TMS320C55x DSP Algebraic Instruction Set Reference Guide

Literature Number: SPRU375E
April 2001

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgment, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Customers are responsible for their applications using TI components.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, license, warranty or endorsement thereof.

Reproduction of 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. Representation or reproduction of this information with alteration voids all warranties provided for an associated TI product or service, is an unfair and deceptive business practice, and TI is not responsible nor liable for any such use.

Resale of TI's products or services with *statements different from or beyond the parameters* stated by TI for that products or service voids all express and any implied warranties for the associated TI product or service, is an unfair and deceptive business practice, and TI is not responsible nor liable for any such use.

Also see: Standard Terms and Conditions of Sale for Semiconductor Products.
www.ti.com/sc/docs/stdterms.htm

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265

Preface

Read This First

About This Manual

The TMS320C55x™ is a fixed-point digital signal processor (DSP) in the TMS320™ 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 (arithmetical, bit manipulation, logical, move, and program control).

Related Documentation From Texas Instruments

The following books describe the C55x™ 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 DSP CPU Reference Guide (literature number SPRU371) describes the architecture, registers, and operation of the CPU. This book also describes how to make individual portions of the DSP inactive to save power.

TMS320C55x DSP Mnemonic Instruction Set Reference Guide (literature number SPRU374) 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 DSP Algebraic Instruction Set Reference Guide (literature number SPRU375) describes the algebraic instructions individually. It also includes a summary of the instruction set, a list of the instruction opcodes, and a cross-reference to the mnemonic instruction set.

TMS320C55x Optimizing C Compiler User's Guide (literature number SPRU281) describes the 'C55x 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.

TMS320C55x DSP Programmer's Guide (literature number SPRU376) describes ways to optimize C and assembly code for the TMS320C55x DSPs and includes application program examples.

Code Composer Studio™ contains the following online guides:

TMS320C55x DSP Instruction Sets Online Guide describes the algebraic and mnemonic instructions individually. It also includes the parallelism features and rules of the instruction sets , a summary of the instruction sets, a list of the instruction opcodes, and a cross-reference of the mnemonic instruction sets.

TMS320C55x DSP Registers Online Guide describes the registers inside the TMS320C55x DSPs and shows the addresses for DSP registers that are mapped to memory.

TMS320C55x DSP CPU Online Guide describes the architecture and operation of the CPU inside the TMS320C55x DSPs. This guide also describes how to make individual portions of the DSP inactive to save power.

Trademarks

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

Contents

Tables

Chapter 1

Terms, Symbols, and Abbreviations

This chapter lists and defines the terms, symbols, and abbreviations used in the TMS320C55x™ DSP algebraic instruction set summary and in the individual instruction descriptions. Also provided are instruction set notes and rules.

Topic **Page**

1.1 Instruction Set Terms, Symbols, and Abbreviations

Table 1–1 and Table 1–2 list the terms, symbols, and abbreviations 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
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 (see Table 1–3) or postmodified (see Table 1–4) in the address generation unit.
ARx, ARy	Auxiliary register: AR0, AR1, AR2, AR3, AR4, AR5, AR6, AR7
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
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 1.3.
CSR	Computed single-repeat register
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.
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
Dx	Data address label coded on x bits (absolute address)
kx	Unsigned constant coded on x bits

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

Symbol	Meaning
Kx	Signed constant coded on x bits
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)
Lmem	Long-word single data memory access (32-bit data access). Same legal inputs as Smem.
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
Parallel Enable Bit	Indicates if the instruction contains a parallel enable bit.
Pipeline	Limiting execution pipeline phase: D Decode AD Address 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
rnd	If the optional rnd keyword is applied to the instruction, rounding is performed in the instruction
RPTC	Single-repeat counter register
saturate	If the optional saturate keyword is applied to the input operand, the 40-bit output of the operation is saturated
SHFT	Immediate shift value, 0 to 15
SHIFTW	Immediate shift value, –32 to +31
Size	Instruction size in bytes.
Smem	Word single data memory access (16-bit data access)
SP	Data stack pointer

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

Symbol	Meaning
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
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
Tx, Ty	Temporary register (Tx): T0, T1, T2, T3
uns	If the optional uns keyword is applied to the input operand, the operand is zero extended
XAdst	Destination extended register: All 23 bits of 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
XARx	All 23 bits of auxiliary register: XAR0, XAR1, XAR2, XAR3, XAR4, XAR5, XAR6, XAR7
XAsrc	Source extended register: All 23 bits of 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
xdst	Accumulator: AC0, AC1, AC2, AC3
	Destination extended register: All 23 bits of 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

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

Symbol	Meaning
xsrc	Accumulator: AC0, AC1, AC2, AC3
	Source extended register: All 23 bits of 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
|			Bitwise OR	Left to right
^			Bitwise exclusive OR (XOR)	Left to right

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

1.2 Auxiliary Register Modifications

Table 1–3 lists the available premodifications and Table 1–4 lists the available postmodifications to the auxiliary registers.

Table 1–3. Auxiliary Register Premodifications

Operand	Modification
*+ARn	ARn is incremented by 1 before the address is generated: ARn = ARn + 1
*–ARn	ARn is decremented by 1 before the address is generated: ARn = ARn – 1
*ARn(AR0)	ARn is not modified. ARn is used as a base pointer. AR0 is used as an offset from that base pointer.
	See Compatible mode caution, section 1.2.1.
*ARn(T0)	ARn is not modified. ARn is used as a base pointer. T0 is used as an offset from that base pointer.
	See Enhanced mode caution, section 1.2.2.
*ARn(T1)	ARn is not modified. ARn is used as a base pointer. T1 is used as an offset from that base pointer.
*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.
	See instruction size note, section 1.2.3.
*+ARn(#K16)	The 16-bit signed constant (K16) is added to ARn before the address is generated: ARn = ARn + K16
	See instruction size note, section 1.2.3.
*ARn(short(#k3))	ARn is not modified. ARn is used as a base pointer. The 3–bit constant (k3) is used as an offset from that base pointer. k3 is in the range 1 to 7.
*CDP(#K16)	CDP is not modified. CDP is used as a base pointer. The 16–bit signed constant (K16) is used as an offset from that base pointer.
	See instruction size note, section 1.2.3.
*+CDP(#K16)	The 16-bit signed constant (K16) is added to CDP before the address is generated: CDP = CDP + K16
	See instruction size note, section 1.2.3.

Table 1–4. Auxiliary Register Postmodifications

Operand	Modification
*ARn+	ARn is incremented by 1 after the address is generated: ARn = ARn + 1
*ARn–	ARn is decremented by 1 after the address is generated: ARn = ARn – 1
*(ARn + AR0)	AR0 is added to ARn after the address is generated: ARn = ARn + AR0
	See Compatible mode caution, section 1.2.1.
*(ARn + T0)	T0 is added to ARn after the address is generated: ARn = ARn + T0
	See Enhanced mode caution, section 1.2.2.
*(ARn – AR0)	AR0 is subtracted from ARn after the address is generated: ARn = ARn – AR0
	See Compatible mode caution, section 1.2.1.
*(ARn – T0)	T0 is subtracted from ARn after the address is generated: ARn = ARn – T0
	See Enhanced mode caution, section 1.2.2.
*(ARn + T0B)	T0 is added to ARn after the address is generated: ARn = ARn + T0 (The addition is done with reverse carry propagation.)
	This operand cannot be used for circular addressing.
*(ARn – T0B)	T0 is subtracted from ARn after the address is generated: ARn = ARn – T0 (The subtraction is done with reverse carry propagation.)
	This operand cannot be used for circular addressing.
*(ARn + T1)	T1 is added to ARn after the address is generated: ARn = ARn + T1
*(ARn – T1)	T1 is subtracted from ARn after the address is generated: ARn = ARn – T1
*CDP+	CDP is incremented by 1 after the address is generated: CDP = CDP + 1
*CDP–	CDP is decremented by 1 after the address is generated: CDP = CDP – 1

1.2.1 Compatible Mode Caution

- ❑ This modifier is available when C54CM = 1
- ❑ This modifier is usable when .c54cm_on is active at assembly time

1.2.2 Enhanced Mode Caution

- ❑ This modifier is available when C54CM = 0
- ❑ This modifier is usable when .c54cm_off is active at assembly time

1.2.3 Auxiliary Register Premodification Instruction Size 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.

1.3 Instruction Set Conditional (cond) Field

The following paragraphs provide the testing conditions available in the cond field of the conditional instructions.

1.3.1 Accumulator (ACx) Content

The available conditions testing the accumulator content against 0:

ACx == #0 (the content is equal to 0)	ACx != #0 (the content is not equal to 0)
ACx < #0 (the content is less than 0)	ACx <= #0 (the content is less than or equal to 0)
ACx > #0 (the content is greater than 0)	ACx >= #0 (the content is greater than or equal to 0)

The comparison against 0 depends on M40 status bit:

- If M40 = 0, ACx(31–0) is compared to 0.
- If M40 = 1, ACx(39–0) is compared to 0.

1.3.2 Accumulator Overflow Status Bit (ACOVx)

The available conditions testing the accumulator overflow status bit (ACOVx) against 1; when the optional ! symbol is used before the bit designation, the bit can be tested against 0.

overflow(ACx)	!overflow(ACx)

When these conditions are used, the corresponding accumulator overflow status bit is cleared to 0.

1.3.3 Auxiliary Register (ARx) Content

The available conditions testing the auxiliary register (ARx) content against 0:

*ARx == #0 (the content is equal to 0)	*ARx != #0 (the content is not equal to 0)
*ARx < #0 (the content is less than 0)	*ARx <= #0 (the content is less than or equal to 0)
*ARx > #0 (the content is greater than 0)	*ARx >= #0 (the content is greater than or equal to 0)

1.3.4 CARRY Status Bit

The available conditions testing the CARRY status bit against 1; when the optional ! symbol is used before the bit designation, the bit can be tested against 0.

CARRY	!CARRY

1.3.5 Temporary Register (Tx) Content

The available conditions testing the temporary register (Tx) content against 0:

Tx == #0 (the content is equal to 0)	Tx != #0 (the content is not equal to 0)
Tx < #0 (the content is less than 0)	Tx <= #0 (the content is less than or equal to 0)
Tx > #0 (the content is greater than 0)	Tx >= #0 (the content is greater than or equal to 0)

1.3.6 Test Control Flags, TC1 and TC2

The available conditions testing the test control flags (TC1 and TC2). Each of the bits can be tested independently against 1; when the optional ! symbol is used before the bit designation, the bits can be tested independently against 0.

TCx	!TCx

TC1 and TC2 can be combined with an AND (&), OR (|), and XOR (^) logical bit combinations:

TC1 & TC2	TC1 & !TC2
!TC1 & TC2	!TC1 & !TC2
TC1 | TC2	TC1 | !TC2
!TC1 | TC2	!TC1 | !TC2
TC1 ^ TC2	TC1 ^ !TC2
!TC1 ^ TC2	!TC1 ^ !TC2

1.4 Affect of Status Bits

1.4.1 Accumulator Overflow Status Bit (ACOVx)

The 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 an overflow is detected, the destination accumulator overflow status bit is set to 1.

1.4.2 C54CM Status Bit

- When C54CM = 0, the enhanced mode, the CPU supports code originally developed for a TMS320C55x™ 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 TMS320C54x™ DSP.

1.4.3 CARRY Status Bit

- When M40 = 0, the carry/borrow is detected at bit position 31
- When M40 = 1, the carry/borrow is detected at bit position 39

When performing a logical shift or signed shift that affects the CARRY status bit and the shift count is zero, the CARRY status bit is cleared to 0.

1.4.4 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.

1.4.5 INTM Status Bit

The INTM bit globally enables or disables the maskable interrupts. This bit has no 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.

1.4.6 M40 Status Bit

- 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™ DSP compatibility mode
  - for conditional instructions, the comparison against 0 (zero) is performed on 32 bits, ACx(31–0)
- 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)

1.4.6.1 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
  - if M40 = 1, the shifted-out bit is extracted at bit position 39
- After shifting, unless otherwise noted, when M40 = 0:
  - overflow is detected at bit position 31 (if an overflow is detected, the destination ACOVx bit is set)
  - 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™ DSP compatibility mode

❑ After shifting, unless otherwise noted, when M40 = 1:

- overflow is detected at bit position 39 (if an overflow is detected, the destination ACOVx bit is set)
- 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:

- 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, register saturation values are 7FFFh (positive overflow) or 8000h (negative overflow)

1.4.6.2 M40 Status Bit When Logically Shifting

In D-unit shifter:

❑ When shifting to the LSBs:

- 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

1.4.7 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 (2^{15}) 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.
```

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

1.4.8 SATA Status Bit

This status bit controls operations performed in the A unit.

- 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).

1.4.9 SATD Status Bit

This 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.

1.4.10 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 × 18000h is saturated to 00 7FFF FFFFh (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.

1.4.11 SXMD Status Bit

This status bit controls operations performed in the D unit.

- When SXMD = 0, input operands are zero extended.
- When SXMD = 1, input operands are sign extended.

1.4.12 Test Control Status Bit (TCx)

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

1.5 Instruction Set Notes and Rules

1.5.1 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
```

1.5.2 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(Cmem))),
ACy = m40(rnd(uns(Ymem) * uns(Cmem)))
```

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

```
AC0 = AC1  ||  AC2 = AC3

AC0 = AC1  ||
AC2 = AC3

AC0 = AC1
|| AC2 = AC3

AC0 = AC1
||
AC2 = AC3
```

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

```
blockrepeat {
       instr
        instr
          :
        instr
     }

localrepeat {
        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:

```
localrepeat {
        instr1
        instr2
   Label:
       }
       instr3
```

and

```
localrepeat {
        instr1
        instr2
       }
    Label:
       instr3
```

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
```

1.5.2.1 Reserved Words

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

1.5.2.2 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 ‘l’. 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 these syntax rules:

Rule 1

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

- a number (`#123`)
- an identifier (`#F00`)
- a parenthesized expression (`#(F00 + 2)`)

  Note that ‘#’ is not used inside the expression

For addresses only, rules 2 and 3 also apply.

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 Smem-absolute 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))`

1.5.2.3 Memory Operands

- Syntax of Smem is the same as that of Lmem or Baddr.
- 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+ * *CDP+)),
AC1 = (AC1 + (*AR3+ * *CDP+))
```

- An optional mmr prefix is allowed to be specified for indirect memory operands, for example, `mmr(*AR0)`. 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.

1.5.2.4 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).

1.5.2.5 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:

```
AC0 += *AR4
```

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 picked. This is not a valid instruction — an error is printed.

Chapter 2

Parallelism Features and Rules

This chapter describes the parallelism features and rules of the TMS320C55x™ DSP algebraic instruction set.

Topic **Page**

2.1 Parallelism Features

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

- 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 Rules

Parallelism between two instructions and only two instructions is allowed if all the rules are respected. The execution of a forbidden paralleled pair is not assured although the device is designed to execute a NOP (no operation) instruction.

2.2.1 Rule 1: Instruction length less than 6 bytes

Two instructions can be assembled in parallel if the added length of the instructions does not exceed 48 bits (6 bytes).

2.2.2 Rule 2: Instruction set support for parallelism

Two instructions can be assembled in parallel:

- ❑ If one of the two instructions is provided with a parallel enable bit. The hardware support for such a type of parallelism is called parallel enable mechanism.
- ❑ If both of the instructions make single data memory accesses (Smem only, or dbl(Lmem) only) in indirect addressing mode as specified in Rule 8 (Instructions with Smem Operands and Instructions with dbl(Lmem) Operands). The hardware support for such a type of parallelism is called soft dual mechanism.

2.2.3 Rule 3: Hardware resource conflicts

Two instructions can be paralleled if the memory bus, cross unit bus, and constant bus do not compete for access.

2.2.4 Rule 4: Parallelism between the A-unit, the D-unit, and the P-unit

Parallelism between the three main computation units of the device is allowed without restriction. An operation executed within a single unit can be paralleled with a second operation executed in one of the two other computation units.

2.2.5 Rule 5: Parallelism within the P-unit

The device allows for any parallelism between the following subunits:

- ❑ the P-unit load path
- ❑ the P-unit store path
- ❑ the P-unit control operators

The following summarizes parallelism between the subunits:

Example	Instruction Types
`BRC1 = #3` `|| T1 = BRC0`	P-unit load and P-unit store
`BRC1 = @variable` `|| if( AC0 >= #0) goto #label`	P-unit load and P-unit control operator
`T1 = BRC1` `|| repeat(#5)`	P-unit store and P-unit control operator

In addition to the previous parallelism combinations, the device allows for parallelism within the P-unit:

- ❑ two load operations in parallel with the P-unit
- ❑ two store operations in parallel with the P-unit

The following summarizes parallelism within the P-unit:

Example	Instruction Types
`BRC1 = #4` `|| BRC0 = T1`	Two P-unit loads
`*AR3 = BRC0` `|| *AR5 = BRC1`	Two P-unit stores

2.2.6 Rule 6: Parallelism within the D-Unit

The device allows for any parallelism between the following subunits:

- ❑ the D-unit load path
- ❑ the D-unit store path
- ❑ the D-unit swap operator
- ❑ the D-unit ALU, shifter, DMAC operators (all considered a single operator)
- ❑ the D-unit shift and store path

Parallelism between the ALU, shifter, and DMAC is not allowed. The following table summarizes parallelism between the subunits:

Example	Instruction Types
`AC1 = #3` `|| dbl(*AR4) = AC2`	D-unit load and D-unit store
`AC1 = @variable` `|| swap(AC0, AC2)`	D-unit load and D-unit swap
`AC1 = @variable << #16` `|| AC3 = AC1`	D-unit load and D-unit ALU
`AC1 = #3 << #16` `|| AC3 = AC3 * T1`	D-unit load and D-unit MAC
`AC1 = @variable` `|| AC3 = AC1 << #2`	D-unit load and D-unit shifter
`AC1 = *AR1` `|| *AR1 = hi(AC1 << #3)`	D-unit load and D-unit shift and store
`@variable = AC1` `|| swap(pair(AC0), pair(AC2))`	D-unit store and D-unit swap
`@variable = hi(AC1)` `|| AC3 = AC1`	D-unit store and D-unit ALU
`@variable = pair(hi(AC0))` `|| AC3 = AC3 * T1`	D-unit store and D-unit MAC
`@variable = AC1` `|| AC3 = AC1 << T2`	D-unit store and D-unit shifter
`*AR2 = AC1` `|| *AR1 = hi(AC1 << #3)`	D-unit store and D-unit shift and store
`swap(AC0, AC2)` `|| AC3 = AC1`	D-unit swap and D-unit ALU
`swap(AC0, AC2)` `|| AC3 = AC3 * T1`	D-unit swap and D-unit MAC
`swap(AC1, AC3)` `|| AC2 = AC1 << #2`	D-unit swap and D-unit shifter
`swap(pair(AC0), pair(AC2))` `|| *AR1 = hi(AC1 << T2)`	D-unit swap and D-unit shift and store
`AC3 = AC1 & *AR2` `|| *AR1 = hi(AC1 << T2)`	D-unit ALU and D-unit shift and store
`AC3 = AC3 * T1` `|| *AR1 = hi(rnd(AC1 << #3))`	D-unit MAC and D-unit shift and store

Considerations for the D-unit shift and store path:

- ❑ D-unit shift and store operations are not allowed in parallel with other instructions using the D-unit shifter.
- ❑ A maximum of two accumulators can be selected as source operands of the instructions to be executed in parallel within the D-unit.

In addition to the previous parallelism combinations, the device allows for parallelism within the D-unit:

- ❑ two load operations in parallel with the D-unit
- ❑ two store operations in parallel with the D-unit

The following summarizes parallelism within the D-unit:

Example	Instruction Types
`AC1 = *AR3` `|| AC2 = *AR4 << #16`	Two D-unit loads
`*AR2 = AC1` `|| *AR4 = AC2`	Two D-unit stores

2.2.7 Rule 7: Parallelism within the A-unit (excluding the data address generation units)

Excluding X, Y, C, and SP data address generation unit operators, the device allows for any parallelism between the following subunits:

- ❑ the A-unit load path
- ❑ the A-unit store path
- ❑ the A-unit swap operator
- ❑ the A-unit ALU operator

Two A-unit ALU operations or two A-unit swap operations cannot be performed in parallel. The following table summarizes parallelism between the subunits:

Example	Instruction Types
`AR1 = #3` `|| *AR4 = AR2`	A-unit load and A-unit store
`AR1 = @variable` `|| AR3 = AC1`	A-unit load and A-unit ALU
`AR1 = #3` `|| AR3 = AR3 + AR1`	A-unit load and A-unit ALU
`AR1 = @variable` `|| swap(pair(T0), pair(T2))`	A-unit load and A-unit swap
`@variable = AR1` `|| AR3 = AR3 + AC1`	A-unit store and A-unit ALU
`@variable = AR1` `|| swap(pair(T0), pair(T2))`	A-unit store and A-unit swap
`AR3 = AR2 & *AR2` `|| swap(block(AR4), block(T0))`	A-unit ALU and A-unit swap

In addition to the previous parallelism combinations, the device allows for parallelism within the A-unit:

- ❑ two load operations in parallel with the A-unit.
- ❑ two store operations in parallel with the A-unit.

The following summarizes parallelism within the A-unit:

Example	Instruction Types
`AR1 = *AR3` `|| AR2 = *AR4`	Two A-unit loads
`*AR3 = AR1` `|| *AR4 = AR2`	Two A-unit stores

2.2.8 Rule 8: Parallelism within the A-unit data address generation unit

The data address generation unit (DAGEN) contains four operators:

- ❑ DAGEN X and DAGEN Y are the most generic of the operators, they allow you to generate any of the following addressing modes:
  - ■ Single data memory addressing, Smem or dbl(Lmem)
  - ■ Indirect dual data memory addressing (Xmem, Ymem)
  - ■ Coefficient data memory addressing, Cmem
  - ■ Register bit addressing, Baddr or pair(Baddr)

  DAGEN X and Y operators are also used to perform pointer modification with the mar() instructions.
- ❑ DAGEN C is a dedicated operator that is used for coefficient data memory addressing (Cmem).
- ❑ DAGEN SP is a dedicated operator that is used to address the data and system stacks.

The device enables you to parallel two instructions that each use the address generation unit to generate data memory or register bit addresses. This feature provides:

- ❑ The full bandwidth of the memory buses
- ❑ Memory-based instruction set flexibility

2.2.9 Rule 9: Modifier limitations

When the following addressing modifiers are used within one instruction, this instruction cannot be put in parallel with another instruction:

- *ARn(K16)
- *+ARn(K16)
- *CDP(K16)
- *+CDP(K16)
- *abs16(#k16)
- *(#k23)
- *port(#k16)

2.2.10 Rule 10: Illegal program control instruction in paralleled instruction pair

The following control instructions cannot be executed in parallel with any instructions:

- `idle`
- `intr(k5)`
- `reset`
- `trap(k5)`

2.2.11 Rule 11: Illegal multicycle instruction in paralleled instruction pair

Multicycle instructions are allowed in a paralleled instruction pair combination as long as the other instruction is a single cycle instruction.

2.2.12 Rule 12: Parallelism with memory-mapped register and peripheral port register access qualifiers

An instruction that uses the mmap() qualifier to indicate an access to a memory-mapped register or registers cannot be placed in parallel with another instruction.

An instruction that uses the readport() or writeport() qualifier to indicate an access to an I/O space cannot be placed in parallel with another instruction.

2.2.13 Hardware Overwrite Rules

These hardware overwrite rules are applied within the device when the parallelism rules are not respected by the execution of an instruction. The assembler tool may generate warning messages instead of error messages for detectable errors.

2.2.13.1 Instruction constant priority

If two paralleled instructions have conflicting use of constants on the KAB or KDB buses, then the constant of the instruction encoded at the higher address (second instruction) is used for both instructions. An example of a conflict where both instructions of a paralleled pair use the KDB bus:

```
AC1 = AC1 + #1
|| T2 = #0xFFFF
```

2.2.13.2 Register update priority in the Address pipeline phase

The auxiliary (ARx), temporary (Tx), and coefficient pointer (CDP) registers can be updated in the address pipeline phase:

- With operations performed in DAGEN X, Y, C (Smem, dbl(Lmem), Xmem, Ymem, Cmem, Baddr, and pair(Baddr) addressing modes, and mar() instructions).
- With swap() instructions.

If two paralleled instructions have conflicting updates of a register, then the destination register is updated with the following priority:

- DAGEN X (highest priority)
- DAGEN Y
- DAGEN C
- swap() instruction (lowest priority)

2.2.13.3 Register update priority in the Execute pipeline phase

If two paralleled instructions have two identical destination registers (or status bits), then the second instruction encoded at the higher address has priority over the first instruction. A similar mechanism exists for single instructions that have two destination registers that are identical.

2.2.13.4 Status bit update priority

If two paralleled instructions have conflicting updates of a status bit located in status registers ST0, ST1, ST2, or ST3, then the status bit is updated with the following priority:

- Operations performed in the D-unit. (highest priority)
- Operations performed in the A-unit.
- Memory-mapped register (MMR) access to ST0, ST1, ST2, and ST3.
- Operations performed in the P-unit (Reset due to condition evaluation by program control instructions). (lowest priority)

2.3 Instructions Using Single Data Memory Operands, Smem and dbl(Lmem)

The hardware support for this type of parallelism is called soft dual mechanism. Instructions having single data memory operands Smem can be paralleled with each other:

- ❑ if both instructions make indirect addressing to these memory operands
- ❑ if the modifiers used to modify the pointers are those allowed for indirect dual data memory addressing (Xmem, Ymem).

Some restrictions apply:

- ❑ Instructions embedding a Cmem (and Smem) operand can only be paralleled with instructions having an Smem read operand.
- ❑ The following instructions cannot be paralleled using the soft dual mechanism:
  - 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`
  - Instructions embedding delay(Smem), which moves the memory operands Smem to the next higher address.
    - `delay(Smem)`
    - `ACx = rnd(ACx + (Smem * Cmem)) [,T3 = Smem], delay(Smem)`
  - Instructions reading a memory operand Smem, modifying it in the A or D-unit ALU, and storing it back to the same memory location.
    - `Smem = Smem & k16`
    - `Smem = Smem | k16`
    - `Smem = Smem ^ k16`
    - `Smem = Smem + K16`
    - `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)`
  - Instructions performing memory-to-memory moves.
    - `Smem = Cmem`
    - `Cmem = Smem`

Instructions having single data memory operands dbl(Lmem) can be paralleled with each other:

- ❑ if both instructions make indirect addressing to these memory operands
- ❑ if the modifiers used to modify the pointers are those allowed for indirect dual data memory addressing (Xmem, Ymem).

2.4 Instructions with Xmem, Ymem, and Cmem Operands

Instructions having the following data memory operands cannot be paralleled with instructions using any of the four DAGEN operators:

- ❑ Indirect dual data memory addressing (Xmem, Ymem)
- ❑ Coefficient data memory addressing (Cmem) in some cases

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.
- ❑ and the Cmem operand is defined as unsigned, 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, Ymem operand must also be defined as unsigned (and reciprocally).

2.5 MAR Instruction

The following MAR (modify auxiliary register) instructions can be paralleled with each other:

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

The MAR instruction can also be executed in parallel with instructions using the following addressing modes:

- ❑ Single data memory addressing, Smem or dbl(Lmem)
- ❑ Register bit addressing, Baddr or pair(Baddr)
- ❑ Data and system stack addressing instructions

2.6 Instructions Addressing the Data or System Stack

Instructions addressing the data or system stack cannot be paralleled with each other. These set of instructions include:

- ❑ all push() to the top of stack instructions
- ❑ all pop() to the top of stack instructions
- ❑ all conditional and unconditional call() instructions
- ❑ all conditional and unconditional return() from subroutine instructions
- ❑ trap(), intr(), and return_int instructions

Some instructions addressing the data or system stack can be paralleled with instructions using other DAGEN operators.

Any of the following instructions addressing the data or system stack (DAGEN SP operator) …	**can use any of the following instructions using only the DAGEN X, Y, or C operators**
`dst1, dst2 = pop()`	an Smem or dbl(Lmem) write operand
`dst = pop()`	Xmem and Ymem write operands
`ACx = dbl(pop())`	a Baddr or pair(Baddr) operand
`if (cond) return`	ARn_mod operand
`return`	
`return_int`	

Any of the following instructions addressing the data or system stack (DAGEN SP operator) …	**can use any of the following instructions using only the DAGEN X, Y, or C operators**
`push(src1, src2)`	an Smem or dbl(Lmem) read operand
`push(src)`	Xmem and Ymem read operands
`dbl(push(ACx))`	
`if (cond) call #label`	
`call #label`	
`call ACx`	

Chapter 3

Instruction Set Summary

The TMS320C55x™ DSP algebraic instruction set can be divided into six basic types of operations:

- Arithmetical operations
- Bit manipulation operations
- Extended adress register (XAR) operations
- Logical operations
- Move operations
- Program-control operations

In this chapter, each of the types of operations is divided into smaller groups of instructions with similar functions. With each instruction listing, you will find the availability of a parallel enable bit, word count (size), cycle time, what pipeline stage the instruction is executed, and in what unit the instruction is executed.

Topic **Page**

3.1 Arithmetical Operations

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
Absolute Distance (page 4-2)					
abdst(Xmem,Ymem,ACx,ACy)	No	4	1	X	D unit MAC and ALU
Absolute Value (page 4-4)					
dst = |src|	Yes	2	1	X	A or D unit ALU
Addition (page 4-70)					
dst = dst + src	Yes	2	1	X	A or D unit ALU
dst = dst + k4	Yes	2	1	X	A or D unit ALU
dst = src + K16	No	4	1	X	A or D unit ALU
dst = src + Smem	No	3	1	X	A or D unit ALU
ACy = ACy + (ACx << Tx)	Yes	2	1	X	D unit ALU and shifter
ACy = ACy + (ACx << #SHIFTW)	Yes	3	1	X	D unit ALU and shifter
ACy = ACx + (K16 << #16)	No	4	1	X	D unit ALU
ACy = ACx + (K16 << #SHFT)	No	4	1	X	D unit ALU and shifter
ACy = ACx + (Smem << Tx)	No	3	1	X	D unit ALU and shifter
ACy = ACx + (Smem << #16)	No	3	1	X	D unit ALU
ACy = ACx + uns(Smem) + CARRY	No	3	1	X	D unit ALU
ACy = ACx + uns(Smem)	No	3	1	X	D unit ALU
ACy = ACx + (uns(Smem) << #SHIFTW)	No	4	1	X	D unit ALU and shifter
ACy = ACx + dbl(Lmem)	No	3	1	X	D unit ALU
ACx = (Xmem << #16) + (Ymem << #16)	No	3	1	X	D unit ALU
Smem = Smem + K16	No	4	1	X	D unit ALU
ACy = rnd(ACy + |ACx|)	Yes	2	1	X	D unit MAC

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
Compare and Select Extremum (page 4-146)					D unit ALU
max_diff(ACx,ACy,ACz,ACw)	Yes	3	1	X	
max_diff_dbl(ACx,ACy,ACz,ACw,TRNx)	Yes	3	1	X	
min_diff(ACx,ACy,ACz,ACw)	Yes	3	1	X	
min_diff_dbl(ACx,ACy,ACz,ACw,TRNx)	Yes	3	1	X	
Conditional Addition/Subtraction (page 4-157)					
ACy = adsc(Smem,ACx,TCx)	No	3	1	X	D unit ALU
ACy = adsc(Smem,ACx,TC1,TC2)	No	3	1	X	D unit ALU
ACy = ads2c(Smem,ACx,Tx,TC1,TC2)	No	3	1	X	D unit shifter
Conditional Shift (page 4-165)					
ACx = sftc(ACx,TCx)	Yes	2	1	X	D unit shifter
Conditional Subtract (page 4-166)					
subc(Smem,ACx,ACy)	No	3	1	X	D unit ALU
Dual 16-Bit Arithmetic (page 4-168)					D unit ALU
HI(ACx) = Smem + Tx, LO(ACx) = Smem – Tx	No	3	1	X	
HI(ACx) = Smem – Tx, LO(ACx) = Smem + Tx	No	3	1	X	
HI(ACy) = HI(Lmem) + HI(ACx), LO(ACy) = LO(Lmem) + LO(ACx)	No	3	1	X	
HI(ACy) = HI(ACx) – HI(Lmem), LO(ACy) = LO(ACx) – LO(Lmem)	No	3	1	X	
HI(ACy) = HI(Lmem) – HI(ACx), LO(ACy) = LO(Lmem) – LO(ACx)	No	3	1	X	
HI(ACx) = Tx – HI(Lmem), LO(ACx) = Tx – LO(Lmem)	No	3	1	X	
HI(ACx) = HI(Lmem) + Tx, LO(ACx) = LO(Lmem) + Tx	No	3	1	X	
HI(ACx) = HI(Lmem) – Tx, LO(ACx) = LO(Lmem) – Tx	No	3	1	X	
HI(ACx) = HI(Lmem) + Tx, LO(ACx) = LO(Lmem) – Tx	No	3	1	X	

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
HI(ACx) = HI(Lmem) – Tx, LO(ACx) = LO(Lmem) + Tx	No	3	1	X	D unit ALU
Dual Multiply (Accumulate/Subtract) (page 4-189)					D unit MACs
ACx = M40(rnd(uns(Xmem) * uns(Cmem))), ACy = M40(rnd(uns(Ymem) * uns(Cmem)))	No	4	1	X	
ACx = M40(rnd(ACx + (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(uns(Ymem) * uns(Cmem)))	No	4	1	X	
ACx = M40(rnd(ACx – (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(uns(Ymem) * uns(Cmem)))	No	4	1	X	
mar(Xmem), ACx = M40(rnd(uns(Ymem) * uns(Cmem)))	No	4	1	X	
ACx = M40(rnd(ACx + (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(ACy + (uns(Ymem) * uns(Cmem))))	No	4	1	X	
ACx = M40(rnd(ACx – (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(ACy + (uns(Ymem) * uns(Cmem))))	No	4	1	X	
mar(Xmem), ACx = M40(rnd(ACx + (uns(Ymem) * uns(Cmem))))	No	4	1	X	
ACx = M40(rnd(ACx – (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(ACy – (uns(Ymem) * uns(Cmem))))	No	4	1	X	
mar(Xmem), ACx = M40(rnd(ACx – (uns(Ymem) * uns(Cmem))))	No	4	1	X	
ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(ACy + (uns(Ymem) * uns(Cmem))))	No	4	1	X	
ACx = M40(rnd(uns(Xmem) * uns(Cmem))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(Cmem))))	No	4	1	X	
ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(Cmem))))	No	4	1	X	
ACx = M40(rnd(ACx – (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(Cmem))))	No	4	1	X	
mar(Xmem), ACx = M40(rnd((ACx >> #16) + (uns(Ymem) * uns(Cmem))))	No	4	1	X	
mar(Xmem), mar(Ymem), mar(Cmem)	No	4	1	X	

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
Finite Impulse Response Filter, Symmetrical/Antisymmetrical (page 4-228)					D unit MAC and ALU
firs(Xmem,Ymem,Cmem,ACx,ACy)	No	4	1	X	
firsn(Xmem,Ymem,Cmem,ACx,ACy)	No	4	1	X	
Implied Paralleled Instructions (page 4-234)					
ACy = rnd(Tx * Xmem), Ymem = HI(ACx << T2) [,T3 = Xmem]	No	4	1	X	D unit MAC and shifter
ACy = rnd(ACy + (Tx * Xmem)), Ymem = HI(ACx << T2) [,T3 = Xmem]	No	4	1	X	D unit MAC and shifter
ACy = rnd(ACy – (Tx * Xmem)), Ymem = HI(ACx << T2) [,T3 = Xmem]	No	4	1	X	D unit MAC and shifter
ACy = ACx + (Xmem << #16), Ymem = HI(ACy << T2)	No	4	1	X	D unit ALU and shifter
ACy = (Xmem << #16) – ACx, Ymem = HI(ACy << T2)	No	4	1	X	D unit ALU and shifter
ACy = Xmem << #16, Ymem = HI(ACx << T2)	No	4	1	X	D unit ALU and shifter
ACx = rnd(ACx + (Tx * Xmem)), ACy = Ymem << #16 [,T3 = Xmem]	No	4	1	X	D unit MAC
ACx = rnd(ACx – (Tx * Xmem)), ACy = Ymem << #16 [,T3 = Xmem]	No	4	1	X	D unit MAC
Least Mean Square (LMS) (page 4-251)					
lms(Xmem,Ymem,ACx,ACy)	No	4	1	X	D unit MAC and ALU
Maximum Comparison (page 4-263)					
dst = max(src,dst)	Yes	2	1	X	A or D unit ALU
Minimum Comparison (page 4-266)					
dst = min(src,dst)	Yes	2	1	X	A or D unit ALU
Memory Comparison (page 4-278)					
TCx = (Smem == K16)	No	4	1	X	A unit ALU

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
Modify Auxiliary Register (MAR) (page 4-288)					A unit DAGENs
mar(TAy + TAx)	Yes	3	1	AD	
mar(TAy – TAx)	Yes	3	1	AD	
mar(TAy = TAx)	Yes	3	1	AD	
mar(TAx + k8)	Yes	3	1	AD	
mar(TAx – k8)	Yes	3	1	AD	
mar(TAx = k8)	Yes	3	1	AD	
mar(TAx = D16)	No	4	1	AD	
mar(Smem)	No	2	1	AD	
Modify Data Stack Pointer (SP) (page 4-298)					
SP = SP + K8	Yes	2	1	AD	A unit ALU
Multiply (page 4-299)					D unit MAC
ACy = rnd(ACx * ACx)	Yes	2	1	X	
ACy = rnd(ACy * ACx)	Yes	2	1	X	
ACy = rnd(ACx * Tx)	Yes	2	1	X	
ACy = rnd(ACx * K8)	Yes	3	1	X	
ACy = rnd(ACx * K16)	No	4	1	X	
ACx = rnd(Smem * Cmem) [,T3 = Smem]	No	3	1	X	
ACx = rnd(Smem * Smem) [,T3 = Smem]	No	3	1	X	
ACy = rnd(Smem * ACx) [,T3 = Smem]	No	3	1	X	
ACx = rnd(Smem * K8) [,T3 = Smem]	No	4	1	X	
ACx = M40(rnd(uns(Xmem) * uns(Ymem))) [,T3 = Xmem]	No	4	1	X	
ACx = rnd(uns(Tx * Smem)) [,T3 = Smem]	No	3	1	X	

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
Multiply and Accumulate (MAC) (page 4-314)					D unit MAC
ACy = rnd(ACy + (ACx * ACx))	Yes	2	1	X	
ACy = rnd(ACy + (ACx * Tx))	Yes	2	1	X	
ACy = rnd((Acy * Tx) + ACx)	Yes	2	1	X	
ACy = rnd(ACx + (Tx * K8))	Yes	3	1	X	
ACy = rnd(ACx + (Tx * K16))	No	4	1	X	
ACx = rnd(ACx + (Smem * Cmem)) [,T3 = Smem]	No	3	1	X	
ACx = rnd(ACx + (Smem * Cmem)) [,T3 = Smem], delay(Smem)	No	3	1	X	
ACy = rnd(ACx + (Smem * Smem)) [,T3 = Smem]	No	3	1	X	
ACy = rnd(ACy + (Smem * ACx)) [,T3 = Smem]	No	3	1	X	
ACy = rnd(ACx + (Tx * Smem)) [,T3 = Smem]	No	3	1	X	
ACy = rnd(ACx + (Smem * K8)) [,T3 = Smem]	No	4	1	X	
ACy = M40(rnd(ACx + (uns(Xmem) * uns(Ymem)))) [,T3 = Xmem]	No	4	1	X	
ACy = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(Ymem)))) [,T3 = Xmem]	No	4	1	X	
Multiply and Subtract (MAS) (page 4-332)					D unit MAC
ACy = rnd(ACy – (ACx * ACx))	Yes	2	1	X	
ACy = rnd(ACy – (ACx * Tx))	Yes	2	1	X	
ACx = rnd(ACx – (Smem * Cmem)) [,T3 = Smem]	No	3	1	X	
ACy = rnd(ACx – (Smem * Smem)) [,T3 = Smem]	No	3	1	X	
ACy = rnd(ACy – (Smem * ACx)) [,T3 = Smem]	No	3	1	X	
ACy = rnd(ACx – (Tx * Smem)) [,T3 = Smem]	No	3	1	X	
ACy = M40(rnd(ACx – (uns(Xmem) * uns(Ymem)))) [,T3 = Xmem]	No	4	1	X	
Negation (page 4-343)					
dst = –src	Yes	2	1	X	A or D unit ALU

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
Normalization (page 4-346)					A unit ALU and D unit shifter
ACy = mant(ACx), Tx = –exp(ACx)	Yes	3	1	X	
Tx = exp(ACx)	Yes	3	1	X	
Register Comparison (page 4-378)					A or D unit ALU
TCx = uns(src RELOP dst)	Yes	3	1	X	
TCx = TCy & uns(src RELOP dst)	Yes	3	1	X	
TCx = !TCy & uns(src RELOP dst)	Yes	3	1	X	
TCx = TCy | uns(src RELOP dst)	Yes	3	1	X	
TCx = !TCy | uns(src RELOP dst)	Yes	3	1	X	
Round (page 4-414)					
ACy = rnd(ACx)	Yes	2	1	X	D unit ALU
Saturate (page 4-416)					
ACy = saturate(rnd(ACx))	Yes	2	1	X	D unit ALU
Signed Shift (page 4-418)					
dst = dst >> #1	Yes	2	1	X	A unit ALU or D unit shifter
dst = dst << #1	Yes	2	1	X	A unit ALU or D unit shifter
ACy = ACx << Tx	Yes	2	1	X	D unit shifter
ACy = ACx <<C Tx	Yes	2	1	X	D unit shifter
ACy = ACx << #SHIFTW	Yes	3	1	X	D unit shifter
ACy = ACx <<C #SHIFTW	Yes	3	1	X	D unit shifter
Square Distance (page 4-450)					
sqdst(Xmem,Ymem,ACx,ACy)	No	4	1	X	D unit MAC and ALU

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
Subtraction (page 4-456)					
dst = dst – src	Yes	2	1	X	A or D unit ALU
dst = dst – k4	Yes	2	1	X	A or D unit ALU
dst = src – K16	No	4	1	X	A or D unit ALU
dst = src – Smem	No	3	1	X	A or D unit ALU
dst = Smem – src	No	3	1	X	A or D unit ALU
ACy = ACy – (ACx << Tx)	Yes	2	1	X	D unit ALU and shifter
ACy = ACy – (ACx << #SHIFTW)	Yes	3	1	X	D unit ALU and shifter
ACy = ACx – (K16 << #16)	No	4	1	X	D unit ALU
ACy = ACx – (K16 << #SHFT)	No	4	1	X	D unit ALU and shifter
ACy = ACx – (Smem << Tx)	No	3	1	X	D unit ALU and shifter
ACy = ACx – (Smem << #16)	No	3	1	X	D unit ALU
ACy = (Smem << #16) – ACx	No	3	1	X	D unit ALU
ACy = ACx – uns(Smem) – BORROW	No	3	1	X	D unit ALU
ACy = ACx – uns(Smem)	No	3	1	X	D unit ALU
ACy = ACx – (uns(Smem) << #SHIFTW)	No	4	1	X	D unit ALU and shifter
ACy = ACx – dbl(Lmem)	No	3	1	X	D unit ALU
ACy = dbl(Lmem) – ACx	No	3	1	X	D unit ALU
ACx = (Xmem << #16) – (Ymem << #16)	No	3	1	X	D unit ALU

3.2 Bit Manipulation Operations

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
Bit Field Comparison (page 4-91)					
TCx = Smem & k16	No	4	1	X	A unit ALU
Bit Field Expand (page 4-93)					
dst = field_expand(ACx,k16)	No	4	1	X	D unit shifter
Bit Field Extract (page 4-94)					
dst = field_extract(ACx,k16)	No	4	1	X	D unit shifter
Memory Bit Test/Set/Clear/Complement (page 4-269)					A unit ALU
TCx = bit(Smem,src)	No	3	1	X	
cbit(Smem,src)	No	3	1	X	
bit(Smem,src) = #0	No	3	1	X	
bit(Smem,src) = #1	No	3	1	X	
TCx = bit(Smem,k4), bit(Smem,k4) = #1	No	3	1	X	
TCx = bit(Smem,k4), bit(Smem,k4) = #0	No	3	1	X	
TCx = bit(Smem,k4), cbit(Smem,k4)	No	3	1	X	
TCx = bit(Smem,k4)	No	3	1	X	
Register Bit Test/Set/Clear/Complement (page 4-371)					A or D unit ALU
TCx = bit(src,Baddr)	No	3	1	X	
cbit(src,Baddr)	No	3	1	X	
bit(src,Baddr) = #0	No	3	1	X	
bit(src,Baddr) = #1	No	3	1	X	
bit(src,pair(Baddr))	No	3	1	X	

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
Status Bit Set/Clear (page 4-452)					A unit ALU
bit(ST0,k4) = #0	Yes	2	1	X	
bit(ST0,k4) = #1	Yes	2	1	X	
bit(ST1,k4) = #0	Yes	2	1	X	
bit(ST1,k4) = #1	Yes	2	1	X	
bit(ST2,k4) = #0	Yes	2	1	X	
bit(ST2,k4) = #1	Yes	2	1	X	
bit(ST3,k4) = #0	Yes	2	1†	X	
bit(ST3,k4) = #1	Yes	2	1†	X	

† When these instructions are 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.

3.3 Extended Auxiliary Register (XAR) Operations

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
Extended Auxiliary Register Move (page 4-227)				
xdst = xsrc	No	2	1	X
Load Effective Address to Extended Auxiliary Register (page 4-256)				
XAdst = mar(Smem)	No	3	1	AD
XAdst = k23	No	6	1	AD
Load Extended Auxiliary Register from Memory (page 4-257)				
XAdst = dbl(Lmem)	No	3	1	X
Pop Extended Auxiliary Register from Stack Pointers (page 4-355)				
xdst = popboth()	Yes	2	1	X
Push Extended Auxiliary Register to Stack Pointers (page 4-363)				
pshboth(xsrc)	Yes	2	1	X
Store Extended Auxiliary Register to Memory (page 4-455)				
dbl(Lmem) = XAsrc	No	3	1	X

3.4 Logical Operations

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
Bit Field Counting (page 4-92)					
Tx = count(ACx,ACy,TCx)	Yes	3	1	X	D unit shifter and A unit ALU
Bitwise Complement (page 4-95)					
dst = ~src	Yes	2	1	X	A or D unit ALU
Bitwise AND (page 4-96)					
dst = dst & src	Yes	2	1	X	A or D unit ALU
dst = src & k8	Yes	3	1	X	A or D unit ALU
dst = src & k16	No	4	1	X	A or D unit ALU
dst = src & Smem	No	3	1	X	A or D unit ALU
ACy = ACy & (ACx <<< #SHIFTW)	Yes	3	1	X	D unit shifter
ACy = ACx & (k16 <<< #16)	No	4	1	X	D unit ALU
ACy = ACx & (k16 <<< #SHFT)	No	4	1	X	D unit shifter
Smem = Smem & k16	No	4	1	X	A unit ALU
Bitwise OR (page 4-105)					
dst = dst | src	Yes	2	1	X	A or D unit ALU
dst = src | k8	Yes	3	1	X	A or D unit ALU
dst = src | k16	No	4	1	X	A or D unit ALU
dst = src | Smem	No	3	1	X	A or D unit ALU
ACy = ACy | (ACx <<< #SHIFTW)	Yes	3	1	X	D unit shifter
ACy = ACx | (k16 <<< #16)	No	4	1	X	D unit ALU
ACy = ACx | (k16 <<< #SHFT)	No	4	1	X	D unit shifter
Smem = Smem | k16	No	4	1	X	A unit ALU

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
Bitwise XOR (page 4-114)					
dst = dst ^ src	Yes	2	1	X	A or D unit ALU
dst = src ^ k8	Yes	3	1	X	A or D unit ALU
dst = src ^ k16	No	4	1	X	A or D unit ALU
dst = src ^ Smem	No	3	1	X	A or D unit ALU
ACy = ACy ^ (ACx <<< #SHIFTW)	Yes	3	1	X	D unit shifter
ACy = ACx ^ (k16 <<< #16)	No	4	1	X	D unit ALU
ACy = ACx ^ (k16 <<< #SHFT)	No	4	1	X	D unit shifter
Smem = Smem ^ k16	No	4	1	X	A unit ALU
Logical Shift (page 4-258)					
dst = dst <<< #1	Yes	2	1	X	A unit ALU or D unit shifter
dst = dst >>> #1	Yes	2	1	X	A unit ALU or D unit shifter
ACy = ACx <<< Tx	Yes	2	1	X	D unit shifter
ACy = ACx <<< #SHIFTW	Yes	3	1	X	D unit shifter
Negation (page 4-343)					
dst = –src	Yes	2	1	X	A or D unit ALU
Rotate Left (page 4-410)					
dst = BitOut \\ src \\ BitIn	Yes	3	1	X	A unit ALU or D unit shifter
Rotate Right (page 4-412)					
dst = BitIn // src // BitOut	Yes	3	1	X	A unit ALU or D unit shifter

3.5 Miscellaneous Operations

Syntax	Size	Cycles	Pipeline
Linear/Circular Addressing Qualifiers (page 4-253)			
linear()	1	1	AD
circular()	1	1	AD
Memory-Mapped Register Access Qualifier (page 4-268)			
mmap()	1	1	D
Peripheral Port Register Access Qualifiers (page 4-350)			
readport()	1	1	D
writeport()	1	1	D

3.6 Move Operations

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
Accumulator, Address, or Data Register Content Swap (page 4-7)					
swap(ARx, Tx)	Yes	2	1	AD	A unit register file
swap(Tx, Ty)	Yes	2	1	AD	A unit register file
swap(ARx, ARy)	Yes	2	1	AD	A unit register file
swap(ACx, ACy)	Yes	2	1	X	D unit register file
swap(pair(ARx), pair(Tx))	Yes	2	1	AD	A unit register file
swap(pair(T0), pair(T2))	Yes	2	1	AD	A unit register file
swap(pair(AR0), pair(AR2))	Yes	2	1	AD	A unit register file
swap(pair(AC0), pair(AC2))	Yes	2	1	X	D unit register file
swap(block(AR4), block(T0))	Yes	2	1	AD	A unit register file
Accumulator, Address, or Data Register Load (page 4-19)					
dst = k4	Yes	2	1	X	A or D unit register file
dst = –k4	Yes	2	1	X	A or D unit register file
dst = K16	No	4	1	X	A or D unit register file
dst = Smem	No	2	1	X	A or D unit register file
dst = uns(high_byte(Smem))	No	3	1	X	A or D unit register file
dst = uns(low_byte(Smem))	No	3	1	X	A or D unit register file
ACx = K16 << #16	No	4	1	X	D unit ALU
ACx = K16 << #SHFT	No	4	1	X	D unit shifter
ACx = rnd(Smem << Tx)	No	3	1	X	D unit shifter
ACx = low_byte(Smem) << #SHIFTW	No	3	1	X	D unit shifter

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
ACx = high_byte(Smem) << #SHIFTW	No	3	1	X	D unit shifter
ACx = Smem << #16	No	2	1	X	D unit ALU
ACx = uns(Smem)	No	3	1	X	D unit register file
ACx = uns(Smem) << #SHIFTW	No	4	1	X	D unit shifter
ACx = M40(dbl(Lmem))	No	3	1	X	D unit register file
LO(ACx) = Xmem, HI(ACx) = Ymem	No	3	1	1	D unit register file
pair(HI(ACx)) = Lmem	No	3	1	X	D unit register file
pair(LO(ACx)) = Lmem	No	3	1	X	D unit register file
pair(TAx) = Lmem	No	3	1	X	A unit register file
Accumulator, Address, or Data Register Move (page 4-41)					
dst = src	Yes	2	1	X	A or D unit ALU
TAx = HI(ACx)	Yes	2	1	X	A unit ALU
HI(ACx) = TAx	Yes	2	1	X	D unit ALU
Accumulator, Address, or Data Register Store (page 4-46)					
Smem = src	No	2	1	X	A or D unit register file
high_byte(Smem) = src	No	3	1	X	A or D unit register file
low_byte(Smem) = src	No	3	1	X	A or D unit register file
Smem = HI(ACx)	No	2	1	X	D unit register file
Smem = HI(rnd(ACx))	No	3	1	X	D unit shifter
Smem = LO(ACx << Tx)	No	3	1	X	D unit shifter
Smem = HI(rnd(ACx << Tx))	No	3	1	X	D unit shifter
Smem = LO(ACx << #SHIFTW)	No	3	1	X	D unit shifter
Smem = HI(ACx << #SHIFTW)	No	3	1	X	D unit shifter

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
Smem = HI(rnd(ACx << #SHIFTW))	No	4	1	X	D unit shifter
Smem = HI(saturate(uns(rnd(ACx))))	No	3	1	X	D unit shifter
Smem = HI(saturate(uns(rnd(ACx << Tx))))	No	3	1	X	D unit shifter
Smem = HI(saturate(uns(rnd(ACx << #SHIFTW))))	No	4	1	X	D unit shifter
dbl(Lmem) = ACx	No	3	1	X	D unit register file
dbl(Lmem) = saturate(uns(ACx))	No	3	1	X	D unit shifter
Lmem = pair(HI(ACx))	No	3	1	X	D unit register file
Lmem = pair(LO(ACx))	No	3	1	X	D unit register file
Lmem = pair(TAx)	No	3	1	X	A unit register file
HI(Lmem) = HI(ACx) >> #1, LO(Lmem) = LO(ACx) >> #1	No	3	1	X	D unit shifter
Xmem = LO(ACx), Ymem = HI(ACx)	No	3	1	X	D unit register file
Memory Delay (page 4-279)					
delay(Smem)	No	2	1	X	A or D unit register file
Memory-to-Memory Move/Memory Initialization (page 4-280)					A or D unit register file
Smem = Cmem	No	3	1	X	
Cmem = Smem	No	3	1	X	
Smem = K8	No	3	1	X	
Smem = K16	No	4	1	X	
Lmem = dbl(Cmem)	No	3	1	X	
dbl(Cmem) = Lmem	No	3	1	X	
dbl(Ymem) = dbl(Xmem)	No	3	1	X	
Ymem = Xmem	No	3	1	X	

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
Pop Top of Stack (TOS) (page 4-356)					A or D unit register file
dst1,dst2 = pop()	Yes	2	1	X	
dst = pop()	Yes	2	1	X	
dst,Smem = pop()	No	3	1	X	
ACx = dbl(pop())	Yes	2	1	X	
Smem = pop()	No	2	1	X	
dbl(Lmem) = pop()	No	2	1	X	
Push to Top of Stack (TOS) (page 4-364)					A or D unit register file
push(src1,src2)	Yes	2	1	X	
push(src)	Yes	2	1	X	
push(src,Smem)	No	3	1	X	
dbl(push(ACx))	Yes	2	1	X	
push(Smem)	No	2	1	X	
push(dbl(Lmem))	No	2	1	X	
Specific CPU Register Load (page 4-436)					A or D unit register file
BK03 = k12	Yes	3	1	AD	
BK47 = k12	Yes	3	1	AD	
BKC = k12	Yes	3	1	AD	
BRC0 = k12	Yes	3	1	AD	
BRC1 = k12	Yes	3	1	AD	
CSR = k12	Yes	3	1	AD	
MDP = k7	Yes	3	1	AD	
PDP = k9	Yes	3	1	AD	
BOF01 = k16	No	4	1	AD	
BOF23 = k16	No	4	1	AD	
BOF45 = k16	No	4	1	AD	
BOF67 = k16	No	4	1	AD	
BOFC = k16	No	4	1	AD	
CDP = k16	No	4	1	AD	

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
DP = k16	No	4	1	AD	A or D unit register file
SP = k16	No	4	1	AD	
SSP = k16	No	4	1	AD	
BK03 = Smem	No	3	1	X	
BK47 = Smem	No	3	1	X	
BKC = Smem	No	3	1	X	
BOF01 = Smem	No	3	1	X	
BOF23 = Smem	No	3	1	X	
BOF45 = Smem	No	3	1	X	
BOF67 = Smem	No	3	1	X	
BOFC = Smem	No	3	1	X	
BRC0 = Smem	No	3	1	X	
BRC1 = Smem	No	3	1	X	
CDP = Smem	No	3	1	X	
CSR = Smem	No	3	1	X	
DP = Smem	No	3	1	X	
MDP = Smem	No	3	1	X	
PDP = Smem	No	3	1	X	
SP = Smem	No	3	1	X	
SSP = Smem	No	3	1	X	
TRN0 = Smem	No	3	1	X	
TRN1 = Smem	No	3	1	X	
RETA = dbl(Lmem)	No	3	5	X	
Specific CPU Register Move (page 4-442)					A unit ALU
BRC0 = TAx	Yes	2	1	X	
BRC1 = TAx	Yes	2	1	X	
CDP = TAx	Yes	2	1	X	
CSR = TAx	Yes	2	1	X	
SP = TAx	Yes	2	1	X	
SSP = TAx	Yes	2	1	X	
TAx = BRC0	Yes	2	1	X	

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
TAx = BRC1	Yes	2	1	X	A unit ALU
TAx = CDP	Yes	2	1	X	
TAx = RPTC	Yes	2	1	X	
TAx = SP	Yes	2	1	X	
TAx = SSP	Yes	2	1	X	
Specific CPU Register Store (page 4-446)					A or D unit register file
Smem = BK03	No	3	1	X	
Smem = BK47	No	3	1	X	
Smem = BKC	No	3	1	X	
Smem = BOF01	No	3	1	X	
Smem = BOF23	No	3	1	X	
Smem = BOF45	No	3	1	X	
Smem = BOF67	No	3	1	X	
Smem = BOFC	No	3	1	X	
Smem = BRC0	No	3	1	X	
Smem = BRC1	No	3	1	X	
Smem = CDP	No	3	1	X	
Smem = CSR	No	3	1	X	
Smem = DP	No	3	1	X	
Smem = MDP	No	3	1	X	
Smem = PDP	No	3	1	X	
Smem = SP	No	3	1	X	
Smem = SSP	No	3	1	X	
Smem = TRN0	No	3	1	X	
Smem = TRN1	No	3	1	X	
dbl(Lmem) = RETA	No	3	5	X	

3.7 Program Control Operations

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
Branch Conditionally (page 4-123)				
if (cond) goto l4	No	2	6/5	R
if (cond) goto L8	Yes	3	6/5	R
if (cond) goto L16	No	4	6/5	R
if (cond) goto P24	No	5	5/5	R
x/y cycles: x cycles = condition true, y cycles = condition false				
Branch Unconditionally (page 4-130)				
goto ACx	No	2	10	X
goto L7	Yes	2	6†	AD
goto L16	Yes	3	6†	AD
goto P24	No	4	5	D
† These instructions execute in 3 cycles if the addressed instruction is in the instruction buffer unit.				
Branch on Auxiliary Register Not Zero (page 4-134)				
if (ARn_mod != #0) goto L16	No	4	6/5	AD
x/y cycles: x cycles = condition true, y cycles = condition false				
Call Conditionally (page 4-137)				
if (cond) call L16	No	4	6/5	R
if (cond) call P24	No	5	5/5	R
x/y cycles: x cycles = condition true, y cycles = condition false				
Call Unconditionally (page 4-140)				
call ACx	No	2	10	X
call L16	Yes	3	6	AD
call P24	No	4	5	D
Compare and Branch (page 4-144)				
Compare (uns(src RELOP K8)) goto L8	No	4	7/6	X
x/y cycles: x cycles = condition true, y cycles = condition false				

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
Execute Conditionally (page 4-220)				
if (cond) execute(AD_Unit)	No	2	1	AD
if (cond) execute(D_Unit)	No	2	1	X
Idle (page 4-233)				
idle	No	4	?	D
No Operation (NOP) (page 4-345)				
nop	Yes	1	1	D
nop_16	Yes	2	1	D
Repeat Block of Instructions Unconditionally (page 4-389)				
localrepeat{}	Yes	2	1	AD
blockrepeat{}	Yes	3	1	AD
Repeat Single Instruction Conditionally (page 4-395)				
while (cond && (RPTC < k8)) repeat	Yes	3	1	AD
Repeat Single Instruction Unconditionally (page 4-398)				
repeat(CSR)	Yes	2	1	AD
repeat(CSR), CSR += TAx	Yes	2	1	X
repeat(k8)	Yes	2	1	AD
repeat(CSR), CSR += k4	Yes	2	1	X
repeat(CSR), CSR –= k4	Yes	2	1	X
repeat(k16)	Yes	3	1	AD
Return Conditionally (page 4-406)				
if (cond) return x/y cycles: x cycles = condition true, y cycles = condition false	Yes	3	5/5	R
Return Unconditionally (page 4-408)				
return	Yes	2	5	D

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
Return from Interrupt (page 4-409)				
return_int	Yes	2	5	D
Software Interrupt (page 4-431)				
intr(k5)	No	2	3	D
Software Reset (page 4-433)				
reset	No	2	?	D
Software Trap (page 4-434)				
trap(k5)	No	2	?	D

Chapter 4

Instruction Set Descriptions

This chapter provides detailed information on the TMS320C55x™ 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 3 for a summary of the instruction set. See Chapter 5 for the opcode of each instruction syntax.

4.1 Absolute Distance (abdst)

Syntax Characteristics

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

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 cleared to 0, compatibility is ensured.

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

Status Bits

Affected by FRCT, C54CM, M40, SATD, SXMD

Affects ACOVx, ACOVy, CARRY

Repeat

This instruction can be repeated.

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

4.2 Absolute Value

Syntax Characteristics

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

Operands dst, src

Description

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

- When the destination register (dst) is an accumulator (ACx):
  - 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 (ACx) 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 (TAx):
  - 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 (TAx) 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 cleared to 0.

Status Bits

Affected by C54CM, M40, SATA, SATD, SXMD

Affects ACOVx, CARRY

Repeat

This instruction can be repeated.

Examples

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

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

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

Syntax

AC1 = |AR1|

Description

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

Syntax

T1 = |AC0|

Description

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

Syntax

T1 = |AC0|

Description

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

4.3 Accumulator, Auxiliary, or Temporary Register Content Swap

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[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
[5]	swap(T0, T2)	Yes	2	1	AD
[6]	swap(T1, T3)	Yes	2	1	AD
[7]	swap(AR0, AR2)	Yes	2	1	AD
[8]	swap(AR1, AR3)	Yes	2	1	AD
[9]	swap(AR0, AR1)	Yes	2	1	AD
[10]	swap(AC0, AC2)	Yes	2	1	X
[11]	swap(AC1, AC3)	Yes	2	1	X
[12]	swap(pair(AR4), pair(T0))	Yes	2	1	AD
[13]	swap(pair(AR6), pair(T2))	Yes	2	1	AD
[14]	swap(pair(T0), pair(T2))	Yes	2	1	AD
[15]	swap(pair(AR0), pair(AR2))	Yes	2	1	AD
[16]	swap(pair(AC0), pair(AC2))	Yes	2	1	X
[17]	swap(block(AR4), block(T0))	Yes	2	1	AD

Brief Description

These instructions perform parallel moves between accumulators, auxiliary registers, or temporary registers. These operations are performed in a dedicated datapath independent of the A-unit operators and D-unit operators.

These instructions allow you to move the content of the first accumulator, auxiliary or temporary register to the second accumulator, auxiliary or temporary register, and reciprocally to move the content of the second register to the first register.

Auxiliary and temporary register swapping is performed in the address phase of the pipeline. Accumulator swapping is performed in the execute phase of the pipeline.

Status Bits

Affected by none

Affects none

4.3.1 Auxiliary and Temporary Register Content Swap: swap(ARx, Tx)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[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

Operands

ARx, Tx

Description

This instruction performs parallel moves between auxiliary registers and temporary registers. These operations are performed in a dedicated datapath 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 temporary register to the auxiliary register.

Auxiliary and temporary register swapping is performed in the address phase of the pipeline.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
swap(AR4, T0)	The content of AR4 is moved to T0 and the content of T0 is moved to AR4.

```
Before              After
T0      6500        T0      0300
AR4     0300        AR4     6500
```

4.3.2 Temporary Register Content Swap: swap(Tx, Ty)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	swap(T0, T2)	Yes	2	1	AD
[6]	swap(T1, T3)	Yes	2	1	AD

Operands

Tx

Description

This instruction performs parallel moves between two temporary registers. These operations are performed in a dedicated datapath independent of the A-unit operators.

This instruction moves the content of the first temporary register (Tx) to the second temporary register (Ty), and reciprocally moves the content of the second temporary register to the first temporary register.

temporary register swapping is performed in the address phase of the pipeline.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

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	T0	0300
T2	0300	T2	6500

4.3.3 Auxiliary Register Content Swap: swap(ARx, ARy)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	swap(AR0, AR2)	Yes	2	1	AD
[8]	swap(AR1, AR3)	Yes	2	1	AD
[9]	swap(AR0, AR1)	Yes	2	1	AD

Operands

ARx

Description

This instruction performs parallel moves between two auxiliary registers. These operations are performed in a dedicated datapath independent of the A-unit operators.

This instruction moves the content of the first auxiliary register (ARx) to the second auxiliary register (ARy), and reciprocally moves the content of the second auxiliary register to the first auxiliary register.

Auxiliary register swapping is performed in the address phase of the pipeline.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

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
```

4.3.4 Accumulator Content Swap: swap(ACx, ACy)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[10]	swap(AC0, AC2)	Yes	2	1	X
[11]	swap(AC1, AC3)	Yes	2	1	X

Operands

ACx

Description

This instruction performs parallel moves between two accumulators. These operations are performed in a dedicated datapath independent of the D-unit operators.

This instruction moves the content of the first accumulator (ACx) to the second accumulator (ACy), and reciprocally moves the content of the second accumulator to the first accumulator.

Accumulator swapping is performed in the execute phase of the pipeline.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

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
```

4.3.5 Auxiliary and Temporary Register Pair Content Swap: swap(pair(AR4), pair(T0))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[12]	swap(pair(AR4), pair(T0))	Yes	2	1	AD

Operands

AR4, T0

Description

This instruction performs two parallel moves between two auxiliary registers (AR4 and AR5) and two temporary registers (T0 and T1) in one cycle. These operations are performed in a dedicated datapath independent of the A-unit operators.

This instruction performs two parallel moves:

- the content of AR4 to T0, and reciprocally the content of T0 to AR4
- the content of AR5 to T1, and reciprocally the content of T1 to AR5

Auxiliary and temporary register swapping is performed in the address phase of the pipeline.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

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
T0	6788	T0	0200
T1	0200	T1	0300

4.3.6 Auxiliary and Temporary Register Pair Content Swap: swap(pair(AR6), pair(T2))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[13]	swap(pair(AR6), pair(T2))	Yes	2	1	AD

Operands

AR6, T2

Description

This instruction performs two parallel moves between two auxiliary registers (AR6 and AR7) and two temporary registers (T2 and T3) in one cycle. These operations are performed in a dedicated datapath independent of the A-unit operators.

This instruction performs two parallel moves:

- the content of AR6 to T2, and reciprocally the content of T2 to AR6
- the content of AR7 to T3, and reciprocally the content of T3 to AR7

Auxiliary and temporary register swapping is performed in the address phase of the pipeline.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
swap(pair(AR6), pair(T2))	The following two swap instructions are performed in parallel: 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	
AR6	0200	AR6	6788
AR7	0300	AR7	0200
T2	6788	T2	0200
T3	0200	T3	0300

4.3.7 Temporary Register Pair Content Swap: swap(pair(T0), pair(T2))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[14]	swap(pair(T0), pair(T2))	Yes	2	1	AD

Operands Tx

Description

This instruction performs two parallel moves between four temporary registers (T0 and T2, T1 and T3) in one cycle. These operations are performed in a dedicated datapath independent of the A-unit operators.

This instruction performs two parallel moves:

- ❏ the content of T0 to T2, and reciprocally the content of T2 to T0
- ❏ the content of T1 to T3, and reciprocally the content of T3 to T1

temporary register swapping is performed in the address phase of the pipeline.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

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.

Before		After	
T0	0200	T0	6788
T1	0300	T1	0200
T2	6788	T2	0200
T3	0200	T3	0300

4.3.8 Auxiliary Register Pair Content Swap: swap(pair(AR0), pair(AR2))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[15]	swap(pair(AR0), pair(AR2))	Yes	2	1	AD

Operands

ARx

Description

This instruction performs two parallel moves between four auxiliary registers (AR0 and AR2, AR1 and AR3) in one cycle. These operations are performed in a dedicated datapath independent of the A-unit operators.

This instruction performs two parallel moves:

- the content of AR0 to AR2, and reciprocally the content of AR2 to AR0
- the content of AR1 to AR3, and reciprocally the content of AR3 to AR1

Auxiliary register swapping is performed in the address phase of the pipeline.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

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

4.3.9 Accumulator Pair Content Swap: swap(pair(AC0), pair(AC2))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[16]	swap(pair(AC0), pair(AC2))	Yes	2	1	X

Operands

ACx

Description

This instruction performs two parallel moves between four accumulators (AC0 and AC2, AC1 and AC3) in one cycle. These operations are performed in a dedicated datapath independent of the D-unit operators.

This instruction performs two parallel moves:

- ❑ the content of AC0 to AC2, and reciprocally the content of AC2 to AC0
- ❑ the content of AC1 to AC3, and reciprocally the content of AC3 to AC1

Accumulator swapping is performed in the execute phase of the pipeline.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

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.

```
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
```

4.3.10 Auxiliary and Temporary Register Content Swap: swap(block(AR4), block(T0))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[17]	swap(block(AR4), block(T0))	Yes	2	1	AD

Operands AR4, T0

Description

This instruction performs four parallel moves between four auxiliary registers (AR4, AR5, AR6, and AR7) and four temporary registers (T0, T1, T2, and T3) in one cycle. These operations are performed in a dedicated datapath independent of the A-unit operators.

This instruction performs four parallel moves:

- the content of AR4 to T0, and reciprocally the content of T0 to AR4
- the content of AR5 to T1, and reciprocally the content of T1 to AR5
- the content of AR6 to T2, and reciprocally the content of T2 to AR6
- the content of AR7 to T3, and reciprocally the content of T3 to AR7

Auxiliary and temporary register swapping is performed in the address phase of the pipeline.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

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
T3	0FD3	T3	0400

4.4 Accumulator, Auxiliary, or Temporary Register Load

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	X
[4]	dst = Smem	No	2	1	X
[5]	dst = uns(high_byte(Smem))	No	3	1	X
[6]	dst = uns(low_byte(Smem))	No	3	1	X
[7]	ACx = K16 << #16	No	4	1	X
[8]	ACx = K16 << #SHFT	No	4	1	X
[9]	ACx = rnd(Smem << Tx)	No	3	1	X
[10]	ACx = low_byte(Smem) << #SHIFTW	No	3	1	X
[11]	ACx = high_byte(Smem) << #SHIFTW	No	3	1	X
[12]	ACx = Smem << #16	No	2	1	X
[13]	ACx = uns(Smem)	No	3	1	X
[14]	ACx = uns(Smem) << #SHIFTW	No	4	1	X
[15]	ACx = M40(dbl(Lmem))	No	3	1	X
[16]	LO(ACx) = Xmem, HI(ACx) = Ymem	No	3	1	X
[17]	pair(HI(ACx)) = Lmem	No	3	1	X
[18]	pair(LO(ACx)) = Lmem	No	3	1	X
[19]	pair(TAx) = Lmem	No	3	1	X

Brief Description

This instruction loads a 4-bit unsigned constant, k4, 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 destination (dst) register.

Status Bits

Affected by C54CM, M40, RDM, SATD, SXMD

Affects ACOVx

4.4.1 Accumulator, Auxiliary, or Temporary Register Load: dst = k4

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = k4	Yes	2	1	X

Operands dst, k4

Description

This instruction loads the 4-bit unsigned constant, k4, to the destination (dst) register.

- When the destination register is an accumulator:
  - The 4-bit constant, k4, is zero extended to 40 bits.
  - 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.
  - 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 none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = #2	AC0 is loaded with the unsigned 4-bit value (2).

4.4.2 Accumulator, Auxiliary, or Temporary Register Load: dst = –k4

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	dst = –k4	Yes	2	1	X

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 none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = #2	AC0 is loaded with a 2s complement representation of the unsigned 4-bit value (2).

4.4.3 Accumulator, Auxiliary, or Temporary Register Load: dst = K16

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	dst = K16	No	4	1	X

Operands dst, K16

Description

This instruction loads the 16-bit signed constant, K16, to the destination (dst) register.

- When the destination register is an accumulator, the 16-bit constant, K16, is sign extended to 40 bits according to SXMD.
- When the destination register is an auxiliary or temporary register, 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 SXMD

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC1 = #248	AC1 is loaded with the signed 16-bit value (248).

```
Before                          After
AC1         00 0200 FC00        AC1         00 0000 00F8
```

4.4.4 Accumulator, Auxiliary, or Temporary Register Load: dst = Smem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	dst = Smem	No	2	1	X

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 SXMD

Affects none

Repeat

This instruction can be repeated.

Example

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

4.4.5 Accumulator, Auxiliary, or Temporary Register Load: dst = uns(high_byte(Smem))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	dst = uns(high_byte(Smem))	No	3	1	X

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 content of the memory location is zero extended to 40 bits, if the optional uns keyword is applied to the input operand.
  - The content of the memory location is sign extended to 40 bits according to SXMD, if the optional uns keyword is not applied to the input operand.
  - 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 zero extended to 16 bits, if the optional uns keyword is applied to the input operand.
  - The content of the memory location is sign extended to 16 bits regardless of SXMD, if the optional uns keyword is not applied to the input operand.
  - 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 SXMD

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = uns(high_byte(*AR3))	The high-byte content addressed by AR3 is sign extended to 40 bits and loaded into AC0.

4.4.6 Accumulator, Auxiliary, or Temporary Register Load: dst = uns(low_byte(Smem))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	dst = uns(low_byte(Smem))	No	3	1	X

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 content of the memory location is zero extended to 40 bits, if the optional uns keyword is applied to the input operand.
  - The content of the memory location is sign extended to 40 bits according to SXMD, if the optional uns keyword is not applied to the input operand.
  - 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 zero extended to 16 bits, if the optional uns keyword is applied to the input operand.
  - The content of the memory location is sign extended to 16 bits regardless of SXMD, if the optional uns keyword is not applied to the input operand.
  - 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 SXMD

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = uns(low_byte(*AR3))	The low-byte content addressed by AR3 is sign extended to 40 bits and loaded into AC0.

4.4.7 Accumulator Load: ACx = K16 << #16

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	ACx = K16 << #16	No	4	1	X

Operands ACx, K16

Description

This instruction loads the 16-bit signed constant, K16, shifted left by 16 bits to the accumulator (ACx):

- The 16-bit constant, K16, is sign extended to 40 bits according to SXMD.
- The shift operation is identical 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 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 (–2) shifted left by 16 bits.

4.4.8 Accumulator Load: ACx = K16 << #SHFT

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	ACx = K16 << #SHFT	No	4	1	X

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 identical 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 SXMD

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = #–2 << #15	AC0 is loaded with the signed 16-bit value (–2) shifted left by 15 bits.

4.4.9 Accumulator Load: ACx = rnd(Smem << Tx)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[9]	ACx = rnd(Smem << Tx)	No	3	1	X

Operands ACx, Tx, Smem

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 identical 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 ACOVx

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = (*AR3 << T0)	AC0 is loaded with the content addressed by AR3 shifted by the content of T0.

4.4.10 Accumulator Load: ACx = low_byte(Smem) << #SHIFTW

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[10]	ACx = low_byte(Smem) << #SHIFTW	No	3	1	X

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 identical 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 ACOVx

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = low_byte(*AR3) << #31	The low-byte content addressed by AR3 is shifted left by 31 bits and loaded into AC0.

4.4.11 Accumulator Load: ACx = high_byte(Smem) << #SHIFTW

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[11]	ACx = high_byte(Smem) << #SHIFTW	No	3	1	X

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 identical 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 ACOVx

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = high_byte(*AR3) << #31	The high-byte content addressed by AR3 is shifted left by 31 bits and loaded into AC0.

4.4.12 Accumulator Load: ACx = Smem << #16

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[12]	ACx = Smem << #16	No	2	1	X

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 identical 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 M40, SATD, SXMD

Affects ACOVx

Repeat

This instruction can be repeated.

Example

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
```

4.4.13 Accumulator Load: ACx = uns(Smem)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[13]	ACx = uns(Smem)	No	3	1	X

Operands ACx, Smem

Description

This instruction loads the content of a memory (Smem) location to the accumulator (ACx):

- The content of the memory location is zero extended to 40 bits, if the optional uns keyword is applied to the input operand.
- The content of the memory location is sign extended to 40 bits according to SXMD, if the optional uns keyword is not applied to the input operand.
- 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.

Example

Syntax	Description
AC0 = uns(*AR3)	The content addressed by AR3 is zero extended to 40 bits and loaded into AC0.

4.4.14 Accumulator Load: ACx = uns(Smem) << #SHIFTW

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[14]	ACx = uns(Smem) << #SHIFTW	No	4	1	X

Operands ACx, SHIFTW, Smem

Description

This instruction loads the content of a memory (Smem) location, shifted by the 6-bit value, SHIFTW, to the to the accumulator (ACx):

- The content of the memory location is zero extended to 40 bits, if the optional uns keyword is applied to the input operand.
- The content of the memory location is sign extended to 40 bits according to SXMD, if the optional uns keyword is not applied to the input operand.
- The input operand is shifted by the 6-bit value in the D-unit shifter. The shift operation is identical 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 ACOVx

Repeat

This instruction cannot be repeated.

Example

Syntax	Description
AC0 = uns(*AR3) << #31	The content addressed by AR3 is zero extended to 40 bits, shifted left by 31 bits, and loaded into AC0.

4.4.15 Accumulator Load: ACx = M40(dbl(Lmem))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[15]	ACx = M40(dbl(Lmem))	No	3	1	X

Operands

ACx, Lmem

Description

This instruction loads the content of data memory operand (Lmem) to the accumulator (ACx):

- The input operand 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.
- Status bit M40 is locally set to 1, if the optional M40 keyword is applied to the input operand.

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

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = (dbl(*AR3–))	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.

4.4.16 Accumulator Load: LO(ACx) = Xmem, HI(ACx) = Ymem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[16]	LO(ACx) = Xmem, HI(ACx) = Ymem	No	3	1	X

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 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 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 7FFFh.

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, SATD, SXMD

Affects ACOVx

Repeat

This instruction can be repeated.

Example

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).

4.4.17 Accumulator Load: pair(HI(ACx)) = Lmem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[17]	pair(HI(ACx)) = Lmem	No	3	1	X

Operands

ACx, Lmem

Description

This instruction loads the 16 highest bits of data memory operand (Lmem) to the 16 highest bits of the accumulator (ACx) and loads the 16 lowest bits of data memory operand (Lmem) to the 16 highest bits of accumulator AC(x + 1):

- 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.
- Valid accumulator destinations are AC0 and AC2.

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 operation.

Status Bits

Affected by M40, SATD, SXMD

Affects ACOVx, ACOV(x + 1)

Repeat

This instruction can be repeated.

Example

Syntax	Description
pair(HI(AC2)) = *AR3+	The 16 highest bits of the content at the location addressed by AR3 are loaded into AC2(31–16) and the 16 lowest bits of the content at the location addressed by AR3 + 1 are loaded into AC3(31–16). AR3 is incremented by 1.

```
Before                          After
AC2         00 0200 FC00        AC2         00 3400 0000
AC3         00 0000 0000        AC3         00 0FD3 0000
AR3                 0200        AR3                 0201
200                 3400        200                 3400
201                 0FD3        201                 0FD3
```

4.4.18 Accumulator Load: pair(LO(ACx)) = Lmem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[18]	pair(LO(ACx)) = Lmem	No	3	1	X

Operands

ACx, Lmem

Description

This instruction loads the 16 highest bits of data memory operand (Lmem) to the 16 lowest bits of the accumulator (ACx) and loads the 16 lowest bits of data memory operand (Lmem) to the 16 lowest bits of accumulator AC(x + 1):

- 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.
- Valid accumulator destinations are AC0 and AC2.

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.

Example

Syntax	Description
pair(LO(AC0)) = *AR3	The 16 highest bits of the content at the location addressed by AR3 are loaded into AC0(15–0) and the 16 lowest bits of the content at the location addressed by AR3 + 1 are loaded into AC1(15–0).

4.4.19 Auxiliary or Temporary Register Load: pair(TAx) = Lmem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[19]	pair(TAx) = Lmem	No	3	1	X

Operands

TAx, Lmem

Description

This instruction loads the 16 highest bits of data memory operand (Lmem) to the data or auxiliary register (TAx) and loads the 16 lowest bits of data memory operand (Lmem) to data or auxiliary register TA(x + 1):

- The load operation in the data or auxiliary register uses a dedicated path independent of the A-unit ALU.
- Valid auxiliary register destinations are AR0, AR2, AR4, and AR6.
- Valid temporary register destinations 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 none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
pair(T0) = *AR3	The 16 highest bits of the content at the location addressed by AR3 are loaded into T0 and the 16 lowest bits of the content at the location addressed by AR3 + 1 are loaded into T1.

4.5 Accumulator, Auxiliary, or Temporary Register Move

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = src	Yes	2	1	X
[2]	TAx = HI(ACx)	Yes	2	1	X
[3]	HI(ACx) = TAx	Yes	2	1	X

Brief Description

This instruction moves:

- ❑ the content of the source (src) register to the destination (dst) register.
- ❑ the 16 MSBs of the selected accumulator (ACx) to the destination auxiliary or temporary register (TAx).
- ❑ the content of the auxiliary or temporary register (TAx) to the high part of the selected accumulator (ACx).

When the destination register is an accumulator, the 40-bit move operation is performed in the D-unit ALU. When the destination register is an auxiliary or temporary register, the 16-bit move operation is performed in the A-unit ALU.

Status Bits

Affected by M40, SATD, SXMD

Affects ACOVx

4.5.1 Accumulator, Auxiliary, or Temporary Register Move: dst = src

Syntax Characteristics

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

Operands dst, src

Description

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 low bits 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 register 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

Repeat

This instruction can be repeated.

Example

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

4.5.2 Accumulator Move: TAx = HI(ACx)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	TAx = HI(ACx)	Yes	2	1	X

Operands

ACx, TAx

Description

This instruction moves the 16 MSBs of the accumulator (ACx) 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 none

Affects none

Repeat

This instruction can be repeated.

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
```

4.5.3 Auxiliary or Temporary Register Move: HI(ACx) = TAx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	HI(ACx) = TAx	Yes	2	1	X

Operands

ACx, TAx

Description

This instruction moves the content of the auxiliary or temporary register (TAx) to the high part of the accumulator (ACx):

- The 16-bit move operation is performed in the D-unit ALU.
- During the 16-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 low bits of the source register are sign extended to 40 bits according to SXMD.

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

Repeat

This instruction can be repeated.

Example

Syntax	Description
HI(AC0) = T0	The content of T0 is copied to AC0(31–16).

4.6 Accumulator, Auxiliary, or Temporary Register Store

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	Smem = src	No	2	1	X
[2]	high_byte(Smem) = src	No	3	1	X
[3]	low_byte(Smem) = src	No	3	1	X
[4]	Smem = HI(ACx)	No	2	1	X
[5]	Smem = HI(rnd(ACx))	No	3	1	X
[6]	Smem = LO(ACx << Tx)	No	3	1	X
[7]	Smem = HI(rnd(ACx << Tx))	No	3	1	X
[8]	Smem = LO(ACx << #SHIFTW)	No	3	1	X
[9]	Smem = HI(ACx << #SHIFTW)	No	3	1	X
[10]	Smem = HI(rnd(ACx << #SHIFTW))	No	4	1	X
[11]	Smem = HI(saturate(uns(rnd(ACx))))	No	3	1	X
[12]	Smem = HI(saturate(uns(rnd(ACx << Tx))))	No	3	1	X
[13]	Smem = HI(saturate(uns(rnd(ACx << #SHIFTW))))	No	4	1	X
[14]	dbl(Lmem) = ACx	No	3	1	X
[15]	dbl(Lmem) = saturate(uns(ACx))	No	3	1	X
[16]	HI(Lmem) = HI(ACx) >> #1, LO(Lmem) = LO(ACx) >> #1	No	3	1	X
[17]	Lmem = pair(HI(ACx))	No	3	1	X
[18]	Lmem = pair(LO(ACx))	No	3	1	X
[19]	Lmem = pair(TAx)	No	3	1	X
[20]	Xmem = LO(ACx), Ymem = HI(ACx)	No	3	1	X

Brief Description

This instruction stores the content of the selected source (src) register 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

4.6.1 Accumulator, Auxiliary, or Temporary Register Store: Smem = src

Syntax Characteristics

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

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.

Example

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
```

4.6.2 Accumulator, Auxiliary, or Temporary Register Store: high_byte(Smem) = src

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	high_byte(Smem) = src	No	3	1	X

Operands

Smem, src

Description

This instruction stores the low part (bits 7–0) content of the source (src) register to the high byte of the memory (Smem) location.

- When the source register is an accumulator:
  - The low part of the accumulator, ACx(7–0), is stored to the high 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 high byte of 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.

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
```

4.6.3 Accumulator, Auxiliary, or Temporary Register Store: low_byte(Smem) = src

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	low_byte(Smem) = src	No	3	1	X

Operands

Smem, src

Description

This instruction stores the low part (bits 7–0) content of the source (src) register to the low byte of the memory (Smem) location.

- 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.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
low_byte(*AR3) = AC0	The content of AC0(7–0) is stored in the low byte at the location addressed by AR3.

4.6.4 Accumulator Store: Smem = HI(ACx)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	Smem = HI(ACx)	No	2	1	X

Operands

ACx, Smem

Description

This instruction stores the high part of the accumulator, ACx(31–16), to the memory (Smem) location.

- The high part of the accumulator, ACx(31–16), 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.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
*AR3 = HI(AC0)	The content of AC0(31–16) is stored at the location addressed by AR3.

4.6.5 Accumulator Store: Smem = HI(rnd(ACx))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	Smem = HI(rnd(ACx))	No	3	1	X

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.
- The high part of the accumulator, ACx(31–16), is stored to the memory location.

Status Bits

Affected by RDM

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
*AR3 = HI(rnd(AC0))	The content of AC0(31–16) is rounded and stored at the location addressed by AR3.

4.6.6 Accumulator Store: Smem = LO(ACx << Tx)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	Smem = LO(ACx << Tx)	No	3	1	X

Operands ACx, Tx, Smem

Description

- This instruction shifts the accumulator, ACx, by the content of Tx and stores 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.
- After the shift, the low part of the accumulator, ACx(15–0), is stored to the memory location.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with C54CM = 1:

- 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 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, SXMD

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
*AR3 = LO(AC0 << T0)	The content of AC0 is shifted by the content of T0 and AC0(15–0) is stored at the location addressed by AR3.

4.6.7 Accumulator Store: Smem = HI(rnd(ACx << Tx))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	Smem = HI(rnd(ACx << Tx))	No	3	1	X

Operands ACx, Tx, Smem

Description

This instruction shifts the accumulator, ACx, by the content of Tx and stores 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.
- 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 C54CM = 1:

- 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 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, SXMD

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
*AR3 = HI(rnd(AC0 << T0))	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.

4.6.8 Accumulator Store: Smem = LO(ACx << #SHIFTW)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	Smem = LO(ACx << #SHIFTW)	No	3	1	X

Operands ACx, SHIFTW, Smem

Description

This instruction shifts the accumulator, ACx, by the 6-bit value, SHIFTW, and stores 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.
- After the shift, the low part of the accumulator, ACx(15–0), is stored to the memory location.

Status Bits

Affected by SXMD

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
*AR3 = LO(AC0 << #31)	The content of AC0 is shifted left by 31 bits and AC0(15–0) is stored at the location addressed by AR3.

4.6.9 Accumulator Store: Smem = HI(ACx << #SHIFTW)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[9]	Smem = HI(ACx << #SHIFTW)	No	3	1	X

Operands ACx, SHIFTW, Smem

Description

This instruction shifts the accumulator, ACx, by the 6-bit value, SHIFTW, and stores 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.
- After the shift, the low part of the accumulator, ACx(31–16), is stored to the memory location.

Status Bits

Affected by SXMD

Affects none

Repeat

This instruction can be repeated.

Example

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.

4.6.10 Accumulator Store: Smem = HI(rnd(ACx << #SHIFTW))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[10]	Smem = HI(rnd(ACx << #SHIFTW))	No	4	1	X

Operands

ACx, SHIFTW, Smem

Description

This instruction shifts the accumulator, ACx, by the 6-bit value, SHIFTW, and stores 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.
- Rounding is performed in the D-unit shifter according to RDM, if the optional rnd keyword is applied to the input operand.
- After the shift, the high part of the accumulator, ACx(31–16), is stored to the memory location.

Status Bits

Affected by RDM, SXMD

Affects none

Repeat

This instruction cannot be repeated.

Example

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.

4.6.11 Accumulator Store: Smem = HI(saturate(uns(rnd(ACx))))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[11]	Smem = HI(saturate(uns(rnd(ACx))))	No	3	1	X

Operands ACx, Smem

Description

This instruction stores the high part of the accumulator, ACx(31–16), to the memory (Smem) location.

- If the optional uns keyword is applied to the input operand, the input operand is considered unsigned.
- Rounding is performed in the D-unit shifter according to RDM, if the optional rnd keyword is applied to the input operand.
- 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).
- 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 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:
  - 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, SXMD

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
*AR3 = HI(saturate(uns(rnd(AC0))))	The unsigned content of AC0 is rounded, is saturated, and AC0(31–16) is stored at the location addressed by AR3.

4.6.12 Accumulator Store: Smem = HI(saturate(uns(rnd(ACx << Tx))))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[12]	Smem = HI(saturate(uns(rnd(ACx << Tx))))	No	3	1	X

Operands ACx, Tx, Smem

Description

This instruction shifts the accumulator, ACx, by the content of Tx and stores 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.

- If the optional uns keyword is applied to the input operand, the input operand is considered unsigned.
- 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.
- Rounding is performed in the D-unit shifter according to RDM, if the optional rnd keyword is applied to the input operand.
- 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).
- 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 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 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 are used to 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, SXMD

Affects none

Repeat

This instruction can be repeated.

Example

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.

4.6.13 Accumulator Store: Smem = HI(saturate(uns(rnd(ACx << #SHIFTW))))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[13]	Smem = HI(saturate(uns(rnd(ACx << #SHIFTW))))	No	4	1	X

Operands ACx, SHIFTW, Smem

Description

This instruction shifts the accumulator, ACx, by the 6-bit value, SHIFTW, and stores ACx(31–16) to the memory (Smem) location.

- If the optional uns keyword is applied to the input operand, the input operand is considered unsigned.
- The input operand is shifted by the 6-bit value 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.
- Rounding is performed in the D-unit shifter according to RDM, if the optional rnd keyword is applied to the input operand.
- 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).
- 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 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 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, SXMD

Affects none

Repeat

This instruction cannot be repeated.

Example

Syntax	Description
*AR3 = HI(saturate(uns(rnd(AC0 << #31))))	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.

4.6.14 Accumulator Store: dbl(Lmem) = ACx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[14]	dbl(Lmem) = ACx	No	3	1	X

Operands

ACx, Lmem

Description

This instruction stores the content of the accumulator, ACx(31–0), to the 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.

Status Bits

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

Example

Syntax	Description
dbl(*AR3) = AC0	The content of AC0 is stored at the locations addressed by AR3 and AR3 + 1.

4.6.15 Accumulator Store: dbl(Lmem) = saturate(uns(ACx))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[15]	dbl(Lmem) = saturate(uns(ACx))	No	3	1	X

Operands

ACx, Lmem

Description

This instruction stores the content of the accumulator, ACx(31–0), to the data memory operand (Lmem).

- If the optional uns keyword is applied to the input operand, the input operand is considered unsigned.
- 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).
- 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 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 SXMD

Affects none

Repeat

This instruction can be repeated.

Example

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.

4.6.16 Accumulator Store: HI(Lmem) = HI(ACx) >> #1, LO(Lmem) = LO(ACx) >> #1

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[16]	HI(Lmem) = HI(ACx) >> #1, LO(Lmem) = LO(ACx) >> #1	No	3	1	X

Operands

ACx, Lmem

Description

This instruction is executed in the D-unit shifter:

- The 16 highest bits of the accumulator (ACx), 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 of ACx, 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).

Status Bits

Affected by SXMD

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
HI(*AR1) = HI(AC0) >> #1, 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.

4.6.17 Accumulator Store: Lmem = pair(HI(ACx))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[17]	Lmem = pair(HI(ACx))	No	3	1	X

Operands ACx, Lmem

Description

This instruction stores the 16 highest bits of the accumulator (ACx) to the 16 highest bits of the data memory operand (Lmem) and stores the 16 highest bits of AC(x + 1) to the16 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 and AC2.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

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.

```
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
```

4.6.18 Accumulator Store: Lmem = pair(LO(ACx))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[18]	Lmem = pair(LO(ACx))	No	3	1	X

Operands

ACx, Lmem

Description

This instruction stores the 16 lowest bits of the accumulator (ACx) to the 16 highest bits of the data memory operand (Lmem) and stores the 16 lowest bits of AC(x + 1) to the16 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 and AC2.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

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.

4.6.19 Auxiliary or Temporary Register Store: Lmem = pair(TAx)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[19]	Lmem = pair(TAx)	No	3	1	X

Operands

TAx, Lmem

Description

This instruction stores the content of the data 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 register destinations are AR0, AR2, AR4, and AR6.
- Valid temporary register destinations are T0 and T2.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
*AR3 = pair(T0)	The content of T0 is stored at the location addressed by AR3 and the content of T1 is stored at the location addressed by AR3 + 1.

4.6.20 Accumulator Store: Xmem = LO(ACx), Ymem = HI(ACx)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[20]	Xmem = LO(ACx), Ymem = HI(ACx)	No	3	1	X

Operands

ACx, Xmem, Ymem

Description

This instruction stores the 16 lowest bits of the accumulator (ACx) to data memory operand Xmem and stores the 16 highest bits of ACx to data memory operand Ymem.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
*AR1 = LO(AC0), *AR2 = HI(AC0)	The content of AC0(15–0) is stored at the location addressed by AR1 and the 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

4.7 Addition

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = dst + src	Yes	2	1	X
[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 << Tx)	Yes	2	1	X
[6]	ACy = ACy + (ACx << #SHIFTW)	Yes	3	1	X
[7]	ACy = ACx + (K16 << #16)	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 + dbl(Lmem)	No	3	1	X
[15]	ACx = (Xmem << #16) + (Ymem << #16)	No	3	1	X
[16]	Smem = Smem + K16	No	4	1	X
[17]	ACy = rnd(ACy + |ACx|)	Yes	2	1	X

Brief Description

These instructions perform an addition operation:

- ❑ In the D-unit ALU, if the destination operand is an accumulator (ACx).
- ❑ In the A-unit ALU, if the destination operand is an auxiliary or temporary register (TAx).
- ❑ In the D-unit ALU, if the destination operand is the memory (Smem).
- ❑ In the D-unit shifter, if the instruction has a shift quantity other than the immediate 16 bit shift.

Status Bits

Affected by CARRY, C54CM, M40, SATA, SATD, SXMD

Affects ACOVx, ACOVy, CARRY

4.7.1 Addition: dst = dst + src

Syntax Characteristics

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

Operands dst, src

Description

This instruction performs an addition operation between two registers contents.

- 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.

4.7.2 Addition: dst = dst + k4

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	dst = dst + k4	Yes	2	1	X

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.

Example

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

4.7.3 Addition: dst = src + K16

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	dst = src + K16	No	4	1	X

Operands dst, K16, src

Description

This instruction performs an addition operation between a register content and a 16-bit signed constant, K16.

- 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 16-bit constant, K16, 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.

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.

4.7.4 Addition: dst = src + Smem

Syntax Characteristics

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

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.

Example

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	0

4.7.5 Addition: ACy = ACy + (ACx << Tx)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	ACy = ACy + (ACx << Tx)	Yes	2	1	X

Operands ACx, ACy, Tx

Description

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 identical 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
- 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.

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.

4.7.6 Addition: ACy = ACy + (ACx << #SHIFTW)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	ACy = ACy + (ACx << #SHIFTW)	Yes	3	1	X

Operands ACx, ACy, SHIFTW

Description

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 40 bits in the D-unit shifter.
- Input operands are sign extended to 40 bits according to SXMD.
- The shift operation is identical 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.

Example

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

4.7.7 Addition: ACy = ACx +(K16 << #16)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	ACy = ACx + (K16 << #16)	No	4	1	X

Operands ACx, ACy, K16

Description

This instruction performs an addition operation between an accumulator content ACx and the 16-bit signed constant, K16, 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 identical 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.

Example

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.

4.7.8 Addition: ACy = ACx + (K16 << #SHFT)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	ACy = ACx + (K16 << #SHFT)	No	4	1	X

Operands ACx, ACy, K16, SHFT

Description

This instruction performs an addition operation between an accumulator content ACx and the 16-bit signed constant, K16, shifted left by the 4-bit value, SHFT.

- 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 identical 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 M40, SATD, SXMD

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

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.

4.7.9 Addition: ACy = ACx + (Smem << Tx)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[9]	ACy = ACx + (Smem << Tx)	No	3	1	X

Operands ACx, ACy, Tx, Smem

Description

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 identical 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
- 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.

Example

Syntax

AC0 = AC1 + (*AR1 << T0)

Description

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

4.7.10 Addition: ACy = ACx + (Smem << #16)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[10]	ACy = ACx + (Smem << #16)	No	3	1	X

Operands ACx, ACy, Smem

Description

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 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 identical to the signed shift instruction.
- Overflow detection and CARRY status bit depends on M40. If the result of the addition generates a carry, the CARRY status bit is set; 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 C54CM, M40, SATD, SXMD

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

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.

4.7.11 Addition: ACy = ACx + uns(Smem) + CARRY

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[11]	ACy = ACx + uns(Smem) + CARRY	No	3	1	X

Operands ACx, ACy, Smem

Description

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 operation is performed on 40 bits in the D-unit ALU.
- Input operands are sign extended to 40 bits according to SXMD.
  - The content of the memory location is zero extended to 40 bits, if the optional uns keyword is applied to the input operand.
  - The content of the memory location is sign extended to 40 bits according to SXMD, if the optional uns keyword is not applied to the input operand.
- 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.

Status Bits

Affected by CARRY, M40, SATD, SXMD

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

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.

4.7.12 Addition: ACy = ACx + uns(Smem)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[12]	ACy = ACx + uns(Smem)	No	3	1	X

Operands ACx, ACy, Smem

Description

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

- The operation is performed on 40 bits in the D-unit ALU.
- Input operands are sign extended to 40 bits according to SXMD.
  - The content of the memory location is zero extended to 40 bits, if the optional uns keyword is applied to the input operand.
  - The content of the memory location is sign extended to 40 bits according to SXMD, if the optional uns keyword is not applied to the input operand.
- 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.

Status Bits

Affected by M40, SATD, SXMD

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

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.

4.7.13 Addition: ACy = ACx + (uns(Smem) << #SHIFTW)

Syntax Characteristics

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

Operands 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 sign extended to 40 bits according to SXMD.
  - The content of the memory location is zero extended to 40 bits, if the optional uns keyword is applied to the input operand.
  - The content of the memory location is sign extended to 40 bits according to SXMD, if the optional uns keyword is not applied to the input operand.
- The shift operation is identical 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 cannot be repeated.

Example

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.

4.7.14 Addition: ACy = ACx + dbl(Lmem)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[14]	ACy = ACx + dbl(Lmem)	No	3	1	X

Operands ACx, ACy, Lmem

Description

This instruction performs an addition operation between the accumulator content ACx and the content of data memory operand dbl(Lmem).

- 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
- 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.
- 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

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 + dbl(*AR3+)	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.

4.7.15 Addition: ACx = (Xmem << #16) + (Ymem << #16)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[15]	ACx = (Xmem << #16) + (Ymem << #16)	No	3	1	X

Operands ACx, Xmem, Ymem

Description

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 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 identical 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 ACOVx, CARRY

Repeat

This instruction can be repeated.

Example

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.

4.7.16 Addition: Smem = Smem + K16

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[16]	Smem = Smem + K16	No	4	1	X

Operands K16, Smem

Description

This instruction performs an addition operation between a 16-bit signed constant, K16, and the content of a memory (Smem) location.

- The operation is performed on 40 bits in the D-unit ALU.
- Input operands are sign extended to 40 bits according to SXMD and shifted by 16 bits to the MSBs before being added.
- 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.

Compatibility with C54x devices (C54CM = 1)

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

Status Bits

Affected by SATD, SXMD

Affects ACOV0, CARRY

Repeat

This instruction cannot be repeated.

Example

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.

4.7.17 Addition: ACy = rnd(ACy + |ACx|)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[17]	ACy = rnd(ACy + |ACx|)	Yes	2	1	X

Operands

ACx, ACy

Description

This instruction is performed in the D-unit MAC:

- The absolute value of accumulator ACx is computed by multiplying ACx(32–16) by 00001h or 1FFFFh depending on bit 32 of the source accumulator.
- If FRCT = 1, the absolute value is multiplied by 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 is set.
- When an addition overflow is detected, the accumulator is saturated according to SATD.
- The result of the absolute value of the higher part of the source accumulator is in the lower part of the destination accumulator.

Compatibility with C54x devices (C54CM = 1)

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

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

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

4.8 Bit Field Comparison

Syntax Characteristics

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

Operands k16, Smem

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:

```
if( ((Smem) AND k16 ) == 0)
    TCx = 0
else
    TCx = 1
```

Status Bits

Affected by none

Affects TCx

Repeat

This instruction cannot be repeated.

Examples

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	
*AR0	0040	*AR0	0040
TC1	0	TC1	1

Syntax	Description
TC2 = *AR3 & #00A0h	The unsigned 16-bit value (00A0h) is ANDed with the content addressed by AR3. The result is 0, TC2 is cleared to 0.

Before		After	
*AR3	0040	*AR3	0040
TC2	0	TC2	0

4.9 Bit Field Counting

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	Tx = count(ACx, ACy, TCx)	Yes	3	1	X

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.

Examples

Syntax	Description
T1 = count(AC1, AC2, TC1)	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       0F E340 5678
T1                  0000      T1                000B
TC1                    0      TC1                  1
```

4.10 Bit Field Expand

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = field_expand(ACx, k16)	No	4	1	X

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 (EFC) 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.

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
AC0(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
```

4.11 Bit Field Extract

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = field_extract(ACx, k16)	No	4	1	X

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 (EFC) 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.

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

```
#k16 (8024h)          1000 0000 0010 0100
AC0(15–0)             0101 0101 1010 1010
T2                    0000 0000 0000 0010
```

```
Before                          After
AC0         00 2300 55AA        AC0         00 2300 55AA
T2                  0000        T2                  0002
```

4.12 Bitwise Complement (NOT)

Syntax Characteristics

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

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.

Example

Syntax	Description
AC1 = ~AC0	The content of AC0 is complemented and the result is stored in AC1.

```
Before                          After
AC0          7E 2355 4FC0       AC0          7E 2355 4FC0
AC1          00 2300 5678       AC1          81 DCAA B03F
```

4.13 Bitwise AND

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

Brief 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, M40

Affects none

4.13.1 Bitwise AND: dst = dst & src

Syntax Characteristics

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

Operands

dst, src

Description

This instruction performs a bitwise AND operation between two registers content.

- 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 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
```

4.13.2 Bitwise AND: dst = src & k8

Syntax Characteristics

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

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.

Example

Syntax	Description
AC0 = AC1 & #00FFh	The content of AC1 is ANDed with the unsigned 8-bit value (FFh) and the result is stored in AC0.

4.13.3 Bitwise AND: dst = src & k16

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	dst = src & k16	No	4	1	X

Operands dst, k16, src

Description

This instruction performs a bitwise AND operation between a source (src) register content and a 16-bit value, k16.

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

4.13.4 Bitwise AND: dst = src & Smem

Syntax Characteristics

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

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.

Example

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

4.13.5 Bitwise AND: ACy = ACy & (ACx <<< #SHIFTW)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	ACy = ACy & (ACx <<< #SHIFTW)	Yes	3	1	X

Operands

ACx, ACy, SHIFTW

Description

This instruction performs a bitwise AND operation between an accumulator (ACy) content and a shifted accumulator (ACx) content.

- The shift and AND operations are performed in one cycle in the D-unit shifter.
- Input operands are zero extended to 40 bits.
- The input operand (ACx) is shifted by an immediate value in the D-unit shifter.
- 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 locally set to 1. The 8 upper bits of the 40-bit intermediary result are not cleared.

Status Bits

Affected by C54CM, M40

Affects none

Repeat

This instruction can be repeated.

Example

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

4.13.6 Bitwise AND: ACy = ACx & (k16 <<< #16)

Syntax Characteristics

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

Operands ACx, ACy, k16

Description

This instruction performs a bitwise AND operation between an accumulator (ACx) content and a shifted 16-bit value, k16.

- ❏ 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.

Example

Syntax	Description
AC0 = AC1 & (#FFFFh <<< #16)	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.

4.13.7 Bitwise AND: ACy = ACx & (k16 <<< #SHFT)

Syntax Characteristics

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

Operands

ACx, ACy, k16, SHFT

Description

This instruction performs a bitwise AND operation between an accumulator (ACx) content and a shifted 16-bit value, k16.

- The shift and AND operations are performed in one cycle in the D-unit shifter.
- Input operands are zero extended to 40 bits.
- The input operand (k16) is shifted by an immediate value in the D-unit shifter.
- The CARRY status bit is not affected by the logical shift operation.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

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.

4.13.8 Bitwise AND: Smem = Smem & k16

Syntax Characteristics

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

Operands k16, Smem

Description

This instruction performs a bitwise AND operation between a memory (Smem) location and a 16-bit value, 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 cannot be repeated.

Example

Syntax	Description
*AR1 = *AR1 & #0FC0	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

4.14 Bitwise OR

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

Brief 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, M40

Affects none

4.14.1 Bitwise OR: dst = dst | src

Syntax Characteristics

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

Operands

dst, src

Description

This instruction performs a bitwise OR operation between two registers content.

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

4.14.2 Bitwise OR: dst = src | k8

Syntax Characteristics

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

Operands dst, k8, src

Description

This instruction performs a bitwise OR 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.

Example

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.

4.14.3 Bitwise OR: dst = src | k16

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	dst = src | k16	No	4	1	X

Operands dst, k16, src

Description

This instruction performs a bitwise OR operation between a source (src) register content and a 16-bit value, k16.

- 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
AC0 = AC1 | #FFFFh	The content of AC1 is ORed with the unsigned 16-bit value (FFFFh) and the result is stored in AC0.

4.14.4 Bitwise OR: dst = src | Smem

Syntax Characteristics

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

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.

Example

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

4.14.5 Bitwise OR: ACy = ACy | (ACx <<< #SHIFTW)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	ACy = ACy | (ACx <<< #SHIFTW)	Yes	3	1	X

Operands

ACx, ACy, SHIFTW

Description

This instruction performs a bitwise OR operation between an accumulator (ACy) content and a shifted accumulator (ACx) content.

- The shift and OR operations are performed in one cycle in the D-unit shifter.
- Input operands are zero extended to 40 bits.
- The input operand (ACx) is shifted by an immediate value in the D-unit shifter.
- 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 locally set to 1. The 8 upper bits of the 40-bit intermediary result are not cleared.

Status Bits

Affected by C54CM, M40

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC1 = AC1 | (AC0 <<< #4)	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
```

4.14.6 Bitwise OR: ACy = ACx | (k16 <<< #16)

Syntax Characteristics

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

Operands

ACx, ACy, k16

Description

This instruction performs a bitwise OR operation between an accumulator (ACx) content and a shifted 16-bit value, k16.

- 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.

Example

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.

4.14.7 Bitwise OR: ACy = ACx | (k16 <<< #SHFT)

Syntax Characteristics

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

Operands

ACx, ACy, k16, SHFT

Description

This instruction performs a bitwise OR operation between an accumulator (ACx) content and a shifted 16-bit value, k16.

- The shift and OR operations are performed in one cycle in the D-unit shifter.
- Input operands are zero extended to 40 bits.
- The input operand (k16) is shifted by an immediate value in the D-unit shifter.
- The CARRY status bit is not affected by the logical shift operation

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

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

4.14.8 Bitwise OR: Smem = Smem | k16

Syntax Characteristics

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

Operands k16, Smem

Description

This instruction performs a bitwise OR operation between a memory (Smem) location and a 16-bit value, 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 cannot 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
```

4.15 Bitwise XOR

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

Brief 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, M40

Affects none

4.15.1 Bitwise XOR: dst = dst ^ src

Syntax Characteristics

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

Operands

dst, src

Description

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

- 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
```

4.15.2 Bitwise XOR: dst = src ^ k8

Syntax Characteristics

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

Operands dst, k8, src

Description

This instruction performs a bitwise exclusive-OR (XOR) 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.

Example

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

4.15.3 Bitwise XOR: dst = src ^ k16

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	dst = src ^ k16	No	4	1	X

Operands dst, k16, src

Description

This instruction performs a bitwise exclusive-OR (XOR) operation between a source (src) register content and a 16-bit value, k16.

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

4.15.4 Bitwise XOR: dst = src ^ Smem

Syntax Characteristics

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

Operands dst, Smem, src

Description

This instruction performs a bitwise exclusive-OR (XOR) 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.

Example

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

4.15.5 Bitwise XOR: ACy = ACy ^ (ACx <<< #SHIFTW)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	ACy = ACy ^ (ACx <<< #SHIFTW)	Yes	3	1	X

Operands

ACx, ACy, SHIFTW

Description

This instruction performs a bitwise exclusive-OR (XOR) operation between an accumulator (ACy) content and a shifted accumulator (ACx) content.

- The shift and XOR operations are performed in one cycle in the D-unit shifter.
- Input operands are zero extended to 40 bits.
- The input operand (ACx) is shifted by an immediate value in the D-unit shifter.
- 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 locally set to 1. The 8 upper bits of the 40-bit intermediary result are not cleared.

Status Bits

Affected by C54CM, M40

Affects none

Repeat

This instruction can be repeated.

Example

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.

4.15.6 Bitwise XOR: ACy = ACx ^ (k16 <<< #16)

Syntax Characteristics

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

Operands

ACx, ACy, k16

Description

This instruction performs a bitwise exclusive-OR (XOR) operation between an accumulator (ACx) content and a shifted 16-bit value, k16.

- 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.

Example

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.

4.15.7 Bitwise XOR: ACy = ACx ^ (k16 <<< #SHFT)

Syntax Characteristics

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

Operands

ACx, ACy, k16, SHFT

Description

This instruction performs a bitwise exclusive-OR (XOR) operation between an accumulator (ACx) content and a shifted 16-bit value, k16.

- The shift and XOR operations are performed in one cycle in the D-unit shifter.
- Input operands are zero extended to 40 bits.
- The input operand (k16) is shifted by an immediate value in the D-unit shifter.
- The CARRY status bit is not affected by the logical shift operation.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

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.

4.15.8 Bitwise XOR: Smem = Smem ^ k16

Syntax Characteristics

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

Operands k16, Smem

Description

This instruction performs a bitwise exclusive-OR (XOR) operation between a memory (Smem) location and a 16-bit value, 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 cannot be repeated.

Example

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

4.16 Branch Conditionally

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	if (cond) goto l4	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

x/y cycles: x cycles = condition true, y cycles = condition false

Brief 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 l4, 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.

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

4.16.1 Branch Conditionally: if (cond) goto l4

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	if (cond) goto l4	No	2	6/5	R

x/y cycles: x cycles = condition true, y cycles = condition false

Operands

cond, l4

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 l4. 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.

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.

Examples

Syntax	Description
if (*AR0 != #0) goto branch	The content of AR0 is not equal to 0, control is passed to the program address label defined by branch.

```
        if (*AR0 != #0) goto branch
        … …                                  address:  004057
        … …
branch  ……                                           00405A
:
```

Before		After	
AR0	3000	AR0	3000
PC	004055	PC	00405A

Syntax	Description
if (TC1) goto branch	TC1 is set to 1, control is passed to the program address label defined by branch.

```
        if (TC1) goto branch
        … …                                  address:  00406C
        … …
branch  … …                                           00406E
:
```

Before		After	
TC1	1	TC1	1
PC	00406A	PC	00406E

4.16.2 Branch Conditionally: if (cond) goto Lx

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

x/y cycles: x cycles = condition true, y cycles = condition false

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.

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.

Examples

Syntax	Description
if (*AR0 != #0) goto branch	The content of AR0 is not equal to 0, control is passed to the program address label defined by branch.

```
branch  ……                                          00305A
:
        if (*AR0 != #0) goto branch
        … …                                address:  004057
        … …
```

Before		After	
AR0	3000	AR0	3000
PC	004055	PC	00305A

Syntax	Description
if (TC1) goto branch	TC1 is set to 1, control is passed to the program address label defined by branch.

```
branch  … …                                          00306E
:
        if (TC1) goto branch
        … …                                address:  00406C
        … …
```

Before		After	
TC1	1	TC1	1
PC	00406A	PC	00306E

4.16.3 Branch Conditionally: if (cond) goto P24

Syntax Characteristics

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

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.

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.

Examples

Syntax	Description
if (*AR0 != #0) goto branch	The content of AR0 is not equal to 0, control is passed to the program address label defined by branch.

```
        .sect “code1”
        … …
        if (*AR0 != #0) goto branch
        … …                                  address:  004057
        .sect “code2”
branch  ……                                          00F05A
:
```

Before		After	
AR0	3000	AR0	3000
PC	004055	PC	00F05A

Syntax	Description
if (TC1) goto branch	TC1 is set to 1, control is passed to the program address label defined by branch.

```
        .sect “code1”
        if (TC1) goto branch
        … …                                  address:  00406C
        .sect “code2”
branch  … …                                          00F06E
:
```

Before		After	
TC1	1	TC1	1
PC	00406A	PC	00F06E

4.17 Branch Unconditionally

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

† Instructions [2] and [3] execute in 3 cycles if the addressed instruction is in the instruction buffer unit.

Brief 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

4.17.1 Branch Unconditionally: goto ACx

Syntax Characteristics

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

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
```

4.17.2 Branch Unconditionally: goto Lx

Syntax Characteristics

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

† Executes in 3 cycles if the addressed instruction is in the instruction buffer unit.

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 = #1                          address:  004044
        … …
branch  … …                                         006047
:
        AC0 = #0
```

```
Before                      After
PC              004042      PC              006047
AC0       00 0000 0001      AC0       00 0000 0000
```

4.17.3 Branch Unconditionally: goto P24

Syntax Characteristics

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

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                          address:  004044
        AC0 = #1
        … …
branch  … …                                           006047
:
        AC0 = #0
```

Before		After	
PC	004042	PC	006047
AC0	00 0000 0001	AC0	00 0000 0000

4.18 Branch on Auxiliary Register Not Zero

Syntax Characteristics

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

x/y cycles: x cycles = condition true, y cycles = condition false

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 none

Affects none

Repeat

This instruction cannot be repeated.

Examples

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	00400C

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
:
```

Before		**After**	
AR3	0000	AR3	FFFF
PC	00400F	PC	004013

4.19 Call Conditionally

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

x/y cycles: x cycles = condition true, y cycles = condition false

Brief 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.

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

4.19.1 Call Conditionally: if (cond) call L16

Syntax Characteristics

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

x/y cycles: x cycles = condition true, y cycles = condition false

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.

If a subroutine call occurs:

- The stack pointer (SP) is decremented by 1 word in the address phase of the pipeline. The 16 LSBs of RETA (return auxiliary register) are pushed to the top of SP.
- The system stack pointer (SSP) is decremented by 1 word. The 8 MSBs of RETA and the control flow execution context flags register (CFCT) are pushed to the top of SSP.
- The return address of the subroutine is saved in RETA. The active control flow execution context flags are saved in CFCT.
- The program counter (PC) is loaded with the subroutine program address. The active control flow execution context flags are cleared.

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 (AC1 >= #2000h) call (subroutine)	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.

4.19.2 Call Conditionally: if (cond) call P24

Syntax Characteristics

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

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 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.

If a subroutine call occurs:

- The stack pointer (SP) is decremented by 1 word in the address phase of the pipeline. The 16 LSBs of RETA (return auxiliary register) are pushed to the top of SP.
- The system stack pointer (SSP) is decremented by 1 word. The 8 MSBs of RETA and the control flow execution context flags register (CFCT) are pushed to the top of SSP.
- The return address of the subroutine is saved in RETA. The active control flow execution context flags are saved in CFCT.
- The program counter (PC) is loaded with the subroutine program address. The active control flow execution context flags are cleared.

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 (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.

4.20 Call Unconditionally

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

Brief 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.

These instructions cannot be repeated.

Status Bits

Affected by none

Affects none

4.20.1 Call Unconditionally: call ACx

Syntax Characteristics

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

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.

- The stack pointer (SP) is decremented by 1 word in the address phase of the pipeline. The 16 LSBs of RETA (return auxiliary register) 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 8 MSBs of RETA and the control flow execution context flags register (CFCT) are pushed to the top of SSP.
- The return address of the called subroutine is saved in RETA. The active control flow execution context flags are saved in CFCT.
- The PC is loaded with the subroutine program address. The active control flow execution context flags are cleared.

Status Bits

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

Example

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

4.20.2 Call Unconditionally: call L16

Syntax Characteristics

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

Operands

L16

Description

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

- The stack pointer (SP) is decremented by 1 word in the address phase of the pipeline. The 16 LSBs of RETA (return auxiliary register) 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 8 MSBs of RETA and the control flow execution context flags register (CFCT) are pushed to the top of SSP.
- The return address of the called subroutine is saved in RETA. The active control flow execution context flags are saved in CFCT.
- The PC is loaded with the subroutine program address. The active control flow execution context flags are cleared.

Status Bits

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

Example

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

4.20.3 Call Unconditionally: call P24

Syntax Characteristics

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

Operands

P24

Description

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

- The stack pointer (SP) is decremented by 1 word in the address phase of the pipeline. The 16 LSBs of RETA (return auxiliary register) 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 8 MSBs of RETA and the control flow execution context flags register (CFCT) are pushed to the top of SSP.
- The return address of the called subroutine is saved in RETA. The active control flow execution context flags are saved in CFCT.
- The PC is loaded with the subroutine program address. The active control flow execution context flags are cleared.

Status Bits

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

Example

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.

4.21 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	X

x/y cycles: x cycles = condition true, y cycles = condition false

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 unsigned 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 can be repeated.

Examples

Syntax	Description
compare (AC0 >= #12) goto branch	The content of AC0 is compared to the signed 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
if (T1 != #1) goto branch	The content of T1 is not equal to 1, program control is passed to the next instruction (the branch is not taken).

```
        if (T1 != #1) goto branch
        … …                                  address:  00407D
        … …
branch  … …                                            004080
:
```

Before		After	
T1	0000	T1	0000
PC	4079	PC	407D

4.22 Compare and Select Extremum

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	max_diff(ACx, ACy, ACz, ACw)	Yes	3	1	X
[2]	max_diff_dbl(ACx, ACy, ACz, ACw, TRNx)	Yes	3	1	X
[3]	min_diff(ACx, ACy, ACz, ACw)	Yes	3	1	X
[4]	min_diff_dbl(ACx, ACy, ACz, ACw, TRNx)	Yes	3	1	X

Brief Description

Instructions [1] and [3] perform two paralleled 16-bit extremum selections in the D-unit ALU. Instructions [2] and [4] perform a single 40-bit extremum selection in the D-unit ALU.

Status Bits

Affected by C54CM, M40, SATD

Affects ACOVw, CARRY

4.22.1 Compare and Select Extremum: max_diff(ACx, ACy, ACz, ACw)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	max_diff(ACx, ACy, ACz, ACw)	Yes	3	1	X

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 dual 16-bit arithmetic operation 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 7FFFh (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.

Example

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

4.22.2 Compare and Select Extremum: max_diff_dbl(ACx, ACy, ACz, ACw, TRNx)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	max_diff_dbl(ACx, ACy, ACz, ACw, TRNx)	Yes	3	1	X

Operands ACw, ACx, ACy, ACz, TRNx

Description

This instruction performs a single 40-bit extremum selection in the D-unit ALU. This instruction performs a maximum search.

- ACx and ACy are the two source accumulators.
- The difference between the source accumulators is stored in accumulator ACw.
- The subtraction computation is identical to the subtraction 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.
- The extremum between the source accumulators is stored in accumulator ACz.
- The extremum computation is similar to the maximum instruction. However, the CARRY status bit is not updated by the extremum search but by the subtraction instruction.
- According to the extremum found, a decision bit is shifted in TRNx from the MSBs to the LSBs. If the extremum value is ACx, the decision bit is cleared to 0; otherwise, it is set to 1.

  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.

Example

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	0080	TRN1	0040
ACOV3	0	ACOV3	0
CARRY	0	CARRY	0

4.22.3 Compare and Select Extremum: min_diff(ACx, ACy, ACz, ACw)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	min_diff(ACx, ACy, ACz, ACw)	Yes	3	1	X

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 dual 16-bit arithmetic operation 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 7FFFh (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.

Example

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 2400 2222	AC0	10 2400 2222
AC1	00 8000 DDDE	AC1	FF 8000 BBBC
AC2	10 2400 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

4.22.4 Compare and Select Extremum: min_diff_dbl(ACx, ACy, ACz, ACw, TRNx)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	min_diff_dbl(ACx, ACy, ACz, ACw, TRNx)	Yes	3	1	X

Operands ACw, ACx, ACy, ACz, TRNx

Description

This instruction performs a single 40-bit extremum selection in the D-unit ALU. This instruction performs a minimum search.

- ACx and ACy are the two source accumulators.
- The difference between the source accumulators is stored in accumulator ACw.
- The subtraction computation is identical to the subtraction 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.
- The extremum between the source accumulators is stored in accumulator ACz.
- The extremum computation is similar to the maximum instruction. However, the CARRY status bit is not updated by the extremum search but by the subtraction instruction.
- According to the extremum found, a decision bit is shifted in TRNx from the MSBs to the LSBs. If the extremum value is ACx, the decision bit is cleared to 0; otherwise, it is set to 1.

  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.

Example

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.

4.23 Conditional Addition or Subtraction (adsc)

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

Brief Description

This instruction evaluates the selected TCx status bit 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 bit is performed during the Execute phase of the instruction.

Status Bits

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

Affects ACOVy, CARRY

4.23.1 Conditional Addition or Subtraction: ACy = ads2c(Smem, ACx, Tx, TC1, TC2)

Syntax Characteristics

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

Operands ACx, ACy, Tx, Smem, TC1, 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.

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)

If TC2 = 1 and TC1 = 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.

If TC2 = 0 and TC1 = 1, then ACy = 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.

- ❏ 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 identical 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.

If TC2 = 1 and TC1 = 0, 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.

If TC2 = 0 and TC1 = 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.

- 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 identical 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
- 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.

Example

Syntax

AC2 = ads2c(*AR2, AC0, T1, TC1, TC2)

Description

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

4.23.2 Conditional Addition or Subtraction: ACy = adsc(Smem, ACx, TCx)

Syntax Characteristics

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

Operands ACx, ACy, Smem, TCx

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.

If TCx = 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 ALU.
- ❏ Input operands are sign extended to 40 bits according to SXMD.
- ❏ The shift operation is identical 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.

If TCx = 0, 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 ALU.
- ❏ Input operands are sign extended to 40 bits according to SXMD.
- ❏ The shift operation is identical 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.

Examples

Syntax	Description
AC0 = adsc(*AR3, AC1, TC1)	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	M40	0
ACOV1	0	ACOV1	1
CARRY	0	CARRY	1

4.23.3 Conditional Addition or Subtraction: ACy = adsc(Smem, ACx, TC1, TC2)

Syntax Characteristics

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

Operands ACx, ACy, Smem, TC1, TC2

Description

This instruction evaluates the selected TCx status bit 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 bit 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

If 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 (ACOVx) is set.
  - ■ the destination register (ACx) is saturated according to SATD.

If TC2 = 0 and TC1 = 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 ALU.
- ❑ Input operands are sign extended to 40 bits according to SXMD.
- ❑ The shift operation is identical 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.

If TC2 = 0 and TC1 = 0, 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 ALU.
- ❏ Input operands are sign extended to 40 bits according to SXMD.
- ❏ The shift operation is identical 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.

Example

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.

4.24 Conditional Shift (sftc)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = sftc(ACx, TCx)	Yes	2	1	X

Operands ACx, TCx

Description

If the source accumulator ACx(39–0) is equal to 0, this instruction sets the TCx status bit to 1.

If the source accumulator ACx(31–0) has two sign 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 ACx(31–0) does not have two sign bits, this instruction sets the TCx status bit to 1.

The sign bits are extracted at bit positions 31 and 30.

Status Bits

Affected by none

Affects TCx

Repeat

This instruction can be repeated.

Examples

Syntax

AC0 = sftc(AC0, TC1)

Description

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

AC0 = sftc(AC0, TC2)

Description

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 1234 0000   AC0        00 2468 0000
TC2                   0   TC2                   0
```

4.25 Conditional Subtract (subc)

Syntax Characteristics

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

Operands ACx, ACy, Smem

Description

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 identical 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
```

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.

Examples

Syntax
subc(*AR1, AC0, AC1)

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
repeat (CSR)
subc(*AR1, AC1, AC1)

Description
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 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 0746 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

4.26 Dual 16-Bit Arithmetic

	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	HI(ACx) = Smem + Tx, LO(ACx) = Smem – Tx	No	3	1	X
[2]	HI(ACx) = Smem – Tx, LO(ACx) = Smem + Tx	No	3	1	X
[3]	HI(ACy) = HI(Lmem) + HI(ACx), LO(ACy) = LO(Lmem) + LO(ACx)	No	3	1	X
[4]	HI(ACy) = HI(ACx) – HI(Lmem), LO(ACy) = LO(ACx) – LO(Lmem)	No	3	1	X
[5]	HI(ACy) = HI(Lmem) – HI(ACx), LO(ACy) = LO(Lmem) – LO(ACx)	No	3	1	X
[6]	HI(ACx) = Tx – HI(Lmem), LO(ACx) = Tx – LO(Lmem)	No	3	1	X
[7]	HI(ACx) = HI(Lmem) + Tx, LO(ACx) = LO(Lmem) + Tx	No	3	1	X
[8]	HI(ACx) = HI(Lmem) – Tx, LO(ACx) = LO(Lmem) – Tx	No	3	1	X
[9]	HI(ACx) = HI(Lmem) + Tx, LO(ACx) = LO(Lmem) – Tx	No	3	1	X
[10]	HI(ACx) = HI(Lmem) – Tx, LO(ACx) = LO(Lmem) + Tx	No	3	1	X

Brief Description

These instructions perform two paralleled arithmetical operations in one cycle:

- ❑ addition and subtraction
- ❑ subtraction and addition
- ❑ two additions
- ❑ two subtractions

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

4.26.1 Parallel 16-Bit Addition and Subtraction: HI(ACx) = Smem + Tx, LO(ACx) = Smem – Tx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	HI(ACx) = Smem + Tx, LO(ACx) = Smem – Tx	No	3	1	X

Operands ACx, Tx, Smem

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 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).

Status Bits

Affected by C54CM, SATD, SXMD

Affects ACOVx, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
HI(AC1) = *AR1 + T1, LO(AC1) = *AR1 – T1	Both instructions are performed in parallel. The content addressed by AR1 is added 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

4.26.2 Parallel 16-Bit Subtraction and Addition: HI(ACx) = Smem – Tx, LO(ACx) = Smem + Tx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	HI(ACx) = Smem – Tx, LO(ACx) = Smem + Tx	No	3	1	X

Operands ACx, Tx, Smem

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 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).

Status Bits

Affected by C54CM, SATD, SXMD

Affects ACOVx, CARRY

Repeat

This instruction can be repeated.

Example

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).

4.26.3 Parallel 16-Bit Additions: HI(ACy) = HI(Lmem) + HI(ACx), LO(ACy) = LO(Lmem) + LO(ACx)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	HI(ACy) = HI(Lmem) + HI(ACx), LO(ACy) = LO(Lmem) + LO(ACx)	No	3	1	X

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 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).

Status Bits

Affected by C54CM, SATD, SXMD

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

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).

4.26.4 Parallel 16-Bit Subtractions: HI(ACy) = HI(ACx) – HI(Lmem), LO(ACy) = LO(ACx) – LO(Lmem)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	HI(ACy) = HI(ACx) – HI(Lmem), LO(ACy) = LO(ACx) – LO(Lmem)	No	3	1	X

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 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).

Status Bits

Affected by C54CM, SATD, SXMD

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

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).

4.26.5 Parallel 16-Bit Subtractions: HI(ACy) = HI(Lmem) – HI(ACx), LO(ACy) = LO(Lmem) – LO(ACx)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	HI(ACy) = HI(Lmem) – HI(ACx), LO(ACy) = LO(Lmem) – LO(ACx)	No	3	1	X

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 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).

Status Bits

Affected by C54CM, SATD, SXMD

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

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).

4.26.6 Parallel 16-Bit Subtractions: HI(ACx) = Tx – HI(Lmem), LO(ACx) = Tx – LO(Lmem)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	HI(ACx) = Tx – HI(Lmem), LO(ACx) = Tx – LO(Lmem)	No	3	1	X

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 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).

Status Bits

Affected by C54CM, SATD, SXMD

Affects ACOVx, CARRY

Repeat

This instruction can be repeated.

Example

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).

4.26.7 Parallel 16-Bit Additions: HI(ACx) = HI(Lmem) + Tx, LO(ACx) = LO(Lmem) + Tx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	HI(ACx) = HI(Lmem) + Tx, LO(ACx) = LO(Lmem) + Tx	No	3	1	X

Operands ACx, Tx, 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 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 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).

Status Bits

Affected by C54CM, SATD, SXMD

Affects ACOVx, CARRY

Repeat

This instruction can be repeated.

Example

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 added to the content addressed by AR3 + 1 and the result is stored in AC0(15–0).

4.26.8 Parallel 16-Bit Subtractions: HI(ACx) = HI(Lmem) – Tx, LO(ACx) = LO(Lmem) – Tx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	HI(ACx) = HI(Lmem) – Tx, LO(ACx) = LO(Lmem) – Tx	No	3	1	X

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 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).

Status Bits

Affected by C54CM, SATD, SXMD

Affects ACOVx, CARRY

Repeat

This instruction can be repeated.

Example

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).

4.26.9 Parallel 16-Bit Addition and Subtraction: HI(ACx) = HI(Lmem) + Tx, LO(ACx) = LO(Lmem) – Tx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[9]	HI(ACx) = HI(Lmem) + Tx, LO(ACx) = LO(Lmem) – Tx	No	3	1	X

Operands ACx, Tx, Lmem

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 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).

Status Bits

Affected by C54CM, SATD, SXMD

Affects ACOVx, CARRY

Repeat

This instruction can be repeated.

Example

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).

4.26.10 Parallel 16-Bit Subtraction and Addition: HI(ACx) = HI(Lmem) – Tx, LO(ACx) = LO(Lmem) + Tx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[10]	HI(ACx) = HI(Lmem) – Tx, LO(ACx) = LO(Lmem) + Tx	No	3	1	X

Operands ACx, Tx, Lmem

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 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).

Status Bits

Affected by C54CM, SATD, SXMD

Affects ACOVx, CARRY

Repeat

This instruction can be repeated.

Example

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).

4.27 Dual Multiply (Accumulate/Subtract)

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipe-line
[1]	ACx = M40(rnd(uns(Xmem) * uns(Cmem))), ACy = M40(rnd(uns(Ymem) * uns(Cmem)))	No	4	1	X
[2]	ACx = M40(rnd(ACx + (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(uns(Ymem) * uns(Cmem)))	No	4	1	X
[3]	ACx = M40(rnd(ACx – (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(uns(Ymem) * uns(Cmem)))	No	4	1	X
[4]	mar(Xmem), ACx = M40(rnd(uns(Ymem) * uns(Cmem)))	No	4	1	X
[5]	ACx = M40(rnd(ACx + (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(ACy + (uns(Ymem) * uns(Cmem))))	No	4	1	X
[6]	ACx = M40(rnd(ACx – (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(ACy + (uns(Ymem) * uns(Cmem))))	No	4	1	X
[7]	mar(Xmem), ACx = M40(rnd(ACx + (uns(Ymem) * uns(Cmem)))	No	4	1	X
[8]	ACx = M40(rnd(ACx – (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(ACy – (uns(Ymem) * uns(Cmem))))	No	4	1	X
[9]	mar(Xmem), ACx = M40(rnd(ACx – (uns(Ymem) * uns(Cmem))))	No	4	1	X
[10]	ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(ACy + (uns(Ymem) * uns(Cmem))))	No	4	1	X
[11]	ACx = M40(rnd(uns(Xmem) * uns(Cmem))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(Cmem))))	No	4	1	X
[12]	ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(Cmem))))	No	4	1	X
[13]	ACx = M40(rnd(ACx – (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(Cmem))))	No	4	1	X
[14]	mar(Xmem), ACx = M40(rnd((ACx >> #16) + (uns(Ymem) * uns(Cmem))))	No	4	1	X
[15]	mar(Xmem), mar(Ymem), mar(Cmem)	No	4	1	X

Brief Description

These instructions perform the following parallel operations in one cycle:

- ❑ two multiplies
- ❑ multiply and accumulate (MAC), and multiply
- ❑ multiply and subtract (MAS), and multiply
- ❑ modify auxiliary register (MAR) and multiply
- ❑ two multiply and accumulates (MACs)
- ❑ multiply and subtract (MAS), and multiply and accumulate (MAC)

- ❑ modify auxiliary register (MAR), and multiply and accumulate (MAC)
- ❑ two multiply and subtracts (MASs)
- ❑ modify auxiliary register (MAR), and multiply and subtract (MAS)
- ❑ multiply, and multiply and accumulate (MAC)
- ❑ three modify auxiliary registers (MARs)

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVx, ACOVy

4.27.1 Parallel Multiplies: ACx = M40(rnd(uns(Xmem) * uns(Cmem))), ACy = M40(rnd(uns(Ymem) * uns(Cmem)))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = M40(rnd(uns(Xmem) * uns(Cmem))), ACy = M40(rnd(uns(Ymem) * uns(Cmem)))	No	4	1	X

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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign 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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

- ❏ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand. When Xmem memory operand is defined as unsigned, Ymem should also be defined as unsigned (and reciprocally).
- ❏ 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 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, SMUL, M40, RDM, SATD

Affects ACOVx, ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = uns(*AR3) * uns(*CDP), AC1 = uns(*AR4) * uns(*CDP)	Both instructions are performed in parallel. The unsigned content addressed by AR3 is multiplied by the unsigned content addressed by the coefficient data pointer register (CDP) and the result is stored in AC0. The unsigned content addressed by AR4 is multiplied by the unsigned content addressed by CDP and the result is stored in AC1.

4.27.2 Parallel MAC and Multiply: ACx = M40(rnd(ACx + (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(uns(Ymem) * uns(Cmem)))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACx = M40(rnd(ACx + (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(uns(Ymem) * uns(Cmem)))	No	4	1	X

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, sign 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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

- ❏ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand. When Xmem memory operand is defined as unsigned, Ymem should also be defined as unsigned (and reciprocally).
- ❏ If FRCT = 1, the output of the multiplier is shifted 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. 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, SMUL, M40, RDM, SATD

Affects ACOVx, ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = AC0 + (uns(*AR3) * uns(*CDP)), AC1 = uns(*AR4) * uns(*CDP)	Both instructions are performed in parallel. The unsigned content addressed by AR3 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. The unsigned content addressed by AR4 is multiplied by the unsigned content addressed by CDP and the result is stored in AC1.

4.27.3 Parallel MAS and Multiply: ACx = M40(rnd(ACx – (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(uns(Ymem) * uns(Cmem)))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ACx = M40(rnd(ACx – (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(uns(Ymem) * uns(Cmem)))	No	4	1	X

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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign 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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

- ❏ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand. When Xmem memory operand is defined as unsigned, Ymem should also be defined as unsigned (and reciprocally).
- ❏ If FRCT = 1, the output of the multiplier is shifted 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, SMUL, M40, RDM, SATD

Affects ACOVx, ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = AC0 – (uns(*AR3) * uns(*CDP)), AC1 = uns(*AR4) * uns(*CDP)	Both instructions are performed in parallel. The unsigned content addressed by AR3 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. The unsigned content addressed by AR4 is multiplied by the unsigned content addressed by CDP and the result is stored in AC1.

4.27.4 Parallel MAR and Multiply: mar(Xmem), ACx = M40(rnd(uns(Ymem) * uns(Cmem)))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	mar(Xmem), ACx = M40(rnd(uns(Ymem) * uns(Cmem)))	No	4	1	X

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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

- ❑ 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 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, SMUL, M40, RDM, SATD

Affects ACOVx

Repeat

This instruction can be repeated.

Example

Syntax	Description
mar(*AR3+), AC0 = uns(*AR4) * uns(*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 AC0.

4.27.5 Parallel MACs: ACx = M40(rnd(ACx + (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(ACy + (uns(Ymem) * uns(Cmem))))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	ACx = M40(rnd(ACx + (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(ACy + (uns(Ymem) * uns(Cmem))))	No	4	1	X

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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign 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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

- ❏ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand. When Xmem memory operand is defined as unsigned, Ymem should also be defined as unsigned (and reciprocally).
- ❏ 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.
- ❏ 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, SMUL, M40, RDM, SATD

Affects ACOVx, ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = AC0 + (uns(*AR3) * uns(*CDP)), AC1 = AC1 + (uns(*AR4) * uns(*CDP))	Both instructions are performed in parallel. The unsigned content addressed by AR3 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. The unsigned content addressed by AR4 multiplied by the unsigned content addressed by CDP is added to the content of AC1 and the result is stored in AC1.

4.27.6 Parallel MAS and MAC: ACx = M40(rnd(ACx – (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(ACy + (uns(Ymem) * uns(Cmem))))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	ACx = M40(rnd(ACx – (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(ACy + (uns(Ymem) * uns(Cmem))))	No	4	1	X

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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign 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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

- ❏ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand. When Xmem memory operand is defined as unsigned, Ymem should also be defined as unsigned (and reciprocally).
- ❏ If FRCT = 1, the output of the multiplier is shifted 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, SMUL, M40, RDM, SATD

Affects ACOVx, ACOVy

Repeat

This instruction can be repeated.

Example

Syntax

AC0 = M40 (rnd (AC0 – (uns(*AR0) * uns(*CDP)))),
AC1 = M40 (rnd (AC1 + (uns(*AR1) * uns(*CDP))))

Description

Both instructions are performed in parallel. The unsigned content addressed by AR0 multiplied by the unsigned content addressed by the coefficient data pointer register (CDP) is subtracted from the content of AC0. The result is rounded and stored in AC0. The unsigned content addressed by AR1 multiplied by 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

4.27.7 Parallel MAR and MAC: mar(Xmem), ACx = M40(rnd(ACx + (uns(Ymem) * uns(Cmem))))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	mar(Xmem), ACx = M40(rnd(ACx + (uns(Ymem) * uns(Cmem))))	No	4	1	X

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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

- 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 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.
- 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, SMUL, M40, RDM, SATD

Affects ACOVx

Repeat

This instruction can be repeated.

Example

Syntax	Description
mar(*AR3+), AC0 = AC0 + (uns(*AR4) * uns(*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.

4.27.8 Parallel MASs: ACx = M40(rnd(ACx – (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(ACy – (uns(Ymem) * uns(Cmem))))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	ACx = M40(rnd(ACx – (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(ACy – (uns(Ymem) * uns(Cmem))))	No	4	1	X

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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign 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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

- The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand. When Xmem memory operand is defined as unsigned, Ymem should also be defined as unsigned (and reciprocally).
- 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.
- 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, SMUL, M40, RDM, SATD

Affects ACOVx, ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = AC0 – (uns(*AR3) * uns(*CDP)), AC1 = AC1 – (uns(*AR4) * uns(*CDP))	Both instructions are performed in parallel. The unsigned content addressed by AR3 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. The unsigned content addressed by AR4 multiplied by the unsigned content addressed by CDP is subtracted from the content of AC1 and the result is stored in AC1.

4.27.9 Parallel MAR and MAS: mar(Xmem), ACx = M40(rnd(ACx – (uns(Ymem) * uns(Cmem))))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[9]	mar(Xmem), ACx = M40(rnd(ACx – (uns(Ymem) * uns(Cmem))))	No	4	1	X

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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

- ❏ 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 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.
- ❏ 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, SMUL, M40, RDM, SATD

Affects ACOVx

Repeat

This instruction can be repeated.

Example

Syntax	Description
mar(*AR3+), AC0 = AC0 – (uns(*AR4) * uns(*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.

4.27.10 Parallel MACs: ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(ACy + (uns(Ymem) * uns(Cmem))))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[10]	ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(ACy + (uns(Ymem) * uns(Cmem))))	No	4	1	X

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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign 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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

- ❑ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand. When Xmem memory operand is defined as unsigned, Ymem should also be defined as unsigned (and reciprocally).
- ❑ If FRCT = 1, the output of the multiplier is shifted 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 shifted right by 16 bits. The shifting operation is performed with a sign extension of source accumulator ACx(39).
- ❑ 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, SMUL, M40, RDM, SATD

Affects ACOVx, ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = (AC0 >> #16) + (uns(*AR3) * uns(*CDP)), AC1 = AC1 + (uns(*AR4) * uns(*CDP))	Both instructions are performed in parallel. The unsigned content addressed by AR3 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. The unsigned content addressed by AR4 multiplied by the unsigned content addressed by CDP is added to the content of AC1 and the result is stored in AC1.

4.27.11 Parallel Multiply and MAC: ACx = M40(rnd(uns(Xmem) * uns(Cmem))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(Cmem))))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[11]	ACx = M40(rnd(uns(Xmem) * uns(Cmem))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(Cmem))))	No	4	1	X

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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign 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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

- ❏ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand. When Xmem memory operand is defined as unsigned, Ymem should also be defined as unsigned (and reciprocally).
- ❏ If FRCT = 1, the output of the multiplier is shifted 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 shifted right by 16 bits. The shifting operation is performed with a sign extension of source accumulator ACy(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 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, SMUL, M40, RDM, SATD

Affects ACOVx, ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = uns(*AR3) * uns(*CDP), AC1 = (AC1 >> #16) + (uns(*AR4) * uns(*CDP))	Both instructions are performed in parallel. The unsigned content addressed by AR3 is multiplied by the unsigned content addressed by the coefficient data pointer register (CDP) and the result is stored in AC0. The unsigned content addressed by AR4 multiplied by the unsigned content addressed by CDP is added to the content of AC1 shifted right by 16 bits and the result is stored in AC1.

4.27.12 Parallel MACs: ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(Cmem))))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[12]	ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(Cmem))))	No	4	1	X

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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign 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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

- ❏ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand. When Xmem memory operand is defined as unsigned, Ymem should also be defined as unsigned (and reciprocally).
- ❏ 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 shifted 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 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, SMUL, M40, RDM, SATD

Affects ACOVx, ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = (AC0 >> #16) + (uns(*AR3) * uns(*CDP)), AC1 = (AC1 >> #16) + (uns(*AR4) * uns(*CDP))	Both instructions are performed in parallel. The unsigned content addressed by AR3 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. The unsigned content addressed by AR4 multiplied by the unsigned content addressed by CDP is added to the content of AC1 shifted right by 16 bits and the result is stored in AC1.

4.27.13 Parallel MAS and MAC: ACx = M40(rnd(ACx – (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(Cmem))))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[13]	ACx = M40(rnd(ACx – (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(Cmem))))	No	4	1	X

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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign 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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

- ❏ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand. When Xmem memory operand is defined as unsigned, Ymem should also be defined as unsigned (and reciprocally).
- ❏ If FRCT = 1, the output of the multiplier is shifted 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 shifted right by 16 bits. The shifting operation is performed with a sign extension of source accumulator ACy(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 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, SMUL, M40, RDM, SATD

Affects ACOVx, ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	**Description**
AC0 = AC0 – (uns(*AR3) * uns(*CDP)), AC1 = (AC1 >> #16) + (uns(*AR4) * uns(*CDP))	Both instructions are performed in parallel. The unsigned content addressed by AR3 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. The unsigned content addressed by AR4 multiplied by the unsigned content addressed by CDP is added to the content of AC1 shifted right by 16 bits and the result is stored in AC1.

4.27.14 Parallel MAR and MAC: mar(Xmem), ACx = M40(rnd((ACx >> #16) + (uns(Ymem) * uns(Cmem))))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[14]	mar(Xmem), ACx = M40(rnd((ACx >> #16) + (uns(Ymem) * uns(Cmem))))	No	4	1	X

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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

- The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand. When Xmem memory operand is defined as unsigned, Ymem should also be defined as unsigned (and reciprocally).
- 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 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, SMUL, M40, RDM, SATD

Affects ACOVx

Repeat

This instruction can be repeated.

Example

Syntax

mar(*AR2+),
AC0 = ((AC0 >> #16) + (uns(*AR1) * uns(*CDP)))

Description

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		**After**	
AC0	00 6900 0000	AC0	00 95C0 9200
AC1	00 0023 0000	AC1	00 0023 0000
*AR1	EF00	*AR1	EF00
AR2	0201	AR2	0202
*CDP	A067	*CDP	A067
ACOV0	0	ACOV0	1
ACOV1	0	ACOV1	0
CARRY	0	CARRY	0
M40	0	M40	0
FRCT	0	FRCT	0
SATD	0	SATD	0

4.27.15 Parallel MARs: mar(Xmem), mar(Ymem), mar(Cmem)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[15]	mar(Xmem), mar(Ymem), mar(Cmem)	No	4	1	X

Operands

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 operand Xmem
- the content of data memory operand Ymem
- the content of a data memory operand Cmem, addressed using the coefficient addressing mode

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
mar(*AR3+), mar(*AR4–), mar(*CDP)	AR3 is incremented by 1. AR4 is decremented by 1. CDP is not modified.

4.28 Execute Conditionally

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	X

Brief 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. 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 conditions that can be tested are defined by the cond field.

- ❏ These instructions may be executed alone.
- ❏ These instructions may be executed with two paralleled instructions.
- ❏ These instructions may be executed with the instruction with which it is paralleled.
- ❏ These instructions may be executed with the previous instruction.
- ❏ These instructions may be executed with the previous instruction and two paralleled instructions.
- ❏ These instructions cannot be repeated.
- ❏ These instructions cannot control the execution of the following program control instructions:

goto	(cond) goto	intr	return_int	trap
call	(cond) call	idle	(cond) execute(AD_unit)	
return	(cond) return	reset	(cond) execute(D_unit)	
blockrepeat	localrepeat	repeat	while (cond) repeat	

Status Bits

Affected by ACOVx, CARRY, C54CM, M40, TCx

Affects ACOVx

4.28.1 Execute Conditionally (Address Phase): if (cond) execute(AD_unit)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	if (cond) execute(AD_unit)	No	2	1	AD

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.

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 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
label:
```

- 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:

```
      previous_instruction
      || if(cond) execute(AD_unit)
      instruction_1_executes_conditionally
      || instruction_2_executes_conditionally
label:
```

- This instruction cannot control the execution of the following program control instructions:

goto	(cond) goto	intr	return_int	trap
call	(cond) call	idle	(cond) execute(AD_unit)	
return	(cond) return	reset	(cond) execute(D_unit)	
blockrepeat	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.

Examples

Syntax	Description
if (TC1) execute(AD_unit) mar(*AR1+) AC1 = AC1 + *AR1	TC1 is equal to 1, the next instruction is executed (AR1 is incremented by 1). The content of AC1 is added to the content addressed by AR1 + 1 (2021h) and the result is stored in AC1.

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

if (TC1) execute(AD_unit)

mar(*AR1+)

AC1 = AC1 + *AR1

Description

TC1 is not equal to 1, the next instruction is not executed (AR1 is not incremented). The content of AC1 is added to the content addressed by AR1 (2020h) and the result is stored in AC1.

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

4.28.2 Execute Conditionally (Execute Phase): if (cond) execute(D_unit)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	if (cond) execute(D_unit)	No	2	1	X

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.

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 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 control the execution of the following program control instructions:

goto	(cond) goto	intr	return_int	trap
call	(cond) call	idle	(cond) execute(AD_unit)	
return	(cond) return	reset	(cond) execute(D_unit)	
blockrepeat	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.

Examples

Syntax	Description
if (TC1) execute(D_unit) mar(*AR1+) AC1 = AC1 + *AR1	TC1 is equal to 1, the next instruction is executed (AR1 is incremented by 1). The content of AC1 is added to the content addressed by AR1 + 1 (2021h) and the result is stored in AC1.

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

if (TC1) execute(D_unit)

mar(*AR1+)

AC1 = AC1 + *AR1

Description

TC1 is not equal to 1, the next instruction would not be executed; however, 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 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

4.29 Extended Auxiliary Register Move

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	xdst = xsrc	No	2	1	X

Operands xdst, xsrc

Description

This instruction moves the content of the source register (xsrc) to the destination register (xdst):

- When the destination register (xdst) is an accumulator (ACx) and the source register (xsrc) is a 23-bit register (XARx, XSP, XSSP, XDP, or XCDP):
  - The 23-bit move operation is performed in the D-unit ALU.
  - The upper bits of ACx are filled with 0.
- When the source register (xsrc) is an accumulator (ACx) and the destination register (xdst) is a 23-bit register (XARx, XSP, XSSP, XDP, or XCDP):
  - The 23-bit move operation is performed in the A-unit ALU.
  - The lower 23 bits of ACx are loaded into xdst.
- When both the source register (xsrc) and the destination register (xdst) are accumulators, the temporary register move instruction (dst = src) is assembled.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
XAR1 = AC0	The lower 23 bits of AC0 are loaded into XAR1.

4.30 Finite Impulse Response Filter,Symmetrical/Antisymmetrical

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

Brief Description

These instructions perform two parallel operations in one cycle.

- The firs() operation is executed:

```
ACy = ACy + (ACx * Cmem),
ACx = (Xmem << #16) + (Ymem << #16)
```

- The firsn() operation is executed:

```
ACy = ACy + (ACx * Cmem),
ACx = (Xmem << #16) – (Ymem << #16)
```

Status Bits

Affected by FRCT, SMUL, C54CM, M40, SATD, SXMD

Affects ACOVx, ACOVy, CARRY

4.30.1 Symmetrical FIR Filter: firs(Xmem, Ymem, Cmem, ACx, ACy)

Syntax Characteristics

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

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 identical 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, no overflow detection, report, and saturation is done after the shifting operation.

Status Bits

Affected by FRCT, SMUL, C54CM, M40, SATD, SXMD

Affects ACOVx, ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
firs(*AR0, *AR1, *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	*AR0	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

4.30.2 Antisymmetrical FIR Filter: firsn(Xmem, Ymem, Cmem, ACx, ACy)

Syntax Characteristics

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

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 operation is identical 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, no overflow detection, report, and saturation is done after the shifting operation.

Status Bits

Affected by FRCT, SMUL, C54CM, M40, SATD, SXMD

Affects ACOVx, ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
firsn(*AR0, *AR1, *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.

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

4.31 Idle

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	idle	No	4	?	D

Operands

none

Description

This instruction forces the program being executed to wait until an interrupt or a reset occurs. The power-down mode that the processor operates in depends on a configuration register accessible through the peripheral access mechanism.

Status Bits

Affected by INTM

Affects none

Repeat

This instruction cannot be repeated.

4.32 Implied Paralleled Instructions

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(Tx * Xmem), Ymem = HI(ACx << T2) [, T3 = Xmem]	No	4	1	X
[2]	ACy = rnd(ACy + (Tx * Xmem)), Ymem = HI(ACx << T2) [, T3 = Xmem]	No	4	1	X
[3]	ACy = rnd(ACy – (Tx * Xmem)), Ymem = HI(ACx << T2) [, T3 = Xmem]	No	4	1	X
[4]	ACy = ACx + (Xmem << #16), Ymem = HI(ACy << T2)	No	4	1	X
[5]	ACy = (Xmem << #16) – ACx, Ymem = HI(ACy << T2)	No	4	1	X
[6]	ACy = Xmem << #16, Ymem = HI(ACx << T2)	No	4	1	X
[7]	ACx = rnd(ACx + (Tx * Xmem)), ACy = Ymem << #16 [, T3 = Xmem]	No	4	1	X
[8]	ACx = rnd(ACx – (Tx * Xmem)), ACy = Ymem << #16 [, T3 = Xmem]	No	4	1	X

Brief Description

These instructions perform the following parallel operations:

- ❑ multiply and store
- ❑ multiply and accumulate (MAC), and store
- ❑ multiply and subtract (MAS), and store
- ❑ addition and store
- ❑ subtraction and store
- ❑ load and store
- ❑ multiply and accumulate (MAC), and load
- ❑ multiply and subtract (MAS), and load

Status Bits

Affected by FRCT, SMUL, C54CM, M40, RDM, SATD, SXMD

Affects ACOVx, ACOVy, CARRY

4.32.1 Parallel Multiply and Store: ACy = rnd(Tx * Xmem), Ymem = HI(ACx << T2) [, T3 = Xmem]

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(Tx * Xmem), Ymem = HI(ACx << T2) [, T3 = Xmem]	No	4	1	X

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 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, ACOVy 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.

Status Bits

Affected by FRCT, SMUL, C54CM, M40, RDM, SATD, SXMD

Affects ACOVy

Repeat

This instruction can be repeated.

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.

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
T0	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

4.32.2 Parallel MAC and Store: ACy = rnd(ACy + (Tx * Xmem)), Ymem = HI(ACx << T2) [, T3 = Xmem]

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACy = rnd(ACy + (Tx * Xmem)), Ymem = HI(ACx << T2) [, T3 = Xmem]	No	4	1	X

Operands ACx, ACy, Tx, Xmem, Ymem

Description

This instruction performs two operations in parallel: multiply and accumulate (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 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 is set.
- When an addition overflow is detected, ACOVy 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.

Status Bits

Affected by FRCT, SMUL, C54CM, M40, RDM, SATD, SXMD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = AC0 + (T0 * *AR3), *AR4 = HI(AC1 << T2)	Both instructions are performed in parallel. The content addressed by AR3 multiplied by the content of T0 is added to the content of AC0 and 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.

4.32.3 Parallel MAS and Store: ACy = rnd(ACy – (Tx * Xmem)), Ymem = HI(ACx << T2) [, T3 = Xmem]

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ACy = rnd(ACy – (Tx * Xmem)), Ymem = HI(ACx << T2) [, T3 = Xmem]	No	4	1	X

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 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 is set.
- ❑ When an overflow is detected, ACOVy 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.

Status Bits

Affected by FRCT, SMUL, C54CM, M40, RDM, SATD, SXMD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = AC0 – (T0 * *AR3), *AR4 = HI(AC1 << T2)	Both instructions are performed in parallel. The content addressed by AR3 multiplied by the content of T0 is subtracted from the content of AC0 and 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.

4.32.4 Parallel Addition and Store: ACy = ACx + (Xmem << #16), Ymem = HI(ACy << T2)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	ACy = ACx + (Xmem << #16), Ymem = HI(ACy << T2)	No	4	1	X

Operands ACx, ACy, T2, Xmem, Ymem

Description

This instruction performs two operations in parallel: addition and store.

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

The second operation shifts the accumulator ACy by the content of T2 and stores ACy(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, ACy(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.

Status Bits

Affected by C54CM, M40, SATD, SXMD

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

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.

4.32.5 Parallel Subtraction and Store: ACy = (Xmem << #16) – ACx, Ymem = HI(ACy << T2)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	ACy = (Xmem << #16) – ACx, Ymem = HI(ACy << T2)	No	4	1	X

Operands ACx, ACy, T2, Xmem, Ymem

Description

This instruction performs two operations in parallel: subtraction and store.

The first operation subtracts an accumulator content from 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 identical 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 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.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

The second operation shifts the accumulator ACy by the content of T2 and stores ACy(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, ACy(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.

Status Bits

Affected by C54CM, M40, SATD, SXMD

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = (*AR3 << #16) – AC1, *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.

4.32.6 Parallel Load and Store: ACy = Xmem << #16, Ymem = HI(ACx << T2)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	ACy = Xmem << #16, Ymem = HI(ACx << T2)	No	4	1	X

Operands 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 identical 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)

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.

Status Bits

Affected by C54CM, M40, SATD, SXMD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

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.

4.32.7 Parallel MAC and Load: ACx = rnd(ACx + (Tx * Xmem)), ACy = Ymem << #16 [, T3 = Xmem]

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	ACx = rnd(ACx + (Tx * Xmem)), ACy = Ymem << #16 [, T3 = Xmem]	No	4	1	X

Operands

ACx, ACy, Tx, Xmem, Ymem

Description

This instruction performs two operations in parallel: multiply and accumulate (MAC), and load.

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 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 is set.
- When an addition overflow is detected, ACOVx 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 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 identical 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.

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD, SXMD

Affects ACOVx, ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = AC0 + (T0 * *AR3), AC1 = *AR4 << #16	Both instructions are performed in parallel. The content addressed by AR3 multiplied by the content of T0 is added to the content of AC0 and the result is stored in AC0. The content addressed by AR4 shifted left by 16 bits is stored in AC1.

4.32.8 Parallel MAS and Load: ACx = rnd(ACx – (Tx * Xmem)), ACy = Ymem << #16 [, T3 = Xmem]

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	ACx = rnd(ACx – (Tx * Xmem)), ACy = Ymem << #16 [, T3 = Xmem]	No	4	1	X

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 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, ACOVx 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 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 identical 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.

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD, SXMD

Affects ACOVx, ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = AC0 – (T0 * *AR3), AC1 = *AR4 << #16	Both instructions are performed in parallel. The content addressed by AR3 multiplied by the content of T0 is subtracted from the content of AC0 and the result is stored in AC0. The content addressed by AR4 shifted left by 16 bits is stored in AC1.

4.33 Least Mean Square (LMS)

Syntax Characteristics

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

Operands ACx, ACy, Xmem, Ymem

Description

This instruction performs two paralleled operations in one cycle: multiply and accumulate (MAC), and addition. The instruction is executed:

```
ACy = ACy + (Xmem * Ymem),
ACx = rnd (ACx + (Xmem << #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 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 identical 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.
- ❏ Rounding is performed according to RDM.

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 FRCT, SMUL, C54CM, M40, RDM, SATD, SXMD

Affects ACOVx, ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
lms(*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	*AR0	1000
*AR1	2000	*AR1	2000
ACOV0	0	ACOV0	0
ACOV1	0	ACOV1	0
CARRY	0	CARRY	0
FRCT	0	FRCT	0

4.34 Linear/Circular Addressing Qualifiers

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

Brief Description

These instructions are instruction qualifiers that can be paralleled with any instruction making an indirect Smem, Xmem, Ymem, Lmem, Baddr, or Cmem addressing. These instructions cannot be executed in parallel with other types of instructions and they cannot be executed alone.

Status Bits

Affected by none

Affects none

4.34.1 Linear Addressing Qualifier: linear()

Syntax Characteristics

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

Operands

none

Description

This instruction is an instruction qualifier that can be paralleled with any instruction making an indirect Smem, Xmem, Ymem, Lmem, Baddr, or Cmem addressing. This instruction cannot be executed in parallel with other types of instructions and it cannot be executed alone.

When this instruction is used in parallel, all modifications of ARx and CDP pointer registers used in the indirect addressing mode are done linearly (as if ST2_55 register bits 0 to 8 were cleared to 0).

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

4.34.2 Circular Addressing Qualifier: circular()

Syntax Characteristics

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

Operands none

Description

This instruction is an instruction qualifier that can be paralleled with any instruction making an indirect Smem, Xmem, Ymem, Lmem, Baddr, or Cmem addressing. This instruction cannot be executed in parallel with other types of instructions and it cannot be executed alone.

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).

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

4.35 Load Effective Address to Extended Auxiliary Register

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	XAdst = mar(Smem)	No	3	1	AD
[2]	XAdst = k23	No	6	1	AD

Operands k23, Smem, XAdst

Description

If a memory (Smem) location is specified, 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.

If a 23-bit unsigned constant (k23) is specified, this instruction loads the constant into the 23-bit destination register (XARx, XSP, XSSP, XDP, or XCDP).

The premodification or postmodification of the auxiliary register (ARx), the use of *port(#K), or 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.

Examples

Syntax	Description
XAR0 = mar(*AR1+)	The content of AR1 is incremented by 1 and loaded into XAR0.
XAR0 = #7FFFFFh	Move the 23-bit value (7FFFFFh) into XAR0.

4.36 Load Extended Auxiliary Register from Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	XAdst = dbl(Lmem)	No	3	1	X

Operands Lmem , XAdst

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.

Example

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 0FD3
AR3	0200	AR3	0200
200	3492	200	3492
201	0FD3	201	0FD3

4.37 Logical Shift

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = dst <<< #1	Yes	2	1	X
[2]	dst = dst >>> #1	Yes	2	1	X
[3]	ACy = ACx <<< Tx	Yes	2	1	X
[4]	ACy = ACx <<< #SHIFTW	Yes	3	1	X

Brief Description

These instructions perform an unsigned shift by an immediate value, 1 or SHIFTW, or the content of a temporary register (Tx):

- 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

Affects CARRY

4.37.1 Logical Shift Left: dst = dst <<< #1

Syntax Characteristics

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

Operands

dst

Description

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.

Examples

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 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       M40                     0
```

4.37.2 Logical Shift Right: dst = dst >>> #1

Syntax Characteristics

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

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.

Examples

Syntax	Description
AC0 = AC0 >>> #1	The content of AC0 is logically shifted right by 1 bit and the result is stored in AC0.

4.37.3 Logical Shift: ACy = ACx <<< Tx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ACy = ACx <<< Tx	Yes	2	1	X

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.
- ❑ 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.

Examples

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       T0                   FFFA
M40                     0       M40                     0
```

4.37.4 Logical Shift: ACy = ACx <<< #SHIFTW

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	ACy = ACx <<< #SHIFTW	Yes	3	1	X

Operands ACx, ACy, SHIFTW

Description

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 performed on 40 bits in the D-unit shifter.
- The shift operation is performed according to M40.
- The CARRY status bit contains the shifted-out bit. When the shift count is zero, SHIFTW = 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.

Status Bits

Affected by M40

Affects CARRY

Repeat

This instruction can be repeated.

Examples

Syntax	Description
AC0 = AC1 <<< #31	The content of AC1 is logically shifted left by 31 bits and the result is stored in AC0.

4.38 Maximum Comparison (max)

Syntax Characteristics

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

Operands dst, src

Description

These instructions perform a maximum comparison in the D-unit ALU or in the A-unit ALU. Two accumulator, auxiliary registers, and s contents are compared. When an accumulator ACx is compared with an auxiliary or 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 is the source operand (src) of the instruction, the 16 LSBs of the auxiliary or 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 :

- 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 , 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 is the source operand (src) of the instruction, the 16 LSBs of the auxiliary or 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.

Examples

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

Syntax	Description
AC1 = max(AR1, A C1)	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

4.39 Minimum Comparison (min)

Syntax Characteristics

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

Operands dst, src

Description

These instructions perform a minimum comparison in the D-unit ALU or in the A-unit ALU. Two accumulator, auxiliary registers, and s contents are compared. When an accumulator ACx is compared with an auxiliary or 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 is the source operand (src) of the instruction, the 16 LSBs of the auxiliary or 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 :
  - 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 , 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 is the source operand (src) of the instruction, the 16 LSBs of the auxiliary or 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.

Example

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

4.40 Memory-Mapped Register Access Qualifier (mmap)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	mmap()	No	1	1	D

Operands none

Description

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:

- ❑ alone
- ❑ 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 Accumulator, Auxiliary, or Temporary Register Load instructions [5], [6], [10], and [11]; Accumulator, Auxiliary, or Temporary Register Store 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.

Example

Syntax	Description
T2 = @(AC0_L) || mmap()	AC0_L is a keyword representing AC0(15–0). The content of AC0(15–0) is copied into T2.

4.41 Memory Bit Test/Set/Clear/Complement

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	TCx = bit(Smem, src)	No	3	1	X
[2]	cbit(Smem, src)	No	3	1	X
[3]	bit(Smem, src) = #0	No	3	1	X
[4]	bit(Smem, src) = #1	No	3	1	X
[5]	TCx = bit(Smem, k4), bit(Smem, k4) = #1	No	3	1	X
[6]	TCx = bit(Smem, k4), bit(Smem, k4) = #0	No	3	1	X
[7]	TCx = bit(Smem, k4), cbit(Smem, k4)	No	3	1	X
[8]	TCx = bit(Smem, k4)	No	3	1	X

Brief Description

These instructions perform a bit manipulation in the A-unit ALU. These instructions manipulate a single bit of a memory (Smem) location. The bit manipulated is defined by either the content of the source (src) operand or a 4-bit immediate value, k4. The following operations can be performed on the selected bit:

- ❏ copy the bit into TCx
- ❏ complement the bit in Smem
- ❏ clear the bit in Smem
- ❏ set the bit in Smem
- ❏ copy the bit into TCx and set the bit in Smem
- ❏ copy the bit into TCx and clear the bit in Smem
- ❏ copy the bit into TCx and complement the bit in Smem

Status Bits

Affected by none

Affects TCx

4.41.1 Memory Bit Test: TCx = bit(Smem, src)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	TCx = bit(Smem, src)	No	3	1	X

Operands Smem, src, TCx

Description

This instruction performs a bit manipulation in the A-unit ALU. The instruction tests a single bit, as defined by the content of the source (src) operand, of a memory (Smem) location. 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

Affects TCx

Repeat

This instruction can be repeated.

Examples

Syntax

TC1 = bit(*AR0, AC0)

Description

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 0008   AC0        00 0000 0008
*AR0             00C0     *AR0             00C0
TC1                 0     TC1                 0
```

Syntax

TC2 = bit(*AR0, AC0)

Description

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 TC2.

```
Before                    After
AC0        00 0000 0007   AC0        00 0000 0007
*AR0             00C0     *AR0             00C0
TC2                 1     TC2                 1
```

4.41.2 Memory Bit Complement: cbit(Smem, src)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	cbit(Smem, src)	No	3	1	X

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.

Example

Syntax	Description
cbit(*AR3, AC0)	The bit at the position defined by AC0(3–0) in the content addressed by AR3 is complemented.

4.41.3 Memory Bit Clear: bit(Smem, src) = #0

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	bit(Smem, src) = #0	No	3	1	X

Operands 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 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.

Example

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.

4.41.4 Memory Bit Set: bit(Smem, src) = #1

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	bit(Smem, src) = #1	No	3	1	X

Operands

Smem, src

Description

This instruction performs a bit manipulation in the A-unit ALU. The instruction sets to 1 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.

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.

4.41.5 Memory Bit Test/Set: TCx = bit(Smem, k4), bit(Smem, k4) = #1

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	TCx = bit(Smem, k4), bit(Smem, k4) = #1	No	3	1	X

Operands k4, Smem, TCx

Description

This instruction performs a bit manipulation in the A-unit ALU. The instruction tests a single bit, as defined by a 4-bit immediate value, k4, of a memory (Smem) location. The tested bit is copied into status bit TCx and is set to 1 in Smem.

Status Bits

Affected by none

Affects TCx

Repeat

This instruction can be repeated.

Examples

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.
TC2 = 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 TC2. The selected bit (12) in the content addressed by AR3 is set to 1.

4.41.6 Memory Bit Test/Clear: TCx = bit(Smem, k4), bit(Smem, k4) = #0

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	TCx = bit(Smem, k4), bit(Smem, k4) = #0	No	3	1	X

Operands

k4, Smem, TCx

Description

This instruction performs a bit manipulation in the A-unit ALU. The instruction tests a single bit, as defined by a 4-bit immediate value, k4, of a memory (Smem) location. The tested bit is copied into status bit TCx and is cleared to 0 in Smem.

Status Bits

Affected by none

Affects TCx

Repeat

This instruction can be repeated.

Examples

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.
TC2 = 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 TC2. The selected bit (12) in the content addressed by AR3 is cleared to 0.

4.41.7 Memory Bit Test/Complement: TCx = bit(Smem, k4), cbit(Smem, k4)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	TCx = bit(Smem, k4), cbit(Smem, k4)	No	3	1	X

Operands k4, Smem, TCx

Description

This instruction performs a bit manipulation in the A-unit ALU. The instruction tests a single bit, as defined by a 4-bit immediate value, k4, of a memory (Smem) location and the tested bit is copied into status bit TCx and is complemented in Smem.

Status Bits

Affected by none

Affects TCx

Repeat

This instruction can be repeated.

Examples

Syntax	Description
TC1 = bit(*AR0, #12), cbit(*AR0, #12)	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	
*AR0	0040	*AR0	1040
TC1	0	TC1	0

Syntax	Description
TC2 = bit(*AR3, #12), cbit(*AR3, #12)	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 TC2. The selected bit (12) in the content addressed by AR3 is complemented.

Before		After	
*AR3	0040	*AR3	1040
TC2	0	TC2	0

4.41.8 Memory Bit Test: TCx = bit(Smem, k4)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	TCx = bit(Smem, k4)	No	3	1	X

Operands k4, Smem, TCx

Description

This instruction performs a bit manipulation in the A-unit ALU. The instruction tests a single bit, as defined by a 4-bit immediate value, k4, of a memory (Smem) location. The tested bit is copied into status bit TCx.

Status Bits

Affected by none

Affects TCx

Repeat

This instruction can be repeated.

Examples

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.
TC2 = 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 TC2.

4.42 Memory Comparison

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	TCx = (Smem == K16)	No	4	1	X

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.

Status Bits

Affected by none

Affects TCx

Repeat

This instruction can be repeated.

Examples

Syntax
TC1 = (*AR1+ == #400h)

Description
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
TC2 = (*AR1 == #400h)

Description
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
```

4.43 Memory Delay

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	delay(Smem)	No	2	1	X

Operands

Smem

Description

This instruction copies the content of the memory (Smem) location into the next higher address. 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 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 soft dual memory addressing mode mechanism can not be applied to this instruction.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

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

4.44 Memory-to-Memory Move/Memory Initialization

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	Smem = Cmem	No	3	1	X
[2]	Cmem = Smem	No	3	1	X
[3]	Smem = K8	No	3	1	X
[4]	Smem = K16	No	4	1	X
[5]	Lmem = dbl(Cmem)	No	3	1	X
[6]	dbl(Cmem) = Lmem	No	3	1	X
[7]	dbl(Ymem) = dbl(Xmem)	No	3	1	X
[8]	Ymem = Xmem	No	3	1	X

Brief Description

These instructions initialize data memory locations. They use a dedicated datapath to perform the operation.

Status Bits

Affected by none

Affects none

4.44.1 Memory-to-Memory Move: Smem = Cmem

Syntax Characteristics

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

Operands

Cmem, Smem

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 (contrary to instructions like Dual Multiply and Accumulate using the Cmem operand). 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.

Example

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

4.44.2 Memory-to-Memory Move: Cmem = Smem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	Cmem = Smem	No	3	1	X

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 (contrary to instructions like Dual Multiply and Accumulate using the Cmem operand). 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.

Example

Syntax	Description
coef(*CDP) = *AR3	The content addressed by AR3 is copied in the location addressed by the coefficient data pointer register (CDP).

4.44.3 Memory Initialization: Smem = Kx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	Smem = K8	No	3	1	X
[4]	Smem = K16	No	4	1	X

Operands

Kx, Smem

Description

This instruction stores an 8-bit signed constant, K8, or a 16-bit signed constant, K16, to a memory (Smem) location. For instruction [3], the immediate value is always signed extended to 16 bits before being stored in memory.

Status Bits

Affected by none

Affects none

Repeat

Instruction [3] can be repeated. Instruction [4] cannot be repeated.

Example

Syntax	Description
*(#0501h) = #248	The signed 16-bit value (248) is loaded to address 501h.

```
Before                  After
0501        FC00        0501        F800
```

4.44.4 Memory-to-Memory Move: Lmem = dbl(Cmem)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	Lmem = dbl(Cmem)	No	3	1	X

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 (contrary to instructions like Dual Multiply and Accumulate using the Cmem operand). 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.

Example

Syntax	Description
AR1 = dbl(coef((CDP + T0)))	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	
T0	0005	T0	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

4.44.5 Memory-to-Memory Move: dbl(Cmem) = Lmem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	dbl(Cmem) = Lmem	No	3	1	X

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 (contrary to instructions like Dual Multiply and Accumulate using the Cmem operand). 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.

Example

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.

4.44.6 Memory-to-Memory Move: dbl(Ymem) = dbl(Xmem)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	dbl(Ymem) = dbl(Xmem)	No	3	1	X

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.

Example

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

4.44.7 Memory-to-Memory Move: Ymem = Xmem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	Ymem = Xmem	No	3	1	X

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.

Example

Syntax	Description
*AR3 = *AR5	The content addressed by AR5 is copied in the location addressed by AR3.

4.45 Modify Auxiliary Register (MAR)

No.	Syntax†	Parallel Enable Bit	Size	Cycles	Pipeline
[1a]	mar(TAy + TAx)	Yes	3	1	AD
[1b]	mar(TAy + TAx)	Yes	3	1	AD
[2a]	mar(TAy – TAx)	Yes	3	1	AD
[2b]	mar(TAy – TAx)	Yes	3	1	AD
[3a]	mar(TAy = TAx)	Yes	3	1	AD
[3b]	mar(TAy = TAx)	Yes	3	1	AD
[4a]	mar(TAx + k8)	Yes	3	1	AD
[4b]	mar(TAx + k8)	Yes	3	1	AD
[5a]	mar(TAx – k8)	Yes	3	1	AD
[5b]	mar(TAx – k8)	Yes	3	1	AD
[6a]	mar(TAx = k8)	Yes	3	1	AD
[6b]	mar(TAx = k8)	Yes	3	1	AD
[7]	mar(TAx = D16)	No	4	1	AD
[8]	mar(Smem)	No	2	1	AD

† For instructions [1]–[6], the assembler selects the correct instruction opcode depending on the instruction position in a paralleled pair.

Brief Description

These instructions perform, in the A-unit address generation units:

- ❑ an addition between two auxiliary or temporary registers, TAy and TAx, and stores the result in TAy
- ❑ a subtraction between two auxiliary or temporary registers, TAy and TAx, and stores the result in TAy
- ❑ a move from the auxiliary or temporary registers TAx to data or auxiliary register TAy
- ❑ an addition between the auxiliary or temporary registers TAx and the unsigned constant k8, and stores the result in TAx
- ❑ a subtraction between the auxiliary or temporary registers TAx and the unsigned constant k8, and stores the result in TAx
- ❑ a load in the auxiliary or temporary registers TAx of the unsigned constant k8
- ❑ a load in the auxiliary or temporary registers TAx of the absolute data address signed constant D16
- ❑ an 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.

For instructions [1], [2], [4], [5], and [8], 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.

Status Bits

Affected by	ST2_55
Affects	none

4.45.1 Modify Auxiliary Register: mar(TAy + TAx)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	mar(TAy + TAx)	Yes	3	1	AD

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 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.

Examples

Syntax	Description
mar(AR0 + AR1)	The content of AR0 is added to the content of AR1 and the result is stored in AR0.
mar(T0 + T1)	The content of T0 is added to the content of T1 and the result is stored in T0.

4.45.2 Modify Auxiliary Register: mar(TAy – TAx)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	mar(TAy – TAx)	Yes	3	1	AD

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 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.

Examples

Syntax	Description
mar(AR0 – AR1)	The content of AR1 is subtracted from the content of AR0 and the result is stored in AR0.
mar(T0 – T1)	The content of T1 is subtracted from the content of T0 and the result is stored in T0.

4.45.3 Modify Auxiliary Register: mar(TAy = TAx)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	mar(TAy = TAx)	Yes	3	1	AD

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 register or temporary register TAy. The operation is performed in the address phase of the pipeline; however, data memory is not accessed.

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 none

Affects none

Repeat

This instruction can be repeated.

Examples

Syntax	Description
mar(AR0 = AR1)	The content of AR1 is copied to AR0.
mar(T0 = T1)	The content of T1 is copied to T0.

4.45.4 Modify Auxiliary Register: mar(TAx + k8)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	mar(TAx + k8)	Yes	3	1	AD

Operands

TAx, k8

Description

This instruction performs, in the A-unit address generation units, an addition between the auxiliary or temporary registers TAx and the unsigned constant k8, 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, 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.

Examples

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.
mar(AR0 + #255)	The unsigned 8-bit value (255) is added to the content of AR0 and the result is stored in AR0.

4.45.5 Modify Auxiliary Register: mar(TAx – k8)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	mar(TAx – k8)	Yes	3	1	AD

Operands

TAx, k8

Description

This instruction performs, in the A-unit address generation units, a subtraction between the auxiliary or temporary registers TAx and the unsigned constant k8, 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, 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.

Examples

Syntax	Description
mar(AR0 – #255)	The unsigned 8-bit value (255) is subtracted from the content of AR0 and the result is stored in AR0.
mar(T0 – #255)	The unsigned 8-bit value (255) is subtracted from the content of T0 and the result is stored in T0.

4.45.6 Modify Auxiliary Register: mar(TAx = k8)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	mar(TAx = k8)	Yes	3	1	AD

Operands

TAx, k8

Description

This instruction performs, in the A-unit address generation units, a load in the auxiliary or temporary registers TAx of the unsigned constant k8. The operation is performed in the address phase of the pipeline; however, data memory is not accessed.

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 none

Affects none

Repeat

This instruction can be repeated.

Examples

Syntax	Description
mar(AR0 = #255)	The unsigned 8-bit value (255) is copied to AR0.
mar(T0 = #255)	The unsigned 8-bit value (255) is copied to T0.

4.45.7 Modify Auxiliary Register: mar(TAx = D16)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	mar(TAx = D16)	No	4	1	AD

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.

Compatibility with C54x devices (C54CM = 1)

In the translated code section, the mar() instruction must be executed with C54CM set to 1.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
mar(T1 = #FFFFh)	The address FFFFh is copied to T1.

4.45.8 Modify Auxiliary Register: mar(Smem)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	mar(Smem)	No	2	1	AD

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.

Status Bits

Affected by ST2_55

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
mar(*AR3+)	The content of AR3 is incremented by 1.

4.46 Modify Data Stack Pointer (SP)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	SP = SP + K8	Yes	2	1	AD

Operands K8

Description

This instruction performs an addition in the A-unit ALU in the address phase of the pipeline. The 8-bit 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.

The latencies versus any address generation through SP is 3 cycles.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
SP = SP + #127	The 8-bit value (127) is sign extended to 16 bits and added to the stack pointer (SP).

4.47 Multiply (MPY)

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(ACx * ACx)	Yes	2	1	X
[2]	ACy = rnd(ACy * ACx)	Yes	2	1	X
[3]	ACy = rnd(ACx * Tx)	Yes	2	1	X
[4]	ACy = rnd(ACx * K8)	Yes	3	1	X
[5]	ACy = rnd(ACx * K16)	No	4	1	X
[6]	ACx = rnd(Smem * Cmem) [,T3 = Smem]	No	3	1	X
[7]	ACx = rnd(Smem * Smem) [,T3 = Smem]	No	3	1	X
[8]	ACy = rnd(Smem * ACx) [,T3 = Smem]	No	3	1	X
[9]	ACx = rnd(Smem * K8) [,T3 = Smem]	No	4	1	X
[10]	ACx = M40(rnd(uns(Xmem) * uns(Ymem))) [,T3 = Xmem]	No	4	1	X
[11]	ACx = rnd(uns(Tx * Smem)) [,T3 = Smem]	No	3	1	X

Brief Description

This instruction performs a multiplication in the D-unit MAC. The input operands of the multiplier are:

- ACx(32–16)
- the content of Tx, sign extended to 17 bits
- the 8-bit signed constant, K8, sign extended to 17 bits
- the 16-bit signed constant, K16, sign extended to 17 bits
- the content of a memory (Smem) location, sign extended to 17 bits
- the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits
- 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

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVx, ACOVy

4.47.1 Multiply: ACy = rnd(ACx * ACx)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(ACx * ACx)	Yes	2	1	X

Operands

ACx, 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 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 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, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 * AC1	The content of AC1 is squared and the result is stored in AC0.

4.47.2 Multiply: ACy = rnd(ACy * ACx)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACy = rnd(ACy * ACx)	Yes	2	1	X

Operands

ACx, 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 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 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, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC1 = AC0 * AC1	The content of AC1 is multiplied by the content of AC0 and the result 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

4.47.3 Multiply: ACy = rnd(ACx * Tx)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ACy = rnd(ACx * Tx)	Yes	2	1	X

Operands ACx, ACy, Tx

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 Tx, 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.
- 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.

Compatibility with C54x devices (C54CM = 1)

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

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 * T0	The content of AC1 is multiplied by the content of T0 and the result is stored in AC0.

4.47.4 Multiply: ACy = rnd(ACx * K8)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	ACy = rnd(ACx * K8)	Yes	3	1	X

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 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, RDM

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 * #–2	The content of AC1 is multiplied by a signed 8-bit value (–2) and the result is stored in AC0.

4.47.5 Multiply: ACy = rnd(ACx * K16)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	ACy = rnd(ACx * K16)	No	4	1	X

Operands

ACx, ACy, K16

Description

This instruction performs a multiplication in the D-unit MAC. The input operands of the multiplier are ACx(32–16) and the 16-bit signed constant, K16, 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.
- 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.

Compatibility with C54x devices (C54CM = 1)

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

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 * #–64	The content of AC1 is multiplied by a signed 16-bit value (–64) and the result is stored in AC0.

4.47.6 Multiply: ACx = rnd(Smem * Cmem) [,T3 = Smem]

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	ACx = rnd(Smem * Cmem) [,T3 = Smem]	No	3	1	X

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 (Smem) location, sign extended to 17 bits, 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.
- 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 store the 16-bit data-memory operand Smem in 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, SMUL, M40, RDM, SATD

Affects ACOVx

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = *AR3 * *CDP	The content addressed by AR3 is multiplied by the content addressed by the coefficient data pointer register (CDP) and the result is stored in AC0.

4.47.7 Multiply: ACx = rnd(Smem * Smem) [,T3 = Smem]

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	ACx = rnd(Smem * Smem) [,T3 = Smem]	No	3	1	X

Operands

ACx, Smem

Description

This instruction performs a multiplication 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, 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 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 T3.

Compatibility with C54x devices (C54CM = 1)

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

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVx

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = *AR3 * *AR3	The content addressed by AR3 is squared and the result is stored in AC0.

4.47.8 Multiply: ACy = rnd(Smem * ACx) [,T3 = Smem]

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	ACy = rnd(Smem * ACx) [,T3 = Smem]	No	3	1	X

Operands

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 (Smem) location, 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.
- 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 store the 16-bit data-memory operand Smem in T3.

Compatibility with C54x devices (C54CM = 1)

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

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

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

4.47.9 Multiply: ACx = rnd(Smem * K8) [,T3 = Smem]

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[9]	ACx = rnd(Smem * K8) [,T3 = Smem]	No	4	1	X

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 (Smem) location, 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 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 T3.

Compatibility with C54x devices (C54CM = 1)

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

Status Bits

Affected by FRCT, RDM

Affects none

Repeat

This instruction cannot be repeated.

Example

Syntax	Description
AC0 = *AR3 * #–2	The content addressed by AR3 is multiplied a signed 8-bit value (–2) and the result is stored in AC0.

4.47.10 Multiply: ACx = M40(rnd(uns(Xmem) * uns(Ymem))) [,T3 = Xmem]

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[10]	ACx = M40(rnd(uns(Xmem) * uns(Ymem))) [,T3 = Xmem]	No	4	1	X

Operands ACx, Xmem, Ymem

Description

This instruction performs a multiplication 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.

- 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 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.

This instruction provides the option to store the 16-bit data-memory operand Xmem in T3.

Compatibility with C54x devices (C54CM = 1)

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

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVx

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = uns(*AR3) * uns(*AR4)	The unsigned content addressed by AR3 is multiplied by the unsigned content addressed by AR4 and the result is stored in AC0.

4.47.11 Multiply: ACx = rnd(uns(Tx * Smem)) [,T3 = Smem]

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[11]	ACx = rnd(uns(Tx * Smem)) [,T3 = Smem]	No	3	1	X

Operands ACx, Tx, Smem

Description

This instruction performs a multiplication in the D-unit MAC. The input operands of the multiplier are the content of Tx and the content of a memory (Smem) location.

- The input operands are 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 left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits.
- The 32-bit result of the multiplication is zero extended to 40 bits, if the optional uns keyword is applied to the instruction.
- 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 store the 16-bit data-memory operand Smem in T3.

Compatibility with C54x devices (C54CM = 1)

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

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVx

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = uns(T0 * *AR3)	The unsigned content addressed by AR3 is multiplied by the unsigned content of T0 and the result is stored in AC0.

4.48 Multiply and Accumulate (MAC)

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(ACy + (ACx * ACx))	Yes	2	1	X
[2]	ACy = rnd(ACy + (ACx * Tx))	Yes	2	1	X
[3]	ACy = rnd((ACy * Tx) + ACx)	Yes	2	1	X
[4]	ACy = rnd(ACx + (Tx * K8))	Yes	3	1	X
[5]	ACy = rnd(ACx + (Tx * K16))	No	4	1	X
[6]	ACx = rnd(ACx + (Smem * Cmem)) [,T3 = Smem]	No	3	1	X
[7]	ACx = rnd(ACx + (Smem * Cmem)) [,T3 = Smem], delay(Smem)	No	3	1	X
[8]	ACy = rnd(ACx + (Smem * Smem)) [,T3 = Smem]	No	3	1	X
[9]	ACy = rnd(ACy + (Smem * ACx)) [,T3 = Smem]	No	3	1	X
[10]	ACy = rnd(ACx + (Tx * Smem)) [,T3 = Smem]	No	3	1	X
[11]	ACy = rnd(ACx + (Smem * K8)) [,T3 = Smem]	No	4	1	X
[12]	ACy = M40(rnd(ACx + (uns(Xmem) * uns(Ymem)))) [,T3 = Xmem]	No	4	1	X
[13]	ACy = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(Ymem)))) [,T3 = Xmem]	No	4	1	X

Brief Description

This instruction performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are:

- ACx(32–16)
- the content of Tx, sign extended to 17 bits
- the 8-bit signed constant, K8, sign extended to 17 bits
- the 16-bit signed constant, K16, sign extended to 17 bits
- the content of a memory (Smem) location, sign extended to 17 bits
- the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits
- 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

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVx, ACOVy

4.48.1 Multiply and Accumulate (MAC): ACy = rnd(ACy + (ACx * ACx))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(ACy + (ACx * ACx))	Yes	2	1	X

Operands

ACx, ACy

Description

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 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, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

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.

4.48.2 Multiply and Accumulate (MAC): ACy = rnd(ACy + (ACx * Tx))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACy = rnd(ACy + (ACx * Tx))	Yes	2	1	X

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 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 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, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

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

4.48.3 Multiply and Accumulate (MAC): ACy = rnd((ACy * Tx) + ACx)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ACy = rnd((ACy * Tx) + ACx)	Yes	2	1	X

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 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 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, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC1 = rnd((AC1 * T1) + AC0)	The content of AC1 multiplied by the content of T1 is added to the content of AC0. The result is rounded and stored in AC1.

4.48.4 Multiply and Accumulate (MAC): ACy = rnd(ACx + (Tx * K8))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	ACy = rnd(ACx + (Tx * K8))	Yes	3	1	X

Operands

ACx, ACy, Tx, K8

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 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 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.

Example

Syntax	Description
AC0 = AC1 + (T0 * K8)	The content of T0 multiplied by a signed 8-bit value is added to the content of AC1 and the result is stored in AC0.

4.48.5 Multiply and Accumulate (MAC): ACy = rnd(ACx + (Tx * K16))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	ACy = rnd(ACx + (Tx * K16))	No	4	1	X

Operands ACx, ACy, Tx, K16

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 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 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, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 + (T0 * K16)	The content of T0 multiplied by a signed 16-bit value is added to the content of AC1 and the result is stored in AC0.

4.48.6 Multiply and Accumulate (MAC): ACx = rnd(ACx + (Smem * Cmem)) [,T3 = Smem]

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	ACx = rnd(ACx + (Smem * Cmem)) [,T3 = Smem]	No	3	1	X

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 (Smem) location, sign extended to 17 bits, 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 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 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 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, SMUL, M40, RDM, SATD

Affects ACOVx

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC2 = rnd(AC2 + (*AR1 * *CDP))	The content addressed by AR1 multiplied by 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 0000	AC2	00 EC00 0000
AR1	0302	AR2	0302
CDP	0202	CDP	0202
302	FE00	302	FE00
202	0040	202	0040
ACOV2	0	ACOV2	1

4.48.7 Parallel Multiply and Accumulate (MAC) and Delay: ACx = rnd(ACx + (Smem * Cmem)) [,T3 = Smem], delay(Smem)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	ACx = rnd(ACx + (Smem * Cmem)) [,T3 = Smem], delay(Smem)	No	3	1	X

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 (Smem) location, sign extended to 17 bits, 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 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 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 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 set to 0, compatibility is ensured.

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVx

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = AC0 + (*AR3 * *CDP), delay(*AR3)	The content addressed by AR3 multiplied by the content addressed by the coefficient data pointer register (CDP) is added to the content of AC0 and the result is stored in AC0. The content addressed by AR3 is copied into the next higher address.

4.48.8 Multiply and Accumulate (MAC): ACy = rnd(ACx + (Smem * Smem)) [,T3 = Smem]

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	ACy = rnd(ACx + (Smem * Smem)) [,T3 = Smem]	No	3	1	X

Operands

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, 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.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit 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 T3.

Compatibility with C54x devices (C54CM = 1)

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

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

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.

4.48.9 Multiply and Accumulate (MAC): ACy = rnd(ACy + (Smem * ACx)) [,T3 = Smem]

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[9]	ACy = rnd(ACy + (Smem * ACx)) [,T3 = Smem]	No	3	1	X

Operands

ACx, ACy, Smem

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 a memory (Smem) location, 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.
- 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 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 T3.

Compatibility with C54x devices (C54CM = 1)

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

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

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

4.48.10 Multiply and Accumulate (MAC): ACy = rnd(ACx + (Tx * Smem)) [,T3 = Smem]

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[10]	ACy = rnd(ACx + (Tx * Smem)) [,T3 = Smem]	No	3	1	X

Operands

ACx, ACy, Tx, 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 Tx, sign extended to 17 bits, and the content of a memory (Smem) location, 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 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 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 T3.

Compatibility with C54x devices (C54CM = 1)

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

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 + (T0 * *AR3)	The content addressed by AR3 multiplied by the content of T0 is added to the content of AC1 and the result is stored in AC0.

4.48.11 Multiply and Accumulate (MAC): ACy = rnd(ACx + (Smem * K8)) [,T3 = Smem]

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[11]	ACy = rnd(ACx + (Smem * K8)) [,T3 = Smem]	No	4	1	X

Operands ACx, ACy, K8, 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, and the 8-bit signed constant, K8, sign extended to 17 bits.

- If FRCT = 1, the output of the multiplier is shifted 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 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 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 cannot be repeated.

Example

Syntax	Description
AC0 = AC1 + (*AR3 * K8)	The content addressed by AR3 multiplied by a signed 8-bit value is added to the content of AC1 and the result is stored in AC0.

4.48.12 Multiply and Accumulate (MAC): ACy = M40(rnd(ACx + (uns(Xmem) * uns(Ymem)))) [,T3 = Xmem]

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[12]	ACy = M40(rnd(ACx + (uns(Xmem) * uns(Ymem)))) [,T3 = Xmem]	No	4	1	X

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, sign extended to 17 bits, and the content of data memory operand Ymem, sign extended to 17 bits.

- 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 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 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 T3.

Compatibility with C54x devices (C54CM = 1)

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

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax

AC3 = rnd(AC3 + (uns(*AR2+) * uns(*AR3+)))

Description

The unsigned content addressed by AR2 multiplied by 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	M40	0
SATD	0	SATD	0
FRCT	0	FRCT	0

4.48.13 Multiply and Accumulate (MAC): ACy = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(Ymem)))) [,T3 = Xmem]

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[13]	ACy = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(Ymem)))) [,T3 = Xmem]	No	4	1	X

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, sign extended to 17 bits, and the content of data memory operand Ymem, sign extended to 17 bits.

- ❏ 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 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.
- ❏ Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit 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 T3.

Compatibility with C54x devices (C54CM = 1)

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

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = (AC1 >> #16) + (uns(*AR3) * uns(*AR4))	The unsigned content addressed by AR3 multiplied by the unsigned content addressed by AR4 is added to the content of AC1 shifted right by 16 bits and the result is stored in AC0.

4.49 Multiply and Subtract (MAS)

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(ACy – (ACx * ACx))	Yes	2	1	X
[2]	ACy = rnd(ACy – (ACx * Tx))	Yes	2	1	X
[3]	ACx = rnd(ACx – (Smem * Cmem)) [,T3 = Smem]	No	3	1	X
[4]	ACy = rnd(ACx – (Smem * Smem)) [,T3 = Smem]	No	3	1	X
[5]	ACy = rnd(ACy – (Smem * ACx)) [,T3 = Smem]	No	3	1	X
[6]	ACy = rnd(ACx – (Tx * Smem)) [,T3 = Smem]	No	3	1	X
[7]	ACy = M40(rnd(ACx – (uns(Xmem) * uns(Ymem)))) [,T3 = Xmem]	No	4	1	X

Brief Description

This instruction performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are:

- ACx(32–16)
- the content of Tx, sign extended to 17 bits
- the content of a memory (Smem) location, sign extended to 17 bits
- the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits
- 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

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVx, ACOVy

4.49.1 Multiply and Subtract (MAS): ACy = rnd(ACy – (ACx * ACx))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(ACy – (ACx * ACx))	Yes	2	1	X

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 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, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

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.

4.49.2 Multiply and Subtract (MAS): ACy = rnd(ACy – (ACx * Tx))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACy = rnd(ACy – (ACx * Tx))	Yes	2	1	X

Operands

ACx, ACy, Tx

Description

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 Tx, 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 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 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, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax

AC1 = rnd(AC1 – (AC0 * T1))

Description

The content of AC0 multiplied by 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
M40	0	M40	0
ACOV1	0	ACOV1	0
FRCT	0	FRCT	0

4.49.3 Multiply and Subtract (MAS): ACx = rnd(ACx – (Smem * Cmem)) [,T3 = Smem]

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ACx = rnd(ACx – (Smem * Cmem)) [,T3 = Smem]	No	3	1	X

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 (Smem) location, sign extended to 17 bits, 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 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 store the 16-bit data-memory operand Smem in 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, SMUL, M40, RDM, SATD

Affects ACOVx

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC2 = rnd(AC2 – (*AR1 * *CDP))	The content addressed by AR1 multiplied by 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

4.49.4 Multiply and Subtract (MAS): ACy = rnd(ACx – (Smem * Smem)) [,T3 = Smem]

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	ACy = rnd(ACx – (Smem * Smem)) [,T3 = Smem]	No	3	1	X

Operands ACx, ACy, 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 (Smem) location, 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 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 store the 16-bit data-memory operand Smem in T3.

Compatibility with C54x devices (C54CM = 1)

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

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 – (*AR3 * *AR3)	The content addressed by AR3 squared is subtracted from the content of AC1 and the result is stored in AC0.

4.49.5 Multiply and Subtract (MAS): ACy = rnd(ACy – (Smem * ACx)) [,T3 = Smem]

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	ACy = rnd(ACy – (Smem * ACx)) [,T3 = Smem]	No	3	1	X

Operands

ACx, ACy, Smem

Description

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 (Smem) location, 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 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 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 T3.

Compatibility with C54x devices (C54CM = 1)

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

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = AC0 – (*AR3 * AC1)	The content addressed by AR3 multiplied by the content of AC1 is subtracted from the content of AC0 and the result is stored in AC0.

4.49.6 Multiply and Subtract (MAS): ACy = rnd(ACx – (Tx * Smem)) [,T3 = Smem]

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	ACy = rnd(ACx – (Tx * Smem)) [,T3 = Smem]	No	3	1	X

Operands

ACx, ACy, Tx, 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 Tx, sign extended to 17 bits, and the content of a memory (Smem) location, 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 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 store the 16-bit data-memory operand Smem in T3.

Compatibility with C54x devices (C54CM = 1)

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

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 – (T0 * *AR3)	The content addressed by AR3 multiplied by the content of T0 is subtracted from the content of AC1 and the result is stored in AC0.

4.49.7 Multiply and Subtract (MAS): ACy = M40(rnd(ACx – (uns(Xmem) * uns(Ymem)))) [,T3 = Xmem]

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	ACy = M40(rnd(ACx – (uns(Xmem) * uns(Ymem)))) [,T3 = Xmem]	No	4	1	X

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, sign extended to 17 bits, and the content of data memory operand Ymem, sign extended to 17 bits.

- ❑ 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 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.

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 T3.

Compatibility with C54x devices (C54CM = 1)

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

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC3 = AC3 – (uns(*AR2+) * uns(*AR3+))	The unsigned content addressed by AR2 multiplied by the unsigned content addressed by AR3 is subtracted from the content of AC3 and the result is stored in AC3. AR2 and AR3 are both incremented by 1.

Before		After	
AC3	00 2300 EC00	AC3	FF B3E0 EC00
AR2	302	AR2	303
AR3	202	AR3	203
ACOV3	0	ACOV3	0
302	FE00	302	FE00
202	7000	202	7000
FRCT	0	FRCT	0

4.50 Negation

Syntax Characteristics

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

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.

Example

Syntax	Description
AC0 = –AC1	The 2s complement of the content of AC1 is stored in AC0.

4.51 No Operation (NOP)

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	nop	Yes	1	1	D
[2]	nop_16	Yes	2	1	D

Operands none

Description

Instruction [1] increments the program counter register (PC) by 1 byte. Instruction [2] increments the PC by 2 bytes.

Status Bits

Affected by none*

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
nop	The program counter (PC) is incremented by 1 byte.

4.52 Normalization

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = mant(ACx), Tx = –exp(ACx)	Yes	3	1	X
[2]	Tx = exp(ACx)	Yes	3	1	X

Brief Description

Instruction [1] 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.

Instruction [2] 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.

Instruction [2] produces in Tx the opposite result than computed by instruction [1].

Status Bits

Affected by none

Affects none

4.52.1 Normalization: ACy = mant(ACx), Tx = –exp(ACx)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = mant(ACx), Tx = –exp(ACx)	Yes	3	1	X

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 Tx. The A-unit is used to make the move operation. The mantissa is stored in 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.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Examples

Syntax

AC1 = mant(AC0), T1 = –exp(AC0)

Description

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

AC1 = mant(AC0), T1 = –exp(AC0)

Description

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	00 E804 0000	AC0	00 E804 0000
AC1	FF FFFF F001	AC1	00 7402 0000
T1	0000	T1	0001

Syntax

AC2 = mant(AC2), T2 = –exp(AC2)

Description

The exponent is computed by subtracting the number of leading bits in the content of AC2 from 8. The exponent value is a signed 2s-complement value in the –31 to 8 range and is stored in T2. The mantissa is computed by aligning the content of AC2 on a signed 32-bit representation. The mantissa value is stored in AC2.

Before		After	
AC2	00 0723 2400	AC2	00 7232 4000
T2	0000	T2	FFFC

Syntax

AC1 = mant(AC1), T3 = –exp(AC1)

Description

The exponent is computed by subtracting the number of leading bits in the content of AC1 from 8. The exponent value is a signed 2s-complement value in the –31 to 8 range and is stored in T3. The mantissa is computed by aligning the content of AC1 on a signed 32-bit representation. The mantissa value is stored in AC1.

Before		After	
AC1	F9 3400 8600	AC0	FF 9340 0860
T1	0000	T1	0004

4.52.2 Exponent: Tx = exp(ACx)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	Tx = exp(ACx)	Yes	3	1	X

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 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 instruction [1].

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Examples

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
```

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          07 8543 2105       AC0          07 8543 2105
T1                   0000       T1                   FFFC
```

4.53 Peripheral Port Register Access Qualifiers

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	readport()	No	1	1	D
[2]	writeport()	No	1	1	D

Brief 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.

- ❏ Operand qualifier [1] may be paralleled with any instruction making a word single data memory access Smem or Xmem that is used to read a memory operand.
- ❏ Operand qualifier [2] may be paralleled with any instruction making a word single data memory access Smem or Ymem that is used to write a memory operand, except instructions using the keyword delay().

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 unsigned 16-bit 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 these operand qualifiers.

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 in parallel with these operand qualifiers.

- ❏ *ARn(#K16)
- ❏ *+ARn(#K16)
- ❏ *CDP(#K16)
- ❏ *+CDP(#K16)

Status Bits

Affected by none

Affects none

4.53.1 Peripheral Port Register Access Qualifier: readport()

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	readport()	No	1	1	D

Operands

none

Description

This operand qualifer allows 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.

- This operand qualifer may be paralleled with any instruction making a word single data memory access Smem or Xmem that is used to read a memory operand.
- This operand qualifer cannot be executed in parallel with other types of instructions; however, the memory move instruction, Smem = Cmem, can also be paralleled with this readport() qualifier.
- This operand qualifer cannot be executed alone.

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 unsigned 16-bit 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 this operand qualifier.

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 in parallel with this operand qualifier.

- *ARn(#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

Syntax	Description
T2 = *AR3 || readport()	The content addressed by AR3 (I/O address) is loaded into T2.

4.53.2 Peripheral Port Register Access Qualifier: writeport()

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	writeport()	No	1	1	D

Operands none

Description

This operand qualifer allows 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.

- This operand qualifer may be paralleled with any instruction making a word single data memory access Smem or Ymem that is used to write a memory operand, except instructions using the key-word delay().
- This operand qualifer cannot be executed in parallel with other types of instructions; however, the memory move instruction, Cmem = Smem, can also be paralleled with this writeport() qualifier.
- This operand qualifer cannot be executed alone.

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 unsigned 16-bit 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 this operand qualifier.

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 in parallel with this operand qualifier.

- *ARn(#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

Syntax	Description
*AR3 = T2 || writeport()	The content of T2 is written to the location addressed by AR3 (I/O address).

4.54 Pop Extended Auxiliary Register from Stack Pointers

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	xdst = popboth()	Yes	2	1	X

Operands

xdst

Description

This instruction moves the content of two 16-bit data memory locations addressed by the 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.

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.

4.55 Pop Top of Stack (TOS)

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst1, dst2 = pop()	Yes	2	1	X
[2]	dst = pop()	Yes	2	1	X
[3]	dst, Smem = pop()	No	3	1	X
[4]	ACx = dbl(pop())	Yes	2	1	X
[5]	Smem = pop()	No	2	1	X
[6]	dbl(Lmem) = pop()	No	2	1	X

Brief Description

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

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

4.55.1 Pop Top of Stack: dst1, dst2 = pop()

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst1, dst2 = pop()	Yes	2	1	X

Operands

dst

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.

Example

Syntax	Description
AC0, AC1 = pop()	The content of the memory location pointed by the 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

4.55.2 Pop Top of Stack: dst = pop()

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	dst = pop()	Yes	2	1	X

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 can be repeated.

Example

Syntax	Description
AC0 = pop()	The content of the memory location pointed by the stack pointer (SP) is copied to AC0(15–0). Bits 39–16 of AC0 are unchanged. The SP is incremented by 1.

4.55.3 Pop Top of Stack: dst, Smem = pop()

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	dst, Smem = pop()	No	3	1	X

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.

Example

Syntax	Description
AC0, *AR3 = pop()	The content of the memory location pointed by the 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.

4.55.4 Pop Top of Stack: ACx = dbl(pop())

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	ACx = dbl(pop())	Yes	2	1	X

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 can be repeated.

Example

Syntax	Description
AC1 = dbl(pop())	The content of the memory location pointed by the 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
```

4.55.5 Pop Top of Stack: Smem = pop()

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	Smem = pop()	No	2	1	X

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.

Example

Syntax	Description
*AR1 = pop()	The content of the memory location pointed by the 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

4.55.6 Pop Top of Stack: dbl(Lmem) = pop()

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	dbl(Lmem) = pop()	No	2	1	X

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 16 lowest bits of the 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.

Example

Syntax	Description
dbl(*AR3–) = pop()	The content of the memory location pointed by the 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.

4.56 Push Extended Auxiliary Register to Stack Pointers

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	pshboth(xsrc)	Yes	2	1	X

Operands

xsrc

Description

This instruction moves the lower 32 bits of ACx or the content of the 23-bit source register (XARx, XSP, XSSP, XDP, or XCDP) to the two 16-bit memory locations addressed by the stack pointer (SP) and system stack pointer (SSP).

The content of xsrc(15–0) is moved to the location addressed by SP and the content of xsrc(31–16) is moved to the location addressed by SSP.

When xsrc is a 23-bit register, the upper 9 bits of the location addressed by SSP are filled with 0.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

4.57 Push to Top of Stack (TOS)

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	push(src1, src2)	Yes	2	1	X
[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

Brief Description

These instructions move one or two operands to the data memory location addressed by the data stack pointer (SP). The operands may be:

- an accumulator, auxiliary or temporary register
- a data memory location

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

4.57.1 Push to Top of Stack: push(src1, src2)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	push(src1, src2)	Yes	2	1	X

Operands

src

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.

Example

Syntax	Description
push(AR0, AC1)	The 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

4.57.2 Push to Top of Stack: push(src)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	push(src)	Yes	2	1	X

Operands

src

Description

This instruction decrements SP by 1, then moves the content of the source register (src) to the 16-bit data memory location pointed by SP. 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.

Example

Syntax	Description
push(AC0)	The SP is decremented by 1. The content of AC0(15–0) is copied to the memory location pointed by the stack pointer (SP).

4.57.3 Push to Top of Stack: push(src, Smem)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	push(src, Smem)	No	3	1	X

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.

Example

Syntax	Description
push(AC0, *AR3)	The SP is decremented by 2. The content of AC0(15–0) is copied to the memory location pointed by the stack pointer (SP) and the content addressed by AR3 is copied to the memory location pointed by SP + 1.

4.57.4 Push to Top of Stack: dbl(push(ACx))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	dbl(push(ACx))	Yes	2	1	X

Operands

ACx

Description

This instruction decrements SP by 2, then moves the content of the 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 can be repeated.

Example

Syntax	Description
dbl(push(AC0))	The SP is decremented by 2. The content of AC0(31–16) is copied to the memory location pointed by the stack pointer (SP) and the content of AC0(15–0) is copied to the memory location pointed by SP + 1.

4.57.5 Push to Top of Stack: push(Smem)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	push(Smem)	No	2	1	X

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.

Example

Syntax	Description
push(*AR1)	The SP is decremented by 1. The content addressed by AR1 is copied to the memory location pointed by the stack pointer (SP).

Before		After	
*AR1	6903	*AR1	6903
SP	0305	SP	0304
304	0000	304	6903
305	0300	305	0300

4.57.6 Push to Top of Stack: push(dbl(Lmem))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	push(dbl(Lmem))	No	2	1	X

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.

Example

Syntax	Description
push(dbl(*AR3–))	The 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 the stack pointer (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.

4.58 Register Bit Test/Set/Clear/Complement

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	TCx = bit(src, Baddr)	No	3	1	X
[2]	cbit(src, Baddr)	No	3	1	X
[3]	bit(src, Baddr) = #0	No	3	1	X
[4]	bit(src, Baddr) = #1	No	3	1	X
[5]	bit(src, pair(Baddr))	No	3	1	X

Brief Description

These instructions perform bit manipulations:

- ❑ In the D-unit ALU, if the register operand is an accumulator.
- ❑ In the A-unit ALU, if the register operand is an auxiliary or temporary register.

These instructions manipulate a single bit (instructions [1]–[4]) or two consecutive bits (instruction [5]) of the source (src) register. The bit manipulated is defined by a bit address (Baddr). The following operations can be performed on the selected bit:

- ❑ copy the bit into TCx
- ❑ complement the bit in src
- ❑ clear the bit in src
- ❑ set the bit in src

Status Bits

Affected by none

Affects TCx

4.58.1 Register Bit Test: TCx = bit(src, Baddr)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	TCx = bit(src, Baddr)	No	3	1	X

Operands Baddr, src, TCx

Description

This instruction performs a bit manipulation:

- In the D-unit ALU, if the source (src) register operand is an accumulator.
- In the A-unit ALU, if the source (src) register operand is an auxiliary or temporary register.

The instruction tests a single bit of the source register location as defined by the bit addressing mode, Baddr. The tested bit is copied into the selected TCx status 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.
- 0–15 when accessing auxiliary or temporary register bits (only the 4 LSBs of the generated address are used to determine the bit position).

Status Bits

Affected by none

Affects TCx

Repeat

This instruction can be repeated.

Example

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
T0              FE00    T0              FE00
TC1                0    TC1                1
```

4.58.2 Register Bit Complement: cbit(src, Baddr)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	cbit(src, Baddr)	No	3	1	X

Operands

Baddr, src

Description

This instruction performs a bit manipulation:

- In the D-unit ALU, if the source (src) register operand is an accumulator.
- In the A-unit ALU, if the source (src) register operand is an auxiliary or temporary register.

The instruction complements a single bit, as defined by the bit addressing mode, Baddr, of the source register.

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, the selected register bit value does not change.
- 0–15 when accessing auxiliary or temporary register bits (only the 4 LSBs of the generated address are used to determine the bit position).

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

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

4.58.3 Register Bit Clear: bit(src, Baddr) = #0

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	bit(src, Baddr) = #0	No	3	1	X

Operands Baddr, src

Description

This instruction performs a bit manipulation:

- In the D-unit ALU, if the source (src) register operand is an accumulator.
- In the A-unit ALU, if the source (src) register operand is an auxiliary or temporary register.

The instruction clears to 0 a single bit, as defined by the bit addressing mode, Baddr, of the source register.

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, the selected register bit value does not change.
- 0–15 when accessing auxiliary or temporary register bits (only the 4 LSBs of the generated address are used to determine the bit position).

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

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.

4.58.4 Register Bit Set: bit(src, Baddr) = #1

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	bit(src, Baddr) = #1	No	3	1	X

Operands

Baddr, src

Description

This instruction performs a bit manipulation:

- In the D-unit ALU, if the source (src) register operand is an accumulator.
- In the A-unit ALU, if the source (src) register operand is an auxiliary or temporary register.

The instruction sets to 1 a single bit, as defined by the bit addressing mode, Baddr, of the source register.

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, the selected register bit value does not change.
- 0–15 when accessing auxiliary or temporary register bits (only the 4 LSBs of the generated address are used to determine the bit position).

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

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.

4.58.5 Register Bit Pair Test: bit(src, pair(Baddr))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	bit(src, pair(Baddr))	No	3	1	X

Operands Baddr, src

Description

This instruction performs a bit manipulation:

- In the D-unit ALU, if the source (src) register operand is an accumulator.
- In the A-unit ALU, if the source (src) register operand is an auxiliary or temporary register.

The instruction tests two consecutive bits of the source register location as defined by the bit addressing mode, Baddr, and Baddr + 1. The tested bits are copied into status bits TC1 and TC2:

- TC1 tests the bit that is defined by Baddr
- TC2 tests the bit defined by Baddr + 1

The generated bit address must be within:

- 0–38 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 0–38:
  - 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.

Example

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	E0 1234 0000	AC0	E0 1234 0000
AR1	0026	AR1	0026
T0	0001	T0	0001
TC1	0	TC1	1
TC2	0	TC2	0

4.59 Register Comparison

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	TCx = uns(src RELOP dst)	Yes	3	1	X
[2]	TCx = TCy & uns(src RELOP dst)	Yes	3	1	X
[3]	TCx = !TCy & uns(src RELOP dst)	Yes	3	1	X
[4]	TCx = TCy | uns(src RELOP dst)	Yes	3	1	X
[5]	TCx = !TCy | uns(src RELOP dst)	Yes	3	1	X

Brief Description

These instructions perform 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.

Status Bits

Affected by C54CM, M40, TCy

Affects TCx

4.59.1 Register Comparison: TCx = uns(src RELOP dst)

Syntax Characteristics

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

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 keywords 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.

Examples

Syntax	Description
TC1= (AC1 = = T1)	The 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

4.59.2 Register Comparison/AND: TCx = TCy & uns(src RELOP dst)

Syntax Characteristics

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

Operands dst, RELOP, src, TCx, TCy

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 keywords 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.

Example

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

4.59.3 Register Comparison/AND: TCx = !TCy & uns(src RELOP dst)

Syntax Characteristics

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

Operands dst, RELOP, src, TCx, TCy

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 keywords 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.

Example

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	0

4.59.4 Register Comparison/OR: TCx = TCy | uns(src RELOP dst)

Syntax Characteristics

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

Operands dst, RELOP, src, TCx, TCy

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 keywords 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.

Example

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

4.59.5 Register Comparison/OR: TCx = !TCy | uns(src RELOP dst)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	TCx = !TCy | uns(src RELOP dst)	Yes	3	1	X

Operands dst, RELOP, src, TCx, TCy

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 keywords 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.

Example

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

4.60 Repeat Block of Instructions Unconditionally

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	localrepeat{}	Yes	2	1	AD
[2]	blockrepeat{}	Yes	3	1	AD

Brief Description

These instructions repeat a block of instructions the number of times specified by:

- ❑ the content of BRC0 + 1, if no loop has already been detected.
- ❑ the content of BRS1 + 1, if one level of the loop has already been detected.

Loop structures defined by these instructions must have the following characteristics:

- ❑ The minimum number of cycles executed within one loop iteration is 2 cycles.
- ❑ The maximum loop size is 64K bytes.
- ❑ The block-repeat counter registers (BRCx) must be read 3 full cycles before the end of the loops in order to extract the correct loop iteration number from these registers without any pipeline stall.
- ❑ The block-repeat operation can only be cleared by branching to a destination address outside the active block-repeat loop.

These instructions cannot be repeated.

Status Bits

Affected by none

Affects none

4.60.1 Repeat Block of Instructions Unconditionally: localrepeat{}

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	localrepeat{}	Yes	2	1	AD

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 address 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 address 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.

Loop structures defined by this instruction must have the following characteristics:

- The minimum number of cycles executed within one loop iteration is 2 cycles.
- 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.
- The block-repeat counter registers (BRCx) must be read 3 full cycles before the end of the loops in order to extract the correct loop iteration number from these registers without any pipeline stall.

A loop is defined as local when all the code of the loop is repeatedly executed from within the instruction buffer queue:

- Local loop sizes are limited to 61 bytes. (The assembler checks that the code size of the loop excluding the last (paralleled) instruction is less than or equal to 55 bytes.)
- Nested local repeat block instructions are allowed.
- Local loop code must not include the following instructions:

goto	reset	idle	return
call	intr	blockrepeat	trap

- The only branch instructions allowed in a localrepeat structure are the branch instructions with a target branch address pointing to an instruction included within the loop code and being at a higher address than the branching instruction. In this case, the branch conditionally instruction is executed in 3 cycles and the condition is evaluated in the address phase of the pipeline (there is a 3-cycle latency on the condition setting).

Compatibility with C54x devices (C54CM = 1)

When 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-repeat 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.

Status Bits

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

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

4.60.2 Repeat Block of Instructions Unconditionally: blockrepeat{}

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	blockrepeat{}	Yes	3	1	AD

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 address 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 address 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.

Loop structures defined by these instructions must have the following characteristics:

- The minimum number of cycles executed within one loop iteration is 2 cycles.
- 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.
- The block-repeat counter registers (BRCx) must be read 3 full cycles before the end of the loops in order to extract the correct loop iteration number from these registers without any pipeline stall.

Compatibility with C54x devices (C54CM = 1)

When 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-repeat 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.

Status Bits

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

Example

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

4.61 Repeat Single Instruction Conditionally

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	while (cond && (RPTC < k8)) repeat	Yes	3	1	AD

Operands cond, k8

Description

This instruction evaluates a single condition defined by the cond field and as long as the condition is true, the next instruction or the next two paralleled instructions is repeated the number of times specified by an 8-bit immediate value, k8 + 1. The maximum number of executions of a given instruction or paralleled instructions is 2^8 –1 (255).

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.

The 8 MSBs of RPTC:

- ❏ Are loaded with the cond code at the address phase of the pipeline.
- ❏ Are untouched during the while() / repeat() structure execution.

At each step of the iteration, the condition defined by the cond field is tested in the execute phase of the pipeline. When the condition becomes false, the instruction repetition stops.

- ❏ If the condition becomes false at any execution of the repeated instruction, the 8 LSBs of RPTC are corrected to indicate exactly how many iterations were not performed.
- ❏ 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 [1] 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 met	RPTC[7:0] content after exiting loop	CSR content after exiting loop
At 1st iteration	RPTCinit + 1	3
At 2nd iteration	RPTCinit	3
At 3rd iteration	RPTC – 1	3
…	…	…
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.

When the while() / repeat() structure contains any form of a store-to-memory instruction, the store-to-memory instruction is only disabled one cycle after the condition is evaluated to be false. Therefore, the store-to-memory instruction is executed once more than other processing instructions updating CPU registers. This enables you to store the last values obtained in these registers when the condition was met.

Instead of programming a number of iterations (minus 1) equal to 0, it is recommended that you use the conditional execute() structure.

The following instructions cannot be used in a repeat single mechanism:

goto	localrepeat	reset	idle	RETA = dbl(Lmem)
call	repeat	intr	(cond) execute(AD_unit)	
return	blockrepeat	trap	(cond) execute(D_unit)	

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
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 7B40
T0	0340	T0	0340
*AR1	2354	*AR1	2354
RPTC	4106†	RPTC	0000

† At the address phase of the pipeline, RPTC is automatically initialized to 4107h and then is immediately decreased to 4106h.

4.62 Repeat Single Instruction Unconditionally

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	repeat(CSR)	Yes	2	1	AD
[2]	repeat(CSR), CSR += TAx	Yes	2	1	X
[3]	repeat(k8)	Yes	2	1	AD
[4]	repeat(CSR), CSR += k4	Yes	2	1	X
[5]	repeat(CSR), CSR –= k4	Yes	2	1	X
[6]	repeat(k16)	Yes	3	1	AD

Brief 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).

With the A-unit ALU, instructions [2], [4], and [5] allow the content of CSR to be modified. 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.

The following instructions cannot be used in a repeat single mechanism:

goto	localrepeat	reset	idle	TAx = RPTC
call	repeat	intr	(cond) execute(AD_unit)	RETA = dbl(Lmem)
return	blockrepeat	trap	(cond) execute(D_unit)	32-bit instructions using *(#k23) absolute addressing mode

These instructions cannot be repeated.

Status Bits

Affected by none

Affects none

4.62.1 Repeat Single Instruction Unconditionally: repeat(CSR)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	repeat(CSR)	Yes	2	1	AD

Operands

none

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.

The repeat single mechanism triggered by this instruction is interruptible.

Two paralleled instructions can be repeated when following the parallelism general rules.

The following instructions cannot be used in a repeat single mechanism:

goto	localrepeat	reset	idle	TAx = RPTC
call	repeat	intr	(cond) execute(AD_unit)	RETA = dbl(Lmem)
return	blockrepeat	trap	(cond) execute(D_unit)	32-bit instructions using *(#k23) absolute addressing mode

Status Bits

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

Example

Syntax

repeat(CSR)

AC1 = AC1 + *AR3+ * *AR4+

Description

The single instruction following the repeat instruction is repeated as defined 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

4.62.2 Repeat Single Instruction Unconditionally: repeat(CSR), CSR +=TAx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	repeat(CSR), CSR += TAx	Yes	2	1	X

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.

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.

The following instructions cannot be used in a repeat single mechanism:

goto	localrepeat	reset	idle	TAx = RPTC
call	repeat	intr	(cond) execute(AD_unit)	RETA = dbl(Lmem)
return	blockrepeat	trap	(cond) execute(D_unit)	32-bit instructions using *(#k23) absolute addressing mode

Status Bit

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

Example

Syntax	Description
repeat(CSR), CSR += T1	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.

4.62.3 Repeat Single Instruction Unconditionally: repeat(kx)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	repeat(k8)	Yes	2	1	AD
[6]	repeat(k16)	Yes	3	1	AD

Operands

kx

Description

This instruction repeats the next instruction or the next two paralleled instructions the number of times specified an immediate value, kx + 1. The repeat counter register (RPTC):

- Is loaded with the immediate value 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.

The repeat single mechanism triggered by this instruction is interruptible.

Two paralleled instructions can be repeated when following the parallelism general rules.

The following instructions cannot be used in a repeat single mechanism:

goto	localrepeat	reset	idle	TAx = RPTC
call	repeat	intr	(cond) execute(AD_unit)	RETA = dbl(Lmem)
return	blockrepeat	trap	(cond) execute(D_unit)	32-bit instructions using *(#k23) absolute addressing mode

Status Bits

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

Examples

Syntax	Description
repeat(#3) AC1 = AC1 + *AR3+ * *AR4+	The single instruction following the repeat instruction is repeated four times. This instruction example provides the same result as instruction [1].

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).

4.62.4 Repeat Single Instruction Unconditionally: repeat(CSR), CSR += k4

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	repeat(CSR), CSR += k4	Yes	2	1	X

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.

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.

The following instructions cannot be used in a repeat single mechanism:

goto	localrepeat	reset	idle	TAx = RPTC
call	repeat	intr	(cond) execute(AD_unit)	RETA = dbl(Lmem)
return	blockrepeat	trap	(cond) execute(D_unit)	32-bit instructions using *(#k23) absolute addressing mode

Status Bits

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

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 incremented by the unsigned 4-bit value (2).

4.62.5 Repeat Single Instruction Unconditionally: repeat(CSR), CSR –= k4

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	repeat(CSR), CSR –= k4	Yes	2	1	X

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.

With the A-unit ALU, this instruction allows the content of CSR to be decremented 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.

The following instructions cannot be used in a repeat single mechanism:

goto	localrepeat	reset	idle	TAx = RPTC
call	repeat	intr	(cond) execute(AD_unit)	RETA = dbl(Lmem)
return	blockrepeat	trap	(cond) execute(D_unit)	32-bit instructions using *(#k23) absolute addressing mode

Status Bits

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

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).

4.63 Return Conditionally

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	if (cond) return	Yes	3	5/5	R

x/y cycles: x cycles = condition true, y cycles = condition false

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.

When the return from subroutine occurs:

- The program counter (PC) is loaded with RETA content (the return address of the calling subroutine). The active control flow execution context flags are updated with the CFCT content.
- The 16 LSBs of RETA are popped from the top of the stack pointer (SP). The SP is incremented by 1 word.
- The 8 MSBs of RETA and CFCT are popped from the top of the system stack pointer (SSP). The SSP is incremented by 1 word.

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 (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	

4.64 Return Unconditionally

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	return	Yes	2	5	D

Operands none

Description

This instruction passes control back to the calling subroutine.

- The program counter (PC) is loaded with RETA content (the return address of the calling subroutine). The active control flow execution context flags are updated with the CFCT content.
- The 16 LSBs of RETA are popped from the top of the stack pointer (SP). The SP is incremented by 1 word in the address phase of the pipeline.
- The 8 MSBs of RETA and CFCT 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.

Status Bits

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

Example

Syntax	Description
return	The program counter is loaded with the return address of the calling subroutine.

4.65 Return from Interrupt

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	return_int	Yes	2	5	D

Operands none

Description

This instruction passes control back to the interrupted task.

- ❑ The program counter (PC) is loaded with RETA content (the return address of the interrupted task). The active control flow execution context flags are updated with the CFCT content.
- ❑ The 16 LSBs of RETA are popped from the top of the stack pointer (SP). The SP is incremented by 1 word in the address phase of the pipeline.
- ❑ The 8 MSBs of RETA and CFCT 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 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 debug status register (DBGSTAT) content is popped from the top of SSP. The SSP is incremented by 1 word in the access phase of the pipeline.
- ❑ The 16 bits of status register 2, ST2_55[15–0], are popped from the top of SP. The SP is incremented by 1 word in the read phase of the pipeline.
- ❑ The 7 higher bits of status register 0, ST0_55[15–9], are popped from the top of SSP. The SSP is incremented by 1 word in the read phase of the pipeline.

Status Bits

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

Example

Syntax	Description
return_int	The program counter (PC) is loaded with the return address of the interrupted task.

4.66 Rotate Left

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = BitOut \\ src \\ BitIn				
[1a]	dst = TC2 \\ src \\ TC2	Yes	3	1	X
[1b]	dst = TC2 \\ src \\ CARRY	Yes	3	1	X
[1c]	dst = CARRY \\ src \\ TC2	Yes	3	1	X
[1d]	dst = CARRY \\ src \\ CARRY	Yes	3	1	X

Operands BitIn, BitOut, 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.

Example

Syntax	Description
AC1 = CARRY \\ AC1 \\ TC2	The value of TC2 before the execution of the instruction (1) 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	0F E340 5678	AC1	00 C680 ACF1
TC2	1	TC2	1
CARRY	1	CARRY	1
M40	0	M40	0

4.67 Rotate Right

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = BitIn // src // BitOut				
[1a]	dst = TC2 // src // TC2	Yes	3	1	X
[1b]	dst = TC2 // src // CARRY	Yes	3	1	X
[1c]	dst = CARRY // src // TC2	Yes	3	1	X
[1d]	dst = CARRY // src // CARRY	Yes	3	1	X

Operands BitIn, BitOut, 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.

Example

Syntax	Description
AC1 = TC2 // AC0 // TC2	The value of TC2 before the execution of the instruction (1) 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

4.68 Round

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(ACx)	Yes	2	1	X

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:
  - When RDM = 0, the biased rounding to the infinite is performed. 8000h (2^{15}) 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 (2^{15}) 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.

Examples

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

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 7023	AC0	EF 0FF0 7023
AC1	00 0000 0000	AC1	EF 0FF0 0000
RDM	1	RDM	1
M40	0	M40	0
SATD	0	SATD	0
ACOV1	0	ACOV1	1

4.69 Saturate

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = saturate(rnd(ACx))	Yes	2	1	X

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^{15}) 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 (2^{15}) 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

Affects ACOVy

Repeat

This instruction can be repeated.

Examples

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 0FF0 8023	AC0	EF 0FF0 8023
AC1	00 0000 0000	AC1	FF 8000 0000
ACOV1	0	ACOV1	1

Syntax	Description
AC1 = saturate(rnd(AC0))	The 32-bit width content of AC0 is saturated. The saturated value, 00 7FFF FFFFh, 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	0	RDM	0
ACOV1	0	ACOV1	1

4.70 Signed Shift

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = dst >> #1	Yes	2	1	X
[2]	dst = dst << #1	Yes	2	1	X
[3]	ACy = ACx << Tx	Yes	2	1	X
[4]	ACy = ACx <<C Tx	Yes	2	1	X
[5]	ACy = ACx << #SHIFTW	Yes	3	1	X
[6]	ACy = ACx <<C #SHIFTW	Yes	3	1	X

Brief Description

These instructions perform an unsigned shift by an immediate value, 1 or SHIFTW, or the content of a temporary register (Tx):

- 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

4.70.1 Signed Shift Right: dst = dst >> #1

Syntax Characteristics

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

Operands dst

Description

This instruction shifts right by 1 bit the content of the destination register (dst).

If the destination operand (dst) is an accumulator:

- 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 right by 1 bit:
  - 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.
- After shifting, unless otherwise noted, when M40 = 0:
  - overflow is detected at bit position 31
  - 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 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.
- Bit 15 is sign extended.
- After shifting, unless otherwise noted:
  - overflow is detected at bit position 15
  - 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 none

Repeat

This instruction 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.

4.70.2 Signed Shift Left: dst = dst << #1

Syntax Characteristics

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

Operands dst

Description

This instruction shifts left by 1 bit the content of the destination register (dst).

If the destination operand (dst) is an accumulator:

- 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 left by 1 bit:
  - 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.
- After shifting, unless otherwise noted, when M40 = 0:
  - overflow is detected at bit position 31 (if an overflow is detected, the destination ACOVx bit is set)
  - 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 ACOVx 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)

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.

Example

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

4.70.3 Signed Shift: ACy = ACx << Tx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ACy = ACx << Tx	Yes	2	1	X

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.
- 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 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.
- 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.

Example

Syntax	Description
AC0 = AC1 << T0	The content of AC1 is shifted by the content of T0 and the result is stored in AC0.

4.70.4 Signed Shift: ACy = ACx <<C Tx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	ACy = ACx <<C Tx	Yes	2	1	X

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.
- 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 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.
- 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.

Example

Syntax	Description
AC2 = AC2 <<C T1	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

4.70.5 Signed Shift: ACy = ACx << #SHIFTW

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	ACy = ACx << #SHIFTW	Yes	3	1	X

Operands ACx, ACy, SHIFTW

Description

This instruction shifts by a 6-bit value, SHIFTW, the accumulator (ACx) content.

- 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 6-bit value, SHIFTW:
  - 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.
- 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 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.

Examples

Syntax	Description
AC0 = AC1 << #31	The content of AC1 is shifted left by 31 bits and the result is stored in AC0.
AC0 = AC1 << #–32	The content of AC1 is shifted right by 32 bits and the result is stored in AC0.

4.70.6 Signed Shift: ACy = ACx <<C #SHIFTW

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	ACy = ACx <<C #SHIFTW	Yes	3	1	X

Operands ACx, ACy, SHIFTW

Description

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 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 6-bit value, SHIFTW:
  - 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, SHIFTW = 0, the CARRY status bit is cleared to 0.
- 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 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, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC1 = AC0 <<C #–5	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

4.71 Software Interrupt

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	intr(k5)	No	2	3	D

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 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.

When the control is passed to the ISR:

- ❏ The stack pointer (SP) is decremented by 1 word in the address phase of the pipeline. The 16 bits of status register 2, ST2_55[15–0], 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 7 higher bits of status register 0, ST0_55[15–9], concatenated to 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 (DBGSTAT) 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 RETA (return auxiliary register) are pushed to the top of SP.
- ❏ The SSP is decremented by 1 word in the read phase of the pipeline. The 8 MSBs of RETA and the control flow execution context flags register (CFCT) are pushed to the top of SSP.
- ❏ The return address of the interrupt is saved in RETA. The active control flow execution context flags are saved in CFCT.
- ❏ The program counter (PC) is loaded with the ISR program address. The active control flow execution context flags are cleared.

Status Bits

Affected by none

Affects INTM

Repeat

This instruction cannot be repeated.

Example

Syntax	Description
intr(#3)	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 (3).

4.72 Software Reset

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	reset	No	2	?	D

Operands none

Description

This instruction performs a nonmaskable software reset that can be used any time to put the device in a known state.

The reset instruction affects ST0_55, ST1_55, ST2_55, IFR0, and IFR1 registers but does not affect status register ST3_55 and interrupt vectors pointer registers (IVPD and IVPH). When the reset instruction is acknowledged, the INTM is set to 1 to disable maskable interrupts. All pending interrupts in IFR0 and IFR1 are cleared. The initialization of the system control register, the interrupt vectors pointer, and the peripheral registers is different from the initialization performed by a hardware reset.

Status Bits

Affected by none

Affects IFR0, IFR1, ST0_55, ST1_55, ST2_55

Repeat

This instruction cannot be repeated.

4.73 Software Trap

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	trap(k5)	No	2	?	D

Operands k5

Description

This instruction passes control to a specified interrupt service routine (ISR). 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 and this instruction does not affect INTM bit. This instruction is not maskable.

When the control is passed to the ISR:

- The stack pointer (SP) is decremented by 1 word in the address phase of the pipeline. The 16 bits of status register 2, ST2_55[15–0], 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 7 higher bits of status register 0, ST0_55[15–9], concatenated to 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 (DBGSTAT) 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 RETA (return auxiliary register) are pushed to the top of SP.
- The SSP is decremented by 1 word in the read phase of the pipeline. The 8 MSBs of RETA and the control flow execution context flags register (CFCT) are pushed to the top of SSP.
- The return address of the interrupt is saved in RETA. The active control flow execution context flags are saved in CFCT.
- The program counter (PC) is loaded with the ISR program address. The active control flow execution context flags are cleared.

Status Bits

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

Example

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).

4.74 Specific CPU Register Load

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	BK03 = k12	Yes	3	1	AD
[2]	BK47 = k12	Yes	3	1	AD
[3]	BKC = k12	Yes	3	1	AD
[4]	BRC0 = k12	Yes	3	1	AD
[5]	BRC1 = k12	Yes	3	1	AD
[6]	CSR = k12	Yes	3	1	AD
[7]	MDP = k7	Yes	3	1	AD
[8]	PDP = k9	Yes	3	1	AD
[9]	BOF01 = k16	No	4	1	AD
[10]	BOF23 = k16	No	4	1	AD
[11]	BOF45 = k16	No	4	1	AD
[12]	BOF67 = k16	No	4	1	AD
[13]	BOFC = k16	No	4	1	AD
[14]	CDP = k16	No	4	1	AD
[15]	DP = k16	No	4	1	AD
[16]	SP = k16	No	4	1	AD
[17]	SSP = k16	No	4	1	AD
[18]	BK03 = Smem	No	3	1	X
[19]	BK47 = Smem	No	3	1	X
[20]	BKC = Smem	No	3	1	X
[21]	BOF01 = Smem	No	3	1	X
[22]	BOF23 = Smem	No	3	1	X
[23]	BOF45 = Smem	No	3	1	X
[24]	BOF67 = Smem	No	3	1	X
[25]	BOFC = Smem	No	3	1	X
[26]	BRC0 = Smem	No	3	1	X
[27]	BRC1 = Smem	No	3	1	X
[28]	CDP = Smem	No	3	1	X
[29]	CSR = Smem	No	3	1	X
[30]	DP = Smem	No	3	1	X
[31]	MDP = Smem	No	3	1	X
[32]	PDP = Smem	No	3	1	X
[33]	SP = Smem	No	3	1	X
[34]	SSP = Smem	No	3	1	X
[35]	TRN0 = Smem	No	3	1	X

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[36]	TRN1 = Smem	No	3	1	X
[37]	RETA = dbl(Lmem)	No	3	5	X

Brief Description

These instructions load an unsigned constant, kx, the content of a memory (Smem) location, or the content of a data memory operand (Lmem) to a selected destination CPU register.

Status Bits

Affected by none

Affects none

4.74.1 Specific CPU Register Load: cpureg = kx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	BK03 = k12	Yes	3	1	AD
[2]	BK47 = k12	Yes	3	1	AD
[3]	BKC = k12	Yes	3	1	AD
[4]	BRC0 = k12	Yes	3	1	AD
[5]	BRC1 = k12	Yes	3	1	AD
[6]	CSR = k12	Yes	3	1	AD
[7]	MDP = k7	Yes	3	1	AD
[8]	PDP = k9	Yes	3	1	AD
[9]	BOF01 = k16	No	4	1	AD
[10]	BOF23 = k16	No	4	1	AD
[11]	BOF45 = k16	No	4	1	AD
[12]	BOF67 = k16	No	4	1	AD
[13]	BOFC = k16	No	4	1	AD
[14]	CDP = k16	No	4	1	AD
[15]	DP = k16	No	4	1	AD
[16]	SP = k16	No	4	1	AD
[17]	SSP = k16	No	4	1	AD

Operands

kx

Description

This instruction loads 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 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.

4.74.2 Specific CPU Register Load: cpureg = Smem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[18]	BK03 = Smem	No	3	1	X
[19]	BK47 = Smem	No	3	1	X
[20]	BKC = Smem	No	3	1	X
[21]	BOF01 = Smem	No	3	1	X
[22]	BOF23 = Smem	No	3	1	X
[23]	BOF45 = Smem	No	3	1	X
[24]	BOF67 = Smem	No	3	1	X
[25]	BOFC = Smem	No	3	1	X
[26]	BRC0 = Smem	No	3	1	X
[27]	BRC1 = Smem	No	3	1	X
[28]	CDP = Smem	No	3	1	X
[29]	CSR = Smem	No	3	1	X
[30]	DP = Smem	No	3	1	X
[31]	MDP = Smem	No	3	1	X
[32]	PDP = Smem	No	3	1	X
[33]	SP = Smem	No	3	1	X
[34]	SSP = Smem	No	3	1	X
[35]	TRN0 = Smem	No	3	1	X
[36]	TRN1 = Smem	No	3	1	X

Operands Smem

Description

This instruction loads 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 MDP, PDP, DP, SP, SSP, CDP, BOFx, 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 [27], when BRC1 is loaded, the block repeat save register (BRS1) is loaded with the same value.

Status Bits

Affected by none

Affects none

Repeat

Instruction [30] cannot be repeated; all other instructions can be repeated.

4.74.3 Return Address Register Load: RETA = dbl(Lmem)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[37]	RETA = dbl(Lmem)	No	3	5	X

Operands

Lmem

Description

This instruction 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 16 highest bits of Lmem are loaded into the CFCT register and into 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 this instruction 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

This instruction can be repeated.

4.75 Specific CPU Register Move

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	BRC0 = TAx	Yes	2	1	X
[2]	BRC1 = TAx	Yes	2	1	X
[3]	CDP = TAx	Yes	2	1	X
[4]	CSR = TAx	Yes	2	1	X
[5]	TAx = BRC0	Yes	2	1	X
[6]	TAx = BRC1	Yes	2	1	X
[7]	TAx = CDP	Yes	2	1	X
[8]	TAx = RPTC	Yes	2	1	X
[9]	TAx = SP	Yes	2	1	X
[10]	TAx = SSP	Yes	2	1	X
[11]	SP = TAx	Yes	2	1	X
[12]	SSP = TAx	Yes	2	1	X

Brief Description

These instructions move the content of the auxiliary or temporary register (TAx) to the selected CPU register, or move 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.

Status Bits

Affected by none

Affects none

4.75.1 Auxiliary or Temporary Register Move: cpureg = TAx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	BRC0 = TAx	Yes	2	1	X
[2]	BRC1 = TAx	Yes	2	1	X
[3]	CDP = TAx	Yes	2	1	X
[4]	CSR = TAx	Yes	2	1	X
[11]	SP = TAx	Yes	2	1	X
[12]	SSP = TAx	Yes	2	1	X

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.

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

4.75.2 BRCx Register Move: TAx = BRCx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	TAx = BRC0	Yes	2	1	X
[6]	TAx = BRC1	Yes	2	1	X

Operands

TAx

Description

This instruction moves the content of the selected BRCx to the auxiliary or temporary register (TAx). Since BRCx is decremented in the address phase of the last instruction of a loop, these move instructions have a 3-cycle latency requirement versus the last instruction of a loop.

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.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
T1 = BRC1	The content of block repeat register (BRC1) is copied to T1.

Before		After	
T1	0034	T1	00EA
BRC1	00EA	BRC1	00EA

4.75.3 Specific CPU Register Move: TAx = cpureg

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	TAx = CDP	Yes	2	1	X
[8]	TAx = RPTC	Yes	2	1	X
[9]	TAx = SP	Yes	2	1	X
[10]	TAx = SSP	Yes	2	1	X

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 [7], [9], and [10], 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

Instructions [7], [9], and [10] can be repeated. Instruction [8] cannot be repeated.

Example

Syntax	Description
T1 = RPTC	The content of single-repeat counter (RPTC) is copied to T1.

Before		After	
T1	0034	T1	00EA
RPTC	00EA	RPTC	00EA

4.76 Specific CPU Register Store

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	Smem = BK03	No	3	1	X
[2]	Smem = BK47	No	3	1	X
[3]	Smem = BKC	No	3	1	X
[4]	Smem = BOF01	No	3	1	X
[5]	Smem = BOF23	No	3	1	X
[6]	Smem = BOF45	No	3	1	X
[7]	Smem = BOF67	No	3	1	X
[8]	Smem = BOFC	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 = DP	No	3	1	X
[14]	Smem = MDP	No	3	1	X
[15]	Smem = PDP	No	3	1	X
[16]	Smem = SP	No	3	1	X
[17]	Smem = SSP	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

Brief Description

These instructions store the content of the selected source CPU register to a memory (Smem) location or a data memory operand (Lmem).

Status Bits

Affected by none

Affects none

4.76.1 Specific CPU Register Store: Smem = cpureg

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	Smem = BK03	No	3	1	X
[2]	Smem = BK47	No	3	1	X
[3]	Smem = BKC	No	3	1	X
[4]	Smem = BOF01	No	3	1	X
[5]	Smem = BOF23	No	3	1	X
[6]	Smem = BOF45	No	3	1	X
[7]	Smem = BOF67	No	3	1	X
[8]	Smem = BOFC	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 = DP	No	3	1	X
[14]	Smem = MDP	No	3	1	X
[15]	Smem = PDP	No	3	1	X
[16]	Smem = SP	No	3	1	X
[17]	Smem = SSP	No	3	1	X
[18]	Smem = TRN0	No	3	1	X
[19]	Smem = TRN1	No	3	1	X

Operands

Smem

Description

This instruction stores the content of the source CPU register to a memory (Smem) location.

The block repeat register (BRCx) is decremented in the address phase of the last instruction of the loop. Instructions [9] and [10] have a 3-cycle latency requirement versus the last instruction of the loop.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Examples

Syntax	Description
*AR1+ = SP	The content of the 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

Syntax	Description
*AR1+ = SSP	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

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

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

4.76.2 Return Address Register Store: dbl(Lmem) = RETA

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[20]	dbl(Lmem) = RETA	No	3	5	X

Operands

Lmem

Description

This instruction stores 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) 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 this instruction 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

This instruction can be repeated.

Example

Syntax	Description
dbl(*AR3) = RETA	The contents of the RETA and CFCT are stored in the location addressed by AR3 and AR3 + 1.

4.77 Square Distance (sqdst)

Syntax Characteristics

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

Operands ACx, ACy, Xmem, Ymem

Description

This instruction performs two parallel operations: multiply and accumulate (MAC), and subtract. The operations are executed in the D-unit MAC and D-unit ALU:

```
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 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 identical 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 FRCT, SMUL, C54CM, M40, SATD, SXMD

Affects ACOVx, ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

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	*AR0	0055
*AR1	00AA	*AR1	00AA
ACOV0	0	ACOV0	0
ACOV1	0	ACOV1	0
CARRY	0	CARRY	0
FRCT	0	FRCT	0

4.78 Status Bit Set/Clear

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

† When instruction [7] or [8] 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.

Brief Description

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

These instructions set to 1 or clear to 0 a single bit, as defined by a 4-bit immediate value, k4, in the selected status register (ST0_55, ST1_55, ST2_55, or ST3_55). Note that C55x status bit mapping does not correspond to C54x status bits.

These instructions cannot be repeated.

Status Bits

Affected by none

Affects Selected status bits

4.78.1 Status Bit Clear: bit(STx, k4) = #0

Syntax Characteristics

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

† When instruction [7] 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.

Operands

k4, STx_55

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_55, ST1_55, ST2_55, or ST3_55).

Compatibility with C54x devices (C54CM = 1)

C55x status bit mapping does not correspond to C54x status bits.

Status Bits

Affected by none

Affects Selected status bits

Repeat

This instruction cannot be repeated.

Example

Syntax	Description
bit(ST2, #ST2_AR1LC) = #0; AR1LC = bit 2	The ST2 bit position defined by the label (ST2_AR1LC, bit 2) is cleared to 0.

Before		After	
ST2	0006	ST2	0002

4.78.2 Status Bit Set: bit(STx, k4) = #1

Syntax Characteristics

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

† When instruction [8] 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.

Operands

k4, STx_55

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_55, ST1_55, ST2_55, or ST3_55).

Compatibility with C54x devices (C54CM = 1)

C55x status bit mapping does not correspond to C54x status bits.

Status Bits

Affected by none

Affects Selected status bits

Repeat

This instruction cannot be repeated.

Example

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
```

4.79 Store Extended Auxiliary Register to Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dbl(Lmem) = XAsrc	No	3	1	X

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.

Example

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

4.80 Subtraction

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = dst – src	Yes	2	1	X
[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	X
[18]	ACx = (Xmem << #16) – (Ymem << #16)	No	3	1	X

Brief Description

These instructions perform a subtraction operation:

- ❑ In the D-unit ALU, if the destination operand is an accumulator (ACx).
- ❑ In the A-unit ALU, if the destination operand is an auxiliary or temporary register (TAx).
- ❑ In the D-unit shifter, if the instruction has a shift quantity other than the immediate 16 bit shift.

Status Bits

Affected by CARRY, C54CM, M40, SATA, SATD, SXMD

Affects ACOVx, ACOVy, CARRY

4.80.1 Subtraction: dst = dst – src

Syntax Characteristics

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

Operands dst, src

Description

This instruction performs a subtraction operation between two registers contents.

- 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. 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.

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 subtracted from the content of AC0 and the result is stored in AC0.

4.80.2 Subtraction: dst = dst – k4

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	dst = dst – k4	Yes	2	1	X

Operands

dst, k4

Description

This instruction subtracts a 4-bit unsigned constant, k4, from a register content.

- 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.

Example

Syntax	Description
AC0 = AC0 – k4	An unsigned 4-bit value is subtracted from the content of AC0 and the result is stored in AC0.

4.80.3 Subtraction: dst = src – K16

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	dst = src – K16	No	4	1	X

Operands dst, K16, src

Description

This instruction subtracts a 16-bit signed constant, K16, 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 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.

Status Bits

Affected by M40, SATA, SATD, SXMD

Affects ACOVx, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 – K16	A signed 16-bit value is subtracted from the content of AC1 and the result is stored in AC0.

4.80.4 Subtraction: dst = src – Smem

Syntax Characteristics

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

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.

Status Bits

Affected by M40, SATA, SATD, SXMD

Affects ACOVx, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 – *AR3	The content addressed by AR3 is subtracted from the content of AC1 and the result is stored in AC0.

4.80.5 Subtraction: dst = Smem – src

Syntax Characteristics

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

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.

Status Bits

Affected by M40, SATA, SATD, SXMD

Affects ACOVx, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = *AR3 – AC1	The content of AC1 is subtracted from the content addressed by AR3 and the result is stored in AC0.

4.80.6 Subtraction: ACy = ACy – (ACx << Tx)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	ACy = ACy – (ACx << Tx)	Yes	2	1	X

Operands ACx, ACy, Tx

Description

This instruction subtracts an accumulator content ACx shifted by the content of Tx 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 identical 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
- 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.

Example

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.

4.80.7 Subtraction: ACy = ACy – (ACx << #SHIFTW)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	ACy = ACy – (ACx << #SHIFTW)	Yes	3	1	X

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 identical 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.

Example

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.

4.80.8 Subtraction: ACy = ACx – (K16 << #16)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	ACy = ACx – (K16 << #16)	No	4	1	X

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 identical 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.

Example

Syntax	Description
AC0 = AC1 – (K16 << #16)	A signed 16-bit value shifted left by 16 bits is subtracted from the content of AC1 and the result is stored in AC0.

4.80.9 Subtraction: ACy = ACx – (K16 << #SHFT)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[9]	ACy = ACx – (K16 << #SHFT)	No	4	1	X

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 identical 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.

Example

Syntax	Description
AC1 = AC0 – (#9800h << #5)	The signed 16-bit value (9800h) shifted left by 5 bits is subtracted from the content of AC0 and the result is stored in AC1.

4.80.10 Subtraction: ACy = ACx – (Smem << Tx)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[10]	ACy = ACx – (Smem << Tx)	No	3	1	X

Operands ACx, ACy, Tx, Smem

Description

This instruction subtracts the content of a memory (Smem) location shifted by the content of Tx 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 identical 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
- ❏ 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.

Example

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.

4.80.11 Subtraction: ACy = ACx – (Smem << #16)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[11]	ACy = ACx – (Smem << #16)	No	3	1	X

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 identical 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 C54CM, M40, SATD, SXMD

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

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.

4.80.12 Subtraction: ACy = (Smem << #16) – ACx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[12]	ACy = (Smem << #16) – ACx	No	3	1	X

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 identical 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.

Example

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.

4.80.13 Subtraction: ACy = ACx – uns(Smem) – BORROW

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[13]	ACy = ACx – uns(Smem) – BORROW	No	3	1	X

Operands ACx, ACy, Smem

Description

This instruction subtracts the logical complement of the CARRY status bit (borrow) and 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 sign extended to 40 bits according to SXMD.
  - ■ The content of the memory location is zero extended to 40 bits, if the optional uns keyword is applied to the input operand.
  - ■ The content of the memory location is sign extended to 40 bits according to SXMD, if the optional uns keyword is not applied to the input operand.
- ❏ 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 CARRY, M40, SATD, SXMD

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC1 = AC0 – uns(*AR1) – BORROW	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 0FFF
AR1	0302	AR1	0302
302	F000	302	F000
CARRY	0	CARRY	1

4.80.14 Subtraction: ACy = ACx – uns(Smem)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[14]	ACy = ACx – uns(Smem)	No	3	1	X

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 sign extended to 40 bits according to SXMD.
  - The content of the memory location is zero extended to 40 bits, if the optional uns keyword is applied to the input operand.
  - The content of the memory location is sign extended to 40 bits according to SXMD, if the optional uns keyword is not applied to the input operand.
- 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

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 – uns(*AR3)	The unsigned content addressed by AR3 is subtracted from the content of AC1 and the result is stored in AC0.

4.80.15 Subtraction: ACy = ACx – (uns(Smem) << #SHIFTW)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[15]	ACy = ACx – (uns(Smem) << #SHIFTW)	No	4	1	X

Operands ACx, ACy, SHIFTW, Smem

Description

This instruction subtracts the content of a memory (Smem) location shifted by the 6-bit value, SHIFTW, 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 content of the memory location is zero extended to 40 bits, if the optional uns keyword is applied to the input operand.
  - The content of the memory location is sign extended to 40 bits according to SXMD, if the optional uns keyword is not applied to the input operand.
- The shift operation is identical 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 cannot be repeated.

Example

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.

4.80.16 Subtraction: ACy = ACx – dbl(Lmem)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[16]	ACy = ACx – dbl(Lmem)	No	3	1	X

Operands ACx, ACy, Lmem

Description

This instruction subtracts the content of data memory operand dbl(Lmem) from an accumulator content ACx.

- 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
- 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 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

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 – dbl(*AR3+)	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.

4.80.17 Subtraction: ACy = dbl(Lmem) – ACx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[17]	ACy = dbl(Lmem) – ACx	No	3	1	X

Operands ACx, ACy, Lmem

Description

This instruction subtracts an accumulator content ACx from the content of data memory operand dbl(Lmem).

- 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
- 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 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

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
AC0 = dbl(*AR3) – AC1	The content of AC1 is subtracted from the content (long word) addressed by AR3 and AR3 + 1 and the result is stored in AC0.

4.80.18 Subtraction: ACx = (Xmem << #16) – (Ymem << #16)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[18]	ACx = (Xmem << #16) – (Ymem << #16)	No	3	1	X

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 identical 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.

Example

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.

Chapter 5

Instruction Opcodes in Sequential Order

This chapter provides the opcode in sequential order for each instruction syntax.

Topic **Page**

5.1 Instruction Set Opcodes

Table 5–1 lists the opcodes of the instruction set.

Table 5–1. Instruction Set Opcodes

Opcode	Algebraic syntax
0000000E XDDDX000	xdst = popboth()
0000000E XSSSX000	pshboth(xsrc)
0000000E xCCCCCCC kkkkkkkk	while (cond && (RPTC < k8)) repeat
0000001E xCCCCCCC xxxxxxxx	if (cond) return
0000010E xCCCCCCC LLLLLLLL	if (cond) goto L8
0000011E LLLLLLLL LLLLLLLL	goto L16
0000100E LLLLLLLL LLLLLLLL	call L16
0000101E GGGGGGGG Glllllll	
0000110E kkkkkkkk kkkkkkkk	repeat(k16)
0000111E llllllll llllllll	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 << #SHIFTW)
0001000E DDSS0101 xxSHIFTW	ACy = ACx << #SHIFTW
0001000E DDSS0110 xxSHIFTW	ACy = ACx <<C #SHIFTW
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 FDDD0utt	TCx = TCy & uns(src RELOP dst)
0001001E FSSScc01 FDDD1utt	TCx = !TCy & uns(src RELOP dst)
0001001E FSSScc10 FDDD0utt	TCx = TCy | uns(src RELOP dst)
0001001E FSSScc10 FDDD1utt	TCx = !TCy | uns(src RELOP dst)
0001001E FSSSxx11 FDDD0xvv	dst = BitOut \\ src \\ BitIn

Table 5–1. Instruction Set Opcodes (Continued)

Opcode	Algebraic syntax
0001001E FSSSxx11 FDDD1xvv	dst = BitIn // src // BitOut
0001010E FSSSxxxx FDDD0000	mar(TAy + TAx)
0001010E FSSSxxxx FDDD0001	mar(TAy = TAx)
0001010E FSSSxxxx FDDD0010	mar(TAy – TAx)
0001010E PPPPPPPP FDDD0100	mar(TAx + k8)
0001010E PPPPPPPP FDDD0101	mar(TAx = k8)
0001010E PPPPPPPP FDDD0110	mar(TAx – k8)
0001010E FSSSxxxx FDDD1000	mar(TAy + TAx)
0001010E FSSSxxxx FDDD1001	mar(TAy = TAx)
0001010E FSSSxxxx FDDD1010	mar(TAy – TAx)
0001010E PPPPPPPP FDDD1100	mar(TAx + k8)
0001010E PPPPPPPP FDDD1101	mar(TAx = k8)
0001010E PPPPPPPP FDDD1110	mar(TAx – k8)
0001011E xxxxxkkk kkkk0000	MDP = k7
0001011E xxxkkkkk kkkk0011	PDP = k9
0001011E kkkkkkkk kkkk0100	BK03 = k12
0001011E kkkkkkkk kkkk0101	BK47 = k12
0001011E kkkkkkkk kkkk0110	BKC = k12
0001011E kkkkkkkk kkkk1000	CSR = k12
0001011E kkkkkkkk kkkk1001	BRC0 = k12
0001011E kkkkkkkk kkkk1010	BRC1 = k12
0001011E xxxxxxxk kkkk11xx	
0001100E kkkkkkkk FDDDFSSS	dst = src & k8
0001101E kkkkkkkk FDDDFSSS	dst = src | k8
0001110E kkkkkkkk FDDDFSSS	dst = src ^ k8
0001111E KKKKKKKK SSDDxx0%	ACy = rnd(ACx * K8)
0001111E KKKKKKKK SSDDss1%	ACy = 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

Table 5–1. Instruction Set Opcodes (Continued)

Opcode	Algebraic syntax
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 (Note: FSSS = src1, FDDD = src2)	push(src1, src2)
0011101E FSSSFDDD (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
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
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
0100011E xxxx1000	
0100011E xxxx1001	
0100011E xxxx1010	

Table 5–1. Instruction Set Opcodes (Continued)

Opcode	Algebraic syntax
0100011E xxxx1100	
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
0100100E xxxxx101	return_int
0100101E 0LLLLLLL	goto L7
0100101E 1lllllll	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))
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
0101010E DDSS000%	ACy = rnd(ACy + |ACx|)
0101010E DDSS001%	ACy = rnd(ACy + (ACx * ACx))
0101010E DDSS010%	ACy = rnd(ACy – (ACx * ACx))
0101010E DDSS011%	ACy = rnd(ACy * ACx)
0101010E DDSS100%	ACy = rnd(ACx * ACx)
0101010E DDSS101%	ACy = rnd(ACx)
0101010E DDSS110%	ACy = saturate(rnd(ACx))
0101011E DDSSss0%	ACy = rnd(ACy + (ACx * Tx))
0101011E DDSSss1%	ACy = rnd(ACy – (ACx * Tx))

Table 5–1. Instruction Set Opcodes (Continued)

Opcode	Algebraic syntax
0101100E DDSSss0%	ACy = rnd(ACx * Tx)
0101100E DDSSss1%	ACy = 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 <<< Tx
0101110E DDSSss01	ACy = ACx << Tx
0101110E DDSSss10	ACy = ACx <<C Tx
0101111E 00kkkkkk	swap( )
01100lll lCCCCCCC	if (cond) goto l4
01101000 xCCCCCCC PPPPPPPP PPPPPPPP PPPPPPPP	if (cond) goto P24
01101001 xCCCCCCC PPPPPPPP PPPPPPPP PPPPPPPP	if (cond) call P24
01101010 PPPPPPPP PPPPPPPP PPPPPPPP	goto P24
01101100 PPPPPPPP PPPPPPPP PPPPPPPP	call P24
01101101 xCCCCCCC LLLLLLLL LLLLLLLL	if (cond) goto L16
01101110 xCCCCCCC LLLLLLLL LLLLLLLL	if (cond) call L16
01101111 FSSSccxu KKKKKKKK LLLLLLLL	compare (uns(src RELOP K8)) goto L8
01110000 KKKKKKKK KKKKKKKK SSDDSHFT	ACy = ACx + (K16 << #SHFT)
01110001 KKKKKKKK KKKKKKKK SSDDSHFT	ACy = ACx – (K16 << #SHFT)
01110010 kkkkkkkk kkkkkkkk SSDDSHFT	ACy = ACx & (k16 <<< #SHFT)
01110011 kkkkkkkk kkkkkkkk SSDDSHFT	ACy = ACx | (k16 <<< #SHFT)
01110100 kkkkkkkk kkkkkkkk SSDDSHFT	ACy = ACx ^ (k16 <<< #SHFT)
01110101 KKKKKKKK KKKKKKKK xxDDSHFT	ACx = K16 << #SHFT
01110110 kkkkkkkk kkkkkkkk FDDD00SS	dst = field_extract(ACx,k16)
01110110 kkkkkkkk kkkkkkkk FDDD01SS	dst = field_expand(ACx,k16)
01110110 KKKKKKKK KKKKKKKK FDDD10xx	dst = K16
01110111 DDDDDDDD DDDDDDDD FDDDxxxx	mar(TAx = D16)
01111000 kkkkkkkk kkkkkkkk xxx0000x	DP = k16
01111000 kkkkkkkk kkkkkkkk xxx0001x	SSP = k16
01111000 kkkkkkkk kkkkkkkk xxx0010x	CDP = k16
01111000 kkkkkkkk kkkkkkkk xxx0011x	BOF01 = k16
01111000 kkkkkkkk kkkkkkkk xxx0100x	BOF23 = k16

Table 5–1. Instruction Set Opcodes (Continued)

Opcode	Algebraic syntax
01111000 kkkkkkkk kkkkkkkk xxx0101x	BOF45 = k16
01111000 kkkkkkkk kkkkkkkk xxx0110x	BOF67 = k16
01111000 kkkkkkkk kkkkkkkk xxx0111x	BOFC = k16
01111000 kkkkkkkk kkkkkkkk xxx1000x	SP = k16
01111001 KKKKKKKK KKKKKKKK SSDDxx0%	ACy = rnd(ACx * K16)
01111001 KKKKKKKK KKKKKKKK SSDDss1%	ACy = rnd(ACx + (Tx * K16))
01111010 KKKKKKKK KKKKKKKK SSDD000x	ACy = ACx + (K16 << #16)
01111010 KKKKKKKK KKKKKKKK SSDD001x	ACy = ACx – (K16 << #16)
01111010 kkkkkkkk kkkkkkkk SSDD010x	ACy = ACx & (k16 <<< #16)
01111010 kkkkkkkk kkkkkkkk SSDD011x	ACy = ACx | (k16 <<< #16)
01111010 kkkkkkkk kkkkkkkk SSDD100x	ACy = ACx ^ (k16 <<< #16)
01111010 KKKKKKKK KKKKKKKK xxDD101x	ACx = K16 << #16
01111010 xxxxxxxx xxxxxxxx xxxx110x	idle
01111011 KKKKKKKK KKKKKKKK FDDDFSSS	dst = src + K16
01111100 KKKKKKKK KKKKKKKK FDDDFSSS	dst = src – K16
01111101 kkkkkkkk kkkkkkkk FDDDFSSS	dst = src & k16
01111110 kkkkkkkk kkkkkkkk FDDDFSSS	dst = src | k16
01111111 kkkkkkkk kkkkkkkk FDDDFSSS	dst = src ^ k16
10000000 XXXMMMYY YMMM00xx	dbl(Ymem) = dbl(Xmem)
10000000 XXXMMMYY YMMM01xx	Ymem = Xmem
10000000 XXXMMMYY YMMM10SS	Xmem = LO(ACx), Ymem = HI(ACx)
10000001 XXXMMMYY YMMM00DD	ACx = (Xmem << #16) + (Ymem << #16)
10000001 XXXMMMYY YMMM01DD	ACx = (Xmem << #16) – (Ymem << #16)
10000001 XXXMMMYY YMMM10DD	LO(ACx) = Xmem, HI(ACx) = Ymem
10000010 XXXMMMYY YMMM00mm uuDDDDg%	ACx = M40(rnd(uns(Xmem) * uns(Cmem))), ACy = M40(rnd(uns(Ymem) * uns(Cmem)))
10000010 XXXMMMYY YMMM01mm uuDDDDg%	ACx = M40(rnd(ACx + (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(uns(Ymem) * uns(Cmem)))
10000010 XXXMMMYY YMMM10mm uuDDDDg%	ACx = M40(rnd(ACx – (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(uns(Ymem) * uns(Cmem)))
10000010 XXXMMMYY YMMM11mm uuxxDDg%	mar(Xmem), ACx = M40(rnd(uns(Ymem) * uns(Cmem)))

Table 5–1. Instruction Set Opcodes (Continued)

Opcode	Algebraic syntax
10000011 XXXMMMYY YMMM00mm uuDDDDg%	ACx = M40(rnd(ACx + (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(ACy + (uns(Ymem) * uns(Cmem))))
10000011 XXXMMMYY YMMM01mm uuDDDDg%	ACx = M40(rnd(ACx – (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(ACy + (uns(Ymem) * uns(Cmem))))
10000011 XXXMMMYY YMMM10mm uuDDDDg%	ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(ACy + (uns(Ymem) * uns(Cmem))))
10000011 XXXMMMYY YMMM11mm uuxxDDg%	mar(Xmem), ACx = M40(rnd(ACx + (uns(Ymem) * uns(Cmem))))
10000100 XXXMMMYY YMMM00mm uuDDDDg%	ACx = M40(rnd(ACx – (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(Cmem))))
10000100 XXXMMMYY YMMM01mm uuxxDDg%	mar(Xmem), ACx = M40(rnd((ACx >> #16) + (uns(Ymem) * uns(Cmem))))
10000100 XXXMMMYY YMMM10mm uuDDDDg%	ACx = M40(rnd(uns(Xmem) * uns(Cmem))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(Cmem)))
10000100 XXXMMMYY YMMM11mm uuDDDDg%	ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(Cmem))))
10000101 XXXMMMYY YMMM00mm uuxxDDg%	mar(Xmem), ACx = M40(rnd(ACx – (uns(Ymem) * uns(Cmem))))
10000101 XXXMMMYY YMMM01mm uuDDDDg%	ACx = M40(rnd(ACx – (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(ACy – (uns(Ymem) * uns(Cmem))))
10000101 XXXMMMYY YMMM10mm xxxxxxxx	mar(Xmem) ,mar(Ymem) ,mar(Cmem)
10000101 XXXMMMYY YMMM11mm DDx0DDU%	firs(Xmem, Ymem, Cmem, ACx, ACy)
10000101 XXXMMMYY YMMM11mm DDx1DDU%	firsn(Xmem, Ymem, Cmem, ACx, ACy)
10000110 XXXMMMYY YMMMxxDD 000guuU%	ACx = M40(rnd(uns(Xmem) * uns(Ymem))) [,T3 = Xmem]
10000110 XXXMMMYY YMMMSSDD 001guuU%	ACy = M40(rnd(ACx + (uns(Xmem) * uns(Ymem)))) [,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)))) [,T3 = Xmem]
10000110 XXXMMMYY YMMMDDDD 100xssU%	ACx = rnd(ACx – (Tx * Xmem)), ACy = Ymem << #16 [,T3 = Xmem]
10000110 XXXMMMYY YMMMDDDD 101xssU%	ACx = rnd(ACx + (Tx * Xmem)), ACy = Ymem << #16 [,T3 = Xmem]
10000110 XXXMMMYY YMMMDDDD 110xxxx%	lms(Xmem, Ymem, ACx, ACy)
10000110 XXXMMMYY YMMMDDDD 1110xxn%	sqdst(Xmem, Ymem, ACx, ACy)
10000110 XXXMMMYY YMMMDDDD 1111xxn%	abdst(Xmem, Ymem, ACx, ACy)

Table 5–1. Instruction Set Opcodes (Continued)

Opcode	Algebraic syntax
10000111 XXXMMMYY YMMMSSDD 000xssU%	ACy = rnd(Tx * Xmem), Ymem = HI(ACx << T2) [,T3 = Xmem]
10000111 XXXMMMYY YMMMSSDD 001xssU%	ACy = rnd(ACy + (Tx * Xmem)), Ymem = HI(ACx << T2) [,T3 = Xmem]
10000111 XXXMMMYY YMMMSSDD 010xssU%	ACy = rnd(ACy – (Tx * Xmem)), Ymem = HI(ACx << T2) [,T3 = Xmem]
10000111 XXXMMMYY YMMMSSDD 100xxxxx	ACy = ACx + (Xmem << #16), Ymem = HI(ACy << T2)
10000111 XXXMMMYY YMMMSSDD 101xxxxx	ACy = (Xmem << #16) – ACx, Ymem = HI(ACy << T2)
10000111 XXXMMMYY YMMMSSDD 110xxxxx	ACy = Xmem << #16, Ymem = HI(ACx << T2)
10001xxx	
10010001 xxxxxxSS	goto ACx
10010010 xxxxxxSS	call ACx
10010100 xxxxxxxx	reset
10010101 0xxkkkkk	intr(k5)
10010101 1xxkkkkk	trap(k5)
10010110 0CCCCCCC	if (cond) execute(AD_unit)
10010110 1CCCCCCC	if (cond) execute(D_unit)
10010111	
10011000	mmap()
10011001	readport()
10011010	writeport()
10011100	linear()
10011101	circular()
10011110 0CCCCCCC	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)

Table 5–1. Instruction Set Opcodes (Continued)

Opcode	Algebraic syntax
10110111 AAAAAAAI	push(dbl(Lmem))
10111000 AAAAAAAI	dbl(Lmem) = pop()
10111011 AAAAAAAI	Smem = pop()
101111SS AAAAAAAI	Smem = HI(ACx)
1100FSSS AAAAAAAI	Smem = src
11010000 AAAAAAAI U%DDxxmm	ACx = rnd(ACx + (Smem * Cmem)) [,T3 = Smem], delay(Smem)
11010001 AAAAAAAI U%DD00mm	ACx = rnd(Smem * Cmem) [,T3 = Smem]
11010001 AAAAAAAI U%DD01mm	ACx = rnd(ACx + (Smem * Cmem)) [,T3 = Smem]
11010001 AAAAAAAI U%DD10mm	ACx = rnd(ACx – (Smem * 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%DDu1ss	ACx = rnd(uns(Tx * Smem)) [,T3 = Smem]
11010100 AAAAAAAI U%DDssSS	ACy = rnd(ACx + (Tx * Smem)) [,T3 = Smem]
11010101 AAAAAAAI U%DDssSS	ACy = rnd(ACx – (Tx * Smem)) [,T3 = Smem]
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	BOF01 = Smem
11011100 AAAAAAAI 0011xx10	BOF23 = Smem
11011100 AAAAAAAI 0100xx10	BOF45 = Smem
11011100 AAAAAAAI 0101xx10	BOF67 = Smem

Table 5–1. Instruction Set Opcodes (Continued)

Opcode	Algebraic syntax
11011100 AAAAAAAI 0110xx10	BOFC = Smem
11011100 AAAAAAAI 0111xx10	SP = Smem
11011100 AAAAAAAI 1000xx10	SSP = Smem
11011100 AAAAAAAI 1001xx10	BK03 = Smem
11011100 AAAAAAAI 1010xx10	BK47 = Smem
11011100 AAAAAAAI 1011xx10	BKC = Smem
11011100 AAAAAAAI 1100xx10	MDP = Smem
11011100 AAAAAAAI 1111xx10	PDP = Smem
11011100 AAAAAAAI x000xx11	CSR = Smem
11011100 AAAAAAAI x001xx11	BRC0 = Smem
11011100 AAAAAAAI x010xx11	BRC1 = Smem
11011100 AAAAAAAI x011xx11	TRN0 = Smem
11011100 AAAAAAAI 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 = rnd(Smem << Tx)
11011110 AAAAAAAI SSDD0000	ACy = adsc(Smem, ACx, TC1)
11011110 AAAAAAAI SSDD0001	ACy = adsc(Smem, ACx, TC2)
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, LO(ACx) = Smem – Tx
11011110 AAAAAAAI ssDD1001	HI(ACx) = Smem – Tx, 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)

Table 5–1. Instruction Set Opcodes (Continued)

Opcode	Algebraic syntax
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
11100101 AAAAAAAI 000110xx	Smem = CDP
11100101 AAAAAAAI 001010xx	Smem = BOF01
11100101 AAAAAAAI 001110xx	Smem = BOF23
11100101 AAAAAAAI 010010xx	Smem = BOF45
11100101 AAAAAAAI 010110xx	Smem = BOF67
11100101 AAAAAAAI 011010xx	Smem = BOFC
11100101 AAAAAAAI 011110xx	Smem = SP
11100101 AAAAAAAI 100010xx	Smem = SSP
11100101 AAAAAAAI 100110xx	Smem = BK03
11100101 AAAAAAAI 101010xx	Smem = BK47
11100101 AAAAAAAI 101110xx	Smem = BKC
11100101 AAAAAAAI 110010xx	Smem = MDP
11100101 AAAAAAAI 111110xx	Smem = PDP
11100101 AAAAAAAI x00011xx	Smem = CSR
11100101 AAAAAAAI x00111xx	Smem = BRC0

Table 5–1. Instruction Set Opcodes (Continued)

Opcode	Algebraic syntax
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, LO(Lmem) = LO(ACx) >> #1
11101011 AAAAAAAI xxSS1110	Lmem = pair(HI(ACx))
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)
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 = M40(dbl(Lmem))
11101101 AAAAAAAI xxDD101x	pair(HI(ACx)) = Lmem
11101101 AAAAAAAI xxDD110x	pair(LO(ACx)) = Lmem
11101101 AAAAAAAI FDDD111x	pair(TAx) = Lmem
11101110 AAAAAAAI SSDD000x	HI(ACy) = HI(Lmem) + HI(ACx), LO(ACy) = LO(Lmem) + LO(ACx)

Table 5–1. Instruction Set Opcodes (Continued)

Opcode	Algebraic syntax
11101110 AAAAAAAI SSDD001x	HI(ACy) = HI(ACx) – HI(Lmem), LO(ACy) = LO(ACx) – LO(Lmem)
11101110 AAAAAAAI SSDD010x	HI(ACy) = HI(Lmem) – HI(ACx), LO(ACy) = LO(Lmem) – LO(ACx)
11101110 AAAAAAAI ssDD011x	HI(ACx) = Tx – HI(Lmem), LO(ACx) = Tx – LO(Lmem)
11101110 AAAAAAAI ssDD100x	HI(ACx) = HI(Lmem) + Tx, LO(ACx) = LO(Lmem) + Tx
11101110 AAAAAAAI ssDD101x	HI(ACx) = HI(Lmem) – Tx, LO(ACx) = LO(Lmem) – Tx
11101110 AAAAAAAI ssDD110x	HI(ACx) = HI(Lmem) + Tx, LO(ACx) = LO(Lmem) – Tx
11101110 AAAAAAAI ssDD111x	HI(ACx) = HI(Lmem) – Tx, LO(ACx) = LO(Lmem) + Tx
11101111 AAAAAAAI xxxx00mm	Smem = Cmem
11101111 AAAAAAAI xxxx01mm	Cmem = Smem
11101111 AAAAAAAI xxxx10mm	Lmem = dbl(Cmem)
11101111 AAAAAAAI xxxx11mm	dbl(Cmem) = Lmem
11110000 AAAAAAAI KKKKKKKK KKKKKKKK	TC1 = (Smem == K16)
11110001 AAAAAAAI KKKKKKKK KKKKKKKK	TC2 = (Smem == K16)
11110010 AAAAAAAI kkkkkkkk kkkkkkkk	TC1 = Smem & k16
11110011 AAAAAAAI kkkkkkkk kkkkkkkk	TC2 = Smem & k16
11110100 AAAAAAAI kkkkkkkk kkkkkkkk	Smem = Smem & k16
11110101 AAAAAAAI kkkkkkkk kkkkkkkk	Smem = Smem | k16
11110110 AAAAAAAI kkkkkkkk kkkkkkkk	Smem = Smem ^ k16
11110111 AAAAAAAI KKKKKKKK KKKKKKKK	Smem = Smem + K16
11111000 AAAAAAAI KKKKKKKK xxDDx0U%	ACx = rnd(Smem * K8) [,T3 = Smem]
11111000 AAAAAAAI KKKKKKKK SSDDx1U%	ACy = rnd(ACx + (Smem * K8)) [,T3 = Smem]
11111001 AAAAAAAI uxSHIFTW SSDD00xx	ACy = ACx + (uns(Smem) << #SHIFTW)
11111001 AAAAAAAI uxSHIFTW SSDD01xx	ACy = ACx – (uns(Smem) << #SHIFTW)
11111001 AAAAAAAI uxSHIFTW xxDD10xx	ACx = uns(Smem) << #SHIFTW
11111010 AAAAAAAI xxSHIFTW SSxxx0x%	Smem = HI(rnd(ACx << #SHIFTW))
11111010 AAAAAAAI uxSHIFTW SSxxx1x%	Smem = HI(saturate(uns(rnd(ACx << #SHIFTW))))
11111011 AAAAAAAI KKKKKKKK KKKKKKKK	Smem = K16
11111100 AAAAAAAI LLLLLLLL LLLLLLLL	if (ARn_mod != #0) goto L16

5.2 Instruction Set Opcode Symbols and Abbreviations

Table 5–2 lists the symbols and abbreviations used in the instruction set opcode.

Table 5–2. Instruction Set Opcode Symbols and Abbreviations

Bit Field Name	Bit Field 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 *(ARn + T0), when C54CM = 1
	PPP0 1001	*(ARn – T0), when C54CM = 0 *(ARn – T0), when C54CM = 1
	PPP0 1011	*ARn(T0), when C54CM = 0 *ARn(T0), when C54CM = 1
	PPP0 1101	*ARn(#K16)
	PPP0 1111	*+ARn(#K16)
	PPP1 0011	*(ARn + T1), when ARMS = 0 *ARn(short(#1)), when ARMS = 1
	PPP1 0101	*(ARn – T1), when ARMS = 0 *ARn(short(#2)), when ARMS = 1

Table 5–2. Instruction Set Opcode Symbols and Abbreviations (Continued)

Bit Field Name	Bit Field Value	Bit Field Description
	PPP1 0111	*ARn(T1), when ARMS = 0 *ARn(short(#3)), when ARMS = 1
	PPP1 1001	*+ARn, when ARMS = 0 *ARn(short(#4)), when ARMS = 1
	PPP1 1011	*–ARn, when ARMS = 0 *ARn(short(#5)), when ARMS = 1
	PPP1 1101	*(ARn + T0B), when ARMS = 0 *ARn(short(#6)), when ARMS = 1
	PPP1 1111	*(ARn – T0B), when ARMS = 0 *ARn(short(#7)), when ARMS = 1
	PPP encodes 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 temporary 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(ACx) 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
	110 1000	TC1 & TC2

Table 5–2. Instruction Set Opcode Symbols and Abbreviations (Continued)

Bit Field Name	Bit Field Value	Bit Field Description
	110 1001	TC1 & !TC2
	110 1010	!TC1 & TC2
	110 1011	!TC1 & !TC2
	110 11xx	Reserved
	111 00SS	!overflow(ACx) 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 ss		Destination or 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)

Table 5–2. Instruction Set Opcode Symbols and Abbreviations (Continued)

Bit Field Name	Bit Field Value	Bit Field Description
DD SS		Destination or Source 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 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 5–2. Instruction Set Opcode Symbols and Abbreviations (Continued)

Bit Field Name	Bit Field 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 Register Content Swap 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 5–2. Instruction Set Opcode Symbols and Abbreviations (Continued)

Bit Field Name	Bit Field 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
KKK . . . K		Signed constant of n bits
lll . . . l		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		Coefficient 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 *(ARn + AR0), when C54CM = 1
	100	*(ARn + T1)
	101	*(ARn – T0), when C54CM = 0 *(ARn – AR0), when C54CM = 1

Table 5–2. Instruction Set Opcode Symbols and Abbreviations (Continued)

Bit Field Name	Bit Field Value	Bit Field Description
	110	*(ARn – T1)
	111	*ARn(T0), when C54CM = 0 *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
SHIFTW		6-bit immediate shift value, –32 to +31
SHFT		4-bit immediate shift value, 0 to 15
tt	00	Bit 0: destination TCy bit of Register Comparison instruction
	01	Bit 1: source TCx bit of Register Comparison 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

Table 5–2. Instruction Set Opcode Symbols and Abbreviations (Continued)

Bit Field Name	Bit Field Value	Bit Field Description
x		Reserved bit
XDDD XSSS		Destination or Source accumulator or extended register. All 23 bits of stack pointer (XSP), system stack pointer (XSSP), data page pointer (XDP), coefficient data pointer (XCDP), and extended auxiliary register (XARx).
XXX YYY		Auxiliary register designation for Xmem or Ymem addressing mode:
	000	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)

Chapter 6

Cross-Reference of Algebraic and Mnemonic Instruction Sets

This chapter provides a cross-reference between the algebraic instruction set and the mnemonic instruction set. For more information on the mnemonic instruction set, see *TMS320C55x DSP Mnemonic Instruction Set Reference Guide*, SPRU374.

Algebraic Syntax	Mnemonic Syntax
Absolute Distance	**Absolute Distance**
abdst(Xmem, Ymem, ACx, ACy)	ABDST Xmem, Ymem, ACx, ACy
Absolute Value	**Absolute Value**
dst = |src|	ABS [src,] dst
Accumulator, Auxiliary, or Temporary Register Content Swap	**Accumulator, Auxiliary, or Temporary Register Content Swap**
swap(ARx, Tx)	SWAP ARx, Tx
swap(Tx, Ty)	SWAP Tx, Ty
swap(ARx, ARy)	SWAP ARx, ARy
swap(ACx, ACy)	SWAP ACx, ACy
swap(pair(ARx), pair(Tx))	SWAPP ARx, Tx
swap(pair(T0), pair(T2))	SWAPP T0, T2
swap(pair(AR0), pair(AR2))	SWAPP AR0, AR2
swap(pair(AC0), pair(AC2))	SWAPP AC0, AC2
swap(block(AR4), block(T0))	SWAP4 AR4, T0

Algebraic Syntax	Mnemonic Syntax
Accumulator, Auxiliary, or Temporary Register Load	**Accumulator, Auxiliary, or Temporary Register Load**
dst = k4	MOV k4, dst
dst = –k4	MOV –k4, dst
dst = K16	MOV K16, dst
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
ACx = K16 << #16	MOV K16 << #16, ACx
ACx = K16 << #SHFT	MOV K16 << #SHFT, ACx
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 = M40(dbl(Lmem))	MOV[40] dbl(Lmem), ACx
LO(ACx) = Xmem, HI(ACx) = Ymem	MOV Xmem, Ymem, ACx
pair(HI(ACx)) = Lmem	MOV dbl(Lmem), pair(HI(ACx))
pair(LO(ACx)) = Lmem	MOV dbl(Lmem), pair(LO(ACx))
pair(TAx) = Lmem	MOV dbl(Lmem), pair(TAx)
Accumulator, Auxiliary, or Temporary Register Move	**Accumulator, Auxiliary, or Temporary Register Move**
dst = src	MOV src, dst
TAx = HI(ACx)	MOV HI(ACx), TAx
HI(ACx) = TAx	MOV TAx, HI(ACx)
Accumulator, Auxiliary, or Temporary Register Store	**Accumulator, Auxiliary, or Temporary Register Store**
Smem = src	MOV src, Smem
high_byte(Smem) = src	MOV src, high_byte(Smem)
low_byte(Smem) = src	MOV src, low_byte(Smem)
Smem = HI(ACx)	MOV HI(ACx), Smem

Algebraic Syntax	Mnemonic Syntax
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
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)
Lmem = pair(HI(ACx))	MOV pair(HI(ACx)), dbl(Lmem)
Lmem = pair(LO(ACx))	MOV pair(LO(ACx)), dbl(Lmem)
Lmem = pair(TAx)	MOV pair(TAx), dbl(Lmem)
HI(Lmem) = HI(ACx) >> #1, LO(Lmem) = LO(ACx) >> #1	MOV ACx >> #1, dual(Lmem)
Xmem = LO(ACx), Ymem = HI(ACx)	MOV ACx, Xmem, Ymem
Addition	**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 << Tx)	ADD ACx << Tx, ACy
ACy = ACy + (ACx << #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

Algebraic Syntax	Mnemonic Syntax
Smem = Smem + K16	ADD K16, Smem
ACy = rnd(ACy + |ACx|)	ADD[R]V [ACx,] ACy
Bit Field Comparison	**Bit Field Comparison**
TCx = Smem & k16	BAND Smem, k16, TCx
Bit Field Counting	**Bit Field Counting**
Tx = count(ACx, ACy, TCx)	BCNT ACx, ACy, TCx, Tx
Bit Field Expand	**Bit Field Expand**
dst = field_expand(ACx, k16)	BFXPA k16, ACx, dst
Bit Field Extract	**Bit Field Extract**
dst = field_extract(ACx, k16)	BFXTR k16, ACx, dst
Bitwise Complement	**Bitwise Complement**
dst = ~src	NOT [src,] dst
Bitwise 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
Smem = Smem & k16	AND k16, Smem
Bitwise 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 | (ACx <<< #SHIFTW)	OR ACx << #SHIFTW[, ACy]

Algebraic Syntax	Mnemonic Syntax
ACy = ACx | (k16 <<< #16)	OR k16 << #16, [ACx,] ACy
ACy = ACx | (k16 <<< #SHFT)	OR k16 << #SHFT, [ACx,] ACy
Smem = Smem | k16	OR k16, Smem
Bitwise XOR	**Bitwise XOR**
dst = dst ^ src	XOR src, dst
dst = src ^ k8	XOR k8, src, dst
dst = src ^ 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
Smem = Smem ^ k16	XOR k16, Smem
Branch Conditionally	**Branch Conditionally**
if (cond) goto l4	BCC l4, cond
if (cond) goto L8	BCC L8, cond
if (cond) goto L16	BCC L16, cond
if (cond) goto P24	BCC P24, cond
Branch Unconditionally	**Branch Unconditionally**
goto ACx	B ACx
goto L7	B L7
goto L16	B L16
goto P24	B P24
Branch on Auxiliary Register Not Zero	**Branch on Auxiliary Register Not Zero**
if (ARn_mod != #0) goto L16	BCC L16, ARn_mod != #0
Call Conditionally	**Call Conditionally**
if (cond) call L16	CALLCC L16, cond
if (cond) call P24	CALLCC P24, cond

Algebraic Syntax	Mnemonic Syntax
Call Unconditionally	**Call Unconditionally**
call ACx	CALL ACx
call L16	CALL L16
call P24	CALL P24
Compare and Branch	**Compare and Branch**
compare (uns(src RELOP K8)) goto L8	BCC[U] L8, src RELOP K8
Compare and Select Extremum	**Compare and Select Extremum**
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
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
Conditional Addition/Subtraction	**Conditional Addition/Subtraction**
ACy = adsc(Smem, ACx, TCx)	ADDSUBCC Smem, ACx, TCx, ACy
ACy = adsc(Smem, ACx, TC1, TC2)	ADDSUBCC Smem, ACx, TC1, TC2, ACy
ACy = ads2c(Smem, ACx, Tx, TC1, TC2)	ADDSUB2CC Smem, ACx, Tx, TC1, TC2, ACy
Conditional Shift	**Conditional Shift**
ACx = sftc(ACx, TCx)	SFTCC ACx, TCx
Conditional Subtract	**Conditional Subtract**
subc(Smem, ACx, ACy)	SUBC Smem, [ACx,] ACy
Dual 16-Bit Arithmetic	**Dual 16-Bit Arithmetic**
HI(ACx) = Smem + Tx, LO(ACx) = Smem – Tx	ADDSUB Tx, Smem, ACx
HI(ACx) = Smem – Tx, LO(ACx) = Smem + Tx	SUBADD Tx, Smem, ACx
HI(ACy) = HI(Lmem) + HI(ACx), LO(ACy) = LO(Lmem) + LO(ACx)	ADD dual(Lmem), [ACx,] ACy
HI(ACy) = HI(ACx) – HI(Lmem), LO(ACy) = LO(ACx) – LO(Lmem)	SUB dual(Lmem), [ACx,] ACy
HI(ACy) = HI(Lmem) – HI(ACx), LO(ACy) = LO(Lmem) – LO(ACx)	SUB ACx, dual(Lmem), ACy

Algebraic Syntax	Mnemonic Syntax
HI(ACx) = Tx – HI(Lmem), LO(ACx) = Tx – LO(Lmem)	SUB dual(Lmem), Tx, ACx
HI(ACx) = HI(Lmem) + Tx, LO(ACx) = LO(Lmem) + Tx	ADD dual(Lmem), Tx, ACx
HI(ACx) = HI(Lmem) – Tx, LO(ACx) = LO(Lmem) – Tx	SUB Tx, dual(Lmem), ACx
HI(ACx) = HI(Lmem) + Tx, LO(ACx) = LO(Lmem) – Tx	ADDSUB Tx, dual(Lmem), ACx
HI(ACx) = HI(Lmem) – Tx, LO(ACx) = LO(Lmem) + Tx	SUBADD Tx, dual(Lmem), ACx
Dual Multiply (Accumulate/Subtract)	**Dual Multiply (Accumulate/Subtract)**
ACx = M40(rnd(uns(Xmem) * uns(Cmem))), ACy = M40(rnd(uns(Ymem) * uns(Cmem)))	MPY[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx :: MPY[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy
ACx = M40(rnd(ACx + (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(uns(Ymem) * uns(Cmem)))	MAC[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx :: MPY[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy
ACx = M40(rnd(ACx – (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(uns(Ymem) * uns(Cmem)))	MAS[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx :: MPY[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy
mar(Xmem), ACx = M40(rnd(uns(Ymem) * uns(Cmem)))	AMAR Xmem :: MPY[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACx
ACx = M40(rnd(ACx + (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(ACy + (uns(Ymem) * uns(Cmem))))	MAC[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx :: MAC[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy
ACx = M40(rnd(ACx – (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(ACy + (uns(Ymem) * uns(Cmem))))	MAS[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx :: MAC[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy
mar(Xmem), ACx = M40(rnd(ACx + (uns(Ymem) * uns(Cmem))))	AMAR Xmem :: MAC[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACx
ACx = M40(rnd(ACx – (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(ACy – (uns(Ymem) * uns(Cmem))))	MAS[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx :: MAS[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy
mar(Xmem), ACx = M40(rnd(ACx – (uns(Ymem) * uns(Cmem))))	AMAR Xmem :: MAS[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACx
ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(ACy + (uns(Ymem) * uns(Cmem))))	MAC[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx >> #16 :: MAC[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy
ACx = M40(rnd(uns(Xmem) * uns(Cmem))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(Cmem))))	MPY[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx :: MAC[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy >> #16
ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(Cmem))))	MAC[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx >> #16 :: MAC[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy >> #16

Algebraic Syntax	Mnemonic Syntax
ACx = M40(rnd(ACx – (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(Cmem))))	MAS[R][40] [uns()Xmem[)], [uns()Cmem[)], ACx :: MAC[R][40] [uns()Ymem[)], [uns()Cmem[)], ACy >> #16
mar(Xmem), ACx = M40(rnd((ACx >> #16) + (uns(Ymem) * uns(Cmem))))	AMAR Xmem :: MAC[R][40] [uns()Ymem[)], [uns()Cmem[)], ACx >> #16
mar(Xmem), mar(Ymem), mar(Cmem)	AMAR Xmem, Ymem, Cmem
Execute Conditionally	**Execute Conditionally**
if (cond) execute(AD_Unit)	XCC [label,]cond
if (cond) execute(D_Unit)	XCCPART [label,]cond
Extended Auxiliary Register Move	**Extended Auxiliary Register Move**
xdst = xsrc	MOV xsrc, xdst
Finite Impulse Response Filter, Symmetrical/ Antisymmetrical	**Finite Impulse Response Filter, Symmetrical/ Antisymmetrical**
firs(Xmem, Ymem, Cmem, ACx, ACy)	FIRSADD Xmem, Ymem, Cmem, ACx, ACy
firsn(Xmem, Ymem, Cmem, ACx, ACy)	FIRSSUB Xmem, Ymem, Cmem, ACx, ACy
Idle	**Idle**
idle	IDLE
Implied Paralleled Instructions	**Implied Paralleled Instructions**
ACy = rnd(Tx * Xmem), Ymem = HI(ACx << T2) [,T3 = Xmem]	MPYM[R] [T3 =]Xmem, Tx, ACy :: MOV HI(ACx << T2), Ymem
ACy = rnd(ACy + (Tx * Xmem)), Ymem = HI(ACx << T2) [,T3 = Xmem]	MACM[R] [T3 =]Xmem, Tx, ACy :: MOV HI(ACx << T2), Ymem
ACy = rnd(ACy – (Tx * Xmem)), Ymem = HI(ACx << T2) [,T3 = Xmem]	MASM[R] [T3 =]Xmem, Tx, ACy :: MOV HI(ACx << T2), Ymem
ACy = ACx + (Xmem << #16), Ymem = HI(ACy << T2)	ADD Xmem << #16, ACx, ACy :: MOV HI(ACy << T2), Ymem
ACy = (Xmem << #16) – ACx, Ymem = HI(ACy << T2)	SUB Xmem << #16, ACx, ACy :: MOV HI(ACy << T2), Ymem
ACy = Xmem << #16, Ymem = HI(ACx << T2)	MOV Xmem << #16, ACy :: MOV HI(ACx << T2), Ymem
ACx = rnd(ACx + (Tx * Xmem)), ACy = Ymem << #16 [,T3 = Xmem]	MACM[R] [T3 =]Xmem, Tx, ACx :: MOV Ymem << #16, ACy

Algebraic Syntax	Mnemonic Syntax
ACx = rnd(ACx – (Tx * Xmem)), ACy = Ymem << #16 [,T3 = Xmem]	MASM[R] [T3 =]Xmem, Tx, ACx :: MOV Ymem << #16, ACy
Least Mean Square (LMS)	**Least Mean Square (LMS)**
lms(Xmem, Ymem, ACx, ACy)	LMS Xmem, Ymem, ACx, ACy
Linear/Circular Addressing Qualifiers	**Linear/Circular Addressing Qualifiers**
linear()	<instruction>.LR
circular()	<instruction>.CR
Load Effective Address to Extended Auxiliary Register	**Load Effective Address to Extended Auxiliary Register**
XAdst = mar(Smem)	AMAR Smem, XAdst
XAdst = k23	AMOV k23, XAdst
Load Extended Auxiliary Register from Memory	**Load Extended Auxiliary Register from Memory**
XAdst = dbl(Lmem)	MOV dbl(Lmem), XAdst
Logical Shift	**Logical Shift**
dst = dst <<< #1	SFTL dst, #1
dst = dst >>> #1	SFTL dst, #–1
ACy = ACx <<< Tx	SFTL ACx, Tx[, ACy]
ACy = ACx <<< #SHIFTW	SFTL ACx, #SHIFTW[, ACy]
Maximum Comparison	**Maximum Comparison**
dst = max(src, dst)	MAX [src,] dst
Minimum Comparison	**Minimum Comparison**
dst = min(src, dst)	MIN [src,] dst
Memory-Mapped Register Access Qualifier	**Memory-Mapped Register Access Qualifier**
mmap()	mmap

Algebraic Syntax	Mnemonic Syntax
Memory Bit Test/Set/Clear/Complement	**Memory Bit Test/Set/Clear/Complement**
TCx = bit(Smem, src)	BTST src, Smem, TCx
cbit(Smem, src)	BNOT src, Smem
bit(Smem, src) = #0	BCLR src, Smem
bit(Smem, src) = #1	BSET src, Smem
TCx = bit(Smem, k4), bit(Smem, k4) = #1	BTSTSET k4, Smem, TCx
TCx = bit(Smem, k4), bit(Smem, k4) = #0	BTSTCLR k4, Smem, TCx
TCx = bit(Smem, k4), cbit(Smem, k4)	BTSTNOT k4, Smem, TCx
TCx = bit(Smem, k4)	BTST k4, Smem, TCx
Memory Comparison	**Memory Comparison**
TCx = (Smem == K16)	CMP Smem == K16, TCx
Memory Delay	**Memory Delay**
delay(Smem)	DELAY Smem
Memory-to-Memory Move/Memory Initialization	**Memory-to-Memory Move/Memory Initialization**
Smem = Cmem	MOV Cmem, Smem
Cmem = Smem	MOV Smem, Cmem
Smem = K8	MOV K8, Smem
Smem = K16	MOV K16, Smem
Lmem = dbl(Cmem)	MOV Cmem, dbl(Lmem)
dbl(Cmem) = Lmem	MOV dbl(Lmem), Cmem
dbl(Ymem) = dbl(Xmem)	MOV dbl(Xmem), dbl(Ymem)
Ymem = Xmem	MOV Xmem, Ymem
Modify Auxiliary Register (MAR)	**Modify Auxiliary Register (MAR)**
mar(TAy + TAx)	AADD TAx, TAy
mar(TAy – TAx)	ASUB TAx, TAy
mar(TAy = TAx)	AMOV TAx, TAy
mar(TAx + k8)	AADD k8, TAx
mar(TAx – k8)	ASUB k8, TAx

Algebraic Syntax	Mnemonic Syntax
mar(TAx = k8)	AMOV k8, TAx
mar(TAx = D16)	AMOV D16, TAx
mar(Smem)	AMAR Smem
Modify Data Stack Pointer (SP)	**Modify Data Stack Pointer (SP)**
SP = SP + K8	AADD K8, SP
Multiply (MPY)	**Multiply (MPY)**
ACy = rnd(ACx * ACx)	SQR[R] [ACx,] ACy
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 = rnd(ACx * K16)	MPYK[R] K16, [ACx,] ACy
ACx = rnd(Smem * Cmem)[, T3 = Smem]	MPYM[R] [T3 =]Smem, Cmem, ACx
ACx = rnd(Smem * Smem)[, T3 = Smem]	SQRM[R] [T3 =]Smem, 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 and Accumulate (MAC)	**Multiply and Accumulate (MAC)**
ACy = rnd(ACy + (ACx * ACx))	SQA[R] [ACx,] ACy
ACy = rnd(ACy + (ACx * Tx))	MAC[R] ACx, Tx, ACy[, ACy]
ACy = rnd((ACy * Tx) + ACx)	MAC[R] ACy, Tx, ACx, ACy
ACy = rnd(ACx + (Tx * K8))	MACK[R] Tx, K8, [ACx,] ACy
ACy = rnd(ACx + (Tx * K16))	MACK[R] Tx, K16, [ACx,] ACy
ACx = rnd(ACx + (Smem * Cmem))[, T3 = Smem]	MACM[R] [T3 =]Smem, Cmem, ACx
ACx = rnd(ACx + (Smem * Cmem))[, T3 = Smem], delay(Smem)	MACM[R]Z [T3 =]Smem, Cmem, ACx
ACy = rnd(ACx + (Smem * Smem))[, T3 = Smem]	SQAM[R] [T3 =]Smem, [ACx,] ACy
ACy = rnd(ACy + (Smem * ACx))[, T3 = Smem]	MACM[R] [T3 =]Smem, [ACx,] ACy
ACy = 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

Algebraic Syntax	Mnemonic Syntax
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))))[, T3 = Xmem]	MACM[R][40] [T3 =][uns(]Xmem[)], [uns(]Ymem[)], ACx >> #16[, ACy]
Multiply and Subtract (MAS)	**Multiply and Subtract (MAS)**
ACy = rnd(ACy – (ACx * ACx))	SQS[R] [ACx,] ACy
ACy = rnd(ACy – (ACx * Tx))	MAS[R] Tx, [ACx,] ACy
ACx = rnd(ACx – (Smem * Cmem))[, T3 = Smem]	MASM[R] [T3 =]Smem, Cmem, ACx
ACy = rnd(ACx – (Smem * Smem))[, T3 = Smem]	SQSM[R] [T3 =]Smem, [ACx,] ACy
ACy = rnd(ACy – (Smem * ACx))[, T3 = Smem]	MASM[R] [T3 =]Smem, [ACx,] ACy
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
Negation	**Negation**
dst = –src	NEG [src,] dst
No Operation (NOP)	**No Operation (NOP)**
nop	NOP
nop_16	NOP_16
Normalization	**Normalization**
ACy = mant(ACx), Tx = –exp(ACx)	MANT ACx, ACy :: NEXP ACx, Tx
Tx = exp(ACx)	EXP ACx, Tx
Peripheral Port Register Access Qualifiers	**Peripheral Port Register Access Qualifier**
readport()	port(Smem)
writeport()	port(Smem)
Pop Extended Auxiliary Register from Stack Pointers	**Pop Extended Auxiliary Register from Stack Pointers**
xdst = popboth()	POPBOTH xdst

Algebraic Syntax	Mnemonic Syntax
Pop Top of Stack (TOS)	**Pop Top of Stack (TOS)**
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 Extended Auxiliary Register to Stack Pointers	**Push Extended Auxiliary Register to Stack Pointers**
pshboth(xsrc)	PSHBOTH xsrc
Push to Top of Stack (TOS)	**Push to Top of Stack (TOS)**
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)
Register Bit Test/Set/Clear/Complement	**Register Bit Test/Set/Clear/Complement**
TCx = bit(src, Baddr)	BTST Baddr, src, TCx
cbit(src, Baddr)	BNOT Baddr, src
bit(src, Baddr) = #0	BCLR Baddr, src
bit(src, Baddr) = #1	BSET Baddr, src
bit(src, pair(Baddr))	BTSTP Baddr, src
Register Comparison	**Register Comparison**
TCx = uns(src RELOP dst)	CMP[U] src RELOP dst, TCx
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
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

Algebraic Syntax	Mnemonic Syntax
Repeat Block of Instructions Unconditionally	**Repeat Block of Instructions Unconditionally**
localrepeat{}	RPTBLOCAL pmad
blockrepeat{}	RPTB pmad
Repeat Single Instruction Conditionally	**Repeat Single Instruction Conditionally**
while (cond && (RPTC < k8)) repeat	RPTCC k8, cond
Repeat Single Instruction Unconditionally	**Repeat Single Instruction Unconditionally**
repeat(CSR)	RPT CSR
repeat(CSR), CSR += TAx	RPTADD CSR, TAx
repeat(k8)	RPT k8
repeat(CSR), CSR += k4	RPTADD CSR, k4
repeat(CSR), CSR –= k4	RPTSUB CSR, k4
repeat(k16)	RPT k16
Return Conditionally	**Return Conditionally**
if (cond) return	RETCC cond
Return Unconditionally	**Return Unconditionally**
return	RET
Return from Interrupt	**Return from Interrupt**
return_int	RETI
Rotate Left	**Rotate Left**
dst = BitOut \\ src \\ BitIn	ROL BitOut, src, BitIn, dst
Rotate Right	**Rotate Right**
dst = BitIn // src // BitOut	ROR BitIn, src, BitOut, dst
Round	**Round**
ACy = rnd(ACx)	ROUND [ACx,] ACy

Algebraic Syntax	Mnemonic Syntax
Saturate	**Saturate**
ACy = saturate(rnd(ACx))	SAT[R] [ACx,] ACy
Signed Shift	**Signed Shift**
dst = dst >> #1	SFTS dst, #–1
dst = dst << #1	SFTS dst, #1
ACy = ACx << Tx	SFTS ACx, Tx[, ACy]
ACy = ACx <<C Tx	SFTSC ACx, Tx[, ACy]
ACy = ACx << #SHIFTW	SFTS ACx, #SHIFTW[, ACy]
ACy = ACx <<C #SHIFTW	SFTSC ACx, #SHIFTW[, ACy]
Software Interrupt	**Software Interrupt**
intr(k5)	INTR k5
Software Reset	**Software Reset**
reset	RESET
Software Trap	**Software Trap**
trap(k5)	TRAP k5
Specific CPU Register Load	**Specific CPU Register Load**
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
MDP = k7	MOV k7, DPH
PDP = k9	MOV k9, PDP
BOF01 = k16	MOV k16, BSA01
BOF23 = k16	MOV k16, BSA23
BOF45 = k16	MOV k16, BSA45
BOF67 = k16	MOV k16, BSA67
BOFC = k16	MOV k16, BSAC

Algebraic Syntax	Mnemonic Syntax
CDP = k16	MOV k16, CDP
DP = k16	MOV k16, DP
SP = k16	MOV k16, SP
SSP = k16	MOV k16, SSP
BK03 = Smem	MOV Smem, BK03
BK47 = Smem	MOV Smem, BK47
BKC = Smem	MOV Smem, BKC
BOF01 = Smem	MOV Smem, BSA01
BOF23 = Smem	MOV Smem, BSA23
BOF45 = Smem	MOV Smem, BSA45
BOF67 = Smem	MOV Smem, BSA67
BOFC = Smem	MOV Smem, BSAC
BRC0 = Smem	MOV Smem, BRC0
BRC1 = Smem	MOV Smem, BRC1
CDP = Smem	MOV Smem, CDP
CSR = Smem	MOV Smem, CSR
DP = Smem	MOV Smem, DP
MDP = 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
Specific CPU Register Move	**Specific CPU Register Move**
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
TAx = BRC0	MOV BRC0, TAx
TAx = BRC1	MOV BRC1, TAx

Algebraic Syntax	Mnemonic Syntax
TAx = CDP	MOV CDP, TAx
TAx = RPTC	MOV RPTC, TAx
TAx = SP	MOV SP, TAx
TAx = SSP	MOV SSP, TAx
Specific CPU Register Store	**Specific CPU Register Store**
Smem = BK03	MOV BK03, Smem
Smem = BK47	MOV BK47, Smem
Smem = BKC	MOV BKC, Smem
Smem = BOF01	MOV BSA01, Smem
Smem = BOF23	MOV BSA23, Smem
Smem = BOF45	MOV BSA45, Smem
Smem = BOF67	MOV BSA67, Smem
Smem = BOFC	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 = MDP	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)
Square Distance	**Square Distance**
sqdst(Xmem, Ymem, ACx, ACy)	SQDST Xmem, Ymem, ACx, ACy
Status Bit Set/Clear	**Status Bit Set/Clear**
bit(STx, k4) = #0	BCLR k4, STx_55
bit(STx, k4) = #1	BSET k4, STx_55

Algebraic Syntax	Mnemonic Syntax
Store Extended Auxiliary Register to Memory	**Store Extended Auxiliary Register to Memory**
dbl(Lmem) = XAsrc	MOV XAsrc, dbl(Lmem)
Subtraction	**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 << #SHIFTW)	SUB ACx << #SHIFTW, ACy
ACy = ACx – (K16 << #16)	SUB K16 << #16, [ACx,] ACy
ACy = ACx – (K16 << #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 – 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
ACx = (Xmem << #16) – (Ymem << #16)	SUB Xmem, Ymem, ACx

Index

E

F

G

I

L

M

N

O

P

R

S

T

U

W

X