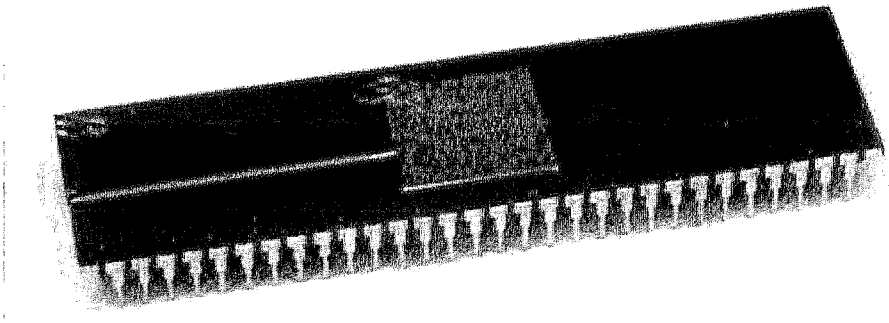
H30

ECE441

Exam Notes



68000 Microprocessor

PLEASE DO NOT WRITE ON THIS NOTE

ABCD — Add Decimal with Extend — ABCD

Operation: (Source)$_{10}$ + (Destination)$_{10}$ + X → Destination

Assembler Syntax: ABCD Dy, Dx
ABCD −(Ay), −(Ax)

Attributes: Size = (Byte)

Description: Add the source operand to the destination operand along with the extend bit, and store the result in the destination location. The addition is performed using binary coded decimal arithmetic. The operands may be addressed in two different ways:

1. Data register to data register: The operands are contained in the data registers specified in the instruction.
2. Memory to memory: The operands are addressed with the predecrement addressing mode using the address registers specified in the instruction.

This operation is a byte operation only.

Condition Codes:

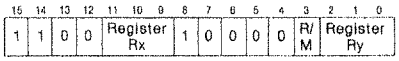
X	N	Z	V	C
*	U	*	U	*

N Undefined.
Z Cleared if the result is non-zero. Unchanged otherwise.
V Undefined.
C Set if a carry (decimal) was generated. Cleared otherwise.
X Set the same as the carry bit.

NOTE

Normally the Z condition code bit is set via programming before the start of an operation. This allows successful tests for zero results upon completion of multiple-precision operations.

Instruction Format:



Instruction Fields:

Register Rx field — Specifies the destination register:
If R/M = 0, specifies a data register.
If R/M = 1, specifies an address register for the predecrement addressing mode.

R/M field — Specifies the operand addressing mode:
0 — The operation is data register to data register.
1 — The operation is memory to memory.

Register Ry field — Specifies the source register:
If R/M = 0, specifies a data register.
If R/M = 1, specifies an address register for the predecrement addressing mode.

ADD — Add Binary — ADD

Operation: (Source) + (Destination) → Destination

Assembler Syntax: ADD <ea>, Dn
ADD Dn, <ea>

Attributes: Size = (Byte, Word, Long)

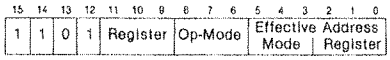
Description: Add the source operand to the destination operand, and store the result in the destination location. The size of the operation may be specified to be byte, word, or long. The mode of the instruction indicates which operand is the source and which is the destination as well as the operand size.

Condition Codes:

X	N	Z	V	C
*	*	*	*	*

N Set if the result is negative. Cleared otherwise.
Z Set if the result is zero. Cleared otherwise.
V Set if an overflow is generated. Cleared otherwise.
C Set if a carry is generated. Cleared otherwise.
X Set the same as the carry bit.

Instruction Format:



Instruction Fields:

Register field — Specifies any of the eight data registers.

Op-Mode field —

Byte	Word	Long	Operation
000	001	010	(<Dn>) + (<ea>) → <Dn>
100	101	110	(<ea>) + (<Dn>) → <ea>

Effective Address field — Determines addressing mode:

a. If the location specified is a source operand, then all addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An*	001	register number	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	111	010
− (An)	100	register number	d(PC, Xi)	111	011
d(An)	101	register number	Imm	111	100

*Word and Long only.

— Continued —

ADD — Add Binary — ADD

Effective Address field (Continued)

b. If the location specified is a destination operand, then only alterable memory addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	—	—	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	—	—
− (An)	100	register number	d(PC, Xi)	—	—
d(An)	101	register number	Imm	—	—

Notes:

1. If the destination is a data register, then it cannot be specified by using the destination <ea> mode, but must use the destination Dn mode instead.
2. ADDA is used when the destination is an address register. ADDI and ADDQ are used when the source is immediate data. Most assemblers automatically make this distinction.

ADDA — Add Address — ADDA

Operation: (Source) + (Destination) → Destination

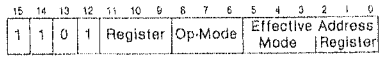
Assembler Syntax: ADD <ea>, An

Attributes: Size = (Word, Long)

Description: Add the source operand to the destination address register, and store the result in the address register. The size of the operation may be specified to be word or long. The entire destination address register is used regardless of the operation size.

Condition Codes: Not affected.

Instruction Format:



Instruction Fields:

Register field — Specifies any of the eight address registers. This is always the destination.

Op-Mode field — Specifies the size of the operation:
011 — word operation. The source operand is sign-extended to a long operand and the operation is performed on the address register using all 32 bits.
111 — long operation.

Effective Address field — Specifies the source operand. All addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An	001	register number	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	111	010
− (An)	100	register number	d(PC, Xi)	111	011
d(An)	101	register number	Imm	111	100

ADDI — Add Immediate

Operation: Immediate Data + (Destination) → Destination

Assembler Syntax: ADDI #<data>, <ea>

Attributes: Size = (Byte, Word, Long)

Description: Add the immediate data to the destination operand, and store the result in the destination location. The size of the operation may be specified to be byte, word, or long. The size of the immediate data matches the operation size.

Condition Codes:

X	N	Z	V	C
*	*	*	*	*

- N Set if the result is negative. Cleared otherwise.
- Z Set if the result is zero. Cleared otherwise.
- V Set if an overflow is generated. Cleared otherwise.
- C Set if a carry is generated. Cleared otherwise.
- X Set the same as the carry bit.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	1	0	Size		Effective Address Mode			Register		
Word Data (16 bits)								Byte Data (8 bits)							
Long Data (32 bits, including previous word)															

Instruction Fields:

Size field — Specifies the size of the operation:
- 00 — byte operation.
- 01 — word operation.
- 10 — long operation.

Effective Address field — Specifies the destination operand. Only data alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	—	—
– (An)	100	register number	d(PC, Xi)	—	—
d(An)	101	register number	Imm	—	—

Immediate field — (Data immediately following the instruction):
- If size = 00, then the data is the low order byte of the immediate word.
- If size = 01, then the data is the entire immediate word.
- If size = 10, then the data is the next two immediate words.

ADDQ — Add Quick

Operation: Immediate Data + (Destination) → Destination

Assembler Syntax: ADDQ #<data>, <ea>

Attributes: Size = (Byte, Word, Long)

Description: Add the immediate data to the operand at the destination location. The data range is from 1 to 8. The size of the operation may be specified to be byte, word, or long. Word and long operations are also allowed on the address registers and the condition codes are not affected. The entire destination address register is used regardless of the operation size.

Condition Codes:

X	N	Z	V	C
*	*	*	*	*

- N Set if the result is negative. Cleared otherwise.
- Z Set if the result is zero. Cleared otherwise.
- V Set if an overflow is generated. Cleared otherwise.
- C Set if a carry is generated. Cleared otherwise.
- X Set the same as the carry bit.

The condition codes are not affected if an addition to an address register is made.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	Data			0	Size		Effective Address Mode			Register		

Instruction Fields:

Data field — Three bits of immediate data, 0, 1-7 representing a range of 8, 1 to 7 respectively.

Size field — Specifies the size of the operation:
- 00 — byte operation.
- 01 — word operation.
- 10 — long operation.

Effective Address field — Specifies the destination location. Only alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An*	001	register number	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	—	—
– (An)	100	register number	d(PC, Xi)	—	—
d(An)	101	register number	Imm	—	—

*Word and Long only.

ADDX — Add Extended

Operation: (Source) + (Destination) + X → Destination

Assembler Syntax: ADDX Dy, Dx
ADDX – (Ay), – (Ax)

Attributes: Size = (Byte, Word, Long)

Description: Add the source operand to the destination operand along with the extend bit and store the result in the destination location. The operands may be addressed in two different ways:
1. Data register to data register: the operands are contained in data registers specified in the instruction.
2. Memory to memory: the operands are addressed with the predecrement addressing mode using the address registers specified in the instruction.

The size of the operation may be specified to be byte, word, or long.

Condition Codes:

X	N	Z	V	C
*	*	*	*	*

- N Set if the result is negative. Cleared otherwise.
- Z Cleared if the result is non-zero. Unchanged otherwise.
- V Set if an overflow is generated. Cleared otherwise.
- C Set if a carry is generated. Cleared otherwise.
- X Set the same as the carry bit.

NOTE

Normally the Z condition code bit is set via programming before the start of an operation. This allows successful tests for zero results upon completion of multiple-precision operations.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	Register Rx			1	Size		0	0	R/M	Register Ry		

Instruction Fields:

Register Rx field — Specifies the destination register:
- If R/M = 0, specifies a data register.
- If R/M = 1, specifies an address register for the predecrement addressing mode.

Size field — Specifies the size of the operation:
- 00 — byte operation.
- 01 — word operation.
- 10 — long operation.

— Continued —

ADDX — Add Extended

Instruction Fields: (Continued)

R/M field — Specifies the operand addressing mode:
- 0 — The operation is data register to data register.
- 1 — The operation is memory to memory.

Register Ry field — Specifies the source register:
- If R/M = 0, specifies a data register.
- If R/M = 1, specifies an address register for the predecrement addressing mode.

AND

AND Logical

Operation: (Source)∧(Destination) → Destination

Assembler Syntax: AND <ea>, Dn
AND Dn, <ea>

Attributes: Size = (Byte, Word, Long)

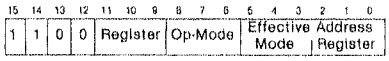
Description: AND the source operand to the destination operand and store the result in the destination location. The size of the operation may be specified to be byte, word, or long. The contents of an address register may not be used as an operand.

Condition Codes:

X	N	Z	V	C
—	*	*	0	0

N Set if the most significant bit of the result is set. Cleared otherwise.
Z Set if the result is zero. Cleared otherwise.
V Always cleared.
C Always cleared.
X Not affected.

Instruction Format:



Instruction Fields:

Register field — Specifies any of the eight data registers.

Op-Mode field —

Byte	Word	Long	Operation
000	001	010	(<Dn>) ∧ (<ea>) → <Dn>
100	101	110	(<ea>) ∧ (<Dn>) → <ea>

Effective Address field — Determines addressing mode:

If the location specified is a source operand then only data addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	111	010
– (An)	100	register number	d(PC, Xi)	111	011
d(An)	101	register number	imm	111	100

— Continued —

AND

AND Logical

Effective Address field (Continued)

If the location specified is a destination operand then only alterable memory addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	—	—	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	—	—
– (An)	100	register number	d(PC, Xi)	—	—
d(An)	101	register number	imm	—	—

Notes:

1. If the destination is a data register, then it cannot be specified by using the destination <ea> mode, but must use the destination Dn mode instead.
2. ANDI is used when the source is immediate data. Most assemblers automatically make this distinction.

ANDI

AND Immediate

Operation: Immediate Data ∧ (Destination) → Destination

Assembler Syntax: ANDI #<data>, <ea>

Attributes: Size = (Byte, Word, Long)

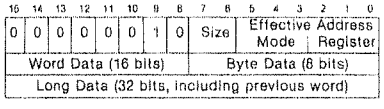
Description: AND the immediate data to the destination operand and store the result in the destination location. The size of the operation may be specified to be byte, word, or long. The size of the immediate data matches the operation size.

Condition Codes:

X	N	Z	V	C
—	*	*	0	0

N Set if the most significant bit of the result is set. Cleared otherwise.
Z Set if the result is zero. Cleared otherwise.
V Always cleared.
C Always cleared.
X Not affected.

Instruction Format:



Instruction Fields:

Size field — Specifies the size of the operation:
00 — byte operation.
01 — word operation.
10 — long operation.

Effective Address field — Specifies the destination operand. Only data alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	—	—
– (An)	100	register number	d(PC, Xi)	—	—
d(An)	101	register number	Imm	—	—

Immediate field — (Data immediately following the instruction):
If size = 00, then the data is the low order byte of the immediate word.
If size = 01, then the data is the entire immediate word.
If size = 10, then the data is the next two immediate words.

ANDI to CCR

AND Immediate to Condition Codes

Operation: (Source)∧CCR → CCR

Assembler Syntax: ANDI #xxx, CCR

Attributes: Size = (Byte)

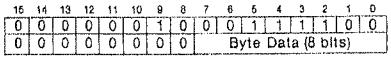
Description: AND the immediate operand with the condition codes and store the result in the low-order byte of the status register.

Condition Codes:

X	N	Z	V	C
*	*	*	*	*

N Cleared if bit 3 of immediate operand is zero. Unchanged otherwise.
Z Cleared if bit 2 of immediate operand is zero. Unchanged otherwise.
V Cleared if bit 1 of immediate operand is zero. Unchanged otherwise.
C Cleared if bit 0 of immediate operand is zero. Unchanged otherwise.
X Cleared if bit 4 of immediate operand is zero. Unchanged otherwise.

Instruction Format:



ANDI to SR

AND Immediate to the Status Register (Privileged Instruction)

ANDI to SR

Operation: If supervisor state
then (Source)∧SR → SR
else TRAP

Assembler Syntax: ANDI #xxx, SR

Attributes: Size = (Word)

Description: AND the immediate operand with the contents of the status register and store the result in the status register. All bits of the status register are affected.

Condition Codes:

X	N	Z	V	C
*	*	*	*	*

N Cleared if bit 3 of immediate operand is zero. Unchanged otherwise.
Z Cleared if bit 2 of immediate operand is zero. Unchanged otherwise.
V Cleared if bit 1 of immediate operand is zero. Unchanged otherwise.
C Cleared if bit 0 of immediate operand is zero. Unchanged otherwise.
X Cleared if bit 4 of immediate operand is zero. Unchanged otherwise.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	0	0	1	1	1	1	1	0	0
Word Data (16 bits)															

ASL, ASR

Arithmetic Shift

ASL, ASR

Operation: (Destination) Shifted by <count> → Destination

Assembler Syntax: ASd Dx, Dy
ASd #<data>, Dy
ASd <ea>

Attributes: Size = (Byte, Word, Long)

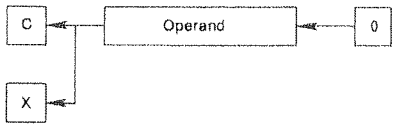
Description: Arithmetically shift the bits of the operand in the direction specified. The carry bit receives the last bit shifted out of the operand. The shift count for the shifting of a register may be specified in two different ways:

1. Immediate: the shift count is specified in the instruction (shift range, 1-8).
2. Register: the shift count is contained in a data register specified in the instruction.

The size of the operation may be specified to be byte, word, or long. The content of memory may be shifted one bit only and the operand size is restricted to a word.

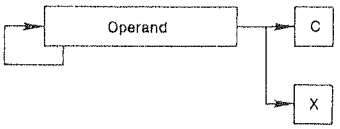
For ASL, the operand is shifted left; the number of positions shifted is the shift count. Bits shifted out of the high order bit go to both the carry and the extend bits; zeroes are shifted into the low order bit. The overflow bit indicates if any sign changes occur during the shift.

ASL:



For ASR, the operand is shifted right; the number of positions shifted is the shift count. Bits shifted out of the low order bit go to both the carry and the extend bits; the sign bit is replicated into the high order bit.

ASR:



— Continued —

ASL, ASR

Arithmetic Shift

ASL, ASR

Condition Codes:

X	N	Z	V	C
*	*	*	*	*

N Set if the most significant bit of the result is set. Cleared otherwise.
Z Set if the result is zero. Cleared otherwise.
V Set if the most significant bit is changed at any time during the shift operation. Cleared otherwise.
C Set according to the last bit shifted out of the operand. Cleared for a shift count of zero.
X Set according to the last bit shifted out of the operand. Unaffected for a shift count of zero.

Instruction Format (Register Shifts):

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	Count/ Register			dr	Size		i/r	0	0	Register		

Instruction Fields (Register Shifts):

Count/Register field — Specifies shift count or register where count is located:
If i/r = 0, the shift count is specified in this field. The values 0, 1-7 represent a range of 8, 1 to 7 respectively.
If i/r = 1, the shift count (modulo 64) is contained in the data register specified in this field.

dr field — Specifies the direction of the shift:
0 — shift right.
1 — shift left.

Size field — Specifies the size of the operation:
00 — byte operation.
01 — word operation.
10 — long operation.

i/r field —
If i/r = 0, specifies immediate shift count.
If i/r = 1, specifies register shift count.

Register field — Specifies a data register whose content is to be shifted.

Instruction Format (Memory Shifts):

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	0	0	0	dr	1	1	Effective Address Mode			Register		

— Continued —

ASL, ASR

Arithmetic Shift

ASL, ASR

Instruction Fields (Memory Shifts):

dr field — Specifies the direction of the shift:
0 — shift right.
1 — shift left.

Effective Address field — Specifies the operand to be shifted. Only memory alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	—	—	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	—	—
– (An)	100	register number	d(PC, Xi)	—	—
d(An)	101	register number	imm	—	—

Bcc — Branch Conditionally

Operation: If (condition true) then PC + d → PC

Assembler Syntax: Bcc <label>

Attributes: Size = (Byte, Word)

Description: If the specified condition is met, program execution continues at location (PC) + displacement. Displacement is a twos complement integer which counts the relative distance in bytes. The value in PC is the current instruction location plus two. If the 8-bit displacement in the instruction word is zero, then the 16-bit displacement (word immediately following the instruction) is used. "cc" may specify the following conditions:

CC	carry clear	0100	$\overline{C}$	LS	low or same	0011	C + Z
CS	carry set	0101	C	LT	less than	1101	$N \cdot \overline{V} + \overline{N} \cdot V$
EQ	equal	0111	Z	MI	minus	1011	N
GE	greater or equal	1100	$N \cdot V + \overline{N} \cdot \overline{V}$	NE	not equal	0110	$\overline{Z}$
GT	greater than	1110	$N \cdot V \cdot \overline{Z} + \overline{N} \cdot \overline{V} \cdot \overline{Z}$	PL	plus	1010	$\overline{N}$
HI	high	0010	$\overline{C} \cdot \overline{Z}$	VC	overflow clear	1000	$\overline{V}$
LE	less or equal	1111	$Z + N \cdot \overline{V} + \overline{N} \cdot V$	VS	overflow set	1001	V

Condition Codes: Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	Condition				8-bit Displacement							
16-bit Displacement if 8-bit Displacement = 0															

Instruction Fields:

Condition field — One of fourteen conditions discussed in description.

8-bit Displacement field — Twos complement integer specifying the relative distance (in bytes) between the branch instruction and the next instruction to be executed if the condition is met.

16-bit Displacement field — Allows a larger displacement than 8 bits. Used only if the 8-bit displacement is equal to zero.

Note: A short branch to the immediately following instruction cannot be done because it would result in a zero offset which forces a word branch instruction definition.

BCHG — Test a Bit and Change

Instruction Format (Bit Number Static):

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	0	0	0	1	Effective Address Mode			Register		
0	0	0	0	0	0	0	0	bit number							

Instruction Fields (Bit Number Static):

Effective Address field — Specifies the destination location. Only data alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	—	—
– (An)	100	register number	d(PC, Xi)	—	—
d(An)	101	register number	Imm	—	—

*Long only; all others are byte only.

bit number field — Specifies the bit numbers.

BCHG — Test a Bit and Change

Operation: ~(<bit number>) OF Destination → Z;
~(<bit number>) OF Destination → <bit number> OF Destination

Assembler Syntax: BCHG Dn, <ea>
BCHG #<data>, <ea>

Attributes: Size = (Byte, Long)

Description: A bit in the destination operand is tested and the state of the specified bit is reflected in the Z condition code. After the test, the state of the specified bit is changed in the destination. If a data register is the destination, then the bit numbering is modulo 32 allowing bit manipulation on all bits in a data register. If a memory location is the destination, a byte is read from that location, the bit operation performed using the bit number modulo 8, and the byte written back to the location with zero referring to the least-significant bit. The bit number for this operation may be specified in two different ways:

1. Immediate — the bit number is specified in a second word of the instruction.
2. Register — the bit number is contained in a data register specified in the instruction.

Condition Codes:

X	N	Z	V	C
—	—	*	—	—

N Not affected.
Z Set if the bit tested is zero. Cleared otherwise.
V Not affected.
C Not affected.
X Not affected.

Instruction Format (Bit Number Dynamic):

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	Register			1	0	1	Effective Address Mode			Register		

Instruction Fields (Bit Number Dynamic):

Register field — Specifies the data register whose content is the bit number.

Effective Address field — Specifies the destination location. Only data alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn*	000	register number	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	—	—
– (An)	100	register number	d(PC, Xi)	—	—
d(An)	101	register number	Imm	—	—

*Long only; all others are byte only.

— Continued —

BCLR — Test a Bit and Clear

Operation: ~(<bit number>) OF Destination) → Z;
0 → <bit number> OF Destination

Assembler Syntax: BLCR Dn, <ea>
BCLR #<data>, <ea>

Attributes: Size = (Byte, Long)

Description: A bit in the destination operand is tested and the state of the specified bit is reflected in the Z condition code. After the test, the specified bit is cleared in the destination. If a data register is the destination, then the bit numbering is modulo 32 allowing bit manipulation on all bits in a data register. If a memory location is the destination, a byte is read from that location, the bit operation performed using the bit number modulo 8, and the byte written back to the location with zero referring to the least-significant bit. The bit number for this operation may be specified in two different ways:

1. Immediate — the bit number is specified in a second word of the instruction.
2. Register — the bit number is contained in a data register specified in the instruction.

Condition Codes:

X	N	Z	V	C
—	—	*	—	—

N Not affected.
Z Set if the bit tested is zero. Cleared otherwise.
V Not affected.
C Not affected.
X Not affected.

Instruction Format (Bit Number Dynamic):

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	Register			1	1	0	Effective Address Mode			Register		

Instruction Fields (Bit Number Dynamic):

Register field — Specifies the data register whose content is the bit number.

Effective Address field — Specifies the destination location. Only data alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn*	000	register number	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	—	—
– (An)	100	register number	d(PC, Xi)	—	—
d(An)	101	register number	Imm	—	—

*Long only; all others are byte only.

— Continued —

BCLR — Test a Bit and Clear

BCLR

Instruction Format (Bit Number Static):

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	0	0	1	0	Effective Address					
										Mode			Register		
0	0	0	0	0	0	0	0	bit number							

Instruction Fields (Bit Number Static):

Effective Address field — Specifies the destination location. Only data alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn*	000	register only	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An)+	011	register number	d(PC)	—	—
−(An)	100	register number	d(PC, Xi)	—	—
d(An)	101	register number	Imm	—	—

*Long only; all others are byte only.

bit number field — Specifies the bit number.

BKPT — Breakpoint

BKPT

Operation Acknowledge breakpoint
then Trap as illegal instruction

Assembler Syntax: BKPT #<data>

Attributes: Unsized

Description: This instruction is used to support the program breakpoint function for debug monitors and real-time hardware emulators, and the operation will be dependent on the implementation. Execution of this instruction will cause the processor to run a breakpoint acknowledge bus cycle, with zeros on all address lines.

Following the termination of the breakpoint acknowledge, the processor will then take an illegal instruction exception.

Condition Codes: Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	0	0	0	1	0	0	1	Vector		

Instruction Fields:

Vector field — Specifies the breakpoint number.

BRA — Branch Always

BRA

Operation: PC + d → PC

Assembler Syntax: BRA <label>

Attributes: Size = (Byte, Word)

Description: Program execution continues at location (PC) + displacement. Displacement is a twos complement integer which counts the relative distance in bytes. The value in PC is the current instruction location plus two. If the 8-bit displacement in the instruction word is zero, then the 16-bit displacement (word immediately following the instruction) is used.

Condition Codes: Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	0	0	0	8-bit Displacement							
16-bit Displacement if 8-bit Displacement = 0															

Instruction Fields:

8-bit Displacement field — Twos complement integer specifying the relative distance (in bytes) between the branch instruction and the next instruction to be executed if the condition is met.

16-bit Displacement field — Allows a larger displacement than 8 bits. Used only if the 8-bit displacement is equal to zero.

Note: A short branch to the immediately following instruction cannot be done because it would result in a zero offset which forces a word branch instruction definition.

BSET — Test a Bit and Set

BSET

Operation: −(<bit number>) OF Destination → Z
1 → <bit number> OF Destination

Assembler Syntax: BSET Dn, <ea>
BSET #<data>, <ea>

Attributes: Size = (Byte, Long)

Description: A bit in the destination operand is tested and the state of the specified bit is reflected in the Z condition code. After the test, the specified bit is set in the destination. If a data register is the destination, then the bit numbering is modulo 32, allowing bit manipulation on all bits in a data register. If a memory location is the destination, a byte is read from that location, the bit operation performed using the bit number modulo 8, and the byte written back to the location with zero referring to the least-significant bit. The bit number for this operation may be specified in two different ways:

1. Immediate — the bit number is specified in a second word of the instruction.
2. Register — the bit number is contained in a data register specified in the instruction.

Condition Codes:

X	N	Z	V	C
—	—	*	—	—

N Not affected.
Z Set if the bit tested is zero. Cleared otherwise.
V Not affected.
C Not affected.
X Not affected.

Instruction Format (Bit Number Dynamic):

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	Register			1	1	1	Effective Address					
										Mode			Register		

Instruction Fields (Bit Number Dynamic):

Register field — Specifies the data register whose content is the bit number.

Effective Address field — Specifies the destination location. Only data alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn*	000	register number	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An)+	011	register number	d(PC)	—	—
−(An)	100	register number	d(PC, Xi)	—	—
d(An)	101	register number	Imm	—	—

*Long only; all others are byte only

— Continued —

BSET — Test a Bit and Set

BSET

Instruction Format (Bit Number Static):

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	0	0	1	1	Effective Address Mode			Register		
0	0	0	0	0	0	0	0	bit number							

Instruction Fields (Bit Number Static):

Effective Address field — Specifies the destination location. Only data alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn*	000	register number	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An)+	011	register number	d(PC)	—	—
–(An)	100	register number	d(PC, Xi)	—	—
d(An)	101	register number	Imm	—	—

*Long only; all others are byte only.

bit number field — Specifies the bit numbe[illegible].

BSR — Branch to Subroutine

BSR

Operation: PC → –(SP); PC + d → PC

Assembler Syntax: BSR <label>

Attributes: Size = (Byte, Word)

Description: The long word address of the instruction immediately following the BSR instruction is pushed onto the system stack. Program execution then continues at location (PC) + displacement. Displacement is a twos complement integer which counts the relative distances in bytes. The value in PC is the current instruction location plus two. If the 8-bit displacement in the instruction word is zero, then the 16-bit displacement (word immediately following the instruction) is used.

Condition Codes: Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	0	0	1	8-bit Displacement							
16-bit Displacement if 8-bit Displacement = 0															

Instruction Fields:

8-bit Displacement field — Twos complement integer specifying the relative distance (in bytes) between the branch instruction and the next instruction to be executed if the condition is met.

16-bit Displacement field — Allows a larger displacement than 8 bits. Used only if the 8-bit displacement is equal to zero.

Note: A short subroutine branch to the immediately following instruction cannot be done because it would result in a zero offset which forces a word branch instruction definition.

BTST — Test a Bit

BTST

Operation: ~(<bit number>) OF Destination → Z

Assembler Syntax: BTST Dn, <ea>
BTST #<data>, <ea>

Attributes: Size = (Byte, Long)

Description: A bit in the destination operand is tested and the state of the specified bit is reflected in the Z condition code. If a data register is the destination, then the bit numbering is modulo 32, allowing bit manipulation on all bits in a data register. If a memory location is the destination, a byte is read from that location, and the bit operation performed using the bit number modulo 8 with zero referring to the least-significant bit. The bit number for this operation may be specified in two different ways:

1. Immediate — the bit number is specified in a second word of the instruction.
2. Register — the bit number is contained in a data register specified in the instruction.

Condition Codes:

X	N	Z	V	C
—	—	*	—	—

N Not affected.
Z Set if the bit tested is zero. Cleared otherwise.
V Not affected.
C Not affected.
X Not affected.

Instruction Format (Bit Number Dynamic):

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	Register			1	0	0	Effective Address Mode			Register		

Instruction Fields (Bit Number Dynamic):

Register field — Specifies the data register whose content is the bit number.

Effective Address field — Specifies the destination location. Only data addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn*	000	register number	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An)+	011	register number	d(PC)	111	010
–(An)	100	register number	d(PC, Xi)	111	011
d(An)	101	register number	Imm	111	100

*Long only; all others are byte only.

— Continued —

BTST — Test a Bit

BTST

Instruction Format (Bit Number Static):

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	0	0	0	0	Effective Address Mode			Register		
0	0	0	0	0	0	0	0	bit number							

Instruction Fields (Bit Number Static):

Effective Address field — Specifies the destination location. Only data addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn*	000	register number	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An)+	011	register number	d(PC)	111	010
–(An)	100	register number	d(PC, Xi)	111	011
d(An)	101	register number	Imm	—	—

*Long only; all others are byte only.

bit number field — Specifies the bit number.

CHK — Check Register Against Bounds

Operation: If Dn<0 or Dn> (<ea>) then TRAP

Assembler Syntax: CHK <ea>, Dn

Attributes: Size = (Word)

Description: The content of the low order word in the data register specified in the instruction is examined and compared to the upper bound. The upper bound is a twos complement integer. If the register value is less than zero or greater than the upper bound contained in the operand word, then the processor initiates exception processing. The vector number is generated to reference the CHK instruction exception vector.

Condition Codes:

X	N	Z	V	C
—	*	U	U	U

N Set if Dn<0; cleared if Dn> (<ea>). Undefined otherwise.
Z Undefined.
V Undefined.
C Undefined.
X Not affected.

Instruction Format:

15	14	13	12	11 10 9	8	7	6	5 4 3	2 1 0
0	1	0	0	Register	1	1	0	Effective Address Mode	Effective Address Register

Instruction Fields:
Register field — Specifies the data register whose content is checked.
Effective Address field — Specifies the upper bound operand word. Only data addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An)+	011	register number	d(PC)	111	010
–(An)	100	register number	d(PC, Xi)	111	011
d(An)	101	register number	Imm	111	100

CLR — Clear an Operand

Operation: 0 → Destination

Assembler Syntax: CLR <ea>

Attributes: Size = (Byte, Word, Long)

Description: The destination is cleared to all zero bits. The size of the operation may be specified to be byte, word, or long.

Condition Codes:

X	N	Z	V	C
—	0	1	0	0

N Always cleared.
Z Always set.
V Always cleared.
C Always cleared.
X Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7 6	5 4 3	2 1 0
0	1	0	0	0	0	1	0	Size	Effective Address Mode	Effective Address Register

Instruction Fields:
Size field — Specifies the size of the operation:
00 — byte operation.
01 — word operation.
10 — long operation.
Effective Address field — Specifies the destination location. Only data alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An)+	011	register number	d(PC)	—	—
–(An)	100	register number	d(PC, Xi)	—	—
d(An)	101	register number	Imm	—	—

Note: A memory destination is read before it is written to.

CMP — Compare

Operation: (Destination) – (Source)

Assembler Syntax: CMP <ea>, Dn

Attributes: Size = (Byte, Word, Long)

Description: Subtract the source operand from the destination operand and set the condition codes according to the result; the destination location is not changed. The size of the operation may be specified to be byte, word, or long.

Condition Codes:

X	N	Z	V	C
—	*	*	*	*

N Set if the result is negative. Cleared otherwise.
Z Set if the result is zero. Cleared otherwise.
V Set if an overflow is generated. Cleared otherwise.
C Set if a borrow is generated. Cleared otherwise.
X Not affected.

Instruction Format:

15	14	13	12	11 10 9	8 7 6	5 4 3	2 1 0
1	0	1	1	Register	Op-Mode	Effective Address Mode	Effective Address Register

Instruction Fields:
Register field — Specifies the destination data register.
Op-Mode field —

Byte	Word	Long	Operation
000	001	010	(<Dn>) – (<ea>)

Effective Address field — Specifies the source operand. All addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An*	001	register number	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An)+	011	register number	d(PC)	111	010
–(An)	100	register number	d(PC, Xi)	111	011
d(An)	101	register number	Imm	111	100

*Word and Long only.

Note: CMPA is used when the destination is an address register; CMPI is used when the source is immediate data. CMPM is used for memory to memory compares. Most assemblers automatically make this distinction.

CMPA — Compare Address

Operation: (Destination) – (Source)

Assembler Syntax: CMPA <ea>, An

Attributes: Size = (Word, Long)

Description: Subtract the source operand from the destination address register and set the condition codes according to the result; the address register is not changed. The size of the operation may be specified to be word or long. Word length source operands are sign extended to 32 bit quantities before the operation is done.

Condition Code:

X	N	Z	V	C
—	*	*	*	*

N Set if the result is negative. Cleared otherwise.
Z Set if the result is zero. Cleared otherwise.
V Set if an overflow is generated. Cleared otherwise.
C Set if a borrow is generated. Cleared otherwise.
X Not affected.

Instruction Format:

15	14	13	12	11 10 9	8 7 6	5 4 3	2 1 0
1	0	1	1	Register	Op-Mode	Effective Address Mode	Effective Address Register

Instruction Fields:
Register field — Specifies the destination address register.
Op-Mode field — Specifies the size of the operation:
011 — word operation. The source operand is sign-extended to a long operand and the operation is performed on the address register using all 32 bits.
111 — long operation.
Effective Address field — Specifies the source operand. All addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An	001	register number	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An)+	011	register number	d(PC)	111	010
–(An)	100	register number	d(PC, Xi)	111	011
d(An)	101	register number	Imm	111	100

CMPI — Compare Immediate — CMPI

Operation: (Destination) – Immediate Data

Assembler Syntax: CMPI #<data>, <ea>

Attributes: Size = (Byte, Word, Long)

Description: Subtract the immediate data from the destination operand and set the condition codes according to the result; the destination location is not changed. The size of the operation may be specified to be byte, word, or long. The size of the immediate data matches the operation size.

Condition Codes:

X	N	Z	V	C
—	*	*	*	*

N Set if the result is negative. Cleared otherwise.
Z Set if the result is zero. Cleared otherwise.
V Set if an overflow is generated. Cleared otherwise.
C Set if a borrow is generated. Cleared otherwise.
X Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	0	Size		Effective Address Mode			Register		
Word Data (16 bits)								Byte Data (8 bits)							
Long Data (32 bits, including previous word)															

Instruction Fields:

Size field — Specifies the size of the operation:
00 — byte operation.
01 — word operation.
10 — long operation.

Effective Address field — Specifies the destination operand. Only data alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	—	—
– (An)	100	register number	d(PC, Xi)	—	—
d(An)	101	register number	Imm	—	—

Immediate field — (Data immediately following the instruction):
If size = 00, then the data is the low order byte of the immediate word.
If size = 01, then the data is the entire immediate word.
If size = 10, then the data is the next two immediate words.

BTST — Test a Bit — BTST

Instruction Format (Bit Number Static):

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	0	0	0	0	Effective Address Mode			Register		
0	0	0	0	0	0	0	0	bit number							

Instruction Fields (Bit Number Static):

Effective Address field — Specifies the destination location. Only data addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn*	000	register number	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	111	010
– (An)	100	register number	d(PC, Xi)	111	011
d(An)	101	register number	Imm	—	—

*Long only; all others are byte only.

bit number field — Specifies the bit number.

DBcc — Test Condition, Decrement, and Branch — DBcc

Operation: If (condition false)
then Dn – 1 → Dn;
If Dn ≠ – 1
then PC + d → PC
else PC + 2 → PC (Fall through to next instruction)

Assembler Syntax: DBcc Dn, <label>

Attributes: Size = (Word)

Description: This instruction is a looping primitive of three parameters: a condition, a data register, and a displacement. The instruction first tests the condition to determine if the termination condition for the loop has been met, and if so, no operation is performed. If the termination condition is not true, the low order 16 bits of the counter data register are decremented by one. If the result is – 1, the counter is exhausted and execution continues with the next instruction. If the result is not equal to – 1, execution continues at the location indicated by the current value of PC plus the sign-extended 16-bit displacement. The value in PC is the current instruction location plus two "cc" may specify the following conditions:

CC	carry clear	0100	$\bar{C}$	LS	low or same	0011	C + Z
CS	carry set	0101	C	LT	less than	1101	$N \cdot \bar{V} + \bar{N} \cdot V$
EQ	equal	0111	Z	MI	minus	1011	N
F	false	0001	0	NE	not equal	0110	$\bar{Z}$
GE	greater or equal	1100	$N \cdot V + \bar{N} \cdot \bar{V}$	PL	plus	1010	$\bar{N}$
GT	greater than	1110	$N \cdot V \cdot \bar{Z} + \bar{N} \cdot \bar{V} \cdot \bar{Z}$	T	true	0000	1
HI	high	0010	$\bar{C} \cdot \bar{Z}$	VC	overflow clear	1000	$\bar{V}$
LE	less or equal	1111	$Z + N \cdot \bar{V} + \bar{N} \cdot V$	VS	overflow set	1001	V

Condition Codes: Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	Condition				1	1	0	0	1	Register		
Displacement															

Instruction Fields:

Condition field — One of the sixteen conditions discussed in description.
Register field — Specifies the data register which is the counter.
Displacement field — Specifies the distance of the branch (in bytes).

Notes:

1. The terminating condition is like that defined by the UNTIL loop constructs of high-level languages. For example: DBMI can be stated as "decrement and branch until minus."

— Continued —

DBcc — Test Condition, Decrement and Branch — DBcc

Notes: (Continued)

2. Most assemblers accept DBRA for DBF for use when no condition is required for termination of a loop.
3. There are two basic ways of entering a loop; at the beginning or by branching to the trailing DBcc instruction. If a loop structure terminated with DBcc is entered at the beginning, the control index count must be one less than the number of loop executions desired. This count is useful for indexed addressing modes and dynamically specified bit operations. However, when entering a loop by branching directly to the trailing DBcc instruction, the control index should equal the loop execution count. In this case, if a zero count occurs, the DBcc instruction will not branch causing complete bypass of the main loop.

DIVS — Signed Divide

DIVS

Operation: (Destination)/(Source)→Destination

Assembler Syntax: DIVS <ea>, Dn

Attributes: Size = (Word)

Description: Divide the destination operand by the source operand and store the result in the destination. The destination operand is a long operand (32 bits) and the source operand is a word operand (16 bits). The operation is performed using signed arithmetic. The result is a 32-bit result such that:

1. The quotient is in the lower word (least significant 16-bits).
2. The remainder is in the upper word (most significant 16-bits).

The sign of the remainder is always the same as the dividend unless the remainder is equal to zero. Two special conditions may arise:

1. Division by zero causes a trap.
2. Overflow may be detected and set before completion of the instruction. If overflow is detected, the condition is flagged but the operands are unaffected.

Condition Codes:

X	N	Z	V	C
—	*	*	*	0

- N Set if the quotient is negative. Cleared otherwise. Undefined if overflow.
- Z Set if the quotient is zero. Cleared otherwise. Undefined if overflow.
- V Set if division overflow is detected. Cleared otherwise.
- C Always cleared.
- X Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	Register			1	1	1	Effective Address Mode			Register		

Instruction Fields:

Register field — Specifies any of the eight data registers. This field always specifies the destination operand.

Effective Address field — Specifies the source operand. Only data addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An)+	011	register number	d(PC)	111	010
−(An)	100	register number	d(PC, Xi)	111	011
d(An)	101	register number	Imm	111	100

Note: Overflow occurs if the quotient is larger than a 16-bit signed integer.

DIVU — Unsigned Divide

DIVU

Operation: (Destination)/(Source)→Destination

Assembler Syntax: DIVU <ea>, Dn

Attributes: Size = (Word)

Description: Divide the destination operand by the source operand and store the result in the destination. The destination operand is a long operand (32 bits) and the source operand is a word (16 bit) operand. The operation is performed using unsigned arithmetic. The result is a 32-bit result such that:

1. The quotient is in the lower word (least significnat 16 bits).
2. The remainder is in the upper word (most significant 16 bits).

Two special conditions may arise:

1. Division by zero causes a trap.
2. Overflow may be detected and set before completion of the instruction. If overflow is detected, the condition is flagged but the operands are unaffected.

Condition Codes:

X	N	Z	V	C
—	*	*	*	0

- N Set if the most significant bit of the quotient is set. Cleared otherwise. Undefined if overflow.
- Z Set if the quotient is zero. Cleared otherwise. Undefined if overflow.
- V Set if division overflow is detected. Cleared otherwise.
- C Always cleared.
- X Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	Register			0	1	1	Effective Address Mode			Register		

Instruction Fields:

Register field — specifies any of the eight data registers. This field always specifies the destination operand.

Effective Address field — Specifies the source operand. Only data addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An)+	011	register number	d(PC)	111	010
−(An)	100	register number	d(PC, Xi)	111	011
d(An)	101	register number	Imm	111	100

Note: Overflow occurs if the quotient is larger than a 16-bit unsigned integer.

EOR — Exclusive OR Logical

EOR

Operation: (Source)⊕(Destination)→Destination

Assembler Syntax: EOR Dn, <ea>

Attributes: Size = (Byte, Word, Long)

Description: Exclusive OR the source operand to the destination operand and store the result in the destination location. The size of the operation may be specified to be byte, word, or long. This operation is restricted to data registers as the source operand. The destination operand is specified in the effective address field.

Condition Codes:

X	N	Z	V	C
—	*	*	0	0

- N Set if the most significant bit of the result is set. Cleared otherwise.
- Z Set if the result is zero. Cleared otherwise.
- V Always cleared.
- C Always cleared.
- X Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	Register			Op-Mode			Effective Address Mode			Register		

Instruction Fields:

Register field — Specifies any of the eight data registers.

Op-Mode field —

Byte	Word	Long	Operation
100	101	110	(<ea>)⊕(<Dx>)→<ea>

Effective Address field — Specifies the destination operand. Only data alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An)+	011	register number	d(PC)	—	—
−(An)	100	register number	d(PC, Xi)	—	—
d(An)	101	register number	Imm	—	—

Note: Memory to data register operations are not allowed. EORI is used when the source is immediate data. Most assemblers automatically make this distinction.

EORI — Exclusive OR Immediate

EORI

Operation: Immediate Data ⊕ (Destination)→Destination

Assembler Syntax: EORI #<data>, <ea>

Attributes: Size = (Byte, Word, Long)

Description: Exclusive OR the immediate data to the destination operand and store the result in the destination location. The size of the operation may be specified to be byte, word, or long. The immediate data matches the operation size.

Condition Codes:

X	N	Z	V	C
—	*	*	0	0

- N Set if the most significant bit of the result is set. Cleared otherwise.
- Z Set if the result is zero. Cleared otherwise.
- V Always cleared.
- C Always cleared.
- X Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	0	Size		Effective Address Mode			Register		
Word Data (16 bits)										Byte Data (8 bits)					
Long Data (32 bits, including previous word)															

Instruction Fields:

Size field — Specifies the size of the operation:

- 00 — byte operation.
- 01 — word operation.
- 10 — long operation.

Effective Address field — Specifies the destination operand. Only data alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An)+	011	register number	d(PC)	—	—
−(An)	100	register number	d(PC, Xi)	—	—
d(An)	101	register number	Imm	—	—

Immediate field — (Data immediately following the instruction):

If size = 00, then the data is the low order byte of the immediate word.
If size = 01, then the data is the entire immediate word.
If size = 10, then the data is the next two immediate words.

EORI to CCR — Exclusive OR Immediate to Condition Codes

EORI to CCR

Operation: (Source) ⊕ CCR → CCR

Assembler Syntax: EORI #xxx, CCR

Attributes: Size = (Byte)

Description: Exclusive OR the immediate operand with the condition codes and store the result in the low-order byte of the status register.

Condition Codes:

X	N	Z	V	C
*	*	*	*	*

- N Changed if bit 3 of immediate operand is one. Unchanged otherwise.
- Z Changed if bit 2 of immediate operand is one. Unchanged otherwise.
- V Changed if bit 1 of immediate operand is one. Unchanged otherwise.
- C Changed if bit 0 of immediate operand is one. Unchanged otherwise.
- X Changed if bit 4 of immediate operand is one. Unchanged otherwise.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	0	0	0	1	1	1	1	0	0
0	0	0	0	0	0	0	0	Byte Data (8 bits)							

EORI to SR — Exclusive OR Immediate to the Status Register (Privileged Instruction)

EORI to SR

Operation: If supervisor state
then (Source) ⊕ SR → SR
else TRAP

Assembler Syntax: EORI #xxx, SR

Attributes: Size = (Word)

Description: Exclusive OR the immediate operand with the contents of the status register and store the result in the status register. All bits of the status register are affected.

Condition Codes:

X	N	Z	V	C
*	*	*	*	*

- N Changed if bit 3 of immediate operand is one. Unchanged otherwise.
- Z Changed if bit 2 of immediate operand is one. Unchanged otherwise.
- V Changed if bit 1 of immediate operand is one. Unchanged otherwise.
- C Changed if bit 0 of immediate operand is one. Unchanged otherwise.
- X Changed if bit 4 of immediate operand is one. Unchanged otherwise.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	0	0	1	1	1	1	1	0	0
Word Data (16 bits)															

EXG — Exchange Registers

EXG

Operation: Rx ↔ Ry

Assembler Syntax: EXG Rx, Ry

Attributes: Size = (Long)

Description: Exchange the contents of two registers. This exchange is always a long (32 bit) operation. Exchange works in three modes:

1. Exchange data registers.
2. Exchange address registers.
3. Exchange a data register and an address register.

Condition Codes: Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	Register Rx			1	Op-Mode					Register Ry		

Instruction Fields:

Register Rx field — Specifies either a data register or an address register depending on the mode. If the exchange is between data and address registers, this field always specifies the data register.

Op-Mode field — Specifies whether exchanging:

- 01000 — data registers.
- 01001 — address registers.
- 10001 — data register and address register.

Register Ry field — Specifies either a data register or an address register depending on the mode. If the exchange is between data and address registers, this field always specifies the address register.

EXT — Sign Extend

EXT

Operation: (Destination) Sign-extended → Destination

Assembler Syntax: EXT Dn

Attributes: Size = (Word, Long)

Description: Extend the sign bit of a data register from a byte to a word or from a word to a long operand depending on the size selected. If the operation is word sized, bit [7] of the designated data register is copied to bits [15:8] of that data register. If the operation is long sized, bit [15] of the designated data register is copied to bits [31:16] of that data register.

Condition Codes:

X	N	Z	V	C
—	*	*	0	0

- N Set if the result is negative. Cleared otherwise.
- Z Set if the result is zero. Cleared otherwise.
- V Always cleared.
- C Always cleared.
- X Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	0	Op-Mode			0	0	0	Register		

Instruction Fields:

Op-Mode Field — Specifies the size of the sign-extension operation:

- 010 — Sign-extend low order byte of data register to word.
- 011 — Sign-extend low order word of data register to long.

Register field — Specifies the data register whose content is to be sign-extended.

ILLEGAL — Illegal Instruction — ILLEGAL

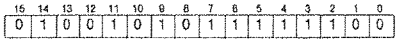
Operation: PC → −(SSP); SR → −(SSP)
(Illegal Instruction Vector) → PC

Attributes: None

Description: This bit pattern causes an illegal instruction exception. All other illegal instruction bit patterns are reserved for future extension of the instruction set.

Condition Codes: Not affected.

Instruction Format:



JMP — Jump — JMP

Operation: Destination → PC

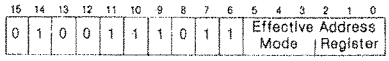
Assembler Syntax: JMP <ea>

Attributes: Unsized

Description: Program execution continues at the effective address specified by the instruction. The address is specified by the control addressing modes.

Condition Codes: Not affected.

Instruction Format:



Instruction Fields:
Effective Address field — Specifies the address of the next instruction. Only control addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	—	—	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An)+	—	—	d(PC)	111	010
−(An)	—	—	d(PC, Xi)	111	011
d(An)	101	register number	Imm	—	—

JSR — Jump to Subroutine — JSR

Operation: PC → −(SP); Destination → PC

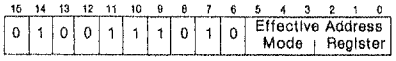
Assembler Syntax: JSR <ea>

Attributes: Unsized

Description: The long word address of the instruction immediately following the JSR instruction is pushed onto the system stack. Program execution then continues at the address specifed in the instruction.

Condition Codes: Not affected.

Instruction Format:



Instruction Fields:
Effective Address field — Specifies the address of the next instruction. Only control addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	—	—	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An)+	—	—	d(PC)	111	010
−(An)	—	—	d(PC, Xi)	111	011
d(An)	101	register number	Imm	—	—

LEA — Load Effective Address — LEA

Operation: Destination → An

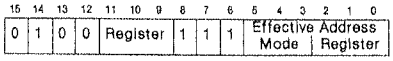
Assembler Syntax: LEA <ea>, An

Attributes: Size = (Long)

Description: The effective address is loaded into the specified address register. All 32 bits of the address register are affected by this instruction.

Condition Codes: Not affected.

Instruction Format:



Instruction Fields:
Register field — Specifies the address register which is to be loaded with the effective address.
Effective Address field — Specifies the address to be loaded into the address register. Only control addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	—	—	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An)+	—	—	d(PC)	111	010
−(An)	—	—	d(PC, Xi)	111	011
d(An)	101	register number	Imm	—	—

LINK — Link and Allocate

Operation: An → −(SP); SP → An; SP + d → SP

Assembler Syntax: LINK An, #<displacement>

Attributes: Unsized

Description: The current content of the specified address register is pushed onto the stack. After the push, the address register is loaded from the updated stack pointer. Finally, the 16-bit sign-extended displacement is added to the stack pointer. The content of the address register occupies two words on the stack. A negative displacement is specified to allocate stack area.

Condition Codes: Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	0	1	0	Register		
Displacement															

Instruction Fields:

Register field — Specifies the address register through which the link is to be constructed.

Displacement field — Specifies the twos complement integer which is to be added to the stack pointer.

Note: LINK and UNLK can be used to maintain a linked list of local data and parameter areas on the stack for nested subroutine calls.

LSL, LSR — Logical Shift

Operation: (Destination) Shifted by <count> → Destination

Assembler Syntax:
LSd Dx, Dy
LSd #<data>, Dy
LSd <ea>

Attributes: Size = (Byte, Word, Long)

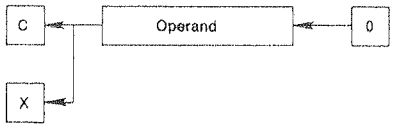
Description: Shift the bits of the operand in the direction specified. The carry bit receives the last bit shifted out of the operand. The shift count for the shifting of a register may be specified in two different ways:

1. Immediate — the shift count is specified in the instruction (shift range 1-8).
2. Register — the shift count is contained in a data register specified in the instruction.

The size of the operation may be specified to be byte, word, or long. The content of memory may be shifted one bit only and the operand size is restricted to a word.

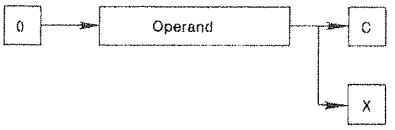
For LSL, the operand is shifted left; the number of positions shifted is the shift count. Bits shifted out of the high order bit go to both the carry and the extend bits; zeroes are shifted into the low order bit.

LSL:



For LSR, the operand is shifted right; the number of positions shifted is the shift count. Bits shifted out of the low order bit go to both the carry and the extend bits; zeroes are shifted into the high order bit.

LSR:



— Continued —

LSL, LSR — Logical Shift

Condition Codes:

X	N	Z	V	C
*	*	*	0	*

N Set if the result is negative. Cleared otherwise.
Z Set if the result is zero. Cleared otherwise.
V Always cleared.
C Set according to the last bit shifted out of the operand. Cleared for a shift count of zero.
X Set according to the last bit shifted out of the operand. Unaffected for a shift count of zero.

Instruction Format (Register Shifts):

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	Count/ Register			dr	Size		i/r	0	1	Register		

Instruction Fields (Register Shifts):

Count/Register field —
If i/r = 0, the shift count is specified in this field. The values 0, 1-7 represent a range of 8, 1 to 7 respectively.
If i/r = 1, the shift count (modulo 64) is contained in the data register specified in this field.

dr field — Specifies the direction of the shift:
0 — shift right.
1 — shift left.

Size field — Specifies the size of the operation:
00 — byte operation.
01 — word operation.
10 — long operation.

i/r field —
If i/r = 0, specifies immediate shift count.
If i/r = 1, specifies register shift count.

Register field — Specifies a data register whose content is to be shifted.

— Continued —

LSL, LSR — Logical Shift

Instruction Format (Memory Shifts):

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	0	0	1	dr	1	1	Effective Address Mode			Register		

Instruction Fields (Memory Shifts):

dr field — Specifies the direction of the shift:
0 — shift right.
1 — shift left.

Effective Address field — Specifies the operand to be shifted. Only memory alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	—	—	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An)+	011	register number	d(PC)	—	—
−(An)	100	register number	d(PC, Xi)	—	—
d(An)	101	register number	Imm	—	—

MOVE Move Data from Source to Destination MOVE

Operation: (Source) → Destination

Assembler Syntax: MOVE <ea>, <ea>

Attributes: Size = (Byte, Word, Long)

Description: Move the content of the source to the destination location. The data is examined as it is moved, and the condition codes set accordingly. The size of the operation may be specified to be byte, word, or long.

Condition Codes:

X	N	Z	V	C
—	*	*	0	0

N Set if the result is negative. Cleared otherwise.
Z Set if the result is zero. Cleared otherwise.
V Always cleared.
C Always cleared.
X Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	Size		Destination Register			Destination Mode			Source Mode			Source Register		

Instruction Fields:

Size field — Specifies the size of the operand to be moved:
01 — byte operation.
11 — word operation.
10 — long operation.

Destination Effective Address field — Specifies the destination location. Only data alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	—	—
– (An)	100	register number	d(PC, Xi)	—	—
d(An)	101	register number	Imm	—	—

— Continued —

MOVE Move Data from Source to Destination MOVE

Instruction Fields: (Continued)

Source Effective Address field — Specifies the source operand. All addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An*	001	register number	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	111	010
– (An)	100	register number	d(PC, Xi)	111	011
d(An)	101	register number	Imm	111	100

*For byte size operation, address register direct is not allowed.

Notes:
1. MOVEA is used when the destination is an address register. Most assemblers automatically make this distinction.
2. MOVEQ can also be used for certain operations on data registers.

MOVE from CCR Move from the Condition Code Register MOVE from CCR

Operation: CCR → Destination

Assembler Syntax: MOVE CCR, <ea>

Attributes: Size = (Word)

Description: The content of the status register is moved to the destination location. The source operand is a word, but only the low order byte contains the condition codes. The upper byte is all zeros.

Condition Codes: Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	1	0	1	1	Effective Address Mode			Effective Address Register		

Instruction Fields:

Effective Address field — Specifies the destination location. Only data alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	—	—
– (An)	100	register number	d(PC, Xi)	—	—
d(An)	101	register number	Imm	—	—

Note: MOVE to CCR is a word operation. AND, OR, and EOR to CCR are byte operations.

MOVE to CCR Move to Condition Codes MOVE to CCR

Operation: (Source) → CCR

Assembler Syntax: MOVE <ea>, CCR

Attributes: Size = (Word)

Description: The content of the source operand is moved to the condition codes. The source operand is a word, but only the low order byte is used to update the condition codes. The upper byte is ignored.

Condition Codes:

X	N	Z	V	C
*	*	*	*	*

N Set the same as bit 3 of the source operand.
Z Set the same as bit 2 of the source operand.
V Set the same as bit 1 of the source operand.
C Set the same as bit 0 of the source operand.
X Set the same as bit 4 of the source operand.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	1	0	0	1	1	Effective Address Mode			Effective Address Register		

Instruction Fields:

Effective Address field — Specifies the location of the source operand. Only data addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	111	010
– (An)	100	register number	d(PC, Xi)	111	011
d(An)	101	register number	Imm	111	100

Note: MOVE to CCR is a word operation. AND, OR, and EOR to CCR are byte operations.

MOVE to SR

Move to the Status Register (Privileged Instruction)

MOVE to SR

Operation: If supervisor state
then (Source)→SR
else TRAP

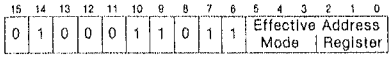
Assembler Syntax: MOVE <ea>, SR

Attributes: Size = (Word)

Description: The content of the source operand is moved to the status register. The source operand is a word and all bits of the status register are affected.

Condition Codes: Set according to the source operand.

Instruction Format:



Instruction Fields:
Effective Address field — Specifies the location of the source operand. Only data addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	111	010
– (An)	100	register number	d(PC, Xi)	111	011
d(An)	101	register number	Imm	111	100

MOVE from SR

Move from the Status Register

MOVE from SR

Operation: SR→Destination

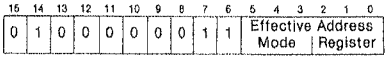
Assembler Syntax: MOVE SR, <ea>

Attributes: Size = (Word)

Description: The content of the status register is moved to the destination location. The operand size is a word.

Condition Codes: Not affected.

Instruction Format:



Instruction Fields:
Effective Address field — Specifies the destination location. Only data alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	—	—
– (An)	100	register number	d(PC, Xi)	—	—
d(An)	101	register number	Imm	—	—

Note: A memory destination is read before it is written to.

MOVE from SR

Move from the Status Register (Privileged Instruction)

MOVE from SR

Operation: If supervisor state
then SR→Destination
else TRAP

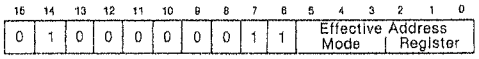
Assembler Syntax: MOVE SR, <ea>

Attributes: Size = (Word)

Description: The content of the status register is moved to the destination location. The operand size is a word.

Condition Codes: Not affected.

Instruction Format:



Instruction Fields:
Effective Address field — Specifies the destination location. Only data alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	—	—
– (An)	100	register number	d(PC, Xi)	—	—
d(An)	101	register number	Imm	—	—

NOTE: Use the MOVE from CCR instruction to access the conditon codes.

MOVE USP

Move User Stack Pointer (Privileged Instruction)

MOVE USP

Operation: If supervisor state
then USP→An;
An→USP
else TRAP

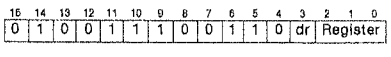
Assembler Syntax: MOVE USP, An
MOVE An, USP

Attributes: Size = (Long)

Description: The contents of the user stack pointer are transferred to or from the specified address register.

Condition Codes: Not affected.

Instruction Format:



Instruction Fields:
dr field — Specifies the direction of transfer:
0 — transfer the address register to the USP.
1 — transfer the USP to the address register.
Register field — Specifies the address register to or from which the user stack pointer is to be transferred.

MOVEA — Move Address

Operation: (Source) → Destination

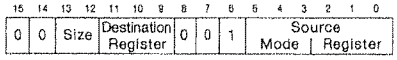
Assembler Syntax: MOVEA <ea>, An

Attributes: Size = (Word, Long)

Description: Move the content of the source to the destination address register. The size of the operation may be specified to be word or long. Word size source operands are sign extended to 32 bit quantities before the operation is done.

Condition Codes: Not affected.

Instruction Format:



Instruction Fields:

Size field — Specifies the size of the operand to be moved:
11 — Word operation. The source operand is sign-extended to a long operand and all 32 bits are loaded into the address register.
10 — Long operation.

Destination Register field — Specifies the destination address register.

Source Effective Address field — Specifies the location of the source operand. All addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An	001	register number	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	111	010
– (An)	100	register number	d(PC, Xi)	111	011
d(An)	101	register number	Imm	111	100

MOVEC — Move to/from Control Register (Privileged Instruction)

Operation: If supervisor state
then Rc → Rn, Rn → Rc
else TRAP

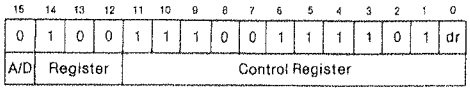
Assembler Syntax: MOVEC Rc, Rn
MOVEC Rn, Rc

Attributes: Size = (Long)

Description: Copy the contents of the specified control register to the specified general register or copy the contents of the specified general register to the specified control register. This is always a 32-bit transfer even though the control register may be implemented with fewer bits. Unimplemented bits are read as zeros.

Condition Codes: Not affected.

Instruction Format:



Instruction Fields:

dr field — Specifies the direction of the transfer:
0—control register to general register.
1—general register to control register.

A/D field — Specifies the type of general register:
0—data register.
1—address register.

Register field — Specifies the register number.

Control Register field — Specifies the control register.

Currently defined control registers are:

Binary	Hex	Name/Function
0000 0000 0000	000	Source Function Code (SFC) register.
0000 0000 0001	001	Destination Function Code (DFC) register.
1000 0000 0000	800	User Stack Pointer.
1000 0000 0001	801	Vector Base Register for exception vector table.

All other codes cause an illegal instruction exception.

MOVEM — Move Multiple Registers

Operation: Registers → Destination
(Source) → Registers

Assembler Syntax: MOVEM <register list>, <ea>
MOVEM <ea>, <register list>

Attributes: Size = (Word, Long)

Description: Selected registers are transferred to or from consecutive memory location starting at the location specified by the effective address. A register is transferred if the bit corresponding to that register is set in the mask field. The instruction selects how much of each register is transferred; either the entire long word can be moved or just the low order word. In the case of a word transfer to the registers, each word is sign-extended to 32 bits (also data registers) and the resulting long word loaded into the associated register.

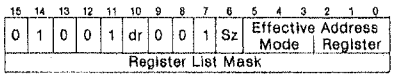
MOVEM allows three forms of address modes: the control modes, the predecrement mode, or the postincrement mode. If the effective address is in one of the control modes, the registers are transferred starting at the specified address and up through higher addresses. The order of transfer is from data register 0 to data register 7, then from address register 0 to address register 7.

If the effective address is in the predecrement mode, only a register to memory operation is allowed. The registers are stored starting at the specified address minus two and down through lower addresses. The order of storing is from address register 7 to address register 0, then from data register 7 to data register 0. The decremented address register is updated to contain the address of the last word stored.

If the effective address is in the postincrement mode, only a memory to register operation is allowed. The registers are loaded starting at the specified address and up through higher addresses. The order of loading is the same as for the control mode addressing. The incremented address register is updated to contain the address of the last word loaded plus two.

Condition Codes: Not affected.

Instruction Format:



— Continued —

MOVEM — Move Multiple Registers

Instruction Fields:

dr field:
Specifies the direction of the transfer:
0 — register to memory
1 — memory to register.

Sz field — Specifies the size of the registers being transferred:
0 — word transfer.
1 — long transfer.

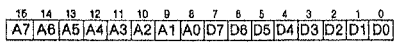
Effective Address field — Specifies the memory address to or from which the registers are to be moved.
For register to memory transfer, only control alterable addressing modes or the predecrement addressing mode are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	—	—	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	—	—	d(PC)	—	—
– (An)	100	register number	d(PC, Xi)	—	—
d(An)	101	register number	Imm	—	—

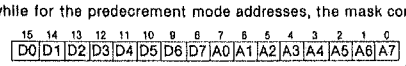
For memory to register transfer, only control addressing modes or the postincrement addressing mode are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	—	—	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	111	010
– (An)	—	—	d(PC, Xi)	111	011
d(An)	101	register number	Imm	—	—

Register List Mask field — Specifies which registers are to be transferred. The low order bit corresponds to the first register to be transferred; the high bit corresponds to the last register to be transferred. Thus, both for control modes and for the postincrement mode addresses, the mask correspondence is



while for the predecrement mode addresses, the mask correspondence is



Note: An extra read bus cycle occurs for memory operands. This amounts to a memory word at one address higher than expected being addressed during operation.

MOVEP Move Peripheral Data MOVEP

Operation: (Source) → Destination

Assembler Syntax: MOVEP Dx, d(Ay)
MOVEP d(Ay), Dx

Attributes: Size = (Word, Long)

Description: Data is transferred between a data register and alternate bytes of memory, starting at the location specified and incrementing by two. The high order byte of the data register is transferred first and the low order byte is transferred last. The memory address is specified using the address register indirect plus displacement addressing mode. If the address is even, all the transfers are made on the high order half of the data bus; if the address is odd, all the transfers are made on the low order half of the data bus.

Example: Long transfer to/from an even address.

Byte organization in register

31–24	23–16	15–8	7–0
hi-order	mid-upper	mid-lower	low-order

Byte organization in memory (low address at top)

15–8	7–0
hi-order	
mid-upper	
mid-lower	
low-order	

Example: Word transfer to/from an odd address.

Byte organization in register

31–24	23–16	15–8	7–0
		hi-order	low-order

Byte organization in memory (low address at top)

15–8	7–0
	hi-order
	low-order

Condition Codes: Not affected.

— Continued —

MOVEP Move Peripheral Data MOVEP

Instruction Format:

15	14	13	12	11–9	8–6	5	4	3	2–0
0	0	0	0	Data Register	Op-Mode	0	0	1	Address Register
Displacement									

Instruction Fields:

Data Register field — Specifies the data register to or from which the data is to be transferred.

Op-Mode field — Specifies the direction and size of the operation:
- 100 — transfer word from memory to register.
- 101 — transfer long from memory to register.
- 110 — transfer word from register to memory.
- 111 — transfer long from register to memory.

Address Register field — Specifies the address register which is used in the address register indirect plus displacement addressing mode.

Displacement field — Specifies the displacement which is used in calculating the operand address.

MOVEQ Move Quick MOVEQ

Operation: Immediate Data → Destination

Assembler Syntax: MOVEQ #<data>, Dn

Attributes: Size = (Long)

Description: Move immediate data to a data register. The data is contained in an 8-bit field within the operation word. The data is sign-extended to a long operand and all 32 bits are transferred to the data register.

Condition Codes:

X	N	Z	V	C
—	*	*	0	0

- N Set if the result is negative. Cleared otherwise.
- Z Set if the result is zero. Cleared otherwise.
- V Always cleared.
- C Always cleared.
- X Not affected.

Instruction Format:

15	14	13	12	11–9	8	7–0
0	1	1	1	Register	0	Data

Instruction Fields:

Register field — Specifies the data register to be loaded.

Data field — 8 bits of data which are sign extended to a long operand.

MOVES Move to/from Address Space (Privileged Instruction) MOVES

Operation: If supervisor state
then Rn → Destination <DFC>
Source <SFC> → Rn
else TRAP

Assembler Syntax: MOVES Rn, <ea>
MOVES <ea>, Rn

Attributes: Size = (Byte, Word, Long)

Description: Move the byte, word, or long operand from the specified general register to a location within the address space specified by the destination function code (DFC) register. Or, move the byte, word, or long operand from a location within the address space specified by the source function code (SFC) register to the specified general register.

If the destination is a data register, the source operand replaces the corresponding low-order bits of the that data register. If the destination is an address register, the source operand is sign-extended to 32 bits and then loaded into that address register.

Condition Codes: Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7–6	5–0
0	0	0	0	1	1	1	0	Size	Effective Address
A/D	Register			dr	0	0	0	0	0	0	0	0	0	0

Instruction Fields:

Size field — Specifies the size of the operation:
- 00—byte operation.
- 01—word operation.
- 10—long operation.

A/D field — Specifies the type of general register:
- 0—data register.
- 1—address register.

Register field — Specifies the register number.

dr field — Specifies the direction of the transfer:
- 0—from <ea> to general register.
- 1—from general register to <ea>.

—Continued—

MOVES Move to/from Address Space (Privileged Instruction) MOVES

Instruction Fields: (continued)

Effective Address field — Specifies the source or destination location within the alternate address space. Only alterable memory addressing modes are allowed as shown:

Addressing Mode	Mode	Register
Dn	—	—
An	—	—
(An)	010	register number
(An)+	011	register number
−(An)	100	register number
d(An)	101	register number

Addressing Mode	Mode	Register
d(An, Xi)	110	register number
Abs.W	111	000
Abs.L	111	001
d(PC)	—	—
d(PC, Xi)	—	—
Imm	—	—

MULS Signed Multiply MULS

Operation: (Source)*(Destination)→ Destination

Assembler Syntax: MULS <ea>, Dn

Attributes: Size = (Word)

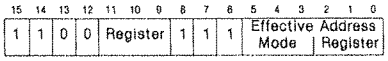
Description: Multiply two signed 16-bit operands yielding a 32-bit signed result. The operation is performed using signed arithmetic. A register operand is taken from the low order word; the upper word is unused. All 32 bits of the product are saved in the destination data register.

Condition Codes:

X	N	Z	V	C
—	*	*	0	0

- N Set if the result is negative. Cleared otherwise.
- Z Set if the result is zero. Cleared otherwise.
- V Always cleared.
- C Always cleared.
- X Not affected.

Instruction Format:



Instruction Fields:

Register field — Specifies one of the data registers. This field always specifies the destination.

Effective Address field — Specifies the source operand. Only data addressing modes are allowed as shown:

Addressing Mode	Mode	Register
Dn	000	register number
An	—	—
(An)	010	register number
(An)+	011	register number
−(An)	100	register number
d(An)	101	register number

Addressing Mode	Mode	Register
d(An, Xi)	110	register number
Abs.W	111	000
Abs.L	111	001
d(PC)	111	010
d(PC, Xi)	111	011
Imm	111	100

MULU Unsigned Multiply MULU

Operation: (Source)*(Destination)→ Destination

Assembler Syntax: MULU <ea>, Dn

Attributes: Size = (Word)

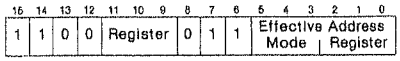
Description: Multiply two unsigned 16-bit operands yielding a 32-bit unsigned result. The operation is performed using unsigned arithmetic. A register operand is taken from the low order word; the upper word is unused. All 32 bits of the product are saved in the destination data register.

Condition Codes:

X	N	Z	V	C
—	*	*	0	0

- N Set if the most significant bit of the result is set. Cleared otherwise.
- Z Set if the result is zero. Cleared otherwise.
- V Always cleared.
- C Always cleared.
- X Not affected.

Instruction Format:



Instruction Fields:

Register field — Specifies one of the data registers. This field always specifies the destination.

Effective Address field — Specifies the source operand. Only data addressing modes are allowed as shown:

Addressing Mode	Mode	Register
Dn	000	register number
An	—	—
(An)	010	register number
(An)+	011	register number
−(An)	100	register number
d(An)	101	register number

Addressing Mode	Mode	Register
d(An, Xi)	110	register number
Abs.W	111	000
Abs.L	111	001
d(PC)	111	010
d(PC, Xi)	111	011
Imm	111	100

NBCD Negate Decimal with Extend NBCD

Operation: 0 − (Destination)$_{10}$ − X→ Destination

Assembler Syntax: NBCD <ea>

Attributes: Size = (Byte)

Description: The operand addressed as the destination and the extend bit are subtracted from zero. The operation is performed using decimal arithmetic. The result is saved in the destination location. This instruction produces the tens complement of the destination if the extend bit is clear, the nines complement if the extend bit is set. This is a byte operation only.

Condition Codes:

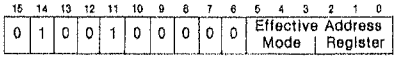
X	N	Z	V	C
*	U	*	U	*

- N Undefined.
- Z Cleared if the result is non-zero. Unchanged otherwise.
- V Undefined.
- C Set if a borrow (decimal) was generated. Cleared otherwise.
- X Set the same as the carry bit.

NOTE

Normally the Z condition code bit is set via programming before the start of an operation. This allows successful tests for zero results upon completion of multiple-precision operations.

Instruction Format:



Instruction Fields:

Effective Address field — Specifies the destination operand. Only data alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register
Dn	000	register number
An	—	—
(An)	010	register number
(An)+	011	register number
−(An)	100	register number
d(An)	101	register number

Addressing Mode	Mode	Register
d(An, Xi)	110	register number
Abs.W	111	000
Abs.L	111	001
d(PC)	—	—
d(PC, Xi)	—	—
Imm	—	—

NEG — Negate

Operation: 0 – (Destination) → Destination

Assembler Syntax: NEG <ea>

Attributes: Size = (Byte, Word, Long)

Description: The operand addressed as the destination is subtracted from zero. The result is stored in the destination location. The size of the operation may be specified to be byte, word, or long.

Condition Codes:

X	N	Z	V	C
*	*	*	*	*

- N Set if the result is negative. Cleared otherwise.
- Z Set if the result is zero. Cleared otherwise.
- V Set if an overflow is generated. Cleared otherwise.
- C Cleared if the result is zero. Set otherwise.
- X Set the same as the carry bit.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	1	0	0	Size		Effective Address Mode			Register		

Instruction Fields:

Size field — Specifies the size of the operation:
- 00 — byte operation.
- 01 — word operation.
- 10 — long operation.

Effective Address field — Specifies the destination operand. Only data alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	—	—
– (An)	100	register number	d(PC, Xi)	—	—
d(An)	101	register number	Imm	—	—

NEGX — Negate with Extend

Operation: 0 – (Destination) – X → Destination

Assembler Syntax: NEGX <ea>

Attributes: Size = (Byte, Word, Long)

Description: The operand addressed as the destination and the extend bit are subtracted from zero. The result is stored in the destination location. The size of the operation may be specified to be byte, word, or long.

Condition Codes:

X	N	Z	V	C
*	*	*	*	*

- N Set if the result is negative. Cleared otherwise.
- Z Cleared if the result is non-zero. Unchanged otherwise.
- V Set if an overflow is generated. Cleared otherwise.
- C Set if a borrow is generated. Cleared otherwise.
- X Set the same as the carry bit.

NOTE

Normally the Z condition code bit is set via programming before the start of an operation. This allows successful tests for zero results upon completion of multiple-precision operations.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	0	Size		Effective Address Mode			Register		

Instruction Fields:

Size field — Specifies the size of the operation:
- 00 — byte operation.
- 01 — word operation.
- 10 — long operation.

Effective Address field — Specifies the destination operand. Only data alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	—	—
– (An)	100	register number	d(PC, Xi)	—	—
d(An)	101	register number	Imm	—	—

NOP — No Operation

Operation: None

Assembler Syntax: NOP

Attributes: Unsized

Description: No operation occurs. The processor state, other than the program counter, is unaffected. Execution continues with the instruction following the NOP instruction.

Condition Codes: Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	0	0	1

NOT — Logical Complement

Operation: ~ (Destination) → Destination

Assembler Syntax: NOT <ea>

Attributes: Size = (Byte, Word, Long)

Description: The ones complement of the destination operand is taken and the result stored in the destination location. The size of the operation may be specified to be byte, word, or long.

Condition Codes:

X	N	Z	V	C
—	*	*	0	0

- N Set if the result is negative. Cleared otherwise.
- Z Set if the result is zero. Cleared otherwise.
- V Always cleared.
- C Always cleared.
- X Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	1	1	0	Size		Effective Address Mode			Register		

Instruction Fields:

Size field — Specifies the size of the operation:
- 00 — byte operation.
- 01 — word operation.
- 10 — long operation.

Effective Address field — Specifies the destination operand. Only data alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	—	—
– (An)	100	register number	d(PC, Xi)	—	—
d(An)	101	register number	Imm	—	—

OR — Inclusive OR Logical

Operation: (Source) v (Destination)→ Destination

Assembler Syntax: OR <ea>, Dn
OR Dn, <ea>

Attributes: Size = (Byte, Word, Long)

Description: Inclusive OR the source operand to the destination operand and store the result in the destination location. The size of the operation may be specified to be byte, word, or long. The contents of an address register may not be used as an operand.

Condition Codes:

X	N	Z	V	C
—	*	*	0	0

- N Set if the most significant bit of the result is set. Cleared otherwise.
- Z Set if the result is zero. Cleared otherwise.
- V Always cleared.
- C Always cleared.
- X Not affected.

Instruction Format:

15	14	13	12	11 10 9	8 7 6	5 4 3	2 1 0
1	0	0	0	Register	Op-Mode	Effective Address Mode	Effective Address Register

Instruction Fields:

Register field — Specifies any of the eight data registers.

Op-Mode field —

Byte	Word	Long	Operation
000	001	010	(<Dn>) v (<ea>)→ <Dn>
100	101	110	(<ea>) v (<Dn>)→ <ea>

Effective Address field —

If the location specified is a source operand then only data addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	111	010
– (An)	100	register number	d(PC, Xi)	111	011
d(An)	101	register number	Imm	111	100

— Continued —

Effective Address field (Continued)

If the location specified is a destination operand then only memory alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	—	—	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	—	—
– (An)	100	register number	d(PC, Xi)	—	—
d(An)	101	register number	Imm	—	—

Notes:

1. If the destination is a data register, then it cannot be specified by using the destination <ea> mode, but must use the destination Dn mode instead.
2. ORI is used when the source is immediate data. Most assemblers automatically make this distinction.

ORI — Inclusive OR Immediate

Operation: Immediate Data v (Destination)→ Destination

Assembler Syntax: ORI #<data>, <ea>

Attributes: Size = (Byte, Word, Long)

Description: Inclusive OR the immediate data to the destination operand and store the result in the destination location. The size of the operation may be specified to be byte, word, or long. The size of the immediate data matches the operation size.

Condition Codes:

X	N	Z	V	C
—	*	*	0	0

- N Set if the most significant bit of the result is set. Cleared otherwise.
- Z Set if the result is zero. Cleared otherwise.
- V Always cleared.
- C Always cleared.
- X Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7 6	5 4 3	2 1 0
0	0	0	0	0	0	0	0	Size	Effective Address Mode	Effective Address Register
Word Data (16 bites)									Byte Data (8 bits)	
Long Data (32 bits, including previous word)										

Instruction Fields:

Size field — Specifies the size of the operation:
- 00 — byte operation.
- 01 — word operation.
- 10 — long operation.

Effective Address field — Specifies the destination operand. Only data alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	—	—
– (An)	100	register number	d(PC, Xi)	—	—
d(An)	101	register number	Imm	—	—

Immediate field — (Data immediately following the instruction):
- If size = 00, then the data is the low order byte of the immediate word.
- If size = 01, then the data is the entire immediate word.
- If size = 10, then the data is the next two immediate words.

ORI to CCR — Inclusive OR Immediate to Condition Codes

Operation: (Source) v CCR→ CCR

Assembler Syntax: ORI #xxx, CCR

Attributes: Size = (Byte)

Description: Inclusive OR the immediate operand with the condition codes and store the result in the low-order byte of the status register.

Condition Codes:

X	N	Z	V	C
*	*	*	*	*

- N Set if bit 3 of immediate operand is one. Unchanged otherwise.
- Z Set if bit 2 of immediate operand is one. Unchanged otherwise.
- V Set if bit 1 of immediate operand is one. Unchanged otherwise.
- C Set if bit 0 of immediate operand is one. Unchanged otherwise.
- X Set if bit 4 of immediate operand is one. Unchanged otherwise.

Instruction Format:

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	1	1	1	1	0	0
0	0	0	0	0	0	0	0	Byte Data (8 bits)							

ORI to SR

Inclusive OR Immediate to the Status Register (Privileged Instruction)

ORI to SR

Operation: If supervisor state
then (Source) v SR → SR
else TRAP

Assembler Syntax: ORI #xxx, SR

Attributes: Size = (Word)

Description: Inclusive OR the immediate operand with the contents of the status register and store the result in the status register. All bits of the status register are affected.

Condition Codes:

X	N	Z	V	C
*	*	*	*	*

N Set if bit 3 of immediate operand is one. Unchanged otherwise.
Z Set if bit 2 of immediate operand is one. Unchanged otherwise.
V Set if bit 1 of immediate operand is one. Unchanged otherwise.
C Set if bit 0 of immediate operand is one. Unchanged otherwise.
X Set if bit 4 of immediate operand is one. Unchanged otherwise.

Instruction Format:

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	1	1	1	1	1	0	0
Word Data (16 bits)															

PEA

Push Effective Address

PEA

Operation: Destination → −(SP)

Assembler Syntax: PEA <ea>

Attributes: Size = (Long)

Description: The effective address is computed and pushed onto the stack. A long word address is pushed onto the stack.

Condition Codes: Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	0	0	0	1	Effective Address Mode			Register		

Instruction Fields:
Effective Address field — Specifies the address to be pushed onto the stack. Only control addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	—	—	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	—	—	d(PC)	111	010
− (An)	—	—	d(PC, Xi)	111	011
d(An)	101	register number	imm	—	—

RESET

Reset External Devices (Privileged Instruction)

RESET

Operation: If supervisor state
then Assert RESET Line
else TRAP

Assembler Syntax: RESET

Attributes: Unsized

Description: The reset line is asserted causing all external devices to be reset. The processor state, other than the program counter, is unaffected and execution continues with the next instruction.

Condition Codes: Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	0	0	0

ROL ROR

Rotate (without Extend)

ROL ROR

Operation: (Destination) Rotated by <count> → Destination

Assembler Syntax: ROd Dx, Dy
ROd #<data>, Dy
ROd <ea>

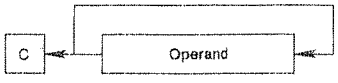
Attributes: Size = (Byte, Word, Long)

Description: Rotate the bits of the operand in the direction specified. The extend bit is not included in the rotation. The shift count for the rotation of a register may be specified in two different ways:
1. Immediate — the shift count is specified in the instruction (shift range, 1-8).
2. Register — the shift count is contained in a data register specified in the instruction.

The size of the operation may be specified to be byte, word, or long. The content of memory may be rotated one bit only and the operand size is restricted to a word.

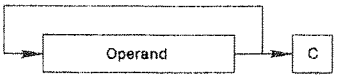
For ROL, the operand is rotated left; the number of positions shifted is the shift count. Bits shifted out of the high order bit go to both the carry bit and back into the low order bit. The extend bit is not modified or used.

ROL:



For ROR, the operand is rotated right; the number of position shifted is the shift count. Bits shifted out of the low order bit go to both the carry bit and back into the high order bit. The extend bit is not modified or used.

ROR:



Condition Codes:

X	N	Z	V	C
—	*	*	0	*

N Set if the most significant bit of the result is set. Cleared otherwise.
Z Set if the result is zero. Cleared otherwise.
V Always cleared.
C Set according to the last bit shfited out of the operand. Cleared for a shift count of zero.
X Not affected.

— Continued —

ROL ROR

Rotate (Without Extend)

ROL ROR

Instruction Format (Register Rotate):

15	14	13	12	11 10 9	8	7 6	5	4 3	2 1 0
1	1	1	0	Count/ Register	dr	Size	i/r	1 1	Register

Instruction Fields (Register Rotate):

Count/Register field —
If i/r = 0, the rotate count is specified in this field. The values 0, 1-7 represent a range of 8, 1 to 7 respectively.
If i/r = 1, the rotate count (modulo 64) is contained in the data register specified in this field.

dr field — Specifies the direction of the rotate:
0 — rotate right.
1 — rotate left.

Size field — Specifies the size of the operation:
00 — byte operation.
01 — word operation.
10 — long operation.

i/r field —
If i/r = 0, specifies immediate rotate count.
If i/r = 1, specifies register rotate count.

Register field — Specifies a data register whose content is to be rotated.

Instruction Format (Memory Rotate):

15	14	13	12	11	10	9	8	7	6	5 4 3	2 1 0
1	1	1	0	0	1	1	dr	1	1	Effective Address Mode	Register

Instruction Fields (Memory Rotate):

dr field — Specifies the direction of the rotate:
0 — rotate right
1 — rotate left.

Effective Address field — Specifies the operand to be rotated. Only memory alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	—	—	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	—	—
– (An)	100	register number	d(PC, Xi)	—	—
d(An)	101	register number	Imm	—	—

ROXL ROXR

Rotate with Extend

ROXL ROXR

Operation: (Destination) Rotated by <count> → Destination

Assembler Syntax:
ROXd Dx, Dy
ROXd #<data>, Dy
ROXd <ea>

Attributes: Size = (Byte, Word, Long)

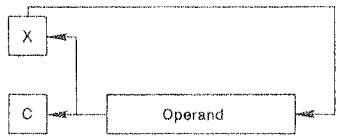
Description: Rotate the bits of the destination operand in the direction specified. The extend bit is included in the rotation. The shift count for the rotation of a register may be specified in two different ways:

1. Immediate — the shift count is specified in the instruction (shift range, 1-8).
2. Register — the shift count is contained in a data register specified in the instruction.

The size of the operation may be specified to be byte, word, or long. The content of memory may be rotated one bit only and the operand size is restricted to a word.

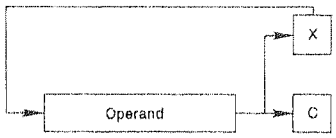
For ROXL, the operand is rotated left; the number of positions shifted is the shift count. Bits shifted out of the high order bit go to both the carry and extend bits; the previous value of the extend bit is shifted into the low order bit.

ROXL:



For ROXR, the operand is rotated right; the number of positions shifted is the shift count. Bits shifted out of the low order bit go to both the carry and extend bits; the previous value of the extend bit is shifted into the high order bit.

ROXR:



— Continued —

ROXL ROXR

Rotate with Extend

ROXL ROXR

Condition Codes:

X	N	Z	V	C
*	*	*	0	*

N Set if the most significant bit of the result is set. Cleared otherwise.
Z Set if the result is zero. Cleared otherwise.
V Always cleared.
C Set according to the last bit shifted out of the operand. Set to the value of the extend bit for a shift count of zero.
X Set according to the last bit shifted out of the operand. Unaffected for a shift count of zero.

Instruction Format (Register Rotate):

15	14	13	12	11 10 9	8	7 6	5	4 3	2 1 0
1	1	1	0	Count/ Register	dr	Size	i/r	1 0	Register

Instruction Fields (Register Rotate):

Count/Register field:
If i/r = 0, the rotate count is specified in this field. The values 0, 1-7 represent range of 8, 1 to 7 respectively.
If i/r = 1, the rotate count (modulo 64) is contained in the data register specified in this field.

dr field — Specifies the direction of the rotate:
0 — rotate right.
1 — rotate left.

Size field — Specifies the size of the operation:
00 — byte operation.
01 — word operation.
10 — long operation.

i/r field —
If i/r = 0, specifies immediate rotate count.
If i/r = 1, specifies register rotate count.

Register field — Specifies a data register whose content is to be rotated.

— Continued —

ROXL ROXR

Rotate with Extend

ROXL ROXR

Instruction Format (Memory Rotate):

15	14	13	12	11	10	9	8	7	6	5 4 3	2 1 0
1	1	1	0	0	1	0	dr	1	1	Effective Address Mode	Register

Instruction Fields (Memory Rotate):

dr field — Specifies the direction of the rotate:
0 — rotate right.
1 — rotate left.

Effective Address field — Specifies the operand to be rotated. Only memory alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	—	—	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	—	—
– (An)	100	register number	d(PC, Xi)	—	—
d(An)	101	register number	Imm	—	—

RTD

RTD

Return and Deallocate Parameters

Operation: (SP)+ → PC; SP + d → SP

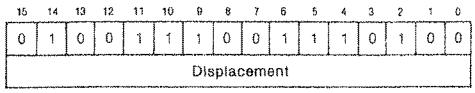
Assembler Syntax: RTD #<displacement>

Attributes: Unsized

Description: The program counter is pulled from the stack. The previous program counter value is lost. After the program counter is read from the stack, the displacement value is sign-extended to 32 bits and added to the stack pointer.

Condition Codes: Not affected.

Instruction Format:



Instruction Field:

Displacement field — Specifies the twos complement integer which is to be sign-extended and added to the stack pointer.

RTE

Return from Exception (Privileged Instruction)

RTE

Operation: If supervisor state
then (SP)+ → SR; (SP)+ → PC
else TRAP

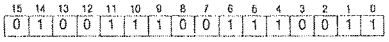
Assembler Syntax: RTE

Attributes: Unsized

Description: The status register and program counter are pulled from the system stack. The previous status register and program counter are lost. All bits in the status register are affected.

Condition Codes: Set according to the content of the word on the stack.

Instruction Format:



RTE

RTE

Return from Exception (Privileged Instruction)

Operation: If supervisor state
then (SP)+ → SR; (SP)+ → PC
If (SP)+ = long format
then full restore
else TRAP

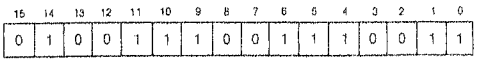
Assembler Syntax: RTE

Attributes: Unsized

Description: The status register and program counter are pulled from the system stack. The previous status register and program counter are lost. The vector offset word is also pulled from the stack and the format field is examined to determine the amount of information to be restored.

Condition Codes: Set according to the content of the word on the stack.

Instruction Format:



Vector Offset Word Format:

Vector Offset Word Format Fields:

Format Field: — Specifies the amount of information to be restored.
0000 — Short. Four words are to be removed from the top of the stack.
1000 — Long. Twenty-nine words are to be removed from the top of the stack.
Any Other
Pattern — Error. The processor takes the format error exception.

RTR

RTR

Return and Restore Condition Codes

Operation: (SP)+ → CC; (SP)+ → PC

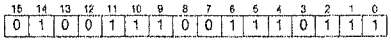
Assembler Syntax: RTR

Attributes: Unsized

Description: The condition codes and program counter are pulled from the stack. The previous condition codes and program counter are lost. The supervisor portion of the status register is unaffected.

Condition Codes: Set according to the content of the word on the stack.

Instruction Format:



RTS — Return from Subroutine

Operation: (SP)+ → PC

Assembler Syntax: RTS

Attributes: Unsized

Description: The program counter is pulled from the stack. The previous program counter is lost.

Condition Codes: Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	1	0	1

SBCD — Subtract Decimal with Extend

Operation: (Destination)$_{10}$ – (Source)$_{10}$ – X → Destination

Assembler Syntax: SBCD Dy, Dx
SBCD –(Ay), –(Ax)

Attributes: Size = (Byte)

Description: Subtract the source operand from the destination operand along with the extend bit and store the result in the destination location. The subtraction is performed using binary coded decimal arithmetic. The operands may be addressed in two different ways:

1. Data register to data register: The operands are contained in the data registers specified in the instruction.
2. Memory to memory: The operands are addressed with the predecrement addressing mode using the address registers specified in the instruction.

This operation is a byte operation only.

Condition Codes:

X	N	Z	V	C
*	U	*	U	*

N Undefined.
Z Cleared if the result is non-zero. Unchanged otherwise.
V Undefined.
C Set if a borrow (decimal) is generated. Cleared otherwise.
X Set the same as the carry bit.

NOTE

Normally the Z condition code bit is set via programming before the start of an operation. This allows successful tests for zero results upon completion of multiple-precision operations.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	Register Rx			1	0	0	0	0	R/M	Register Ry		

Instruction Fields:

Register Rx field — Specifies the destination register:
If R/M = 0, specifies a data register.
If R/M = 1, specifies an address register for the predecrement addressing mode.

R/M field — Specifies the operand addressing mode:
0 — The operation is data register to data register.
1 — The operation is memory to memory.

Register Ry field — Specifies the source register:
If R/M = 0, specifies a data register.
If R/M = 1, specifies an address register for the predecrement addressing mode.

Scc — Set According to Condition

Operation: If (Condition True)
then 1s → Destination
else 0s → Destination

Assembler Syntax: Scc < ea >

Attributes: Size = (Byte)

Description: The specified condition code is tested; if the condition is true, the byte specified by the effective address is set to TRUE (all ones), otherwise that byte is set to FALSE (all zeroes). "cc" may specify the following conditions:

CC	carry clear	0100	$\overline{C}$	LS	low or same	0011	$C + Z$
CS	carry set	0101	C	LT	less than	1101	$N \cdot \overline{V} + \overline{N} \cdot V$
EQ	equal	0111	Z	MI	minus	1011	N
F	false	0001	0	NE	not equal	0110	$\overline{Z}$
GE	greater or equal	1100	$N \cdot V + \overline{N} \cdot \overline{V}$	PL	plus	1010	$\overline{N}$
GT	greater than	1110	$N \cdot V \cdot \overline{Z} + \overline{N} \cdot \overline{V} \cdot \overline{Z}$	T	true	0000	1
HI	high	0010	$\overline{C} \cdot \overline{Z}$	VC	overflow clear	1000	$\overline{V}$
LE	less or equal	1111	$Z + N \cdot \overline{V} + \overline{N} \cdot V$	VS	overflow set	1001	V

Condition Codes: Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	Condition				1	1	Effective Address Mode			Register		

Instruction Fields:

Condition field — One of sixteen conditions discussed in description.
Effective Address field — Specifies the location in which the true/false byte is to be stored. Only data alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An)+	011	register number	d(PC)	—	—
–(An)	100	register number	d(PC, Xi)	—	—
d(An)	101	register number	Imm	—	—

Notes:
1. A memory destination is read before being written to.
2. An arithmetic one and zero result may be generated by following the Scc instruction with a NEG instruction.

STOP — Load Status Register and Stop (Privileged Instruction)

Operation: If supervisor state
then Immediate Data → SR; STOP
else TRAP

Assembler Syntax: STOP #xxx

Attributes: Unsized

Description: The immediate operand is moved into the entire status register; the program counter is advanced to point to the next instruction and the processor stops fetching and executing instructions. Execution of instructions resumes when a trace, interrupt, or reset exception occurs. A trace exception will occur if the trace state is on when the STOP instruction is executed. If an interrupt request arrives whose priority is higher than the current processor priority, an interrupt exception occurs, otherwise the interrupt request has no effect. If the bit of the immediate data corresponding to the S-bit is off, execution of the instruction will cause a privilege violation. External reset will always initiate reset exception processing.

Condition Codes: Set according to the immediate operand.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	0	1	0
Immediate Data															

Instruction Fields:

Immediate field — Specifies the data to be loaded into the status register.

SUB — Subtract Binary

Operation: (Destination) – (Source) → Destination

Assembler Syntax: SUB <ea>, Dn
SUB Dn, <ea>

Attributes: Size = (Byte, Word, Long)

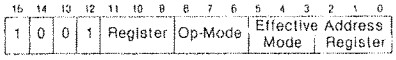
Description: Subtract the source operand from the destination operand and store the result in the destination. The size of the operation may be specified to be byte, word, or long. The mode of the instruction indicates which operand is the source and which is the destination as well as the operand size.

Condition Codes:

X	N	Z	V	C
*	*	*	*	*

N Set if the result is negative. Cleared otherwise.
Z Set if the result is zero. Cleared otherwise.
V Set if an overflow is generated. Cleared otherwise.
C Set if a borrow is generated. Cleared otherwise.
X Set the same as the carry bit.

Instruction Format:



Instruction Fields:

Register field — Specifies any of the eight data registers.

Op-Mode field —

Byte	Word	Long	Operation
000	001	010	(<Dn>) – (<ea>) → <Dn>
100	101	110	(<ea>) – (<Dn>) → <ea>

Effective Address field — Determines addressing mode:

If the location specified is a source operand, then all addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An*	001	register number	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	111	010
– (An)	100	register number	d(PC, Xi)	111	011
d(An)	101	register number	Imm	111	100

*For byte size operation, address register direct is not allowed.

— Continued —

SUB — Subtract Binary

Effective Address field (Continued)

If the location specified is a destination operand, then only alterable memory addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	—	—	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	—	—
– (An)	100	register number	d(PC, Xi)	—	—
d(An)	101	register number	Imm	—	—

Notes:
1. If the destination is a data register, then it cannot be specified by using the destination <ea> mode, but must use the destination Dn mode instead.
2. SUBA is used when the destination is an address register. SUBI and SUBQ are used when the source is immediate data. Most assemblers automatically make this distinction.

SUBA — Subtract Address

Operation: (Destination) – (Source) → Destination

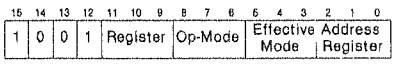
Assembler Syntax: SUBA <ea>, An

Attributes: Size = (Word, Long)

Description: Subtract the source operand from the destination address register and store the result in the address register. The size of the operation may be specified to be word or long. Word size source operands are sign extended to 32 bit quantities before the operation is done.

Condition Codes: Not affected.

Instruction Format:



Instruction Fields:

Register field — Specifies any of the eight address registers. This is always the destination.

Op-Mode field — Specifies the size of the operation:

011 — Word operation. The source operand is sign-extended to a long operand and the operation is performed on the address register using all 32 bits.

111 — Long operations.

Effective Address field — Specifies the source operand. All addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An	001	register number	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	111	010
– (An)	100	register number	d(PC, Xi)	111	011
d(An)	101	register number	Imm	111	100

SUBI — Subtract Immediate

Operation: (Destination) – Immediate Data → Destination

Assembler Syntax: SUBI #<data>, <ea>

Attributes: Size = (Byte, Word, Long)

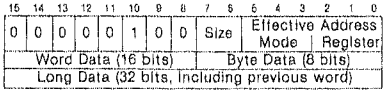
Description: Subtract the immediate data from the destination operand and store the result in the destination location. The size of the operation may be specified to be byte, word, or long. The size of the immediate data matches the operation size.

Condition Codes:

X	N	Z	V	C
*	*	*	*	*

N Set if the result is negative. Cleared otherwise.
Z Set if the result is zero. Cleared otherwise.
V Set if an overflow is generated. Cleared otherwise.
C Set if a borrow is generated. Cleared otherwise.
X Set the same as the carry bit.

Instruction Format:



Instruction Fields:

Size field — Specifies the size of the operation.

00 — byte operation.
01 — word operation.
10 — long operation.

Effective Address field — Specifies the destination operand. Only data alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	—	—
– (An)	100	register number	d(PC, Xi)	—	—
d(An)	101	register number	Imm	—	—

Immediate field — (Data immediately following the instruction)

If size = 00, then the data is the low order byte of the immediate word.
If size = 01, then the data is the entire immediate word.
If size = 10, then the data is the next two immediate words.

SUBQ Subtract Quick SUBQ

Operation: (Destination) – Immediate Data → Destination

Assembler Syntax: SUBQ #<data>, <ea>

Attributes: Size = (Byte, Word, Long)

Description: Subtract the immediate data from the destination operand. The data range is from 1-8. The size of the operation may be specified to be byte, word, or long. Word and long operations are also allowed on the address registers and the condition codes are not affected. Word size source operands are sign extended to 32 bit quantities before the operation is done.

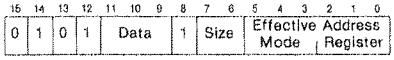
Condition Codes:

X	N	Z	V	C
*	*	*	*	*

- N Set if the result is negative. Cleared otherwise.
- Z Set if the result is zero. Cleared otherwise.
- V Set if an overflow is generated. Cleared otherwise.
- C Set if a borrow is generated. Cleared otherwise.
- X Set the same as the carry bit.

The condition codes are not affected if a subtraction from an address register is made.

Instruction Format:



Instruction Fields:

Data field — Three bits of immediate data, 0, 1-7 representing a range of 8, 1 to 7 respectively.

Size field — Specifies the size of the operation:
- 00 — byte operation.
- 01 — word operation.
- 10 — long operation.

Effective Address field — Specifies the destination location. Only alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An*	001	register number	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An)+	011	register number	d(PC)	—	—
–(An)	100	register number	d(PC, Xi)	—	—
d(An)	101	register number	Imm	—	—

*Word and Long only.

SUBX Subtract with Extend SUBX

Operation: (Destination) – (Source) – X → Destination

Assembler Syntax: SUBX Dy, Dx
SUBX –(Ay), –(Ax)

Attributes: Size = (Byte, Word, Long)

Description: Subtract the source operand from the destination operand along with the extend bit and store the result in the destination location. The operands may be addressed in two different ways:
1. Data register to data register: The operands are contained in data registers specified in the instruction.
2. Memory to memory. The operands are contained in memory and addressed with the predecrement addressing mode using the address registers specified in the instruction.

The size of the operation may be specified to be byte, word, or long.

Condition Codes:

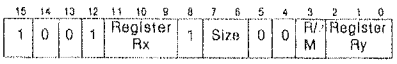
X	N	Z	V	C
*	*	*	*	*

- N Set if the result is negative. Cleared otherwise.
- Z Cleared if the result is non-zero. Unchanged otherwise.
- V Set if an overflow is generated. Cleared otherwise.
- C Set if a carry is generated. Cleared otherwise.
- X Set the same as the carry bit.

NOTE

Normally the Z condition code bit is set via programming before the start of an operation. This allows successful tests for zero results upon completion of multiple-precision operations.

Instruction Format:



— Continued —

SUBX Subtract with Extend SUBX

Instruction Fields:

Register Rx field — Specifies the destination register:
- If R/M = 0, specifies a data register.
- If R/M = 1, specifies an address register for the predecrement addressing mode.

Size field — Specifies the size of the operation:
- 00 — byte operation.
- 01 — word operation.
- 10 — long operation.

R/M field — Specifies the operand addressing mode:
- 0 — The operation is data register to data register.
- 1 — The operation is memory to memory.

Register Ry field — Specifies the source register:
- If R/M = 0, specifies a data register.
- If R/M = 1, specifies an address register for the predecrement addressing mode.

SWAP Swap Register Halves SWAP

Operation: Register [31:16] ↔ Register [15:0]

Assembler Syntax: SWAP Dn

Attributes: Size = (Word)

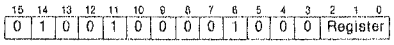
Description: Exchange the 16-bit halves of a data register.

Condition Codes:

X	N	Z	V	C
—	*	*	0	0

- N Set if the most significant bit of the 32-bit result is set. Cleared otherwise.
- Z Set if the 32-bit result is zero. Cleared otherwise.
- V Always cleared.
- C Always cleared.
- X Not affected.

Instruction Format:



Instruction Fields:

Register field — Specifies the data register to swap.

TAS

Test and Set an Operand

TAS

Operation: (Destination) Tested → CC; 1 → bit 7 OF Destination

Assembler Syntax: TAS <ea>

Attributes: Size = (Byte)

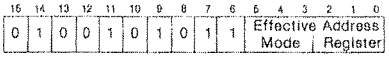
Description: Test and set the byte operand addressed by the effective address field. The current value of the operand is tested and N and Z are set accordingly. The high order bit of the operand is set. The operation is indivisible (using a read-modify-write memory cycle) to allow synchronization of several processors.

Condition Codes:

X	N	Z	V	C
—	*	*	0	0

- N Set if the most significant bit of the operand was set. Cleared otherwise.
- Z Set if the operand was zero. Cleared otherwise.
- V Always cleared.
- C Always cleared.
- X Not affected.

Instruction Format:



Instruction Fields:

Effective Address field — Specifies the location of the tested operand. Only data alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	—	—
– (An)	100	register number	d(PC, Xi)	—	—
d(An)	101	register number	Imm	—	—

Note: Bus error retry is inhibited on the read portion of the TAS read-modify-write bus cycle to ensure system integrity. The bus error exception is always taken.

TRAP

Trap

TRAP

Operation: PC → – (SSP); SR → – (SSP); (Vector) → PC

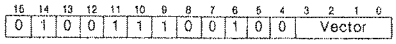
Assembler Syntax: TRAP #<vector>

Attributes: Unsized

Description: The processor initiates exception processing. The vector number is generated to reference the TRAP instruction exception vector specified by the low order four bits of the instruction. Sixteen TRAP instruction vectors are available.

Condition Codes: Not affected.

Instruction Format:



Instruction Fields:

Vector field — Specifies which trap vector contains the new program counter to be loaded.

TRAPV

Trap on Overflow

TRAPV

Operation: If V then TRAP

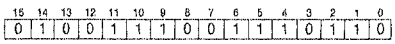
Assembler Syntax: TRAPV

Attributes: Unsized

Description: If the overflow condition is on, the processor initiates exception processing. The vector number is generated to reference the TRAPV exception vector. If the overflow condition is off, no operation is performed and execution continues with the next instruction in sequence.

Condition Codes: Not affected.

Instruction Format:



TST

Test an Operand

TST

Operation: (Destination) Tested → CC

Assembler Syntax: TST <ea>

Attributes: Size = (Byte, Word, Long)

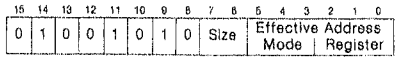
Description: Compare the operand with zero. No results are saved; however, the condition codes are set according to results of the test. The size of the operation may be specified to be byte, word, or long.

Condition Codes:

X	N	Z	V	C
—	*	*	0	0

- N Set if the operand is negative. Cleared otherwise.
- Z Set if the operand is zero. Cleared otherwise.
- V Always cleared.
- C Always cleared.
- X Not affected.

Instruction Format:



Instruction Fields:

Size field — Specifies the size of the operation:
- 00 — byte operation.
- 01 — word operation.
- 10 — long operation.

Effective Address field — Specifies the destination operand. Only data alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	register number	d(An, Xi)	110	register number
An	—	—	Abs.W	111	000
(An)	010	register number	Abs.L	111	001
(An) +	011	register number	d(PC)	—	—
– (An)	100	register number	d(PC, Xi)	—	—
d(An)	101	register number	Imm	—	—

UNLK — Unlink — UNLK

Operation: An → SP; (SP) + → An

Assembler Syntax: UNLK An

Attributes: Unsized

Description: The stack pointer is loaded from the specified address register. The address register is then loaded with the long word pulled from the top of the stack.

Condition Codes: Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	0	1	1	Register		

Instruction Fields:

Register field — specifies the address register through which the unlinking is to be done.

APPENDIX A
CONDITION CODES

CONDITION CODE REGISTER

The condition code register portion of the status register contains five bits:

N — Negative
Z — Zero
V — Overflow
C — Carry
X — Extend

The first four bits are true condition code bits in that they reflect the condition of the result of a processor operation. The X bit is an operand for multiprecision computations. The carry bit (C) and the multiprecision operand extend bit (X) are separate in the MC68000 to simplify the programming model.

CONDITION CODE REGISTER NOTATION

The description of the effect on the condition codes is given in the following form:

	X	N	Z	V	C
Condition Codes:					

where:

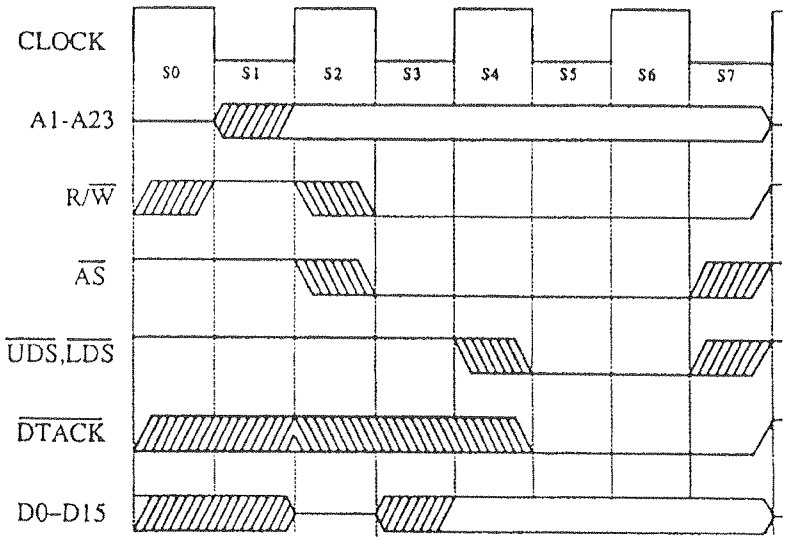
N (negative)	Set if the most significant bit of the result is set. Cleared otherwise.
Z (zero)	Set if the result equals zero. Cleared otherwise.
V (overflow)	Set if there was an arithmetic overflow. This implies that the result is not representable in the operand size. Cleared otherwise.
C (carry)	Set if a carry is generated out of the most significant bit of the operands for an addition. Also set if a borrow is generated in a subtraction. Cleared otherwise.
X (extend)	Transparent to data movement. When affected, it is set the same as the C bit.

The notational convention that appears in the representation of the condition code register is:

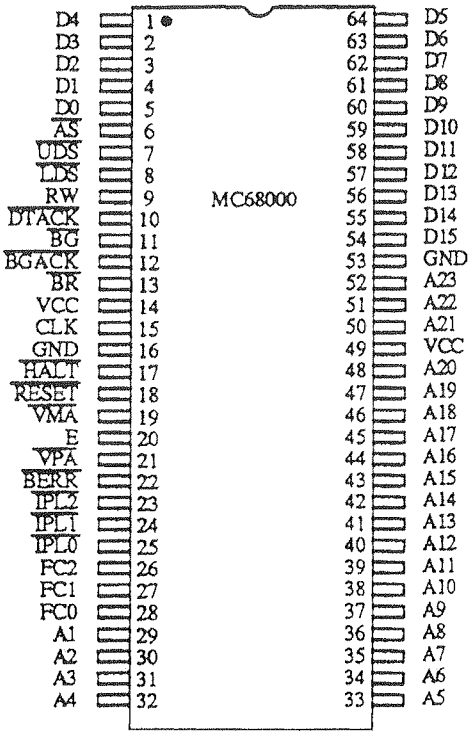
* set according to the result of the operation
— not affected by the operation
0 cleared
1 set
U undefined after the operation

MC68000 – WRITE BUS CYCLE OPERATION (WORD)

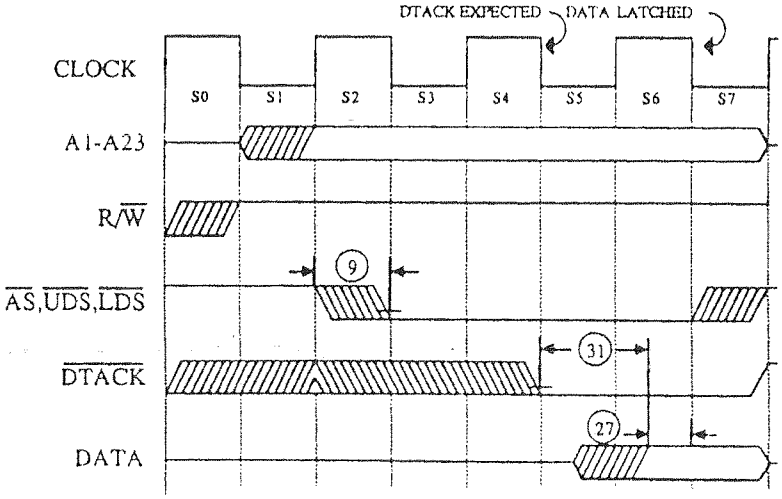
(4 CLOCK CYCLES)



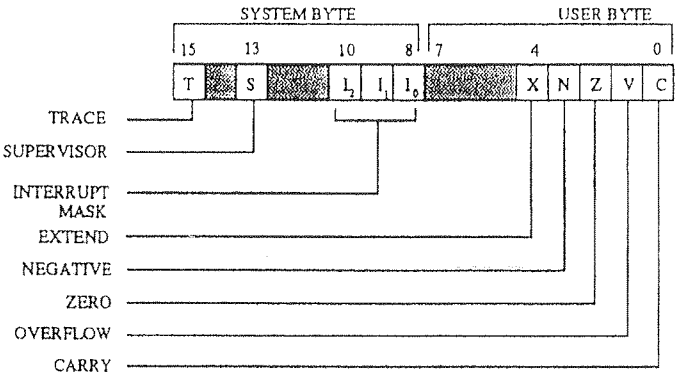
MC68000 – FOOTPRINT



MC68000 – READ BUS CYCLE OPERATION (WORD)

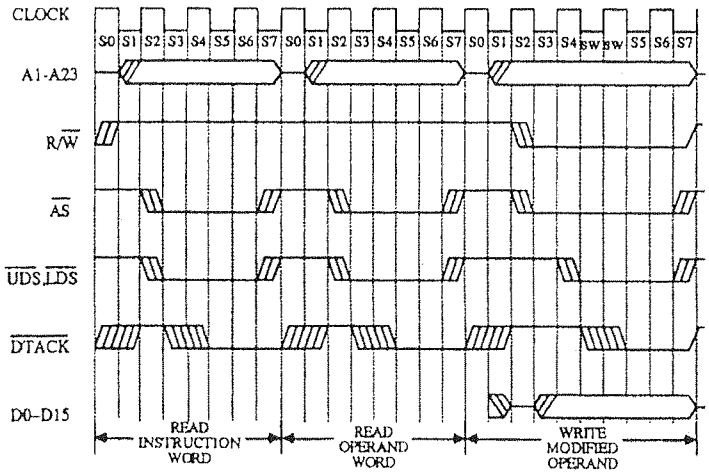


MC68000 STATUS REGISTER



UNUSED BITS ARE ZERO

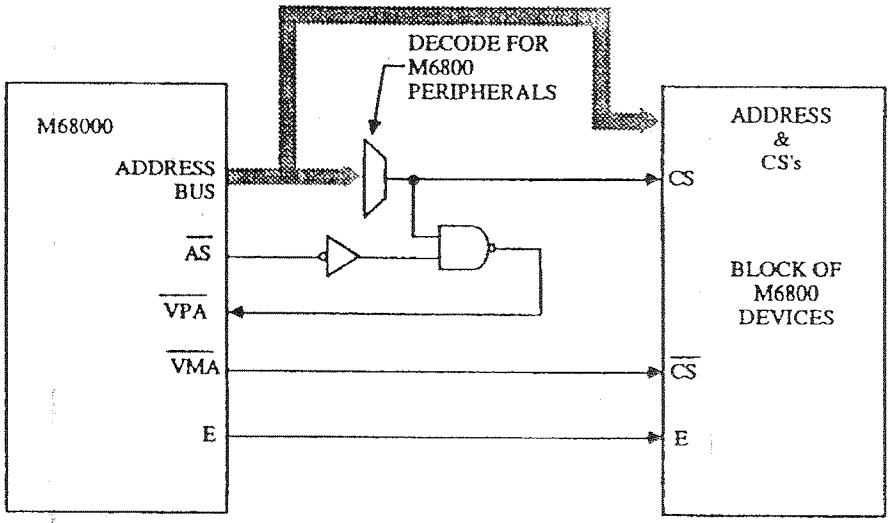
MC68000 – READ/MODIFY/WRITE INSTRUCTION (WORD)



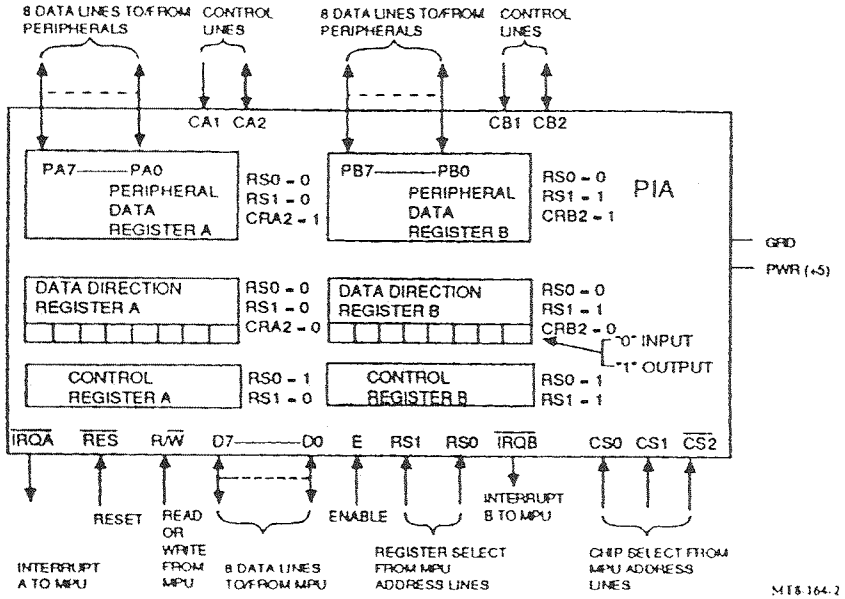
MC68000 – FUNCTION CONTROL STATES

FC2	FC1	FC0	STATE	MODE
0	0	0	RESERVED – MOTOROLA	USER
0	0	1	DATA SPACE	USER
0	1	0	PROGRAM SPACE	USER
0	1	1	RESERVED – USER	USER
1	0	0	RESERVED – MOTOROLA	SUPERVISOR
1	0	1	DATA SPACE	SUPERVISOR
1	1	0	PROGRAM SPACE	SUPERVISOR
1	1	1	INTERRUPT ACKNOWLEDGE	SUPERVISOR

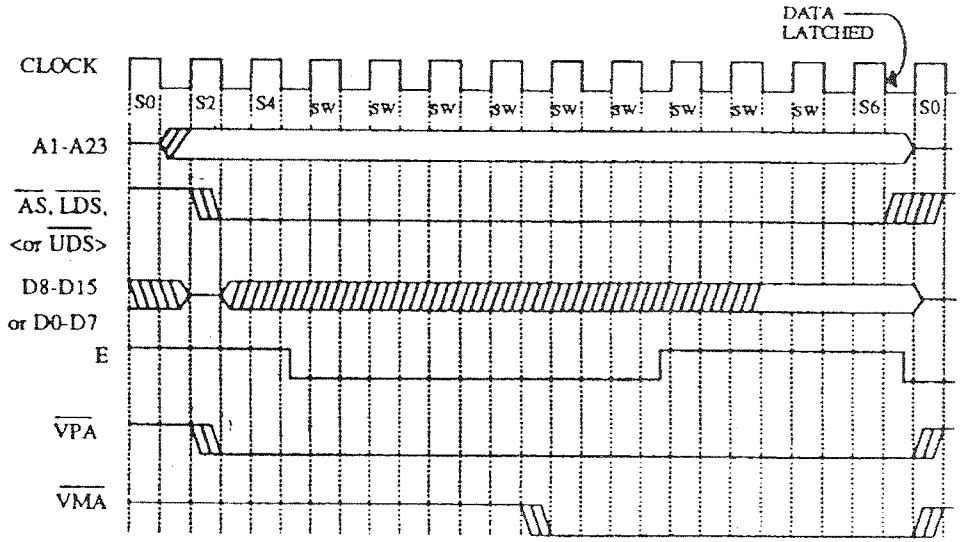
MC68000 – CONNECTION OF 6800 PERIPHERALS

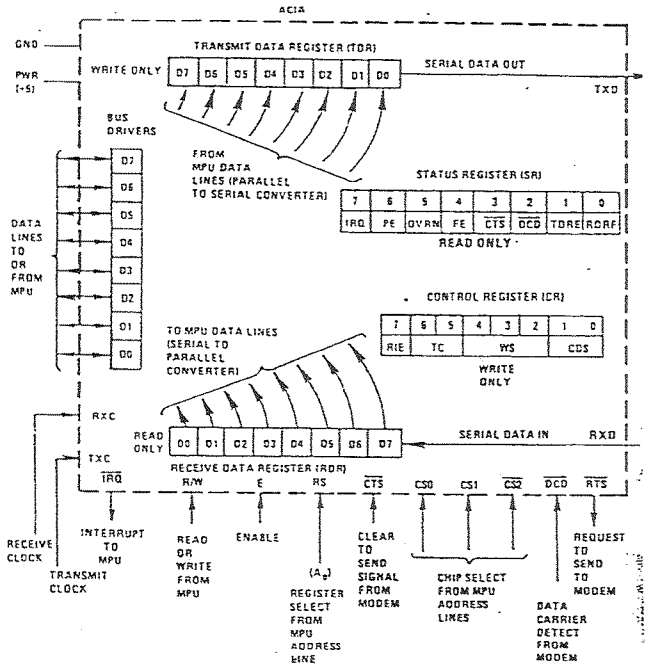


MC6821 – PIA DIAGRAM



MC68000 – 6800 PERIPHERAL CYCLE (READ)





ACIA CONTROL REGISTER FORMAT

B7 RIE	B6 TC2	B5 TC1	B4 WS3	B3 WS2	B2 WS1	B1 C2	B0 C1

B1	B0	FUNCTION (Tx. Rx)	MAX DATA CLOCK RATE
0	0	÷ 1	500 KHz
0	1	÷ 16	800 KHz
1	0	÷ 64	800 KHz
1	1	MASTER RESET	

B4	B3	B2	WORD LENGTH	+ PARITY	+ STOP BITS
0	0	0	7	EVEN	2
0	0	1	7	ODD	2
0	1	0	7	EVEN	1
0	1	1	7	ODD	1
1	0	0	8	NONE	2
1	0	1	8	NONE	1
1	1	0	8	EVEN	1
1	1	1	8	ODD	1

BITS CR5 AND CR6 HAVE THE FOLLOWING SYSTEM APPLICATION:

CR6	CR5	
0	0	THE $\overline{\text{RTS}}$ PIN IS LOW AND TRANSMIT INTERRUPTS ARE INHIBITED. THIS IS THE CODE USED WHEN REQUESTING THAT THE COMMUNICATIONS CHANNEL BE SET-UP. IT IS NOT CLEAR TO SEND DATA YET.
0	1	THE $\overline{\text{RTS}}$ PIN IS LOW AND THE COMMUNICATIONS CHANNEL HAS BEEN SET UP. THEREFORE, THIS CODE IS USED TO GENERATE IRQ'S VIA THE TDRE BIT IN THE STATUS REGISTER.
1	0	THE $\overline{\text{RTS}}$ PIN IS HIGH AND TRANSMIT INTERRUPTS ARE INHIBITED. THIS CODE CAN BE USED TO "KNOCK-DOWN" THE COMMUNICATIONS CHANNEL.
1	1	THE $\overline{\text{RTS}}$ PIN IS LOW (KEEP UP COMMUNICATIONS CHANNEL), A BREAK SIGNAL (LOW LEVEL ON TRANSMIT DATA OUT LINE) IS TRANSMITTED. THIS IS USED TO INTERRUPT THE REMOTE SYSTEM.

BIT 7 — RECEIVER INTERRUPT ENABLE (RIE)

"1" – ENABLES INTERRUPTS CAUSED BY

A) RECEIVER DATA REGISTER FULL GOING HIGH

B) A LOW TO HIGH TRANSITION ON THE DATA CARRIER DETECT SIGNAL LINE

"0" – INHIBITS INTERRUPTS DUE TO RECEIVE DATA REGISTER FULL OR LOSS OF RECEIVE DATA CARRIER.

FIGURE 18 – CONTROL WORD FORMAT