bit position 15.
  - When an overflow is detected, the destination register is saturated according to SATA.

### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

### **SUB** Subtraction

Affected by M40, SATA, SATD, SXMD **Status Bits** 

> Affects ACOVx, CARRY

This instruction can be repeated. Repeat

| Syntax           | Description                                                                                         |
|------------------|-----------------------------------------------------------------------------------------------------|
| AC0 = *AR3 – AC1 | The content of AC1 is subtracted from the content addressed by AR3 and the result is stored in AC0. |

### **Syntax Characteristics**

| No.      | Syntax         |      |                           |                                                                        | Parallel<br>Enable Bit | Size     | Cycles       | Pipeline    |
|----------|----------------|------|---------------------------|------------------------------------------------------------------------|------------------------|----------|--------------|-------------|
| [6]      | ACy = ACy - (A | Cx < | < Tx)                     |                                                                        | Yes                    | 2        | 1            | Х           |
| Opcode   |                |      |                           |                                                                        | 01                     | 01 1     | 01E DDS      | SS ss01     |
| Operands |                |      | x, ACy, Tx                |                                                                        |                        |          |              |             |
| Descri   | ption          |      |                           | n subtracts an accum<br>accumulator content                            |                        | ACx s    | hifted by t  | he content  |
|          |                |      | The opera                 | ition is performed on                                                  | 40 bits in the [       | D-unit s | shifter.     |             |
|          |                |      | Input oper                | Input operands are sign extended to 40 bits according to SXMD.         |                        |          |              |             |
|          |                |      | The shift of              | The shift operation is equivalent to the signed shift instruction.     |                        |          |              |             |
|          |                |      | subtractio                | detection and CAF<br>n borrow bit is reported<br>cal complement of the | ed in the CARF         | RY stat  |              |             |
|          |                |      | When an o                 | overflow is detected,                                                  | the accumulate         | or is sa | turated a    | ccording to |
|          |                | Co   | mpatibility               | with C54x devices                                                      | (C54CM=1)              |          |              |             |
|          |                |      | nen this inst<br>4CM = 1: | ruction is executed wi                                                 | th M40 = 0, cor        | npatibi  | lity is ensu | ıred. When  |
|          |                |      |                           | ediary shift operation is<br>w detection, report,                      | •                      |          | -            |             |
|          |                |      | Tx define a               | Bs of Tx are used to cashift quantity within modulo 16 operation       | -32 to +31. Who        | en the   | value is be  | etween –32  |
| Status   | Bits           | Aff  | ected by                  | C54CM, M40, SAT                                                        | D, SXMD                |          |              |             |
|          |                | Aff  | ects                      | ACOVy, CARRY                                                           |                        |          |              |             |
| Repeat   | :              | Thi  | is instructio             | n can be repeated.                                                     |                        |          |              |             |
| Evama    | lo             |      |                           |                                                                        |                        |          |              |             |

| Syntax                  | Description                                                                                                            |
|-------------------------|------------------------------------------------------------------------------------------------------------------------|
| AC0 = AC0 - (AC1 << T0) | The content of AC1 shifted by the content of T0 is subtracted from the content of AC0 and the result is stored in AC0. |

### **Syntax Characteristics**

| No. | Syntax                       | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|------------------------------|------------------------|------|--------|----------|
| [7] | ACy = ACy - (ACx << #SHIFTW) | Yes                    | 3    | 1      | X        |

#### **Opcode**

0001 000E DDSS 0100 xxSH IFTW

**Operands** 

ACx, ACy, SHIFTW

Description

This instruction subtracts an accumulator content ACx shifted by the 6-bit value, SHIFTW, from an accumulator content ACy.

- ☐ The operation is performed on 40 bits in the D-unit shifter.
- ☐ Input operands are sign extended to 40 bits according to SXMD.
- ☐ The shift operation is equivalent to the signed shift instruction.
- Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.
- ☐ When an overflow is detected, the accumulator is saturated according to SATD.

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, an intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.

**Status Bits** 

Affected by C54CM, M40, SATD, SXMD

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

| Syntax                   | Description                                                                                                       |
|--------------------------|-------------------------------------------------------------------------------------------------------------------|
| AC0 = AC0 - (AC1 << #31) | The content of AC1 shifted left by 31 bits is subtracted from the content of AC0 and the result is stored in AC0. |

### **Syntax Characteristics**

| No. | Syntax                                        | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|-----------------------------------------------|------------------------|------|--------|----------|
| [8] | ACy = ACx - <b>(</b> K16 <b>&lt;&lt; #16)</b> | No                     | 4    | 1      | Х        |

### **Opcode**

0111 1010 KKKK KKKK KKKK KKKK SSDD 001x

#### **Operands**

ACx, ACy, K16

#### Description

This instruction subtracts the 16-bit signed constant, K16, shifted left by 16 bits from an accumulator content ACx.

- The operation is performed on 40 bits in the D-unit ALU.
- ☐ Input operands are sign extended to 40 bits according to SXMD.
- ☐ The shift operation is equivalent to the signed shift instruction.
- Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.
- ☐ When an overflow is detected, the accumulator is saturated according to SATD.

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, an intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.

#### **Status Bits**

Affected by

C54CM, M40, SATD, SXMD

Affects

ACOVy, CARRY

### Repeat

This instruction can be repeated.

| Syntax                     | Description                                                                                                                  |
|----------------------------|------------------------------------------------------------------------------------------------------------------------------|
| AC0 = AC1 – (FFFFh << #16) | A signed 16-bit value (FFFFh) shifted left by 16 bits is subtracted from the content of AC1 and the result is stored in AC0. |

#### **Syntax Characteristics**

| No. | Syntax                     | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|----------------------------|------------------------|------|--------|----------|
| [9] | ACy = ACx - (K16 << #SHFT) | No                     | 4    | 1      | Х        |

**Opcode** 

0111 0001 KKKK KKKK KKKK KKKK SSDD SHFT

**Operands** 

ACx, ACy, K16, SHFT

Description

This instruction subtracts the 16-bit signed constant, K16, shifted left by the 4-bit value, SHFT, from an accumulator content ACx.

- ☐ The operation is performed on 40 bits in the D-unit shifter.
- ☐ Input operands are sign extended to 40 bits according to SXMD.
- ☐ The shift operation is equivalent to the signed shift instruction.
- Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.
- ☐ When an overflow is detected, the accumulator is saturated according to SATD.

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, an intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.

**Status Bits** 

Affected by M40, SATD, SXMD

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

| Syntax                     | Description                                                                                                                 |
|----------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| AC1 = AC0 - (#9800h << #5) | A signed 16-bit value (9800h) shifted left by 5 bits is subtracted from the content of AC0 and the result is stored in AC1. |

### **Syntax Characteristics**

| No.                                   | Syntax          |                                                                                                                                                                             |                                                                                     | Parallel<br>Enable Bit | Size   | Cycles      | Pipeline   |  |  |  |
|---------------------------------------|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|------------------------|--------|-------------|------------|--|--|--|
| [10]                                  | ACy = ACx - (3) | Smem << Tx)                                                                                                                                                                 |                                                                                     | No                     | 3      | 1           | Х          |  |  |  |
| Opcod                                 | e               |                                                                                                                                                                             | 1101                                                                                | 1101 AA                | AA A   | AAI SSI     | DD ss01    |  |  |  |
| Operar                                | nds             | ACx, ACy, Sr                                                                                                                                                                | nem, Tx                                                                             |                        |        |             |            |  |  |  |
| Descri                                | ption           |                                                                                                                                                                             | on subtracts the content of Tx from an accumulato                                   | •                      | •      | n) locatior | shifted by |  |  |  |
|                                       |                 | The oper                                                                                                                                                                    | ☐ The operation is performed on 40 bits in the D-unit shifter.                      |                        |        |             |            |  |  |  |
|                                       |                 | ☐ Input ope                                                                                                                                                                 | ☐ Input operands are sign extended to 40 bits according to SXMD.                    |                        |        |             |            |  |  |  |
|                                       |                 | ☐ The shift operation is equivalent to the signed shift instruction.                                                                                                        |                                                                                     |                        |        |             |            |  |  |  |
|                                       |                 | Overflow detection and CARRY status bit depends on M subtraction borrow bit is reported in the CARRY status bit; the bit is the logical complement of the CARRY status bit. |                                                                                     |                        |        |             |            |  |  |  |
|                                       |                 | ☐ When an SATD.                                                                                                                                                             |                                                                                     |                        |        |             |            |  |  |  |
|                                       |                 | Compatibility with C54x devices (C54CM = 1)                                                                                                                                 |                                                                                     |                        |        |             |            |  |  |  |
|                                       |                 | When this instruction is executed with $M40 = 0$ , compatibility is ensured. When $C54CM = 1$ :                                                                             |                                                                                     |                        |        |             |            |  |  |  |
|                                       |                 | An intermediary shift operation is performed as if M40 is locally set to 1 a no overflow detection, report, and saturation is done after the shifting operation.            |                                                                                     |                        |        |             |            |  |  |  |
|                                       |                 | Tx define                                                                                                                                                                   | Bs of Tx are used to deto<br>a shift quantity within –32<br>modulo 16 operation tra | 2 to +31. Who          | en the | value is be | etween –32 |  |  |  |
| Status                                | Bits            | Affected by                                                                                                                                                                 | C54CM, M40, SATD,                                                                   | SXMD                   |        |             |            |  |  |  |
|                                       |                 | Affects                                                                                                                                                                     | ACOVy, CARRY                                                                        |                        |        |             |            |  |  |  |
| Repeat This instruction can be repeat |                 |                                                                                                                                                                             |                                                                                     |                        |        |             |            |  |  |  |

| Syntax                   | Description                                                                                                                      |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------|
| AC0 = AC1 - (*AR3 << T0) | The content addressed by AR3 shifted by the content of T0 is subtracted from the content of AC1 and the result is stored in AC0. |

### **Syntax Characteristics**

| No.  | Syntax                    | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|------|---------------------------|------------------------|------|--------|----------|
| [11] | ACy = ACx - (Smem << #16) | No                     | 3    | 1      | Χ        |

#### **Opcode**

1101 1110 AAAA AAAI SSDD 0101

**Operands** 

ACx, ACy, Smem

Description

This instruction subtracts the content of a memory (Smem) location shifted left by 16 bits from an accumulator content ACx.

- ☐ The operation is performed on 40 bits in the D-unit ALU.
- ☐ Input operands are sign extended to 40 bits according to SXMD.
- ☐ The shift operation is equivalent to the signed shift instruction.
- Overflow detection and CARRY status bit depends on M40. If the result of the subtraction generates a borrow, the CARRY status bit is cleared; otherwise, the CARRY status bit is not affected.
- ☐ When an overflow is detected, the accumulator is saturated according to SATD.

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, an intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.

**Status Bits** 

Affected by C54C

C54CM, M40, SATD, SXMD

Affects

ACOVy, CARRY

Repeat

This instruction can be repeated.

| Syntax                    | Description                                                                                                                 |
|---------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| AC0 = AC1 - (*AR3 << #16) | The content addressed by AR3 shifted left by 16 bits is subtracted from the content of AC1 and the result is stored in AC0. |

### **Syntax Characteristics**

| No.  | Syntax                    | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|------|---------------------------|------------------------|------|--------|----------|
| [12] | ACy = (Smem << #16) - ACx | No                     | 3    | 1      | Χ        |

### **Opcode**

| 1101 | 1110 | AAAA | AAAI | SSDD | 0110

### **Operands**

ACx, ACy, Smem

#### Description

This instruction subtracts an accumulator content ACx from the content of a memory (Smem) location shifted left by 16 bits.

- The operation is performed on 40 bits in the D-unit ALU.
- ☐ Input operands are sign extended to 40 bits according to SXMD.
- ☐ The shift operation is equivalent to the signed shift instruction.
- Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.
- ☐ When an overflow is detected, the accumulator is saturated according to SATD.

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, an intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.

#### **Status Bits**

Affected by

C54CM, M40, SATD, SXMD

Affects

ACOVy, CARRY

### Repeat

This instruction can be repeated.

| Syntax                    | Description                                                                                                                 |
|---------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| AC0 = (*AR3 << #16) - AC1 | The content of AC1 is subtracted from the content addressed by AR3 shifted left by 16 bits and the result is stored in AC0. |

### **Syntax Characteristics**

| No.         | Syntax              |        |               |                                             |               | Parallel<br>Enable Bit | Size     | Cycles       | Pipeline    |
|-------------|---------------------|--------|---------------|---------------------------------------------|---------------|------------------------|----------|--------------|-------------|
| [13]        | $ACy = ACx - \iota$ | ıns(Sn | nem) – BOF    | RROW                                        |               | No                     | 3        | 1            | X           |
| Opcod       | e                   |        |               |                                             | 1101          | 1111 AA                | AA A     | AAI   SSI    | DD 101u     |
| Operar      | nds                 | AC     | x, ACy, Sm    | nem                                         |               |                        |          |              |             |
| Description |                     | (bo    |               | on subtracts<br>the content o               | _             |                        |          |              |             |
|             |                     |        | The opera     | ation is perfo                              | ormed on 40   | ) bits in the [        | D-unit A | ALU.         |             |
|             |                     |        | Input ope     | rands are ex                                | tended to 4   | 0 bits accore          | ding to  | uns.         |             |
|             |                     |        |               | optional uns<br>memory loc                  |               |                        |          |              | the content |
|             |                     |        |               | optional unsint of the mer                  | •             |                        |          |              |             |
|             |                     |        | subtractio    | detection a<br>on borrow bit<br>cal complen | is reported   | in the CARF            | RY stat  |              |             |
|             |                     |        | When an SATD. | overflow is o                               | letected, the | e accumulato           | or is sa | turated a    | ccording to |
|             |                     | Со     | mpatibility   | y with C54x                                 | devices (C    | C54CM = 1)             |          |              |             |
|             |                     | Wh     | en this ins   | truction is ex                              | cecuted with  | n M40 = 0, co          | ompati   | bility is er | nsured.     |
| Status      | Bits                | Affe   | ected by      | CARRY, I                                    | М40, SATD,    | SXMD                   |          |              |             |
|             |                     | Affe   | ects          | ACOVy, C                                    | CARRY         |                        |          |              |             |

This instruction can be repeated.

Repeat

| Syntax | Description                                                                                                                                                   |
|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
|        | The complement of the CARRY bit (1) and the unsigned content addressed by AR1 (F000h) are subtracted from the content of AC0 and the result is stored in AC1. |

| Before |         |      | After |    |      |      |
|--------|---------|------|-------|----|------|------|
| AC0    | 00 EC00 | 0000 | AC0   | 00 | EC00 | 0000 |
| AC1    | 00 0000 | 0000 | AC1   | 00 | EBFF | OFFF |
| AR1    |         | 0302 | AR1   |    |      | 0302 |
| 302    |         | F000 | 302   |    |      | F000 |
| CARRY  |         | 0    | CARRY |    |      | 1    |

#### **Syntax Characteristics**

| No.   | Syntax                |      | Parallel<br>Enable Bit | Size | Cycles  | Pipeline |
|-------|-----------------------|------|------------------------|------|---------|----------|
| [14]  | ACy = ACx - uns(Smem) |      | No                     | 3    | 1       | Х        |
| Opcod | e                     | 1101 | 1111 AA                | AA A | AAI SSI | DD 111u  |

## **Operands**

ACx, ACy, Smem

#### Description

This instruction subtracts the content of a memory (Smem) location from an accumulator content ACx.

- ☐ The operation is performed on 40 bits in the D-unit ALU.
- ☐ Input operands are extended to 40 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 40 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 40 bits according to SXMD.
- Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.
- ☐ When an overflow is detected, the accumulator is saturated according to SATD.

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

**Status Bits** Affected by M40, SATD, SXMD

> ACOVy, CARRY Affects

Repeat This instruction can be repeated.

| Syntax | Description                                                                                                  |
|--------|--------------------------------------------------------------------------------------------------------------|
| \ /    | The unsigned content addressed by AR3 is subtracted from the content of AC1 and the result is stored in AC0. |

#### **Syntax Characteristics**

| No.         | Syntax      |                                                                      |                                                                                                                                      |                                                            | Parallel<br>Enable Bit | Size     | Cycles     | Pipeline     |  |
|-------------|-------------|----------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------|------------------------|----------|------------|--------------|--|
| [15]        | ACy = ACx - | (uns(Smem) <<                                                        | #SHIFTW)                                                                                                                             |                                                            | No                     | 4        | 1          | Х            |  |
| Opcode      | e           |                                                                      | 1111                                                                                                                                 | 1001 AAAA                                                  | AAAI ux                | SH II    | FTW   SSI  | DD 01xx      |  |
| Operan      | nds         | ACx, ACy,                                                            | SHIFTW, S                                                                                                                            | mem                                                        |                        |          |            |              |  |
| Description |             |                                                                      | This instruction subtracts the content of a memory (Smem) location shifted the 6-bit value, SHIFTW, from an accumulator content ACx. |                                                            |                        |          |            |              |  |
|             |             | ☐ The op                                                             | eration is p                                                                                                                         | erformed on 40                                             | bits in the [          | D-unit s | hifter.    |              |  |
|             |             | ☐ Input o                                                            | oerands ar                                                                                                                           | e extended to 4                                            | 0 bits accore          | ding to  | uns.       |              |  |
|             |             |                                                                      | •                                                                                                                                    | uns keyword is a<br>/ location is zero                     |                        |          | •          | the content  |  |
|             |             | cor                                                                  | •                                                                                                                                    | uns keyword is<br>memory locatior                          |                        |          |            |              |  |
|             |             | ☐ The shift operation is equivalent to the signed shift instruction. |                                                                                                                                      |                                                            |                        |          |            |              |  |
|             |             | subtrac                                                              | tion borrov                                                                                                                          | on and CARRY<br>bit is reported<br>plement of the C        | in the CARF            | RY state |            |              |  |
|             |             | ☐ When a SATD.                                                       | an overflow                                                                                                                          | is detected, the                                           | e accumulato           | or is sa | turated a  | ccording to  |  |
|             |             | Compatibility with C54x devices (C54CM = 1)                          |                                                                                                                                      |                                                            |                        |          |            |              |  |
|             |             | C54CM = 1                                                            | , an interm<br>d no overfl                                                                                                           | s executed with I<br>rediary shift ope<br>ow detection, re | ration is per          | rforme   | d as if M4 | 0 is locally |  |
| Status      | Bits        | Affected by                                                          | C54C                                                                                                                                 | M, M40, SATD,                                              | SXMD                   |          |            |              |  |
|             |             | Affects                                                              | ACO\                                                                                                                                 | /y, CARRY                                                  |                        |          |            |              |  |

## Example

Repeat

| Syntax                         | Description                                                                                                                          |
|--------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| AC0 = AC1 - (uns(*AR3) << #31) | The unsigned content addressed by AR3 shifted left by 31 bits is subtracted from the content of AC1 and the result is stored in AC0. |

This instruction can be repeated.

| No. Syntax           |                                                                                                          |                                                                                            | Parallel<br>Enable Bit | Size     | Cycles       | Pipeline   |  |  |  |
|----------------------|----------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------|------------------------|----------|--------------|------------|--|--|--|
| [16]  ACy = ACx - db | ol(Lmem)                                                                                                 |                                                                                            | No                     | 3        | 1            | X          |  |  |  |
| Opcode               |                                                                                                          | 1110                                                                                       | 1101 AA                | AA A     | AAI   SSD    | D 001n     |  |  |  |
| Operands             | ACx, ACy, Ln                                                                                             | nem                                                                                        |                        |          |              |            |  |  |  |
| Description          | This instruction subtracts the content of data memory operand dbl(Lmem) from an accumulator content ACx. |                                                                                            |                        |          |              |            |  |  |  |
|                      | ☐ The data                                                                                               | memory operand dbl(Lm                                                                      | nem) addres            | ses ar   | e aligned:   |            |  |  |  |
|                      |                                                                                                          | ■ if Lmem address is even: most significant word = Lmem, least significant word = Lmem + 1 |                        |          |              |            |  |  |  |
|                      | ■ if Lmem address is odd: most significant word = Lmem, leasignificant word = Lmem - 1                   |                                                                                            |                        |          |              |            |  |  |  |
|                      | ☐ The operation is performed on 40 bits in the D-unit ALU.                                               |                                                                                            |                        |          |              |            |  |  |  |
|                      | ☐ Input operands are sign extended to 40 bits according to SXMD.                                         |                                                                                            |                        |          |              |            |  |  |  |
|                      | subtraction                                                                                              | detection and CARRY on borrow bit is reported it is complement of the C                    | in the CARF            | RY stat  |              |            |  |  |  |
|                      | ☐ When an SATD.                                                                                          | overflow is detected, the                                                                  | accumulato             | or is sa | turated ad   | cording to |  |  |  |
|                      | Compatibility with C54x devices (C54CM = 1)                                                              |                                                                                            |                        |          |              |            |  |  |  |
|                      | When this ins                                                                                            | struction is executed with                                                                 | M40 = 0, co            | ompati   | bility is en | sured.     |  |  |  |
| Status Bits          | Affected by                                                                                              | M40, SATD, SXMD                                                                            |                        |          |              |            |  |  |  |
|                      | Affects                                                                                                  | ACOVy, CARRY                                                                               |                        |          |              |            |  |  |  |
| Repeat               | This instruction                                                                                         | on can be repeated.                                                                        |                        |          |              |            |  |  |  |
| Example              |                                                                                                          |                                                                                            |                        |          |              |            |  |  |  |

| Syntax | Description                                                                                                                                                                                                                      |
|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|        | The content (long word) addressed by AR3 and AR3 + 1 is subtracted from the content of AC1 and the result is stored in AC0. Because this instruction is a long-operand instruction, AR3 is incremented by 2 after the execution. |

| No.                             | Cumtav                 |                                                                                                                                                                                                 |               |                                     |           | Parallel      | Ci-c     | Cyala -      | Din alin -   |
|---------------------------------|------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|-------------------------------------|-----------|---------------|----------|--------------|--------------|
| No.                             | Syntax                 |                                                                                                                                                                                                 |               |                                     |           | Enable Bit    | Size     | Cycles       | Pipeline     |
| [17]                            | ACy = <b>dbl(</b> Lmer | n <b>)</b> –                                                                                                                                                                                    | ACx           |                                     |           | No            | 3        | 1            | Х            |
| Opcode                          |                        |                                                                                                                                                                                                 |               |                                     | 1110      | 1101 AA       | AA A     | AAI SSI      | DD 010x      |
| Operar                          | nds                    | AC                                                                                                                                                                                              | x, ACy, Lm    | em                                  |           |               |          |              |              |
| Description                     |                        |                                                                                                                                                                                                 |               | n subtracts an ac<br>and dbl(Lmem). | cumula    | tor content A | Cx froi  | m the con    | tent of data |
|                                 |                        |                                                                                                                                                                                                 | The data r    | memory operand                      | d dbl(Ln  | nem) addres   | sses ar  | e aligned    | :            |
|                                 |                        | ■ if Lmem address is even: most significant word = Lmem, least<br>significant word = Lmem + 1                                                                                                   |               |                                     |           |               |          |              |              |
|                                 |                        | ■ if Lmem address is odd: most significant word = Lmem, least<br>significant word = Lmem - 1                                                                                                    |               |                                     |           |               |          | nem, least   |              |
|                                 |                        | ☐ The operation is performed on 40 bits in the D-unit ALU.                                                                                                                                      |               |                                     |           |               |          |              |              |
|                                 |                        | ☐ Input operands are sign extended to 40 bits according to SXMD.                                                                                                                                |               |                                     |           |               |          |              |              |
|                                 |                        | Overflow detection and CARRY status bit depends on M40. The<br>subtraction borrow bit is reported in the CARRY status bit; the borrow bit<br>is the logical complement of the CARRY status bit. |               |                                     |           |               |          |              |              |
|                                 |                        |                                                                                                                                                                                                 | When an o     | overflow is detec                   | ted, the  | e accumulate  | or is sa | turated a    | ccording to  |
| Compatibility with C54x devices |                        |                                                                                                                                                                                                 |               | ices (C                             | 54CM = 1) |               |          |              |              |
|                                 |                        | Wh                                                                                                                                                                                              | nen this inst | ruction is execut                   | ted with  | M40 = 0, c    | ompati   | bility is er | sured.       |
| Status                          | Bits                   | Aff                                                                                                                                                                                             | ected by      | M40, SATD, S                        | XMD       |               |          |              |              |
|                                 |                        | Aff                                                                                                                                                                                             | ects          | ACOVy, CARF                         | RY        |               |          |              |              |
| Repeat                          | :                      | Thi                                                                                                                                                                                             | is instructio | n can be repeate                    | ed.       |               |          |              |              |
| Examp                           | le                     |                                                                                                                                                                                                 |               |                                     |           |               |          |              |              |

| Syntax                                  | Description                                                                                                                 |
|-----------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| • • • • • • • • • • • • • • • • • • • • | The content of AC1 is subtracted from the content (long word) addressed by AR3 and AR3 + 1 and the result is stored in AC0. |

#### **Syntax Characteristics**

| No.   | Syntax                              |      | Parallel<br>Enable Bit | Size  | Cycles  | Pipeline |
|-------|-------------------------------------|------|------------------------|-------|---------|----------|
| [18]  | ACx = (Xmem << #16) - (Ymem << #16) |      | No                     | 3     | 1       | Х        |
| Opcod | e                                   | 1000 | 0001 XX                | XM MI | NYY YMM | M 01DD   |

#### Operands

ACx, Xmem, Ymem

#### Description

This instruction subtracts the content of data memory operand Ymem, shifted left 16 bits, from the content of data memory operand Xmem, shifted left 16 bits

- ☐ The operation is performed on 40 bits in the D-unit ALU.
- ☐ Input operands are sign extended to 40 bits according to SXMD.
- ☐ The shift operation is equivalent to the signed shift instruction.
- Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.
- ☐ When an overflow is detected, the accumulator is saturated according to SATD.

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, an intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.

#### Status Bits

Affected by

C54CM, M40, SATD, SXMD

Affects

ACOVx, CARRY

#### Repeat

This instruction can be repeated.

| Syntax                              | Description                                                                                                                                                   |
|-------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AC0 = (*AR3 << #16) - (*AR4 << #16) | The content addressed by AR4 shifted left by 16 bits is subtracted from the content addressed by AR3 shifted left by 16 bits and the result is stored in AC0. |

## SUB::MOV

#### Subtraction with Parallel Store Accumulator Content to Memory

#### **Syntax Characteristics**

| No.    | Syntax                                |                                                                                                                                                                                                                                                      |                                                                                                                                                                                 | Parallel<br>Enable Bit                          | Size                        | Cycles                               | Pipeline                                |  |  |
|--------|---------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------|-----------------------------|--------------------------------------|-----------------------------------------|--|--|
| [1]    | ACy = (Xmem -<br>Ymem = <b>HI(</b> AC |                                                                                                                                                                                                                                                      |                                                                                                                                                                                 | No                                              | 4                           | 1                                    | Х                                       |  |  |
| Opcod  | e                                     |                                                                                                                                                                                                                                                      | 1000 0111   XXXI                                                                                                                                                                | MMYY YMM                                        | ИM SS                       | SDD   101                            | .x xxxx                                 |  |  |
| Opera  | nds                                   | AC                                                                                                                                                                                                                                                   | x, ACy, T2, Xmem, Ymem                                                                                                                                                          |                                                 |                             |                                      |                                         |  |  |
| Descri | ption                                 | Thi                                                                                                                                                                                                                                                  | is instruction performs two operat                                                                                                                                              | ions in paralle                                 | l: subti                    | action ar                            | nd store.                               |  |  |
|        |                                       |                                                                                                                                                                                                                                                      | e first operation subtracts an accu<br>mory operand Xmem shifted left l                                                                                                         |                                                 | nt from                     | the cont                             | ent of data                             |  |  |
|        |                                       |                                                                                                                                                                                                                                                      | The operation is performed on 4                                                                                                                                                 | 0 bits in the D                                 | -unit A                     | LU.                                  |                                         |  |  |
|        |                                       |                                                                                                                                                                                                                                                      | ☐ Input operands are sign extended to 40 bits according to SXMD.                                                                                                                |                                                 |                             |                                      |                                         |  |  |
| _      |                                       |                                                                                                                                                                                                                                                      | ☐ The shift operation is equivalent to the signed shift instruction.                                                                                                            |                                                 |                             |                                      |                                         |  |  |
|        |                                       |                                                                                                                                                                                                                                                      | Overflow detection and CARF subtraction borrow bit is reported is the logical complement of the intermediary shift operation is per no overflow detection, report, a operation. | d in the CARR<br>CARRY status<br>erformed as if | Y statu<br>bit. W<br>M40 is | us bit; the<br>hen C540<br>locally s | borrow bit<br>CM = 1, an<br>et to 1 and |  |  |
|        |                                       |                                                                                                                                                                                                                                                      | When an overflow is detected, the SATD.                                                                                                                                         | ne accumulato                                   | r is sat                    | turated a                            | ccording to                             |  |  |
|        |                                       | The second operation shifts the accumulator ACy by the content of T2 a stores ACy(31–16) to data memory operand Ymem. If the 16-bit value in is not within –32 to +31, the shift is saturated to –32 or +31 and the shift performed with this value. |                                                                                                                                                                                 |                                                 |                             |                                      | alue in T2                              |  |  |
|        |                                       |                                                                                                                                                                                                                                                      | The input operand is shifted in the                                                                                                                                             | ne D-unit shift                                 | er acco                     | ording to                            | SXMD.                                   |  |  |
|        |                                       |                                                                                                                                                                                                                                                      | After the shift, the high part of th the memory location.                                                                                                                       | e accumulato                                    | r, ACy(                     | 31–16), i                            | s stored to                             |  |  |

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When this instruction is executed with C54CM = 1, the 6 LSBs of T2 are used to determine the shift quantity. The 6 LSBs of T2 define a shift quantity within -32 to +31. When the 16-bit value in T2 is between -32 to -17, a modulo 16 operation transforms the shift quantity to within -16 to -1.

|             | ☐ If the SST bit = 1 and the SXMD bit = 0, then the saturate and uns key are applied to the instruction regardless of the optional keywords se by the user, with the following syntax: ACy = (Xmem << #16) - ACx, Ymem = HI(saturate(uns(ACy << T2))) |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |  |  |  |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|--|--|
|             | _                                                                                                                                                                                                                                                     | is applied to<br>by the user<br>ACy = | bit = 1 and the SXMD bit = 1, then only the saturate keyword to the instruction regardless of the optional keywords selected to the instruction regardless of the optional keywords selected to the instruction regardless of the optional keywords selected to the instruction regardless of the optional keywords selected to the instruction regardless of the optional keywords selected to the instruction regardless of the optional keywords selected to the instruction regardless of the optional keywords selected to the instruction regardless of the optional keywords selected to the instruction regardless of the optional keywords selected to the instruction regardless of the optional keywords selected to the instruction regardless of the optional keywords selected to the instruction regardless of the optional keywords selected to the instruction regardless of the optional keywords selected to the instruction regardless of the optional keywords selected to the instruction regardless of the optional keywords selected to the instruction regardless of the optional keywords selected to the instruction regardless of the optional keywords are regardless of the optional keywords and the instruction regardless of the optional keywords are regardless of the optional keywords are regardless of the optional keywords are regardless. |  |  |  |
| Status Bits | Affe                                                                                                                                                                                                                                                  | cted by                               | C54CM, M40, RDM, SATD, SST, SXMD                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |  |  |  |
|             | Affe                                                                                                                                                                                                                                                  | cts                                   | ACOVy, CARRY                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |  |  |  |
| Repeat      | This                                                                                                                                                                                                                                                  | instruction                           | on can be repeated.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |  |  |
| See Also    | See the following other related instructions:                                                                                                                                                                                                         |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |  |  |  |
|             |                                                                                                                                                                                                                                                       | Addition or                           | Subtraction Conditionally                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |  |  |  |
|             |                                                                                                                                                                                                                                                       | Addition or                           | Subtraction Conditionally with Shift                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |  |  |  |
|             | ☐ Addition, Subtraction, or Move Accumulator Content Conditionally                                                                                                                                                                                    |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |  |  |  |
|             |                                                                                                                                                                                                                                                       | Dual 16-Bit                           | Addition and Subtraction                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |  |  |  |
|             |                                                                                                                                                                                                                                                       | Dual 16-Bit                           | Subtractions                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |  |  |  |
|             |                                                                                                                                                                                                                                                       | Dual 16-Bit                           | Subtraction and Addition                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |  |  |  |
|             |                                                                                                                                                                                                                                                       | Subtraction                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |  |  |  |
|             |                                                                                                                                                                                                                                                       | Subtract Co                           | onditionally                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |  |  |  |

| Syntax                                             | Description                                                                                                                                                                                                                                                                  |
|----------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AC0 = (*AR3 << #16) – AC1,<br>*AR4 = HI(AC0 << T2) | Both instructions are performed in parallel. The content of AC1 is subtracted from the content addressed by AR3 shifted left by 16 bits and the result is stored in AC0. The content of AC0 is shifted by the content of T2, and AC0(31–16) is stored at the address of AR4. |

## Swap Accumulator Content

## **Syntax Characteristics**

| No.     | Syntax        |                                                                                                                                                         |                                                                            | Parallel<br>Enable Bit | Size    | Cycles     | Pipeline |
|---------|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------|------------------------|---------|------------|----------|
|         | swap(ACx, ACy | <b>/</b> )                                                                                                                                              |                                                                            |                        |         |            |          |
| [1]     | swap(AC0, AC  | 2)                                                                                                                                                      |                                                                            | Yes                    | 2       | 1          | X        |
| [2]     | swap(AC1, AC  | 3)                                                                                                                                                      |                                                                            | Yes                    | 2       | 1          | X        |
| Opcode  | е             | swap(AC0,                                                                                                                                               | AC2)                                                                       | 010                    | )1 11   | 1E 000     | 0 0000   |
|         |               | swap(AC1,                                                                                                                                               | AC3)                                                                       | 010                    | 1 11    | 1E 000     | 0 0001   |
| Operar  | nds           | ACx, ACy                                                                                                                                                |                                                                            |                        |         |            |          |
| Descrip | otion         | This instruction performs parallel moves between two accumulators. The operations are performed in a dedicated datapath independent of the D operators. |                                                                            |                        |         |            |          |
|         |               | accumulator (                                                                                                                                           | n moves the content of the ACy), and reciprocall to the first accumulator. |                        |         |            |          |
|         |               | Accumulator s                                                                                                                                           | wapping is performed i                                                     | n the execute          | e phase | e of the p | ipeline. |
| Status  | Bits          | Affected by                                                                                                                                             | none                                                                       |                        |         |            |          |
|         |               | Affects                                                                                                                                                 | none                                                                       |                        |         |            |          |
| Repeat  | :             | This instruction                                                                                                                                        | n can be repeated.                                                         |                        |         |            |          |
| See Als | SO            | See the followi                                                                                                                                         | ing other related instru                                                   | ctions:                |         |            |          |
|         |               | ☐ Swap Accu                                                                                                                                             | umulator Pair Content                                                      |                        |         |            |          |
|         |               | Swap Auxi                                                                                                                                               | iliary Register Content                                                    |                        |         |            |          |
|         |               | ☐ Swap Auxi                                                                                                                                             | iliary and Temporary R                                                     | egister Conte          | ent     |            |          |
|         |               | ☐ Swap Tem                                                                                                                                              | porary Register Conte                                                      | nt                     |         |            |          |
| Examp   | le            |                                                                                                                                                         |                                                                            |                        |         |            |          |

| Syntax         | Description                                                                |
|----------------|----------------------------------------------------------------------------|
| swap(AC0, AC2) | The content of AC0 is moved to AC2 and the content of AC2 is moved to AC0. |

| Before |    |      |      | After |    |      |      |  |
|--------|----|------|------|-------|----|------|------|--|
| AC0    | 01 | E500 | 0030 | AC0   | 00 | 2800 | 0200 |  |
| AC2    | 00 | 2800 | 0200 | AC2   | 01 | E500 | 0030 |  |

## Swap Accumulator Pair Content

#### **Syntax Characteristics**

| No.                                                                                                                                                                                                                                                                             | Syntax                     |                                                   | Parallel<br>Enable Bit | Size                    | Cycles    | Pipeline |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------|---------------------------------------------------|------------------------|-------------------------|-----------|----------|
| [1]                                                                                                                                                                                                                                                                             | swap(pair(AC               | 0), pair(AC2))                                    | Yes                    | 2                       | 1         | Х        |
| Opcod                                                                                                                                                                                                                                                                           | е                          |                                                   | 010                    | )1 11                   | 1E 000    | 1 0000   |
| Operar                                                                                                                                                                                                                                                                          | nds                        | AC0, AC2                                          |                        |                         |           |          |
| Description  This instruction performs two parallel moves between four accumulator and AC2, AC1 and AC3) in one cycle. These operations are performed dedicated datapath independent of the D-unit operators. Accum swapping is performed in the execute phase of the pipeline. |                            |                                                   |                        | ormed in a              |           |          |
|                                                                                                                                                                                                                                                                                 |                            | This instruction performs two parallel moves:     |                        |                         |           |          |
|                                                                                                                                                                                                                                                                                 |                            | ☐ the content of AC0 to AC2, and reciprocally the |                        | e content of AC2 to AC0 |           |          |
|                                                                                                                                                                                                                                                                                 |                            | ☐ the content of AC1 to AC3, and re               | eciprocally th         | e conte                 | ent of AC | 3 to AC1 |
| Status                                                                                                                                                                                                                                                                          | atus Bits Affected by none |                                                   |                        |                         |           |          |
|                                                                                                                                                                                                                                                                                 |                            | Affects none                                      |                        |                         |           |          |
| Repeat                                                                                                                                                                                                                                                                          | t                          | This instruction can be repeated.                 |                        |                         |           |          |
| See Als                                                                                                                                                                                                                                                                         | so                         | See the following other related instructions:     |                        |                         |           |          |
|                                                                                                                                                                                                                                                                                 |                            | Swap Accumulator Content                          |                        |                         |           |          |
|                                                                                                                                                                                                                                                                                 |                            | ☐ Swap Auxiliary Register Pair Con                | itent                  |                         |           |          |
|                                                                                                                                                                                                                                                                                 |                            | ☐ Swap Auxiliary and Temporary R                  | egister Pair C         | Conten                  | t         |          |

#### **Example**

| Syntax                     | Description                                                                   |
|----------------------------|-------------------------------------------------------------------------------|
| swap(pair(AC0), pair(AC2)) | The following two swap instructions are performed in parallel: the content of |
|                            | AC0 is moved to AC2 and the content of AC2 is moved to AC0, and the content   |
|                            | of AC1 is moved to AC3 and the content of AC3 is moved to AC1.                |

☐ Swap Temporary Register Pair Content

| Before |    |      |      | After |    |      |      |
|--------|----|------|------|-------|----|------|------|
| AC0    | 01 | E500 | 0030 | AC0   | 00 | 2800 | 0200 |
| AC1    | 00 | FFFF | 0000 | AC1   | 00 | 8800 | 0800 |
| AC2    | 00 | 2800 | 0200 | AC2   | 01 | E500 | 0030 |
| AC3    | 00 | 8800 | 0800 | AC3   | 00 | FFFF | 0000 |

## Swap Auxiliary Register Content

|         |               |                   |                                                                                 | Parallel      |          |            |             |
|---------|---------------|-------------------|---------------------------------------------------------------------------------|---------------|----------|------------|-------------|
| No.     | Syntax        |                   |                                                                                 | Enable Bit    | Size     | Cycles     | Pipeline    |
|         | swap(ARx, ARy | <b>'</b> )        |                                                                                 |               |          |            |             |
| [1]     | swap(AR0, AR1 | 1)                |                                                                                 | Yes           | 2        | 1          | AD          |
| [2]     | swap(AR0, AR2 | 2)                |                                                                                 | Yes           | 2        | 1          | AD          |
| [3]     | swap(AR1, AR3 | 3)                |                                                                                 | Yes           | 2        | 1          | AD          |
| Opcode  | e             | swap(AR0,         | AR1)                                                                            | 01            | 01 11    | 1E 001     | 1 1000      |
|         |               | swap(AR0,         | AR2)                                                                            | 01            | 01 11    | 1E 000     | 0 1000      |
|         |               | swap(AR1,         | AR3)                                                                            | 01            | 01 11    | 1E 000     | 0 1001      |
| Operan  | nds           | ARx, ARy          |                                                                                 | '             |          | '          |             |
| Descrip | otion         |                   | n performs parallel mons are performed in a rs.                                 |               |          | •          | •           |
|         |               | second auxilia    | n moves the content of<br>try register (ARy), and<br>ry register to the first a | reciprocally  | moves    | •          | ,           |
|         |               | Auxiliary regist  | er swapping is perform                                                          | ed in the add | dress pl | nase of th | e pipeline. |
| Status  | Bits          | Affected by       | none                                                                            |               |          |            |             |
|         |               | Affects           | none                                                                            |               |          |            |             |
| Repeat  | :             | This instruction  | n can be repeated.                                                              |               |          |            |             |
| See Als | so            | See the following | ing other related instru                                                        | ctions:       |          |            |             |
|         |               | ☐ Swap Acc        | umulator Content                                                                |               |          |            |             |
|         |               | ☐ Swap Auxi       | iliary and Temporary R                                                          | egister Cont  | ent      |            |             |
|         |               | ☐ Swap Auxi       | iliary Register Pair Con                                                        | tent          |          |            |             |
|         |               | ☐ Swap Tem        | porary Register Conte                                                           | nt            |          |            |             |
| Examp   | le            |                   |                                                                                 |               |          |            |             |

| Syntax         | Description                                                                |
|----------------|----------------------------------------------------------------------------|
| swap(AR0, AR2) | The content of AR0 is moved to AR2 and the content of AR2 is moved to AR0. |

| Before |      | After |      |
|--------|------|-------|------|
| AR0    | 6500 | AR0   | 0300 |
| AR2    | 0300 | AR2   | 6500 |

## **SWAPP**

## Swap Auxiliary Register Pair Content

#### **Syntax Characteristics**

| No.     | Syntax       |                                                                                                                                                                                                                                                                                    | Parallel<br>Enable Bit | Size   | Cycles    | Pipeline  |
|---------|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|--------|-----------|-----------|
| [1]     | swap(pair(AF | R0), pair(AR2))                                                                                                                                                                                                                                                                    | Yes                    | 2      | 1         | AD        |
| Opcode  | e            |                                                                                                                                                                                                                                                                                    | 010                    | )1 11  | 1E 000    | 1 1000    |
| Operar  | nds          | AR0, AR2                                                                                                                                                                                                                                                                           |                        |        |           |           |
| Descri  | ption        | This instruction performs two parallel moves between four auxiliary regi (AR0 and AR2, AR1 and AR3) in one cycle. These operations are perform a dedicated datapath independent of the A-unit operators. Auxiliary reg swapping is performed in the address phase of the pipeline. |                        |        |           | performed |
|         |              | This instruction performs two para                                                                                                                                                                                                                                                 | llel moves:            |        |           |           |
|         |              | the content of AR0 to AR2, an                                                                                                                                                                                                                                                      | d reciprocally th      | e cont | ent of AR | 2 to AR0  |
|         |              | the content of AR1 to AR3, an                                                                                                                                                                                                                                                      | d reciprocally th      | e cont | ent of AR | 3 to AR1  |
| Status  | Bits         | Affected by none                                                                                                                                                                                                                                                                   |                        |        |           |           |
|         |              | Affects none                                                                                                                                                                                                                                                                       |                        |        |           |           |
| Repeat  | t ·          | This instruction can be repeated.                                                                                                                                                                                                                                                  |                        |        |           |           |
| See Als | so           | See the following other related instructions:                                                                                                                                                                                                                                      |                        |        |           |           |
|         |              | Swap Accumulator Pair Conte                                                                                                                                                                                                                                                        | ent                    |        |           |           |
|         |              | Swap Auxiliary Register Conte                                                                                                                                                                                                                                                      | ent                    |        |           |           |
|         |              | Swap Auxiliary and Temporary                                                                                                                                                                                                                                                       | / Register Pair C      | Conten | t         |           |
|         |              | ☐ Swap Temporary Register Pai                                                                                                                                                                                                                                                      | r Content              |        |           |           |

| Syntax                     | Description                                                                                                                                               |
|----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| swap(pair(AR0), pair(AR2)) | The following two swap instructions are performed in parallel: the content of AR0 is moved to AR2 and the content of AR2 is moved to AR0, and the content |
|                            | of AR1 is moved to AR3 and the content of AR3 is moved to AR1.                                                                                            |

| Before |      | After |      |
|--------|------|-------|------|
| AR0    | 0200 | AR0   | 6788 |
| AR1    | 0300 | AR1   | 0200 |
| AR2    | 6788 | AR2   | 0200 |
| AR3    | 0200 | AR3   | 0300 |

## Swap Auxiliary and Temporary Register Content

| No.     | Syntax                                                                                                                                                                                     |                             | Parall<br>Enable    |            | Cycles | Pipeline  |
|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------|---------------------|------------|--------|-----------|
|         | swap(ARx, Tx)                                                                                                                                                                              |                             |                     |            |        |           |
| [1]     | swap(AR4, T0)                                                                                                                                                                              |                             | Yes                 | 2          | 1      | AD        |
| [2]     | swap(AR5, T1)                                                                                                                                                                              |                             | Yes                 | 2          | 1      | AD        |
| [3]     | swap(AR6, T2)                                                                                                                                                                              |                             | Yes                 | 2          | 1      | AD        |
| [4]     | swap(AR7, T3)                                                                                                                                                                              |                             | Yes                 | 2          | 1      | AD        |
| Opcod   | e                                                                                                                                                                                          | swap(AR4, T0)               |                     | 0101 11    | 1E 000 | 0 1100    |
|         |                                                                                                                                                                                            | swap(AR5, T1)               |                     | 0101 11    | 1E 000 | 0 1101    |
|         |                                                                                                                                                                                            | swap(AR6, T2)               |                     | 0101 11    | 1E 000 | 0 1110    |
|         |                                                                                                                                                                                            | swap(AR7, T3)               |                     | 0101 11    | 1E 000 | 0 1111    |
| Operar  | nds                                                                                                                                                                                        | ARx, Tx                     |                     |            |        |           |
| Descri  | tion This instruction performs parallel moves between auxiliary registers a temporary registers. These operations are performed in a dedicated datapa independent of the A-unit operators. |                             |                     |            |        |           |
|         | This instruction moves the content of the auxiliary register (ARx) to the temporary register (Tx), and reciprocally moves the content of the temporaregister to the auxiliary register.    |                             |                     |            |        | ,         |
|         | Auxiliary and temporary register swapping is performed in the address pha of the pipeline.                                                                                                 |                             |                     |            |        | ess phase |
| Status  | Bits                                                                                                                                                                                       | Affected by none            |                     |            |        |           |
|         |                                                                                                                                                                                            | Affects none                |                     |            |        |           |
| Repeat  | i                                                                                                                                                                                          | This instruction can be rep | eated.              |            |        |           |
| See Als | so                                                                                                                                                                                         | See the following other re  | ated instructions:  |            |        |           |
|         |                                                                                                                                                                                            | ☐ Swap Accumulator Co       | ontent              |            |        |           |
|         |                                                                                                                                                                                            | Swap Auxiliary Regist       | er Content          |            |        |           |
|         |                                                                                                                                                                                            | ☐ Swap Auxiliary and Te     | mporary Register Pa | air Conten | t      |           |
|         |                                                                                                                                                                                            | ☐ Swap Auxiliary and Te     | mporary Register Pa | airs Conte | nt     |           |
|         |                                                                                                                                                                                            | ☐ Swap Temporary Reg        | ister Content       |            |        |           |

| Syntax        | Description                                                              |
|---------------|--------------------------------------------------------------------------|
| swap(AR4, T0) | The content of AR4 is moved to T0 and the content of T0 is moved to AR4. |

| Before |      | After |      |
|--------|------|-------|------|
| TO     | 6500 | T0    | 0300 |
| AR4    | 0300 | AR4   | 6500 |

## **SWAPP**

## Swap Auxiliary and Temporary Register Pair Content

| No.     | Syntax         |            |                            |                         |                                                                      | Parallel<br>Enable Bit | Size             | Cycles                 | Pipeline                   |
|---------|----------------|------------|----------------------------|-------------------------|----------------------------------------------------------------------|------------------------|------------------|------------------------|----------------------------|
|         | swap(pair(ARx) | ), pai     | r(Tx) <b>)</b>             |                         |                                                                      |                        |                  |                        |                            |
| [1]     | swap(pair(AR4  | l), pai    | ir(T0))                    |                         |                                                                      | Yes                    | 2                | 1                      | AD                         |
| [2]     | swap(pair(AR6  | i), pai    | ir(T2))                    |                         |                                                                      | Yes                    | 2                | 1                      | AD                         |
| Opcod   | e              | s'         | wap(pair                   | (AR4),                  | pair(T0))                                                            | 010                    | )1 11            | 1E   00C               | 1 1100                     |
|         |                | s'         | wap(pair                   | (AR6),                  | pair(T2))                                                            | 010                    | 1 11             | 1E 000                 | 1 1110                     |
| Operar  | nds            | AR         | x, Tx                      |                         |                                                                      |                        |                  |                        |                            |
| Descri  | ption          | and<br>a d | l two tempo<br>ledicated d | rary regis<br>atapath i | s two parallel<br>sters in one cy<br>ndependent o<br>ping is perforn | cle. These o           | oeratio<br>opera | ns are pe<br>ators. Au | erformed in<br>xiliary and |
|         |                | Inst       | truction [1] p             | performs                | two parallel m                                                       | ioves:                 |                  |                        |                            |
|         |                |            | the conten                 | t of AR4                | to T0, and rec                                                       | iprocally the          | conter           | nt of T0 to            | AR4                        |
|         |                |            | the conten                 | t of AR5                | to T1, and rec                                                       | iprocally the          | conter           | nt of T1 to            | AR5                        |
|         |                | Inst       | ruction [2]                | performs                | two parallel m                                                       | oves:                  |                  |                        |                            |
|         |                |            | the conten                 | t of AR6                | to T2, and rec                                                       | iprocally the          | conter           | nt of T2 to            | AR6                        |
|         |                |            | the conten                 | t of AR7                | to T3, and rec                                                       | iprocally the          | conter           | nt of T3 to            | AR7                        |
| Status  | Bits           | Affe       | ected by                   | none                    |                                                                      |                        |                  |                        |                            |
|         |                | Affe       | ects                       | none                    |                                                                      |                        |                  |                        |                            |
| Repeat  | t              | This       | s instruction              | n can be i              | repeated.                                                            |                        |                  |                        |                            |
| See Als | so             | See        | the followi                | ng other                | related instruc                                                      | ctions:                |                  |                        |                            |
|         |                |            | Swap Accu                  | umulator                | Pair Content                                                         |                        |                  |                        |                            |
|         |                |            | Swap Auxi                  | liary Reg               | ister Pair Con                                                       | tent                   |                  |                        |                            |
|         |                |            | Swap Auxi                  | liary and               | Temporary Re                                                         | egister Conte          | ent              |                        |                            |
|         |                |            | Swap Auxi                  | liary and               | Temporary Re                                                         | egister Pairs          | Conte            | nt                     |                            |
|         |                |            | Swap Tem                   | porary R                | egister Pair Co                                                      | ontent                 |                  |                        |                            |

## Example

| Syntax                    | Description                                                                                                                                             |
|---------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| swap(pair(AR4), pair(T0)) | The following two swap instructions are performed in parallel: the content of AR4 is moved to T0 and the content of T0 is moved to AR4, and the content |
|                           | of AR5 is moved to T1 and the content of T1 is moved to AR5.                                                                                            |

| Before |      | After |      |
|--------|------|-------|------|
| AR4    | 0200 | AR4   | 6788 |
| AR5    | 0300 | AR5   | 0200 |
| TO     | 6788 | T0    | 0200 |
| T1     | 0200 | T1    | 0300 |

Instruction Set Descriptions 5-636

## Swap Auxiliary and Temporary Register Pairs Content

| No.     | Syntax        |                                                 |                                                                                                                 | Parallel<br>Enable Bit                     | Size                      | Cycles                              | Pipeline              |
|---------|---------------|-------------------------------------------------|-----------------------------------------------------------------------------------------------------------------|--------------------------------------------|---------------------------|-------------------------------------|-----------------------|
| [1]     | swap(block(AF | R4), block(T0))                                 |                                                                                                                 | Yes                                        | 2                         | 1                                   | AD                    |
| Opcode  | e             |                                                 |                                                                                                                 | 010                                        | )1 11                     | 1E   001                            | 0 1100                |
| Operan  | ds            | AR4, T0                                         |                                                                                                                 |                                            |                           |                                     |                       |
| Descrip | otion         | (AR4, AR5, AR<br>in one cycle.<br>independent o | performs four parallel 6, and AR7) and four to These operations are f the A-unit operator formed in the address | emporary reg<br>performed<br>rs. Auxiliary | isters (<br>in a d<br>and | (T0, T1, T<br>dedicated<br>temporar | 2, and T3) I datapath |
|         |               | This instruction                                | performs four parallel                                                                                          | moves:                                     |                           |                                     |                       |
|         |               | the content                                     | of AR4 to T0, and rec                                                                                           | iprocally the                              | conter                    | t of T0 to                          | AR4                   |
|         |               | ☐ the content                                   | of AR5 to T1, and rec                                                                                           | iprocally the                              | conten                    | t of T1 to                          | AR5                   |
|         |               | ☐ the content                                   | of AR6 to T2, and rec                                                                                           | iprocally the                              | conten                    | t of T2 to                          | AR6                   |
|         |               | ☐ the content                                   | of AR7 to T3, and rec                                                                                           | iprocally the                              | conten                    | t of T3 to                          | AR7                   |
| Status  | Bits          | Affected by                                     | none                                                                                                            |                                            |                           |                                     |                       |
|         |               | Affects                                         | none                                                                                                            |                                            |                           |                                     |                       |
| Repeat  |               | This instruction                                | can be repeated.                                                                                                |                                            |                           |                                     |                       |
| See Als | 80            | See the following                               | ng other related instruc                                                                                        | tions:                                     |                           |                                     |                       |
|         |               | ☐ Swap Auxil                                    | iary and Temporary Re                                                                                           | egister Conte                              | nt                        |                                     |                       |
|         |               | ☐ Swap Auxil                                    | iary and Temporary Re                                                                                           | egister Pair C                             | content                   | t                                   |                       |

## Example

| Syntax                       | Description                                                                                                                                                                                                                                                                                                                                                             |
|------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| swap (block(AR4), block(T0)) | The following four swap instructions are performed in parallel: the content of AR4 is moved to T0 and the content of T0 is moved to AR4, the content of AR5 is moved to T1 and the content of T1 is moved to AR5, the content of AR6 is moved to T2 and the content of T2 is moved to AR6, and the content of AR7 is moved to T3 and the content of T3 is moved to AR7. |

| Before |      | After |      |
|--------|------|-------|------|
| AR4    | 0200 | AR4   | 0030 |
| AR5    | 0300 | AR5   | 0200 |
| AR6    | 0240 | AR6   | 3400 |
| AR7    | 0400 | AR7   | 0FD3 |
| T0     | 0030 | T0    | 0200 |
| T1     | 0200 | T1    | 0300 |
| T2     | 3400 | T2    | 0240 |
| Т3     | 0FD3 | Т3    | 0400 |

5-638 Instruction Set Descriptions

## Swap Temporary Register Content

## **Syntax Characteristics**

| No.    | Syntax       |                                                                                                                                                                  | Parallel<br>Enable Bit | Size   | Cycles    | Pipeline   |
|--------|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|--------|-----------|------------|
|        | swap(Tx, Ty) |                                                                                                                                                                  |                        |        |           |            |
| [1]    | swap(T0, T2) |                                                                                                                                                                  | Yes                    | 2      | 1         | AD         |
| [2]    | swap(T1, T3) |                                                                                                                                                                  | Yes                    | 2      | 1         | AD         |
| Opcod  | е            | swap(T0, T2)                                                                                                                                                     | 010                    | )1 11  | 1E 000    | 0 0100     |
|        |              | swap(T1, T3)                                                                                                                                                     | 010                    | )1 11  | 1E 000    | 0 0101     |
| Operar | nds          | Tx, Ty                                                                                                                                                           |                        |        |           |            |
| Descri | ption        | This instruction performs parallel moves between two temporary register. These operations are performed in a dedicated datapath independent of A-unit operators. |                        |        |           | •          |
|        |              | This instruction moves the content of second temporary register (Ty), and second temporary register to the first                                                 | reciprocally           | moves  | •         | ` '        |
|        |              | Temporary register swapping is per pipeline.                                                                                                                     | rformed in th          | ne add | lress pha | ase of the |
| Status | Bits         | Affected by none                                                                                                                                                 |                        |        |           |            |
|        |              | Affects none                                                                                                                                                     |                        |        |           |            |
| Repeat | t            | This instruction can be repeated.                                                                                                                                |                        |        |           |            |
| See Al | so           | See the following other related instruc                                                                                                                          | ctions:                |        |           |            |
|        |              | Swap Accumulator Content                                                                                                                                         |                        |        |           |            |
|        |              | ☐ Swap Auxiliary Register Content                                                                                                                                |                        |        |           |            |
|        |              | ☐ Swap Auxiliary and Temporary R                                                                                                                                 | egister Conte          | ent    |           |            |
|        |              | ☐ Swap Temporary Register Pair C                                                                                                                                 | ontent                 |        |           |            |
| Examp  | ole          |                                                                                                                                                                  |                        |        |           |            |

| Syntax       | Description                                                            |
|--------------|------------------------------------------------------------------------|
| swap(T0, T2) | The content of T0 is moved to T2 and the content of T2 is moved to T0. |

| Before |      | After |      |
|--------|------|-------|------|
| T0     | 6500 | TO    | 0300 |
| T2     | 0300 | T2    | 6500 |

## **SWAPP**

## Swap Temporary Register Pair Content

#### **Syntax Characteristics**

| No.         | Syntax      |                                                                                                                                                                                                                                                                                      |                          | Parallel<br>Enable Bit | Size   | Cycles   | Pipeline   |
|-------------|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------|------------------------|--------|----------|------------|
| [1]         | swap(pair(T | )), pair(T2))                                                                                                                                                                                                                                                                        |                          | Yes                    | 2      | 1        | AD         |
| Opcod       | le          |                                                                                                                                                                                                                                                                                      |                          | 010                    | )1 11  | 1E 000   | 01 0100    |
| Opera       | nds         | T0, T2                                                                                                                                                                                                                                                                               |                          | •                      |        | j        |            |
| Descri      | ption       | This instruction performs two parallel moves between four temporary region (T0 and T2, T1 and T3) in one cycle. These operations are performed dedicated datapath independent of the A-unit operators. Temporary regions swapping is performed in the address phase of the pipeline. |                          |                        |        |          | ormed in a |
|             |             | This instruction                                                                                                                                                                                                                                                                     | n performs two parallel  | moves:                 |        |          |            |
|             |             | ☐ the conten                                                                                                                                                                                                                                                                         | t of T0 to T2, and recip | orocally the c         | ontent | of T2 to | ГО         |
|             |             | ☐ the conten                                                                                                                                                                                                                                                                         | t of T1 to T3, and recip | orocally the c         | ontent | of T3 to | Γ1         |
| Status Bits |             | Affected by                                                                                                                                                                                                                                                                          | none                     |                        |        |          |            |
|             |             | Affects                                                                                                                                                                                                                                                                              | none                     |                        |        |          |            |
| Repea       | t           | This instruction                                                                                                                                                                                                                                                                     | n can be repeated.       |                        |        |          |            |
| See Al      | so          | See the followi                                                                                                                                                                                                                                                                      | ng other related instru  | ctions:                |        |          |            |
|             |             | ☐ Swap Accu                                                                                                                                                                                                                                                                          | umulator Pair Content    |                        |        |          |            |
|             |             | Swap Auxi                                                                                                                                                                                                                                                                            | iliary Register Pair Cor | ntent                  |        |          |            |
|             |             | ☐ Swap Auxi                                                                                                                                                                                                                                                                          | iliary and Temporary R   | egister Pair (         | Conten | t        |            |

#### **Example**

| Syntax                   | Description                                                                                                                                                                                                      |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| swap(pair(T0), pair(T2)) | The following two swap instructions are performed in parallel: the content of T0 is moved to T2 and the content of T2 is moved to T0, and the content of T1 is moved to T3 and the content of T3 is moved to T1. |

☐ Swap Temporary Register Content

| Before |      | After |      |
|--------|------|-------|------|
| TO     | 0200 | T0    | 6788 |
| T1     | 0300 | T1    | 0200 |
| T2     | 6788 | T2    | 0200 |
| T3     | 0200 | T3    | 0300 |

## **BTST**

## Test Accumulator, Auxiliary, or Temporary Register Bit

| 1<br>1<br>AI   FSS:<br>AI   FSS:                                                                                                                                                                                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |  |  |  |  |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|--|--|--|--|
| AI   FSS                                                                                                                                                                                                                   | S 1000                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |  |  |  |  |  |
|                                                                                                                                                                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |  |  |  |  |
| AI   FSS                                                                                                                                                                                                                   | S 1001                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |  |  |  |  |  |
|                                                                                                                                                                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |  |  |  |  |
|                                                                                                                                                                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |  |  |  |  |
| <b>Description</b> This instruction performs a bit manipulation:                                                                                                                                                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |  |  |  |  |
| l is an acc                                                                                                                                                                                                                | cumulator.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |  |  |  |  |  |
| In the A-unit ALU, if the source (src) register operand is an auxiliary or temporary register.                                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |  |  |  |  |
| e instruction tests a single bit of the source register location as defined by bit addressing mode, Baddr. The tested bit is copied into the selected TCx atus bit. The generated bit address must be within:              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |  |  |  |  |
| 0–39 when accessing accumulator bits (only the 6 LSBs of the generated bit address are used to determine the bit position). If the generated bit address is not within 0–39, 0 is stored into the selected TCx status bit. |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |  |  |  |  |
| its (only th                                                                                                                                                                                                               | ne 4 LSBs<br>n).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |  |  |  |  |  |
|                                                                                                                                                                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |  |  |  |  |
|                                                                                                                                                                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |  |  |  |  |
|                                                                                                                                                                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |  |  |  |  |
| ee Also See the following other related instructions:                                                                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |  |  |  |  |
| Bit                                                                                                                                                                                                                        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |  |  |  |  |
| egister Bi                                                                                                                                                                                                                 | t                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |  |  |  |  |  |
| t                                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |  |  |  |  |
| it Pair                                                                                                                                                                                                                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |  |  |  |  |
|                                                                                                                                                                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |  |  |  |  |
| ) E E                                                                                                                                                                                                                      | ation as of the general TCx straits (only the position of the general TCx straits (only the |  |  |  |  |  |

| Syntax              | Description                                                                              |
|---------------------|------------------------------------------------------------------------------------------|
| TC1 = bit(T0, @#12) | The bit at the position defined by the register bit address (12) in T0 is tested and the |
|                     | tested bit is copied into TC1.                                                           |

| Before |      | After |      |
|--------|------|-------|------|
| TO     | FE00 | TO    | FE00 |
| TC1    | 0    | TC1   | 1    |

#### **BTSTP**

#### Test Accumulator, Auxiliary, or Temporary Register Bit Pair

| Syntax | c Characteris | etics                                                                                            |             |                     |        |            |              |
|--------|---------------|--------------------------------------------------------------------------------------------------|-------------|---------------------|--------|------------|--------------|
| No.    | Syntax        |                                                                                                  |             | arallel<br>able Bit | Size   | Cycles     | Pipeline     |
| [1]    | bit(src, pair | (Baddr <b>))</b>                                                                                 |             | No                  | 3      | 1          | Х            |
| Opcod  | le            | 1                                                                                                | 1110 11     | 00 AA               | AA AZ  | AAI FSS    | SS 010x      |
| Opera  | nds           | Baddr, src                                                                                       |             |                     |        |            |              |
| Descri | ption         | This instruction performs a bit n                                                                | nanipulatio | n:                  |        |            |              |
|        |               | ☐ In the D-unit ALU, if the sou                                                                  | rce (src) r | egister (           | operan | d is an ac | cumulator.   |
|        |               | In the A-unit ALU, if the so temporary register.                                                 | urce (src)  | register            | opera  | nd is an a | auxiliary or |
|        |               | The instruction tests two conse defined by the bit addressing mo copied into status bits TC1 and | ode, Baddr  |                     |        | •          |              |
|        |               | ■ TC1 tests the bit that is                                                                      | defined by  | y Baddr             |        |            |              |
|        |               | ■ TC2 tests the bit define                                                                       | d by Badd   | r + 1               |        |            |              |
|        |               | The generated bit address mus                                                                    | t be within | :                   |        |            |              |
|        |               | ☐ 0-38 when accessing accur<br>bit address are used to det                                       |             |                     |        |            | •            |

- address is not within 0-38:
  - If the generated bit address is 39, bit 39 of the register is stored into TC1 and 0 is stored into TC2.
  - In all other cases, 0 is stored into TC1 and TC2.
- ☐ 0-14 when accessing auxiliary or temporary register bits (only the 4 LSBs of the generated address are used to determine the bit position). If the generated bit address is not within 0-14:
  - If the generated bit address is 15, bit 15 of the register is stored into TC1 and 0 is stored into TC2.
  - In all other cases, 0 is stored into TC1 and TC2.

**Status Bits** Affected by none

> Affects TC1, TC2

| Repeat   | This instruction can be repeated.                              |  |  |  |
|----------|----------------------------------------------------------------|--|--|--|
| See Also | See the following other related instructions:                  |  |  |  |
|          | ☐ Clear Accumulator, Auxiliary, or Temporary Register Bit      |  |  |  |
|          | ☐ Complement Accumulator, Auxiliary, or Temporary Register Bit |  |  |  |
|          | ☐ Set Accumulator, Auxiliary, or Temporary Register Bit        |  |  |  |
|          | ☐ Test Accumulator, Auxiliary, or Temporary Register Bit       |  |  |  |
|          | ☐ Test Memory Bit                                              |  |  |  |

| Syntax                  | Description                                                                                                                                                                                                                             |
|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| bit(AC0, pair(AR1(T0))) | The bit at the position defined by the content of AR1(T0) in AC0 is tested and the tested bit is copied into TC1. The bit at the position defined by the content of AR1(T0) + 1 in AC0 is tested and the tested bit is copied into TC2. |

| Before |    |      |      | After |    |      |      |
|--------|----|------|------|-------|----|------|------|
| AC0    | ΕO | 1234 | 0000 | AC0   | ΕO | 1234 | 0000 |
| AR1    |    |      | 0026 | AR1   |    |      | 0026 |
| T0     |    |      | 0001 | T0    |    |      | 0001 |
| TC1    |    |      | 0    | TC1   |    |      | 1    |
| TC2    |    |      | 0    | TC2   |    |      | 0    |

Test Memory Bit

#### **Syntax Characteristics**

| No. | Syntax               | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|-----|----------------------|------------------------|------|--------|----------|
| [1] | TCx = bit(Smem, src) | No                     | 3    | 1      | Х        |
| [2] | TCx = bit(Smem, k4)  | No                     | 3    | 1      | X        |

#### Description

These instructions perform a bit manipulation in the A-unit ALU. These instructions test a single bit of a memory (Smem) location. The bit tested is defined by either the content of the source (src) operand or a 4-bit immediate value, k4. The tested bit is copied into the selected TCx status bit.

For instruction [1], the generated bit address must be within 0-15 (only the 4 LSBs of the register are used to determine the bit position).

| S | ŀа | tı | 16 | R | its |
|---|----|----|----|---|-----|
|   |    |    |    |   |     |

Affected by none

Affects TCx

#### See Also

See the following other related instructions:

- Clear Memory Bit
- ☐ Complement Memory Bit
- ☐ Set Memory Bit
- ☐ Test Accumulator, Auxiliary, or Temporary Register Bit
- Test Accumulator, Auxiliary, or Temporary Register Bit Pair
- ☐ Test and Clear Memory Bit
- ☐ Test and Complement Memory Bit
- ☐ Test and Set Memory Bit

#### Test Memory Bit

#### **Syntax Characteristics**

| No.  | Syntax                                | Parallel<br>Enable Bit | Size | Cycles | Pipeline |
|------|---------------------------------------|------------------------|------|--------|----------|
| [1a] | TC1 = bit(Smem, src)                  | No                     | 3    | 1      | Х        |
| [1b] | TC2 = bit(Smem, src)                  | No                     | 3    | 1      | X        |
|      | · · · · · · · · · · · · · · · · · · · |                        |      |        |          |

 Opcode
 TC1
 1110
 0000
 AAAA
 AAAI
 FSSS
 xxx0

 TC2
 1110
 0000
 AAAA
 AAAI
 FSSS
 xxx1

**Operands** Smem, src, TCx

**Description** This instruction performs a bit manipulation in the A-unit ALU. This instruction

tests a single bit of a memory (Smem) location. The bit tested is defined by the content of the source (src) operand. The tested bit is copied into the selected

TCx status bit.

The generated bit address must be within 0–15 (only the 4 LSBs of the register

are used to determine the bit position).

Status Bits Affected by none

Affects TCx

**Repeat** This instruction can be repeated.

| Syntax               | Description                                                                                                                  |
|----------------------|------------------------------------------------------------------------------------------------------------------------------|
| TC1 = bit(*AR0, AC0) | The bit at the position defined by AC0(3–0) in the content addressed by AR0 is tested and the tested bit is copied into TC1. |

| Before |    |      |      | After |    |      |      |  |
|--------|----|------|------|-------|----|------|------|--|
| AC0    | 00 | 0000 | 8000 | AC0   | 00 | 0000 | 0008 |  |
| *AR0   |    |      | 00C0 | *AR0  |    |      | 00C0 |  |
| TC1    |    |      | 0    | TC1   |    |      | 0    |  |

## Test Memory Bit

## **Syntax Characteristics**

| No      | Cumtav        |                  |                                                         |        | Parallel     | Ci-o      | Cycles                | Dinalina    |
|---------|---------------|------------------|---------------------------------------------------------|--------|--------------|-----------|-----------------------|-------------|
| No.     | Syntax        |                  |                                                         |        | Enable Bit   | Size      | Cycles                | Pipeline    |
| [2a]    | TC1 = bit(Sme | m, k4 <b>)</b>   |                                                         |        | No           | 3         | 1                     | Χ           |
| [2b]    | TC2 = bit(Sme | m, k4 <b>)</b>   |                                                         |        | No           | 3         | 1                     | X           |
| Opcode  | •             | TC1              |                                                         | 11101  | 1100 AA      | 7\7\ 7\7  | лл I 1 <i>-1-1</i> -1 | k xx00      |
| Opcou   | <b>G</b>      | 101              |                                                         | 1 1101 | IIOO   AA    | AA AA     | THI VVV               | L AXUU      |
|         |               | TC2              |                                                         | 1101   | 1100 AA      | AA AA     | AAI   kkk             | k xx01      |
| Operar  | nds           | k4, Smem, TC     | X                                                       |        |              |           |                       |             |
| Descrip | otion         | tests a single b | n performs a bit<br>bit of a memory<br>e value, k4. The | (Smem) | location. Th | ne bit te | ested is d            | efined by a |
| Status  | Bits          | Affected by      | none                                                    |        |              |           |                       |             |
|         |               | Affects          | TCx                                                     |        |              |           |                       |             |
| Repeat  | :             | This instruction | n can be repeat                                         | ed.    |              |           |                       |             |

| Syntax               | Description                                                                    |
|----------------------|--------------------------------------------------------------------------------|
| TC1 = bit(*AR3, #12) | The bit at the position defined by an unsigned 4-bit value (12) in the content |
|                      | addressed by AR3 is tested and the tested bit is copied into TC1.              |

## **BTSTCLR**

Test and Clear Memory Bit

## **Syntax Characteristics**

| No.                                                                                                                                                                                                                       | Syntax         |                              |                         |           | Parallel<br>Enable Bit | Size  | Cycles    | Pipeline |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|------------------------------|-------------------------|-----------|------------------------|-------|-----------|----------|
| [1]                                                                                                                                                                                                                       | TC1 = bit(Smer | m, k4 <b>), bit(</b> Smem, l | <4 <b>)</b> = <b>#0</b> |           | No                     | 3     | 1         | Х        |
| [2]                                                                                                                                                                                                                       | TC2 = bit(Smer | m, k4 <b>), bit(</b> Smem, ł | <4 <b>)</b> = <b>#0</b> |           | No                     | 3     | 1         | Х        |
| Opcode                                                                                                                                                                                                                    | e              | TC1                          | 1                       | 1110      | 0011 AA                | AA AZ | AAI   kkk | k 010x   |
|                                                                                                                                                                                                                           |                | TC2                          | 1                       | 1110      | 0011 AA                | AA AA | AAI kkk   | k 011x   |
| Operan                                                                                                                                                                                                                    | ıds            | k4, Smem, TCx                |                         |           |                        |       |           |          |
| <b>Description</b> This instruction performs a bit manipulation in the A-unit ALU tests a single bit, as defined by a 4-bit immediate value, k (Smem) location. The tested bit is copied into status bit TCx a 0 in Smem. |                |                              |                         | e, k4, of | a memory               |       |           |          |
| Status                                                                                                                                                                                                                    | Bits           | Affected by                  | none                    |           |                        |       |           |          |
|                                                                                                                                                                                                                           |                | Affects                      | TCx                     |           |                        |       |           |          |
| Repeat                                                                                                                                                                                                                    |                | This instruction             | can be repeated         | l.        |                        |       |           |          |
| See Als                                                                                                                                                                                                                   | 50             | See the following            | ng other related in     | nstruc    | ctions:                |       |           |          |
|                                                                                                                                                                                                                           |                | ☐ Clear Memo                 | ory Bit                 |           |                        |       |           |          |
|                                                                                                                                                                                                                           |                | Complement                   | nt Memory Bit           |           |                        |       |           |          |
|                                                                                                                                                                                                                           |                | ☐ Set Memory                 | / Bit                   |           |                        |       |           |          |
|                                                                                                                                                                                                                           |                | ☐ Test and Co                | mplement Mem            | ory Bi    | t                      |       |           |          |
|                                                                                                                                                                                                                           |                | ☐ Test and Se                | et Memory Bit           |           |                        |       |           |          |
|                                                                                                                                                                                                                           |                | ☐ Test Memor                 | y Bit                   |           |                        |       |           |          |

| Syntax                                    | Description                                                                                                                                                                                                              |
|-------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TC1 = bit(*AR3, #12), bit(*AR3, #12) = #0 | The bit at the position defined by the unsigned 4-bit value (12) in the content addressed by AR3 is tested and the tested bit is copied into TC1. The selected bit (12) in the content addressed by AR3 is cleared to 0. |

## **BTSTNOT**

## Test and Complement Memory Bit

## **Syntax Characteristics**

| No.     | Syntax         |                                                                                                                   |           | Parallel<br>Enable Bit | Size   | Cycles    | Pipeline |
|---------|----------------|-------------------------------------------------------------------------------------------------------------------|-----------|------------------------|--------|-----------|----------|
| [1]     | TC1 = bit(Smen | n, k4 <b>), cbit(</b> Smem, k4 <b>)</b>                                                                           |           | No                     | 3      | 1         | Х        |
| [2]     | TC2 = bit(Smen | n, k4 <b>), cbit(</b> Smem, k4 <b>)</b>                                                                           |           | No                     | 3      | 1         | Х        |
| Opcode  | e              | TC1                                                                                                               | 1110      | 0011 AA                | AA AA  | AAI   kkk | k 100x   |
|         |                | TC2                                                                                                               | 1110      | 0011 AA                | AA AA  | AAI kkk   | k 101x   |
| Operan  | ds             | k4, Smem, TCx                                                                                                     |           |                        |        |           |          |
| Descrip | otion          | This instruction performs a bit tests a single bit, as defined (Smem) location and the test complemented in Smem. | by a 4-   | bit immediat           | e valu | e, k4, of | a memory |
| Status  | Bits           | Affected by none                                                                                                  |           |                        |        |           |          |
|         |                | Affects TCx                                                                                                       |           |                        |        |           |          |
| Repeat  |                | This instruction can be repeat                                                                                    | ed.       |                        |        |           |          |
| See Als | 60             | See the following other related                                                                                   | d instruc | tions:                 |        |           |          |
|         |                | ☐ Clear Memory Bit                                                                                                |           |                        |        |           |          |
|         |                | ☐ Complement Memory Bit                                                                                           |           |                        |        |           |          |
|         |                | ☐ Set Memory Bit                                                                                                  |           |                        |        |           |          |
|         |                | ☐ Test and Clear Memory B                                                                                         | it        |                        |        |           |          |
|         |                | ☐ Test and Set Memory Bit                                                                                         |           |                        |        |           |          |
|         |                | ☐ Test Memory Bit                                                                                                 |           |                        |        |           |          |

| Syntax | Description                                                                                                                                                                                                              |
|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|        | The bit at the position defined by the unsigned 4-bit value (12) in the content addressed by AR0 is tested and the tested bit is copied into TC1. The selected bit (12) in the content addressed by AR0 is complemented. |

| Before |      | After |      |
|--------|------|-------|------|
| *ARO   | 0040 | *ARO  | 1040 |
| TC1    | 0    | TC1   | 0    |

## **BTSTSET**

Test and Set Memory Bit

## **Syntax Characteristics**

| No.     | Syntax        |                           |                                                            |           | Parallel<br>Enable Bit | Size    | Cycles     | Pipeline |
|---------|---------------|---------------------------|------------------------------------------------------------|-----------|------------------------|---------|------------|----------|
| [1]     | TC1 = bit(Sme | m, k4 <b>), bit(</b> Smer | n, k4 <b>) = #1</b>                                        |           | No                     | 3       | 1          | Х        |
| [2]     | TC2 = bit(Sme | m, k4 <b>), bit(</b> Smer | n, k4 <b>)</b> = <b>#1</b>                                 |           | No                     | 3       | 1          | Х        |
| Opcod   | e             | TC1                       |                                                            | 1110      | 0011 AA                | AA A    | AAI   kkk  | k 000x   |
|         |               | TC2                       |                                                            | 1110      | 0011 AA                | AA A    | AAI   kkk  | k 001x   |
| Operar  | nds           | k4, Smem, T               | Сх                                                         |           |                        |         |            |          |
| Descri  | ption         | tests a single            | on performs a bit<br>bit, as defined<br>tion. The tested b | by a 4-   | bit immedia            | te valu | ie, k4, of | a memory |
| Status  | Bits          | Affected by               | none                                                       |           |                        |         |            |          |
|         |               | Affects                   | TCx                                                        |           |                        |         |            |          |
| Repeat  | t             | This instruction          | on can be repeat                                           | ed.       |                        |         |            |          |
| See Als | so            | See the follow            | wing other related                                         | d instruc | ctions:                |         |            |          |
|         |               | ☐ Clear Me                | mory Bit                                                   |           |                        |         |            |          |
|         |               | ☐ Complen                 | nent Memory Bit                                            |           |                        |         |            |          |
|         |               | ☐ Set Mem                 | ory Bit                                                    |           |                        |         |            |          |
|         |               | ☐ Test and                | Clear Memory B                                             | it        |                        |         |            |          |
|         |               | ☐ Test and                | Complement Me                                              | mory B    | it                     |         |            |          |

#### **Example**

| Syntax                                    | Description                                                                                                                                                                                                          |
|-------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TC1 = bit(*AR3, #12), bit(*AR3, #12) = #1 | The bit at the position defined by the unsigned 4-bit value (12) in the content addressed by AR3 is tested and the tested bit is copied into TC1. The selected bit (12) in the content addressed by AR3 is set to 1. |

☐ Test Memory Bit

# **Chapter 6**

# **Instruction Opcodes in Sequential Order**

This chapter provides the opcode in sequential order for each TMS320C55x $^{\text{TM}}$  DSP instruction syntax.

| Topi | c Page                                                |
|------|-------------------------------------------------------|
| 6.1  | Instruction Set Opcodes 6-2                           |
| 6.2  | Instruction Set Opcode Symbols and Abbreviations 6-19 |

## 6.1 Instruction Set Opcodes

Table 6–1 lists the opcodes of the instruction set. See Table 6–2 (page 6-19) for a list of the symbols and abbreviations used in the instruction set opcode. See Table 1–1 (page 1-2) and Table 1–2 (page 1-6) for a list of the terms, symbols, and abbreviations used in the algebraic syntax.

Table 6-1. Instruction Set Opcodes

| Opcode                     | Algebraic syntax                      |
|----------------------------|---------------------------------------|
| 0000000E xCCCCCCC kkkkkkkk | while (cond && (RPTC < k8)) repeat    |
| 0000001E xCCCCCCC xxxxxxxx | if (cond) return                      |
| 0000010E xCCCCCCC LLLLLLL  | if (cond) goto L8                     |
| 0000011E LLLLLLL LLLLLLL   | goto L16                              |
| 0000100E LLLLLLL LLLLLLL   | call L16                              |
| 0000110E kkkkkkkk kkkkkkk  | repeat(k16)                           |
| 0000111E 11111111 11111111 | blockrepeat{}                         |
| 0001000E DDSS0000 xxSHIFTW | ACy = ACy & (ACx <<< #SHIFTW)         |
| 0001000E DDSS0001 xxSHIFTW | ACy = ACy   (ACx <<< #SHIFTW)         |
| 0001000E DDSS0010 xxSHIFTW | ACy = ACy ^ (ACx <<< #SHIFTW)         |
| 0001000E DDSS0011 xxSHIFTW | ACy = ACy + (ACx << #SHIFTW)          |
| 0001000E DDSS0100 xxSHIFTW | $ACy = ACy - (ACx \ll \#SHIFTW)$      |
| 0001000E DDSS0101 xxSHIFTW | ACy = ACx << #SHIFTW                  |
| 0001000E DDSS0110 xxSHIFTW | ACy = ACx < <c #shiftw<="" td=""></c> |
| 0001000E DDSS0111 xxSHIFTW | ACy = ACx <<< #SHIFTW                 |
| 0001000E xxSS1000 xxddxxxx | Tx = exp(ACx)                         |
| 0001000E DDSS1001 xxddxxxx | ACy = mant(ACx), Tx = -exp(ACx)       |
| 0001000E xxSS1010 SSddxxxt | Tx = count(ACx, ACy, TCx)             |
| 0001000E DDSS1100 SSDDnnnn | max_diff(ACx,ACy,ACz,ACw)             |
| 0001000E DDSS1101 SSDDxxxr | max_diff_dbl(ACx,ACy,ACz,ACw,TRNx)    |
| 0001000E DDSS1110 SSDDxxxx | min_diff(ACx,ACy,ACz,ACw)             |
| 0001000E DDSS1111 SSDDxxxr | min_diff_dbl(ACx,ACy,ACz,ACw,TRNx)    |
| 0001001E FSSScc00 FDDDxuxt | TCx = uns(src RELOP dst)              |
| 0001001E FSSScc01 FDDDOutt | TCx = TCy & uns(src RELOP dst)        |
| 0001001E FSSScc01 FDDD1utt | TCx = !TCy & uns(src RELOP dst)       |
| 0001001E FSSScc10 FDDDOutt | TCx = TCy   uns(src RELOP dst)        |
| 0001001E FSSScc10 FDDD1utt | TCx = !TCy   uns(src RELOP dst)       |
| 0001001E FSSSxx11 FDDD0xvv | dst = BitOut \\ src \\ BitIn          |

Table 6-1. Instruction Set Opcodes (Continued)

| Opcode                                       |               | Algebraic syntax             |
|----------------------------------------------|---------------|------------------------------|
| 0001001E FSSSxx11 FDDD1                      | 1xvv d        | lst = Bitln // src // BitOut |
| 0001010E FSSSxxxx FDDD0                      | 0000 n        | nar(TAy + TAx)               |
| 0001010E FSSSxxxx FDDD0                      | 0001 n        | nar(TAy = TAx)               |
| 0001010E FSSSxxxx FDDD0                      | 0010 n        | nar(TAy – TAx)               |
| 0001010E PPPPPPPP FDDD0                      | 0100 n        | nar(TAx + P8)                |
| 0001010E PPPPPPPP FDDD0                      | 0101 <b>n</b> | nar(TAx = P8)                |
| 0001010E PPPPPPPP FDDD0                      | 0110 <b>n</b> | nar(TAx – P8)                |
| 0001010E FSSSxxxx FDDD                       | 1000 <b>n</b> | nar(TAy + TAx)               |
| 0001010E FSSSxxxx FDDD                       | 1001 <b>n</b> | nar(TAy = TAx)               |
| 0001010E FSSSxxxx FDDD                       | 1010 <b>n</b> | nar(TAy – TAx)               |
| 0001010E PPPPPPPP FDDD3                      | 1100 <b>n</b> | nar(TAx + P8)                |
| 0001010E PPPPPPPP FDDD3                      | 1101 <b>n</b> | nar(TAx = P8)                |
| 0001010E PPPPPPPP FDDD3                      | 1110 <b>n</b> | nar(TAx – P8)                |
| 0001010E XACS0001 XACD0<br>(Note: for DAG_X) | 0000 <b>n</b> | nar(XACdst + XACsrc)         |
| 0001010E XACS0001 XACD0<br>(Note: for DAG_X) | 0001 <b>n</b> | nar(XACdst = XACsrc)         |
| 0001010E XACS0001 XACD0<br>(Note: for DAG_X) | 0010 <b>n</b> | nar(XACdst - XACsrc)         |
| 0001010E XACS0001 XACD1<br>(Note: for DAG_Y) | 1000 <b>n</b> | nar(XACdst + XACsrc)         |
| 0001010E XACS0001 XACD1<br>(Note: for DAG_Y) | 1001 <b>n</b> | nar(XACdst = XACsrc)         |
| 0001010E XACS0001 XACD<br>(Note: for DAG_Y)  | 1010 <b>n</b> | nar(XACdst - XACsrc)         |
| 0001011E xxxxxkkk kkkk                       | 0000          | DPH = k7                     |
| 0001011E xxxkkkkk kkkk                       | 0011 F        | PDP = k9                     |
| 0001011E kkkkkkkk kkkk                       | 0100 B        | 3K03 = k12                   |
| 0001011E kkkkkkkk kkkk                       | 0101 E        | 3K47 = k12                   |
| 0001011E kkkkkkkk kkkk                       | 0110 E        | 3KC = k12                    |
| 0001011E kkkkkkkk kkkk                       | 1000 C        | CSR = k12                    |
| 0001011E kkkkkkkk kkkk                       | 1001 E        | BRC0 = k12                   |
| 0001011E kkkkkkkk kkkk                       | 1010 E        | BRC1 = k12                   |
| 0001100E kkkkkkk FDDDI                       | FSSS d        | dst = src & k8               |
| 0001101E kkkkkkkk FDDDI                      | FSSS d        | dst = src   k8               |

Table 6-1. Instruction Set Opcodes (Continued)

| Opcode                                                | Algebraic syntax                                          |
|-------------------------------------------------------|-----------------------------------------------------------|
| 0001110E kkkkkkk FDDDFSSS                             | dst = src ^ k8                                            |
| 0001111E KKKKKKKK SSDDxx0%                            | $ACy = \frac{\text{rnd}(ACx * K8)}{\text{rnd}(ACx * K8)}$ |
| 0001111E KKKKKKKK SSDDss1%                            | $ACy = \frac{rnd(ACx + (Tx * K8))}{}$                     |
| 0010000E                                              | nop                                                       |
| 0010001E FSSSFDDD                                     | dst = src                                                 |
| 0010010E FSSSFDDD                                     | dst = dst + src                                           |
| 0010011E FSSSFDDD                                     | dst = dst - src                                           |
| 0010100E FSSSFDDD                                     | dst = dst & src                                           |
| 0010101E FSSSFDDD                                     | dst = dst   src                                           |
| 0010110E FSSSFDDD                                     | dst = dst ^ src                                           |
| 0010111E FSSSFDDD                                     | dst = max(src, dst)                                       |
| 0011000E FSSSFDDD                                     | dst = min(src, dst)                                       |
| 0011001E FSSSFDDD                                     | dst =  src                                                |
| 0011010E FSSSFDDD                                     | dst = -src                                                |
| 0011011E FSSSFDDD                                     | dst = ~src                                                |
| 0011100E FSSSFDDD<br>(Note: FSSS = src1, FDDD = src2) | push(src1, src2)                                          |
| 0011101E FSSSFDDD<br>(Note: FSSS = dst1, FDDD = dst2) | dst1, dst2 = pop()                                        |
| 0011110E kkkkFDDD                                     | dst = k4                                                  |
| 0011111E kkkkFDDD                                     | dst = -k4                                                 |
| 0100000E kkkkFDDD                                     | dst = dst + k4                                            |
| 0100001E kkkkFDDD                                     | dst = dst - k4                                            |
| 01000101 11110010                                     | lock()                                                    |
| 0100010E 00SSFDDD                                     | TAx = HI(ACx)                                             |
| 0100010E 01x0FDDD                                     | dst = dst >> #1                                           |
| 0100010E 01x1FDDD                                     | dst = dst << #1                                           |
| 0100010E 1000FDDD                                     | TAx = SP                                                  |
| 0100010E 1001FDDD                                     | TAx = SSP                                                 |
| 0100010E 1010FDDD                                     | TAx = CDP                                                 |
| 0100010E 1100FDDD                                     | TAx = BRC0                                                |
| 0100010E 1101FDDD                                     | TAx = BRC1                                                |
| 0100010E 1110FDDD                                     | TAx = RPTC                                                |

Table 6-1. Instruction Set Opcodes (Continued)

| Opcode            | Algebraic syntax        |  |
|-------------------|-------------------------|--|
| 0100011E kkkk0000 | bit(ST0, k4) = #0       |  |
| 0100011E kkkk0001 | bit(ST0, k4) = #1       |  |
| 0100011E kkkk0010 | bit(ST1, k4) = #0       |  |
| 0100011E kkkk0011 | bit(ST1, k4) = #1       |  |
| 0100011E kkkk0100 | bit(ST2, k4) = #0       |  |
| 0100011E kkkk0101 | bit(ST2, k4) = #1       |  |
| 0100011E kkkk0110 | bit(ST3, k4) = #0       |  |
| 0100011E kkkk0111 | bit(ST3, k4) = #1       |  |
| 0100100E xxxxx000 | repeat(CSR)             |  |
| 0100100E FSSSx001 | repeat(CSR), CSR += TAx |  |
| 0100100E kkkkx010 | repeat(CSR), CSR += k4  |  |
| 0100100E kkkkx011 | repeat(CSR), CSR -= k4  |  |
| 0100100E xxxxx100 | return                  |  |
| 01001000 xxxxx100 | return_int              |  |
| 0100101E OLLLLLL  | goto L7                 |  |
| 0100101E 11111111 | localrepeat{}           |  |
| 0100110E kkkkkkkk | repeat(k8)              |  |
| 0100111E KKKKKKKK | SP = SP + K8            |  |
| 0101000E FDDDx000 | dst = dst <<< #1        |  |
| 0101000E FDDDx001 | dst = dst >>> #1        |  |
| 0101000E FDDDx010 | dst = pop()             |  |
| 0101000E xxDDx011 | ACx = dbl(pop())        |  |
| 0101000E FSSSx110 | push(src)               |  |
| 0101000E xxSSx111 | dbl(push(ACx))          |  |
| 0101000E XDDD0100 | xdst = popboth()        |  |
| 0101000E XSSS0101 | pshboth(xsrc)           |  |
| 0101001E FSSS00DD | HI(ACx) = TAx           |  |
| 0101001E FSSS1000 | SP = TAx                |  |
| 0101001E FSSS1001 | SSP = TAx               |  |
| 0101001E FSSS1010 | CDP = TAx               |  |
| 0101001E FSSS1100 | CSR = TAx               |  |
| 0101001E FSSS1101 | BRC1 = TAx              |  |
| 0101001E FSSS1110 | BRC0 = TAx              |  |

Table 6-1. Instruction Set Opcodes (Continued)

| Opcode                                          | Algebraic syntax                                              |
|-------------------------------------------------|---------------------------------------------------------------|
| 0101010E DDSS000%                               | $ACy = \frac{rnd(ACy +  ACx )}{rnd(ACy +  ACx )}$             |
| 0101010E DDSS001%                               | $ACy = \frac{rnd(ACy + (ACx * ACx))}{rnd(ACy + (ACx * ACx))}$ |
| 0101010E DDSS010%                               | $ACy = \frac{rnd(ACy - (ACx * ACx))}{rnd(ACy - (ACx * ACx))}$ |
| 0101010E DDSS011%                               | $ACy = \frac{rnd}{ACy} * ACx)$                                |
| 0101010E DDSS100%                               | $ACy = \frac{rnd}{ACx} * ACx$                                 |
| 0101010E DDSS101%                               | $ACy = \frac{rnd(ACx)}{rnd(ACx)}$                             |
| 0101010E DDSS110%                               | ACy = saturate(rnd(ACx))                                      |
| 0101011E DDSSss0%                               | $ACy = \frac{rnd(ACy + (ACx * Tx))}{rnd(ACy + (ACx * Tx))}$   |
| 0101011E DDSSss1%                               | $ACy = \frac{rnd(ACy - (ACx * Tx))}{rnd(ACy - (ACx * Tx))}$   |
| 0101100E DDSSss0%                               | $ACy = \frac{rnd(ACx * Tx)}{rnd(ACx * Tx)}$                   |
| 0101100E DDSSss1%                               | $ACy = \frac{rnd}{(ACy * Tx) + ACx)}$                         |
| 0101101E DDSSss00                               | ACy = ACy + (ACx << Tx)                                       |
| 0101101E DDSSss01                               | ACy = ACy - (ACx << Tx)                                       |
| 0101101E DDxxxx1t                               | ACx = sftc(ACx,TCx)                                           |
| 0101110E DDSSss00                               | $ACy = ACx \ll Tx$                                            |
| 0101110E DDSSss01                               | $ACy = ACx \ll Tx$                                            |
| 0101110E DDSSss10                               | $ACy = ACx \ll Tx$                                            |
| 0101111E 00kkkkkk                               | swap()                                                        |
| 01100111 lcccccc                                | if (cond) goto I4                                             |
| 01101000 xCCCCCCC PPPPPPPP PPPPPPPPPPPPPPPPPP   | P if (cond) goto P24                                          |
| 01101001 xCCCCCCC PPPPPPPP PPPPPPPPPPPPPPPPPPPP | P if (cond) call P24                                          |
| 01101010 PPPPPPPP PPPPPPPP PPPPPPPP             | P goto P24                                                    |
| 01101100 PPPPPPPP PPPPPPPP PPPPPPP              | P call P24                                                    |
| 01101101 xCCCCCCC LLLLLLL LLLLLLL               | L if (cond) goto L16                                          |
| 01101110 xCCCCCCC LLLLLLL LLLLLLL               | L if (cond) call L16                                          |
| 01101111 FSSSccxu KKKKKKK LLLLLLL               | L compare (uns(src RELOP K8)) goto L8                         |
| 01110000 KKKKKKKK KKKKKKKK SSDDSHF              | T $ACy = ACx + (K16 \ll \#SHFT)$                              |
| 01110001 KKKKKKKK KKKKKKKK SSDDSHF              | T $ACy = ACx - (K16 \ll \#SHFT)$                              |
| 01110010 kkkkkkkk kkkkkkkk SSDDSHF              | T ACy = ACx & $(k16 <<< \#SHFT)$                              |
| 01110011 kkkkkkkk kkkkkkkk SSDDSHF              | T $ACy = ACx \mid (k16 <<< \#SHFT)$                           |
| 01110100 kkkkkkkk kkkkkkkk SSDDSHF              | T $ACy = ACx ^ (k16 <<< #SHFT)$                               |

Table 6–1. Instruction Set Opcodes (Continued)

| Opcode                    |            | Algebraic syntax                                            |
|---------------------------|------------|-------------------------------------------------------------|
| 01110101 KKKKKKKK KKKKKKK | K xxDDSHFT | ACx = K16 << #SHFT                                          |
| 01110110 kkkkkkkk kkkkkk  | k FDDD00SS | dst = field_extract(ACx,k16)                                |
| 01110110 kkkkkkkk kkkkkk  | k FDDD01SS | dst = field_expand(ACx,k16)                                 |
| 01110110 KKKKKKKK KKKKKK  | K FDDD10xx | dst = K16                                                   |
| 01110111 DDDDDDDD DDDDDDD | D FDDDxxxx | mar(TAx = D16)                                              |
| 01111000 kkkkkkkk kkkkkk  | k xxx0000x | DP = k16                                                    |
| 01111000 kkkkkkkk kkkkkk  | k xxx0001x | SSP = k16                                                   |
| 01111000 kkkkkkkk kkkkkk  | k xxx0010x | CDP = k16                                                   |
| 01111000 kkkkkkkk kkkkkk  | k xxx0011x | BSA01 = k16                                                 |
| 01111000 kkkkkkkk kkkkkk  | k xxx0100x | BSA23 = k16                                                 |
| 01111000 kkkkkkkk kkkkkk  | k xxx0101x | BSA45 = k16                                                 |
| 01111000 kkkkkkkk kkkkkk  | k xxx0110x | BSA67 = k16                                                 |
| 01111000 kkkkkkkk kkkkkkk | k xxx0111x | BSAC = k16                                                  |
| 01111000 kkkkkkkk kkkkkk  | k xxx1000x | SP = k16                                                    |
| 01111001 KKKKKKKK KKKKKK  | K SSDDxx0% | ACy = rnd(ACx * K16)                                        |
| 01111001 KKKKKKKK KKKKKK  | K SSDDss1% | $ACy = \frac{rnd(ACx + (Tx * K16))}{rnd(ACx + (Tx * K16))}$ |
| 01111010 KKKKKKKK KKKKKK  | K SSDD000x | ACy = ACx + (K16 << #16)                                    |
| 01111010 KKKKKKKK KKKKKKK | K SSDD001x | ACy = ACx - (K16 << #16)                                    |
| 01111010 kkkkkkkk kkkkkk  | k SSDD010x | ACy = ACx & (k16 <<< #16)                                   |
| 01111010 kkkkkkkk kkkkkkk | k SSDD011x | $ACy = ACx \mid (k16 <<< #16)$                              |
| 01111010 kkkkkkkk kkkkkk  | k SSDD100x | $ACy = ACx ^ (k16 <<< #16)$                                 |
| 01111010 KKKKKKKK KKKKKKK | K xxDD101x | ACx = K16 << #16                                            |
| 01111010 xxxxxxxx xxxxxxx | x xxxx110x | idle                                                        |
| 01111011 KKKKKKKK KKKKKKK | K FDDDFSSS | dst = src + K16                                             |
| 01111100 KKKKKKKK KKKKKK  | K FDDDFSSS | dst = src - K16                                             |
| 01111101 kkkkkkkk kkkkkk  | k FDDDFSSS | dst = src & k16                                             |
| 01111110 kkkkkkkk kkkkkkk | k FDDDFSSS | dst = src   k16                                             |
| 01111111 kkkkkkkk kkkkkkk | k FDDDFSSS | $dst = src \wedge k16$                                      |
| 10000000 XXXMMMYY YMMM00x | x          | dbl(Ymem) = dbl(Xmem)                                       |
| 10000000 XXXMMMYY YMMM01x | x          | Ymem = Xmem                                                 |
| 10000000 XXXMMMYY YMMM10S | S          | Xmem = LO(ACx),<br>Ymem = HI(ACx)                           |
| 10000001 XXXMMMYY YMMM00D | D          | ACx = (Xmem << #16) + (Ymem << #16)                         |

Table 6-1. Instruction Set Opcodes (Continued)

|          | Орс      | ode              | Algebraic syntax                                                                                                                   |
|----------|----------|------------------|------------------------------------------------------------------------------------------------------------------------------------|
| 10000001 | XXXMMMYY | YMMM01DD         | ACx = (Xmem << #16) - (Ymem << #16)                                                                                                |
| 10000001 | XXXMMMYY | YMMM10DD         | LO(ACx) = Xmem,<br>HI(ACx) = Ymem                                                                                                  |
| 10000010 | XXXMMMYY | YMMM00mm uuDDDDg | ACx = M40(rnd(uns(Xmem) * uns(coef(Cmem)))),<br>ACy = M40(rnd(uns(Ymem) * uns(coef(Cmem))))                                        |
| 10000010 | XXXMMMYY | YMMM01mm uuDDDDg | ACx = M40(rnd(ACx + (uns(Xmem) * uns(coef(Cmem))))),<br>ACy = M40(rnd(uns(Ymem) * uns(coef(Cmem))))                                |
| 10000010 | XXXMMMYY | YMMM10mm uuDDDDg | ACx = M40(rnd(ACx - (uns(Xmem) * uns(coef(Cmem))))),<br>ACy = M40(rnd(uns(Ymem) * uns(coef(Cmem))))                                |
| 10000010 | XXXMMMYY | YMMM11mm uuxxDDg | <pre>% mar(Xmem), ACx = M40(rnd(uns(Ymem) * uns(coef(Cmem))))</pre>                                                                |
| 10000011 | XXXMMMYY | YMMM00mm uuDDDDg | ACx = M40(rnd(ACx + (uns(Xmem) * uns(coef(Cmem))))),<br>ACy = M40(rnd(ACy + (uns(Ymem) * uns(coef(Cmem)))))                        |
| 10000011 | XXXMMMYY | YMMM01mm uuDDDDg | ACx = $M40(rnd(ACx - (uns(Xmem) * uns(coef(Cmem)))))$ ,<br>ACy = $M40(rnd(ACy + (uns(Ymem) * uns(coef(Cmem)))))$                   |
| 10000011 | XXXMMMYY | YMMM10mm uuDDDDg | % ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(coef(Cmem))))),<br>ACy = M40(rnd(ACy + (uns(Ymem) * uns(coef(Cmem)))))             |
| 10000011 | XXXMMMYY | YMMM11mm uuxxDDg | <pre>% mar(Xmem), ACx = M40(rnd(ACx + (uns(Ymem) * uns(coef(Cmem)))))</pre>                                                        |
| 10000100 | XXXMMMYY | YMMM00mm uuDDDDg | <pre>ACx = M40(rnd(ACx - (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd((ACy &gt;&gt; #16) + (uns(Ymem) * uns(coef(Cmem)))))</pre> |
| 10000100 | XXXMMMYY | YMMM01mm uuxxDDg | <pre>% mar(Xmem), ACx = M40(rnd((ACx &gt;&gt; #16) + (uns(Ymem) * uns(coef(Cmem)))))</pre>                                         |
| 10000100 | XXXMMMYY | YMMM10mm uuDDDDg | <pre>% ACx = M40(rnd(uns(Xmem) * uns(coef(Cmem)))), ACy = M40(rnd((ACy &gt;&gt; #16) + (uns(Ymem) * uns(coef(Cmem)))))</pre>       |
| 10000100 | XXXMMMYY | YMMM11mm uuDDDDg | % ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(coef(Cmem))))),<br>ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(coef(Cmem)))))    |
| 10000101 | XXXMMMYY | YMMM00mm uuxxDDg | <pre>% mar(Xmem), ACx = M40(rnd(ACx - (uns(Ymem) * uns(coef(Cmem)))))</pre>                                                        |
| 10000101 | XXXMMMYY | YMMM01mm uuDDDDg | ACx = M40(rnd(ACx - (uns(Xmem) * uns(coef(Cmem))))),<br>ACy = M40(rnd(ACy - (uns(Ymem) * uns(coef(Cmem)))))                        |
| 10000101 | XXXMMMYY | YMMM10mm xxxxxx  | x mar(Xmem) ,mar(Ymem) ,mar(coef(Cmem))                                                                                            |

Table 6-1. Instruction Set Opcodes (Continued)

| Opcode   |          |          |           | Algebraic syntax                                                                                             |
|----------|----------|----------|-----------|--------------------------------------------------------------------------------------------------------------|
| 10000101 | XXXMMMYY | YMMM11mm | DDx0DDU%  | firs(Xmem, Ymem, coef(Cmem), ACx, ACy)                                                                       |
| 10000101 | XXXMMMYY | YMMM11mm | DDx1DDU%  | firsn(Xmem, Ymem, coef(Cmem), ACx, ACy)                                                                      |
| 10000110 | XXXMMMYY | YMMMxxDD | 000guuU%  | ACx = M40(rnd(uns(Xmem) * uns(Ymem)))<br>[,T3 = Xmem]                                                        |
| 10000110 | XXXMMMYY | YMMMSSDD | 001guuU%  | ACy = M40(rnd(ACx + (uns(Xmem) * uns(Ymem))))<br>[,T3 = Xmem]                                                |
| 10000110 | XXXMMMYY | YMMMSSDD | 010guuU%  | ACy = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(Ymem)))) [,T3 = Xmem]                                          |
| 10000110 | XXXMMMYY | YMMMSSDD | 011guuU%  | ACy = M40(rnd(ACx - (uns(Xmem) * uns(Ymem))))<br>[,T3 = Xmem]                                                |
| 10000110 | XXXMMMYY | YMMMDDDD | 100xssU%  | $ACx = \frac{\text{rnd}(ACx - (Tx * Xmem))}{ACy},$ $ACy = \frac{\text{Ymem} << #16 [,T3 = Xmem]}{ACy}$       |
| 10000110 | XXXMMMYY | YMMMDDDD | 101xssU%  | ACx = rnd(ACx + (Tx * Xmem)),<br>ACy = Ymem << #16 [,T3 = Xmem]                                              |
| 10000110 | XXXMMMYY | YMMMDDDD | 110xxxx%  | Ims(Xmem, Ymem, ACx, ACy)                                                                                    |
| 10000110 | XXXMMMYY | YMMMDDDD | 1110xxn%  | sqdst(Xmem, Ymem, ACx, ACy)                                                                                  |
| 10000110 | XXXMMMYY | YMMMDDDD | 1111xxn%  | abdst(Xmem, Ymem, ACx, ACy)                                                                                  |
| 10000111 | XXXMMMYY | YMMMSSDD | 000xssU%  | ACy = rnd(Tx * Xmem),<br>Ymem = HI(ACx << T2) [,T3 = Xmem]                                                   |
| 10000111 | XXXMMMYY | YMMMSSDD | 001xssU%  | ACy = rnd(ACy + (Tx * Xmem)),<br>Ymem = HI(ACx << T2) [,T3 = Xmem]                                           |
| 10000111 | XXXMMMYY | YMMMSSDD | 010xssU%  | ACy = rnd(ACy - (Tx * Xmem)),<br>Ymem = HI(ACx << T2) [,T3 = Xmem]                                           |
| 10000111 | XXXMMMYY | YMMMSSDD | 01100001  | Imsf(Xmem, Ymem, ACx, ACy)                                                                                   |
| 10000111 | XXXMMMYY | YMMMSSDD | 100xxxxx  | ACy = ACx + (Xmem << #16),<br>Ymem = HI(ACy << T2)                                                           |
| 10000111 | XXXMMMYY | YMMMSSDD | 101xxxxx  | $ACy = (Xmem \ll #16) - ACx,$<br>$Ymem = HI(ACy \ll T2)$                                                     |
| 10000111 | XXXMMMYY | YMMMSSDD | 110xxxxx  | ACy = Xmem << #16,<br>Ymem = HI(ACx << T2)                                                                   |
| 10010000 | XSSSXDDD |          |           | xdst = xsrc                                                                                                  |
| 10010001 | xxxxxxSS |          |           | goto ACx                                                                                                     |
| 10010010 | XXXMMMYY | YMMM00mm | uuDDDDDg% | ACy = M40(rnd(uns(Ymem) * uns(HI(coef(Cmem))))),<br>ACx = M40(rnd(uns(Xmem) * uns(LO(coef(Cmem)))))          |
| 10010010 | XXXMMMYY | YMMM01mm | uuDDDDg%  | ACy = M40(rnd(uns(Ymem) * uns(HI(coef(Cmem))))),<br>ACx = M40(rnd(ACx + uns(Xmem) *<br>uns(LO(coef(Cmem))))) |

Table 6-1. Instruction Set Opcodes (Continued)

|          | Орс      | ode          |        | Algebraic syntax                                                                                                                   |
|----------|----------|--------------|--------|------------------------------------------------------------------------------------------------------------------------------------|
| 10010010 | XXXMMMYY | YMMM10mm uuI | DDDDg% | ACy = M40(rnd(uns(Ymem) * uns(HI(coef(Cmem))))),<br>ACx = M40(rnd(ACx - uns(Xmem) *<br>uns(LO(coef(Cmem)))))                       |
| 10010010 | xxxxxxSS |              |        | call ACx                                                                                                                           |
| 10010011 | XXXMMMYY | YMMM00mm uuI | DDDDg% | ACy = M40(rnd(ACy + uns(Ymem) * uns(HI(coef(Cmem))))),<br>ACx = M40(rnd(ACx + uns(Xmem) * uns(LO(coef(Cmem)))))                    |
| 10010011 | XXXMMMYY | YMMM01mm uuI | DDDDg% | ACy = M40(rnd(ACy + uns(Ymem) * uns(HI(coef(Cmem))))),<br>ACx = M40(rnd(ACx - uns(Xmem) * uns(LO(coef(Cmem)))))                    |
| 10010011 | XXXMMMYY | YMMM10mm uuI | DDDDg% | ACy = M40(rnd(ACy + (uns(Ymem) * uns(HI(coef(Cmem)))))),<br>ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(LO(coef(Cmem))))))       |
| 10010011 | XXXMMMYY | YMMM11mm uuI | DDDDg% | ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(LO(coef(Cmem)))))) |
| 10010100 | XXXMMMYY | YMMM00mm uuI | DDDDg% | ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(HI(coef(Cmem)))))),<br>ACx = M40(rnd(ACx - (uns(Xmem) * uns(LO(coef(Cmem))))))       |
| 10010100 | XXXMMMYY | YMMM10mm uuI | DDDDg% | ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(uns(Xmem) * uns(LO(coef(Cmem)))))                  |
| 10010100 | xxxxxxx  |              |        | reset                                                                                                                              |
| 10010101 | 0xxkkkkk |              |        | intr(k5)                                                                                                                           |
| 10010101 | 1xxkkkkk |              |        | trap(k5)                                                                                                                           |
| 10010101 | XXXMMMYY | YMMM01mm uuI | DDDDg% | ACy = M40(rnd(ACy - (uns(Ymem) * uns(HI(coef(Cmem)))))),<br>ACx = M40(rnd(ACx - (uns(Xmem) * uns(LO(coef(Cmem))))))                |
| 10010110 | 0CCCCCC  |              |        | if (cond) execute(AD_unit)                                                                                                         |
| 10010110 | 1CCCCCCC |              |        | if (cond) execute(D_unit)                                                                                                          |
| 10011000 |          |              |        | mmap()                                                                                                                             |
| 10011001 |          |              |        | readport()                                                                                                                         |
| 10011010 |          |              |        | writeport()                                                                                                                        |

Table 6–1. Instruction Set Opcodes (Continued)

| Opcode                     | Algebraic syntax                                                                  |
|----------------------------|-----------------------------------------------------------------------------------|
| 10011100                   | linear()                                                                          |
| 10011101                   | circular()                                                                        |
| 10011110 OCCCCCC           | if (cond) execute(AD_unit)                                                        |
| 10011110 1CCCCCCC          | if (cond) execute(D_unit)                                                         |
| 10011111 0CCCCCCC          | if (cond) execute(AD_unit)                                                        |
| 10011111 1CCCCCCC          | if (cond) execute(D_unit)                                                         |
| 1010FDDD AAAAAAAI          | dst = Smem                                                                        |
| 101100DD AAAAAAAI          | ACx = Smem << #16                                                                 |
| 10110100 AAAAAAAI          | mar(Smem)                                                                         |
| 10110101 AAAAAAAI          | push(Smem)                                                                        |
| 10110110 AAAAAAAI          | delay(Smem)                                                                       |
| 10110111 AAAAAAAI          | push(dbl(Lmem))                                                                   |
| 10111000 AAAAAAI           | dbl(Lmem) = pop()                                                                 |
| 10111011 AAAAAAI           | Smem = pop()                                                                      |
| 101111SS AAAAAAAI          | Smem = HI(ACx)                                                                    |
| 1100FSSS AAAAAAAI          | Smem = src                                                                        |
| 11010000 AAAAAAAI 0%DD01mm | ACx = rnd(Smem * uns(coef(Cmem)))                                                 |
| 11010000 AAAAAAAI 0%DD10mm | ACx = rnd(ACx + (Smem * uns(coef(Cmem))))                                         |
| 11010000 AAAAAAAI 0%DD11mm | ACx = rnd(ACx - (Smem * uns(coef(Cmem))))                                         |
| 11010000 AAAAAAAI U%DDxxmm | $ACx = \frac{\text{rnd}(ACx + (Smem * coef(Cmem)))}{\text{(Smem)}}$ , delay(Smem) |
| 11010001 AAAAAAAI U%DD00mm | ACx = rnd(Smem * coef(Cmem)) [,T3 = Smem]                                         |
| 11010001 AAAAAAAI U%DD01mm | ACx = rnd(ACx + (Smem * coef(Cmem))) [,T3 = Smem]                                 |
| 11010001 AAAAAAAI U%DD10mm | ACx = rnd(ACx - (Smem * coef(Cmem))) [,T3 = Smem]                                 |
| 11010010 AAAAAAAI U%DD00SS | ACy = rnd(ACy + (Smem * ACx)) [,T3 = Smem]                                        |
| 11010010 AAAAAAAI U%DD01SS | ACy = rnd(ACy - (Smem * ACx)) [,T3 = Smem]                                        |
| 11010010 AAAAAAAI U%DD10SS | ACy = rnd(ACx + (Smem * Smem)) [,T3 = Smem]                                       |
| 11010010 AAAAAAAI U%DD11SS | ACy = rnd(ACx - (Smem * Smem)) [,T3 = Smem]                                       |
| 11010011 AAAAAAAI U%DD00SS | ACy = rnd(Smem * ACx) [,T3 = Smem]                                                |
| 11010011 AAAAAAAI U%DD10xx | ACx = rnd(Smem * Smem) [,T3 = Smem]                                               |
| 11010011 AAAAAAAI U%DDulss | ACx = rnd(uns(Tx * Smem)) [,T3 = Smem]                                            |
| 11010100 AAAAAAAI U%DDssSS | ACy = rnd(ACx + (Tx * Smem)) [,T3 = Smem]                                         |
| 11010101 AAAAAAAI U%DDssSS | $ACy = \frac{rnd(ACx - (Tx * Smem)) [,T3 = Smem]}{}$                              |

Table 6-1. Instruction Set Opcodes (Continued)

| Opcode                     | Algebraic syntax                     |
|----------------------------|--------------------------------------|
| 11010110 AAAAAAAI FDDDFSSS | dst = src + Smem                     |
| 11010111 AAAAAAAI FDDDFSSS | dst = src - Smem                     |
| 11011000 AAAAAAAI FDDDFSSS | dst = Smem - src                     |
| 11011001 AAAAAAAI FDDDFSSS | dst = src & Smem                     |
| 11011010 AAAAAAAI FDDDFSSS | dst = src   Smem                     |
| 11011011 AAAAAAAI FDDDFSSS | dst = src ^ Smem                     |
| 11011100 AAAAAAAI kkkkxx00 | TC1 = bit(Smem, k4)                  |
| 11011100 AAAAAAAI kkkkxx01 | TC2 = bit(Smem, k4)                  |
| 11011100 AAAAAAAI 0000xx10 | DP = Smem                            |
| 11011100 AAAAAAAI 0001xx10 | CDP = Smem                           |
| 11011100 AAAAAAAI 0010xx10 | BSA01 = Smem                         |
| 11011100 AAAAAAI 0011xx10  | BSA23 = Smem                         |
| 11011100 AAAAAAAI 0100xx10 | BSA45 = Smem                         |
| 11011100 AAAAAAAI 0101xx10 | BSA67 = Smem                         |
| 11011100 AAAAAAI 0110xx10  | BSAC = Smem                          |
| 11011100 AAAAAAI 0111xx10  | SP = Smem                            |
| 11011100 AAAAAAI 1000xx10  | SSP = Smem                           |
| 11011100 AAAAAAI 1001xx10  | BK03 = Smem                          |
| 11011100 AAAAAAI 1010xx10  | BK47 = Smem                          |
| 11011100 AAAAAAI 1011xx10  | BKC = Smem                           |
| 11011100 AAAAAAI 1100xx10  | DPH = Smem                           |
| 11011100 AAAAAAI 1111xx10  | PDP = Smem                           |
| 11011100 AAAAAAI x000xx11  | CSR = Smem                           |
| 11011100 AAAAAAI x001xx11  | BRC0 = Smem                          |
| 11011100 AAAAAAI x010xx11  | BRC1 = Smem                          |
| 11011100 AAAAAAI x011xx11  | TRN0 = Smem                          |
| 11011100 AAAAAAI x100xx11  | TRN1 = Smem                          |
| 11011101 AAAAAAAI SSDDss00 | ACy = ACx + (Smem << Tx)             |
| 11011101 AAAAAAAI SSDDss01 | ACy = ACx - (Smem << Tx)             |
| 11011101 AAAAAAAI SSDDss10 | ACy = ads2c(Smem, ACx, Tx, TC1, TC2) |
| 11011101 AAAAAAAI x%DDss11 | $ACx = \frac{rnd}{Smem} << Tx)$      |
| 11011110 AAAAAAAI SSDD0000 | ACy = adsc(Smem, ACx, TC1)           |
| 11011110 AAAAAAAI SSDD0001 | ACy = adsc(Smem, ACx, TC2)           |

Table 6-1. Instruction Set Opcodes (Continued)

| Opcode                     | Algebraic syntax                            |
|----------------------------|---------------------------------------------|
| 11011110 AAAAAAAI SSDD0010 | ACy = adsc(Smem, ACx, TC1, TC2)             |
| 11011110 AAAAAAAI SSDD0011 | subc(Smem, ACx, ACy)                        |
| 11011110 AAAAAAAI SSDD0100 | ACy = ACx + (Smem << #16)                   |
| 11011110 AAAAAAAI SSDD0101 | ACy = ACx - (Smem << #16)                   |
| 11011110 AAAAAAAI SSDD0110 | ACy = (Smem << #16) - ACx                   |
| 11011110 AAAAAAAI ssDD1000 | HI(ACx) = Smem + Tx,<br>LO(ACx) = Smem - Tx |
| 11011110 AAAAAAAI ssDD1001 | HI(ACx) = Smem - Tx,<br>LO(ACx) = Smem + Tx |
| 11011111 AAAAAAAI FDDD000u | dst = uns(high_byte(Smem))                  |
| 11011111 AAAAAAAI FDDD001u | dst = uns(low_byte(Smem))                   |
| 11011111 AAAAAAAI xxDD010u | ACx = uns(Smem)                             |
| 11011111 AAAAAAAI SSDD100u | ACy = ACx + uns(Smem) + CARRY               |
| 11011111 AAAAAAAI SSDD101u | ACy = ACx - uns(Smem) - BORROW              |
| 11011111 AAAAAAAI SSDD110u | ACy = ACx + uns(Smem)                       |
| 11011111 AAAAAAAI SSDD111u | ACy = ACx - uns(Smem)                       |
| 11100000 AAAAAAAI FSSSxxxt | TCx = bit(Smem, src)                        |
| 11100001 AAAAAAAI DDSHIFTW | ACx = low_byte(Smem) << #SHIFTW             |
| 11100010 AAAAAAAI DDSHIFTW | ACx = high_byte(Smem) << #SHIFTW            |
| 11100011 AAAAAAAI kkkk000x | TC1 = bit(Smem, k4), bit(Smem, k4) = #1     |
| 11100011 AAAAAAAI kkkk001x | TC2 = bit(Smem, k4), bit(Smem, k4) = #1     |
| 11100011 AAAAAAAI kkkk010x | TC1 = bit(Smem, k4), bit(Smem, k4) = #0     |
| 11100011 AAAAAAAI kkkk011x | TC2 = bit(Smem, k4), bit(Smem, k4) = #0     |
| 11100011 AAAAAAAI kkkk100x | TC1 = bit(Smem, k4), cbit(Smem, k4)         |
| 11100011 AAAAAAAI kkkk101x | TC2 = bit(Smem, k4), cbit(Smem, k4)         |
| 11100011 AAAAAAAI FSSS1100 | bit(Smem, src) = #1                         |
| 11100011 AAAAAAAI FSSS1101 | bit(Smem, src) = #0                         |
| 11100011 AAAAAAAI FSSS111x | cbit(Smem, src)                             |
| 11100100 AAAAAAAI FSSSx0xx | push(src, Smem)                             |
| 11100100 AAAAAAAI FDDDx1xx | dst, Smem = pop()                           |
| 11100101 AAAAAAAI FSSS01x0 | high_byte(Smem) = src                       |
| 11100101 AAAAAAAI FSSS01x1 | low_byte(Smem) = src                        |
| 11100101 AAAAAAAI 000010xx | Smem = DP                                   |

Table 6-1. Instruction Set Opcodes (Continued)

| Opcode                      | Algebraic syntax                                      |
|-----------------------------|-------------------------------------------------------|
| 11100101 AAAAAAAI 000110xx  | Smem = CDP                                            |
| 11100101 AAAAAAAI 001010xx  | Smem = BSA01                                          |
| 11100101 AAAAAAAI 001110xx  | Smem = BSA23                                          |
| 11100101 AAAAAAAI 010010xx  | Smem = BSA45                                          |
| 11100101 AAAAAAAI 010110xx  | Smem = BSA67                                          |
| 11100101 AAAAAAAI 011010xx  | Smem = BSAC                                           |
| 11100101 AAAAAAAI 011110xx  | Smem = SP                                             |
| 11100101 AAAAAAAI 100010xx  | Smem = SSP                                            |
| 11100101 AAAAAAAI 100110xx  | Smem = BK03                                           |
| 11100101 AAAAAAAI 101010xx  | Smem = BK47                                           |
| 11100101 AAAAAAAI 1011110xx | Smem = BKC                                            |
| 11100101 AAAAAAAI 110010xx  | Smem = DPH                                            |
| 11100101 AAAAAAAI 1111110xx | Smem = PDP                                            |
| 11100101 AAAAAAAI x00011xx  | Smem = CSR                                            |
| 11100101 AAAAAAAI x00111xx  | Smem = BRC0                                           |
| 11100101 AAAAAAAI x01011xx  | Smem = BRC1                                           |
| 11100101 AAAAAAAI x01111xx  | Smem = TRN0                                           |
| 11100101 AAAAAAAI x10011xx  | Smem = TRN1                                           |
| 11100110 AAAAAAAI KKKKKKKK  | Smem = K8                                             |
| 11100111 AAAAAAAI SSss00xx  | Smem = LO(ACx << Tx)                                  |
| 11100111 AAAAAAAI SSss10x%  | Smem = HI(rnd(ACx << Tx))                             |
| 11100111 AAAAAAAI SSss11u%  | Smem = HI(saturate(uns(rnd(ACx << Tx))))              |
| 11101000 AAAAAAAI SSxxx0x%  | Smem = HI(rnd(ACx))                                   |
| 11101000 AAAAAAAI SSxxx1u%  | Smem = HI(saturate(uns(rnd(ACx))))                    |
| 11101001 AAAAAAAI SSSHIFTW  | Smem = LO(ACx << #SHIFTW)                             |
| 11101010 AAAAAAAI SSSHIFTW  | Smem = HI(ACx << #SHIFTW)                             |
| 11101011 AAAAAAAI xxxx01xx  | dbl(Lmem) = RETA                                      |
| 11101011 AAAAAAAI xxSS10x0  | dbl(Lmem) = ACx                                       |
| 11101011 AAAAAAAI xxSS10u1  | dbl(Lmem) = saturate(uns(ACx))                        |
| 11101011 AAAAAAAI FSSS1100  | Lmem = pair(TAx)                                      |
| 11101011 AAAAAAAI xxSS1101  | HI(Lmem) = HI(ACx) >> #1,<br>LO(Lmem) = LO(ACx) >> #1 |
| 11101011 AAAAAAAI xxSS1110  | Lmem = pair(HI(ACx))                                  |

Table 6-1. Instruction Set Opcodes (Continued)

| Орс               | ode      | Algebraic syntax                                              |
|-------------------|----------|---------------------------------------------------------------|
| 11101011 AAAAAAAI | xxSS1111 | Lmem = pair(LO(ACx))                                          |
| 11101100 AAAAAAAI | FSSS000x | bit(src, Baddr) = #1                                          |
| 11101100 AAAAAAAI | FSSS001x | bit(src, Baddr) = #0                                          |
| 11101100 AAAAAAAI | FSSS010x | bit(src, pair(Baddr))                                         |
| 11101100 AAAAAAAI | FSSS011x | cbit(src, Baddr)                                              |
| 11101100 AAAAAAAI | FSSS100t | TCx = bit(src, Baddr)                                         |
| 11101100 AAAAAAAI | XDDD1110 | XAdst = mar(Smem)                                             |
| 11101101 AAAAAAAI | 00DD1010 | pair(HI(ACx)) = Lmem                                          |
| 11101101 AAAAAAAI | 00DD1100 | pair(LO(ACx)) = Lmem                                          |
| 11101101 AAAAAAAI | 00SS1110 | Lmem = pair(HI(ACx))                                          |
| 11101101 AAAAAAAI | 00SS1111 | Lmem = pair(LO(ACx))                                          |
| 11101101 AAAAAAAI | SSDD000n | ACy = ACx + dbl(Lmem)                                         |
| 11101101 AAAAAAAI | SSDD001n | ACy = ACx - dbl(Lmem)                                         |
| 11101101 AAAAAAAI | SSDD010x | ACy = dbl(Lmem) - ACx                                         |
| 11101101 AAAAAAAI | xxxx011x | RETA = dbl(Lmem)                                              |
| 11101101 AAAAAAAI | xxDD100g | $ACx = \frac{M40(dbl(Lmem))}{}$                               |
| 11101101 AAAAAAAI | xxDD101x | pair(HI(ACx)) = Lmem                                          |
| 11101101 AAAAAAAI | xxDD110x | pair(LO(ACx)) = Lmem                                          |
| 11101101 AAAAAAAI | FDDD111x | pair(TAx) = Lmem                                              |
| 11101101 AAAAAAAI | XDDD1111 | XAdst = dbl(Lmem)                                             |
| 11101101 AAAAAAAI | XSSS0101 | dbl(Lmem) = XAsrc                                             |
| 11101110 AAAAAAI  | SSDD000x | HI(ACy) = HI(Lmem) + HI(ACx),<br>LO(ACy) = LO(Lmem) + LO(ACx) |
| 11101110 AAAAAAI  | SSDD001x | HI(ACy) = HI(ACx) - HI(Lmem),<br>LO(ACy) = LO(ACx) - LO(Lmem) |
| 11101110 AAAAAAAI | SSDD010x | HI(ACy) = HI(Lmem) - HI(ACx),<br>LO(ACy) = LO(Lmem) - LO(ACx) |
| 11101110 AAAAAAAI | ssDD011x | HI(ACx) = Tx - HI(Lmem),<br>LO(ACx) = Tx - LO(Lmem)           |
| 11101110 AAAAAAAI | ssDD100x | HI(ACx) = HI(Lmem) + Tx,<br>LO(ACx) = LO(Lmem) + Tx           |
| 11101110 AAAAAAAI | ssDD101x | HI(ACx) = HI(Lmem) - Tx,<br>LO(ACx) = LO(Lmem) - Tx           |
| 11101110 AAAAAAAI | ssDD110x | HI(ACx) = HI(Lmem) + Tx,<br>LO(ACx) = LO(Lmem) - Tx           |

Table 6-1. Instruction Set Opcodes (Continued)

| Oį               | ocode               | Algebraic syntax                                                                                               |
|------------------|---------------------|----------------------------------------------------------------------------------------------------------------|
| 11101110 AAAAAAA | I ssDD111x          | HI(ACx) = HI(Lmem) - Tx,<br>LO(ACx) = LO(Lmem) + Tx                                                            |
| 11101111 AAAAAA  | I xxxx00mm          | Smem = coef(Cmem)                                                                                              |
| 11101111 AAAAAA  | I xxxx01mm          | coef(Cmem) = Smem                                                                                              |
| 11101111 AAAAAA  | I xxxx10mm          | Lmem = dbl(coef(Cmem))                                                                                         |
| 11101111 AAAAAA  | I xxxx11mm          | dbl(coef(Cmem)) = Lmem                                                                                         |
| 11110000 AAAAAA  | I KKKKKKKK KKKKKKK  | TC1 = (Smem == K16)                                                                                            |
| 11110001 AAAAAA  | I KKKKKKKK KKKKKKK  | TC2 = (Smem == K16)                                                                                            |
| 11110010 AAAAAA  | I kkkkkkkk kkkkkkk  | TC1 = Smem & k16                                                                                               |
| 11110011 AAAAAA  | I kkkkkkkk kkkkkkk  | TC2 = Smem & k16                                                                                               |
| 11110100 AAAAAA  | I kkkkkkkk kkkkkkk  | Smem = Smem & k16                                                                                              |
| 11110101 AAAAAA  | I kkkkkkkk kkkkkkk  | Smem = Smem   k16                                                                                              |
| 11110110 AAAAAA  | I kkkkkkkk kkkkkkk  | Smem = Smem ^ k16                                                                                              |
| 11110111 AAAAAA  | I KKKKKKKK KKKKKKK  | Smem = Smem + K16                                                                                              |
| 11111000 AAAAAA  | I KKKKKKKK xxDDx0U% | ACx = rnd(Smem * K8) [,T3 = Smem]                                                                              |
| 11111000 AAAAAA  | I KKKKKKKK SSDDx1U% | ACy = rnd(ACx + (Smem * K8)) [,T3 = Smem]                                                                      |
| 11111001 AAAAAA  | I uxSHIFTW SSDD00xx | ACy = ACx + (uns(Smem) << #SHIFTW)                                                                             |
| 11111001 AAAAAA  | I uxSHIFTW SSDD01xx | ACy = ACx - (uns(Smem) << #SHIFTW)                                                                             |
| 11111001 AAAAAA  | I uxSHIFTW xxDD10xx | ACx = uns(Smem) << #SHIFTW                                                                                     |
| 11111010 AAAAAA  | I xxSHIFTW SSxxx0x% | Smem = HI(rnd(ACx << #SHIFTW))                                                                                 |
| 11111010 AAAAAA  | I uxSHIFTW SSxxx1x% | Smem = HI(saturate(uns(rnd(ACx << #SHIFTW))))                                                                  |
| 11111011 AAAAAA  | I KKKKKKKK KKKKKKK  | Smem = K16                                                                                                     |
| 11111100 AAAAAA  | I LLLLLLLL LLLLLLLL | if (ARn_mod != #0) goto L16                                                                                    |
| 11111101 AAAAAAA | I 000000mm DDDDuug% | ACy = M40(rnd(uns(Smem) * uns(HI(coef(Cmem))))),<br>ACx = M40(rnd(uns(Smem) * uns(LO(coef(Cmem)))))            |
| 11111101 AAAAAAA | I 000001mm DDDDuug% | ACy = M40(rnd(uns(Smem) * uns(HI(coef(Cmem))))),<br>ACx = M40(rnd(ACx + (uns(Smem) *<br>uns(LO(coef(Cmem)))))) |
| 11111101 AAAAAAA | I 000010mm DDDDuug% | ACy = M40(rnd(ACy + (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(uns(Smem) * uns(LO(coef(Cmem)))))       |
| 11111101 AAAAAAA | I 000011mm DDDDuug% | ACy = M40(rnd(uns(Smem) * uns(HI(coef(Cmem))))),<br>ACx = M40(rnd(ACx - (uns(Smem) *<br>uns(LO(coef(Cmem)))))) |

Table 6-1. Instruction Set Opcodes (Continued)

|               | Орс   | ode      |          | Algebraic syntax                                                                                                               |
|---------------|-------|----------|----------|--------------------------------------------------------------------------------------------------------------------------------|
| 11111101 AAA  | AAAAI | 000100mm | DDDDuug% | ACy = M40(rnd(ACy - (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(uns(Smem) * uns(LO(coef(Cmem)))))                       |
| 11111101 AAAA | AAAAI | 000101mm | DDDDuug% | ACy = M40(rnd(ACy + (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx + (uns(Smem) * uns(LO(coef(Cmem))))))               |
| 11111101 AAAA | AAAAI | 000110mm | DDDDuug% | ACy = M40(rnd(ACy + (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx - (uns(Smem) * uns(LO(coef(Cmem))))))               |
| 11111101 AAAA | AAAAI | 000111mm | DDDDuug% | ACy = M40(rnd(ACy - (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx + (uns(Smem) * uns(LO(coef(Cmem))))))               |
| 11111101 AAA  | AAAAI | 001000mm | DDDDuug* | ACy = M40(rnd(ACy + (uns(Smem) * uns(HI(coef(Cmem)))))),<br>ACx = M40(rnd((ACx>>#16) + (uns(Smem) * uns(LO(coef(Cmem))))))     |
| 11111101 AAA  | AAAAI | 001001mm | DDDDuug* | ACy = M40(rnd((ACy>>#16) + (uns(Smem) * uns(HI(coef(Cmem))))),<br>ACx = M40(rnd(ACx - (uns(Smem) * uns(LO(coef(Cmem))))))      |
| 11111101 AAA  | AAAAI | 001010mm | DDDDuug% | ACy = M40(rnd((ACy>>#16) + (uns(Smem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(uns(Smem) * uns(LO(coef(Cmem)))))                 |
| 11111101 AAA  | AAAAI | 001011mm | DDDDuug* | ACy = M40(rnd((ACy>>#16) + (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd((ACx>>#16) + (uns(Smem) * uns(LO(coef(Cmem)))))) |
| 11111101 AAA  | AAAAI | 001100mm | DDDDuug% | ACy = M40(rnd(ACy - (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx - (uns(Smem) * uns(LO(coef(Cmem))))))               |
| 11111101 AAA  | AAAAI | 010000mm | DDDDuug% | ACy = M40(rnd(uns(HI(Lmem)) * uns(HI(coef(Cmem))))),<br>ACx = M40(rnd(uns(LO(Lmem)) * uns(LO(coef(Cmem)))))                    |
| 11111101 AAA  | AAAAI | 010001mm | DDDDuug% | ACy = M40(rnd(uns(HI(Lmem)) * uns(HI(coef(Cmem))))),<br>ACx = M40(rnd(ACx + (uns(LO(Lmem)) *<br>uns(LO(coef(Cmem))))))         |

Table 6-1. Instruction Set Opcodes (Continued)

|                 | Opcode       |          | Algebraic syntax                                                                                                                         |
|-----------------|--------------|----------|------------------------------------------------------------------------------------------------------------------------------------------|
| 11111101 AAAAA  | AAI 010010mm | DDDDuug% | ACy = M40(rnd(ACy + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(uns(LO(Lmem)) * uns(LO(coef(Cmem)))))                         |
| 11111101 AAAAA  | AAI 010011mm | DDDDuug% | ACy = M40(rnd(uns(HI(Lmem)) * uns(HI(coef(Cmem))))),<br>ACx = M40(rnd(ACx - (uns(LO(Lmem)) *<br>uns(LO(coef(Cmem))))))                   |
| 11111101 AAAAA  | AAI 010100mm | DDDDuug% | ACy = M40(rnd(ACy - (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(uns(LO(Lmem)) * uns(LO(coef(Cmem)))))                         |
| 111111101 AAAAA | AAI 010101mm | DDDDuug* | ACy = M40(rnd(ACy + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))),<br>ACx = M40(rnd(ACx + (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))              |
| 111111101 AAAAA | AAI 010110mm | DDDDuug* | ACy = M40(rnd(ACy + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))),<br>ACx = M40(rnd(ACx - (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))              |
| 111111101 AAAAA | AAI 010111mm | DDDDuug% | ACy = M40(rnd(ACy - (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))),<br>ACx = M40(rnd(ACx + (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))              |
| 11111101 AAAAA  | AAI 011000mm | DDDDuug% | ACy = M40(rnd(ACy + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))),<br>ACx = M40(rnd((ACx>>#16) + (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))       |
| 111111101 AAAAA | AAI 011001mm | DDDDuug% | ACy = M40(rnd((ACy>>#16) + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx - (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))          |
| 11111101 AAAAA  | AAI 011010mm | DDDDuug* | ACy = M40(rnd((ACy>>#16) + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(uns(LO(Lmem)) * uns(LO(coef(Cmem)))))                  |
| 11111101 AAAAA  | AAI 011011mm | DDDDuug% | ACy = M40(rnd((ACy>>#16) + (uns(HI(Lmem)) * uns(HI(coef(Cmem))))),<br>ACx = M40(rnd((ACx>>#16) + (uns(LO(Lmem)) * uns(LO(coef(Cmem)))))) |
| 111111101 AAAAA | AAI 011100mm | DDDDuug% | ACy = M40(rnd(ACy - (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx - (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))                 |

## 6.2 Instruction Set Opcode Symbols and Abbreviations

Table 6–2 lists the symbols and abbreviations used in the instruction set opcode.

Table 6–2. Instruction Set Opcode Symbols and Abbreviations

| Bit Field<br>Name | Bit Field<br>Value | Bit Field Description                                          |
|-------------------|--------------------|----------------------------------------------------------------|
| %                 | 0                  | Rounding is disabled                                           |
|                   | 1                  | Rounding is enabled                                            |
| AAAA AAAI         |                    | Smem addressing mode:                                          |
|                   | AAAA AAA0          | @dma, direct memory address (dma) direct access                |
|                   | AAAA AAA1          | Smem indirect memory access:                                   |
|                   | 0001 0001          | ABS16(#k16)                                                    |
|                   | 0011 0001          | *(#k23)                                                        |
|                   | 0101 0001          | *port(#k16)                                                    |
|                   | 0111 0001          | *CDP                                                           |
|                   | 1001 0001          | *CDP+                                                          |
|                   | 1011 0001          | *CDP-                                                          |
|                   | 1101 0001          | *CDP(#K16)                                                     |
|                   | 1111 0001          | *+CDP(#K16)                                                    |
|                   | PPP0 0001          | *ARn                                                           |
|                   | PPP0 0011          | *ARn+                                                          |
|                   | PPP0 0101          | *ARn-                                                          |
|                   | PPP0 0111          | *(ARn + T0), when C54CM = 0<br>*(ARn + T0), when C54CM = 1     |
|                   | PPP0 1001          | *(ARn – T0), when C54CM = 0<br>*(ARn – T0), when C54CM = 1     |
|                   | PPP0 1011          | *ARn(T0), when C54CM = 0<br>*ARn(T0), when C54CM = 1           |
|                   | PPP0 1101          | *ARn(#K16)                                                     |
|                   | PPP0 1111          | *+ARn(#K16)                                                    |
|                   | PPP1 0011          | *(ARn + T1), when ARMS = 0<br>*ARn(short(#1)), when ARMS = 1   |
|                   | PPP1 0101          | *(ARn $-$ T1), when ARMS = 0<br>*ARn(short(#2)), when ARMS = 1 |

Table 6-2. Instruction Set Opcode Symbols and Abbreviations (Continued)

| Bit Field<br>Name | Bit Field<br>Value | Bit Field D                                                                                    | Description                                                              |
|-------------------|--------------------|------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------|
|                   | PPP1 0111          |                                                                                                | when ARMS = 0<br>t(#3)), when ARMS = 1                                   |
|                   | PPP1 1001          |                                                                                                | en ARMS = 0<br>t(#4)), when ARMS = 1                                     |
|                   | PPP1 1011          |                                                                                                | en ARMS = 0<br>t(#5)), when ARMS = 1                                     |
|                   | PPP1 1101          | •                                                                                              | 0B), when ARMS = 0<br>t(#6)), when ARMS = 1                              |
|                   | PPP1 1111          | •                                                                                              | 0B), when ARMS = 0<br>t(#7)), when ARMS = 1                              |
|                   | PPP encode         | s an auxiliary                                                                                 | register (ARn) as for XXX and YYY.                                       |
| cc                |                    | Relational                                                                                     | operators (RELOP):                                                       |
|                   | 00                 | ==                                                                                             | (equal to)                                                               |
|                   | 01                 | <                                                                                              | (less than)                                                              |
|                   | 10                 | >=                                                                                             | (greater than or equal to)                                               |
|                   | 11                 | !=                                                                                             | (not equal to)                                                           |
| CCC CCCC          |                    | Conditional field (cond) on source accumulator, auxiliary, or tempor register; TCx; and CARRY: |                                                                          |
|                   | 000 FSSS           | src == 0                                                                                       | (source is equal to 0)                                                   |
|                   | 001 FSSS           | src != 0                                                                                       | (source is not equal to 0)                                               |
|                   | 010 FSSS           | src < 0                                                                                        | (source is less than 0)                                                  |
|                   | 011 FSSS           | src <= 0                                                                                       | (source is less than or equal to 0)                                      |
|                   | 100 FSSS           | src > 0                                                                                        | (source is greater than 0)                                               |
|                   | 101 FSSS           | src >= 0                                                                                       | (source is greater than or equal to 0)                                   |
|                   | 110 00SS           | overflow(A                                                                                     | Cx) (source accumulator overflow status bit (ACOVx) is tested against 1) |
|                   | 110 0100           | TC1                                                                                            | (status bit is tested against 1)                                         |
|                   | 110 0101           | TC2                                                                                            | (status bit is tested against 1)                                         |
|                   | 110 0110           | CARRY                                                                                          | (status bit is tested against 1)                                         |
|                   | 110 0111           | Reserved                                                                                       |                                                                          |

Table 6–2. Instruction Set Opcode Symbols and Abbreviations (Continued)

| Bit Field<br>Name | Bit Field<br>Value | Bit Field De  | escription                                                              |
|-------------------|--------------------|---------------|-------------------------------------------------------------------------|
|                   | 110 1000           | TC1 & TC2     |                                                                         |
|                   | 110 1001           | TC1 & !TC2    |                                                                         |
|                   | 110 1010           | !TC1 & TC2    |                                                                         |
|                   | 110 1011           | !TC1 & !TC2   | <u>!</u>                                                                |
|                   | 110 11xx           | Reserved      |                                                                         |
|                   | 111 00SS           | !overflow(AC  | cx)(source accumulator overflow status bit (ACOVx) is tested against 0) |
|                   | 111 0100           | !TC1          | (status bit is tested against 0)                                        |
|                   | 111 0101           | !TC2          | (status bit is tested against 0)                                        |
|                   | 111 0110           | !CARRY        | (status bit is tested against 0)                                        |
|                   | 111 0111           | Reserved      |                                                                         |
|                   | 111 1000           | TC1   TC2     |                                                                         |
|                   | 111 1001           | TC1   !TC2    |                                                                         |
|                   | 111 1010           | !TC1   TC2    |                                                                         |
|                   | 111 1011           | !TC1   !TC2   |                                                                         |
|                   | 111 1100           | TC1 ^ TC2     |                                                                         |
|                   | 111 1101           | TC1 ^ !TC2    |                                                                         |
|                   | 111 1110           | !TC1 ^ TC2    |                                                                         |
|                   | 111 1111           | !TC1 ^ !TC2   |                                                                         |
| dd                |                    | Destination t | temporary register (Tx, Ty):                                            |
|                   | 00                 | Temporary re  | egister 0 (T0)                                                          |
|                   | 01                 | Temporary re  | egister 1 (T1)                                                          |
|                   | 10                 | Temporary re  | egister 2 (T2)                                                          |
|                   | 11                 | Temporary re  | egister 3 (T3)                                                          |

Table 6–2. Instruction Set Opcode Symbols and Abbreviations (Continued)

| Bit Field<br>Name | Bit Field<br>Value | Bit Field Description                                                                     |
|-------------------|--------------------|-------------------------------------------------------------------------------------------|
| DD                |                    | Destination accumulator register (ACw, ACx, ACy, ACz):                                    |
|                   | 00                 | Accumulator 0 (AC0)                                                                       |
|                   | 01                 | Accumulator 1 (AC1)                                                                       |
|                   | 10                 | Accumulator 2 (AC2)                                                                       |
|                   | 11                 | Accumulator 3 (AC3)                                                                       |
| DDD               | D                  | Data address label coded on n bits (absolute address)                                     |
| E                 | 0                  | Parallel Enable bit is cleared to 0                                                       |
|                   | 1                  | Parallel Enable bit is set to 1                                                           |
| FDDD<br>FSSS      |                    | Destination or Source accumulator, auxiliary, or temporary register (dst, src, TAx, TAy): |
|                   | 0000               | Accumulator 0 (AC0)                                                                       |
|                   | 0001               | Accumulator 1 (AC1)                                                                       |
|                   | 0010               | Accumulator 2 (AC2)                                                                       |
|                   | 0011               | Accumulator 3 (AC3)                                                                       |
|                   | 0100               | Temporary register 0 (T0)                                                                 |
|                   | 0101               | Temporary register 1 (T1)                                                                 |
|                   | 0110               | Temporary register 2 (T2)                                                                 |
|                   | 0111               | Temporary register 3 (T3)                                                                 |
|                   | 1000               | Auxiliary register 0 (AR0)                                                                |
|                   | 1001               | Auxiliary register 1 (AR1)                                                                |
|                   | 1010               | Auxiliary register 2 (AR2)                                                                |
|                   | 1011               | Auxiliary register 3 (AR3)                                                                |
|                   | 1100               | Auxiliary register 4 (AR4)                                                                |
|                   | 1101               | Auxiliary register 5 (AR5)                                                                |
|                   | 1110               | Auxiliary register 6 (AR6)                                                                |
|                   | 1111               | Auxiliary register 7 (AR7)                                                                |

Table 6–2. Instruction Set Opcode Symbols and Abbreviations (Continued)

| Bit Field<br>Name | Bit Field<br>Value | Bit Field Description                            |
|-------------------|--------------------|--------------------------------------------------|
| g                 | 0                  | 40 keyword is not applied                        |
|                   | 1                  | 40 keyword is applied; M40 is locally set to 1   |
|                   |                    |                                                  |
| kk kkkk           |                    | Swap code for Swap Register Content instruction: |
|                   | 00 0000            | swap(AC0, AC2)                                   |
|                   | 00 0001            | swap(AC1, AC3)                                   |
|                   | 00 0100            | swap(T0, T2)                                     |
|                   | 00 0101            | swap(T1, T3)                                     |
|                   | 00 1000            | swap(AR0, AR2)                                   |
|                   | 00 1001            | swap(AR1, AR3)                                   |
|                   | 00 1100            | swap(AR4, T0)                                    |
|                   | 00 1101            | swap(AR5, T1)                                    |
|                   | 00 1110            | swap(AR6, T2)                                    |
|                   | 00 1111            | swap(AR7, T3)                                    |
|                   | 01 0000            | swap(pair(AC0), pair(AC2))                       |
|                   | 01 0001            | Reserved                                         |
|                   | 01 0100            | swap(pair(T0), pair(T2))                         |
|                   | 01 0101            | Reserved                                         |
|                   | 01 1000            | swap(pair(AR0), pair(AR2))                       |
|                   | 01 1001            | Reserved                                         |
|                   | 01 1100            | swap(pair(AR4), pair(T0))                        |
|                   | 01 1101            | Reserved                                         |
|                   | 01 1110            | swap(pair(AR6), pair(T2))                        |
|                   | 01 1111            | Reserved                                         |
|                   | 10 1000            | Reserved                                         |
|                   | 10 1100            | swap(block(AR4), block(T0))                      |
|                   | 11 1000            | swap(AR0, AR1)                                   |
|                   | 11 1100            | Reserved                                         |
|                   | 1x 0000            | Reserved                                         |
|                   | 1x 0001            | Reserved                                         |

Table 6-2. Instruction Set Opcode Symbols and Abbreviations (Continued)

| Bit Field<br>Name | Bit Field<br>Value | Bit Field Description                                                                        |
|-------------------|--------------------|----------------------------------------------------------------------------------------------|
|                   | 1x 0100            | Reserved                                                                                     |
|                   | 1x 0101            | Reserved                                                                                     |
|                   | 1x 1001            | Reserved                                                                                     |
|                   | 1x 1101            | Reserved                                                                                     |
|                   | 1x 1110            | Reserved                                                                                     |
|                   | 1x 1111            | Reserved                                                                                     |
| kkk k             |                    | Unsigned constant of n bits                                                                  |
| ккк к             |                    | Signed constant of n bits                                                                    |
| 111 1             |                    | Program address label coded on n bits (unsigned offset relative to program counter register) |
| LLL L             |                    | Program address label coded on n bits (signed offset relative to program counter register)   |
| mm                |                    | coef(Cmem)icient addressing mode (Cmem):                                                     |
|                   | 00                 | *CDP                                                                                         |
|                   | 01                 | *CDP+                                                                                        |
|                   | 10                 | *CDP-                                                                                        |
|                   | 11                 | *(CDP + T0)                                                                                  |
| MMM               |                    | Modifier option for Xmem or Ymem addressing mode:                                            |
|                   | 000                | *ARn                                                                                         |
|                   | 001                | *ARn+                                                                                        |
|                   | 010                | *ARn-                                                                                        |
|                   | 011                | *(ARn + T0), when C54CM = 0<br>*(ARn + AR0), when C54CM = 1                                  |
|                   | 100                | *(ARn + T1)                                                                                  |

Table 6–2. Instruction Set Opcode Symbols and Abbreviations (Continued)

| Bit Field<br>Name | Bit Field<br>Value | Bit Field Description                                            |
|-------------------|--------------------|------------------------------------------------------------------|
|                   | 101                | *(ARn – T0), when C54CM = 0<br>*(ARn – AR0), when C54CM = 1      |
|                   | 110                | *(ARn – T1)                                                      |
|                   | 111                | *ARn(T0), when C54CM = 0<br>*ARn(AR0), when C54CM = 1            |
| n                 |                    | Reserved bit                                                     |
| PPP P             |                    | Program or data address label coded on n bits (absolute address) |
| r                 | 0                  | Select TRN0                                                      |
|                   | 1                  | Select TRN1                                                      |
| SHFT              |                    | 4-bit immediate shift value, 0 to 15                             |
| SHIFTW            |                    | 6-bit immediate shift value, –32 to +31                          |
| ss                |                    | Source temporary register (Tx, Ty):                              |
|                   | 00                 | Temporary register 0 (T0)                                        |
|                   | 01                 | Temporary register 1 (T1)                                        |
|                   | 10                 | Temporary register 2 (T2)                                        |
|                   | 11                 | Temporary register 3 (T3)                                        |
| SS                |                    | Source accumulator register (ACw, ACx, ACy, ACz):                |
|                   | 00                 | Accumulator 0 (AC0)                                              |
|                   | 01                 | Accumulator 1 (AC1)                                              |
|                   | 10                 | Accumulator 2 (AC2)                                              |
|                   | 11                 | Accumulator 3 (AC3)                                              |

Table 6–2. Instruction Set Opcode Symbols and Abbreviations (Continued)

| Bit Field<br>Name | Bit Field<br>Value | Bit Field Description                                                                                                                                                                                                           |
|-------------------|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tt                | 00                 | Bit 0: destination TCy bit of Compare Register Content instruction                                                                                                                                                              |
|                   | 01                 | Bit 1: source TCx bit of Compare Register Content instruction                                                                                                                                                                   |
|                   | 10                 | When value = 0: TC1 is selected                                                                                                                                                                                                 |
|                   | 11                 | When value = 1: TC2 is selected                                                                                                                                                                                                 |
| u                 | 0                  | uns keyword is not applied; operand is considered signed                                                                                                                                                                        |
|                   | 1                  | uns keyword is applied; operand is considered unsigned                                                                                                                                                                          |
| U                 | 0                  | No update of T3 with Smem or Xmem content                                                                                                                                                                                       |
|                   | 1                  | T3 is updated with Smem or Xmem content                                                                                                                                                                                         |
| vv                | 00                 | Bit 0: shifted-out bit of Rotate instruction                                                                                                                                                                                    |
|                   | 01                 | Bit 1: shifted-in bit of Rotate instruction                                                                                                                                                                                     |
|                   | 10                 | When value = 0: CARRY is selected                                                                                                                                                                                               |
|                   | 11                 | When value = 1: TC2 is selected                                                                                                                                                                                                 |
| x                 |                    | Reserved bit                                                                                                                                                                                                                    |
| XDDD<br>XSSS      |                    | Destination or Source accumulator or extended register. All 23 bits of stack pointer (XSP), system stack pointer (XSSP), data page pointer (XDP), coef(Cmem)icient data pointer (XCDP), and extended auxiliary register (XARx). |
|                   | 0000               | Accumulator 0 (AC0)                                                                                                                                                                                                             |
|                   | 0001               | Accumulator 1 (AC1)                                                                                                                                                                                                             |
|                   | 0010               | Accumulator 2 (AC2)                                                                                                                                                                                                             |
|                   | 0011               | Accumulator 3 (AC3)                                                                                                                                                                                                             |
|                   | 0100               | Stack pointer (XSP)                                                                                                                                                                                                             |
|                   | 0101               | System stack pointer (XSSP)                                                                                                                                                                                                     |
|                   | 0110               | Data page pointer (XDP)                                                                                                                                                                                                         |
|                   | 0111               | coef(Cmem)icient data pointer (XCDP)                                                                                                                                                                                            |
|                   | 1000               | Auxiliary register 0 (XAR0)                                                                                                                                                                                                     |

Table 6–2. Instruction Set Opcode Symbols and Abbreviations (Continued)

| Bit Field  | Bit Field |                                                                  |
|------------|-----------|------------------------------------------------------------------|
| Name       | Value     | Bit Field Description                                            |
|            | 1001      | Auxiliary register 1 (XAR1)                                      |
|            | 1010      | Auxiliary register 2 (XAR2)                                      |
|            | 1011      | Auxiliary register 3 (XAR3)                                      |
|            | 1100      | Auxiliary register 4 (XAR4)                                      |
|            | 1101      | Auxiliary register 5 (XAR5)                                      |
|            | 1110      | Auxiliary register 6 (XAR6)                                      |
|            | 1111      | Auxiliary register 7 (XAR7)                                      |
|            |           |                                                                  |
| XXX<br>YYY |           | Auxiliary register designation for Xmem or Ymem addressing mode: |
| 111        | 000       | Auxiliany ragiotor 0 (APO)                                       |
|            |           | Auxiliary register 0 (AR0)                                       |
|            | 001       | Auxiliary register 1 (AR1)                                       |
|            | 010       | Auxiliary register 2 (AR2)                                       |
|            | 011       | Auxiliary register 3 (AR3)                                       |
|            | 100       | Auxiliary register 4 (AR4)                                       |
|            | 101       | Auxiliary register 5 (AR5)                                       |
|            | 110       | Auxiliary register 6 (AR6)                                       |
|            | 111       | Auxiliary register 7 (AR7)                                       |

## Cross-Reference of Algebraic and Mnemonic Instruction Sets

This chapter provides a cross-reference between the TMS320C55x<sup>™</sup> DSP algebraic instruction set and the mnemonic instruction set (Table 7–1). For more information on the mnemonic instruction set, see *C55x CPU Mnemonic Instruction Set Reference Guide*, SWPU067.

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets

| Algebraic Syntax                    | Mnemonic Syntax                          |
|-------------------------------------|------------------------------------------|
| Absolute Distance                   | ABDST: Absolute Distance                 |
| abdst(Xmem, Ymem, ACx, ACy)         | ABDST Xmem, Ymem, ACx, ACy               |
| Absolute Value                      | ABS: Absolute Value                      |
| dst =  src                          | ABS [src,] dst                           |
| Addition                            | ADD: Addition                            |
| dst = dst + src                     | ADD [src,] dst                           |
| dst = dst + k4                      | ADD k4, dst                              |
| dst = src + K16                     | ADD K16, [src,] dst                      |
| dst = src + Smem                    | ADD Smem, [src,] dst                     |
| $ACy = ACy + (ACx \ll Tx)$          | ADD ACx << Tx, ACy                       |
| $ACy = ACy + (ACx \ll \#SHIFTW)$    | ADD ACx << #SHIFTW, ACy                  |
| ACy = ACx + (K16 << #16)            | ADD K16 << #16, [ACx,] ACy               |
| ACy = ACx + (K16 << #SHFT)          | ADD K16 << #SHFT, [ACx,] ACy             |
| ACy = ACx + (Smem << Tx)            | ADD Smem << Tx, [ACx,] ACy               |
| ACy = ACx + (Smem << #16)           | ADD Smem << #16, [ACx,] ACy              |
| ACy = ACx + uns(Smem) + CARRY       | ADD [uns(]Smem[)], CARRY, [ACx,] ACy     |
| ACy = ACx + uns(Smem)               | ADD [uns(]Smem[)], [ACx,] ACy            |
| ACy = ACx + (uns(Smem) << #SHIFTW)  | ADD [uns(]Smem[)] << #SHIFTW, [ACx,] ACy |
| ACy = ACx + dbl(Lmem)               | ADD dbl(Lmem), [ACx,] ACy                |
| ACx = (Xmem << #16) + (Ymem << #16) | ADD Xmem, Ymem, ACx                      |
| Smem = Smem + K16                   | ADD K16, Smem                            |

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

| Algebraic Syntax                                                 | Mnemonic Syntax                                                            |
|------------------------------------------------------------------|----------------------------------------------------------------------------|
| Addition with Absolute Value                                     | ADDV: Addition with Absolute Value                                         |
| $ACy = \frac{\text{rnd}(ACy} +  ACx )$                           | ADD[R]V [ACx,] ACy                                                         |
| Addition with Parallel Store Accumulator Content to Memory       | ADD::MOV: Addition with Parallel Store Accumulator Content to Memory       |
| ACy = ACx + (Xmem << #16),<br>Ymem = HI(ACy << T2)               | ADD Xmem << #16, ACx, ACy<br>:: MOV HI(ACy << T2), Ymem                    |
| Addition or Subtraction Conditionally                            | ADDSUBCC: Addition or Subtraction Conditionally                            |
| ACy = adsc(Smem, ACx, TCx)                                       | ADDSUBCC Smem, ACx, TCx, ACy                                               |
| Addition or Subtraction Conditionally with Shift                 | ADDSUB2CC: Addition or Subtraction Conditionally with Shif                 |
| ACy = ads2c(Smem, ACx, Tx, TC1, TC2)                             | ADDSUB2CC Smem, ACx, Tx, TC1, TC2, ACy                                     |
| Addition, Subtraction, or Move Accumulator Content Conditionally | ADDSUBCC: Addition, Subtraction, or Move Accumulator Content Conditionally |
| ACy = adsc(Smem, ACx, TC1, TC2)                                  | ADDSUBCC Smem, ACx, TC1, TC2, ACy                                          |
| Bitwise AND                                                      | AND: Bitwise AND                                                           |
| dst = dst & src                                                  | AND src, dst                                                               |
| dst = src & k8                                                   | AND k8,src, dst                                                            |
| dst = src & k16                                                  | AND k16, src, dst                                                          |
| dst = src & Smem                                                 | AND Smem, src, dst                                                         |
| ACy = ACy & (ACx <<< #SHIFTW)                                    | AND ACx << #SHIFTW[, ACy]                                                  |
| ACy = ACx & (k16 <<< #16)                                        | AND k16 << #16, [ACx,] ACy                                                 |
| ACy = ACx & (k16 <<< #SHFT)                                      | AND k16 << #SHFT, [ACx,] ACy                                               |

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

| Algebraic Syntax                                            | Mnemonic Syntax                                                   |
|-------------------------------------------------------------|-------------------------------------------------------------------|
| Smem = Smem & k16                                           | AND k16, Smem                                                     |
| Bitwise AND Memory with Immediate Value and Compare to Zero | BAND: Bitwise AND Memory with Immediate Value and Compare to Zero |
| TCx = Smem & k16                                            | BAND Smem, k16, TCx                                               |
| Bitwise OR                                                  | OR: Bitwise OR                                                    |
| dst = dst   src                                             | OR src, dst                                                       |
| dst = src   k8                                              | OR k8, src, dst                                                   |
| dst = src   k16                                             | OR k16, src, dst                                                  |
| dst = src   Smem                                            | OR Smem, src, dst                                                 |
| $ACy = ACy \mid (ACx <<< \#SHIFTW)$                         | OR ACx << #SHIFTW[, ACy]                                          |
| $ACy = ACx \mid (k16 <<< #16)$                              | OR k16 << #16, [ACx,] ACy                                         |
| $ACy = ACx \mid (k16 <<< \#SHFT)$                           | OR k16 << #SHFT, [ACx,] ACy                                       |
| Smem = Smem   k16                                           | OR k16, Smem                                                      |
| Bitwise Exclusive OR (XOR)                                  | XOR: Bitwise Exclusive OR (XOR)                                   |
| dst = dst ^ src                                             | XOR src, dst                                                      |
| $dst = src \wedge k8$                                       | XOR k8, src, dst                                                  |
| $dst = src \wedge k16$                                      | XOR k16, src, dst                                                 |
| dst = src ^ Smem                                            | XOR Smem, src, dst                                                |
| $ACy = ACy ^ (ACx <<< #SHIFTW)$                             | XOR ACx << #SHIFTW[, ACy]                                         |
| $ACy = ACx ^ (k16 <<< #16)$                                 | XOR k16 << #16, [ACx,] ACy                                        |
| ACy = ACx ^ (k16 <<< #SHFT)                                 | XOR k16 << #SHFT, [ACx,] ACy                                      |

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

| Algebraic Syntax                      | Mnemonic Syntax                            |
|---------------------------------------|--------------------------------------------|
| Smem = Smem ^ k16                     | XOR k16, Smem                              |
| Branch Conditionally                  | BCC: Branch Conditionally                  |
| if (cond) goto I4                     | BCC I4, cond                               |
| if (cond) goto L8                     | BCC L8, cond                               |
| if (cond) goto L16                    | BCC L16, cond                              |
| if (cond) goto P24                    | BCC P24, cond                              |
| Branch Unconditionally                | B: Branch Unconditionally                  |
| goto ACx                              | В АСх                                      |
| goto L7                               | B L7                                       |
| goto L16                              | B L16                                      |
| goto P24                              | B P24                                      |
| Branch on Auxiliary Register Not Zero | BCC: Branch on Auxiliary Register Not Zero |
| if (ARn_mod != #0) goto L16           | BCC L16, ARn_mod != #0                     |
| Call Conditionally                    | CALLCC: Call Conditionally                 |
| if (cond) call L16                    | CALLCC L16, cond                           |
| if (cond) call P24                    | CALLCC P24, cond                           |
| Call Unconditionally                  | CALL: Call Unconditionally                 |
| call ACx                              | CALL ACx                                   |
| call L16                              | CALL L16                                   |
| call P24                              | CALL P24                                   |

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

| Algebraic Syntax                                                       | Mnemonic Syntax                                                                |
|------------------------------------------------------------------------|--------------------------------------------------------------------------------|
| Circular Addressing Qualifier                                          | .CR: Circular Addressing Qualifier                                             |
| circular()                                                             | <instruction>.CR</instruction>                                                 |
| Clear Accumulator, Auxiliary, or Temporary Register Bit                | BCLR: Clear Accumulator, Auxiliary, or Temporary Register Bi                   |
| bit(src, Baddr) = #0                                                   | BCLR Baddr, src                                                                |
| Clear Memory Bit                                                       | BCLR: Clear Memory Bit                                                         |
| bit(Smem, src) = #0                                                    | BCLR src, Smem                                                                 |
| Clear Status Register Bit                                              | BCLR: Clear Status Register Bit                                                |
| bit(STx, k4) = #0                                                      | BCLR k4, STx_55                                                                |
|                                                                        | BCLR f-name                                                                    |
| Compare Accumulator, Auxiliary, or Temporary Register Content          | CMP: Compare Accumulator, Auxiliary, or Temporary Register Content             |
| TCx = uns(src RELOP dst)                                               | CMP[U] src RELOP dst, TCx                                                      |
| Compare Accumulator, Auxiliary, or Temporary Register Content with AND | CMPAND: Compare Accumulator, Auxiliary, or Temporary Register Content with AND |
| TCx = TCy & uns(src RELOP dst)                                         | CMPAND[U] src RELOP dst, TCy, TCx                                              |
| TCx = !TCy & uns(src RELOP dst)                                        | CMPAND[U] src RELOP dst, !TCy, TCx                                             |
| Compare Accumulator, Auxiliary, or Temporary Register Content with OR  | CMPOR: Compare Accumulator, Auxiliary, or Temporary Register Content with OR   |
| TCx = TCy   uns(src RELOP dst)                                         | CMPOR[U] src RELOP dst, TCy, TCx                                               |
| TCx = !TCy   uns(src RELOP dst)                                        | CMPOR[U] src RELOP dst, !TCy, TCx                                              |

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

| Algebraic Syntax                                                      | Mnemonic Syntax                                                            |
|-----------------------------------------------------------------------|----------------------------------------------------------------------------|
| Compare Accumulator, Auxiliary, or Temporary Register Content Maximum | MAX: Compare Accumulator, Auxiliary, or Temporary Register Content Maximum |
| dst = max(src, dst)                                                   | MAX [src,] dst                                                             |
| Compare Accumulator, Auxiliary, or Temporary Register Content Minimum | MIN: Compare Accumulator, Auxiliary, or Temporary Register Content Minimum |
| dst = min(src, dst)                                                   | MIN [src,] dst                                                             |
| Compare and Branch                                                    | BCC: Compare and Branch                                                    |
| compare (uns(src RELOP K8)) goto L8                                   | BCC[U] L8, src RELOP K8                                                    |
| Compare and Select Accumulator Content Maximum                        | MAXDIFF: Compare and Select Accumulator Content Maximum                    |
| max_diff(ACx, ACy, ACz, ACw)                                          | MAXDIFF ACx, ACy, ACz, ACw                                                 |
| max_diff_dbl(ACx, ACy, ACz, ACw, TRNx)                                | DMAXDIFF ACx, ACy, ACz, ACw, TRNx                                          |
| Compare and Select Accumulator Content Minimum                        | MINDIFF: Compare and Select Accumulator Content Minimum                    |
| min_diff(ACx, ACy, ACz, ACw)                                          | MINDIFF ACx, ACy, ACz, ACw                                                 |
| min_diff_dbl(ACx, ACy, ACz, ACw, TRNx)                                | DMINDIFF ACx, ACy, ACz, ACw, TRNx                                          |
| Compare Memory with Immediate Value                                   | CMP: Compare Memory with Immediate Value                                   |
| TCx = (Smem == K16)                                                   | CMP Smem == K16, TCx                                                       |
| Complement Accumulator, Auxiliary, or Temporary Register Bit          | BNOT: Complement Accumulator, Auxiliary, or Temporary Register Bit         |
| cbit(src, Baddr)                                                      | BNOT Baddr, src                                                            |
|                                                                       |                                                                            |

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

| Algebraic Syntax                                                 | Mnemonic Syntax                                                       |
|------------------------------------------------------------------|-----------------------------------------------------------------------|
| Complement Accumulator, Auxiliary, or Temporary Register Content | NOT: Complement Accumulator, Auxiliary, or Temporary Register Content |
| dst = ~src                                                       | NOT [src,] dst                                                        |
| Complement Memory Bit                                            | BNOT: Complement Memory Bit                                           |
| cbit(Smem, src)                                                  | BNOT src, Smem                                                        |
| Compute Exponent of Accumulator Content                          | EXP: Compute Exponent of Accumulator Content                          |
| Tx = exp(ACx)                                                    | EXP ACx, Tx                                                           |
| Compute Mantissa and Exponent of Accumulator Content             | MANT::NEXP: Compute Mantissa and Exponent of Accumulator Content      |
| ACy = mant(ACx), Tx = -exp(ACx)                                  | MANT ACx, ACy<br>:: NEXP ACx, Tx                                      |
| Count Accumulator Bits                                           | BCNT: Count Accumulator Bits                                          |
| Tx = count(ACx, ACy, TCx)                                        | BCNT ACx, ACy, TCx, Tx                                                |
| Dual 16-Bit Additions                                            | ADD: Dual 16-Bit Additions                                            |
| HI(ACy) = HI(Lmem) + HI(ACx),<br>LO(ACy) = LO(Lmem) + LO(ACx)    | ADD dual(Lmem), [ACx,] ACy                                            |
| HI(ACx) = HI(Lmem) + Tx,<br>LO(ACx) = LO(Lmem) + Tx              | ADD dual(Lmem), Tx, ACx                                               |

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

| Algebraic Syntax                                              | Mnemonic Syntax                              |
|---------------------------------------------------------------|----------------------------------------------|
| Dual 16-Bit Addition and Subtraction                          | ADDSUB: Dual 16-Bit Addition and Subtraction |
| HI(ACx) = Smem + Tx,<br>LO(ACx) = Smem - Tx                   | ADDSUB Tx, Smem, ACx                         |
| HI(ACx) = HI(Lmem) + Tx,<br>LO(ACx) = LO(Lmem) - Tx           | ADDSUB Tx, dual(Lmem), ACx                   |
| Dual 16-Bit Subtractions                                      | SUB: Dual 16-Bit Subtractions                |
| HI(ACy) = HI(ACx) - HI(Lmem),<br>LO(ACy) = LO(ACx) - LO(Lmem) | SUB dual(Lmem), [ACx,] ACy                   |
| HI(ACy) = HI(Lmem) - HI(ACx),<br>LO(ACy) = LO(Lmem) - LO(ACx) | SUB ACx, dual(Lmem), ACy                     |
| HI(ACx) = Tx - HI(Lmem),<br>LO(ACx) = Tx - LO(Lmem)           | SUB dual(Lmem), Tx, ACx                      |
| HI(ACx) = HI(Lmem) - Tx,<br>LO(ACx) = LO(Lmem) - Tx           | SUB Tx, dual(Lmem), ACx                      |
| Dual 16-Bit Subtraction and Addition                          | SUBADD: Dual 16-Bit Subtraction and Addition |
| HI(ACx) = Smem - Tx,<br>LO(ACx) = Smem + Tx                   | SUBADD Tx, Smem, ACx                         |
| HI(ACx) = HI(Lmem) - Tx,<br>LO(ACx) = LO(Lmem) + Tx           | SUBADD Tx, dual(Lmem), ACx                   |
| Execute Conditionally                                         | XCC: Execute Conditionally                   |
| if (cond) execute(AD_Unit)                                    | XCC [label, ]cond                            |
| if (cond) execute(D_Unit)                                     | XCCPART [label, ]cond                        |

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

| Algebraic Syntax                                | Mnemonic Syntax                                          |
|-------------------------------------------------|----------------------------------------------------------|
| Expand Accumulator Bit Field                    | BFXPA: Expand Accumulator Bit Field                      |
| dst = field_expand(ACx, k16)                    | BFXPA k16, ACx, dst                                      |
| Extract Accumulator Bit Field                   | BFXTR: Extract Accumulator Bit Field                     |
| dst = field_extract(ACx, k16)                   | BFXTR k16, ACx, dst                                      |
| Finite Impulse Response Filter, Antisymmetrical | FIRSSUB: Finite Impulse Response Filter, Antisymmetrical |
| firsn(Xmem, Ymem, coef(Cmem), ACx, ACy)         | FIRSSUB Xmem, Ymem, Cmem, ACx, ACy                       |
| Finite Impulse Response Filter, Symmetrical     | FIRSADD: Finite Impulse Response Filter, Symmetrical     |
| firs(Xmem, Ymem, coef(Cmem), ACx, ACy)          | FIRSADD Xmem, Ymem, Cmem, ACx, ACy                       |
| Idle                                            | IDLE                                                     |
| idle                                            | IDLE                                                     |
| Least Mean Square (LMS)                         | LMS: Least Mean Square                                   |
| Ims(Xmem, Ymem, ACx, ACy)                       | LMS Xmem, Ymem, ACx, ACy                                 |
| Imsf(Xmem, Ymem, ACx, ACy)                      | LMSF Xmem, Ymem, ACx, ACy                                |
| Linear Addressing Qualifier                     | .LR: Linear Addressing Qualifier                         |
| linear()                                        | <instruction>.LR</instruction>                           |

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

| Algebraic Syntax                                                               | Mnemonic Syntax                                                                          |
|--------------------------------------------------------------------------------|------------------------------------------------------------------------------------------|
| Load Accumulator from Memory                                                   | MOV: Load Accumulator from Memory                                                        |
| ACx = rnd(Smem << Tx)                                                          | MOV [rnd(]Smem << Tx[)], ACx                                                             |
| ACx = low_byte(Smem) << #SHIFTW                                                | MOV low_byte(Smem) << #SHIFTW, ACx                                                       |
| ACx = high_byte(Smem) << #SHIFTW                                               | MOV high_byte(Smem) << #SHIFTW, ACx                                                      |
| ACx = Smem << #16                                                              | MOV Smem << #16, ACx                                                                     |
| ACx = uns(Smem)                                                                | MOV [uns(]Smem[)], ACx                                                                   |
| ACx = uns(Smem) << #SHIFTW                                                     | MOV [uns(]Smem[)] << #SHIFTW, ACx                                                        |
| $ACx = \frac{M40(dbl(Lmem))}{}$                                                | MOV[40] dbl(Lmem), ACx                                                                   |
| LO(ACx) = Xmem,<br>HI(ACx) = Ymem                                              | MOV Xmem, Ymem, ACx                                                                      |
| Load Accumulator from Memory with Parallel Store Accumulator Content to Memory | MOV::MOV: Load Accumulator from Memory with Parallel Store Accumulator Content to Memory |
| ACy = Xmem << #16,<br>Ymem = HI(ACx << T2)                                     | MOV Xmem << #16, ACy<br>:: MOV HI(ACx << T2), Ymem                                       |
| Load Accumulator Pair from Memory                                              | MOV: Load Accumulator Pair from Memory                                                   |
| pair(HI(ACx)) = Lmem                                                           | MOV dbl(Lmem), pair(HI(ACx))                                                             |
| pair(LO(ACx)) = Lmem                                                           | MOV dbl(Lmem), pair(LO(ACx))                                                             |
| Load Accumulator with Immediate Value                                          | MOV: Load Accumulator with Immediate Value                                               |
| ACx = K16 << #16                                                               | MOV K16 << #16, ACx                                                                      |
| ACx = K16 << #SHFT                                                             | MOV K16 << #SHFT, ACx                                                                    |

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

| Algebraic Syntax                                                        | Mnemonic Syntax                                                              |
|-------------------------------------------------------------------------|------------------------------------------------------------------------------|
| Load Accumulator, Auxiliary, or Temporary Register from Memory          | MOV: Load Accumulator, Auxiliary, or Temporary Register from Memory          |
| dst = Smem                                                              | MOV Smem, dst                                                                |
| dst = uns(high_byte(Smem))                                              | MOV [uns(]high_byte(Smem)[)], dst                                            |
| dst = uns(low_byte(Smem))                                               | MOV [uns(]low_byte(Smem)[)], dst                                             |
| Load Accumulator, Auxiliary, or Temporary Register with Immediate Value | MOV: Load Accumulator, Auxiliary, or Temporary Register with Immediate Value |
| dst = k4                                                                | MOV k4, dst                                                                  |
| dst = -k4                                                               | MOV -k4, dst                                                                 |
| dst = K16                                                               | MOV K16, dst                                                                 |
| Load Auxiliary or Temporary Register Pair from Memory                   | MOV: Load Auxiliary or Temporary Register Pair from Memory                   |
| pair(TAx) = Lmem                                                        | MOV dbl(Lmem), pair(TAx)                                                     |
| Load CPU Register from Memory                                           | MOV: Load CPU Register from Memory                                           |
| BK03 = Smem                                                             | MOV Smem, BK03                                                               |
| BK47 = Smem                                                             | MOV Smem, BK47                                                               |
| BKC = Smem                                                              | MOV Smem, BKC                                                                |
| BSA01 = Smem                                                            | MOV Smem, BSA01                                                              |
| BSA23 = Smem                                                            | MOV Smem, BSA23                                                              |
| BSA45 = Smem                                                            | MOV Smem, BSA45                                                              |
| BSA67 = Smem                                                            | MOV Smem, BSA67                                                              |
| BSAC = Smem                                                             | MOV Smem, BSAC                                                               |
| BRC0 = Smem                                                             | MOV Smem, BRC0                                                               |

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

| Algebraic Syntax                       | Mnemonic Syntax                             |
|----------------------------------------|---------------------------------------------|
| BRC1 = Smem                            | MOV Smem, BRC1                              |
| CDP = Smem                             | MOV Smem, CDP                               |
| CSR = Smem                             | MOV Smem, CSR                               |
| DP = Smem                              | MOV Smem, DP                                |
| DPH = Smem                             | MOV Smem, DPH                               |
| PDP = Smem                             | MOV Smem, PDP                               |
| SP = Smem                              | MOV Smem, SP                                |
| SSP = Smem                             | MOV Smem, SSP                               |
| TRN0 = Smem                            | MOV Smem, TRN0                              |
| TRN1 = Smem                            | MOV Smem, TRN1                              |
| RETA = dbl(Lmem)                       | MOV dbl(Lmem), RETA                         |
| Load CPU Register with Immediate Value | MOV: Load CPU Register with Immediate Value |
| BK03 = k12                             | MOV k12, BK03                               |
| BK47 = k12                             | MOV k12, BK47                               |
| BKC = k12                              | MOV k12, BKC                                |
| BRC0 = k12                             | MOV k12, BRC0                               |
| BRC1 = k12                             | MOV k12, BRC1                               |
| CSR = k12                              | MOV k12, CSR                                |
| DPH = k7                               | MOV k7, DPH                                 |
| PDP = k9                               | MOV k9, PDP                                 |
| BSA01 = k16                            | MOV k16, BSA01                              |
| BSA23 = k16                            | MOV k16, BSA23                              |

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

| Algebraic Syntax                                      | Mnemonic Syntax                                             |
|-------------------------------------------------------|-------------------------------------------------------------|
| BSA45 = k16                                           | MOV k16, BSA45                                              |
| BSA67 = k16                                           | MOV k16, BSA67                                              |
| BSAC = k16                                            | MOV k16, BSAC                                               |
| CDP = k16                                             | MOV k16, CDP                                                |
| DP = k16                                              | MOV k16, DP                                                 |
| SP = k16                                              | MOV k16, SP                                                 |
| SSP = k16                                             | MOV k16, SSP                                                |
| Load Extended Auxiliary Register from Memory          | MOV: Load Extended Auxiliary Register from Memory           |
| XAdst = dbl(Lmem)                                     | MOV dbI(Lmem), XAdst                                        |
| Load Extended Auxiliary Register with Immediate Value | AMOV: Load Extended Auxiliary Register with Immediate Value |
| XAdst = k23                                           | AMOV k23, XAdst                                             |
| Load Memory with Immediate Value                      | MOV: Load Memory with Immediate Value                       |
| Smem = K8                                             | MOV K8, Smem                                                |
| Smem = K16                                            | MOV K16, Smem                                               |
| Lock Access Qualifier                                 | .LK: Lock Access Qualifier                                  |
| lock()                                                | .LK                                                         |
| Memory Delay                                          | DELAY: Memory Delay                                         |
|                                                       | DELAY Smem                                                  |

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

| Algebraic Syntax                                                           | Mnemonic Syntax                                                                   |
|----------------------------------------------------------------------------|-----------------------------------------------------------------------------------|
| Memory-Mapped Register Access Qualifier                                    | mmap: Memory-Mapped Register Access Qualifier                                     |
| mmap()                                                                     | mmap                                                                              |
| Modify Auxiliary Register Content                                          | AMAR: Modify Auxiliary Register Content                                           |
| mar(Smem)                                                                  | AMAR Smem                                                                         |
| Modify Auxiliary Register Content with Parallel Multiply                   | AMAR::MPY: Modify Auxiliary Register Content with Paralle Multiply                |
| mar(Xmem),<br>ACx = M40(rnd(uns(Ymem) * uns(coef(Cmem))))                  | AMAR Xmem :: MPY[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACx                         |
| Modify Auxiliary Register Content with Parallel Multiply and Accumulate    | AMAR::MAC: Modify Auxiliary Register Content with Paralle Multiply and Accumulate |
| mar(Xmem),<br>ACx = M40(rnd(ACx + (uns(Ymem) * uns(coef(Cmem)))))          | AMAR Xmem :: MAC[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACx                         |
| mar(Xmem),<br>ACx = M40(rnd((ACx >> #16) + (uns(Ymem) * uns(coef(Cmem))))) | AMAR Xmem :: MAC[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACx >> #16                  |
| Modify Auxiliary Register Content with Parallel Multiply and Subtract      | AMAR::MAS: Modify Auxiliary Register Content with Paralle Multiply and Subtract   |
| mar(Xmem),<br>ACx = M40(rnd(ACx - (uns(Ymem) * uns(coef(Cmem)))))          | AMAR Xmem :: MAS[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACx                         |
| Modify Auxiliary or Temporary Register Content                             | AMOV: Modify Auxiliary or Temporary Register Content                              |
| mar(TAy = TAx)                                                             | AMOV TAx, TAy                                                                     |
| mar(TAx = P8)                                                              | AMOV P8, TAx                                                                      |
| mar(TAx = D16)                                                             | AMOV D16, TAx                                                                     |
|                                                                            |                                                                                   |

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

| Algebraic Syntax                                              | Mnemonic Syntax                                                     |
|---------------------------------------------------------------|---------------------------------------------------------------------|
| Modify Auxiliary or Temporary Register Content by Addition    | AADD: Modify Auxiliary or Temporary Register Content by Addition    |
| mar(TAy + TAx)                                                | AADD TAx, TAy                                                       |
| mar(TAx + P8)                                                 | AADD P8, TAx                                                        |
| Modify Auxiliary or Temporary Register Content by Subtraction | ASUB: Modify Auxiliary or Temporary Register Content by Subtraction |
| mar(TAy – TAx)                                                | ASUB TAx, TAy                                                       |
| mar(TAx – P8)                                                 | ASUB P8, TAx                                                        |
| Modify Data Stack Pointer                                     | AADD: Modify Data Stack Pointer (SP)                                |
| SP = SP + K8                                                  | AADD K8, SP                                                         |
| Modify Extended Auxiliary Register Content                    | AMAR: Modify Extended Auxiliary Register Content                    |
| XAdst = mar(Smem)                                             | AMAR Smem, XAdst                                                    |
| mar(XACdst = XACsrc) for DAG_X                                | AMOV XACsrc, XACdst for DAG_X                                       |
| mar(XACdst = XACsrc) for DAG_Y                                | AMOV XACsrc, XACdst for DAG_Y                                       |
| Modify Extended Auxiliary Register Content by Addition        | AADD: Modify Extended Auxiliary Register Content by Addition        |
| mar(XACdst + XACsrc) for DAG_X                                | AADD XACsrc, XACdst for DAG_X                                       |
| mar(XACdst + XACsrc) for DAG_Y                                | AADD XACsrc, XACdst for DAG_Y                                       |
|                                                               |                                                                     |

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

| Algebraic Syntax                                             | Mnemonic Syntax                                                   |
|--------------------------------------------------------------|-------------------------------------------------------------------|
| Modify Extended Auxiliary Register Content by Subtraction    | ASUB: Modify Extended Auxiliary Register Content by Subtraction   |
| mar(XACdst – XACsrc) for DAG_X                               | ASUB XACsrc, XACdst for DAG_X                                     |
| mar(XACdst - XACsrc) for DAG_Y                               | ASUB XACsrc, XACdst for DAG_Y                                     |
| Move Accumulator Content to Auxiliary or Temporary Register  | MOV: Move Accumulator Content to Auxiliary or Temporary Register  |
| TAx = HI(ACx)                                                | MOV HI(ACx), TAx                                                  |
| Move Accumulator, Auxiliary, or Temporary Register Content   | MOV: Move Accumulator, Auxiliary, or Temporary Register Content   |
| dst = src                                                    | MOV src, dst                                                      |
| Move Auxiliary or Temporary Register Content to Accumulator  | MOV: Move Auxiliary or Temporary Register Content to Accumulator  |
| HI(ACx) = TAx                                                | MOV TAx, HI(ACx)                                                  |
| Move Auxiliary or Temporary Register Content to CPU Register | MOV: Move Auxiliary or Temporary Register Content to CPU Register |
| BRC0 = TAx                                                   | MOV TAx, BRC0                                                     |
| BRC1 = TAx                                                   | MOV TAx, BRC1                                                     |
| CDP = TAx                                                    | MOV TAx, CDP                                                      |
| CSR = TAx                                                    | MOV TAx, CSR                                                      |
| SP = TAx                                                     | MOV TAx, SP                                                       |
| SSP = TAx                                                    | MOV TAx, SSP                                                      |

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

| Algebraic Syntax                                             | Mnemonic Syntax                                                   |
|--------------------------------------------------------------|-------------------------------------------------------------------|
| Move CPU Register Content to Auxiliary or Temporary Register | MOV: Move CPU Register Content to Auxiliary or Temporary Register |
| TAx = BRC0                                                   | MOV BRC0, TAx                                                     |
| TAx = BRC1                                                   | MOV BRC1, TAx                                                     |
| TAx = CDP                                                    | MOV CDP, TAx                                                      |
| TAx = RPTC                                                   | MOV RPTC, TAx                                                     |
| TAx = SP                                                     | MOV SP, TAX                                                       |
| TAx = SSP                                                    | MOV SSP, TAx                                                      |
| Move Extended Auxiliary Register Content                     | MOV: Move Extended Auxiliary Register Content                     |
| xdst = xsrc                                                  | MOV xsrc, xdst                                                    |
| Move Memory to Memory                                        | MOV: Move Memory to Memory                                        |
| Smem = coef(Cmem)                                            | MOV Cmem, Smem                                                    |
| coef(Cmem) = Smem                                            | MOV Smem, Cmem                                                    |
| Lmem = dbl(coef(Cmem))                                       | MOV Cmem, dbl(Lmem)                                               |
| dbl(coef(Cmem)) = Lmem                                       | MOV dbl(Lmem), Cmem                                               |
| dbl(Ymem) = dbl(Xmem)                                        | MOV dbl(Xmem), dbl(Ymem)                                          |
| Ymem = Xmem                                                  | MOV Xmem, Ymem                                                    |

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

| Algebraic Syntax                                                                                                                                                  | Mnemonic Syntax                                                                                              |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------|
| Multiply                                                                                                                                                          | MPY: Multiply                                                                                                |
| ACy = rnd(ACy * ACx)                                                                                                                                              | MPY[R] [ACx,] ACy                                                                                            |
| ACy = rnd(ACx * Tx)                                                                                                                                               | MPY[R] Tx, [ACx,] ACy                                                                                        |
| ACy = rnd(ACx * K8)                                                                                                                                               | MPYK[R] K8, [ACx,] ACy                                                                                       |
| $ACy = \frac{rnd}{ACx} * K16)$                                                                                                                                    | MPYK[R] K16, [ACx,] ACy                                                                                      |
| ACx = rnd(Smem * uns(coef(Cmem)))                                                                                                                                 | MPY[R] Smem, uns(Cmem), ACx                                                                                  |
| ACx = rnd(Smem * coef(Cmem))[, T3 = Smem]                                                                                                                         | MPYM[R] [T3 = ]Smem, Cmem, ACx                                                                               |
| ACy = rnd(Smem * ACx)[, T3 = Smem]                                                                                                                                | MPYM[R] [T3 = ]Smem, [ACx,] ACy                                                                              |
| ACx = rnd(Smem * K8)[, T3 = Smem]                                                                                                                                 | MPYMK[R] [T3 = ]Smem, K8, ACx                                                                                |
| ACx = M40(rnd(uns(Xmem) * uns(Ymem)))[, T3 = Xmem]                                                                                                                | $MPYM[R][40] \ [T3 = ][uns(]Xmem[)], \ [uns(]Ymem[)], \ ACx$                                                 |
| ACx = rnd(uns(Tx * Smem))[, T3 = Smem]                                                                                                                            | MPYM[R][U][T3 = ]Smem, Tx, ACx                                                                               |
| Multiply with Parallel Multiply and Accumulate                                                                                                                    | MPY::MAC: Multiply with Parallel Multiply and Accumulate                                                     |
| $ \begin{aligned} &ACx = M40(rnd(uns(Xmem)\ ^*\ uns(coef(Cmem)))), \\ &ACy = M40(rnd((ACy >> \#16)\ +\ (uns(Ymem)\ ^*\ uns(coef(Cmem))))) \end{aligned} $         | MPY[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx :: MAC[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy >> #16          |
| ACy = M40(rnd(uns(Smem) * uns(HI(coef(Cmem))))),<br>ACx = M40(rnd(ACx + (uns(Smem) * uns(LO(coef(Cmem))))))                                                       | MPY[R][40] [uns(]Smem[)], [uns(]HI(Cmem)[)], ACy :: MAC[R][40] [uns(]Smem[)], [uns(]LO(Cmem)[)], ACx         |
| $ \begin{aligned} &ACy = M40(rnd(uns(HI(Lmem)) \ ^* \ uns(HI(coef(Cmem))))), \\ &ACx = M40(rnd(ACx + (uns(LO(Lmem)) \ ^* \ uns(LO(coef(Cmem)))))) \end{aligned} $ | MPY[R][40] [uns(]HI(Lmem)[)], [uns(]HI(Cmem)[)], ACy :: MAC[R][40] [uns(]LO(Lmem)[)], [uns(]LO(Cmem)[)], ACx |
| ACy = M40(rnd(uns(Ymem) * uns(HI(coef(Cmem))))),<br>ACx = M40(rnd(ACx + uns(Xmem) * uns(LO(coef(Cmem)))))                                                         | MPY[R][40] [uns(]Ymem[)], [uns(]HI(Cmem)[)], ACy,<br>:: MAC[R][40] [uns(]Xmem[)], [uns(]LO(Cmem)[)], ACx     |

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

| Algebraic Syntax                                                                                                                                                                                             | Mnemonic Syntax                                                                                              |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------|
| Multiply with Parallel Multiply and Subtract                                                                                                                                                                 | MPY::MAS: Multiply with Parallel Multiply and Subtract                                                       |
| ACy = M40(rnd(uns(Smem) * uns(HI(coef(Cmem))))),<br>ACx = M40(rnd(ACx - (uns(Smem) * uns(LO(coef(Cmem))))))                                                                                                  | MPY[R][40] [uns(]Smem[)], [uns(]HI(Cmem)[)], ACy :: MAS[R][40] [uns(]Smem[)], [uns(]LO(Cmem)[)], ACx         |
| $ACy = \frac{M40(\text{rnd(uns(HI(Lmem))} * uns(HI(coef(Cmem)))))}{ACx} + \frac{M40(\text{rnd(ACx} - (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))}{M40(\text{mod(ACx} - (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))}$ | MPY[R][40] [uns(]HI(Lmem)[)], [uns(]HI(Cmem)[)], ACy :: MAS[R][40] [uns(]LO(Lmem)[)], [uns(]LO(Cmem)[)], ACx |
| ACy = M40(rnd(uns(Ymem) * uns(HI(coef(Cmem))))),<br>ACx = M40(rnd(ACx - uns(Xmem) * uns(LO(coef(Cmem)))))                                                                                                    | MPY[R][40] [uns(]Ymem[)], [uns(]HI(Cmem)[)], ACy, :: MAS[R][40] [uns(]Xmem[)], [uns(]LO(Cmem)[)], ACx        |
| Multiply with Parallel Store Accumulator Content to Memory                                                                                                                                                   | MPYM::MOV: Multiply with Parallel Store Accumulator Content to Memory                                        |
| ACy = rnd(Tx * Xmem),<br>Ymem = HI(ACx << T2) [,T3 = Xmem]                                                                                                                                                   | MPYM[R] [T3 = ]Xmem, Tx, ACy<br>:: MOV HI(ACx << T2), Ymem                                                   |
| Multiply and Accumulate (MAC)                                                                                                                                                                                | MAC: Multiply and Accumulate                                                                                 |
| $ACy = \frac{rnd(ACy + (ACx * Tx))}{rnd(ACy + (ACx * Tx))}$                                                                                                                                                  | MAC[R] ACx, Tx, ACy[, ACy]                                                                                   |
| $ACy = \frac{rnd((ACy * Tx) + ACx)}{rnd((ACy * Tx) + ACx)}$                                                                                                                                                  | MAC[R] ACy, Tx, ACx, ACy                                                                                     |
| ACx = rnd(ACx + (Smem * uns(coef(Cmem))))                                                                                                                                                                    | MAC[R] Smem, uns(Cmem), ACx                                                                                  |
| $ACy = \frac{rnd(ACx + (Tx * K8))}{rnd(ACx + (Tx * K8))}$                                                                                                                                                    | MACK[R] Tx, K8, [ACx,] ACy                                                                                   |
| $ACy = \frac{rnd(ACx + (Tx * K16))}{rnd(ACx + (Tx * K16))}$                                                                                                                                                  | MACK[R] Tx, K16, [ACx,] ACy                                                                                  |
| ACx = rnd(ACx + (Smem * coef(Cmem)))[, T3 = Smem]                                                                                                                                                            | MACM[R] [T3 = ]Smem, Cmem, ACx                                                                               |
| ACy = rnd(ACy + (Smem * ACx))[, T3 = Smem]                                                                                                                                                                   | MACM[R] [T3 = ]Smem, [ACx,] ACy                                                                              |
| $ACy = \frac{\text{rnd}(ACx + (Tx * Smem))[, T3 = Smem]}{}$                                                                                                                                                  | MACM[R] [T3 = ]Smem, Tx, [ACx,] ACy                                                                          |
| ACy = rnd(ACx + (Smem * K8))[, T3 = Smem]                                                                                                                                                                    | MACMK[R] [T3 = ]Smem, K8, [ACx,] ACy                                                                         |
| ACy = M40(rnd(ACx + (uns(Xmem) * uns(Ymem))))[, T3 = Xmem]                                                                                                                                                   | MACM[R][40] [T3 = ][uns(]Xmem[)], [uns(]Ymem[)], [ACx,] ACy                                                  |
| ACy = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(Ymem))))<br>[, T3 = Xmem]                                                                                                                                      | MACM[R][40] [T3 = ][uns(]Xmem[)], [uns(]Ymem[)], ACx >> #16 [, ACy]                                          |

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

| Algebraic Syntax                                                                                                                                                                                                         | Mnemonic Syntax                                                                                                        |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------|
| Multiply and Accumulate with Parallel Delay                                                                                                                                                                              | MACMZ: Multiply and Accumulate with Parallel Delay                                                                     |
| ACx = rnd(ACx + (Smem * coef(Cmem)))[, T3 = Smem], delay(Smem)                                                                                                                                                           | MACM[R]Z [T3 = ]Smem, Cmem, ACx                                                                                        |
| Multiply and Accumulate with Parallel Load Accumulator from Memory                                                                                                                                                       | MACM::MOV: Multiply and Accumulate with Parallel Load Accumulator from Memory                                          |
| ACx = rnd(ACx + (Tx * Xmem)),<br>ACy = Ymem << #16 [,T3 = Xmem]                                                                                                                                                          | MACM[R] [T3 = ]Xmem, Tx, ACx<br>:: MOV Ymem << #16, ACy                                                                |
| Multiply and Accumulate with Parallel Multiply                                                                                                                                                                           | MAC::MPY: Multiply and Accumulate with Parallel Multiply                                                               |
| ACx = M40(rnd(ACx + (uns(Xmem) * uns(coef(Cmem))))),<br>ACy = M40(rnd(uns(Ymem) * uns(coef(Cmem))))                                                                                                                      | MAC[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx :: MPY[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy                           |
| ACy = M40(rnd(ACy + (uns(Smem) * uns(HI(coef(Cmem)))))),<br>ACx = M40(rnd(uns(Smem) * uns(LO(coef(Cmem)))))                                                                                                              | MAC[R][40] [uns(]Smem[)], [uns(]HI(Cmem)[)], ACy :: MPY[R][40] [uns(]Smem[)], [uns(]LO(Cmem)[)], ACx                   |
| ACy = M40(rnd((ACy>>#16) + (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(uns(Smem) * uns(LO(coef(Cmem)))))                                                                                                          | MAC[R][40] [uns(]Smem[)], [uns(]HI(Cmem)[)], ACy>>#16<br>:: MPY[R][40] [uns(]Smem[)], [uns(]LO(Cmem)[)], ACx           |
| $ \begin{aligned} &ACy = \frac{M40(rnd(ACy + (uns(HI(Lmem)) \ ^* \ uns(HI(coef(Cmem))))))}{ACx = \frac{M40(rnd(uns(LO(Lmem)) \ ^* \ uns(LO(coef(Cmem)))))}{Model(LO(Lmem)) \ ^* \ uns(LO(coef(Lmem)))))} \end{aligned} $ | MAC[R][40] [uns(]HI(Lmem)[)], [uns(]HI(Cmem)[)], ACy :: MPY[R][40] [uns(]LO(Lmem)[)], [uns(]LO(Cmem)[)], ACx           |
| ACy = M40(rnd((ACy>>#16) + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))),<br>ACx = M40(rnd(uns(LO(Lmem)) * uns(LO(coef(Cmem)))))                                                                                               | MAC[R][40] [uns(]HI(Lmem)[)], [uns(]HI(Cmem)[)], ACy>>#16<br>:: MPY[R][40] [uns(]LO(Lmem)[)], [uns(]LO(Cmem)[)], ACx   |
| ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(HI(coef(Cmem)))))),<br>ACx = M40(rnd(uns(Xmem) * uns(LO(coef(Cmem)))))                                                                                                     | MAC[R][40] [uns(]HI(Ymem)[)], [uns(]HI(Cmem)[)], ACy >> #16<br>:: MPY[R][40] [uns(]LO(Xmem)[)], [uns(]LO(Cmem)[)], ACx |

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

| Algebraic Syntax                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Mnemonic Syntax                                                                                                      |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------|
| Multiply and Accumulate with Parallel Multiply and Subtract                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | MAC::MAS: Multiply and Accumulate with Parallel Multiply and Subtract                                                |
| $ ACy = \frac{M40(rnd(ACy + (uns(Smem) * uns(HI(coef(Cmem))))))}{ACx = \frac{M40(rnd(ACx - (uns(Smem) * uns(LO(coef(Cmem))))))}{ACx = \frac{M40(rnd(ACy + (uns(Smem) * uns(LO(coef(Cmem))))))}{ACy = \frac{M40(rnd(ACy + (uns(Smem) * uns(HI(coef(Cmem))))))}{ACy = \frac{M40(rnd(ACy + (uns(Smem) * uns(HI(coef(Cmem)))))}{ACy = \frac{M40(rnd(ACy + (uns(Smem) * uns(HI(coef(Cmem)))))}{ACy = M40(rnd(ACy + (uns(Uns(Uns(Uns(Uns(Uns(Uns(Uns(Uns(Uns(U$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | MAC[R][40] [uns(]Smem[)], [uns(]HI(Cmem)[)], ACy :: MAS[R][40] [uns(]Smem[)], [uns(]LO(Cmem)[)], ACx                 |
| ACy = M40(rnd((ACy>>#16) + (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx - (uns(Smem) * uns(LO(coef(Cmem))))))                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | MAC[R][40] [uns(]Smem[)], [uns(]HI(Cmem)[)], ACy>>#16 :: MAS[R][40] [uns(]Smem[)], [uns(]LO(Cmem)[)], ACx            |
| $ ACy = \frac{M40(\text{rnd}(ACy + (\text{uns}(HI(Lmem)) * uns}(HI(\text{coef}(Cmem))))))}{ACx = \frac{M40(\text{rnd}(ACx - (\text{uns}(LO(Lmem)) * uns}(LO(\text{coef}(Cmem))))))}{ACx = \frac{M40(\text{rnd}(ACx - (\text{uns}(LD(Lmem)) * uns}(LO(\text{coef}(Cmem))))))}{ACx = \frac{M40(\text{rnd}(ACx - (\text{uns}(LD(Lmem)) * uns}(LD(Lmem))))}{ACx = \frac{M40(\text{rnd}(ACx - (\text{uns}(LD(Lmem)) * uns}(LD(Lmem)))}{ACx = \frac{M40(\text{rnd}(ACx - (\text{uns}(LD$ | MAC[R][40] [uns(]HI(Lmem)[)], [uns(]HI(Cmem)[)], ACy :: MAS[R][40] [uns(]LO(Lmem)[)], [uns(]LO(Cmem)[)], ACx         |
| $ACy = \frac{M40(rnd((ACy>>#16) + (uns(HI(Lmem)) * uns(HI(coef(Cmem))))))}{uns(HI(coef(Cmem)))))},$ $ACx = \frac{M40(rnd(ACx - (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))}{uns(LO(coef(Cmem)))))}$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | MAC[R][40] [uns(]HI(Lmem)[)], [uns(]HI(Cmem)[)], ACy>>#16<br>:: MAS[R][40] [uns(]LO(Lmem)[)], [uns(]LO(Cmem)[)], ACx |
| $ACy = \frac{M40(\text{rnd}(ACy + \text{uns}(Ymem) * \text{uns}(HI(coef(Cmem)))))}{ACx = \frac{M40(\text{rnd}(ACx - \text{uns}(Xmem) * \text{uns}(LO(coef(Cmem)))))}}$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | MAC[R][40] [uns(]Ymem[)], [uns(]HI(Cmem)[)], ACy, :: MAS[R][40] [uns(]Xmem[)], [uns(]LO(Cmem)[)], ACx                |
| $ACy = \frac{M40(rnd((ACy >> #16) + (uns(Ymem) * uns(Hl(coef(Cmem))))))}{uns(Hl(coef(Cmem))))))},$ $ACx = \frac{M40(rnd(ACx - (uns(Xmem) * uns(LO(coef(Cmem))))))}{uns(Hl(coef(Cmem)))))}$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | MAC[R][40] [uns(]HI(Ymem)[)], [uns(]HI(Cmem)[)], ACy >> #16, :: MAS[R][40] [uns(]LO(Xmem)[)], [uns(]LO(Cmem)[)], ACx |
| Multiply and Accumulate with Parallel Store Accumulator Content to Memory                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | MACM::MOV: Multiply and Accumulate with Parallel Store Accumulator Content to Memory                                 |
| ACy = rnd(ACy + (Tx * Xmem)),<br>Ymem = HI(ACx << T2) [,T3 = Xmem]                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | MACM[R] [T3 = ]Xmem, Tx, ACy<br>:: MOV HI(ACx << T2), Ymem                                                           |
| Multiply and Subtract                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | MAS: Multiply and Subtract                                                                                           |
| $ACy = \frac{rnd(ACy - (ACx * Tx))}{rnd(ACy - (ACx * Tx))}$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | MAS[R] Tx, [ACx,] ACy                                                                                                |
| ACx = rnd(ACx - (Smem * uns(coef(Cmem))))                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | MAS[R] Smem, uns(Cmem), ACx                                                                                          |
| ACx = rnd(ACx - (Smem * coef(Cmem)))[, T3 = Smem]                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | MASM[R] [T3 = ]Smem, Cmem, ACx                                                                                       |
| ACy = rnd(ACy - (Smem * ACx))[, T3 = Smem]                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | MASM[R] [T3 = ]Smem, [ACx,] ACy                                                                                      |

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

| Algebraic Syntax                                                                                                                                                                                  | Mnemonic Syntax                                                                                              |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------|
| ACy = rnd(ACx - (Tx * Smem))[, T3 = Smem]                                                                                                                                                         | MASM[R] [T3 = ]Smem, Tx, [ACx,] ACy                                                                          |
| ACy = M40(rnd(ACx - (uns(Xmem) * uns(Ymem))))[, T3 = Xmem]                                                                                                                                        | MASM[R][40] [T3 = ][uns(]Xmem[)], [uns(]Ymem[)], [ACx,] ACy                                                  |
| Multiply and Subtract with Parallel Load Accumulator from Memory                                                                                                                                  | MASM::MOV: Multiply and Subtract with Parallel Load Accumulator from Memory                                  |
| $ACx = \frac{\text{rnd}(ACx - (Tx * Xmem))}{ACy},$ $ACy = \frac{\text{Ymem}}{\text{Xmem}}$                                                                                                        | MASM[R] [T3 = ]Xmem, Tx, ACx<br>:: MOV Ymem << #16, ACy                                                      |
| Multiply and Subtract with Parallel Multiply                                                                                                                                                      | MAS::MPY: Multiply and Subtract with Parallel Multiply                                                       |
| ACx = M40(rnd(ACx - (uns(Xmem) * uns(coef(Cmem))))),<br>ACy = M40(rnd(uns(Ymem) * uns(coef(Cmem))))                                                                                               | MAS[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx :: MPY[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy                 |
| ACy = M40(rnd(ACy - (uns(Smem) * uns(HI(coef(Cmem)))))),<br>ACx = M40(rnd(uns(Smem) * uns(LO(coef(Cmem)))))                                                                                       | MAS[R][40] [uns(]Smem[)], [uns(]HI(Cmem)[)], ACy :: MPY[R][40] [uns(]Smem[)], [uns(]LO(Cmem)[)], ACx         |
| $ \begin{aligned} &ACy = \frac{M40(rnd(ACy - (uns(HI(Lmem)) * uns(HI(coef(Cmem))))))}{ACx = \frac{M40(rnd(uns(LO(Lmem)) * uns(LO(coef(Cmem)))))}{Not(LO(Lmem)) * uns(LO(Lmem))))} \end{aligned} $ | MAS[R][40] [uns(]HI(Lmem)[)], [uns(]HI(Cmem)[)], ACy :: MPY[R][40] [uns(]LO(Lmem)[)], [uns(]LO(Cmem)[)], ACx |
| Multiply and Subtract with Parallel Multiply and Accumulate                                                                                                                                       | MAS::MAC: Multiply and Subtract with Parallel Multiply and Accumulate                                        |
| ACx = M40(rnd(ACx - (uns(Xmem) * uns(coef(Cmem))))),<br>ACy = M40(rnd(ACy + (uns(Ymem) * uns(coef(Cmem)))))                                                                                       | MAS[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx :: MAC[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy                 |
| $ \begin{aligned} &ACx = \frac{M40(rnd(ACx - (uns(Xmem) * uns(coef(Cmem)))))}{ACy = \frac{M40(rnd((ACy >> \#16) + (uns(Ymem) * uns(coef(Cmem)))))}{NCy} \end{aligned} $                           | MAS[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx<br>:: MAC[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy >> #16       |
| $ \begin{aligned} &ACy = \frac{M40(rnd(ACy - (uns(Smem) * uns(HI(coef(Cmem))))))}{ACx = \frac{M40(rnd(ACx + (uns(Smem) * uns(LO(coef(Cmem))))))}{MSModel} \end{aligned} $                         | MAS[R][40] [uns(]Smem[)], [uns(]HI(Cmem)[)], ACy :: MAC[R][40] [uns(]Smem[)], [uns(]LO(Cmem)[)], ACx         |
| $ \begin{aligned} &ACy = \frac{M40(rnd(ACy - (uns(HI(Lmem)) * uns(HI(coef(Cmem))))))}{ACx = \frac{M40(rnd(ACx + (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))}{Noe(LO(Loef(Lmem))))} \end{aligned} $    | MAS[R][40] [uns(]HI(Lmem)[)], [uns(]HI(Cmem)[)], ACy :: MAC[R][40] [uns(]LO(Lmem)[)], [uns(]LO(Cmem)[)], ACx |

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

| Algebraic Syntax                                                                                            | Mnemonic Syntax                                                                                                 |
|-------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------|
| Multiply and Subtract with Parallel Store Accumulator Content to Memory                                     | MASM::MOV: Multiply and Subtract with Parallel Store Accumulator Content to Memory                              |
| ACy = rnd(ACy - (Tx * Xmem)),<br>Ymem = HI(ACx << T2) [,T3 = Xmem]                                          | MASM[R] [T3 = ]Xmem, Tx, ACy<br>:: MOV HI(ACx << T2), Ymem                                                      |
| Negate Accumulator, Auxiliary, or Temporary Register Content                                                | NEG: Negate Accumulator, Auxiliary, or Temporary Registe Content                                                |
| dst = -src                                                                                                  | NEG [src,] dst                                                                                                  |
| No Operation                                                                                                | NOP: No Operation                                                                                               |
| пор                                                                                                         | NOP                                                                                                             |
| nop_16                                                                                                      | NOP_16                                                                                                          |
| Parallel Modify Auxiliary Register Contents                                                                 | AMAR: Parallel Modify Auxiliary Register Contents                                                               |
| mar(Xmem), mar(Ymem), mar(coef(Cmem))                                                                       | AMAR Xmem, Ymem, Cmem                                                                                           |
| Parallel Multiplies                                                                                         | MPY::MPY: Parallel Multiplies                                                                                   |
| ACx = M40(rnd(uns(Xmem) * uns(coef(Cmem)))),<br>ACy = M40(rnd(uns(Ymem) * uns(coef(Cmem))))                 | MPY[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx :: MPY[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy                    |
| ACy = M40(rnd(uns(Smem) * uns(HI(coef(Cmem))))),<br>ACx = M40(rnd(uns(Smem) * uns(LO(coef(Cmem)))))         | MPY[R][40] [uns(]Smem[)], [uns(]HI(Cmem)[)], ACy :: MPY[R][40] [uns(]Smem[)], [uns(]LO(Cmem)[)], ACx            |
| ACy = M40(rnd(uns(HI(Lmem)) * uns(HI(coef(Cmem))))),<br>ACx = M40(rnd(uns(LO(Lmem)) * uns(LO(coef(Cmem))))) | MPY[R][40] [uns(]HI(Lmem)[)], [uns(]HI(Cmem)[)], ACy<br>:: MPY[R][40] [uns(]LO(Lmem)[)], [uns(]LO(Cmem)[)], ACx |
| ACy = M40(rnd(uns(Ymem) * uns(HI(coef(Cmem))))),<br>ACx = M40(rnd(uns(Xmem) * uns(LO(coef(Cmem)))))         | MPY[R][40] [uns(]Ymem[)], [uns(]HI(Cmem)[)], ACy,<br>:: MPY[R][40] [uns(]Xmem[)], [uns(]LO(Cmem)[)], ACx        |

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

| Algebraic Syntax                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Mnemonic Syntax                                                                                                           |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------|
| Parallel Multiply and Accumulates                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | MAC::MAC: Parallel Multiply and Accumulates                                                                               |
| ACx = M40(rnd(ACx + (uns(Xmem) * uns(coef(Cmem))))),<br>ACy = M40(rnd(ACy + (uns(Ymem) * uns(coef(Cmem)))))                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | MAC[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx :: MAC[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy                              |
| $ ACx = \frac{M40(rnd((ACx >> #16) + (uns(Xmem) * uns(coef(Cmem)))))}{ACy = \frac{M4(rnd(ACy + (uns(Ymem) * uns(coef(Cmem)))))}{ACy + (uns(Xmem) * uns(coef(Cmem))))}  $                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | MAC[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx >> #16<br>:: MAC[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy                    |
| $ACx = \frac{M40(rnd((ACx >> #16) + (uns(Xmem) * uns(coef(Cmem)))))}{ACy = \frac{M40(rnd((ACy >> #16) + (uns(Ymem) * uns(coef(Cmem)))))}{ACy = \frac{M40(rnd((ACy >> #16) + (uns(Cmem) * uns(coef(Cmem)))))}{ACy = \frac{M40(rnd((ACy >> #16) + (uns(Cmem) * uns(coef(Cmem))))}{ACy = $ | MAC[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx >> #16<br>:: MAC[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy >> #16             |
| $ACy = \frac{M40(\text{rnd}(ACy + (\text{uns}(Smem) * \text{uns}(HI(coef(Cmem))))))}{ACx = \frac{M40(\text{rnd}(ACx + (\text{uns}(Smem) * \text{uns}(LO(coef(Cmem))))))}}$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | MAC[R][40] [uns(]Smem[)], [uns(]HI(Cmem)[)], ACy :: MAC[R][40] [uns(]Smem[)], [uns(]LO(Cmem)[)], ACx                      |
| $ACy = \frac{M40(\text{rnd}(ACy + (\text{uns}(Smem) * \text{uns}(HI(coef(Cmem))))))}{ACx = \frac{M40(\text{rnd}((ACx>>#16) + (\text{uns}(Smem) * \text{uns}(LO(coef(Cmem))))))}}$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | MAC[R][40] [uns(]Smem[)], [uns(]HI(Cmem)[)], ACy<br>:: MAC[R][40] [uns(]Smem[)], [uns(]LO(Cmem)[)], ACx>>#16              |
| ACy = M40(rnd((ACy>>#16) + (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd((ACx>>#16) + (uns(Smem) * uns(LO(coef(Cmem))))))                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | MAC[R][40] [uns(]Smem[)], [uns(]HI(Cmem)[)], ACy>>#16<br>:: MAC[R][40] [uns(]Smem[)], [uns(]LO(Cmem)[)], ACx>>#16         |
| $ ACy = \frac{M40(rnd(ACy + (uns(HI(Lmem)) * uns(HI(coef(Cmem))))))}{ACx = \frac{M40(rnd(ACx + (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))}{ACx + (uns(LO(Lmem)) * uns(LO(coef(Cmem)))))}  $                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | MAC[R][40] [uns(]HI(Lmem)[)], [uns(]HI(Cmem)[)], ACy :: MAC[R][40] [uns(]LO(Lmem)[)], [uns(]LO(Cmem)[)], ACx              |
| $\label{eq:acy} \begin{split} ACy &= \text{M40(rnd(ACy + (uns(HI(Lmem)) * uns(HI(coef(Cmem))))))}, \\ ACx &= \text{M40(rnd((ACx)>#16) + (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))} \end{split}$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | MAC[R][40] [uns(]HI(Lmem)[)], [uns(]HI(Cmem)[)], ACy<br>:: MAC[R][40] [uns(]LO(Lmem)[)], [uns(]LO(Cmem)[)], ACx>>#16      |
| ACy = M40(rnd((ACy>>#16) + (uns(HI(Lmem)) * uns(HI(coef(Cmem))))))), ACx = M40(rnd((ACx>>#16) + (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | MAC[R][40] [uns(]HI(Lmem)[)], [uns(]HI(Cmem)[)], ACy>>#16<br>:: MAC[R][40] [uns(]LO(Lmem)[)], [uns(]LO(Cmem)[)], ACx>>#16 |
| ACy = M40(rnd(ACy + uns(Ymem) * uns(HI(coef(Cmem))))),<br>ACx = M40(rnd(ACx + uns(Xmem) * uns(LO(coef(Cmem)))))                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | MAC[R][40] [uns(]Ymem[)], [uns(]HI(Cmem)[)], ACy, :: MAC[R][40] [uns(]Xmem[)], [uns(]LO(Cmem)[)], ACx                     |

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

| Algebraic Syntax                                                                                                                                                                   | Mnemonic Syntax                                                                                                             |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| ACy = M40(rnd(ACy + (uns(Ymem) * uns(HI(coef(Cmem)))))),<br>ACx = M40(rnd((ACx >> #16) + (uns(Xmem) *<br>uns(LO(coef(Cmem))))))                                                    | MAC[R][40] [uns(]HI(Ymem)[)], [uns(]HI(Cmem)[)], ACy,<br>:: MAC[R][40] [uns(]LO(Xmem)[)], [uns(]LO(Cmem)[)], ACx >> #16     |
| ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(HI(coef(Cmem)))))),<br>ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(LO(coef(Cmem))))))                                              | MAC[R][40] [uns(]HI(Ymem)[)], [uns(]HI(Cmem)[)], ACy >> #16, :: MAC[R][40] [uns(]LO(Xmem)[)], [uns(]LO(Cmem)[)], ACx >> #16 |
| Parallel Multiply and Subtracts                                                                                                                                                    | MAS::MAS: Parallel Multiply and Subtracts                                                                                   |
| ACx = M40(rnd(ACx - (uns(Xmem) * uns(coef(Cmem))))),<br>ACy = M40(rnd(ACy - (uns(Ymem) * uns(coef(Cmem)))))                                                                        | MAS[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx :: MAS[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy                                |
| $ACy = \frac{M40(\text{rnd}(ACy - (\text{uns}(Smem) * \text{uns}(HI(coef(Cmem))))))}{ACx = \frac{M40(\text{rnd}(ACx - (\text{uns}(Smem) * \text{uns}(LO(coef(Cmem))))))}}$         | MAS[R][40] [uns(]Smem[)], [uns(]HI(Cmem)[)], ACy :: MAS[R][40] [uns(]Smem[)], [uns(]LO(Cmem)[)], ACx                        |
| $ACy = \frac{M40(\text{rnd}(ACy - (\text{uns}(HI(Lmem)) * \text{uns}(HI(coef(Cmem))))))}{ACx = \frac{M40(\text{rnd}(ACx - (\text{uns}(LO(Lmem)) * \text{uns}(LO(coef(Cmem))))))}}$ | MAS[R][40] [uns(]HI(Lmem)[)], [uns(]HI(Cmem)[)], ACy<br>:: MAS[R][40] [uns(]LO(Lmem)[)], [uns(]LO(Cmem)[)], ACx             |
| $ACy = \frac{M40(\text{rnd}(ACy - (\text{uns}(Ymem) * \text{uns}(HI(coef(Cmem))))))}{ACx} = \frac{M40(\text{rnd}(ACx - (\text{uns}(Xmem) * \text{uns}(LO(coef(Cmem))))))}{ACx}$    | MAS[R][40] [uns(]HI(Ymem)[)], [uns(]HI(Cmem)[)], ACy, :: MAS[R][40] [uns(]LO(Xmem)[)], [uns(]LO(Cmem)[)], ACx               |
| Peripheral Port Register Access Qualifiers                                                                                                                                         | port: Peripheral Port Register Access Qualifiers                                                                            |
| readport()                                                                                                                                                                         | port(Smem)                                                                                                                  |
| writeport()                                                                                                                                                                        | port(Smem)                                                                                                                  |
| Pop Accumulator or Extended Auxiliary Register Content from Stack Pointers                                                                                                         | POPBOTH: Pop Accumulator or Extended Auxiliary Register Content from Stack Pointers                                         |
| xdst = popboth()                                                                                                                                                                   | POPBOTH xdst                                                                                                                |

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

| Algebraic Syntax                                                          | Mnemonic Syntax                                                                   |
|---------------------------------------------------------------------------|-----------------------------------------------------------------------------------|
| Pop Top of Stack                                                          | POP: Pop Top of Stack                                                             |
| dst1, dst2 = pop()                                                        | POP dst1, dst2                                                                    |
| dst = pop()                                                               | POP dst                                                                           |
| dst, Smem = pop()                                                         | POP dst, Smem                                                                     |
| ACx = dbl(pop())                                                          | POP ACx                                                                           |
| Smem = pop()                                                              | POP Smem                                                                          |
| dbl(Lmem) = pop()                                                         | POP dbl(Lmem)                                                                     |
| Push Accumulator or Extended Auxiliary Register Content to Stack Pointers | PSHBOTH: Push Accumulator or Extended Auxiliary Registe Content to Stack Pointers |
| pshboth(xsrc)                                                             | PSHBOTH xsrc                                                                      |
| Push to Top of Stack                                                      | PSH: Push to Top of Stack                                                         |
| push(src1, src2)                                                          | PSH src1, src2                                                                    |
| push(src)                                                                 | PSH src                                                                           |
| push(src, Smem)                                                           | PSH src, Smem                                                                     |
| dbl(push(ACx))                                                            | PSH ACx                                                                           |
| push(Smem)                                                                | PSH Smem                                                                          |
| push(dbl(Lmem))                                                           | PSH dbl(Lmem)                                                                     |
| Repeat Block of Instructions Unconditionally                              | RPTB: Repeat Block of Instructions Unconditionally                                |
| localrepeat{ }                                                            | RPTBLOCAL pmad                                                                    |
| blockrepeat{ }                                                            | RPTB pmad                                                                         |

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

| Algebraic Syntax                                            | Mnemonic Syntax                                                     |
|-------------------------------------------------------------|---------------------------------------------------------------------|
| Repeat Single Instruction Conditionally                     | RPTCC: Repeat Single Instruction Conditionally                      |
| while (cond && (RPTC < k8)) repeat                          | RPTCC k8, cond                                                      |
| Repeat Single Instruction Unconditionally                   | RPT: Repeat Single Instruction Unconditionally                      |
| repeat(k8)                                                  | RPT k8                                                              |
| repeat(k16)                                                 | RPT k16                                                             |
| repeat(CSR)                                                 | RPT CSR                                                             |
| Repeat Single Instruction Unconditionally and Decrement CSR | RPTSUB: Repeat Single Instruction Unconditionally and Decrement CSR |
| repeat(CSR), CSR -= k4                                      | RPTSUB CSR, k4                                                      |
| Repeat Single Instruction Unconditionally and Increment CSR | RPTADD: Repeat Single Instruction Unconditionally and Increment CSR |
| repeat(CSR), CSR += TAx                                     | RPTADD CSR, TAx                                                     |
| repeat(CSR), CSR += k4                                      | RPTADD CSR, k4                                                      |
| Return Conditionally                                        | RETCC: Return Conditionally                                         |
| f (cond) return                                             | RETCC cond                                                          |
| Return Unconditionally                                      | RET: Return Unconditionally                                         |
| return                                                      | RET                                                                 |
| Return from Interrupt                                       | RETI: Return from Interrupt                                         |
| return_int                                                  | RETI                                                                |

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

| Algebraic Syntax                                                   | Mnemonic Syntax                                                         |
|--------------------------------------------------------------------|-------------------------------------------------------------------------|
| Rotate Left Accumulator, Auxiliary, or Temporary Register Content  | ROL: Rotate Left Accumulator, Auxiliary, or Temporary Register Content  |
| dst = BitOut \\ src \\ BitIn                                       | ROL BitOut, src, BitIn, dst                                             |
| Rotate Right Accumulator, Auxiliary, or Temporary Register Content | ROR: Rotate Right Accumulator, Auxiliary, or Temporary Register Content |
| dst = BitIn // src // BitOut                                       | ROR BitIn, src, BitOut, dst                                             |
| Round Accumulator Content                                          | ROUND: Round Accumulator Content                                        |
| ACy = rnd(ACx)                                                     | ROUND [ACx,] ACy                                                        |
| Saturate Accumulator Content                                       | SAT: Saturate Accumulator Content                                       |
| ACy = saturate(rnd(ACx))                                           | SAT[R] [ACx,] ACy                                                       |
| Set Accumulator, Auxiliary, or Temporary Register Bit              | BSET: Set Accumulator, Auxiliary, or Temporary Register Bit             |
| bit(src, Baddr) = #1                                               | BSET Baddr, src                                                         |
| Set Memory Bit                                                     | BSET: Set Memory Bit                                                    |
| bit(Smem, src) = #1                                                | BSET src, Smem                                                          |
| Set Status Register Bit                                            | BSET: Set Status Register Bit                                           |
| bit(STx, k4) = #1                                                  | BSET k4, STx_55                                                         |
|                                                                    | BSET f-name                                                             |
| Shift Accumulator Content Conditionally                            | SFTCC: Shift Accumulator Content Conditionally                          |
| ACx = sftc(ACx, TCx)                                               | SFTCC ACx, TCx                                                          |
|                                                                    |                                                                         |

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

| Algebraic Syntax                                                         | Mnemonic Syntax                                                             |
|--------------------------------------------------------------------------|-----------------------------------------------------------------------------|
| Shift Accumulator Content Logically                                      | SFTL: Shift Accumulator Content Logically                                   |
| $ACy = ACx \ll Tx$                                                       | SFTL ACx, Tx[, ACy]                                                         |
| ACy = ACx <<< #SHIFTW                                                    | SFTL ACx, #SHIFTW[, ACy]                                                    |
| Shift Accumulator, Auxiliary, or Temporary Register Content<br>Logically | SFTL: Shift Accumulator, Auxiliary, or Temporary Register Content Logically |
| dst = dst <<< #1                                                         | SFTL dst, #1                                                                |
| dst = dst >>> #1                                                         | SFTL dst, #-1                                                               |
| Signed Shift of Accumulator Content                                      | SFTS: Signed Shift of Accumulator Content                                   |
| $ACy = ACx \ll Tx$                                                       | SFTS ACx, Tx[, ACy]                                                         |
| ACy = ACx << #SHIFTW                                                     | SFTS ACx, #SHIFTW[, ACy]                                                    |
| ACy = ACx << CTx                                                         | SFTSC ACx, Tx[, ACy]                                                        |
| ACy = ACx < <c #shiftw<="" td=""><td>SFTSC ACx, #SHIFTW[, ACy]</td></c>  | SFTSC ACx, #SHIFTW[, ACy]                                                   |
| Signed Shift of Accumulator, Auxiliary, or Temporary Register Content    | SFTS: Signed Shift of Accumulator, Auxiliary, or Temporary Register Content |
| dst = dst >> #1                                                          | SFTS dst, #-1                                                               |
| dst = dst << #1                                                          | SFTS dst, #1                                                                |
| Software Interrupt                                                       | INTR: Software Interrupt                                                    |
| intr(k5)                                                                 | INTR k5                                                                     |
| Software Reset                                                           | RESET: Software Reset                                                       |
| reset                                                                    | RESET                                                                       |

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

| Algebraic Syntax                                              | Mnemonic Syntax                          |
|---------------------------------------------------------------|------------------------------------------|
| Software Trap                                                 | TRAP: Software Trap                      |
| trap(k5)                                                      | TRAP k5                                  |
| Square                                                        | SQR: Square                              |
| $ACy = \frac{rnd(ACx * ACx)}{rnd(ACx * ACx)}$                 | SQR[R] [ACx,] ACy                        |
| ACx = rnd(Smem * Smem)[, T3 = Smem]                           | SQRM[R] [T3 = ]Smem, ACx                 |
| Square and Accumulate                                         | SQA: Square and Accumulate               |
| ACy = rnd(ACy + (ACx * ACx))                                  | SQA[R] [ACx,] ACy                        |
| ACy = rnd(ACx + (Smem * Smem))[, T3 = Smem]                   | SQAM[R] [T3 = ]Smem, [ACx,] ACy          |
| Square and Subtract                                           | SQS: Square and Subtract                 |
| ACy = rnd(ACy - (ACx * ACx))                                  | SQS[R] [ACx,] ACy                        |
| $ACy = \frac{\text{rnd}(ACx - (Smem * Smem))[, T3 = Smem]}{}$ | SQSM[R] [T3 = ]Smem, [ACx,] ACy          |
| Square Distance                                               | SQDST: Square Distance                   |
| sqdst(Xmem, Ymem, ACx, ACy)                                   | SQDST Xmem, Ymem, ACx, ACy               |
| Store Accumulator Content to Memory                           | MOV: Store Accumulator Content to Memory |
| Smem = HI(ACx)                                                | MOV HI(ACx), Smem                        |
| Smem = HI(rnd(ACx))                                           | MOV [rnd(]HI(ACx)[)], Smem               |
| Smem = LO(ACx << Tx)                                          | MOV ACx << Tx, Smem                      |
| Smem = HI(rnd(ACx << Tx))                                     | MOV [rnd(]HI(ACx << Tx)[)], Smem         |
| Smem = LO(ACx << #SHIFTW)                                     | MOV ACx << #SHIFTW, Smem                 |
| Smem = HI(ACx << #SHIFTW)                                     | MOV HI(ACx << #SHIFTW), Smem             |

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

| Algebraic Syntax                                                      | Mnemonic Syntax                                                            |
|-----------------------------------------------------------------------|----------------------------------------------------------------------------|
| Smem = HI(rnd(ACx << #SHIFTW))                                        | MOV [rnd(]HI(ACx << #SHIFTW)[)], Smem                                      |
| Smem = HI(saturate(uns(rnd(ACx))))                                    | MOV [uns(] [rnd(]HI[(saturate](ACx)[)))], Smem                             |
| Smem = HI(saturate(uns(rnd(ACx << Tx))))                              | MOV [uns(] [rnd(]HI[(saturate](ACx << Tx)[)))], Smem                       |
| Smem = HI(saturate(uns(rnd(ACx << #SHIFTW))))                         | MOV [uns(] [rnd(]HI[(saturate](ACx << #SHIFTW)[)))], Smem                  |
| dbl(Lmem) = ACx                                                       | MOV ACx, dbl(Lmem)                                                         |
| dbl(Lmem) = saturate(uns(ACx))                                        | MOV [uns(]saturate(ACx)[)], dbl(Lmem)                                      |
| HI(Lmem) = HI(ACx) >> #1,<br>LO(Lmem) = LO(ACx) >> #1                 | MOV ACx >> #1, dual(Lmem)                                                  |
| Xmem = LO(ACx),<br>Ymem = HI(ACx)                                     | MOV ACx, Xmem, Ymem                                                        |
| Store Accumulator Pair Content to Memory                              | MOV: Store Accumulator Pair Content to Memory                              |
| Lmem = pair(HI(ACx))                                                  | MOV pair(HI(ACx)), dbl(Lmem)                                               |
| Lmem = pair(LO(ACx))                                                  | MOV pair(LO(ACx)), dbl(Lmem)                                               |
| Store Accumulator, Auxiliary, or Temporary Register Content to Memory | MOV: Store Accumulator, Auxiliary, or Temporary Register Content to Memory |
| Smem = src                                                            | MOV src, Smem                                                              |
| high_byte(Smem) = src                                                 | MOV src, high_byte(Smem)                                                   |
| low_byte(Smem) = src                                                  | MOV src, low_byte(Smem)                                                    |
| Store Auxiliary or Temporary Register Pair Content to Memory          | MOV: Store Auxiliary or Temporary Register Pair Content to Memory          |
| Lmem = pair(TAx)                                                      | MOV pair(TAx), dbl(Lmem)                                                   |

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

| Algebraic Syntax                     | Mnemonic Syntax                           |
|--------------------------------------|-------------------------------------------|
| Store CPU Register Content to Memory | MOV: Store CPU Register Content to Memory |
| Smem = BK03                          | MOV BK03, Smem                            |
| Smem = BK47                          | MOV BK47, Smem                            |
| Smem = BKC                           | MOV BKC, Smem                             |
| Smem = BSA01                         | MOV BSA01, Smem                           |
| Smem = BSA23                         | MOV BSA23, Smem                           |
| Smem = BSA45                         | MOV BSA45, Smem                           |
| Smem = BSA67                         | MOV BSA67, Smem                           |
| Smem = BSAC                          | MOV BSAC, Smem                            |
| Smem = BRC0                          | MOV BRC0, Smem                            |
| Smem = BRC1                          | MOV BRC1, Smem                            |
| Smem = CDP                           | MOV CDP, Smem                             |
| Smem = CSR                           | MOV CSR, Smem                             |
| Smem = DP                            | MOV DP, Smem                              |
| Smem = DPH                           | MOV DPH, Smem                             |
| Smem = PDP                           | MOV PDP, Smem                             |
| Smem = SP                            | MOV SP, Smem                              |
| Smem = SSP                           | MOV SSP, Smem                             |
| Smem = TRN0                          | MOV TRN0, Smem                            |
| Smem = TRN1                          | MOV TRN1, Smem                            |
| dbl(Lmem) = RETA                     | MOV RETA, dbl(Lmem)                       |

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

| Algebraic Syntax                                    | Mnemonic Syntax                                          |
|-----------------------------------------------------|----------------------------------------------------------|
| Store Extended Auxiliary Register Content to Memory | MOV: Store Extended Auxiliary Register Content to Memory |
| dbl(Lmem) = XAsrc                                   | MOV XAsrc, dbl(Lmem)                                     |
| Subtract Conditionally                              | SUBC: Subtract Conditionally                             |
| subc(Smem, ACx, ACy)                                | SUBC Smem, [ACx,] ACy                                    |
| Subtraction                                         | SUB: Subtraction                                         |
| dst = dst - src                                     | SUB [src,] dst                                           |
| dst = dst - k4                                      | SUB k4, dst                                              |
| dst = src - K16                                     | SUB K16, [src,] dst                                      |
| dst = src - Smem                                    | SUB Smem, [src,] dst                                     |
| dst = Smem - src                                    | SUB src, Smem, dst                                       |
| ACy = ACy - (ACx << Tx)                             | SUB ACx << Tx, ACy                                       |
| $ACy = ACy - (ACx \ll \#SHIFTW)$                    | SUB ACx << #SHIFTW, ACy                                  |
| ACy = ACx - (K16 << #16)                            | SUB K16 << #16, [ACx,] ACy                               |
| $ACy = ACx - (K16 \ll \#SHFT)$                      | SUB K16 << #SHFT, [ACx,] ACy                             |
| ACy = ACx - (Smem << Tx)                            | SUB Smem << Tx, [ACx,] ACy                               |
| ACy = ACx - (Smem << #16)                           | SUB Smem << #16, [ACx,] ACy                              |
| ACy = (Smem << #16) - ACx                           | SUB ACx, Smem << #16, ACy                                |
| ACy = ACx - uns(Smem) - BORROW                      | SUB [uns(]Smem[)], BORROW, [ACx,] ACy                    |
| $ACy = ACx - \frac{uns(Smem)}{}$                    | SUB [uns(]Smem[)], [ACx,] ACy                            |
| ACy = ACx - (uns(Smem) << #SHIFTW)                  | SUB [uns(]Smem[)] << #SHIFTW, [ACx,] ACy                 |
| ACy = ACx - dbl(Lmem)                               | SUB dbl(Lmem), [ACx,] ACy                                |
| ACy = dbl(Lmem) - ACx                               | SUB ACx, dbl(Lmem), ACy                                  |

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

| Algebraic Syntax                                              | Mnemonic Syntax                                                         |
|---------------------------------------------------------------|-------------------------------------------------------------------------|
| ACx = (Xmem << #16) - (Ymem << #16)                           | SUB Xmem, Ymem, ACx                                                     |
| Subtraction with Parallel Store Accumulator Content to Memory | SUB::MOV: Subtraction with Parallel Store Accumulator Content to Memory |
| ACy = (Xmem << #16) – ACx,<br>Ymem = HI(ACy << T2)            | SUB Xmem << #16, ACx, ACy<br>:: MOV HI(ACy << T2), Ymem                 |
| Swap Accumulator Content                                      | SWAP: Swap Accumulator Content                                          |
| swap(ACx, ACy)                                                | SWAP ACx, ACy                                                           |
| Swap Accumulator Pair Content                                 | SWAPP: Swap Accumulator Pair Content                                    |
| swap(pair(AC0), pair(AC2))                                    | SWAPP AC0, AC2                                                          |
| Swap Auxiliary Register Content                               | SWAP: Swap Auxiliary Register Content                                   |
| swap(ARx, ARy)                                                | SWAP ARx, ARy                                                           |
| Swap Auxiliary Register Pair Content                          | SWAPP: Swap Auxiliary Register Pair Content                             |
| swap(pair(AR0), pair(AR2))                                    | SWAPP AR0, AR2                                                          |
| Swap Auxiliary and Temporary Register Content                 | SWAP: Swap Auxiliary and Temporary Register Content                     |
| swap(ARx, Tx)                                                 | SWAP ARx, Tx                                                            |
| Swap Auxiliary and Temporary Register Pair Content            | SWAPP: Swap Auxiliary and Temporary Register Pair Content               |
| swap(pair(ARx), pair(Tx))                                     | SWAPP ARx, Tx                                                           |
| Swap Auxiliary and Temporary Register Pairs Content           | SWAP4: Swap Auxiliary and Temporary Register Pairs Conten               |
| swap(block(AR4), block(T0))                                   | SWAP4 AR4, T0                                                           |

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

| Algebraic Syntax                                            | Mnemonic Syntax                                              |
|-------------------------------------------------------------|--------------------------------------------------------------|
| Swap Temporary Register Content                             | SWAP: Swap Temporary Register Content                        |
| swap(Tx, Ty)                                                | SWAP Tx, Ty                                                  |
| Swap Temporary Register Pair Content                        | SWAPP: Swap Temporary Register Pair Content                  |
| swap(pair(T0), pair(T2))                                    | SWAPP T0, T2                                                 |
| Test Accumulator, Auxiliary, or Temporary Register Bit      | BTST: Test Accumulator, Auxiliary, or Temporary Register Bit |
| TCx = bit(src, Baddr)                                       | BTST Baddr, src, TCx                                         |
| Test Accumulator, Auxiliary, or Temporary Register Bit Pair | BTSTP: Test Accumulator, Auxiliary, or Temporary Register Bi |
| bit(src, pair(Baddr))                                       | BTSTP Baddr, src                                             |
| Test Memory Bit                                             | BTST: Test Memory Bit                                        |
| TCx = bit(Smem, src)                                        | BTST src, Smem, TCx                                          |
| TCx = bit(Smem, k4)                                         | BTST k4, Smem, TCx                                           |
| Test and Clear Memory Bit                                   | BTSTCLR: Test and Clear Memory Bit                           |
| TCx = bit(Smem, k4),<br>bit(Smem, k4) = #0                  | BTSTCLR k4, Smem, TCx                                        |
| Test and Complement Memory Bit                              | BTSTNOT: Test and Complement Memory Bit                      |
| TCx = bit(Smem, k4),<br>cbit(Smem, k4)                      | BTSTNOT k4, Smem, TCx                                        |

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

| Algebraic Syntax                           | Mnemonic Syntax                  |
|--------------------------------------------|----------------------------------|
| Test and Set Memory Bit                    | BTSTSET: Test and Set Memory Bit |
| TCx = bit(Smem, k4),<br>bit(Smem, k4) = #1 | BTSTSET k4, Smem, TCx            |

## Index

| abdst 5-2 absolute addressing modes 3-3 I/O absolute 3-3 k16 absolute 3-3 k23 absolute 3-3 Absolute Distance (abdst) 5-2 Absolute Value 5-4 Addition 5-7 Addition or Subtraction Conditionally (adsc) 5-31 Addition or Subtraction Conditionally with Shift (ads2c) 5-33 Addition with Absolute Value 5-27 Addition with Parallel Store Accumulator Content to Memory 5-29 Addition, Subtraction, or Move Accumulator Content Conditionally (adsc) 5-36 addressing modes absolute 3-3 direct 3-4 indirect 3-6 introduction 3-2 ads2c 5-33 | addition or subtraction conditionally with shift 5-33 addition with absolute value 5-27 compare memory with immediate value 5-126 compute exponent of accumulator content 5-131 compute mantissa and exponent of accumulator content 5-132 dual 16-bit addition and subtraction 5-140 dual 16-bit additions 5-135 dual 16-bit subtraction and addition 5-154 dual 16-bit subtractions 5-145 finite impulse response filter, antisymmetrical 5-168 finite impulse response filter, symmetrical 5-170 least mean square 5-173, 5-175 multiply 5-269 multiply and accumulate 5-308 multiply and subtract 5-369 negation 5-403 round accumulator content 5-518 saturate accumulator content 5-520 square 5-557 square and subtract 5-563 square distance 5-566 |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| adsc 5-31, 5-36                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | subtract conditionally 5-601                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| affect of status bits 1-9                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | subtraction 5-603                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| algebraic instruction set cross-reference to mnemonic instruction set 7-1                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | В                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| AND 5-38 Antisymmetrical Finite Impulse Response Filter                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | bit field comparison 5-47                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| (firsn) 5-168                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | bit field counting 5-134                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| arithmetic                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | bit field expand 5-166                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| absolute distance 5-2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | bit field extract 5-167                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| absolute value 5-4                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | bit manipulation                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| addition 5-7 addition or subtraction conditionally 5-31, 5-36                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | bitwise AND memory with immediate value and compare to zero 5-47                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

| clear accumulator, auxiliary, or temporary register                                   | circular 5-87                                                          |
|---------------------------------------------------------------------------------------|------------------------------------------------------------------------|
| bit 5-88                                                                              | circular addressing 3-21                                               |
| clear memory bit 5-89                                                                 | Circular Addressing Qualifier (circular) 5-87                          |
| clear status register bit 5-90                                                        | clear                                                                  |
| complement accumulator, auxiliary, or temporary                                       | accumulator bit 5-88                                                   |
| register bit 5-128                                                                    | auxiliary register bit 5-88                                            |
| complement accumulator, auxiliary, or temporary                                       | memory bit 5-89                                                        |
| register content 5-129                                                                | status register bit 5-90                                               |
| complement memory bit 5-130                                                           | temporary register bit 5-88                                            |
| expand accumulator bit field 5-166                                                    | Clear Accumulator Bit 5-88                                             |
| extract accumulator bit field 5-167 set accumulator, auxiliary, or temporary register | Clear Auxiliary Register Bit 5-88                                      |
| bit 5-522                                                                             | Clear Memory Bit 5-89                                                  |
| set memory bit 5-523                                                                  | •                                                                      |
| set status register bit 5-524                                                         | Clear Status Register Bit 5-90                                         |
| test accumulator, auxiliary, or temporary register                                    | Clear Temporary Register Bit 5-88                                      |
| bit 5-641                                                                             | compare                                                                |
| test accumulator, auxiliary, or temporary register                                    | accumulator, auxiliary, or temporary register                          |
| bit pair 5-643                                                                        | content 5-93                                                           |
| test and clear memory bit 5-648                                                       | accumulator, auxiliary, or temporary register                          |
| test and complement memory bit 5-649                                                  | content maximum 5-105                                                  |
| test and set memory bit 5-650                                                         | accumulator, auxiliary, or temporary register<br>content minimum 5-108 |
| test memory bit 5-645                                                                 | accumulator, auxiliary, or temporary register                          |
| Bitwise AND 5-38                                                                      | content with AND 5-95                                                  |
| Bitwise AND Memory with Immediate Value and                                           | accumulator, auxiliary, or temporary register                          |
| Compare to Zero 5-47                                                                  | content with OR 5-100                                                  |
| bitwise complement 5-129                                                              | and branch 5-111                                                       |
| Bitwise Exclusive OR (XOR) 5-57                                                       | and select accumulator content maximum 5-114                           |
| Bitwise OR 5-48                                                                       | and select accumulator content minimum 5-120                           |
|                                                                                       | memory with immediate value 5-126                                      |
| blockrepeat 5-484                                                                     | Compare Accumulator Content 5-93                                       |
| branch                                                                                | Compare Accumulator Content Maximum                                    |
| conditionally 5-66                                                                    | (max) 5-105                                                            |
| on auxiliary register not zero 5-74 unconditionally 5-70                              | Compare Accumulator Content Minimum                                    |
| -                                                                                     | (min) 5-108                                                            |
| Branch Conditionally (if goto) 5-66                                                   | Compare Accumulator Content with AND 5-95                              |
| Branch on Auxiliary Register Not Zero (if                                             | Compare Accumulator Content with OR 5-100                              |
| goto) 5-74                                                                            | Compare and Branch 5-111                                               |
| Branch Unconditionally (goto) 5-70                                                    |                                                                        |
|                                                                                       | compare and goto 5-111                                                 |
| C                                                                                     | Compare and Select Accumulator Content<br>Maximum (max_diff) 5-114     |
|                                                                                       | Compare and Select Accumulator Content Minimum                         |
| call 5-83                                                                             | (min_diff) 5-120                                                       |
| conditionally 5-77                                                                    | Compare Auxiliary Register Content 5-93                                |
| unconditionally 5-83                                                                  | Compare Auxiliary Register Content Maximum                             |
| Call Conditionally (if call) 5-77                                                     | (max) 5-105                                                            |
| Call Unconditionally (call) 5-83                                                      | Compare Auxiliary Register Content Minimum                             |
| cbit 5-128, 5-130                                                                     | (min) 5-108                                                            |
|                                                                                       | \i, 0 100                                                              |

| Compare Auxiliary Register Content with AND 5-95                                                                                                                                                        | shift 5-527<br>subtract 5-601                                                                                                                                                           |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Compare Auxiliary Register Content with                                                                                                                                                                 | count 5-134                                                                                                                                                                             |
| OR 5-100                                                                                                                                                                                                | Count Accumulator Bits (count) 5-134                                                                                                                                                    |
| compare maximum 5-105                                                                                                                                                                                   | Cross-Reference to Algebraic and Mnemonic                                                                                                                                               |
| Compare Memory with Immediate Value 5-126                                                                                                                                                               | Instruction Sets 7-1                                                                                                                                                                    |
| compare minimum 5-108                                                                                                                                                                                   |                                                                                                                                                                                         |
| Compare Temporary Register Content 5-93                                                                                                                                                                 | D                                                                                                                                                                                       |
| Compare Temporary Register Content Maximum (max) 5-105                                                                                                                                                  | delay 5-220                                                                                                                                                                             |
| Compare Temporary Register Content Minimum (min) 5-108                                                                                                                                                  | direct addressing modes 3-4 DP direct 3-4 PDP direct 3-5                                                                                                                                |
| Compare Temporary Register Content with AND 5-95                                                                                                                                                        | register-bit direct 3-5 SP direct 3-5                                                                                                                                                   |
| Compare Temporary Register Content with OR 5-100                                                                                                                                                        | Dual 16-Bit Addition and Subtraction 5-140 Dual 16-Bit Additions 5-135                                                                                                                  |
| complement accumulator bit 5-128 accumulator content 5-129 auxiliary register bit 5-128 auxiliary register content 5-129 memory bit 5-130 temporary register bit 5-128 temporary register content 5-129 | dual 16-bit arithmetic addition and subtraction 5-140 additions 5-135 subtraction and addition 5-154 subtractions 5-145  Dual 16-Bit Subtractions 5-145  Dual 16-Bit Subtractions 5-145 |
| Complement Accumulator Bit (cbit) 5-128                                                                                                                                                                 |                                                                                                                                                                                         |
| Complement Accumulator Content 5-129                                                                                                                                                                    | <b>E</b>                                                                                                                                                                                |
| Complement Auxiliary Register Bit (cbit) 5-128                                                                                                                                                          | Execute Conditionally (if execute) 5-159                                                                                                                                                |
| Complement Auxiliary Register Content 5-129                                                                                                                                                             | exp 5-131, 5-132                                                                                                                                                                        |
| Complement Memory Bit (cbit) 5-130                                                                                                                                                                      | Expand Accumulator Bit Field                                                                                                                                                            |
| Complement Temporary Register Bit (cbit) 5-128                                                                                                                                                          | (field_expand) 5-166                                                                                                                                                                    |
| Complement Temporary Register Content 5-129                                                                                                                                                             | extended auxiliary register (XAR)                                                                                                                                                       |
| Compute Exponent of Accumulator Content (exp) 5-131                                                                                                                                                     | load from memory 5-215<br>load with immediate value 5-216                                                                                                                               |
| Compute Mantissa and Exponent of Accumulator<br>Content 5-132                                                                                                                                           | modify content 5-246 modify content by addition 5-249 modify content by subtraction 5-251                                                                                               |
| cond field 1-7                                                                                                                                                                                          | move content 5-261                                                                                                                                                                      |
| conditional addition or subtraction 5-31 addition or subtraction with shift 5-33                                                                                                                        | pop content from stack pointers 5-468 push content to stack pointers 5-476 store to memory 5-600                                                                                        |
| addition, subtraction, or move accumulator content 5-36                                                                                                                                                 | Extract Accumulator Bit Field (field_extract) 5-167                                                                                                                                     |
| branch 5-66                                                                                                                                                                                             |                                                                                                                                                                                         |
| call 5-77                                                                                                                                                                                               |                                                                                                                                                                                         |
| execute 5-159                                                                                                                                                                                           | field expand F 166                                                                                                                                                                      |
| repeat single instruction 5-495 return 5-508                                                                                                                                                            | field_expand 5-166<br>field_extract 5-167                                                                                                                                               |
| 16tuii 3-300                                                                                                                                                                                            | Held_extract 5-107                                                                                                                                                                      |

| finite impulse response (FIR) filter<br>antisymmetrical 5-168<br>symmetrical 5-170 | instruction set terms, symbols, and abbreviations 1-2 interrupt 5-549     |
|------------------------------------------------------------------------------------|---------------------------------------------------------------------------|
| firs 5-170                                                                         | •                                                                         |
| firsn 5-168                                                                        | intr 5-549                                                                |
|                                                                                    |                                                                           |
| G                                                                                  |                                                                           |
|                                                                                    | Least Mean Square (Ims) 5-173                                             |
| goto 5-70                                                                          | Least Mean Square (Imsf) 5-175                                            |
|                                                                                    | linear 5-179                                                              |
|                                                                                    | Linear Addressing Qualifier (linear) 5-179                                |
|                                                                                    | List of Algebraic Instruction Opcodes 6-1                                 |
| idle 5-172                                                                         | lms 5-173                                                                 |
| if call 5-77                                                                       | Imsf 5-175                                                                |
| if execute 5-159                                                                   | load                                                                      |
|                                                                                    | accumulator from memory 5-180                                             |
| if goto 5-66, 5-74                                                                 | accumulator from memory with parallel store                               |
| if return 5-508                                                                    | accumulator content to memory 5-189                                       |
| indirect addressing modes 3-6                                                      | accumulator pair from memory 5-191 accumulator with immediate value 5-196 |
| AR indirect 3-6<br>CDP indirect 3-16                                               | accumulator, auxiliary, or temporary register from                        |
| coefficient indirect 3-19                                                          | memory 5-199                                                              |
| dual AR indirect 3-14                                                              | accumulator, auxiliary, or temporary register with                        |
| initialize memory 5-217                                                            | immediate value 5-205                                                     |
| instruction qualifier                                                              | auxiliary or temporary register pair from                                 |
| circular addressing 5-87                                                           | memory 5-209                                                              |
| linear addressing 5-179                                                            | CPU register from memory 5-210                                            |
| memory-mapped register access 5-221                                                | CPU register with immediate value 5-213                                   |
| instruction set                                                                    | extended auxiliary register (XAR) from<br>memory 5-215                    |
| abbreviations 1-2                                                                  | extended auxiliary register (XAR) with immediate                          |
| affect of status bits 1-9                                                          | value 5-216                                                               |
| conditional fields 1-7                                                             | memory with immediate value 5-217                                         |
| nonrepeatable instructions 1-20                                                    | Load Accumulator from Memory 5-180, 5-199                                 |
| notes 1-14                                                                         | Load Accumulator from Memory with Parallel Store                          |
| opcode symbols and abbreviations 6-19 opcodes 6-2                                  | Accumulator Content to Memory 5-189                                       |
| operators 1-6                                                                      | Load Accumulator Pair from Memory 5-191                                   |
| rules 1-14                                                                         | Load Accumulator with Immediate Value 5-196,                              |
| symbols 1-2                                                                        | 5-205                                                                     |
| terms 1-2                                                                          | Load Auxiliary Register from Memory 5-199                                 |
| instruction set conditional fields 1-7                                             | Load Auxiliary Register Pair from Memory 5-209                            |
| instruction set notes and rules 1-14                                               | Load Auxiliary Register with Immediate                                    |
| instruction set opcode                                                             | Value 5-205                                                               |
| abbreviations 6-19                                                                 | Load CPU Register from Memory 5-210                                       |
| symbols 6-19                                                                       | Load CPU Register with Immediate Value 5-213                              |
| instruction set opcodes 6-2                                                        | Load Extended Auxiliary Register (XAR) from                               |
| instruction set summary 4-1                                                        | Memory 5-215                                                              |

| Load Extended Auxiliary Register (XAR) with<br>Immediate Value 5-216        | auxiliary or temporary register content by addition 5-237                           |
|-----------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
| Load Memory with Immediate Value 5-217                                      | auxiliary or temporary register content by                                          |
| Load Temporary Register from Memory 5-199                                   | subtraction 5-241                                                                   |
| Load Temporary Register Pair from Memory 5-209                              | auxiliary register content 5-222                                                    |
| Load Temporary Register with Immediate                                      | auxiliary register content with parallel                                            |
| Value 5-205                                                                 | multiply 5-224                                                                      |
|                                                                             | auxiliary register content with parallel multiply<br>and accumulate 5-226           |
| local repeat 5-484                                                          | and accumulate 5-226 auxiliary register content with parallel multiply              |
| lock, access qualifier 5-218                                                | and subtract 5-231                                                                  |
| Lock Access Qualifier 5-218                                                 | data stack pointer 5-245                                                            |
| logical                                                                     | extended auxiliary register (XAR) content 5-246                                     |
| bitwise AND 5-38<br>bitwise OR 5-48                                         | extended auxiliary register (XAR) content by                                        |
| bitwise XOR 5-46                                                            | addition 5-249                                                                      |
| count accumulator bits 5-134                                                | extended auxiliary register (XAR) content by<br>subtraction 5-251                   |
| shift accumulator content logically 5-529                                   |                                                                                     |
| shift accumulator, auxiliary, or temporary register content logically 5-532 | Modify Auxiliary Register Content (mar) 5-222, 5-233                                |
|                                                                             | Modify Auxiliary Register Content by Addition (mar) 5-237                           |
| M                                                                           | Modify Auxiliary Register Content by Subtraction (mar) 5-241                        |
| mant 5-132                                                                  | Modify Auxiliary Register Content with Parallel                                     |
| mar 5-222, 5-233, 5-237, 5-241, 5-246, 5-249, 5-251, 5-406                  | Multiply (mar) 5-224                                                                |
| max 5-105                                                                   | Modify Auxiliary Register Content with Parallel Multiply and Accumulate (mar) 5-226 |
| max_diff 5-114                                                              | Modify Auxiliary Register Content with Parallel                                     |
| max_diff_dbl 5-114                                                          | Multiply and Subtract (mar) 5-231                                                   |
| memory bit                                                                  | Modify Data Stack Pointer 5-245                                                     |
| clear 5-89<br>complement (not) 5-130                                        | Modify Extended Auxiliary Register Content                                          |
| set 5-523                                                                   | (mar) 5-246                                                                         |
| test 5-645<br>test and clear 5-648                                          | Modify Extended Auxiliary Register Content by Addition (mar) 5-249                  |
| test and complement 5-649 test and set 5-650                                | Modify Extended Auxiliary Register Content by Subtraction (mar) 5-251               |
| Memory Delay (delay) 5-220                                                  | Modify Temporary Register Content (mar) 5-233                                       |
| Memory-Mapped Register Access Qualifier (mmap) 5-221                        | Modify Temporary Register Content by Addition (mar) 5-237                           |
| min 5-108                                                                   | Modify Temporary Register Content by Subtraction                                    |
|                                                                             | (mar) 5-241                                                                         |
| min_diff 5-120                                                              | move                                                                                |
| min_diff_dbl 5-120                                                          | accumulator content to auxiliary or temporary                                       |
| mmap 5-221                                                                  | register 5-253                                                                      |
| mnemonic instruction set cross-reference to algebraic instruction set 7-1   | accumulator, auxiliary, or temporary register                                       |
| modify                                                                      | content 5-254 auxiliary or temporary register content to                            |
| auxiliary or temporary register content 5-233                               | accumulator 5-256                                                                   |

auxiliary or temporary register content to CPU register 5-257 CPU register content to auxiliary or temporary register 5-259 extended auxiliary register content 5-261 memory delay 5-220 memory to memory 5-262 pop accumulator or extended auxiliary register content from stack pointers 5-468 pop top of stack 5-469 push accumulator or extended auxiliary register content to stack pointers 5-476 push to top of stack 5-477 swap accumulator content 5-629 swap accumulator pair content 5-630 swap auxiliary and temporary register content 5-633 swap auxiliary and temporary register pair content 5-635 swap auxiliary and temporary register pairs content 5-637 swap auxiliary register content 5-631 swap auxiliary register pair content 5-632 swap temporary register content 5-639 swap temporary register pair content 5-640 Move Accumulator Content 5-254 Move Accumulator Content to Auxiliary Register 5-253 Move Accumulator Content to Temporary Register 5-253 Move Auxiliary Register Content 5-254 Move Auxiliary Register Content to Accumulator 5-256 Move Auxiliary Register Content to CPU Register 5-257 Move CPU Register Content to Auxiliary Register 5-259 Move CPU Register Content to Temporary Register 5-259 Move Extended Auxiliary Register (XAR) Content 5-261 Move Memory to Memory 5-262 Move Temporary Register Content 5-254 Move Temporary Register Content to Accumulator 5-256 Move Temporary Register Content to CPU Register 5-257

Multiply and Accumulate (MAC) 5-308 Multiply and Accumulate with Parallel Delay 5-325 Multiply and Accumulate with Parallel Load Accumulator from Memory 5-327 Multiply and Accumulate with Parallel Multiply 5-329 Multiply and Accumulate with Parallel Multiply and Subtract 5-347 Multiply and Accumulate with Parallel Store Accumulator Content to Memory 5-367 Multiply and Subtract 5-369 Multiply and Subtract with Parallel Load Accumulator from Memory 5-379 Multiply and Subtract with Parallel Multiply 5-381 Multiply and Subtract with Parallel Multiply and Accumulate 5-390 Multiply and Subtract with Parallel Store Accumulator Content to Memory 5-401 Multiply with Parallel Multiply and Accumulate 5-283 Multiply with Parallel Multiply and Subtract 5-295 Multiply with Parallel Store Accumulator Content to Memory 5-305



Negate Accumulator Content 5-403
Negate Auxiliary Register Content 5-403
Negate Temporary Register Content 5-403
negation
accumulator content 5-403
auxiliary register content 5-403
temporary register content 5-403
No Operation (nop) 5-405
nonrepeatable instructions 1-20
nop 5-405



operand qualifier 5-466 OR 5-48



Parallel Modify Auxiliary Register Contents (mar) 5-406

Multiply 5-269

| Parallel Multiplies 5-407                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Pop Accumulator Content from Stack Pointers                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Parallel Multiply and Accumulates 5-419                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | (popboth) 5-468                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Parallel Multiply and Subtracts 5-454                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | Pop Extended Auxiliary Register (XAR) Content from Stack Pointers (popboth) 5-468                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Parallel Multiply and Subtracts 5-454  parallel operations    addition with parallel store accumulator content    to memory 5-29 load accumulator from memory with parallel store    accumulator content to memory 5-189    modify auxiliary register content with parallel    multiply 5-224    modify auxiliary register content with parallel    multiply and accumulate 5-226    modify auxiliary register content with parallel    multiply and subtract 5-231    modify auxiliary register contents 5-406    multiplies 5-407    multiply and accumulate with parallel    delay 5-325    multiply and accumulate with parallel load    accumulator from memory 5-327    multiply and accumulate with parallel multiply and    subtract 5-347    multiply and accumulate with parallel multiply and    subtract 5-347    multiply and accumulate with parallel store    accumulator content to memory 5-367    multiply and subtract with parallel load    accumulator from memory 5-379    multiply and subtract with parallel multiply and    accumulate 5-390    multiply and subtract with parallel store    accumulator content to memory 5-401    multiply and subtract with parallel store    accumulator content to memory 5-401    multiply and subtract with parallel store    accumulator content to memory 5-401    multiply with parallel multiply and    accumulate 5-283 | from Stack Pointers (popboth) 5-468  Pop Top of Stack (pop) 5-469  popboth 5-468  program control  branch conditionally 5-66  branch on auxiliary register not zero 5-74  branch unconditionally 5-70  call conditionally 5-77  call unconditionally 5-83  compare and branch 5-111  execute conditionally 5-159  idle 5-172  no operation 5-405  repeat block of instructions  unconditionally 5-484  repeat single instruction conditionally 5-495  repeat single instruction unconditionally 3-498  repeat single instruction unconditionally and  decrement CSR 5-503  repeat single instruction unconditionally and  increment CSR 5-505  return conditionally 5-508  return from interrupt 5-512  return unconditionally 5-510  software interrupt 5-549  software reset 5-551  software trap 5-555  pshboth 5-476  Push Accumulator Content to Stack Pointers  (pshboth) 5-476  Push Extended Auxiliary Register (XAR) Content to  Stack Pointers (pshboth) 5-476  Push to Top of Stack (push) 5-477 |
| multiply with parallel multiply and                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | R                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| subtract 5-295 multiply with parallel store accumulator content to memory 5-305 subtraction with parallel store accumulator content to memory 5-627                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | readport 5-466 register bit clear 5-88 complement (not) 5-128                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| parallelism basics 2-3                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | set 5-522                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| parallelism features 2-2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | test 5-641<br>test bit pair 5-643                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Peripheral Port Register Access Qualifiers 5-466                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | repeat 5-498, 5-503, 5-505                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| pop 5-469                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Repeat Block of Instructions Unconditionally 5-484                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |

Repeat Single Instruction Conditionally (while shift conditionally 5-527 repeat) 5-495 shift logically 5-529, 5-532 Repeat Single Instruction Unconditionally Shift Temporary Register Content Logically 5-532 (repeat) 5-498 Signed Shift of Accumulator Content 5-535, 5-544 Repeat Single Instruction Unconditionally and Signed Shift of Auxiliary Register Content 5-544 Decrement CSR (repeat) 5-503 Signed Shift of Temporary Register Content 5-544 Repeat Single Instruction Unconditionally and soft-dual parallelism 2-5 Increment CSR (repeat) 5-505 Software Interrupt (intr) 5-549 reset 5-551 Software Reset (reset) 5-551 resource conflicts in a parallel pair 2-4 Software Trap (trap) 5-555 sqdst 5-566 return 5-510 Square 5-557 Return Conditionally (if return) 5-508 Square and Accumulate 5-560 Return from Interrupt (return\_int) 5-512 Square and Subtract 5-563 Return Unconditionally (return) 5-510 Square Distance (sqdst) 5-566 return int 5-512 status register bit rnd 5-518 clear 5-90 Rotate Left Accumulator Content 5-514 set 5-524 Rotate Left Auxiliary Register Content 5-514 store Rotate Left Temporary Register Content 5-514 accumulator content to memory 5-568 accumulator pair content to memory 5-588 Rotate Right Accumulator Content 5-516 accumulator, auxiliary, or temporary register Rotate Right Auxiliary Register Content 5-516 content to memory 5-591 Rotate Right Temporary Register Content 5-516 auxiliary or temporary register pair content to Round Accumulator Content (rnd) 5-518 memory 5-595 rounding 5-518 CPU register content to memory 5-596 extended auxiliary register (XAR) to memory 5-600 Store Accumulator Content to Memory 5-568, 5-591 saturate 5-520 Store Accumulator Pair Content to Memory 5-588 Saturate Accumulator Content (saturate) 5-520 Store Auxiliary Register Content to Memory 5-591 Store Auxiliary Register Pair Content to accumulator bit 5-522 Memory 5-595 auxiliary register bit 5-522 Store CPU Register Content to Memory 5-596 memory bit 5-523 Store Extended Auxiliary Register (XAR) to status register bit 5-524 Memory 5-600 temporary register bit 5-522 Store Temporary Register Content to Set Accumulator Bit 5-522 Memory 5-591 Set Auxiliary Register Bit 5-522 Store Temporary Register Pair Content to Set Memory Bit 5-523 Memory 5-595 Set Status Register Bit 5-524 subc 5-601 Set Temporary Register Bit 5-522 Subtract Conditionally 5-601 sftc 5-527 Subtraction 5-603 Shift Accumulator Content Conditionally Subtraction with Parallel Store Accumulator Content (sftc) 5-527 to Memory 5-627 Shift Accumulator Content Logically 5-529, 5-532 swap 5-629, 5-630, 5-631, 5-632, 5-633, 5-635, Shift Auxiliary Register Content Logically 5-532 5-637, 5-639, 5-640

Swap Accumulator Content (swap) 5-629
Swap Accumulator Pair Content (swap) 5-630
Swap Auxiliary and Temporary Register Content (swap) 5-633
Swap Auxiliary and Temporary Register Pair Content (swap) 5-635
Swap Auxiliary and Temporary Register Pairs Content (swap) 5-637
Swap Auxiliary Register Content (swap) 5-631
Swap Auxiliary Register Pair Content (swap) 5-632
Swap Temporary Register Content (swap) 5-639
Swap Temporary Register Pair Content (swap) 5-639
Swap Temporary Register Pair Content (swap) 5-640
Symmetrical Finite Impulse Response Filter (firs) 5-170



## test

accumulator bit 5-641
accumulator bit pair 5-643
auxiliary register bit 5-641
auxiliary register bit pair 5-643
memory bit 5-645
temporary register bit 5-641
temporary register bit pair 5-643
Test Accumulator Bit 5-641
Test Accumulator Bit Pair 5-643
Test and Clear Memory Bit 5-648
Test and Complement Memory Bit 5-649

Test and Set Memory Bit 5-650
Test Auxiliary Register Bit 5-641
Test Auxiliary Register Bit Pair 5-643
Test Memory Bit 5-645
Test Temporary Register Bit 5-641
Test Temporary Register Bit Pair 5-643
trap 5-555



unconditional
branch 5-70
call 5-83
repeat block of instructions 5-484
repeat single instruction 5-498
repeat single instruction and decrement
CSR 5-503
repeat single instruction and increment
CSR 5-505
return 5-510
return from interrupt 5-512



while repeat 5-495 writeport 5-466



XOR 5-57

## **IMPORTANT NOTICE**

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

**Applications Products Amplifiers** amplifier.ti.com Audio www.ti.com/audio Data Converters Automotive www.ti.com/automotive dataconverter.ti.com DLP® Products Broadband www.dlp.com www.ti.com/broadband DSP Digital Control dsp.ti.com www.ti.com/digitalcontrol Clocks and Timers www.ti.com/clocks Medical www.ti.com/medical Military Interface www.ti.com/military interface.ti.com Optical Networking Logic logic.ti.com www.ti.com/opticalnetwork Power Mgmt power.ti.com Security www.ti.com/security Telephony Microcontrollers microcontroller.ti.com www.ti.com/telephony Video & Imaging www.ti-rfid.com www.ti.com/video RF/IF and ZigBee® Solutions www.ti.com/lprf Wireless www.ti.com/wireless

> Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265 Copyright © 2009, Texas Instruments Incorporated