

# 2 INSTRUCTION SET SUMMARY

This chapter provides a summary of the instructions in the ADSP-219x DSP's instruction set. Chapters 3 through 9 describe these instructions in more detail as follows:

- “ALU Instructions” on page 3-1
- “MAC Instructions” on page 4-1
- “Shifter Instructions” on page 5-1
- “Multifunction Instructions” on page 6-1
- “Data Move Instructions” on page 7-1
- “Program Flow Instructions” on page 8-1
- “Instruction Opcodes” on page 9-1

Also, this chapter identifies mnemonics for using DSP registers, bits, and operating conditions. This information appears in the following summaries:

- “Core Registers Summary” on page 2-3
- “Arithmetic Status (ASTAT) Register” on page 2-5
- “Condition Code (CCODE) Register” on page 2-6
- “Interrupt Control (ICNTL) Register” on page 2-8
- “Interrupt Mask (IMASK) & Latch (IRPTL) Registers” on page 2-10

## Instruction Set Summary

- [“Mode Status \(MSTAT\) Register”](#) on page 2-11
- [“Stack Status \(SSTAT\) Register”](#) on page 2-14
- [“Condition Codes Summary”](#) on page 2-15
- [“ALU Instructions”](#) on page 2-19
- [“Multiplier Instructions”](#) on page 2-20
- [“Shifter Instructions”](#) on page 2-21
- [“Multifunction Instructions”](#) on page 2-24
- [“Data Move Instructions”](#) on page 2-22
- [“Program Flow Instructions”](#) on page 2-23

For information on instruction reference notation, see [“Conventions”](#) on page 1-8.

## Core Registers Summary

The DSP has three categories of registers: core registers, system control registers, and I/O registers. [Table 2-1](#) lists and describes the DSP's core registers. For information about system control and I/O registers, see the *ADSP-219x/2191 DSP Hardware Reference*.

Table 2-1. Core registers

Type	Registers	Function
ALU data	AX0, AX1, AY0, AY1, AR, AF	16-bit data registers (X and Y) provide input for ALU, multiplier, and shifter operations.
Multiplier data	MX0, MX1, MY0, MY1, MR0, MR1, MR2	AR and AF are ALU result and feedback registers. MR and SR are multiplier results and feedback registers. SR also is the shifter results register.
Shifter data	SI, SE, SB, SR0, SR1, SR2	In this text, the word Dreg denotes unrestricted use of data registers as a data register file, while the words XOP and YOP denote restricted use.  The data registers (except AF, SE, and SB) serve as a register file, for unconditional, single-function instructions.
DAG address	I0, I1, I2, I3	DAG1 index registers
	I4, I5, I6, I7	DAG2 index registers
	M0, M1, M2, M3	DAG1 modify registers
	M4, M5, M6, M7	DAG2 modify registers
DAG address	L0, L1, L2, L3	DAG1 length registers
	L4, L5, L6, L7	DAG2 length registers
System control	B0, B1, B2, B3, B4, B5, B6, B7, SYSCCTL, CACTL	DAG1 base address registers (B0-3), DAG2 base address registers (B4-7), System control, Cache control

## Instruction Set Summary

Table 2-1. Core registers

Type	Registers	Function
Program flow	CCODE LPSTACKA LPSTACKP STACKA STACKP	Software condition register Loop PC stack A register, 16 address LSBs Loop PC stack P register, 8 address MSBs PC stack A register, 16 address LSBs PC stack P register, 8 address MSBs
Interrupt	ICNTL IMASK IRPTL	Interrupt control register Interrupt mask register Interrupt latch register
Status	ASTAT MSTAT SSTAT (read-only)	Arithmetic status flags Mode control and status flags System status
Page	DMPG1 DMPG2 IJPG IOPG	DAG1 page register, 8 address MSBs DAG2 page register, 8 address MSBs Indirect jump page register, 8 address MSBs I/O page register, 8 address MSBs
Bus exchange	PX	Holds eight LSBs of 24-bit memory data for transfers between memory and data registers only.
Shifter	SE SB	Shifter exponent register Shifter block exponent register

## Arithmetic Status (ASTAT) Register

The DSP updates the status bits in ASTAT, indicating the status of the most recent ALU, multiplier, or shifter operation.

Table 2-2. ASTAT register bit definitions

Bit	Name	Description
0	AZ	ALU result zero. Logical NOR of all bits written to the ALU result register (AR) or ALU feedback register (AF).  0 = ALU output $\neq$ 0 1 = ALU output = 0
1	AN	ALU result negative. Sign of the value written to the ALU result register (AR) or ALU feedback register (AF).  0 = ALU output positive (+) 1 = ALU output negative (-)
2	AV	ALU result overflow.  0 = No overflow 1 = Overflow
3	AC	ALU result carry.  0 = No carry 1 = Carry
4	AS	ALU $x$ input sign. Sign bit of the ALU $x$ -input operand; set by the ABS instruction only.  0 = Positive (+) 1 = Negative (-)

## Instruction Set Summary

Table 2-2. ASTAT register bit definitions (Cont'd)

Bit	Name	Description
5	AQ	ALU quotient. Sign of the resulting quotient; set by the DIVS or DIVQ instructions. 0 = Positive (+) 1 = Negative (-)
6	MV	Multiplier overflow. Records overflow/underflow condition for MR result register. 0 = No overflow or underflow 1 = Overflow or underflow
7	SS	Shifter input sign. Sign of the shifter input operand. 0 = Positive (+) 1 = Negative (-)
8	SV	Shifter overflow. Records overflow/underflow condition for SR result register. 0 = No overflow or underflow 1 = Overflow or underflow

## Condition Code (CCODE) Register

Using the CCODE register, conditional instructions may base execution on a comparison of the CCODE value (user-selected) and the SWCOND condition (DSP status). The CCODE register holds a value between 0x0 and 0xF, which the instruction tests against when the conditional instruc-

## Condition Code (CCODE) Register

tion uses SWCOND or NOT SWCOND. Note that the CCODE register has a one cycle effect latency.

Table 2-3. CCODE register bit definitions

CCODE Value	Software Condition	
	SWCOND (1010)	NOT SWCOND (1011)
0x00	PF0 pin high	PF0 pin low
0x01	PF1 pin high	PF1 pin low
0x02	PF2 pin high	PF2 pin low
0x03	PF3 pin high	PF3 pin low
0x04	PF4 pin high	PF4 pin low
0x05	PF5 pin high	PF5 pin low
0x06	PF6 pin high	PF6 pin low
0x07	PF7 pin high	PF7 pin low
0x08	AS	NOT AS
0x09	SV	NOT SV
0x0A	PF8 pin high	PF8 pin low
0x0B	PF9 pin high	PF9 pin low
0x0C	PF10 pin high	PF10 pin low
0x0D	PF11 pin high	PF11 pin low
0x0E	PF12 pin high	PF12 pin low
0x0F	PF13 pin high	PF13 pin low

## Instruction Set Summary

### Interrupt Control (ICNTL) Register

Table 2-4. ICNTL register bit definitions

Bit	Name	Description
0	reserved	write 0
1	reserved	write 0
2	reserved	write 0
3	reserved	write 0
4	INE	Interrupt nesting mode enable. 0 = Disabled 1 = Enabled
5	GIE	Global interrupt enable. 0 = Disabled 1 = Enabled
6	reserved	write 0
7	BIASRND	MAC biased rounding mode. 0 = Disabled 1 = Enabled
8-9	reserved	write 0
10	PCSTKE	PC stack interrupt enable. 0 = Disabled 1 = Enabled

## Interrupt Control (ICNTL) Register

Table 2-4. ICNTL register bit definitions (Cont'd)

Bit	Name	Description
11	EMUCNTE	Emulator cycle counter interrupt enable. 0 = Disabled 1 = Enabled
12-15	reserved	write 0

## Instruction Set Summary

### Interrupt Mask (IMASK) & Latch (IRPTL) Registers

Table 2-5. IMASK & IRPTL Registers Bit Definitions

Bit	Name	Description
0	EMU	Emulator. Nonmaskable. Highest priority
1	PWDN	Powerdown. Maskable only with GIE bit in ICNTL.
2	SSTEP	Single-step (during emulation)
3	STACK	Stack interrupt. Generated from any of the following stack status states: (if PCSTKE enabled) PC stack is pushed or popped and hits high-water mark, any stack overflows, or the status or PC stacks underflow.
4	User-defined	
5	User-defined	
6	User-defined	
7	User-defined	
8	User-defined	
9	User-defined	
10	User-defined	
11	User-defined	
12	User-defined	
13	User-defined	
14	User-defined	
15	User-defined	Lowest priority

## Mode Status (MSTAT) Register

Table 2-6. MSTAT register bit definitions

Bit	Name	Description
0	SEC_REG or SR	<p>Secondary data registers.</p> <p>Determines which set of data registers is currently active.</p> <p>0 = Deactivate secondary set of data registers (default). Primary register set (set that is active at reset) enabled and used for normal operation; secondary register set disabled.</p> <p>1 = Activate secondary set of data registers. Secondary register set enabled and used for alternate DSP context (for example, interrupt servicing); primary register set disabled, current contents preserved.</p> <p>For details, see <a href="#">“Switching Contexts” on page 8-16</a>.</p>
1	BIT_REV or BR	<p>Bit-reversed address output.</p> <p>Enables and disables bit-reversed addressing on DAG1 index registers only.</p> <p>0 = Disable</p> <p>1 = Enable</p> <p>For details, see <a href="#">“Bit-Reversed Addressing” on page 7-17</a>.</p>

## Instruction Set Summary

Table 2-6. MSTAT register bit definitions (Cont'd)

Bit	Name	Description															
2	AV_LATCH or OL	<p>ALU overflow latch mode. Determines how the ALU overflow flag, AV, gets cleared.</p> <p>0 = Disable</p> <p>Once an ALU overflow occurs and sets the AV bit in the ASTAT register, the AV bit remains set until explicitly cleared or is cleared by a subsequent ALU operation that does not generate an overflow.</p> <p>1 = Enable</p> <p>Once an ALU overflow occurs and sets the AV bit in the ASTAT register, the AV bit remains set until the application explicitly clears it. For details on clearing the AV bit, see <a href="#">“Bit Manipulation: TSTBIT, SETBIT, CLRBIT, TGLBIT”</a> on page 3-18 and <a href="#">“Register to Register Move”</a> on page 7-22.</p>															
3	AR_SAT or AS	<p>ALU saturation mode.</p> <p>For signed values, determines whether ALU AR results that overflowed or underflowed are saturated or not. Enables or disables saturation for all subsequent ALU operations.</p> <p>0 = Disable</p> <p>AR results remain unsaturated and return as is.</p> <p>1 = Enable</p> <p>AR results saturated according to the state of the AV and AC status flags in ASTAT.</p> <table border="0"> <thead> <tr> <th><u>AV</u></th> <th><u>AC</u></th> <th><u>AR register</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>ALU output</td> </tr> <tr> <td>0</td> <td>1</td> <td>ALU output</td> </tr> <tr> <td>1</td> <td>0</td> <td>0x7FFF</td> </tr> <tr> <td>1</td> <td>1</td> <td>0x8000</td> </tr> </tbody> </table> <p>Only the results written to the AR register are saturated. If results are written to the AF register, wraparound occurs, but the AV and AC flags reflect the saturated result.</p>	<u>AV</u>	<u>AC</u>	<u>AR register</u>	0	0	ALU output	0	1	ALU output	1	0	0x7FFF	1	1	0x8000
<u>AV</u>	<u>AC</u>	<u>AR register</u>															
0	0	ALU output															
0	1	ALU output															
1	0	0x7FFF															
1	1	0x8000															

## Mode Status (MSTAT) Register

Table 2-6. MSTAT register bit definitions (Cont'd)

Bit	Name	Description
4	M_MODE or MM	<p>MAC result mode.</p> <p>Determines the numeric format of multiplier operands. For all MAC operations, the multiplier adjusts the format of the result according to the selected mode.</p> <p>0 = Fractional mode, 1.15 format.</p> <p>1 = Integer mode, 16.0 format.</p> <p>For details, see <a href="#">“Data Format Options” on page 4-3</a>.</p>
5	TIMER or TI	<p>Timer enable.</p> <p>Starts and stops the timer counter.</p> <p>0 = Stops the timer count.</p> <p>1 = Starts the timer count.</p> <p>For details on timer operation, see the <i>ADSP-219x/2191 DSP Hardware Reference</i>.</p>
6	SEC_DAG or SD	<p>Secondary DAG registers.</p> <p>Determines which set of DAG address registers is currently active.</p> <p>0 = Primary registers.</p> <p>1 = Secondary registers.</p> <p>For details, see <a href="#">“Secondary DAG Registers” on page 7-8</a> and <a href="#">“Switching Contexts” on page 8-16</a>.</p>

## Instruction Set Summary

### Stack Status (SSTAT) Register

Table 2-7. SSTAT register bit definitions

Bit	Name	Description
0	PCSTKEMPTY or PCE	PC stack empty. 0 = PC stack contains at least one pushed address. 1 = PC stack is empty.
1	PCSTKFULL or PCF	PC stack full. 0 = PC stack contains at least one empty location. 1 = PC stack is full.
2	PCSTKLVL or PCL	PC stack level. 0 = PC stack contains between 3 and 28 pushed addresses. 1 = PC stack is at or above the high-water mark (28 pushed addresses), or it is at or below the low-water mark (3 pushed addresses).
3	Reserved	
4	LPSTKEMPTY or LSE	Loop stack empty. 0 = Loop stack contains at least one pushed address. 1 = Loop stack is empty.
5	LPSTKFULL or LSF	Loop stack full. 0 = Loop stack contains at least one empty location. 1 = Loop stack is full.

## Condition Codes Summary

Table 2-7. SSTAT register bit definitions (Cont'd)

Bit	Name	Description
6	STSSTKEMPTY or SSE	Status stack empty. 0 = Status stack contains at least one pushed status. 1 = Status stack is empty.
7	STKOVERFLOW or SOV	Stacks overflowed. 0 = Overflow/underflow has not occurred. 1 = At least one of the stacks (PC, loop, counter, status) has overflowed, or the PC or status stack has underflowed.  This bit cleared only on reset. Loop stack underflow is not detected because it occurs only as a result of a POP LOOP operation.

## Condition Codes Summary

Table 2-8. Condition codes summary

Code	Condition	Description
0000	EQ	Equal to zero (= 0).
0001	NE	Not equal to zero ( $\neq 0$ ).
0010	GT	Greater than zero ( $> 0$ ).
0011	LE	Less than or equal to zero ( $\leq 0$ ).
0100	LT	Less than zero ( $< 0$ ).
0101	GE	Greater than or equal to zero ( $\geq 0$ ).
0110	AV	ALU overflow.
0111	NOT AV	Not ALU overflow.

## Instruction Set Summary

Table 2-8. Condition codes summary (Cont'd)

Code	Condition	Description
1000	AC	ALU carry.
1001	NOT AC	Not ALU carry.
1010	SWCOND	SWCOND (based on CCODE register condition). (For CCODE details, see <a href="#">Table 2-3 on page 2-7.</a> )
1011	NOT SWCOND	Not SWCOND (based on CCODE register condition). (For CCODE details, see <a href="#">Table 2-3 on page 2-7.</a> )
1100	MV	MAC overflow.
1101	NOT MV	Not MAC overflow.
1110	NOT CE	Counter not expired.
1111	TRUE	Always true.

# Instruction Summary

The conventions for ADSP-219x instruction syntax descriptions appear in [Table 2-9](#). Other parts of the instruction syntax and opcode information also appear in this section. Following this table, the following sections provide summaries of the DSP’s instruction set:

- [“ALU Instructions” on page 2-19](#)
- [“Multiplier Instructions” on page 2-20](#)
- [“Shifter Instructions” on page 2-21](#)
- [“Data Move Instructions” on page 2-22](#)
- [“Program Flow Instructions” on page 2-23](#)
- [“Multifunction Instructions” on page 2-24](#)

For a list of instructions by types, see [“Instruction Opcodes” on page 9-1](#).

Table 2-9. Instruction Set Notation

Notation	Meaning
UPPERCASE	Explicit syntax— assembler keyword (notation only; assembler is case-insensitive and lowercase is the preferred programming convention)
;	Semicolon—instruction terminator
,	Comma—separates multiple optional items within vertical bars or separates parallel operations in multifunction instructions
option1, option2	Vertical bars—List of options separated with commas (choose one)
[optional]	Square brackets—enclose optional part of instruction
Compute	ALU, multiplier, shifter or multifunction operation

## Instruction Set Summary

Table 2-9. Instruction Set Notation (Cont'd)

Notation	Meaning
ALU, MAC, SHIFT	ALU, multiplier, or shifter operation
Cond	Status condition
Term	Loop termination condition
Reg	Any register from register groups Reg0, Reg1, Reg2, or Reg3
Dreg	Data register (register file) registers—subset of Reg0 registers
Ireg	Any DAG I register
Mreg	Any DAG M register
Lreg	Any DAG L register
Ia	I3-I0 (DAG1 index register)
Mb	M3-M0 (DAG1 modify register)
Ic	I7-I4 (DAG2 index register)
Md	M7-M4 (DAG2 modify register)
<Datan>	n-bit immediate data value
<ImmN>	n-bit immediate modify value
<Addrn>	n-bit immediate address value
<Reladdrn>	n-bit immediate PC-relative address value
Const	constant value; For valid constant values, see <a href="#">Table 3-1 on page 3-3</a> .
C	carry bit
(DB)	Delayed branch

## ALU Instructions

Table 2-10. Summary of ALU instructions

Instruction	Type	Details
$ AR, AF  = Dreg1 +  Dreg2, Dreg2 + C, C ;$	9, 9a	<a href="#">page 3-5</a>
[IF Cond] $ AR, AF  = Xop +  Yop, Yop + C, C, Const, Const + C ;$	9	<a href="#">page 3-5</a>
$ AR, AF  = Dreg1 -  Dreg2, Dreg2 + C - 1, +C - 1 ;$	9, 9a	<a href="#">page 3-8</a>
[IF Cond] $ AR, AF  = Xop -  Yop, Yop + C - 1, +C - 1, Const, Const + C - 1 ;$	9	<a href="#">page 3-8</a>
$ AR, AF  = Dreg2 -  Dreg1, Dreg1 + C - 1 ;$	9, 9a	<a href="#">page 3-12</a>
[IF Cond] $ AR, AF  = Yop -  Xop, Xop + C - 1 ;$	9	<a href="#">page 3-12</a>
[IF Cond] $ AR, AF  = -  Xop + C - 1, Xop + Const, Xop + Const + C - 1 ;$	9	<a href="#">page 3-12</a>
$ AR, AF  = Dreg1  AND, OR, XOR  Dreg2;$	9, 9a	<a href="#">page 3-15</a>
[IF Cond] $ AR, AF  = Xop  AND, OR, XOR   Yop, Const ;$	9	<a href="#">page 3-15</a>
[IF Cond] $ AR, AF  =  TSTBIT, SETBIT, CLRBIT, TGLBIT  n \text{ of } Xop;$	9, 9a	<a href="#">page 3-18</a>
$ AR, AF  = PASS  Dreg1, Dreg2, Const ;$	9, 9a	<a href="#">page 3-20</a>
$ AR, AF  = PASS 0;$	9, 9a	<a href="#">page 3-20</a>
[IF Cond] $ AR, AF  = PASS  Xop, Yop, Const ;$	9	<a href="#">page 3-20</a>
$ AR, AF  = NOT  Dreg ;$	9, 9a	<a href="#">page 3-23</a>
[IF Cond] $ AR, AF  = NOT  Xop, Yop ;$	9	<a href="#">page 3-23</a>
$ AR, AF  = ABS Dreg;$	9, 9a	<a href="#">page 3-26</a>
[IF Cond] $ AR, AF  = ABS Xop;$	9	<a href="#">page 3-26</a>

## Instruction Set Summary

Table 2-10. Summary of ALU instructions (Cont'd)

Instruction	Type	Details
$ AR, AF  = Dreg + 1;$	9, 9a	<a href="#">page 3-29</a>
[IF Cond] $ AR, AF  = Yop + 1;$	9	<a href="#">page 3-29</a>
$ AR, AF  = Dreg - 1;$	9, 9a	<a href="#">page 3-32</a>
[IF Cond] $ AR, AF  = Yop - 1;$	9	<a href="#">page 3-32</a>
DIVS $Yop, Xop;$	24	<a href="#">page 3-35</a>
DIVQ $Xop;$	23	<a href="#">page 3-35</a>
NONE = ALU ( $Xop, Yop$ );	8	<a href="#">page 3-44</a>

## Multiplier Instructions

Table 2-11. Summary of multiplier instructions

Instruction	Type	Details
$ MR, SR  = Dreg1 * Dreg2 [( RND, SS, SU, US, UU )];$	9a	<a href="#">page 4-8</a>
[IF Cond] $ MR, SR  = Xop * Yop [( RND, SS, SU, US, UU )];$	9	<a href="#">page 4-8</a>
[IF Cond] $ MR, SR  = Yop * Xop [( RND, SS, SU, US, UU )];$	9	<a href="#">page 4-8</a>
$ MR, SR  =  MR, SR  + Dreg1 * Dreg2 [( RND, SS, SU, US, UU )];$	9a	<a href="#">page 4-11</a>
[IF Cond] $ MR, SR  =  MR, SR  + Xop * Yop [( RND, SS, SU, US, UU )];$	9	<a href="#">page 4-11</a>
[IF Cond] $ MR, SR  =  MR, SR  + Yop * Xop [( RND, SS, SU, US, UU )];$	9	<a href="#">page 4-11</a>
$ MR, SR  =  MR, SR  - Dreg1 * Dreg2 [( RND, SS, SU, US, UU )];$	9a	<a href="#">page 4-14</a>
[IF Cond] $ MR, SR  =  MR, SR  - Xop * Yop [( RND, SS, SU, US, UU )];$	9	<a href="#">page 4-14</a>

Table 2-11. Summary of multiplier instructions (Cont'd)

Instruction	Type	Details
[IF Cond]  MR, SR  =  MR, SR  – Yop * Xop [( RND, SS, SU, US, UU )];	9	<a href="#">page 4-14</a>
[IF Cond]  MR, SR  = 0;	9	<a href="#">page 4-17</a>
[IF Cond] MR = MR [(RND)]; [IF Cond] SR = SR [(RND)];	9	<a href="#">page 4-19</a>
SAT MR; SAT SR;	25	<a href="#">page 4-21</a>

## Shifter Instructions

Table 2-12. Summary of shifter instructions

Instruction	Type	Details
[IF Cond] SR = [SR OR] ASHIFT Dreg [( HI, LO )];	16	<a href="#">page 5-6</a>
SR = [SR OR] ASHIFT BY <Imm8> [( HI, LO )];	15	<a href="#">page 5-8</a>
[IF Cond] SR = [SR OR] LSHIFT Dreg [( HI, LO )];	16	<a href="#">page 5-10</a>
SR = [SR OR] LSHIFT BY <Imm8> [( HI, LO )];	15	<a href="#">page 5-12</a>
[IF Cond] SR = [SR OR] NORM Dreg [( HI, LO )];	16	<a href="#">page 5-14</a>
[IF Cond] SE = EXP Dreg [( HIX, HI, LO )];	16	<a href="#">page 5-20</a>
[IF Cond] SB = EXPADJ Dreg;	16	<a href="#">page 5-23</a>

# Instruction Set Summary

## Data Move Instructions

Table 2-13. Summary of data move instructions

Instruction	Type	Details
Reg = Reg;	17	<a href="#">page 7-22</a>
DM(<Addr16>) =  Dreg, Ireg, Mreg ;	3	<a href="#">page 7-24</a>
Dreg, Ireg, Mreg  =  DM(<Addr16>) ;	3	<a href="#">page 7-24</a>
<Dreg>, <Reg1>, <Reg2>  = <Data16>;	6, 7, 7A	<a href="#">page 7-27</a>
Reg3 = <Data12>;	33	<a href="#">page 7-27</a>
DM(Ia += Mb), DM(Ic += Md)  = Reg;	32	<a href="#">page 7-30</a>
Reg =  DM(Ia += Mb), DM(Ic += Md) ;	32	<a href="#">page 7-30</a>
DM(Ia + Mb), DM(Ic + Md)  = Reg;	32	<a href="#">page 7-34</a>
Reg =  DM (Ia + Mb), DM (Ic + Md) ;	32	<a href="#">page 7-34</a>
PM(Ia += Mb), PM(Ic += Md)  = Reg;	32	<a href="#">page 7-37</a>
Reg =  PM(Ia += Mb), PM(Ic += Md) ;	32	<a href="#">page 7-37</a>
PM(Ia + Mb), PM(Ic + Md)  = Reg;	32	<a href="#">page 7-41</a>
Reg =  PM(Ia + Mb), PM(Ic + Md) ;	32	<a href="#">page 7-41</a>
DM(Ireg1 += Mreg1) =  Ireg2, Mreg2, Lreg2 ,  Ireg2, Mreg2, Lreg2  = Ireg1;	32A	<a href="#">page 7-45</a>
Dreg = DM(Ireg += <Imm8>);	29	<a href="#">page 7-49</a>
DM(Ireg += <Imm8>) = Dreg;	29	<a href="#">page 7-49</a>
Dreg = DM(Ireg + <Imm8>);	29	<a href="#">page 7-52</a>

## Program Flow Instructions

Table 2-13. Summary of data move instructions (Cont'd)

Instruction	Type	Details
DM(Ireg + <Imm8>) = Dreg;	29	<a href="#">page 7-52</a>
DM(Ia += Mb), DM (Ic += Md)  = <Data16>;	22	<a href="#">page 7-55</a>
PM (Ia += Mb), PM (Ic += Md)  = <Data24>;24;	22A	<a href="#">page 7-57</a>
IO(<Addr10>) = Dreg;	34	<a href="#">page 7-59</a>
Dreg = IO (<Addr10>);	34	<a href="#">page 7-59</a>
REG(<Addr8>) = Dreg;	35	<a href="#">page 7-61</a>
Dreg = REG(<Addr8>);	35	<a href="#">page 7-61</a>
MODIFY (Ia += Mb), MODIFY (Ic += Md) ;	21	<a href="#">page 7-63</a>
MODIFY (Ireg += <Imm8>);	21A	<a href="#">page 7-65</a>

## Program Flow Instructions

Table 2-14. Summary of program flow instructions

Instruction	Type	Details
DO <Reladdr12> UNTIL [CE, FOREVER];	11	<a href="#">page 8-22</a>
[IF Cond] JUMP <Reladdr13> [(DB)];	10	<a href="#">page 8-27</a>
CALL <Reladdr16> [(DB)];	10a	<a href="#">page 8-30</a>
JUMP <Reladdr16> [(DB)];	10a	<a href="#">page 8-34</a>
[IF Cond] CALL <Addr24>;	36	<a href="#">page 8-37</a>
[IF Cond] JUMP <Addr24>;	36	<a href="#">page 8-40</a>

## Instruction Set Summary

Table 2-14. Summary of program flow instructions (Cont'd)

Instruction	Type	Details
[IF Cond] CALL <Ireg> [(DB)];	19	<a href="#">page 8-42</a>
[IF Cond] JUMP <Ireg> [(DB)];	19	<a href="#">page 8-45</a>
[IF Cond] RTI [(DB)];	20	<a href="#">page 8-48</a>
[IF Cond] RTS [(DB)];	20	<a href="#">page 8-52</a>
PUSH  PC, LOOP, STS ;	26	<a href="#">page 8-55</a>
POP  PC, LOOP, STS ;	26	<a href="#">page 8-55</a>
FLUSH CACHE;	26	<a href="#">page 8-61</a>
SETINT <Imm4>;	37	<a href="#">page 8-62</a>
CLRINT <Imm4>;	37	<a href="#">page 8-64</a>
NOP;	30	<a href="#">page 8-66</a>
IDLE;	31	<a href="#">page 8-67</a>
ENA   TI, MM, AS, OL, BR, SR, BSR, INT   ;	18	<a href="#">page 8-69</a>
DIS   TI, MM, AS, OL, BR, SR, BSR, INT   ;	18	<a href="#">page 8-69</a>

## Multifunction Instructions

Table 2-15. Summary of multifunction instructions

Instruction	Type	Details
<ALU>, <MAC> , Xop = DM(Ia += Mb), Yop = PM(Ic += Md);	1	<a href="#">page 6-3</a>
Xop = DM(Ia += Mb), Yop = PM(Ic += Md);	1	<a href="#">page 6-7</a>

Table 2-15. Summary of multifunction instructions (Cont'd)

Instruction	Type	Details
<ALU>, <MAC>, <SHIFT>  , Dreg =  DM(Ia += Mb), PM(Ic += Md) ;	4, 12	<a href="#">page 6-10</a>
<ALU>, <MAC>, <SHIFT> ,  DM(Ia += Mb), PM(Ic += Md)  = Dreg;	4, 12	<a href="#">page 6-14</a>
<ALU>, <MAC>, <SHIFT> , Dreg = Dreg;	8, 14	<a href="#">page 6-18</a>

# Instruction Set Summary