MAXM MAXM

Transfer by Magnitude

Operation:

If |B| – |A| ≤ 0 then A → B

Assembler Syntax:

MAXM A,B (parallel move)

Description: Subtract the absolute value (magnitude) of the source accumulator from the absolute value of the destination accumulator. If the difference is negative or zero (i.e. |A| ≥ |B|) then transfer the source accumulator to destination accumulator, otherwise do not change destination accumulator.

This is a 56 bit operation.

Note: The Carry condition code signifies that a transfer has been performed.

Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
✔	✔	×	×	×	×	×	●
CCR							

- ● C Cleared if the conditional transfer was performed. Set otherwise.
- ✔ This bit is changed according to the standard definition
- × This bit is unchanged by the instruction

Instruction Formats and opcodes:

MAXM A, B

23 16	15 8	7 0
DATA BUS MOVE FIELD	0 0 0 1	0 1 0 1
OPTIONAL EFFECTIVE ADDRESS EXTENSION		

A-6.65 Merge Two Half Words (MERGE)

MERGE MERGE

Merge Two Half Words

Operation:

{S[11:0],D[35:24]} → D[47:24]

Assembler Syntax:

MERGE S,D

Description: The contents of bits 11-0 of the source register are concatenated to the contents of bits 35-24 of the destination accumulator. The result is stored in the destination accumulator. This instruction is a 24-bit operation. The remaining bits of the destination accumulator D are not affected.

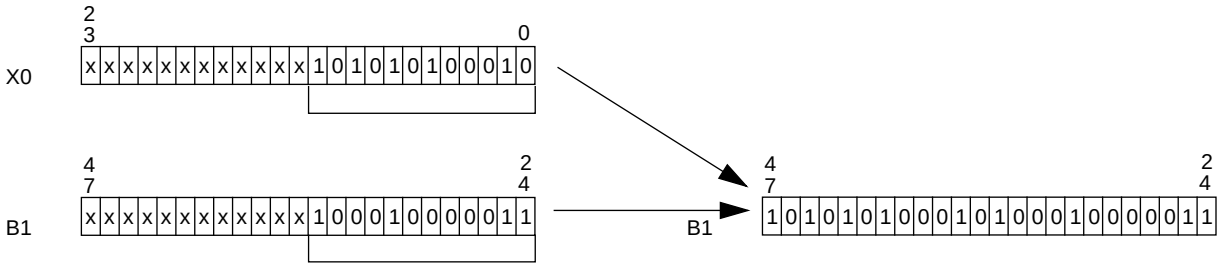
Notes:
1) This instruction may be used in conjunction with EXTRACT or INSERT instructions to concatenate width and offset fields into a control word.
2) In 16 bit arithmetic mode the contents of bits 15-8 of the source register are concatenated to the contents of bits 39-32 of the destination accumulator. The result is placed in bits 47-32 of the destination accumulator.

Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
X	X	X	X	●	●	●	X
CCR							

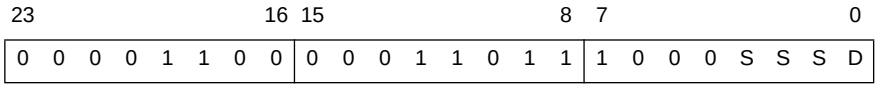
- N Set if bit 47 of the result is set
- Z Set if bits 47-24 of the result are zero
- V Always cleared

Example: MERGE X0,B



Instruction Formats and opcodes:

MERGE S,D



Instruction Fields:

{D} **D** Destination accumulator [A,B] (see Table A-10 on page A-239)

{S} **SSS** Source register [X0,X1,Y0,Y1,A1,B1] (see Table A-15 on page A-240)

MOVE MOVE

Move Data

Operation: S→D

Assembler Syntax: MOVE S,D

Description: Move the contents of the specified data source S to the specified destination D. This instruction is equivalent to a data ALU NOP with a parallel data move.

Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
✔	✔	×	×	×	×	×	×
CCR							

✔ This bit is changed according to the standard definition

× This bit is unchanged by the instruction

Instruction Formats and opcodes:

MOVE S,D

23 16	15 8	7 0
DATA BUS MOVE FIELD	0 0 0 0	0 0 0 0
OPTIONAL EFFECTIVE ADDRESS EXTENSION		

Instruction Fields:

See **Parallel Move Descriptions** for data bus move field encoding.

Parallel Move Descriptions: Thirty of the sixty-two instructions allow an optional parallel data bus movement over the X and/or Y data bus. This allows a data ALU operation to be executed in parallel with up to two data bus moves during the instruction cycle. Ten types of parallel moves are permitted, including register to register moves, register to memory moves, and memory to register moves. However, not all addressing modes are allowed for each type of memory reference. The following section contains detailed descriptions about each type of parallel move operation.

No Parallel Data Move

Operation:

(. . . . . .)

Assembler Syntax:

(. . . . . .)

where (. . . . .) refers to any arithmetic or logical instruction which allows parallel moves.

Description: Many instructions in the instruction set allow parallel moves. The parallel moves have been divided into 10 opcode categories. This category is a parallel move NOP and does not involve data bus move activity.

Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
x	x	x	x	x	x	x	x
CCR							

x This bit is unchanged by the instruction

Instruction Formats and opcodes:

(. . . . . .)

23							16	15							8	7	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	INSTRUCTION OPCODE	

Instruction Format:

(defined by instruction)

Immediate Short Data Move

Operation:

(. . . . .), #xx→D

Assembler Syntax:

(. . . . .) #xx,D

where (. . . . .) refers to any arithmetic or logical instruction which allows parallel moves.

Description: Move the 8-bit immediate data value (#xx) into the destination operand D.

If the destination register D is A0, A1, A2, B0, B1, B2, R0–R7, or N0–N7, the 8-bit immediate short operand is interpreted as an **unsigned integer** and is stored in the specified destination register. That is, the 8-bit data is stored in the eight LS bits of the destination operand, and the remaining bits of the destination operand D are zeroed.

If the destination register D is X0, X1, Y0, Y1, A, or B, the 8-bit immediate short operand is interpreted as a **signed fraction** and is stored in the specified destination register. That is, the 8-bit data is stored in the eight MS bits of the destination operand, and the remaining bits of the destination operand D are zeroed.

If the arithmetic or logical opcode-operand portion of the instruction specifies a given destination accumulator, that same accumulator or portion of that accumulator may not be specified as a destination D in the parallel data bus move operation. Thus, if the opcode-operand portion of the instruction specifies the 56-bit A accumulator as its destination, the parallel data bus move portion of the instruction may not specify A0, A1, A2, or A as its destination D. Similarly, if the opcode-operand portion of the instruction specifies the 56-bit B accumulator as its destination, the parallel data bus move portion of the instruction may not specify B0, B1, B2, or B as its destination D. That is, **duplicate destinations are NOT allowed within the same instruction.**

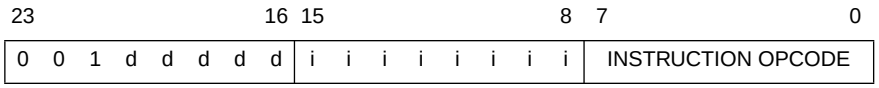
Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
×	×	×	×	×	×	×	×
CCR							

× This bit is unchanged by the instruction

Instruction Formats and opcodes:

(. . . . .) #xx,D



Instruction Fields:

{#xx}	**iiiiiiii**	8-bit Immediate Short Data
{D}	**ddddd**	Destination register [X0,X1,Y0,Y1,A0,B0,A2,B2,A1,B1,A,B,R0-R7,N0-N7] (see Table A-31 on page A-245)

R R

Register to Register Data Move

Operation:

(. . . . .); S→D

Assembler Syntax:

(. . . . .) S,D

where (. . . . .) refers to any arithmetic or logical instruction which allows parallel moves.

Description: Move the source register S to the destination register D.

If the arithmetic or logical opcode-operand portion of the instruction specifies a given destination accumulator, that same accumulator or portion of that accumulator may not be specified as a destination D in the parallel data bus move operation. Thus, if the opcode-operand portion of the instruction specifies the 56-bit A accumulator as its destination, the parallel data bus move portion of the instruction may not specify A0, A1, A2, or A as its destination D. Similarly, if the opcode-operand portion of the instruction specifies the 56-bit B accumulator as its destination, the parallel data bus move portion of the instruction may not specify B0, B1, B2, or B as its destination D. That is, **duplicate destinations are NOT allowed within the same instruction**.

If the opcode-operand portion of the instruction specifies a given source or destination register, that same register or portion of that register may be used as a source S in the parallel data bus move operation. This allows data to be moved in the same instruction in which it is being used as a source operand by a data ALU operation. That is, **duplicate sources are allowed within the same instruction**.

Note: The MOVE A,B operation will result in a 24-bit positive or negative saturation constant being stored in the B1 portion of the B accumulator if the signed integer portion of the A accumulator is in use.

Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
✔	✔	×	×	×	×	×	×
CCR							

✔ This bit is changed according to the standard definition

× This bit is unchanged by the instruction

Instruction Formats and opcodes:

(. . . . .) S,D

23							16	15							8	7	0
0	0	1	0	0	0	e	e	e	e	e	d	d	d	d	d	INSTRUCTION OPCODE	

Instruction Fields:

{S} **eeeee** Source register [X0,X1,Y0,Y1,A0,B0,A2,B2,A1,B1,A,B,R0-R7,N0-N7] (see Table A-31 on page A-245)

{D} **ddddd** Destination register [X0,X1,Y0,Y1,A0,B0,A2,B2,A1,B1,A,B,R0-R7,N0-N7] (see Table A-31 on page A-245)

U U

Address Register Update

Operation:

(. . . . .); ea→Rn

Assembler Syntax:

(. . . . .) ea

where (. . . . .) refers to any arithmetic or logical instruction which allows parallel moves.

Description: Update the specified address register according to the specified effective addressing mode. All update addressing modes may be used.

Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
×	×	×	×	×	×	×	×
CCR							

× This bit is unchanged by the instruction

Instruction Formats and opcodes:

(. . . . .) ea

23								16	15							8	7							0
0	0	1	0	0	0	0	0		0	1	0	M	M	R	R	R	INSTRUCTION OPCODE							

Instruction Fields:

{ea} **MMRRR** Effective Address (see Table A-20 on page A-242)

A-6.71 X Memory Data Move (X:)

X: X:

X Memory Data Move

Operation:	Assembler Syntax:
(. . . . .); X:ea→D	(. . . . .) X:ea,D
(. . . . .); X:aa→D	(. . . . .) X:aa,D
(. . . . .); S→X:ea	(. . . . .) S,X:ea
(. . . . .); S→X:aa	(. . . . .) S,X:aa
X:(Rn+xxx)→D	MOVE X:(Rn+xxx),D
X:(Rn+xxxx)→D	MOVE X:(Rn+xxxx),D
D→X:(Rn+xxx)	MOVE D,X:(Rn+xxx)
D→X:(Rn+xxxx)	MOVE D,X:(Rn+xxxx)

where (. . . . .) refers to any arithmetic or logical instruction which allows parallel moves.

Description: Move the specified word operand from/to X memory. All memory addressing modes, including absolute addressing and 24-bit immediate data, may be used. Absolute short addressing may also be used.

If the arithmetic or logical opcode-operand portion of the instruction specifies a given destination accumulator, that same accumulator or portion of that accumulator may not be specified as a destination D in the parallel data bus move operation. Thus, if the opcode-operand portion of the instruction specifies the 56-bit A accumulator as its destination, the parallel data bus move portion of the instruction may not specify A0, A1, A2, or A as its destination D. Similarly, if the opcode-operand portion of the instruction specifies the 56-bit B accumulator as its destination, the parallel data bus move portion of the instruction may not specify B0, B1, B2, or B as its destination D. That is, **duplicate destinations are NOT allowed within the same instruction**.

If the opcode-operand portion of the instruction specifies a given source or destination register, that same register or portion of that register may be used as a source S in the

parallel data bus move operation. This allows data to be moved in the same instruction in which it is being used as a source operand by a data ALU operation. That is, **duplicate sources are allowed within the same instruction**.

Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
✔	✔	×	×	×	×	×	×
CCR							

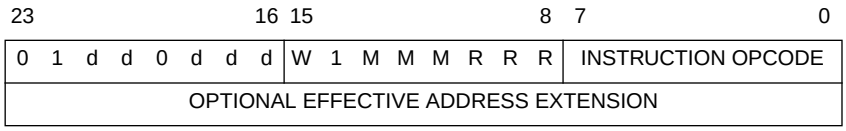
✔ This bit is changed according to the standard definition

× This bit is unchanged by the instruction

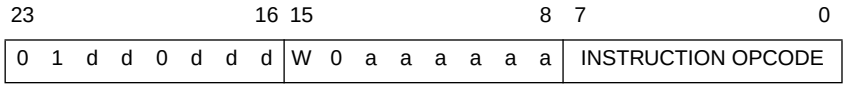
Note: The MOVE A,X:ea operation will result in a 24-bit positive or negative saturation constant being stored in the specified 24-bit X memory location if the signed integer portion of the A accumulator is in use.

Instruction Formats and opcodes 1:

(. . . . .)X:ea,D
(. . . . .)S,X:ea
(. . . . .)#xxxxxx,D



(. . . . .)X:aa,D
(. . . . .)S,X:aa

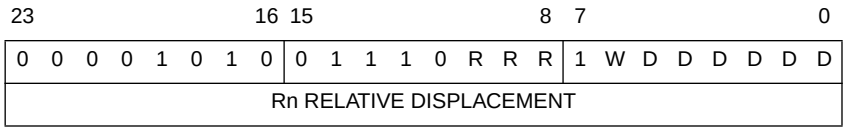


Instruction Fields:

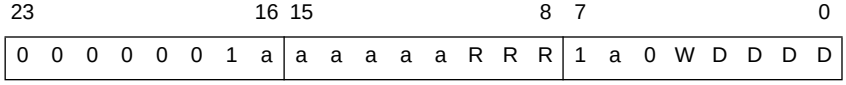
{ea}	**MMMRRR**	Effective Address (see Table A-16 on page A-241)
	W	Read S / Write D bit (see Table A-33 on page A-246)
{S,D}	**ddddd**	Source/Destination registers [X0,X1,Y0,Y1,A0,B0,A2,B2,A1,B1,A,B,R0-R7,N0-N7] (see Table A-31 on page A-245)
{aa}	**aaaaaa**	6-bit Absolute Short Address

Instruction Formats and opcodes 2:

MOVE X:(Rn+xxxx),D
MOVE S,X:(Rn+xxxx)



MOVE X:(Rn+xxx),D
MOVE S,X:(Rn+xxx)



Instruction Fields:

	W	Read S / Write D bit (see Table A-33 on page A-246)
{xxx}	**aaaaaaa**	7-bit sign extended Short Displacement Address
{Rn}	**RRR**	Address register (R0-R7)
{D}	**DDDD**	Source/Destination registers [X0,X1,Y0,Y1,A0,B0,A2,B2,A1,B1,A,B] (see Table A-34 on page A-246)
{S,D}	**DDDDDD**	Source/Destination registers [all on-chip registers] (see Table A-22 on page A-243)

X:R X:R

X Memory and Register Data Move

Operation:	Assembler Syntax:
Class I	
(. . . . .); X:ea→D1; S2→D2	(. . . . .) X:ea,D1 S2,D2
(. . . . .); S1→X:ea; S2→D2	(. . . . .) S1,X:ea S2,D2
(. . . . .); #xxxxxx→D1; S2→D2	(. . . . .) #xxxxxx,D1 S2,D2
Class II	
(. . . . .); A→X:ea; X0→A	(. . . . .) A,X:ea X0,A
(. . . . .); B→X:ea; X0→B	(. . . . .) B,X:ea X0,B

where (. . . . .) refers to any arithmetic or logical instruction which allows parallel moves.

Description: Class I: Move a one-word operand from/to X memory and move another word operand from an accumulator (S2) to an input register (D2). All memory addressing modes, including absolute addressing and 24-bit immediate data, may be used. The register to register move (S2,D2) allows a data ALU accumulator to be moved to a data ALU input register for use as a data ALU operand in the following instruction.

Class II: Move one-word operand from a data ALU accumulator to X memory and one-word operand from data ALU register X0 to a data ALU accumulator. One effective address is specified. All memory addressing modes, excluding long absolute addressing and long immediate data, may be used.

For both Class I and Class II X:R parallel data moves, if the arithmetic or logical opcode-operand portion of the instruction specifies a given destination accumulator, that same accumulator or portion of that accumulator may not be specified as a destination D1 in the parallel data bus move operation. Thus, if the opcode-operand portion of the instruction specifies the 56-bit A accumulator as its destination, the parallel data bus move portion of the instruction may not specify A0, A1, A2, or A as its destination D1. Similarly, if the opcode-operand portion of the instruction specifies the 56-bit B accumulator as its destination, the parallel data bus move portion of the instruction may not specify B0, B1,

B2, or B as its destination D1. That is, **duplicate destinations are NOT allowed within the same instruction**.

If the opcode-operand portion of the instruction specifies a given source or destination register, that same register or portion of that register may be used as a source S1 and/or S2 in the parallel data bus move operation. This allows data to be moved in the same instruction in which it is being used as a source operand by a data ALU operation. That is, **duplicate sources are allowed within the same instruction**. Note that S1 and S2 may specify the same register.

Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
✔	✔	×	×	×	×	×	×
CCR							

✔ This bit is changed according to the standard definition

× This bit is unchanged by the instruction

Class I Instruction Formats and opcodes:

(. . . . .) X:ea,D1 S2,D2
(. . . . .) S1,X:ea S2, D2
(. . . . .) #xxxxxx,D1 S2,D2

23								16	15							8	7							0
0	0	0	1	f	f	d	F		W	0	M	M	M	R	R	R	INSTRUCTION OPCODE							
OPTIONAL EFFECTIVE ADDRESS EXTENSION																								

Instruction Fields:

{ea}	**MMMRRR**	Effective Address (see Table A-15 on page A-240)
	W	Read S1 / Write D1 bit (see Table A-33 on page A-246)
{S1,D1}	**ff**	S1/D1 register [X0,X1,A,B] (see Table A-35 on page A-247)
{S2}	**d**	S2 accumulator [A,B] (see Table A-10 on page A-239)
{D2}	**F**	D2 input register [Y0,Y1] (see Table A-35 on page A-247)

Class II Instruction Formats and opcodes:

(. . . . .)A→X:ea X0→A
(. . . . .)B→X:ea X0→B

23								16	15							8	7							0
0	0	0	0	1	0	0	d		0	0	M	M	M	R	R	R	INSTRUCTION OPCODE							
OPTIONAL EFFECTIVE ADDRESS EXTENSION																								

Instruction Fields:

{ea}	**MMMRRR**	Effective Address (see Table A-19 on page A-242)
	d	Move opcode (see Table A-37 on page A-247)

Y: Y:

Y Memory Data Move

Operation:	Assembler Syntax:
(. . . . .); Y:ea→D	(. . . . .) Y:ea,D
(. . . . .); Y:aa→D	(. . . . .) Y:aa,D
(. . . . .); S→Y:ea	(. . . . .) S,Y:ea
(. . . . .); S→Y:aa	(. . . . .) S,Y:aa
Y:(Rn+xxx)→D	MOVE Y:(Rn+xxx),D
Y:(Rn+xxxx)→D	MOVE Y:(Rn+xxxx),D
D→Y:(Rn+xxx)	MOVE D,Y:(Rn+xxx)
D→Y:(Rn+xxxx)	MOVE D,Y:(Rn+xxxx)

where (. . . . .) refers to any arithmetic or logical instruction which allows parallel moves.

Description: Move the specified word operand from/to Y memory. All memory addressing modes, including absolute addressing and 24-bit immediate data, may be used. Absolute short addressing may also be used.

If the arithmetic or logical opcode-operand portion of the instruction specifies a given destination accumulator, that same accumulator or portion of that accumulator may not be specified as a destination D in the parallel data bus move operation. Thus, if the opcode-operand portion of the instruction specifies the 56-bit A accumulator as its destination, the parallel data bus move portion of the instruction may not specify A0, A1, A2, or A as its destination D. Similarly, if the opcode-operand portion of the instruction specifies the 56-bit B accumulator as its destination, the parallel data bus move portion of the instruction may not specify B0, B1, B2, or B as its destination D. That is, **duplicate destinations are NOT allowed within the same instruction**.

If the opcode-operand portion of the instruction specifies a given source or destination register, that same register or portion of that register may be used as a source S in the

parallel data bus move operation. This allows data to be moved in the same instruction in which it is being used as a source operand by a data ALU operation. That is, **duplicate sources are allowed within the same instruction**.

Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
✔	✔	×	×	×	×	×	×
CCR							

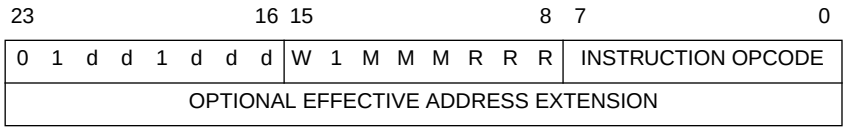
✔ This bit is changed according to the standard definition

× This bit is unchanged by the instruction

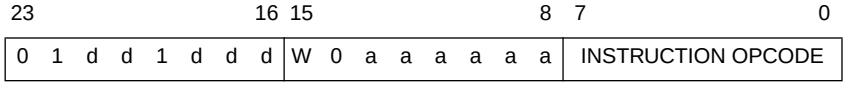
Note: The MOVE A,Y:ea operation will result in a 24-bit positive or negative saturation constant being stored in the specified 24-bit Y memory location if the signed integer portion of the A accumulator is in use.

Instruction Formats and opcodes 1:

(. . . . .)Y:ea,D
(. . . . .)S,Y:ea
(. . . . .)#xxxxxx,D



(. . . . .)Y:aa,D
(. . . . .)S,Y:aa

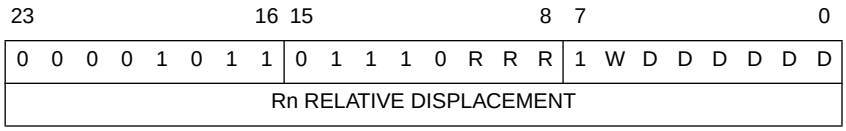


Instruction Fields:

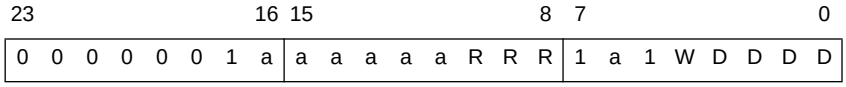
{ea}	**MMMRRR**	Effective Address (see Table A-15 on page A-240)
	W	Read S / Write D bit (see Table A-33 on page A-246)
{S,D}	**ddddd**	Source/Destination registers [X0,X1,Y0,Y1,A0,B0,A2,B2,A1,B1,A,B,R0-R7,N0-N7] (see Table A-31 on page A-245)
{aa}	**aaaaaa**	Absolute Short Address

Instruction Formats and opcodes 2:

MOVE Y:(Rn+xxxx),D
MOVE D,Y:(Rn+xxxx)



MOVE Y:(Rn+xxx),D
MOVE D,Y:(Rn+xxx)



Instruction Fields:

	W	Read S / Write D bit (see Table A-33 on page A-246)
{xxx}	**aaaaaaa**	7-bit sign extended Short Displacement Address
{Rn}	**RRR**	Address register (R0-R7)
{D}	**DDDD**	Source/Destination registers [X0,X1,Y0,Y1,A0,B0,A2,B2,A1,B1,A,B] (see Table A-34 on page A-246)
{S,D}	**DDDDDD**	Source/Destination registers [all on-chip registers] (see Table A-22 on page A-243)

A-6.74 Register and Y Memory Data Move (R:Y)

R:Y R:Y

Register and Y Memory Data Move

Operation:	Assembler Syntax:
Class I	
(. . . . .); S1→D1; Y:ea→D2	(. . . . .) S1,D1 Y:ea,D2
(. . . . .); S1→D1; S2→Y:ea	(. . . . .) S1,D1 S2,Y:ea
(. . . . .); S1→D1; #xxxxxx→D2	(. . . . .) S1,D1 #xxxxxx,D2
Class II	
(. . . . .); Y0 →A; A→Y:ea	(. . . . .) Y0,A A,Y:ea
(. . . . .); Y0→B; B→Y:ea	(. . . . .) Y0,B B,Y:ea

where (. . . . .) refers to any arithmetic or logical instruction which allows parallel moves.

Description: Class I: Move a one-word operand from an accumulator (S1) to an input register (D1) and move another word operand from/to Y memory. All memory addressing modes, including absolute addressing and 24-bit immediate data, may be used. The register to register move (S1,D1) allows a data ALU accumulator to be moved to a data ALU input register for use as a data ALU operand in the following instruction.

Class II: Move one-word operand from a data ALU accumulator to Y memory and one-word operand from data ALU register Y0 to a data ALU accumulator. One effective address is specified. All memory addressing modes, excluding long absolute addressing and long immediate data, may be used. Class II move operations have been added to the R:Y parallel move (and a similar feature has been added to the X:R parallel move) as an added feature available in the first quarter of 1989.

For both Class I and Class II R:Y parallel data moves, if the arithmetic or logical opcode-operand portion of the instruction specifies a given destination accumulator, that same accumulator or portion of that accumulator may not be specified as a destination D2 in the parallel data bus move operation. Thus, if the opcode-operand portion of the instruction specifies the 56-bit A accumulator as its destination, the parallel data bus move portion of the instruction may not specify A0, A1, A2, or A as its destination D2. Similarly, if the opcode-operand portion of the instruction specifies the 56-bit B accumulator as its

destination, the parallel data bus move portion of the instruction may not specify B0, B1, B2, or B as its destination D2. That is, duplicate destinations are NOT allowed within the same instruction.

If the opcode-operand portion of the instruction specifies a given source or destination register, that same register or portion of that register may be used as a source S1 and/or S2 in the parallel data bus move operation. This allows data to be moved in the same instruction in which it is being used as a source operand by a data ALU operation. That is, **duplicate sources are allowed within the same instruction**. Note that S1 and S2 may specify the same register.

Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
✔	✔	×	×	×	×	×	×
CCR							

✔ This bit is changed according to the standard definition

× This bit is unchanged by the instruction

<u>Class I</u> Instruction Formats and opcodes:

(. . . . .)S1,D1 Y:ea,D2
(. . . . .)S1,D1 S2,Y:ea
(. . . . .)S1,D1 #xxxxxx,D2

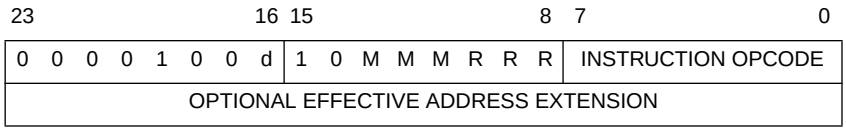
23								16	15							8	7							0
0	0	0	1	d	e	f	f		W	1	M	M	M	R	R	R	INSTRUCTION OPCODE							
OPTIONAL EFFECTIVE ADDRESS EXTENSION																								

Instruction Fields :

{ea}	**MMMRRR**	Effective Address (see Table A-15 on page A-240)
	W	Read S2 / Write D2 bit (see Table A-33 on page A-246)
{S1}	**d**	S1 accumulator [A,B] (see Table A-10 on page A-239)
{D1}	**e**	D1 input register [X0,X1] (see Table A-36 on page A-247)
{S2,D2}	**ff**	S2/D2 register [Y0,Y1,A,B] (see Table A-36 on page A-247)

Class II Instruction Formats and opcodes:

(. . . . .)Y0 → A A → Y:ea
(. . . . .)Y0 → B B → Y:ea



Instruction Fields:

MMMRRR ea=6-bit Effective Address (see Table A-19 on page A-242)
d Move opcode (see Table A-37 on page A-247)

L: L:

Long Memory Data Move

Operation:	Assembler Syntax:
(. . . . .); X:ea → D1; Y:ea → D2	(. . . . .) L:ea,D
(. . . . .); X:aa → D1; Y:aa → D2	(. . . . .) L:aa,D
(. . . . .); S1 → X:ea; S2 → Y:ea	(. . . . .) S,L:ea
(. . . . .); S1 → X:aa; S2 → Y:aa	(. . . . .) S,L:aa

where (. . . . .) refers to any arithmetic or logical instruction which allows parallel moves.

Description: Move one 48-bit long-word operand from/to X and Y memory. Two data ALU registers are concatenated to form the 48-bit long-word operand. This allows efficient moving of both double-precision (high:low) and complex (real:imaginary) data from/to one effective address in L (X:Y) memory. The same effective address is used for both the X and Y memory spaces; thus, only one effective address is required. Note that the A, B, A10, and B10 operands reference a single 48-bit signed (double-precision) quantity while the X, Y, AB, and BA operands reference two separate (i.e., real and imaginary) 24-bit signed quantities. All memory alterable addressing modes may be used. Absolute short addressing may also be used.

If the arithmetic or logical opcode-operand portion of the instruction specifies a given destination accumulator, that same accumulator or portion of that accumulator may not be specified as a destination D in the parallel data bus move operation. Thus, if the opcode-operand portion of the instruction specifies the 56-bit A accumulator as its destination, the parallel data bus move portion of the instruction may not specify A, A10, AB, or BA as destination D. Similarly, if the opcode-operand portion of the instruction specifies the 56-bit B accumulator as its destination, the parallel data bus move portion of the instruction may not specify B, B10, AB, or BA as its destination D. That is, duplicate destinations are NOT allowed within the same instruction.

If the opcode-operand portion of the instruction specifies a given source or destination register, that same register or portion of that register may be used as a source S in the parallel data bus move operation. This allows data to be moved in the same instruction in which it is being used as a source operand by a data ALU operation. That is, duplicate

sources are allowed within the same instruction.

Note: The operands A10, B10, X, Y, AB, and BA may be used only for a 48-bit long memory move as previously described. These operands may not be used in any other type of instruction or parallel move.

Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
✔	✔	×	×	×	×	×	×
CCR							

✔ This bit is changed according to the standard definition

× This bit is unchanged by the instruction

Note: The MOVE A,L:ea operation will result in a 48-bit positive or negative saturation constant being stored in the specified 24-bit X and Y memory locations if the signed integer portion of the A accumulator is in use. The MOVE AB,L:ea operation will result in either one or two 24-bit positive and/or negative saturation constant(s) being stored in the specified 24-bit X and/or Y memory location(s) if the signed integer portion of the A and/or B accumulator(s) is in use.

Instruction Formats and opcodes:

(. . . . .)L:ea,D
(. . . . .)S,L:ea

23							16	15							8	7							0
0	1	0	0	L	0	L	L	W	1	M	M	M	R	R	R	INSTRUCTION OPCODE							
OPTIONAL EFFECTIVE ADDRESS EXTENSION																							

(. . . . .)L:aa,D
(. . . . .)S,L:aa

23							16	15							8	7							0
0	1	0	0	L	0	L	L	W	0	a	a	a	a	a	a	INSTRUCTION OPCODE							

Instruction Fields:

{ea}	**MMMRRR**	Effective Address (see Table A-18 on page A-241)
	W	Read S / Write D bit (see Table A-33 on page A-246)
{L}	**LLL**	Two data ALU registers (see Table A-23 on page A-243)
{aa}	**aaaaaa**	Absolute Short Address

X: Y: X: Y:

XY Memory Data Move

Operation:	Assembler Syntax:
(. . . . .); X:<eax> → D1; Y:<eay> → D2	(. . . . .) X:<eax>,D1 Y:<eay>,D2
(. . . . .); X:<eax> → D1; S2 → Y:<eay>	(. . . . .) X:<eax>,D1 S2,Y:<eay>
(. . . . .); S1 → X:<eax>; Y:<eay> → D2	(. . . . .) S1,X:<eax> Y:<eay>,D2
(. . . . .); S1 → X:<eax>; S2 → Y:<eay>	(. . . . .) S1,X:<eax> S2,Y:<eay>

where (. . . . .) refers to any arithmetic or logical instruction which allows parallel moves.

Description: Move a one-word operand from/to X memory and move another word operand from/to Y memory. Note that two independent effective addresses are specified (<eax> and <eay>) where one of the effective addresses uses the lower bank of address registers (R0–R3) while the other effective address uses the upper bank of address registers (R4–R7). All parallel addressing modes may be used.

If the arithmetic or logical opcode-operand portion of the instruction specifies a given destination accumulator, that same accumulator or portion of that accumulator may not be specified as a destination D1 or D2 in the parallel data bus move operation. Thus, if the opcode-operand portion of the instruction specifies the 56-bit A accumulator as its destination, the parallel data bus move portion of the instruction may not specify A as its destination D1 or D2. Similarly, if the opcode-operand portion of the instruction specifies the 56-bit B accumulator as its destination, the parallel data bus move portion of the instruction may not specify B as its destination D1 or D2. That is, **duplicate destinations are NOT allowed within the same instruction**. D1 and D2 may not specify the same register.

If the instruction specifies an access to an internal X-I/O and internal Y-I/O modules (reflected by the address of the X memory space and of the Y memory space), than only the access to the internal X-I/O module will be executed. The access to the Y-I/O module will be discarded.

If the opcode-operand portion of the instruction specifies a given source or destination register, that same register or portion of that register may be used as a source S1 and/or

S2 in the parallel data bus move operation. This allows data to be moved in the same instruction in which it is being used as a source operand by a data ALU operation. That is, **duplicate sources are allowed within the same instruction**. Note that S1 and S2 may specify the same register.

Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
✔	✔	×	×	×	×	×	×
CCR							

✔ This bit is changed according to the standard definition

× This bit is unchanged by the instruction

Instruction Formats and opcodes:

(. . . . .)X:<eax>,D1 Y:<eay>,D2
(. . . . .)X:<eax>,D1 S2,Y:<eay>
(. . . . .)S1,X:<eax> Y:<eay>,D2
(. . . . .)S1,X:<eax> S2,Y:<eay>

Instruction Fields :

{<eax>} **MMRRR** 5-bit X Effective Address (R0–R3 or R4–R7)
{<eay>} **mmrr** 4-bit Y Effective Address (R4–R7 or R0–R3)
{S1,D1} **ee** S1/D1 register [X0,X1,A,B]
{S2,D2} **ff** S2/D2 register [Y0,Y1,A,B]
MMRRR,mmrr,ee,ff: see Table A-38 on page A-248
W X move Operation Control (See Table A-33 on page A-246)
w Y move Operation Control (See Table A-33 on page A-246)

MOVEC MOVEC

Move Control Register

Operation:	Assembler Syntax:
[X or Y]:ea→D1	MOVE(C) [Xor Y]:ea,D1
[X or Y]:aa→D1	MOVE(C) [Xor Y]:aa,D1
S1→[X or Y]:ea	MOVE(C) S1,[X or Y]:ea
S1→[X or Y]:aa	MOVE(C) S1,[X or Y]:aa
S1→D2	MOVE(C) S1,D2
S2→D1	MOVE(C) S2,D1
#xxxx→D1	MOVE(C) #xxxx,D1
#xx→D1	MOVE(C) #xx,D1

Description: Move the contents of the specified source **control register** S1 or S2 to the specified destination or move the specified source to the specified destination **control register** D1 or D2. The control registers S1 and D1 are a subset of the S2 and D2 register set and consist of the address ALU modifier registers and the program controller registers. These registers may be moved to or from any other register or memory space. All memory addressing modes, as well as an immediate short addressing mode, may be used.

If the system stack register SSH is specified as a source operand, the system stack pointer (SP) is postdecremented by 1 after SSH has been read. If the system stack register SSH is specified as a destination operand, the system stack pointer (SP) is preincremented by 1 before SSH is written. This allows the system stack to be efficiently extended using software stack pointer operations.

Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
●	●	●	●	●	●	●	●
CCR							

For D1 or D2=SR operand :

- S Set according to bit 7 of the source operand
- L Set according to bit 6 of the source operand
- E Set according to bit 5 of the source operand
- U Set according to bit 4 of the source operand
- N Set according to bit 3 of the source operand
- Z Set according to bit 2 of the source operand
- V Set according to bit 1 of the source operand
- C Set according to bit 0 of the source operand

For D1 and D2≠SR operand :

- S Set if data growth been detected
- L Set if data limiting has occurred during the move

Instruction Formats and opcodes:

MOVE(C) [X or Y]:ea,D1
MOVE(C) S1,[X or Y]:ea
MOVE(C) #xxxx,D1

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	1	W	1	M	M	M	R	R	R	O	S	1	d	d	d	d	d
OPTIONAL EFFECTIVE ADDRESS EXTENSION																							

MOVE(C) [X or Y]:aa,D1
MOVE(C) S1,[X or Y]:aa

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	1	W	0	a	a	a	a	a	a	0	S	1	d	d	d	d	d

MOVE(C) S1,D2
MOVE(C) S2,D1

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	0	W	1	e	e	e	e	e	e	1	0	1	d	d	d	d	d

MOVE(C) #xx,D1

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	1	i	i	i	i	i	i	i	i	1	0	1	d	d	d	d	d

Instruction Fields:

{ea}	**MMMRRR**	Effective Address (see Table A-15 on page A-240)
	W	Read S / Write D bit (see Table A-33 on page A-246)
{X/Y}	**S**	Memory Space [X,Y] (see Table A-17 on page A-241)
{S1,D1}	**ddddd**	Program Controller register [M0-M7,EP,VBA,SZ,SR,OMR,SP,SSH,SSL,LA,LC] (see Table A-41 on page A-249)
{aa}	**aaaaaa**	aa=6-bit Absolute Short Address
{S2,D2}	**eeeeee**	S2/D2 register [all on-chip registers] (see Table A-22 on page A-243)
{#xx}	**iiiiiiii**	#xx=8-bit Immediate Short Data

MOVEM MOVEM

Move Program Memory

Operation:	Assembler Syntax:	
S→P:ea	MOVE(M)	S,P:ea
S→P:aa	MOVE(M)	S,P:aa
P:ea→D	MOVE(M)	P:ea,D
P:aa→D	MOVE(M)	P:aa,D

Description: Move the specified operand from/to the specified **program** (P) **memory location**. This is a powerful move instruction in that the source and destination registers S and D may be **any** register. All memory alterable addressing modes may be used as well as the absolute short addressing mode.

If the system stack register SSH is specified as a source operand, the system stack pointer (SP) is postdecremented by 1 after SSH has been read. If the system stack register SSH is specified as a destination operand, the system stack pointer (SP) is preincremented by 1 before SSH is written. This allows the system stack to be efficiently extended using software stack pointer operations.

Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
●	●	●	●	●	●	●	●
CCR							

For D=SR operand :

- S Set according to bit 7 of the source operand
- L Set according to bit 6 of the source operand
- E Set according to bit 5 of the source operand
- U Set according to bit 4 of the source operand
- N Set according to bit 3 of the source operand
- Z Set according to bit 2 of the source operand
- V Set according to bit 1 of the source operand
- C Set according to bit 0 of the source operand

For D≠SR operand :

- S Set if data growth been selected
- L Set if data limiting has occurred during the move

Instruction Formats and opcodes:

MOVE(M) S,P:ea
MOVE(M) P:ea,D

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	1	1	W	1	M	M	M	R	R	R	1	0	d	d	d	d	d	d
OPTIONAL EFFECTIVE ADDRESS EXTENSION																							

MOVE(M) S,P:aa
MOVE(M) P:aa,D

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	1	1	W	0	a	a	a	a	a	a	0	0	d	d	d	d	d	d

Instruction Fields:

{ea}	**MMMRRR**	Effective Address (see Table A-18 on page A-241)
	W	Read S / Write D bit (see Table A-33 on page A-246)
{ S,D}	**dddddd**	Source/Destination register [all on-chip registers] (see Table A-22 on page A-243)
{aa}	**aaaaaa**	Absolute Short Address

A-6.79 Move Peripheral Data (MOVEP)

MOVEP MOVEP

Move Peripheral Data

Operation:	Assembler Syntax:	
[X or Y]:pp → D	MOVEP	[X or Y]:pp,D
[X or Y]:qq → D	MOVEP	[X or Y]:qq,D
[X or Y]:pp → [X or Y]:ea	MOVEP	[X or Y]:pp,[X or Y]:ea
[X or Y]:qq → [X or Y]:ea	MOVEP	[X or Y]:qq,[X or Y]:ea
[X or Y]:pp → P:ea	MOVEP	[X or Y]:pp,P:ea
[X or Y]:qq → P:ea	MOVEP	[X or Y]:qq,P:ea
S → [X or Y]:pp	MOVEP	S,[X or Y]:pp
S → [X or Y]:qq	MOVEP	S,[X or Y]:qq
[X or Y]:ea → [X or Y]:pp	MOVEP	[X or Y]:ea,[X or Y]:pp
[X or Y]:ea → [X or Y]:qq	MOVEP	[X or Y]:ea,[X or Y]:qq
P:ea → [X or Y]:pp	MOVEP	P:ea,[X or Y]:pp
P:ea → [X or Y]:qq	MOVEP	P:ea,[X or Y]:qq

Description: Move the specified operand from/to the specified **X or Y I/O peripheral**. The I/O short addressing mode is used for the I/O peripheral address. All memory addressing modes may be used for the X or Y memory effective address; all memory alterable addressing modes may be used for the P memory effective address. ALL the I/O space ($FFFF80-$FFFFFF) can be accessed, except for the P: reference opcode.

If the system stack register SSH is specified as a source operand, the system stack pointer (SP) is postdecremented by 1 after SSH has been read. If the system stack register SSH is specified as a destination operand, the system stack pointer (SP) is preincremented by 1 before SSH is written. This allows the system stack to be efficiently extended using software stack pointer operations.

Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
●	●	●	●	●	●	●	●
CCR							

For D=SR operand :

- S Set according to bit 7 of the source operand
- L Set according to bit 6 of the source operand
- E Set according to bit 5 of the source operand
- U Set according to bit 4 of the source operand
- N Set according to bit 3 of the source operand
- Z Set according to bit 2 of the source operand
- V Set according to bit 1 of the source operand
- C Set according to bit 0 of the source operand

For D≠SR operand :

- S Set if data growth been selected
- L Set if data limiting has occurred during the move

Instruction Formats and opcodes:

X: or Y: Reference (high I/O address)

MOVEP [X or Y]:pp,[X or Y]:ea
MOVEP [X or Y]:ea,[X or Y]:pp

23								16	15							8	7							0
0	0	0	0	1	0	0	s		W	1	M	M	M	R	R	R	1	S	p	p	p	p	p	p
OPTIONAL EFFECTIVE ADDRESS EXTENSION																								

X: or Y: Reference (low I/O address)

MOVEP X:qq,[X or Y]:ea
MOVEP [X or Y]:ea,X:qq

23								16	15							8	7							0
0	0	0	0	0	1	1	1		W	1	M	M	M	R	R	R	0	S	q	q	q	q	q	q
OPTIONAL EFFECTIVE ADDRESS EXTENSION																								

X: or Y: Reference (low I/O address)

MOVEP Y:qq,[X or Y]:ea
MOVEP [X or Y]:ea,Y:qq

23								16	15							8	7							0
0	0	0	0	0	1	1	1		W	0	M	M	M	R	R	R	1	S	q	q	q	q	q	q
OPTIONAL EFFECTIVE ADDRESS EXTENSION																								

P: Reference (high I/O address)

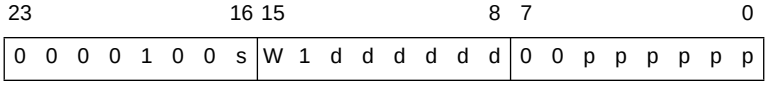
MOVEP P:ea,[X or Y]:pp
MOVEP [X or Y]:pp,P:ea

P: Reference (low I/O address)

MOVEP P:ea,[X or Y]:qq
MOVEP [X or Y]:qq,P:ea

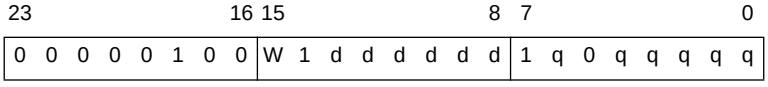
Register Reference (high I/O address)

MOVEP S,[X or Y]:pp
MOVEP [X or Y]:pp,D



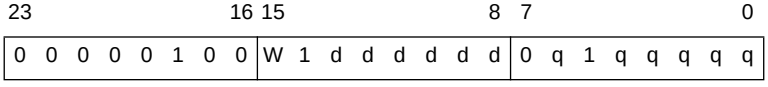
Register Reference: (low I/O address)

MOVEP S,X:qq
MOVEP X:qq,D



Register Reference: (low I/O address)

MOVEP S,Y:qq
MOVEP Y:qq,D



Instruction Fields:

{ea}	**MMMRRR**	Effective Address (see Table A-16 on page A-241)
{pp}	**pppppp**	I/O Short Address [64 addresses: $FFFFC0-$FFFFFF]
{qq}	**qqqqqq**	I/O Short Address [64 addresses: $FFFF80-$FFFFBF]
{X/Y}	**S**	Memory space [X,Y] (see Table A-17 on page A-241)
{X/Y}	**s**	Peripheral space [X,Y] (see Table A-17 on page A-241)
	W	Read/write-peripheral (see Table A-33 on page A-246)
{S,D}	**dddddd**	Source/Destination register [all on-chip registers] (see Table A-22 on page A-243)

MPY MPY

Signed Multiply

Operation:	**Assembler Syntax:**	
±S1∗S2→D (parallel move)	MPY	(±)S1,S2,D (parallel move)
±S1∗S2→D (parallel move)	MPY	(±)S2,S1,D (parallel move)
±(S1∗2^{-n})→D (**no** parallel move)	MPY	(±)S,#n,D (**no** parallel move)

Description: Multiply the two signed 24-bit source operands S1 and S2 and store the resulting product in the specified 56-bit destination accumulator D. Or, multiply the signed 24-bit source operand S by the positive 24-bit immediate operand 2^{-n} and store the resulting product in the specified 56-bit destination accumulator D. The "–" sign option is used to negate the specified product prior to accumulation. The default sign option is "+".

Note: When the processor is in the Double Precision Multiply Mode, the following instructions do not execute in the normal way and should only be used as part of the double precision multiply algorithm:

MPY Y0, X0, A MPY Y0, X0, B

Condition Codes:

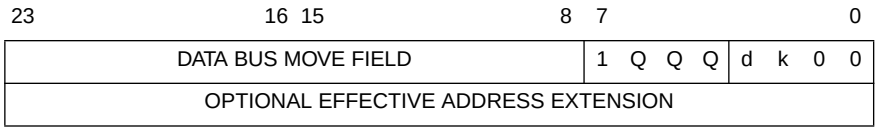
7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
✔	✔	✔	✔	✔	✔	✔	×
CCR							

✔ This bit is changed according to the standard definition

× This bit is unchanged by the instruction

Instruction Formats and opcodes 1:

MPY (±)S1,S2,D
MPY (±)S2,S1,D



Instruction Fields:

{S1,S2} **QQQ** Source registers S1,S2
[X0*X0,Y0*Y0,X1*X0,Y1*Y0,X0*Y1,Y0*X0,X1*Y0,Y1*X1]
(see Table A-26 on page A-244)

{D} **d** Destination accumulator [A,B] (see Table A-10 on page A-239)

{±+/-} **k** Sign [+,-] (see Table A-29 on page A-244)

Instruction Formats and opcode 2:

MPY (±)S,#n,D

Instruction Fields:

{S} **QQ** Source register [Y1,X0,Y0,X1]] (see Table A-27 on page A-244)

{D} **d** Destination accumulator [A,B] (see Table A-10 on page A-239)

{±} **k** Sign [+,-] (see Table A-29 on page A-244)

{#n} **sssss** Immediate operand (see Table A-32 on page A-246)

MPY(su,uu) MPY(su,uu)

Mixed Multiply

Operation:

±S1*S2→D (S1 unsigned, S2 unsigned)

±S1*S2→D (S1 signed, S2 unsigned)

Assembler Syntax:

MPYuu (±)S1,S2,D (no parallel move)

MPYsu (±)S2,S1,D (no parallel move)

Description: Multiply the two 24-bit source operands S1 and S2 and store the resulting product in the specified 56-bit destination accumulator D. One or two of the source operands can be unsigned. The "–" sign option is used to negate the specified product prior to accumulation. The default sign option is "+".

Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
✕	✔	✔	✔	✔	✔	✔	✕
CCR							

✔ This bit is changed according to the standard definition

✕ This bit is unchanged by the instruction

Instruction Formats and opcodes :

MPYsu (±)S1,S2,D
MPYuu (±)S1,S2,D

23							16	15							8	7							0
0	0	0	0	0	0	0	1	0	0	1	0	0	1	1	1	1	s	d	k	Q	Q	Q	Q

Instruction Fields:

{S1,S2}	**QQQQ**	Source registers S1,S2 [all combinations of X0,X1,Y0 and Y1] (see Table A-30 on page A-245)
{D}	**d**	Destination accumulator [A,B] (see Table A-10 on page A-239)
{±}	**k**	Sign [+,-] (see Table A-29 on page A-244)
{s}		[ss,us] (see Table A-40 on page A-249)

MPYI MPYI

Signed Multiply with Immediate Operand

Operation:

±#xxxxxx*S→D

Assembler Syntax:

MPYI (±)#xxxxxx,S,D

Description: Multiply the immediate 24-bit source operand #xxxxxx with the 24-bit register source operand S and store the resulting product in the specified 56-bit destination accumulator D. The "–" sign option is used to negate the specified product prior to accumulation. The default sign option is "+".

Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
×	✔	✔	✔	✔	✔	✔	×
CCR							

✔ This bit is changed according to the standard definition

× This bit is unchanged by the instruction

Instruction Formats and opcode:

MPYI (±)#xxxxxx,S,D

23							16	15							8	7							0
0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	1	1	1	q	q	d	k	0	0
IMMEDIATE DATA EXTENSION																							

Instruction Fields:

{S}	**qq**	Source register [X0,Y0,X1,Y1] (see Table A-28 on page A-244)
{D}	**d**	Destination accumulator [A,B] (see Table A-10 on page A-239)
{±}	**k**	Sign [+,-] (see Table A-29 on page A-244)
#xxxxxx		24-bit Immediate Long Data extension word

A-6.83 Signed Multiply and Round (MPYR)

MPYR MPYR

Signed Multiply and Round

Operation:	**Assembler Syntax:**	
±S1∗S2+r→D (parallel move)	MPYR	(±)S1,S2,D (parallel move)
±S1∗S2+r→D (parallel move)	MPYR	(±)S2,S1,D (parallel move)
±(S1∗2^{-n})+r→D (**no** parallel move)	MPYR	(±)S,#n,D (**no** parallel move)

Description: Multiply the two signed 24-bit source operands S1 and S2 (**or** the signed 24-bit source operand S by the positive 24-bit immediate operand 2^{-n}), round the result using either convergent or two's complement rounding, and store it in the specified 56-bit destination accumulator D.

The "–" sign option is used to negate the product prior to rounding. The default sign option is "+".

The contribution of the LS bits of the result is rounded into the upper portion of the destination accumulator. Once the rounding has been completed, the LS bits of the destination accumulator D are loaded with zeros to maintain an unbiased accumulator value which may be reused by the next instruction. The upper portion of the accumulator contains the rounded result which may be read out to the data buses. Refer to the RND instruction for more complete information on the rounding process.

Condition Codes:

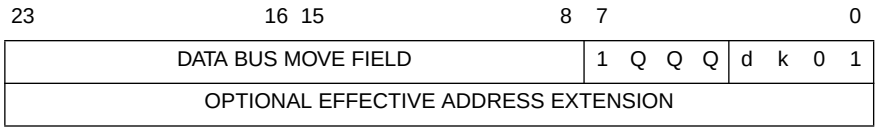
7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
✔	✔	✔	✔	✔	✔	✔	×
CCR							

✔ This bit is changed according to the standard definition

× This bit is unchanged by the instruction

Instruction Formats and opcodes 1:

MPYR (±)S1,S2,D
MPYR (±)S2,S1,D



Instruction Fields 1:

{S1,S2} **QQQ** Source registers S1,S2
[X0*X0,Y0*Y0,X1*X0,Y1*Y0,X0*Y1,Y0*X0,X1*Y0,Y1*X1]
(see Table A-26 on page A-244)

{D} **d** Destination accumulator [A,B] (see Table A-10 on page A-239)

{±} **k** Sign [+,-] (see Table A-29 on page A-244)

Instruction Formats and opcode 2:

MPYR (±)S,#n,D

Instruction Fields 2:

{S} **QQ** Source register [Y1,X0,Y0,X1]] (see Table A-27 on page A-244)

{D} **d** Destination accumulator [A,B] (see Table A-10 on page A-239)

{±} **k** Sign [+,-] (see Table A-29 on page A-244)

{#n} **sssss** Immediate operand (see Table A-32 on page A-246)

MPYRI MPYRI

Signed Multiply and Round with Immediate Operand

Operation:

±#xxxxxx*S+r →D

Assembler Syntax:

MPYRI (±)#xxxxxx,S,D

Description: Multiply the two signed 24-bit source operands #xxxxxx and S, round the result using either convergent or two's complement rounding, and store it in the specified 56-bit destination accumulator D.

The "–" sign option is used to negate the product prior to rounding. The default sign option is "+".

The contribution of the LS bits of the result is rounded into the upper portion of the destination accumulator. Once the rounding has been completed, the LS bits of the destination accumulator D are loaded with zeros to maintain an unbiased accumulator value which may be reused by the next instruction. The upper portion of the accumulator contains the rounded result which may be read out to the data buses. Refer to the RND instruction for more complete information on the rounding process.

Condition Codes:

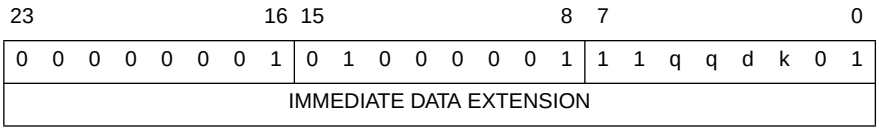
7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
✕	✔	✔	✔	✔	✔	✔	✕
CCR							

✔ This bit is changed according to the standard definition

✕ This bit is unchanged by the instruction

Instruction Formats and opcode:

MPYRI (±)#xxxxxx,S,D



Instruction Fields:

{S}	**qq**	Source register [X0,Y0,X1,Y1] (see Table A-28 on page A-244)
{D}	**d**	Destination accumulator [A,B] (see Table A-10 on page A-239)
{±}	**k**	Sign [+,-] (see Table A-29 on page A-244)
#xxxxxx		24-bit Immediate Long Data extension word

NEG NEG

Negate Accumulator

Operation:

0–D → D (parallel move)

Assembler Syntax:

NEG D (parallel move)

Description: Negate the destination operand D and store the result in the destination accumulator. This is a 56-bit, twos-complement operation.

Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
✔	✔	✔	✔	✔	✔	✔	×
CCR							

✔ This bit is changed according to the standard definition

× This bit is unchanged by the instruction

Instruction Formats and opcodes:

NEG D

23 16	15 8	7 0
DATA BUS MOVE FIELD	0 0 1 1	d 1 1 0
OPTIONAL EFFECTIVE ADDRESS EXTENSION		

Instruction Fields:

{D} **d** Destination accumulator [A,B] (see Table A-10 on page A-239)

NOP NOP

No Operation

Operation:

PC+1→PC

Assembler Syntax:

NOP

Description: Increment the program counter (PC). Pending pipeline actions, if any, are completed. Execution continues with the instruction following the NOP.

Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
×	×	×	×	×	×	×	×
CCR							

× This bit is unchanged by the instruction

Instruction Formats and opcode:

NOP

23							16	15							8	7							0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Instruction Fields : None

NORM NORM

Norm Accumulator Iteration

Operation:

If $\overline{E} \bullet U \bullet \overline{Z}=1$, then ASL D and Rn–1→Rn
else if E=1, then ASR D and Rn+1→R
else NOP

where $\overline{E}$ denotes the logical complement of E, and
where • denotes the logical AND operator

Assembler Syntax:

NORM Rn,D

Description: Perform one normalization iteration on the specified destination operand D, update the specified address register Rn based upon the results of that iteration, and store the result back in the destination accumulator. This is a 56-bit operation. If the accumulator extension is not in use, the accumulator is unnormalized, and the accumulator is not zero, the destination operand is arithmetically shifted one bit to the left, and the specified address register is decremented by 1. If the accumulator extension register is in use, the destination operand is arithmetically shifted one bit to the right, and the specified address register is incremented by 1. If the accumulator is normalized or zero, a NOP is executed and the specified address register is not affected. Since the operation of the NORM instruction depends on the E, U, and Z condition code register bits, these bits must correctly reflect the current state of the destination accumulator prior to executing the NORM instruction.

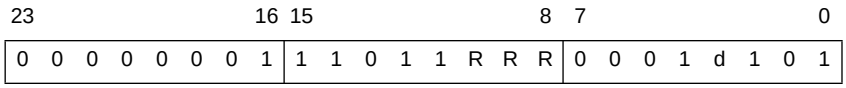
Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
✕	✔	✔	✔	✔	✔	●	✕
CCR							

- ● Set if bit 55 is changed as a result of a left shift
- ✔ This bit is changed according to the standard definition
- ✕ This bit is unchanged by the instruction

Instruction Formats and opcode:

NORM Rn,D



Instruction Fields:

{D} **d** Destination accumulator [A,B] (see Table A-10 on page A-239)
{Rn} **RRR** Address register [R0-R7]

NORMF NORMF

Fast Accumulator Normalization

Operation:

If S[23]=0 then ASR S,D
else ASL -S,D

Assembler Syntax:

NORMF S,D

Description: Arithmetically shift the destination accumulator either left or right as specified by the source operand sign and value. If the source operand is negative then the accumulator is left shifted, and if the source operand is positive then it is right shifted. The source accumulator value should be between +56 to -55 (or +40 to -39 in sixteen bit mode). This instruction can be used to normalize the specified accumulator D, by arithmetically shifting it either left or right so as to bring the leading one or zero to bit location 46. The number of needed shifts is specified by the source operand. This number could be calculated by a previous CLB instruction. For normalization the source accumulator value should be between +8 to -47 (or +8 to -31 in sixteen bit mode).

This is a 56 bit operation.

Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
✕	✔	✔	✔	✔	✔	●	✕
CCR							

● V Set if bit 55 is changed any time during the shift operation. Cleared otherwise.
✔ This bit is changed according to the standard definition
✕ This bit is unchanged by the instruction.

Example:

```
CLB     A,B     ;Count leading bits.

NORMF   B1,A    ;Normalize A.
```

If the base exponent is stored in R1 it can be updated by the following commands.

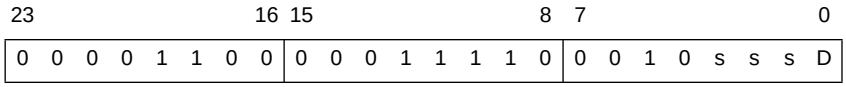
```
MOVE    B1,N1   ;Update N1 with shift amount

MOVE    (R1)+N1 ;Increment or decrement exponent
```

	Before execution	After execution
CLB A,B	A: $20:000000:000000	B: $00:000007:000000
NORMF B1,A	A: $20:000000:000000	A: $00:400000:000000

Explanation of example: Prior to execution, the 56-bit A accumulator contains the value $20:000000:000000. The CLB instruction updates the B accumulator to the number of needed shifts, 7 in this example. The NORMF instruction performs 7 shifts to the right on A accumulator, and normalization of A is achieved. The exponent register is updated according to the number of shifts.

Instruction Formats and opcode

NORMF S,D



Instruction Fields:

{S} **sss** Source register [X0,X1,Y0,Y1,A1,B1] (see Table A-15 on page A-240)
{D} **D** Destination accumulator [A,B] (see Table A-10 on page A-239)

A-6.89 Logical Complement (NOT)

NOT NOT

Logical Compliment

Operation:

$\overline{D[47:24]} \rightarrow D[47:24]$ (parallel move)

where "—" denotes the logical NOT operator

Assembler Syntax:

NOT D (parallel move)

Description: Take the ones complement of bits 47–24 of the destination operand D and store the result back in bits 47–24 of the destination accumulator. This is a 24-bit operation. The remaining bits of D are not affected.

Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
✔	✔	×	×	●	●	●	×
CCR							

- ● N Set if bit 47 of the result is set
- ● Z Set if bits 47–24 of the result are zero
- ● V Always cleared
- ✔ This bit is changed according to the standard definition
- × This bit is unchanged by the instruction

Instruction Formats and opcodes:

NOT D

23 — 16	15 — 8	7 — 0
DATA BUS MOVE FIELD	0 0 0 1	d 1 1 1
OPTIONAL EFFECTIVE ADDRESS EXTENSION		

Instruction Fields:

{D} **d** Destination accumulator [A,B] (see Table A-10 on page A-239)

OR OR

Logical Inclusive OR

Operation:

S+D[47:24] → D[47:24] (parallel move)

#xx+D[47:24] → D[47:24]

#xxxxxx+D[47:24] → D[47:24]

where + denotes the logical inclusive OR operator

Assembler Syntax:

OR S,D (parallel move)

OR #xx,D

OR #xxxxxx,D

Description: Logically inclusive OR the source operand S with bits 47–24 of the destination operand D and store the result in bits 47–24 of the destination accumulator. The source can be a 24-bit register, 6-bit short immediate or 24-bit long immediate. This instruction is a 24-bit operation. The remaining bits of the destination operand D are not affected.

When using 6-bit immediate data, the data is interpreted as an unsigned integer. That is, the 6 bits will be right aligned and the remaining bits will be zeroed to form a 24-bit source operand.

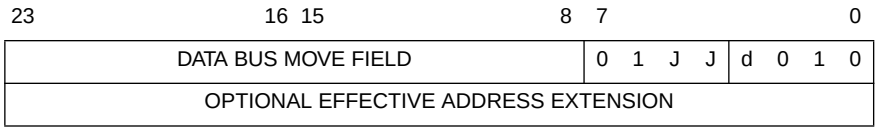
Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
✔	✔	×	×	●	●	●	×
CCR							

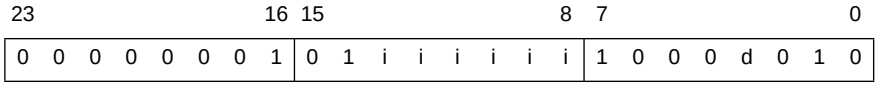
- ● N Set if bit 47 of the result is set
- ● Z Set if bits 47–24 of the result are zero
- ● V Always cleared
- ✔ This bit is changed according to the standard definition
- × This bit is unchanged by the instruction

Instruction Formats and opcodes:

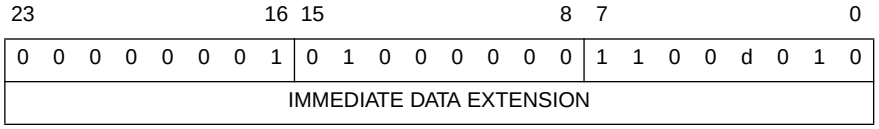
OR S,D



OR #xx,D



OR #xxxxxx,D



Instruction Fields:

{S}	**JJ**	Source input register [X0,X1,Y0,Y1] (see Table A-12 on page A-239)
{D}	**d**	Destination accumulator [A/B] (see Table A-10 on page A-239)
{#xx}	**iiiiii**	6-bit Immediate Short Data
{#xxxxxx}		24-bit Immediate Long Data extension word

ORI ORI

OR Immediate with Control register

Operation:

#xx+D → D

Assembler Syntax:

OR(I) #xx,D

where + denotes the logical inclusive OR operator

Description: Logically OR the 8-bit immediate operand (#xx) with the contents of the destination control register D and store the result in the destination control register. The condition codes are affected only when the condition code register is specified as the destination operand.

Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
●	●	●	●	●	●	●	●
CCR							

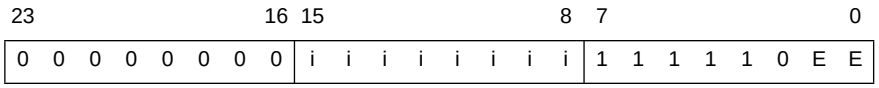
For CCR Operand:

- S Set if bit 7 of the immediate operand is set
- L Set if bit 6 of the immediate operand is set
- E Set if bit 5 of the immediate operand is set
- U Set if bit 4 of the immediate operand is set
- N Set if bit 3 of the immediate operand is set
- Z Set if bit 2 of the immediate operand is set
- V Set if bit 1 of the immediate operand is set
- C Set if bit 0 of the immediate operand is set

For MR and OMR Operands: The condition codes are not affected using these operands.

Instruction Formats and opcodes:

OR(I) #xx,D



Instruction fields:

{D} **EE** Program Controller register [MR,CCR,COM,EOM] (see Table A-13 on page A-239)

{#xx} **iiiiiiii** Immediate Short Data

A-6.92 Program-Cache Flush (PFLUSH)

PFLUSH PFLUSH

Program Cache Flush

Operation:

Flush instruction cache

Assembler Syntax:

PFLUSH

Description: Flush the whole instruction cache, unlock all cache sectors, set the LRU stack and tag registers to their default values.

The PFLUSH instruction is enabled only in Cache Mode. In PRAM Mode it will cause an illegal instruction trap.

Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
x	x	x	x	x	x	x	x
CCR							

x This bit is unchanged by the instruction

Instruction Formats and opcode:

PFLUSH

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Instruction Fields: None

PFLUSHUN PFLUSHUN

Program Cache Flush Unlocked Sectors

Operation:

Flush Unlocked instruction cache sectors

Assembler Syntax:

PFLUSHUN

Description: Flush the instruction cache sectors which are unlocked, set the LRU stack to its default value and set the unlocked tag registers to their default values.

The PFLUSHUN instruction is enabled only in Cache Mode. In PRAM Mode it will cause an illegal instruction trap.

Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
x	x	x	x	x	x	x	x
CCR							

x This bit is unchanged by the instruction

Instruction Formats and opcode:

PFLUSHUN

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Instruction Fields: None

A-6.94 Program-Cache Global Unlock (PFREE)

PFREE PFREE

Program Cache Global Unlock

Operation:

Unlock all locked sectors

Assembler Syntax:

PFREE

Description: Unlock all the locked cache sectors in the instruction cache.

The PFREE instruction is enabled only in Cache Mode. In PRAM Mode it will cause an illegal instruction trap.

Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
×	×	×	×	×	×	×	×
CCR							

× This bit is unchanged by the instruction

Instruction Formats and opcode:

PFREE

23								16	15							8	7							0
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Instruction Fields: None