

# ABCD

## Add Decimal with Extend (M68000 Family)

# ABCD

**Operation:** Source10 + Destination10 + X → Destination

**Assembler** ABCD Dy,Dx

**Syntax:** ABCD – (Ay), – (Ax)

**Attributes:** Size = (Byte)

**Description:** Adds the source operand to the destination operand along with the extend bit, and stores the result in the destination location. The addition is performed using binary-coded decimal arithmetic. The operands, which are packed binary-coded decimal numbers, can 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	*

X — Set the same as the carry bit.

N — Undefined.

Z — Cleared if the result is nonzero; unchanged otherwise.

V — Undefined.

C — Set if a decimal carry was generated; cleared otherwise.

### 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.

# ABCD

## Add Decimal with Extend (M68000 Family)

# ABCD

### 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	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.

# ADD

## Add (M68000 Family)

# ADD

**Operation:** Source + Destination → Destination

**Assembler** ADD < ea > ,Dn

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

**Attributes:** Size = (Byte, Word, Long)

**Description:** Adds the source operand to the destination operand using binary addition and stores the result in the destination location. The size of the operation may be specified as 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
*	*	*	*	*

X — Set the same as the carry bit.

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.

### Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	REGISTER			OPMODE			EFFECTIVE ADDRESS					
											MODE		REGISTER		

# ADD

## Add (M68000 Family)

# ADD

### Instruction Fields:

Register field—Specifies any of the eight data registers.

Opmode field

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

Effective Address field—Determines addressing mode.

- a. If the location specified is a source operand, all addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	reg. number:Dn	(xxx).W	111	000
An*	001	reg. number:An	(xxx).L	111	001
(An)	010	reg. number:An	#<data>	111	100
(An) +	011	reg. number:An			
– (An)	100	reg. number:An			
(d <sub>16</sub> ,An)	101	reg. number:An	(d <sub>16</sub> ,PC)	111	010
(d <sub>8</sub> ,An,Xn)	110	reg. number:An	(d <sub>8</sub> ,PC,Xn)	111	011

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)**	110	reg. number:An	(bd,PC,Xn)†	111	011
([bd,An,Xn],od)	110	reg. number:An	([bd,PC,Xn],od)	111	011
([bd,An],Xn,od)	110	reg. number:An	([bd,PC],Xn,od)	111	011

\*Word and long only

\*\*Can be used with CPU32.

# ADD

## Add (M68000 Family)

# ADD

- b. If the location specified is a destination operand, only memory alterable addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	—	—
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d <sub>16</sub> ,PC)	—	—
(d <sub>8</sub> ,PC,Xn)	—	—

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	—	—
([bd,PC,Xn],od)	—	—
([bd,PC],Xn,od)	—	—

\*Can be used with CPU32

### NOTE

The Dn mode is used when the destination is a data register; the destination < ea > mode is invalid for a data register.

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 (M68000 Family)

# ADDA

**Operation:** Source + Destination → Destination

**Assembler**

**Syntax:** ADDA < ea > , An

**Attributes:** Size = (Word, Long)

**Description:** Adds the source operand to the destination address register and stores the result in the address register. The size of the operation may be specified as word or long. The entire destination address register is used regardless of the operation size.

**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	1	REGISTER				OPMODE			EFFECTIVE ADDRESS					
											MODE		REGISTER			

**Instruction Fields:**

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

Opmode 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.

# ADDA

## Add Address (M68000 Family)

# ADDA

Effective Address field—Specifies the source operand. All addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	001	reg. number:An
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	111	100
(d <sub>16</sub> ,PC)	111	010
(d <sub>8</sub> ,PC,Xn)	111	011

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

\*Can be used with CPU32

# ADDI

## Add Immediate (M68000 Family)

# ADDI

**Operation:** Immediate Data + Destination → Destination

**Assembler**

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

**Attributes:** Size = (Byte, Word, Long)

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

**Condition Codes:**

X	N	Z	V	C
*	*	*	*	*

- X — Set the same as the carry bit.
- 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.

**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			
16-BIT WORD DATA										8-BIT BYTE DATA					
32-BIT LONG DATA															



# ADDI

## Add Immediate (M68000 Family)

# ADDI

### 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 can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d <sub>16</sub> ,PC)	—	—
(d <sub>8</sub> ,PC,Xn)	—	—

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	—	—
([bd,PC,Xn],od)	—	—
([bd,PC],Xn,od)	—	—

\*Can be used with CPU32

Immediate field—Data immediately following the instruction.

If size = 00, the data is the low-order byte of the immediate word.

If size = 01, the data is the entire immediate word.

If size = 10, the data is the next two immediate words.

# ADDQ

## Add Quick (M68000 Family)

# ADDQ

**Operation:** Immediate Data + Destination → Destination

**Assembler**

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

**Attributes:** Size = (Byte, Word, Long)

**Description:** Adds an immediate value of one to eight to the operand at the destination location. The size of the operation may be specified as byte, word, or long. Word and long operations are also allowed on the address registers. When adding to address registers, the condition codes are not altered, and the entire destination address register is used regardless of the operation size.

**Condition Codes:**

X	N	Z	V	C
*	*	*	*	*

- X — Set the same as the carry bit.
- N — Set if the result is negative; cleared otherwise.
- Z — Set if the result is zero; cleared otherwise.
- V — Set if an overflow occurs; cleared otherwise.
- C — Set if a carry occurs; cleared otherwise.

The condition codes are not affected when the destination is an address register.

**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		

# ADDQ

## Add Quick (M68000 Family)

# ADDQ

### Instruction Fields:

**Data field**—Three bits of immediate data representing eight values (0 – 7), with the immediate value zero representing a value of eight.

**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 can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	001	reg. number:An
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d <sub>16</sub> ,PC)	—	—
(d <sub>8</sub> ,PC,Xn)	—	—

### MC68020, MC68030, and MC68040 only

(bd,An,Xn**)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)†	—	—
([bd,PC,Xn],od)	—	—
([bd,PC],Xn,od)	—	—

\*Word and long only.

\*\*Can be used with CPU32.

# ADDX

## Add Extended (M68000 Family)

# ADDX

**Operation:** Source + Destination + X → Destination

**Assembler** ADDX Dy,Dx

**Syntax:** ADDX – (Ay), – (Ax)

**Attributes:** Size = (Byte, Word, Long)

**Description:** Adds the source operand and the extend bit to the destination operand and stores the result in the destination location. The operands can be addressed in two different ways:

1. Data register to data register—The data registers specified in the instruction contain the operands.
2. Memory to memory—The address registers specified in the instruction address the operands using the predecrement addressing mode.

The size of the operation can be specified as byte, word, or long.

### Condition Codes:

X	N	Z	V	C
*	*	*	*	*

X — Set the same as the carry bit.

N — Set if the result is negative; cleared otherwise.

Z — Cleared if the result is nonzero; unchanged otherwise.

V — Set if an overflow occurs; cleared otherwise.

C — Set if a carry is generated; cleared otherwise.

### 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.

# ADDX

## Add Extended (M68000 Family)

# ADDX

### 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

R/M field—Specifies the operand address 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 (M68000 Family)

# AND

**Operation:** Source L Destination → Destination

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

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

**Attributes:** Size = (Byte, Word, Long)

**Description:** Performs an AND operation of the source operand with the destination operand and stores the result in the destination location. The size of the operation can be specified as 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

X — Not affected.

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.

### Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	REGISTER			OPMODE			EFFECTIVE ADDRESS MODE   REGISTER					

### Instruction Fields:

Register field—Specifies any of the eight data registers.

Opmode field

Byte	Word	Long	Operation
000	001	010	< ea > $\wedge$ Dn → Dn
100	101	110	Dn $\wedge$ < ea > → < ea >

# AND

## AND Logical (M68000 Family)

# AND

Effective Address field—Determines addressing mode.

- a. If the location specified is a source operand, only data addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	111	100
(d <sub>16</sub> ,PC)	111	010
(d <sub>8</sub> ,PC,Xn)	111	011

### MC68020, MC68030, and MC68040 only

(bd,An,Xn*	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

\*Can be used with CPU32.

# AND

## AND Logical (M68000 Family)

# AND

- b. If the location specified is a destination operand, only memory alterable addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	—	—
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d <sub>16</sub> ,PC)	—	—
(d <sub>8</sub> ,PC,Xn)	—	—

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	—	—
([bd,PC,Xn],od)	—	—
([bd,PC],Xn,od)	—	—

\*Can be used with CPU32.

### NOTE

The Dn mode is used when the destination is a data register; the destination < ea > mode is invalid for a data register.

Most assemblers use ANDI when the source is immediate data.



# ANDI

## AND Immediate (M68000 Family)

# ANDI

**Operation:** Immediate Data  $\wedge$  Destination  $\rightarrow$  Destination

**Assembler**

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

**Attributes:** Size = (Byte, Word, Long)

**Description:** Performs an AND operation of the immediate data with the destination operand and stores the result in the destination location. The size of the operation can be specified as byte, word, or long. The size of the immediate data matches the operation size.

**Condition Codes:**

X	N	Z	V	C
—	*	*	0	0

- X — Not affected.
- 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.

**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	SIZE		EFFECTIVE ADDRESS					
										MODE		REGISTER			
16-BIT WORD DATA										8-BIT BYTE DATA					
32-BIT LONG DATA															

# ANDI

## AND Immediate (M68000 Family)

# ANDI

### 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 can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d <sub>16</sub> ,PC)	—	—
(d <sub>8</sub> ,PC,Xn)	—	—

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	—	—
([bd,PC,Xn],od)	—	—
([bd,PC],Xn,od)	—	—

\*Can be used with CPU32

Immediate field—Data immediately following the instruction.

If size = 00, the data is the low-order byte of the immediate word.

If size = 01, the data is the entire immediate word.

If size = 10, the data is the next two immediate words.

# ANDI to CCR

## CCR AND Immediate (M68000 Family)

# ANDI to CCR

**Operation:** Source  $\wedge$  CCR  $\rightarrow$  CCR

**Assembler**

**Syntax:** ANDI # < data > ,CCR

**Attributes:** Size = (Byte)

**Description:** Performs an AND operation of the immediate operand with the condition codes and stores the result in the low-order byte of the status register.

**Condition Codes:**

X	N	Z	V	C
*	*	*	*	*

X — Cleared if bit 4 of immediate operand is zero; unchanged otherwise.

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.

**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	0	1	1	1	1	0	0
								8-BIT BYTE DATA							

# ASL, ASR

## Arithmetic Shift (M68000 Family)

# ASL, ASR

**Operation:** Destination Shifted By Count → Destination

**Assembler** ASd Dx,Dy

**Syntax:** ASd # < data > ,Dy  
ASd < ea >  
where d is direction, L or R

**Attributes:** Size = (Byte, Word, Long)

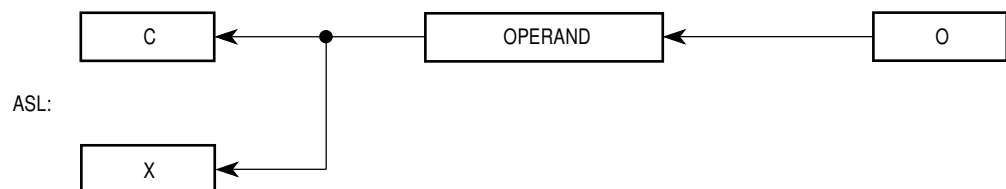
**Description:** Arithmetically shifts the bits of the operand in the direction (L or R) specified.

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 the value in the data register specified in instruction modulo 64.

The size of the operation can be specified as byte, word, or long. An operand in memory can 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; zeros are shifted into the low-order bit. The overflow bit indicates if any sign changes occur during the shift.

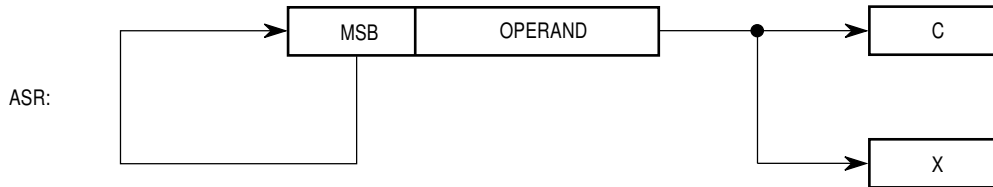


# ASL, ASR

## Arithmetic Shift (M68000 Family)

# ASL, ASR

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 (MSB) is shifted into the high-order bit.



### Condition Codes:

X	N	Z	V	C
*	*	*	*	*

- X — Set according to the last bit shifted out of the operand; unaffected for a shift count of zero.
- 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.

### 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:

Count/Register field—Specifies shift count or register that contains the shift count:

If  $i/r = 0$ , this field contains the shift count. The values 1 – 7 represent counts of 1 – 7; a value of zero represents a count of eight.

If  $i/r = 1$ , this field specifies the data register that contains the shift count (modulo 64).

# ASL, ASR

## Arithmetic Shift (M68000 Family)

# ASL, ASR

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 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		

### Instruction Fields:

dr field—Specifies the direction of the shift.

- 0 — Shift right
- 1 — Shift left

# ASL, ASR

## Arithmetic Shift (M68000 Family)

# ASL, ASR

Effective Address field—Specifies the operand to be shifted. Only memory alterable addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	—	—
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d <sub>16</sub> ,PC)	—	—
(d <sub>8</sub> ,PC,Xn)	—	—

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	—	—
([bd,PC,Xn],od)	—	—
([bd,PC],Xn,od)	—	—

\*Can be used with CPU32.

# Bcc

## Branch Conditionally (M68000 Family)

# Bcc

**Operation:** If Condition True  
Then  $PC + d_n \rightarrow PC$

**Assembler Syntax:** Bcc < label >

**Attributes:** Size = (Byte, Word, Long\*)  
\*(MC68020, MC68030, and MC68040 only)

**Description:** If the specified condition is true, program execution continues at location (PC + displacement). The program counter contains the address of the instruction word for the Bcc instruction plus two. The displacement is a twos-complement integer that represents the relative distance in bytes from the current program counter to the destination program counter. If the 8-bit displacement field in the instruction word is zero, a 16-bit displacement (the word immediately following the instruction) is used. If the 8-bit displacement field in the instruction word is all ones (\$FF), the 32-bit displacement (long word immediately following the instruction) is used. Condition code cc specifies one of the following conditional tests (refer to Table 3-19 for more information on these conditional tests):

Mnemonic	Condition	Mnemonic	Condition
CC(HI)	Carry Clear	LS	Low or Same
CS(LO)	Carry Set	LT	Less Than
EQ	Equal	MI	Minus
GE	Greater or Equal	NE	Not Equal
GT	Greater Than	PL	Plus
HI	High	VC	Overflow Clear
LE	Less or Equal	VS	Overflow Set

**Condition Codes:**

Not affected.



# Bcc

## Branch Conditionally (M68000 Family)

# Bcc

### 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 = \$00															
32-BIT DISPLACEMENT IF 8-BIT DISPLACEMENT = \$FF															

### Instruction Fields:

Condition field—The binary code for one of the conditions listed in the table.

8-Bit Displacement field—Two's complement integer specifying the number of bytes between the branch instruction and the next instruction to be executed if the condition is met.

16-Bit Displacement field—Used for the displacement when the 8-bit displacement field contains \$00.

32-Bit Displacement field—Used for the displacement when the 8-bit displacement field contains \$FF.

### NOTE

A branch to the immediately following instruction automatically uses the 16-bit displacement format because the 8-bit displacement field contains \$00 (zero offset).

# BCHG

## Test a Bit and Change (M68000 Family)

# BCHG

**Operation:** TEST ( < number > of Destination) → Z;  
 TEST ( < number > of Destination) → < bit number > of Destination

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

**Attributes:** Size = (Byte, Long)

**Description:** Tests a bit in the destination operand and sets the Z condition code appropriately, then inverts the specified bit in the destination. When the destination is a data register, any of the 32 bits can be specified by the modulo 32-bit number. When the destination is a memory location, the operation is a byte operation, and the bit number is modulo 8. In all cases, bit zero refers to the least significant bit. The bit number for this operation may be specified in either of two ways:

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

### Condition Codes:

X	N	Z	V	C
—	—	*	—	—

X — Not affected.

N — Not affected.

Z — Set if the bit tested is zero; cleared otherwise.

V — Not affected.

C — Not affected.

# BCHG

## Test a Bit and Change (M68000 Family)

# BCHG

### Instruction Format:

BIT NUMBER DYNAMIC, SPECIFIED IN A REGISTER

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:

Register field—Specifies the data register that contains the bit number.

Effective Address field—Specifies the destination location. Only data alterable addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d <sub>16</sub> ,PC)	—	—
(d <sub>8</sub> ,PC,Xn)	—	—

#### MC68020, MC68030, and MC68040 only

(bd,An,Xn)**	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)†	—	—
([bd,PC,Xn],od)	—	—
([bd,PC],Xn,od)	—	—

\*Long only; all others are byte only.

\*\*Can be used with CPU32.

# BCHG

## Test a Bit and Change (M68000 Family)

# BCHG

### Instruction Format:

BIT NUMBER STATIC, SPECIFIED AS IMMEDIATE DATA

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:

Effective Address field—Specifies the destination location. Only data alterable addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d <sub>16</sub> ,PC)	—	—
(d <sub>8</sub> ,PC,Xn)	—	—

#### MC68020, MC68030, and MC68040 only

(bd,An,Xn)**	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)†	—	(bd,An,Xn)**
([bd,PC,Xn],od)	—	([bd,An,Xn],od)
([bd,PC],Xn,od)	—	([bd,An],Xn,od)

\*Long only; all others are byte only.

\*\*Can be used with CPU32.

Bit Number field—Specifies the bit number.

# BCLR

## Test a Bit and Clear (M68000 Family)

# BCLR

**Operation:** TEST ( < bit number > of Destination) → Z; 0 → < bit number > of Destination

**Assembler** BCLR Dn, < ea >

**Syntax:** BCLR # < data > , < ea >

**Attributes:** Size = (Byte, Long)

**Description:** Tests a bit in the destination operand and sets the Z condition code appropriately, then clears the specified bit in the destination. When a data register is the destination, any of the 32 bits can be specified by a modulo 32-bit number. When a memory location is the destination, the operation is a byte operation, and the bit number is modulo 8. In all cases, bit zero refers to the least significant bit. The bit number for this operation can be specified in either of two ways:

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

### Condition Codes:

X	N	Z	V	C
—	—	*	—	—

X — Not affected.

N — Not affected.

Z — Set if the bit tested is zero; cleared otherwise.

V — Not affected.

C — Not affected.

# BCLR

## Test a Bit and Clear (M68000 Family)

# BCLR

### Instruction Format:

BIT NUMBER DYNAMIC, SPECIFIED IN A REGISTER

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:

Register field—Specifies the data register that contains the bit number.

Effective Address field—Specifies the destination location. Only data alterable addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d <sub>16</sub> ,PC)	—	—
(d <sub>8</sub> ,PC,Xn)	—	—

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)**	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)†	—	—
([bd,PC,Xn],od)	—	—
([bd,PC],Xn,od)	—	—

\*Long only; all others are byte only.

\*\*Can be used with CPU32.

# BCLR

## Test a Bit and Clear (M68000 Family)

# BCLR

### Instruction Format:

BIT NUMBER STATIC, SPECIFIED AS IMMEDIATE DATA

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:

Effective Address field—Specifies the destination location. Only data alterable addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d <sub>16</sub> ,PC)	—	—
(d <sub>8</sub> ,PC,Xn)	—	—

#### MC68020, MC68030, and MC68040 only

(bd,An,Xn)**	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)†	—	—
([bd,PC,Xn],od)	—	—
([bd,PC],Xn,od)	—	—

\*Long only; all others are byte only.

\*\*Can be used with CPU32.

Bit Number field—Specifies the bit number.

**BKPT****Breakpoint****BKPT****(MC68EC000, MC68010, MC68020, MC68030, MC68040, CPU32)****Operation:** Run Breakpoint Acknowledge Cycle; TRAP As Illegal Instruction**Assembler****Syntax:** BKPT # < data >**Attributes:** Unsized

**Description:** For the MC68010, a breakpoint acknowledge bus cycle is run with function codes driven high and zeros on all address lines. Whether the breakpoint acknowledge bus cycle is terminated with  $\overline{DT\ ACK}$ ,  $\overline{BERR}$ , or  $\overline{VPA}$ , the processor always takes an illegal instruction exception. During exception processing, a debug monitor can distinguish different software breakpoints by decoding the field in the BKPT instruction. For the MC68000 and MC68008, the breakpoint cycle is not run, but an illegal instruction exception is taken.

For the MC68020, MC68030, and CPU32, a breakpoint acknowledge bus cycle is executed with the immediate data (value 0 – 7) on bits 2 – 4 of the address bus and zeros on bits 0 and 1 of the address bus. The breakpoint acknowledge bus cycle accesses the CPU space, addressing type 0, and provides the breakpoint number specified by the instruction on address lines A2 – A4. If the external hardware terminates the cycle with  $\overline{DSACKx}$  or  $\overline{STERM}$ , the data on the bus (an instruction word) is inserted into the instruction pipe and is executed after the breakpoint instruction. The breakpoint instruction requires a word to be transferred so, if the first bus cycle accesses an 8-bit port, a second bus cycle is required. If the external logic terminates the breakpoint acknowledge bus cycle with  $\overline{BERR}$  (i.e., no instruction word available), the processor takes an illegal instruction exception.

For the MC68040, this instruction executes a breakpoint acknowledge bus cycle. Regardless of the cycle termination, the MC68040 takes an illegal instruction exception.

For more information on the breakpoint instruction refer to the appropriate user's manual on bus operation.

This instruction supports breakpoints for debug monitors and real-time hardware emulators.



# BKPT

**Breakpoint**  
 (MC68EC000, MC68010, MC68020,  
 MC68030, MC68040, CPU32)

# BKPT

**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 Field:**

Vector field—Contains the immediate data, a value in the range of 0 – 7. This is the breakpoint number.

# BRA

## Branch Always (M68000 Family)

# BRA

**Operation:**  $PC + d_n \rightarrow PC$

**Assembler**

**Syntax:** BRA < label >

**Attributes:** Size = (Byte, Word, Long\*)  
\*(MC68020, MC68030, MC68040 only)

**Description:** Program execution continues at location (PC) + displacement. The program counter contains the address of the instruction word of the BRA instruction plus two. The displacement is a twos complement integer that represents the relative distance in bytes from the current program counter to the destination program counter. If the 8-bit displacement field in the instruction word is zero, a 16-bit displacement (the word immediately following the instruction) is used. If the 8-bit displacement field in the instruction word is all ones (\$FF), the 32-bit displacement (long 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 = \$00															
32-BIT DISPLACEMENT IF 8-BIT DISPLACEMENT = \$FF															

**Instruction Fields:**

8-Bit Displacement field—Twos complement integer specifying the number of bytes between the branch instruction and the next instruction to be executed.

16-Bit Displacement field—Used for a larger displacement when the 8-bit displacement is equal to \$00.

32-Bit Displacement field—Used for a larger displacement when the 8-bit displacement is equal to \$FF.

**NOTE**

A branch to the immediately following instruction automatically uses the 16-bit displacement format because the 8-bit displacement field contains \$00 (zero offset).

# BSET

## Test a Bit and Set (M68000 Family)

# BSET

**Operation:** TEST ( < bit number > of Destination) → Z; 1 → < bit number > of Destination

**Assembler Syntax:** BSET Dn, < ea >

**Syntax:** BSET # < data > , < ea >

**Attributes:** Size = (Byte, Long)

**Description:** Tests a bit in the destination operand and sets the Z condition code appropriately, then sets the specified bit in the destination operand. When a data register is the destination, any of the 32 bits can be specified by a modulo 32-bit number. When a memory location is the destination, the operation is a byte operation, and the bit number is modulo 8. In all cases, bit zero refers to the least significant bit.

The bit number for this operation can be specified in either of two ways:

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

### Condition Codes:

X	N	Z	V	C
—	—	*	—	—

X — Not affected.

N — Not affected.

Z — Set if the bit tested is zero; cleared otherwise.

V — Not affected.

C — Not affected.

# BSET

## Test a Bit and Set (M68000 Family)

# BSET

### Instruction Format:

BIT NUMBER DYNAMIC, SPECIFIED IN A REGISTER

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:

Register field—Specifies the data register that contains the bit number.

Effective Address field—Specifies the destination location. Only data alterable addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d <sub>16</sub> ,PC)	—	—
(d <sub>8</sub> ,PC,Xn)	—	—

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)**	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)†	—	—
([bd,PC,Xn],od)	—	—
([bd,PC],Xn,od)	—	—

\*Long only; all others are byte only.

\*\*Can be used with CPU32.

# BSET

## Test a Bit and Set (M68000 Family)

# BSET

### Instruction Format:

BIT NUMBER STATIC, SPECIFIED AS IMMEDIATE DATA

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	BIT NUMBER								

### Instruction Fields:

Effective Address field—Specifies the destination location. Only data alterable addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d <sub>16</sub> ,PC)	—	—
(d <sub>8</sub> ,PC,Xn)	—	—

#### MC68020, MC68030, and MC68040 only

(bd,An,Xn)**	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)†	—	—
([bd,PC,Xn],od)	—	—
([bd,PC],Xn,od)	—	—

\*Long only; all others are byte only.

\*\*Can be used with CPU32.

Bit Number field—Specifies the bit number.

# BSR

## Branch to Subroutine (M68000 Family)

# BSR

**Operation:** SP – 4 → SP; PC → (SP); PC + d<sub>n</sub> → PC

**Assembler**

**Syntax:** BSR < label >

**Attributes:** Size = (Byte, Word, Long\*)  
\*(MC68020, MC68030, MC68040 only)

**Description:** Pushes the long-word address of the instruction immediately following the BSR instruction onto the system stack. The program counter contains the address of the instruction word plus two. Program execution then continues at location (PC) + displacement. The displacement is a twos complement integer that represents the relative distance in bytes from the current program counter to the destination program counter. If the 8-bit displacement field in the instruction word is zero, a 16-bit displacement (the word immediately following the instruction) is used. If the 8-bit displacement field in the instruction word is all ones (\$FF), the 32-bit displacement (long 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 = \$00															
32-BIT DISPLACEMENT IF 8-BIT DISPLACEMENT = \$FF															

# BSR

## Branch to Subroutine (M68000 Family)

# BSR

### Instruction Fields:

8-Bit Displacement field—Twos complement integer specifying the number of bytes between the branch instruction and the next instruction to be executed.

16-Bit Displacement field—Used for a larger displacement when the 8-bit displacement is equal to \$00.

32-Bit Displacement field—Used for a larger displacement when the 8-bit displacement is equal to \$FF.

### NOTE

A branch to the immediately following instruction automatically uses the 16-bit displacement format because the 8-bit displacement field contains \$00 (zero offset).

# BTST

## Test a Bit (M68000 Family)

# BTST

**Operation:** TEST ( < bit number > of Destination) → Z

**Assembler** BTST Dn, < ea >

**Syntax:** BTST # < data > , < ea >

**Attributes:** Size = (Byte, Long)

**Description:** Tests a bit in the destination operand and sets the Z condition code appropriately. When a data register is the destination, any of the 32 bits can be specified by a modulo 32- bit number. When a memory location is the destination, the operation is a byte operation, and the bit number is modulo 8. In all cases, bit zero refers to the least significant bit. The bit number for this operation can be specified in either of two ways:

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

**Condition Codes:**

X	N	Z	V	C
—	—	*	—	—

X — Not affected.

N — Not affected.

Z — Set if the bit tested is zero; cleared otherwise.

V — Not affected.

C — Not affected.



# BTST

## Test a Bit (M68000 Family)

# BTST

### Instruction Format:

BIT NUMBER DYNAMIC, SPECIFIED IN A REGISTER

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:

Register field—Specifies the data register that contains the bit number.

Effective Address field—Specifies the destination location. Only data addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	111	100
(d <sub>16</sub> ,PC)	111	010
(d <sub>8</sub> ,PC,Xn)	111	011

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)**	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)†	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

\*Long only; all others are byte only.

\*\*Can be used with CPU32.

# BTST

## Test a Bit (M68000 Family)

# BTST

### Instruction Format:

BIT NUMBER STATIC, SPECIFIED AS IMMEDIATE DATA

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:

Effective Address field—Specifies the destination location. Only data addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d <sub>16</sub> ,PC)	111	010
(d <sub>8</sub> ,PC,Xn)	111	011

#### MC68020, MC68030, and MC68040 only

(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

\*Can be used with CPU32.

Bit Number field—Specifies the bit number.

# CHK

## Check Register Against Bounds (M68000 Family)

# CHK

**Operation:** If  $D_n < 0$  or  $D_n > \text{Source}$   
Then TRAP

**Assembler**

**Syntax:** CHK < ea > ,Dn

**Attributes:** Size = (Word, Long\*)  
\*(MC68020, MC68030, MC68040 only)

**Description:** Compares the value in the data register specified in the instruction to zero and to the upper bound (effective address operand). The upper bound is a twos complement integer. If the register value is less than zero or greater than the upper bound, a CHK instruction exception (vector number 6) occurs.

**Condition Codes:**

X	N	Z	V	C
—	*	U	U	U

- X — Not affected.
- N — Set if  $D_n < 0$ ; cleared if  $D_n > \text{effective address operand}$ ; undefined otherwise.
- Z — Undefined.
- V — Undefined.
- C — Undefined.

**Instruction Format:**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	REGISTER			SIZE	0	EFFECTIVE ADDRESS						
											MODE	REGISTER			

# CHK

## Check Register Against Bounds (M68000 Family)

# CHK

### Instruction Fields:

Register field—Specifies the data register that contains the value to be checked.

Size field—Specifies the size of the operation.

11— Word operation

10— Long operation

Effective Address field—Specifies the upper bound operand. Only data addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	111	100
(d <sub>16</sub> ,PC)	111	010
(d <sub>8</sub> ,PC,Xn)	111	011

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

\*Can be used with CPU32.

# CLR

## Clear an Operand (M68000 Family)

# CLR

**Operation:** 0 → Destination

**Assembler**

**Syntax:** CLR < ea >

**Attributes:** Size = (Byte, Word, Long)

**Description:** Clears the destination operand to zero. The size of the operation may be specified as byte, word, or long.

**Condition Codes:**

X	N	Z	V	C
—	0	1	0	0

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

**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			REGISTER		

# CLR

## Clear an Operand (M68000 Family)

# CLR

### 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 can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d <sub>16</sub> ,PC)	—	—
(d <sub>8</sub> ,PC,Xn)	—	—

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	—	—
([bd,PC,Xn],od)	—	—
([bd,PC],Xn,od)	—	—

\*Can be used with CPU32.

### NOTE

In the MC68000 and MC68008 a memory location is read before it is cleared.

# CMP

## Compare (M68000 Family)

# CMP

**Operation:** Destination – Source → cc

**Assembler**

**Syntax:** CMP < ea > , Dn

**Attributes:** Size = (Byte, Word, Long)

**Description:** Subtracts the source operand from the destination data register and sets the condition codes according to the result; the data register is not changed. The size of the operation can be byte, word, or long.

**Condition Codes:**

X	N	Z	V	C
—	*	*	*	*

X — Not affected.

N — Set if the result is negative; cleared otherwise.

Z — Set if the result is zero; cleared otherwise.

V — Set if an overflow occurs; cleared otherwise.

C — Set if a borrow occurs; cleared otherwise.

**Instruction Format:**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	REGISTER			OPMODE			EFFECTIVE ADDRESS MODE			REGISTER		

**Instruction Fields:**

Register field—Specifies the destination data register.

Opmode field

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

# CMP

## Compare (M68000 Family)

# CMP

Effective Address field—Specifies the source operand. All addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An*	001	reg. number:An
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	111	100
(d <sub>16</sub> ,PC)	111	010
(d <sub>8</sub> ,PC,Xn)	111	011

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)**	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)†	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

\*Word and Long only.

\*\*Can be used with CPU32.

### 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 the distinction.



# CMPA

## Compare Address (M68000 Family)

# CMPA

**Operation:** Destination – Source → cc

**Assembler**

**Syntax:** CMPA < ea > , An

**Attributes:** Size = (Word, Long)

**Description:** Subtracts the source operand from the destination address register and sets the condition codes according to the result; the address register is not changed. The size of the operation can be specified as word or long. Word length source operands are sign-extended to 32 bits for comparison.

**Condition Codes:**

X	N	Z	V	C
—	*	*	*	*

- X — Not affected.
- 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.

**Instruction Format:**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	REGISTER			OPMODE			EFFECTIVE ADDRESS MODE			REGISTER		

# CMPA

## Compare Address (M68000 Family)

# CMPA

### Instruction Fields:

Register field—Specifies the destination address register.

Opmode 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 can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	001	reg. number:An
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	111	100
(d <sub>16</sub> ,PC)	111	010
(d <sub>8</sub> ,PC,Xn)	111	011

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

\*Can be used with CPU32.

# CMPI

## Compare Immediate (M68000 Family)

# CMPI

**Operation:** Destination – Immediate Data → cc

**Assembler**

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

**Attributes:** Size = (Byte, Word, Long)

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

**Condition Codes:**

X	N	Z	V	C
—	*	*	*	*

X — Not affected.

N — Set if the result is negative; cleared otherwise.

Z — Set if the result is zero; cleared otherwise.

V — Set if an overflow occurs; cleared otherwise.

C — Set if a borrow occurs; cleared otherwise.

**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			
16-BIT WORD DATA								8-BIT BYTE DATA							
32-BIT LONG DATA															

# CMPI

## Compare Immediate (M68000 Family)

# CMPI

### 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 addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	reg. number:Dn	(xxx).W	111	000
An	—	—	(xxx).L	111	001
(An)	010	reg. number:An	#<data>	—	—
(An) +	011	reg. number:An			
– (An)	100	reg. number:An			
(d <sub>16</sub> ,An)	101	reg. number:An	(d <sub>16</sub> ,PC)*	111	010
(d <sub>8</sub> ,An,Xn)	110	reg. number:An	(d <sub>8</sub> ,PC,Xn)*	111	011

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)**	110	reg. number:An	(bd,PC,Xn)†	111	011
([bd,An,Xn],od)	110	reg. number:An	([bd,PC,Xn],od)	111	011
([bd,An],Xn,od)	110	reg. number:An	([bd,PC],Xn,od)	111	011

\*PC relative addressing modes do not apply to MC68000, MC680008, or MC6801.

\*\*Can be used with CPU32.

Immediate field—Data immediately following the instruction.

If size = 00, the data is the low-order byte of the immediate word.

If size = 01, the data is the entire immediate word.

If size = 10, the data is the next two immediate words.

# CMPM

## Compare Memory (M68000 Family)

# CMPM

**Operation:** Destination – Source → cc

**Assembler**

**Syntax:** CMPM (Ay) + ,(Ax) +

**Attributes:** Size = (Byte, Word, Long)

**Description:** Subtracts the source operand from the destination operand and sets the condition codes according to the results; the destination location is not changed. The operands are always addressed with the postincrement addressing mode, using the address registers specified in the instruction. The size of the operation may be specified as byte, word, or long.

**Condition Codes:**

X	N	Z	V	C
—	*	*	*	*

X — Not affected.

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.

**Instruction Format:**

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

**Instruction Fields:**

Register Ax field—(always the destination) Specifies an address register in the postincrement addressing mode.

Size field—Specifies the size of the operation.

00 — Byte operation

01 — Word operation

10 — Long operation

Register Ay field—(always the source) Specifies an address register in the postincrement addressing mode.

# DBcc

## Test Condition, Decrement, and Branch (M68000 Family)

# DBcc

**Operation:** If Condition False  
Then  $(D_n - 1 \rightarrow D_n; \text{If } D_n \neq -1 \text{ Then } PC + d_n \rightarrow PC)$

**Assembler**

**Syntax:** DBcc Dn, < label >

**Attributes:** Size = (Word)

**Description:** Controls a loop of instructions. The parameters are a condition code, a data register (counter), and a displacement value. The instruction first tests the condition for termination; if it is true, no operation is performed. If the termination condition is not true, the low-order 16 bits of the counter data register decrement by one. If the result is  $-1$ , 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 the program counter plus the sign-extended 16-bit displacement. The value in the program counter is the address of the instruction word of the DBcc instruction plus two. The displacement is a twos complement integer that represents the relative distance in bytes from the current program counter to the destination program counter. Condition code cc specifies one of the following conditional tests (refer to Table 3-19 for more information on these conditional tests):

Mnemonic	Condition	Mnemonic	Condition
CC(HI)	Carry Clear	LS	Low or Same
CS(LO)	Carry Set	LT	Less Than
EQ	Equal	MI	Minus
F	False	NE	Not Equal
GE	Greater or Equal	PL	Plus
GT	Greater Than	T	True
HI	High	VC	Overflow Clear
LE	Less or Equal	VS	Overflow Set

**Condition Codes:**

Not affected.

# DBcc

## Test Condition, Decrement, and Branch (M68000 Family)

# DBcc

### 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		
16-BIT DISPLACEMENT															

### Instruction Fields:

Condition field—The binary code for one of the conditions listed in the table.

Register field—Specifies the data register used as the counter.

Displacement field—Specifies the number of bytes to branch.

### NOTE

The terminating condition is similar to the UNTIL loop clauses of high-level languages. For example: DBMI can be stated as "decrement and branch until minus".

Most assemblers accept DBRA for DBF for use when only a count terminates the loop (no condition is tested).

A program can enter a loop at the beginning or by branching to the trailing DBcc instruction. Entering the loop at the beginning is useful for indexed addressing modes and dynamically specified bit operations. In this case, the control index count must be one less than the desired number of loop executions. However, when entering a loop by branching directly to the trailing DBcc instruction, the control count should equal the loop execution count. In this case, if a zero count occurs, the DBcc instruction does not branch, and the main loop is not executed.

# DIVS, DIVSL

**Signed Divide  
(M68000 Family)**

# DIVS, DIVSL

**Operation:** Destination | Source → Destination

**Assembler Syntax:** DIVS.W < ea > ,Dn32/16 → 16r – 16q  
 \*DIVS.L < ea > ,Dq 32/32 → 32q  
 \*DIVS.L < ea > ,Dr:Dq 64/32 → 32r – 32q  
 \*DIVSL.L < ea > ,Dr:Dq 32/32 → 32r – 32q  
 \*Applies to MC68020, MC68030, MC68040, CPU32 only

**Attributes:** Size = (Word, Long)

**Description:** Divides the signed destination operand by the signed source operand and stores the signed result in the destination. The instruction uses one of four forms. The word form of the instruction divides a long word by a word. The result is a quotient in the lower word (least significant 16 bits) and a remainder in the upper word (most significant 16 bits). The sign of the remainder is the same as the sign of the dividend.

The first long form divides a long word by a long word. The result is a long quotient; the remainder is discarded.

The second long form divides a quad word (in any two data registers) by a long word. The result is a long-word quotient and a long-word remainder.

The third long form divides a long word by a long word. The result is a long-word quotient and a long-word remainder.

Two special conditions may arise during the operation:

1. Division by zero causes a trap.
2. Overflow may be detected and set before the instruction completes. If the instruction detects an overflow, it sets the overflow condition code, and the operands are unaffected.

**Condition Codes:**

X	N	Z	V	C
—	*	*	*	0

X—Not affected.

N — Set if the quotient is negative; cleared otherwise; undefined if overflow or divide by zero occurs.

Z — Set if the quotient is zero; cleared otherwise; undefined if overflow or divide by zero occurs.

V — Set if division overflow occurs; undefined if divide by zero occurs; cleared otherwise.

C — Always cleared.



# DIVS, DIVSL

Signed Divide  
(M68000 Family)

# DIVS, DIVSL

## Instruction Format:

WORD

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 alterable addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	111	100
(d <sub>16</sub> ,PC)	111	010
(d <sub>8</sub> ,PC,Xn)	111	011

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

\*Can be used with CPU32.

## NOTE

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

# DIVS, DIVSL

Signed Divide  
(M68000 Family)

# DIVS, DIVSL

## Instruction Format:

LONG

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	0	0	0	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	REGISTER Dq			1	SIZE	0	0	0	0	0	0	0	REGISTER Dr		

## Instruction Fields:

Effective Address field—Specifies the source operand. Only data alterable addressing modes can be used as listed in the following tables:

### MC68020, MC68030, and MC68040 only

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	111	100
(d <sub>16</sub> ,PC)	111	010
(d <sub>8</sub> ,PC,Xn)	111	011
(bd,PC,Xn)	111	011

### MC68020, MC68030, and MC68040 only

([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

Register Dq field—Specifies a data register for the destination operand. The low-order 32 bits of the dividend comes from this register, and the 32-bit quotient is loaded into this register.

Size field—Selects a 32- or 64-bit division operation.

0 — 32-bit dividend is in register Dq.

1 — 64-bit dividend is in Dr – Dq.

# DIVS, DIVSL

Signed Divide  
(M68000 Family)

# DIVS, DIVSL

Register Dr field—After the division, this register contains the 32-bit remainder. If Dr and Dq are the same register, only the quotient is returned. If the size field is 1, this field also specifies the data register that contains the high-order 32 bits of the dividend.

## NOTE

Overflow occurs if the quotient is larger than a 32-bit signed integer.

# DIVU, DIVUL

## Unsigned Divide (M68000 Family)

# DIVU, DIVUL

**Operation:** Destination | Source → Destination

**Assembler Syntax:** DIVU.W < ea > ,Dn32/16 → 16r – 16q  
 \*DIVU.L < ea > ,Dq 32/32 → 32q  
 \*DIVU.L < ea > ,Dr:Dq 64/32 → 32r – 32q  
 \*DIVUL.L < ea > ,Dr:Dq 32/32 → 32r – 32q  
 \*Applies to MC68020, MC68030, MC68040, CPU32 only.

**Attributes:** Size = (Word, Long)

**Description:** Divides the unsigned destination operand by the unsigned source operand and stores the unsigned result in the destination. The instruction uses one of four forms. The word form of the instruction divides a long word by a word. The result is a quotient in the lower word (least significant 16 bits) and a remainder in the upper word (most significant 16 bits).

The first long form divides a long word by a long word. The result is a long quotient; the remainder is discarded.

The second long form divides a quad word (in any two data registers) by a long word. The result is a long-word quotient and a long-word remainder.

The third long form divides a long word by a long word. The result is a long-word quotient and a long-word remainder.

Two special conditions may arise during the operation:

1. Division by zero causes a trap.
2. Overflow may be detected and set before the instruction completes. If the instruction detects an overflow, it sets the overflow condition code, and the operands are unaffected.

**Condition Codes:**

X	N	Z	V	C
—	*	*	*	0

- X — Not affected.
- N — Set if the quotient is negative; cleared otherwise; undefined if overflow or divide by zero occurs.
- Z — Set if the quotient is zero; cleared otherwise; undefined if overflow or divide by zero occurs.
- V — Set if division overflow occurs; cleared otherwise; undefined if divide by zero occurs.
- C — Always cleared.

# DIVU, DIVUL

Unsigned Divide  
(M68000 Family)

# DIVU, DIVUL

## Instruction Format:

WORD

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 can be used as listed in the following tables:

### MC68020, MC68030, and MC68040 only

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	111	100
(d <sub>16</sub> ,PC)	111	010
(d <sub>8</sub> ,PC,Xn)	111	011

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

\*\*Can be used with CPU32.

## NOTE

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

# DIVU, DIVUL

## Unsigned Divide (M68000 Family)

# DIVU, DIVUL

### Instruction Format:

LONG

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	0	0	0	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	REGISTER Dq			0	SIZE	0	0	0	0	0	0	0	REGISTER Dr		

### Instruction Fields:

Effective Address field—Specifies the source operand. Only data addressing modes can be used as listed in the following tables:

#### MC68020, MC68030, and MC68040 only

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An
(bd,An,Xn)*	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	111	100
(d <sub>16</sub> ,PC)	111	010
(d <sub>8</sub> ,PC,Xn)	111	011
(bd,PC,Xn)*	111	011

#### MC68020, MC68030, and MC68040 only

([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

Register Dq field—Specifies a data register for the destination operand. The low-order 32 bits of the dividend comes from this register, and the 32-bit quotient is loaded into this register.

Size field—Selects a 32- or 64-bit division operation.

0 — 32-bit dividend is in register Dq.

1 — 64-bit dividend is in Dr – Dq.

**DIVU, DIVUL****Unsigned Divide  
(M68000 Family)****DIVU, DIVUL**

Register Dr field—After the division, this register contains the 32-bit remainder. If Dr and Dq are the same register, only the quotient is returned. If the size field is 1, this field also specifies the data register that contains the high-order 32 bits of the dividend.

**NOTE**

Overflow occurs if the quotient is larger than a 32-bit unsigned integer.

# EOR

## Exclusive-OR Logical (M68000 Family)

# EOR

**Operation:** Source  $\oplus$  Destination  $\rightarrow$  Destination

**Assembler**

**Syntax:** EOR Dn, < ea >

**Attributes:** Size = (Byte, Word, Long)

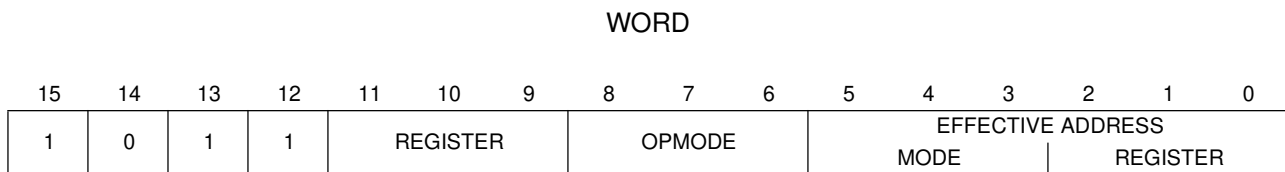
**Description:** Performs an exclusive-OR operation on the destination operand using the source operand and stores the result in the destination location. The size of the operation may be specified to be byte, word, or long. The source operand must be a data register. The destination operand is specified in the effective address field.

**Condition Codes:**

X	N	Z	V	C
—	*	*	0	0

- X — Not affected.
- 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.

**Instruction Format:**



**Instruction Fields:**

Register field—Specifies any of the eight data registers.

Opmode field

<b>Byte</b>	<b>Word</b>	<b>Long</b>	<b>Operation</b>
100	101	110	< ea > $\oplus$ Dn $\rightarrow$ < ea >



# EOR

## Exclusive-OR Logical (M68000 Family)

# EOR

Effective Address field—Specifies the destination ope data alterable addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d <sub>16</sub> ,PC)	—	—
(d <sub>8</sub> ,PC,Xn)	—	—

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	—	—
([bd,PC,Xn],od)	—	—
([bd,PC],Xn,od)	—	—

\*Can be used with CPU32.

### NOTE

Memory-to-data-register operations are not allowed. Most assemblers use EORI when the source is immediate data.

# EORI

## Exclusive-OR Immediate (M68000 Family)

# EORI

**Operation:** Immediate Data  $\oplus$  Destination  $\rightarrow$  Destination

**Assembler**

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

**Attributes:** Size = (Byte, Word, Long)

**Description:** Performs an exclusive-OR operation on the destination operand using the immediate data and the destination operand and stores the result in the destination location. The size of the operation may be specified as byte, word, or long. The size of the immediate data matches the operation size.

**Condition Codes:**

X	N	Z	V	C
—	*	*	0	0

- X — Not affected.
- 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.

**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			
16-BIT WORD DATA								8-BIT BYTE DATA							
32-BIT LONG DATA															

# EORI

## Exclusive-OR Immediate (M68000 Family)

# EORI

### 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 can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d <sub>16</sub> ,PC)	—	—
(d <sub>8</sub> ,PC,Xn)	—	—

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	—	—
([bd,PC,Xn],od)	—	—
([bd,PC],Xn,od)	—	—

\*Can be used with CPU32.

Immediate field—Data immediately following the instruction.

If size = 00, the data is the low-order byte of the immediate word.

If size = 01, the data is the entire immediate word.

If size = 10, the data is next two immediate words.

# EORI to CCR

## Exclusive-OR Immediate to Condition Code (M68000 Family)

# EORI to CCR

**Operation:** Source  $\oplus$  CCR  $\rightarrow$  CCR

**Assembler**

**Syntax:** EORI # < data > ,CCR

**Attributes:** Size = (Byte)

**Description:** Performs an exclusive-OR operation on the condition code register using the immediate operand and stores the result in the condition code register (low-order byte of the status register). All implemented bits of the condition code register are affected.

**Condition Codes:**

X	N	Z	V	C
*	*	*	*	*

X — Changed if bit 4 of immediate operand is one; unchanged otherwise.

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.

**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
								8-BIT BYTE DATA							

# EXG

## Exchange Registers (M68000 Family)

# EXG

**Operation:** Rx  $\leftrightarrow$  Ry

**Assembler Syntax:** EXG Dx,Dy  
EXG Ax,Ay EXG Dx,Ay

**Attributes:** Size = (Long)

**Description:** Exchanges the contents of two 32-bit registers. The instruction performs three types of exchanges.

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	OPMODE				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.

Opmode field—Specifies the type of exchange.

- 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, EXTB

## Sign-Extend (M68000 Family)

# EXT, EXTB

**Operation:** Destination Sign-Extended → Destination

**Assembler** EXT.W Dnextend byte to word

**Syntax:** EXT.L Dnextend word to long word

EXTB.L Dnextend byte to long word (MC68020, MC68030  
MC68040, CPU32)

**Attributes:** Size = (Word, Long)

**Description:** Extends a byte in a data register to a word or a long word, or a word in a data register to a long word, by replicating the sign bit to the left. If the operation extends a byte to a word, bit 7 of the designated data register is copied to bits 15 – 8 of that data register. If the operation extends a word to a long word, bit 15 of the designated data register is copied to bits 31 – 16 of the data register. The EXTB form copies bit 7 of the designated register to bits 31 – 8 of the data register.

### Condition Codes:

X	N	Z	V	C
—	*	*	0	0

X — Not affected.

N — Set if the result is negative; cleared otherwise.

Z — Set if the result is zero; cleared otherwise.

V — Always cleared.

C — Always cleared.

### 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	OPMODE			0	0	0	REGISTER		

### Instruction Fields:

Opmode 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.

111— Sign-extend low-order byte of data register to long.

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

# ILLEGAL

## Take Illegal Instruction Trap (M68000 Family)

# ILLEGAL

**Operation:** \*SSP – 2 → SSP; Vector Offset → (SSP);  
 SSP – 4 → SSP; PC → (SSP);  
 SSP – 2 → SSP; SR → (SSP);  
 Illegal Instruction Vector Address → PC

\*The MC68000 and MC68008 cannot write the vector offset and format code to the system stack.

**Assembler**

**Syntax:** ILLEGAL

**Attributes:** Unsized

**Description:** Forces an illegal instruction exception, vector number 4. All other illegal instruction bit patterns are reserved for future extension of the instruction set and should not be used to force an 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	1	0	1	1	1	1	1	1	0	0

# JMP

## Jump (M68000 Family)

# JMP

**Operation:** Destination Address → PC

**Assembler**

**Syntax:** JMP < ea >

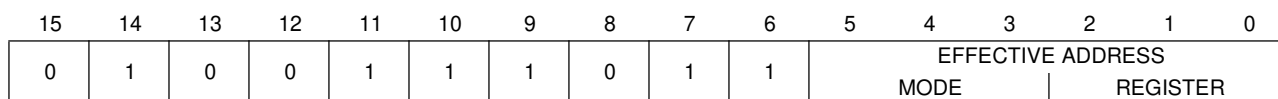
**Attributes:** Unsized

**Description:** Program execution continues at the effective address specified by the instruction. The addressing mode for the effective address must be a control addressing mode.

**Condition Codes:**

Not affected.

**Instruction Format:**



**Instruction Field:**

Effective Address field—Specifies the address of the next instruction. Only control addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	—	—
An	—	—
(An)	010	reg. number:An
(An) +	—	—
– (An)	—	—
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d <sub>16</sub> ,PC)	111	010
(d <sub>8</sub> ,PC,Xn)	111	011

**MC68020, MC68030, and MC68040 only**

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

\*Can be used with CPU32.



# JSR

## Jump to Subroutine (M68000 Family)

# JSR

**Operation:** SP – 4 → Sp; PC → (SP); Destination Address → PC

**Assembler**

**Syntax:** JSR < ea >

**Attributes:** Unsized

**Description:** Pushes the long-word address of the instruction immediately following the JSR instruction onto the system stack. Program execution then continues at the address specified in the 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	1	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		

**Instruction Field:**

Effective Address field—Specifies the address of the next instruction. Only control addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	—	—
An	—	—
(An)	010	reg. number:An
(An) +	—	—
– (An)	—	—
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d <sub>16</sub> ,PC)	111	010
(d <sub>8</sub> ,PC,Xn)	111	011

**MC68020, MC68030, and MC68040 only**

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

\*Can be used with CPU32.

# LEA

## Load Effective Address (M68000 Family)

# LEA

**Operation:** < ea > → An

**Assembler**

**Syntax:** LEA < ea > ,An

**Attributes:** Size = (Long)

**Description:** Loads the effective address into the specified address register. All 32 bits of the address register are affected by this 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	REGISTER			1	1	1	EFFECTIVE ADDRESS					
									MODE			REGISTER			

**Instruction Fields:**

Register field—Specifies the address register to be updated with the effective address.

Effective Address field—Specifies the address to be loaded into the address register. Only control addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	—	—
An	—	—
(An)	010	reg. number:An
(An) +	—	—
– (An)	—	—
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d <sub>16</sub> ,PC)	111	010
(d <sub>8</sub> ,PC,Xn)	111	011

**MC68020, MC68030, and MC68040 only**

(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

\*Can be used with CPU32.

# LINK

## Link and Allocate (M68000 Family)

# LINK

**Operation:**  $SP - 4 \rightarrow SP; An \rightarrow (SP); SP \rightarrow An; SP + d_n \rightarrow SP$

**Assembler**

**Syntax:** LINK An, # < displacement >

**Attributes:** Size = (Word, Long\*)

\*MC68020, MC68030, MC68040 and CPU32 only.

**Description:** Pushes the contents of the specified address register onto the stack. Then loads the updated stack pointer into the address register. Finally, adds the displacement value to the stack pointer. For word-size operation, the displacement is the sign-extended word following the operation word. For long size operation, the displacement is the long word following the operation word. The address register occupies one long word on the stack. The user should specify a negative displacement in order to allocate stack area.

**Condition Codes:**

Not affected.

**Instruction Format:**

WORD

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		
WORD DISPLACEMENT															

**Instruction Format:**

LONG

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	0	0	0	1	REGISTER		
HIGH-ORDER DISPLACEMENT															
LOW-ORDER DISPLACEMENT															

# LINK

## Link and Allocate (M68000 Family)

# LINK

### Instruction Fields:

Register field—Specifies the address register for the link.

Displacement field—Specifies the twos complement integer 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 (M68000 Family)

# LSL, LSR

**Operation:** Destination Shifted By Count → Destination

**Assembler** LSd Dx,Dy

**Syntax:** LSd # < data > ,Dy  
LSd < ea >  
where d is direction, L or R

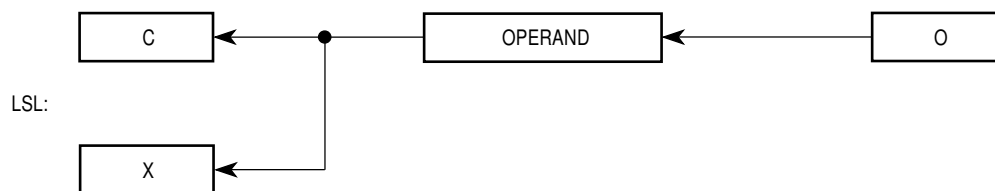
**Attributes:** Size = (Byte, Word, Long)

**Description:** Shifts the bits of the operand in the direction specified (L or R). The carry bit receives the last bit shifted out of the operand. The shift count for the shifting of a register is specified in two different ways:

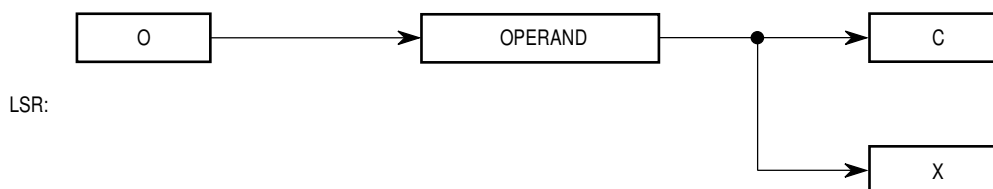
1. Immediate—The shift count (1 – 8) is specified in the instruction.
2. Register—The shift count is the value in the data register specified in the instruction modulo 64.

The size of the operation for register destinations may be specified as byte, word, or long. The contents of memory, < ea > , can be shifted one bit only, and the operand size is restricted to a word.

The LSL instruction shifts the operand to the left the number of positions specified as the shift count. Bits shifted out of the high-order bit go to both the carry and the extend bits; zeros are shifted into the low-order bit.



The LSR instruction shifts the operand to the right the number of positions specified as the shift count. Bits shifted out of the low-order bit go to both the carry and the extend bits; zeros are shifted into the high-order bit.



# LSL, LSR

## Logical Shift (M68000 Family)

# LSL, LSR

### Condition Codes:

X	N	Z	V	C
*	*	*	0	*

- X — Set according to the last bit shifted out of the operand; unaffected for a shift count of zero.
- 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.

### 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:

#### Count/Register field

If  $i/r = 0$ , this field contains the shift count. The values 1 – 7 represent shifts of 1 – 7; value of zero specifies a shift count of eight.

If  $i/r = 1$ , the data register specified in this field contains the shift count (modulo 64).

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 to be shifted.

# LSL, LSR

## Logical Shift (M68000 Family)

# LSL, LSR

### 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:

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 can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	—	—
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d <sub>16</sub> ,PC)	—	—
(d <sub>8</sub> ,PC,Xn)	—	—

#### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	—	—
([bd,PC,Xn],od)	—	—
([bd,PC],Xn,od)	—	—

\*Can be used with CPU32.

# MOVE

## Move Data from Source to Destination (M68000 Family)

# MOVE

**Operation:** Source → Destination

**Assembler**

**Syntax:** MOVE < ea > , < ea >

**Attributes:** Size = (Byte, Word, Long)

**Description:** Moves the data at the source to the destination location and sets the condition codes according to the data. The size of the operation may be specified as byte, word, or long. Condition Codes:

X	N	Z	V	C
—	*	*	0	0

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

**Instruction Format:**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	SIZE		DESTINATION				SOURCE							
				REGISTER			MODE			MODE			REGISTER		

**Instruction Fields:**

- Size field—Specifies the size of the operand to be moved.
  - 01 — Byte operation
  - 11 — Word operation
  - 10 — Long operation



# MOVE

## Move Data from Source to Destination (M68000 Family)

# MOVE

Destination Effective Address field—Specifies the destination location. Only data alterable addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d <sub>16</sub> ,PC)	—	—
(d <sub>8</sub> ,PC,Xn)	—	—

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	—	—
([bd,PC,Xn],od)	—	—
([bd,PC],Xn,od)	—	—

\*Can be used with CPU32.

# MOVE

## Move Data from Source to Destination (M68000 Family)

# MOVE

Source Effective Address field—Specifies the source operand. All addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	001	reg. number:An
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	111	100
(d <sub>16</sub> ,PC)	111	010
(d <sub>8</sub> ,PC,Xn)	111	011

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)**	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)**	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

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

\*\*Can be used with CPU32.

### NOTE

Most assemblers use MOVEA when the destination is an address register.

MOVEQ can be used to move an immediate 8-bit value to a data register.

# MOVEA

## Move Address (M68000 Family)

# MOVEA

**Operation:** Source → Destination

**Assembler**

**Syntax:** MOVEA < ea > ,An

**Attributes:** Size = (Word, Long)

**Description:** Moves the contents of the source to the destination address register. The size of the operation is specified as word or long. Word-size source operands are sign-extended to 32-bit quantities.

**Condition Codes:**

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		0	0	1	MODE		SOURCE REGISTER					

**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.

# MOVEA

## Move Address (M68000 Family)

# MOVEA

Effective Address field—Specifies the location of the source operand. All addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	001	reg. number:An
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	111	100
(d <sub>16</sub> ,PC)	111	010
(d <sub>8</sub> ,PC,Xn)	111	011

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

\*Can be used with CPU32.

# MOVE from CCR

# MOVE from CCR

**Move from the  
Condition Code Register**  
(MC68010, MC68020, MC68030, MC68040, CPU32)

**Operation:** CCR → Destination

**Assembler**

**Syntax:** MOVE CCR, < ea >

**Attributes:** Size = (Word)

**Description:** Moves the condition code bits (zero-extended to word size) to the destination location. The operand size is a word. Unimplemented bits are read as 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			REGISTER		

# MOVE from CCR

**Move from the  
Condition Code Register**  
(MC68010, MC68020, MC68030, MC68040, CPU32)

# MOVE from CCR

**Instruction Field:**

Effective Address field—Specifies the destination location. Only data alterable addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d <sub>16</sub> ,PC)	—	—
(d <sub>8</sub> ,PC,Xn)	—	—

**MC68020, MC68030, and MC68040 only**

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	—	—
([bd,PC,Xn],od)	—	—
([bd,PC],Xn,od)	—	—

\*Can be used with CPU32.

**NOTE**

MOVE from CCR is a word operation. ANDI, ORI, and EORI to CCR are byte operations.

# MOVE to CCR

## Move to Condition Code Register (M68000 Family)

# MOVE to CCR

**Operation:** Source → CCR

**Assembler**

**Syntax:** MOVE < ea > ,CCR

**Attributes:** Size = (Word)

**Description:** Moves the low-order byte of the source operand to the condition code register. The upper byte of the source operand is ignored; the upper byte of the status register is not altered.

**Condition Codes:**

X	N	Z	V	C
*	*	*	*	*

X — Set to the value of bit 4 of the source operand.

N — Set to the value of bit 3 of the source operand.

Z — Set to the value of bit 2 of the source operand.

V — Set to the value of bit 1 of the source operand.

C — Set to the value of bit 0 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			REGISTER		

# MOVE to CCR

## Move to Condition Code Register (M68000 Family)

# MOVE to CCR

### Instruction Field:

Effective Address field—Specifies the location of the source operand. Only data addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	111	100
(d <sub>16</sub> ,PC)	111	010
(d <sub>8</sub> ,PC,Xn)	111	011

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

\*Can be used with CPU32.

### NOTE

MOVE to CCR is a word operation. ANDI, ORI, and EORI to CCR are byte operations.



# MOVEM

## Move Multiple Registers (M68000 Family)

# MOVEM

**Operation:** Registers → Destination; Source → Registers

**Assembler** MOVEM < list > , < ea >

**Syntax:** MOVEM < ea > , < list >

**Attributes:** Size = (Word, Long)

**Description:** Moves the contents of selected registers to or from consecutive memory locations starting at the location specified by the effective address. A register is selected if the bit in the mask field corresponding to that register is set. The instruction size determines whether 16 or 32 bits of each register are transferred. In the case of a word transfer to either address or data registers, each word is sign-extended to 32 bits, and the resulting long word is loaded into the associated register.

Selecting the addressing mode also selects the mode of operation of the MOVEM instruction, and only the control modes, the predecrement mode, and the postincrement mode are valid. If the effective address is specified by one of the control modes, the registers are transferred starting at the specified address, and the address is incremented by the operand length (2 or 4) following each transfer. The order of the registers is from D0 to D7, then from A0 to A7.

If the effective address is specified by the predecrement mode, only a register-to-memory operation is allowed. The registers are stored starting at the specified address minus the operand length (2 or 4), and the address is decremented by the operand length following each transfer. The order of storing is from A7 to A0, then from D7 to D0. When the instruction has completed, the decremented address register contains the address of the last operand stored. For the MC68020, MC68030, MC68040, and CPU32, if the addressing register is also moved to memory, the value written is the initial register value decremented by the size of the operation. The MC68000 and MC68010 write the initial register value (not decremented).

If the effective address is specified by the postincrement mode, only a memory-to-register operation is allowed. The registers are loaded starting at the specified address; the address is incremented by the operand length (2 or 4) following each transfer. The order of loading is the same as that of control mode addressing. When the instruction has completed, the incremented address register contains the address of the last operand loaded plus the operand length. If the addressing register is also loaded from memory, the memory value is ignored and the register is written with the postincremented effective address.

# MOVEM

## Move Multiple Registers (M68000 Family)

# MOVEM

### 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	dr	0	0	1	SIZE	EFFECTIVE ADDRESS					
										MODE		REGISTER			
REGISTER LIST MASK															

### Instruction Fields:

dr field—Specifies the direction of the transfer.

0 — Register to memory.

1 — Memory to register.

Size field—Specifies the size of the registers being transferred.

0 — Word transfer

1 — Long transfer

Effective Address field—Specifies the memory address for the operation. For register-to-memory transfers, only control alterable addressing modes or the predecrement addressing mode can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	—	—
An	—	—
(An)	010	reg. number:An
(An) +	—	—
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d <sub>16</sub> ,PC)	—	—
(d <sub>8</sub> ,PC,Xn)	—	—

#### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	—	—
([bd,PC,Xn],od)	—	—
([bd,PC],Xn,od)	—	—

\*Can be used with CPU32.

# MOVEM

## Move Multiple Registers (M68000 Family)

# MOVEM

For memory-to-register transfers, only control addressing modes or the postincrement addressing mode can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	—	—
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	—	—
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d <sub>16</sub> ,PC)	111	010
(d <sub>8</sub> ,PC,Xn)	111	011

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

\*Can be used with CPU32.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A7	A6	A5	A4	A3	A2	A1	A0	D7	D6	D5	D4	D3	D2	D1	D0

For the predecrement mode addresses, the mask correspondence is reversed:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D0	D1	D2	D3	D4	D5	D6	D7	A0	A1	A2	A3	A4	A5	A6	A7

# MOVEP

## Move Peripheral Data (M68000 Family)

# MOVEP

**Operation:** Source → Destination

**Assembler Syntax:** MOVEP Dx,(d16,Ay)

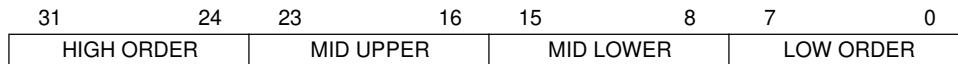
**Syntax:** MOVEP (d16,Ay),Dx

**Attributes:** Size = (Word, Long)

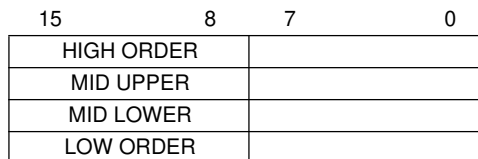
**Description:** Moves data between a data register and alternate bytes within the address space starting at the location specified and incrementing by two. The high-order byte of the data register is transferred first, and the low-order byte is transferred last. The memory address is specified in the address register indirect plus 16-bit displacement addressing mode. This instruction was originally designed for interfacing 8-bit peripherals on a 16-bit data bus, such as the MC68000 bus. Although supported by the MC68020, MC68030, and MC68040, this instruction is not useful for those processors with an external 32-bit bus.

Example: Long transfer to/from an even address.

### Byte Organization in Register



### Byte Organization in 16-Bit Memory (Low Address at Top)



# MOVEP

## Move Peripheral Data (M68000 Family)

# MOVEP

### Byte Organization in 32-Bit Memory

31	24	23	16	15	8	7	0
HIGH ORDER				MID UPPER			
MID LOWER				LOW ORDER			

or

31	24	23	16	15	8	7	0
				HIGH ORDER			
MID UPPER				MID LOWER			
LOW ORDER							

Example: Word transfer to/from (odd address).

### Byte Organization in Register

31	24	23	16	15	8	7	0
				HIGH ORDER		LOW ORDER	

### Byte Organization in 16-Bit Memory (Low Address at Top)

15	8	7	0
		HIGH ORDER	
		LOW ORDER	

### Byte Organization in 32-Bit Memory

31	24	23	16	15	8	7	0
				HIGH ORDER			
LOW ORDER							

or

31	24	23	16	15	8	7	0
HIGH ORDER				LOW ORDER			

# MOVEP

## Move Peripheral Data (M68000 Family)

# MOVEP

### Condition Codes:

Not affected.

### Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	DATA REGISTER				OPMODE			0	0	1	ADDRESS REGISTER		
16-BIT DISPLACEMENT																

### Instruction Fields:

Data Register field—Specifies the data register for the instruction.

Opmode 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 used in the operand address.

# MOVEQ

## Move Quick (M68000 Family)

# MOVEQ

**Operation:** Immediate Data → Destination

**Assembler**

**Syntax:** MOVEQ # < data > ,Dn

**Attributes:** Size = (Long)

**Description:** Moves a byte of immediate data to a 32-bit data register. The data in an 8-bit field within the operation word is sign- extended to a long operand in the data register as it is transferred.

**Condition Codes:**

X	N	Z	V	C
—	*	*	0	0

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

**Instruction Format:**

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

**Instruction Fields:**

Register field—Specifies the data register to be loaded.

Data field—Eight bits of data, which are sign-extended to a long operand.

# MULS

## Signed Multiply (M68000 Family)

# MULS

- Operation:** Source x Destination → Destination
- Assembler Syntax:** MULS.W < ea > ,Dn16 x 16 → 32  
 \*MULS.L < ea > ,DI 32 x 32 → 32  
 \*MULS.L < ea > ,Dh – DI 32 x 32 → 64  
 \*Applies to MC68020, MC68030, MC68040, CPU32
- Attributes:** Size = (Word, Long)

**Description:** Multiplies two signed operands yielding a signed result. This instruction has a word operand form and a long operand form.

In the word form, the multiplier and multiplicand are both word operands, and the result is a long-word operand. A register operand is the low-order word; the upper word of the register is ignored. All 32 bits of the product are saved in the destination data register.

In the long form, the multiplier and multiplicand are both long- word operands, and the result is either a long word or a quad word. The long-word result is the low-order 32 bits of the quad- word result; the high-order 32 bits of the product are discarded.

**Condition Codes:**

X	N	Z	V	C
—	*	*	*	0

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

**NOTE**

Overflow (V = 1) can occur only when multiplying 32-bit operands to yield a 32-bit result. Overflow occurs if the high-order 32 bits of the quad-word product are not the sign extension of the low- order 32 bits.



# MULS

## Signed Multiply (M68000 Family)

# MULS

### Instruction Format:

WORD

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

### Instruction Fields:

Register field—Specifies a data register as the destination.

Effective Address field—Specifies the source operand. Only data alterable addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	111	100
(d <sub>16</sub> ,PC)	111	010
(d <sub>8</sub> ,PC,Xn)	111	011

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

\*Can be used with CPU32.

# MULS

## Signed Multiply (M68000 Family)

# MULS

### Instruction Format:

LONG

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	0	0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	REGISTER DI			1	SIZE	0	0	0	0	0	0	0	REGISTER Dh		

### Instruction Fields:

Effective Address field—Specifies the source operand. Only data addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	reg. number:Dn	(xxx).W	111	000
An	—	—	(xxx).L	111	001
(An)	010	reg. number:An	#<data>	111	100
(An) +	011	reg. number:An			
– (An)	100	reg. number:An			
(d <sub>16</sub> ,An)	101	reg. number:An	(d <sub>16</sub> ,PC)	111	010
(d <sub>8</sub> ,An,Xn)	110	reg. number:An	(d <sub>8</sub> ,PC,Xn)	111	011

#### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An	(bd,PC,Xn)*	111	011
([bd,An,Xn],od)	110	reg. number:An	([bd,PC,Xn],od)	111	011
([bd,An],Xn,od)	110	reg. number:An	([bd,PC],Xn,od)	111	011

\*Can be used with CPU32.

Register DI field—Specifies a data register for the destination operand. The 32-bit multiplicand comes from this register, and the low-order 32 bits of the product are loaded into this register.

Size field—Selects a 32- or 64-bit product.

0 — 32-bit product to be returned to register DI.

1 — 64-bit product to be returned to Dh – DI.

Register Dh field—If size is one, specifies the data register into which the high-order 32 bits of the product are loaded. If Dh = DI and size is one, the results of the operation are undefined. Otherwise, this field is unused.

# MULU

## Unsigned Multiply (M68000 Family)

# MULU

**Operation:** Source x Destination → Destination

**Assembler Syntax:** MULU.W < ea > ,Dn16 x 16 → 32  
 \*MULU.L < ea > ,DI 32 x 32 → 32  
 \*MULU.L < ea > ,Dh – DI 32 x 32 → 64

\*Applies to MC68020, MC68030, MC68040, CPU32 only

**Attributes:** Size = (Word, Long)

**Description:** Multiplies two unsigned operands yielding an unsigned result. This instruction has a word operand form and a long operand form.

In the word form, the multiplier and multiplicand are both word operands, and the result is a long-word operand. A register operand is the low-order word; the upper word of the register is ignored. All 32 bits of the product are saved in the destination data register.

In the long form, the multiplier and multiplicand are both long- word operands, and the result is either a long word or a quad word. The long-word result is the low-order 32 bits of the quad- word result; the high-order 32 bits of the product are discarded.

### Condition Codes:

X	N	Z	V	C
—	*	*	*	0

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

### NOTE

Overflow (V = 1) can occur only when multiplying 32-bit operands to yield a 32-bit result. Overflow occurs if any of the high-order 32 bits of the quad-word product are not equal to zero.

# MULU

## Unsigned Multiply (M68000 Family)

# MULU

### Instruction Format:

WORD

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

### Instruction Fields:

Register field—Specifies a data register as the destination.

Effective Address field—Specifies the source operand. Only data addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	111	100
(d <sub>16</sub> ,PC)	111	010
(d <sub>8</sub> ,PC,Xn)	111	011

#### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

\*Can be used with CPU32.

# MULU

## Unsigned Multiply (M68000 Family)

# MULU

### Instruction Format:

LONG

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	0	0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	REGISTER DI			0	SIZE	0	0	0	0	0	0	0	REGISTER Dh		

### Instruction Fields:

Effective Address field—Specifies the source operand. Only data addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	111	100
(d <sub>16</sub> ,PC)	111	010
(d <sub>8</sub> ,PC,Xn)	111	011

#### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

\*Can be used with CPU32.

Register DI field—Specifies a data register for the destination operand. The 32-bit multiplicand comes from this register, and the low-order 32 bits of the product are loaded into this register.

Size field—Selects a 32- or 64-bit product.

0 — 32-bit product to be returned to register DI.

1 — 64-bit product to be returned to Dh – DI.

Register Dh field—If size is one, specifies the data register into which the high-order 32 bits of the product are loaded. If Dh = DI and size is one, the results of the operation are undefined. Otherwise, this field is unused.

# NBCD

## Negate Decimal with Extend (M68000 Family)

# NBCD

**Operation:**  $0 - \text{Destination}_{10} - X \rightarrow \text{Destination}$

**Assembler**

**Syntax:** NBCD < ea >

**Attributes:** Size = (Byte)

**Description:** Subtracts the destination operand and the extend bit from zero. The operation is performed using binary-coded decimal arithmetic. The packed binary-coded decimal result is saved in the destination location. This instruction produces the tens complement of the destination if the extend bit is zero or the nines complement if the extend bit is one. This is a byte operation only.

**Condition Codes:**

X	N	Z	V	C
*	U	*	U	*

X — Set the same as the carry bit.

N — Undefined.

Z — Cleared if the result is nonzero; unchanged otherwise.

V — Undefined.

C — Set if a decimal borrow occurs; cleared otherwise.

**NOTE**

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

# NBCD

## Negate Decimal with Extend (M68000 Family)

# NBCD

### 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	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		

### Instruction Fields:

Effective Address field—Specifies the destination operand. Only data alterable addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d <sub>16</sub> ,PC)	—	—
(d <sub>8</sub> ,PC,Xn)	—	—

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	—	—
([bd,PC,Xn],od)	—	—
([bd,PC],Xn,od)	—	—

\*Can be used with CPU32.

# NEG

## Negate (M68000 Family)

# NEG

**Operation:** 0 – Destination → Destination

**Assembler**

**Syntax:** NEG < ea >

**Attributes:** Size = (Byte, Word, Long)

**Description:** Subtracts the destination operand from zero and stores the result in the destination location. The size of the operation is specified as byte, word, or long.

**Condition Codes:**

X	N	Z	V	C
*	*	*	*	*

X — Set the same as the carry bit.

N — Set if the result is negative; cleared otherwise.

Z — Set if the result is zero; cleared otherwise.

V — Set if an overflow occurs; cleared otherwise.

C — Cleared if the result is zero; set otherwise.

**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			



# NEG

## Negate (M68000 Family)

# NEG

### 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 can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d <sub>16</sub> ,PC)	—	—
(d <sub>8</sub> ,PC,Xn)	—	—

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	—	—
([bd,PC,Xn],od)	—	—
([bd,PC],Xn,od)	—	—

\*Can be used with CPU32.

# NEGX

## Negate with Extend (M68000 Family)

# NEGX

**Operation:**  $0 - \text{Destination} - X \rightarrow \text{Destination}$

**Assembler**

**Syntax:** `NEGX < ea >`

**Attributes:** Size = (Byte, Word, Long)

**Description:** Subtracts the destination operand and the extend bit from zero. Stores the result in the destination location. The size of the operation is specified as byte, word, or long.

**Condition Codes:**

X	N	Z	V	C
*	*	*	*	*

- X — Set the same as the carry bit.
- N — Set if the result is negative; cleared otherwise.
- Z — Cleared if the result is nonzero; unchanged otherwise.
- V — Set if an overflow occurs; cleared otherwise.
- C — Set if a borrow occurs; cleared otherwise.

**NOTE**

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

# NEGX

## Negate with Extend (M68000 Family)

# NEGX

### 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 can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d <sub>16</sub> ,PC)	—	—
(d <sub>8</sub> ,PC,Xn)	—	—

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	—	—
([bd,PC,Xn],od)	—	—
([bd,PC],Xn,od)	—	—

\*Can be used with CPU32.

# NOP

## No Operation (M68000 Family)

# NOP

**Operation:** None

**Assembler**

**Syntax:** NOP

**Attributes:** Unsized

**Description:** Performs no operation. The processor state, other than the program counter, is unaffected. Execution continues with the instruction following the NOP instruction. The NOP instruction does not begin execution until all pending bus cycles have completed. This synchronizes the pipeline and prevents instruction overlap.

**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 (M68000 Family)

# NOT

**Operation:**            ~ Destination → Destination

**Assembler**

**Syntax:**                NOT < ea >

**Attributes:**            Size = (Byte, Word, Long)

**Description:**Calculates the ones complement of the destination operand and stores the result in the destination location. The size of the operation is specified as byte, word, or long.

**Condition Codes:**

X	N	Z	V	C
—	*	*	0	0

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

**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		

# NOT

## Logical Complement (M68000 Family)

# NOT

### 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 can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d <sub>16</sub> ,PC)	—	—
(d <sub>8</sub> ,PC,Xn)	—	—

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	—	—
([bd,PC,Xn],od)	—	—
([bd,PC],Xn,od)	—	—

\*Can be used with CPU32.

# OR

## Inclusive-OR Logical (M68000 Family)

# OR

**Operation:** Source V Destination → Destination

**Assembler** OR < ea > ,Dn

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

**Attributes:** Size = (Byte, Word, Long)

**Description:** Performs an inclusive-OR operation on the source operand and the destination operand and stores the result in the destination location. The size of the operation is specified as 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

X — Not affected.

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.

### Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	REGISTER			OPMODE			EFFECTIVE ADDRESS					
										MODE			REGISTER		

### Instruction Fields:

Register field—Specifies any of the eight data registers.

Opmode field

Byte	Word	Long	Operation
000	001	010	< ea > V Dn → Dn
100	101	110	Dn V < ea > → < ea >

**OR**
**Inclusive-OR Logical  
(M68000 Family)**
**OR**

Effective Address field—If the location specified is a source operand, only data addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	111	100
(d <sub>16</sub> ,PC)	111	010
(d <sub>8</sub> ,PC,Xn)	111	011

**MC68020, MC68030, and MC68040 only**

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

\*Can be used with CPU32.



# OR

## Inclusive-OR Logical (M68000 Family)

# OR

If the location specified is a destination operand, only memory alterable addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	—	—
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d <sub>16</sub> ,PC)	—	—
(d <sub>8</sub> ,PC,Xn)	—	—

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	—	—
([bd,PC,Xn],od)	—	—
([bd,PC],Xn,od)	—	—

\*Can be used with CPU32.

### NOTE

If the destination is a data register, it must be specified using the destination Dn mode, not the destination < ea > mode.

Most assemblers use ORI when the source is immediate data.

# ORI

## Inclusive-OR (M68000 Family)

# ORI

**Operation:** Immediate Data V Destination → Destination

**Assembler**

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

**Attributes:** Size = (Byte, Word, Long)

**Description:** Performs an inclusive-OR operation on the immediate data and the destination operand and stores the result in the destination location. The size of the operation is specified as byte, word, or long. The size of the immediate data matches the operation size.

**Condition Codes:**

X	N	Z	V	C
—	*	*	0	0

X — Not affected.

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.

**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		REGISTER			
16-BIT WORD DATA								8-BIT BYTE DATA							
32-BIT LONG DATA															

# ORI

## Inclusive-OR (M68000 Family)

# ORI

### 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 can be used as listed in the following tables:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	reg. number:Dn	(xxx).W	111	000
An	—	—	(xxx).L	111	001
(An)	010	reg. number:An	#<data>	—	—
(An) +	011	reg. number:An			
– (An)	100	reg. number:An			
(d <sub>16</sub> ,An)	101	reg. number:An	(d <sub>16</sub> ,PC)	—	—
(d <sub>8</sub> ,An,Xn)	110	reg. number:An	(d <sub>8</sub> ,PC,Xn)	—	—

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An	(bd,PC,Xn)*	—	—
([bd,An,Xn],od)	110	reg. number:An	([bd,PC,Xn],od)	—	—
([bd,An],Xn,od)	110	reg. number:An	([bd,PC],Xn,od)	—	—

\*Can be used with CPU32.

Immediate field—Data immediately following the instruction.

If size = 00, the data is the low-order byte of the immediate word.

If size = 01, the data is the entire immediate word.

If size = 10, the data is the next two immediate words.

# ORI to CCR

## Inclusive-OR Immediate to Condition Codes (M68000 Family)

# ORI to CCR

**Operation:** Source V CCR → CCR

**Assembler**

**Syntax:** ORI # < data > ,CCR

**Attributes:** Size = (Byte)

**Description:** Performs an inclusive-OR operation on the immediate operand and the condition codes and stores the result in the condition code register (low-order byte of the status register). All implemented bits of the condition code register are affected.

**Condition Codes:**

X	N	Z	V	C
*	*	*	*	*

X — Set if bit 4 of immediate operand is one; unchanged otherwise.

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.

**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
								8-BIT BYTE DATA							

# PEA

## Push Effective Address (M68000 Family)

# PEA

**Operation:**  $SP - 4 \rightarrow SP; \langle ea \rangle \rightarrow (SP)$

**Assembler**

**Syntax:** PEA < ea >

**Attributes:** Size = (Long)

**Description:** Computes the effective address and pushes it onto the stack. The effective address is a long address.

**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 Field:**

Effective Address field—Specifies the address to be pushed onto the stack. Only control addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	—	—	(xxx).W	111	000
An	—	—	(xxx).L	111	001
(An)	010	reg. number:An	#<data>	—	—
(An) +	—	—			
– (An)	—	—			
(d <sub>16</sub> ,An)	101	reg. number:An	(d <sub>16</sub> ,PC)	111	010
(d <sub>8</sub> ,An,Xn)	110	reg. number:An	(d <sub>8</sub> ,PC,Xn)	111	011

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An	(bd,PC,Xn)*	111	011
([bd,An,Xn],od)	110	reg. number:An	([bd,PC,Xn],od)	111	011
([bd,An],Xn,od)	110	reg. number:An	([bd,PC],Xn,od)	111	011

\*Can be used with CPU32.

# ROL, ROR

## Rotate (Without Extend) (M68000 Family)

# ROL, ROR

**Operation:** Destination Rotated By < count > → Destination

**Assembler** ROd Dx,Dy

**Syntax:** ROd # < data > ,Dy ROd < ea > where d is direction, L or R

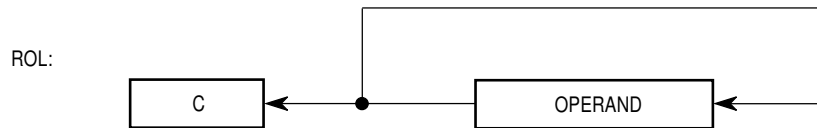
**Attributes:** Size = (Byte, Word, Long)

**Description:** Rotates the bits of the operand in the direction specified (L or R). The extend bit is not included in the rotation. The rotate count for the rotation of a register is specified in either of two ways:

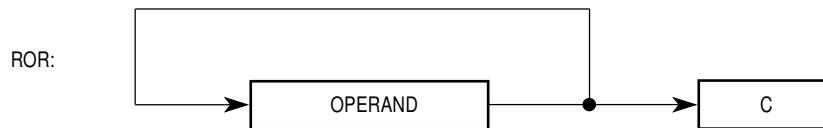
1. Immediate—The rotate count (1 – 8) is specified in the instruction.
2. Register—The rotate count is the value in the data register specified in the instruction, modulo 64.

The size of the operation for register destinations is specified as byte, word, or long. The contents of memory, (ROd < ea > ), can be rotated one bit only, and operand size is restricted to a word.

The ROL instruction rotates the bits of the operand to the left; the rotate count determines the number of bit positions rotated. Bits rotated out of the high-order bit go to the carry bit and also back into the low-order bit.



The ROR instruction rotates the bits of the operand to the right; the rotate count determines the number of bit positions rotated. Bits rotated out of the low-order bit go to the carry bit and also back into the high-order bit.



# ROL,ROR

Rotate (Without Extend)  
(M68000 Family)

# ROL,ROR

## Condition Codes:

X	N	Z	V	C
—	*	*	0	*

X — Not affected.

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 rotated out of the operand; cleared when the rotate count is 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	1	REGISTER	

## Instruction Fields:

Count/Register field:

If  $i/r = 0$ , this field contains the rotate count. The values 1 – 7 represent counts of 1 – 7, and zero specifies a count of eight.

If  $i/r = 1$ , this field specifies a data register that contains the rotate count (modulo 64).

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—Specifies the rotate count location.

If  $i/r = 0$ , immediate rotate count.

If  $i/r = 1$ , register rotate count.

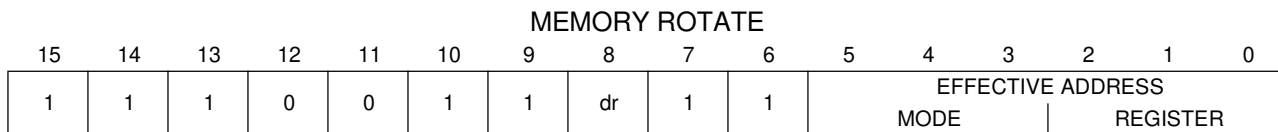
Register field—Specifies a data register to be rotated.

# ROL, ROR

## Rotate (Without Extend) (M68000 Family)

# ROL, ROR

### Instruction Format:



### Instruction Fields:

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 can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	—	—
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d <sub>16</sub> ,PC)	—	—
(d <sub>8</sub> ,PC,Xn)	—	—

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An	(bd,PC,Xn)*	—	—
([bd,An,Xn],od)	110	reg. number:An	([bd,PC,Xn],od)	—	—
([bd,An],Xn,od)	110	reg. number:An	([bd,PC],Xn,od)	—	—

\*Can be used with CPU32.



# ROXL, ROXR

Rotate with Extend  
(M68000 Family)

# ROXL, ROXR

**Operation:** Destination Rotated With X By Count → Destination

**Assembler** ROXd Dx,Dy

**Syntax:** ROXd # < data > ,Dy

ROXd < ea >

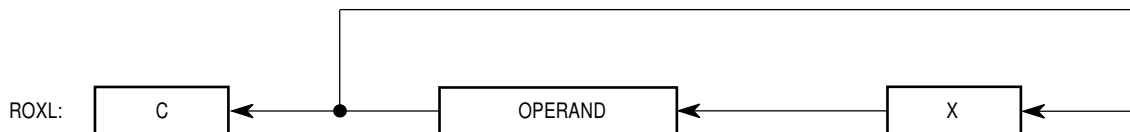
where d is direction, L or R

**Attributes:** Size = (Byte, Word, Long)

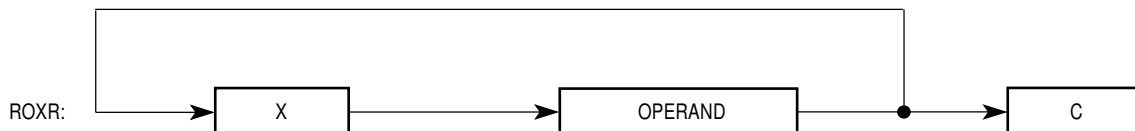
**Description:** Rotates the bits of the operand in the direction specified (L or R). The extend bit is included in the rotation. The rotate count for the rotation of a register is specified in either of two ways:

1. Immediate—The rotate count (1 – 8) is specified in the instruction.
2. Register—The rotate count is the value in the data register specified in the instruction, modulo 64.

The size of the operation for register destinations is specified as byte, word, or long. The contents of memory, < ea > , can be rotated one bit only, and operand size is restricted to a word. The ROXL instruction rotates the bits of the operand to the left; the rotate count determines the number of bit positions rotated. Bits rotated out of the high-order bit go to the carry bit and the extend bit; the previous value of the extend bit rotates into the low-order bit.



The ROXR instruction rotates the bits of the operand to the right; the rotate count determines the number of bit positions rotated. Bits rotated out of the low-order bit go to the carry bit and the extend bit; the previous value of the extend bit rotates into the high-order bit.



# ROXL, ROXR

Rotate with Extend  
(M68000 Family)

# ROXL, ROXR

### Condition Codes:

X	N	Z	V	C
*	*	*	0	*

X — Set to the value of the last bit rotated out of the operand; unaffected when the rotate count is zero.

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 rotated out of the operand; when the rotate count is zero, set to the value of the extend bit.

### 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:

Count/Register field:

If  $i/r = 0$ , this field contains the rotate count. The values 1 – 7 represent counts of 1 – 7, and zero specifies a count of eight.

If  $i/r = 1$ , this field specifies a data register that contains the rotate count (modulo 64).

dr field—Specifies the direction of the rotate.

0 — Rotate right

1 — Rotate left

# ROXL, ROXR

Rotate with Extend  
(M68000 Family)

# ROXL, ROXR

Size field—Specifies the size of the operation.

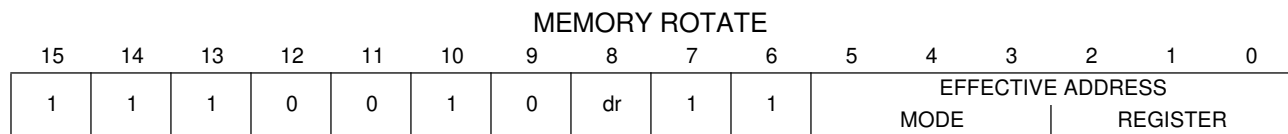
- 00 — Byte operation
- 01 — Word operation
- 10 — Long operation

i/r field—Specifies the rotate count location.

- If i/r = 0, immediate rotate count.
- If i/r = 1, register rotate count.

Register field—Specifies a data register to be rotated.

## Instruction Format:



## Instruction Fields:

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 can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	—	—
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d <sub>16</sub> ,PC)	—	—
(d <sub>8</sub> ,PC,Xn)	—	—

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	—	—
([bd,PC,Xn],od)	—	—
([bd,PC],Xn,od)	—	—

\*Can be used with CPU32.

# RTD

## Return and Deallocate (MC68010, MC68020, MC68030, MC68040, CPU32)

# RTD

**Operation:** (SP) → PC; SP + 4 + d<sub>n</sub> → SP

**Assembler**

**Syntax:** RTD # < displacement >

**Attributes:** Unsized

**Description:** Pulls the program counter value from the stack and adds the sign-extended 16-bit displacement value to the stack pointer. The previous program counter value 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	0
16-BIT DISPLACEMENT															

**Instruction Field:**

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

# RTR

## Return and Restore Condition Codes (M68000 Family)

# RTR

**Operation:** (SP) → CCR; SP + 2 → SP; (SP) → PC; SP + 4 → SP

**Assembler**

**Syntax:** RTR

**Attributes:** Unsized

**Description:** Pulls the condition code and program counter values from the stack. The previous condition code and program counter values are lost. The supervisor portion of the status register is unaffected.

**Condition Codes:**

Set to the condition codes from the stack.

**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	1	1

# RTS

## Return from Subroutine (M68000 Family)

# RTS

**Operation:** (SP) → PC; SP + 4 → SP

**Assembler**

**Syntax:** RTS

**Attributes:** Unsized

**Description:** Pulls the program counter value from the stack. The previous program counter value 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 (M68000 Family)

# SBCD

**Operation:** Destination<sub>10</sub> – Source<sub>10</sub> – X → Destination

**Assembler** SBCD Dx,Dy

**Syntax:** SBCD – (Ax), – (Ay)

**Attributes:** Size = (Byte)

**Description:** Subtracts the source operand and the extend bit from the destination operand and stores the result in the destination location. The subtraction is performed using binary-coded decimal arithmetic; the operands are packed binary-coded decimal numbers. The instruction has two modes:

1. Data register to data register—the data registers specified in the instruction contain the operands.
2. Memory to memory—the address registers specified in the instruction access the operands from memory using the predecrement addressing mode.

This operation is a byte operation only.

### Condition Codes:

X	N	Z	V	C
*	U	*	U	*

X — Set the same as the carry bit.

N — Undefined.

Z — Cleared if the result is nonzero; unchanged otherwise.

V — Undefined.

C — Set if a borrow (decimal) is generated; cleared otherwise.

### 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.

# SBCD

## Subtract Decimal with Extend (M68000 Family)

# SBCD

### Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	REGISTER Dy/Ay			1	0	0	0	0	R/M	REGISTER Dx/Ax		

### Instruction Fields:

Register Dy/Ay 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 Dx/Ax 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 (M68000 Family)

# Scc

**Operation:** If Condition True  
 Then 1s → Destination  
 Else 0s → Destination

**Assembler**

**Syntax:** Scc < ea >

**Attributes:** Size = (Byte)

**Description:** Tests the specified condition code; if the condition is true, sets the byte specified by the effective address to TRUE (all ones). Otherwise, sets that byte to FALSE (all zeros). Condition code cc specifies one of the following conditional tests (refer to Table 3-19 for more information on these conditional tests):

Mnemonic	Condition	Mnemonic	Condition
CC(HI)	Carry Clear	LS	Low or Same
CS(LO)	Carry Set	LT	Less Than
EQ	Equal	MI	Minus
F	False	NE	Not Equal
GE	Greater or Equal	PL	Plus
GT	Greater Than	T	True
HI	High	VC	Overflow Clear
LE	Less or Equal	VS	Overflow Set

**Condition Codes:**

Not affected.

# Scc

## Set According to Condition (M68000 Family)

# Scc

### 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**—The binary code for one of the conditions listed in the table.

**Effective Address field**—Specifies the location in which the TRUE/FALSE byte is to be stored. Only data alterable addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	reg. number:Dn	(xxx).W	111	000
An	—	—	(xxx).L	111	001
(An)	010	reg. number:An	#<data>	—	—
(An) +	011	reg. number:An			
– (An)	100	reg. number:An			
(d <sub>16</sub> ,An)	101	reg. number:An	(d <sub>16</sub> ,PC)	—	—
(d <sub>8</sub> ,An,Xn)	110	reg. number:An	(d <sub>8</sub> ,PC,Xn)	—	—

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An	(bd,PC,Xn)*	—	—
([bd,An,Xn],od)	110	reg. number:An	([bd,PC,Xn],od)	—	—
([bd,An],Xn,od)	110	reg. number:An	([bd,PC],Xn,od)	—	—

\*Can be used with CPU32.

### NOTE

A subsequent NEG.B instruction with the same effective address can be used to change the Scc result from TRUE or FALSE to the equivalent arithmetic value (TRUE = 1, FALSE = 0). In the MC68000 and MC68008, a memory destination is read before it is written.

# SUB

## Subtract (M68000 Family)

# SUB

**Operation:** Destination – Source → Destination

**Assembler** SUB < ea > ,Dn

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

**Attributes:** Size = (Byte, Word, Long)

**Description:** Subtracts the source operand from the destination operand and stores the result in the destination. The size of the operation is specified as byte, word, or long. The mode of the instruction indicates which operand is the source, which is the destination, and which is the operand size.

### Condition Codes:

X	N	Z	V	C
*	*	*	*	*

X — Set to the value of the carry bit.

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.

### Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	REGISTER			OPMODE			EFFECTIVE ADDRESS					
										MODE			REGISTER		

# SUB

## Subtract (M68000 Family)

# SUB

### Instruction Fields:

Register field—Specifies any of the eight data registers.

Opmode field

Byte	Word	Long	Operation
000	001	010	$D_n - \langle ea \rangle \rightarrow D_n$
100	101	110	$\langle ea \rangle - D_n \rightarrow \langle ea \rangle$

Effective Address field—Determines the addressing mode. If the location specified is a source operand, all addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	reg. number:Dn	(xxx).W	111	000
An*	001	reg. number:An	(xxx).L	111	001
(An)	010	reg. number:An	#<data>	111	100
(An) +	011	reg. number:An			
– (An)	100	reg. number:An			
(d <sub>16</sub> ,An)	101	reg. number:An	(d <sub>16</sub> ,PC)	111	010
(d <sub>8</sub> ,An,Xn)	110	reg. number:An	(d <sub>8</sub> ,PC,Xn)	111	011

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)**	110	reg. number:An	(bd,PC,Xn)**	111	011
([bd,An,Xn],od)	110	reg. number:An	([bd,PC,Xn],od)	111	011
([bd,An],Xn,od)	110	reg. number:An	([bd,PC],Xn,od)	111	011

\*For byte-sized operation, address register direct is not allowed.

\*\*Can be used with CPU32.

# SUB

## Subtract (M68000 Family)

# SUB

If the location specified is a destination operand, only memory alterable addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	—	—
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d <sub>16</sub> ,PC)	—	—
(d <sub>8</sub> ,PC,Xn)	—	—

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	—	—
([bd,PC,Xn],od)	—	—
([bd,PC],Xn,od)	—	—

\*Can be used with CPU32.

### NOTE

If the destination is a data register, it must be specified as a destination Dn address, not as a destination < ea > address.

Most assemblers use SUBA when the destination is an address register and SUBI or SUBQ when the source is immediate data.

# SUBA

## Subtract Address (M68000 Family)

# SUBA

**Operation:** Destination – Source → Destination

**Assembler**

**Syntax:** SUBA < ea > ,An

**Attributes:** Size = (Word, Long)

**Description:** Subtracts the source operand from the destination address register and stores the result in the address register. The size of the operation is specified as word or long. Word-sized source operands are sign-extended to 32-bit quantities prior to the subtraction.

**Condition Codes:**

Not affected.

**Instruction Format:**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	REGISTER			OPMODE			EFFECTIVE ADDRESS					
											MODE		REGISTER		

**Instruction Fields:**

Register field—Specifies the destination, any of the eight address registers.

Opmode 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.

# SUBA

## Subtract Address (M68000 Family)

# SUBA

Effective Address field—Specifies the source operand. All addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	001	reg. number:An
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	111	100
(d <sub>16</sub> ,PC)	111	010
(d <sub>8</sub> ,PC,Xn)	111	011

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

(bd,PC,Xn)*	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

\*Can be used with CPU32.

# SUBI

## Subtract Immediate (M68000 Family)

# SUBI

**Operation:** Destination – Immediate Data → Destination

**Assembler**

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

**Attributes:** Size = (Byte, Word, Long)

**Description:** Subtracts the immediate data from the destination operand and stores the result in the destination location. The size of the operation is specified as byte, word, or long. The size of the immediate data matches the operation size.

**Condition Codes:**

X	N	Z	V	C
*	*	*	*	*

- X — Set to the value of the carry bit.
- N — Set if the result is negative; cleared otherwise.
- Z — Set if the result is zero; cleared otherwise.
- V — Set if an overflow occurs; cleared otherwise.
- C — Set if a borrow occurs; cleared otherwise.

**Instruction Format:**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	0	SIZE		EFFECTIVE ADDRESS					
										MODE		REGISTER			
16-BIT WORD DATA								8-BIT BYTE DATA							
32-BIT LONG DATA															



# SUBI

## Subtract Immediate (M68000 Family)

# SUBI

### 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 can be used as listed in the following tables:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	reg. number:Dn	(xxx).W	111	000
An	—	—	(xxx).L	111	001
(An)	010	reg. number:An	#<data>	—	—
(An) +	011	reg. number:An			
– (An)	100	reg. number:An			
(d <sub>16</sub> ,An)	101	reg. number:An	(d <sub>16</sub> ,PC)	—	—
(d <sub>8</sub> ,An,Xn)	110	reg. number:An	(d <sub>8</sub> ,PC,Xn)	—	—

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An	(bd,PC,Xn)*	—	—
([bd,An,Xn],od)	110	reg. number:An	([bd,PC,Xn],od)	—	—
([bd,An],Xn,od)	110	reg. number:An	([bd,PC],Xn,od)	—	—

\*Can be used with CPU32.

Immediate field—Data immediately following the instruction.

If size = 00, the data is the low-order byte of the immediate word.

If size = 01, the data is the entire immediate word.

If size = 10, the data is the next two immediate words.

# SUBQ

## Subtract Quick (M68000 Family)

# SUBQ

**Operation:** Destination – Immediate Data → Destination

**Assembler**

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

**Attributes:** Size = (Byte, Word, Long)

**Description:** Subtracts the immediate data (1 – 8) from the destination operand. The size of the operation is specified as byte, word, or long. Only word and long operations can be used with address registers, and the condition codes are not affected. When subtracting from address registers, the entire destination address register is used, despite the operation size.

**Condition Codes:**

X	N	Z	V	C
*	*	*	*	*

- X — Set to the value of the carry bit.
- N — Set if the result is negative; cleared otherwise.
- Z — Set if the result is zero; cleared otherwise.
- V — Set if an overflow occurs; cleared otherwise.
- C — Set if a borrow occurs; cleared otherwise.

**Instruction Format:**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	DATA			1	SIZE		EFFECTIVE ADDRESS MODE			REGISTER		

# SUBQ

## Subtract Quick (M68000 Family)

# SUBQ

### Instruction Fields:

Data field—Three bits of immediate data; 1 – 7 represent immediate values of 1 – 7, and zero represents eight.

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 can be used as listed in the following tables:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	reg. number:Dn	(xxx).W	111	000
An*	001	reg. number:An	(xxx).L	111	001
(An)	010	reg. number:An	#<data>	—	—
(An) +	011	reg. number:An			
– (An)	100	reg. number:An			
(d <sub>16</sub> ,An)	101	reg. number:An	(d <sub>16</sub> ,PC)	—	—
(d <sub>8</sub> ,An,Xn)	110	reg. number:An	(d <sub>8</sub> ,PC,Xn)	—	—

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)**	110	reg. number:An	(bd,PC,Xn)**	—	—
([bd,An,Xn],od)	110	reg. number:An	([bd,PC,Xn],od)	—	—
([bd,An],Xn,od)	110	reg. number:An	([bd,PC],Xn,od)	—	—

\*Word and long only.

\*\*Can be used with CPU32.

# SUBX

## Subtract with Extend (M68000 Family)

# SUBX

**Operation:** Destination – Source – X → Destination

**Assembler** SUBX Dx,Dy

**Syntax:** SUBX – (Ax), – (Ay)

**Attributes:** Size = (Byte, Word, Long)

**Description:** Subtracts the source operand and the extend bit from the destination operand and stores the result in the destination

**location. The instruction has two modes:**

1. Data register to data register—the data registers specified in the instruction contain the operands.
2. Memory to memory—the address registers specified in the instruction access the operands from memory using the predecrement addressing mode.

The size of the operand is specified as byte, word, or long.

**Condition Codes:**

X	N	Z	V	C
*	*	*	*	*

- X — Set to the value of the carry bit.
- N — Set if the result is negative; cleared otherwise.
- Z — Cleared if the result is nonzero; unchanged otherwise.
- V — Set if an overflow occurs; cleared otherwise.
- C — Set if a borrow occurs; cleared otherwise.

**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.

# SUBX

## Subtract with Extend (M68000 Family)

# SUBX

### Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	REGISTER Dy/Ay			1	SIZE		0	0	R/M	REGISTER Dx/Ax		

### Instruction Fields:

Register Dy/Ay 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 Dx/Ax 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 (M68000 Family)

# SWAP

**Operation:** Register 31 – 16 ↔ Register 15 – 0

**Assembler**

**Syntax:** SWAP Dn

**Attributes:** Size = (Word)

**Description:** Exchange the 16-bit words (halves) of a data register.

**Condition Codes:**

X	N	Z	V	C
—	*	*	0	0

X — Not affected.

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.

**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	0			REGISTER

**Instruction Field:**

Register field—Specifies the data register to swap.

# TAS

## Test and Set an Operand (M68000 Family)

# TAS

**Operation:** Destination Tested → Condition Codes; 1 → Bit 7 of Destination

**Assembler**

**Syntax:** TAS < ea >

**Attributes:** Size = (Byte)

**Description:** Tests and sets the byte operand addressed by the effective address field. The instruction tests the current value of the operand and sets the N and Z condition bits appropriately. TAS also sets the high-order bit of the operand. The operation uses a locked or read-modify-write transfer sequence. This instruction supports use of a flag or semaphore to coordinate several processors.

**Condition Codes:**

X	N	Z	V	C
—	*	*	0	0

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

**Instruction Format:**

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

# TAS

## Test and Set an Operand (M68000 Family)

# TAS

### Instruction Fields:

Effective Address field—Specifies the location of the tested operand. Only data alterable addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	reg. number:Dn	(xxx).W	111	000
An	—	—	(xxx).L	111	001
(An)	010	reg. number:An	#<data>	—	—
(An) +	011	reg. number:An			
– (An)	100	reg. number:An			
(d <sub>16</sub> ,An)	101	reg. number:An	(d <sub>16</sub> ,PC)	—	—
(d <sub>8</sub> ,An,Xn)	110	reg. number:An	(d <sub>8</sub> ,PC,Xn)	—	—

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An	(bd,PC,Xn)*	—	—
([bd,An,Xn],od)	110	reg. number:An	([bd,PC,Xn],od)	—	—
([bd,An],Xn,od)	110	reg. number:An	([bd,PC],Xn,od)	—	—

\*Can be used with CPU32.



# TRAP

## Trap (M68000 Family)

# TRAP

**Operation:** 1 → S-Bit of SR  
 \*SSP – 2 → SSP; Format/Offset → (SSP);  
 SSP – 4 → SSP; PC → (SSP); SSP – 2 → SSP;  
 SR → (SSP); Vector Address → PC

\*The MC68000 and MC68008 do not write vector offset or format code to the system stack.

**Assembler**

**Syntax:** TRAP # < vector >

**Attributes:** Unsized

**Description:** Causes a TRAP # < vector > exception. The instruction adds the immediate operand (vector) of the instruction to 32 to obtain the vector number. The range of vector values is 0 – 15, which provides 16 vectors.

**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	0	VECTOR			

**Instruction Fields:**

Vector field—Specifies the trap vector to be taken.

# TRAPV

## Trap on Overflow (M68000 Family)

# TRAPV

**Operation:** If V  
Then TRAP

**Assembler Syntax:** TRAPV

**Attributes:** Unsized

**Description:** If the overflow condition is set, causes a TRAPV exception with a vector number 7. If the overflow condition is not set, the processor performs no operation 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	1	1	0

# TST

## Test an Operand (M68000 Family)

# TST

**Operation:** Destination Tested → Condition Codes

**Assembler**

**Syntax:** TST < ea >

**Attributes:** Size = (Byte, Word, Long)

**Description:** Compares the operand with zero and sets the condition codes according to the results of the test. The size of the operation is specified as byte, word, or long.

**Condition Codes:**

X	N	Z	V	C
—	*	*	0	0

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

**Instruction Format:**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	1	0	SIZE		EFFECTIVE ADDRESS					
										MODE		REGISTER			

# TST

## Test an Operand (M68000 Family)

# TST

### Instruction Fields:

Size field—Specifies the size of the operation.

- 00 — Byte operation
- 01 — Word operation
- 10 — Long operation

Effective Address field—Specifies the addressing mode for the destination operand as listed in the following tables:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	reg. number:Dn	(xxx).W	111	000
An*	001	reg. number:An	(xxx).L	111	001
(An)	010	reg. number:An	#<data>*	111	100
(An) +	011	reg. number:An			
– (An)	100	reg. number:An			
(d <sub>16</sub> ,An)	101	reg. number:An	(d <sub>16</sub> ,PC)**	111	010
(d <sub>8</sub> ,An,Xn)	110	reg. number:An	(d <sub>8</sub> ,PC,Xn)**	111	011

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)***	110	reg. number:An	(bd,PC,Xn)***	111	011
([bd,An,Xn],od)	110	reg. number:An	([bd,PC,Xn],od)	111	011
([bd,An],Xn,od)	110	reg. number:An	([bd,PC],Xn,od)	111	011

\*MC68020, MC68030, MC68040, and CPU32. Address register direct allowed only for word and long.

\*\*PC relative addressing modes do not apply to MC68000, MC680008, or MC68010.

\*\*\*Can be used with CPU32.

# UNLK

## Unlink (M68000 Family)

# UNLK

**Operation:** An → SP; (SP) → An; SP + 4 → SP

**Assembler**

**Syntax:** UNLK An

**Attributes:** Unsized

**Description:** Loads the stack pointer from the specified address register, then loads the address register 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 Field:**

Register field—Specifies the address register for the instruction.

# ANDI to SR

## AND Immediate to the Status Register (M68000 Family)

# ANDI to SR

**Operation:** If Supervisor State  
Then Source L SR → SR  
ELSE TRAP

**Assembler Syntax:** ANDI # < data > ,SR

**Attributes:** size = (word)

**Description:** Performs an AND operation of the immediate operand with the contents of the status register and stores the result in the status register. All implemented bits of the status register are affected.

**Condition Codes:**

X	N	Z	V	C
*	*	*	*	*

X—Cleared if bit 4 of immediate operand is zero; unchanged otherwise.

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.

**Instruction Format:**

5	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
16-BIT WORD DATA															

# EORI to SR

## Exclusive-OR Immediate to the Status Register (M68000 Family)

# EORI to SR

**Operation:** If Supervisor State  
Then Source  $\oplus$  SR  $\rightarrow$  SR  
ELSE TRAP

**Assembler**

**Syntax:** EORI # < data > ,SR

**Attributes:** Size = (Word)

**Description:** Performs an exclusive-OR operation on the contents of the status register using the immediate operand and stores the result in the status register. All implemented bits of the status register are affected.

**Condition Codes:**

X	N	Z	V	C
*	*	*	*	*

X—Changed if bit 4 of immediate operand is one; unchanged otherwise.

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.

**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
16-BIT WORD DATA															

# MOVE from SR

Move from the Status Register  
(MC68EC000, MC68010, MC68020,  
MC68030, MC68040, CPU32)

# MOVE from SR

**Operation:** If Supervisor State  
Then SR → Destination  
Else TRAP

**Assembler  
Syntax:** MOVE SR, < ea >

**Attributes:** Size = (Word)

**Description:** Moves the data in the status register to the destination location. The destination is word length. Unimplemented bits are read as 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	0	0	1	1	EFFECTIVE ADDRESS MODE		REGISTER			



# MOVE from SR

Move from the Status Register  
(MC68EC000, MC68010, MC68020,  
MC68030, MC68040, CPU32)

# MOVE from SR

## Instruction Field:

Effective Address field—Specifies the destination location. Only data alterable addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
—(An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	—	—
(d <sub>16</sub> ,PC)	—	—
(d <sub>8</sub> ,PC,Xn)	—	—

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn] ,od)	110	reg. number:An
([bd,An],Xn ,od)	110	reg. number:An

(bd,PC,Xn)*	—	—
([bd,PC,Xn] ,od)	—	—
([bd,PC],Xn ,od)	—	—

\*Available for the CPU32.

## NOTE

Use the MOVE from CCR instruction to access only the condition codes.

# MOVE to SR

## Move to the Status Register (M68000 Family)

# MOVE to SR

**Operation:** If Supervisor State  
Then Source → SR  
Else TRAP

**Assembler Syntax:** MOVE < ea > ,SR

**Attributes:** Size = (Word)

**Description:** Moves the data in the source operand to the status register. The source operand is a word, and all implemented bits of the status register are affected.

**Condition Codes:**  
Set according to 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	1	0	1	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		

# MOVE to SR

## Move to the Status Register (M68000 Family)

# MOVE to SR

### Instruction Field:

Effective Address field—Specifies the location of the source operand. Only data addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
—(An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d <sub>16</sub> ,PC)	111	010
(d <sub>8</sub> ,PC,Xn)	111	011

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn] ,od)	110	reg. number:An
([bd,An],Xn ,od)	110	reg. number:An

(bd,PC,Xn)*	111	011
([bd,PC,Xn] ,od)	111	011
([bd,PC],Xn ,od)	111	011

\*Available for the CPU32.

# MOVE USP

## Move User Stack Pointer (M68000 Family)

# MOVE USP

**Operation:** If Supervisor State  
Then USP → An or An → USP  
Else TRAP

**Assembler Syntax:** MOVE USP,An  
MOVE An,USP

**Attributes:** Size = (Long)

**Description:** Moves the contents of the user stack pointer to or from the specified address register.

**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	0	dr	REGISTER		

### Instruction Fields:

- dr field—Specifies the direction of transfer.
  - 0—Transfer the address register to the user stack pointer.
  - 1—Transfer the user stack pointer to the address register.

Register field—Specifies the address register for the operation.

# MOVEC

## Move Control Register (MC68010, MC68020, MC68030, MC68040, CPU32)

# MOVEC

**Operation:** If Supervisor State  
Then Rc → Rn or Rn → Rc  
Else TRAP

**Assembler Syntax:** MOVEC Rc,Rn  
MOVEC Rn,Rc

**Attributes:** Size = (Long)

**Description:** Moves the contents of the specified control register (Rc) to the specified general register (Rn) or copies 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:**

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	1	0	1	dr
A/D		REGISTER				CONTROL REGISTER									

**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

# MOVEC

## Move Control Register (MC68010, MC68020, MC68030, MC68040, CPU32)

# MOVEC

Register field—Specifies the register number.

Control Register field—Specifies the control register.

Hex <sup>1</sup>	Control Register
<b>MC68010/MC68020/MC68030/MC68040/CPU32</b>	
000	Source Function Code (SFC)
001	Destination Function Code (DFC)
800	User Stack Pointer (USP)
801	Vector Base Register (VBR)
<b>MC68020/MC68030/MC68040</b>	
002	Cache Control Register (CACR)
802	Cache Address Register (CAAR) <sup>2</sup>
803	Master Stack Pointer (MSP)
804	Interrupt Stack Pointer (ISP)
<b>MC68040/MC68LC040</b>	
003	MMU Translation Control Register (TC)
004	Instruction Transparent Translation Register 0 (ITT0)
005	Instruction Transparent Translation Register 1 (ITT1)
006	Data Transparent Translation Register 0 (DTT0)
007	Data Transparent Translation Register 1 (DTT1)
805	MMU Status Register (MMUSR)
806	User Root Pointer (URP)
807	Supervisor Root Pointer (SRP)
<b>MC68EC040 only</b>	
004	Instruction Access Control Register 0 (IACR0)
005	Instruction Access Control Register 1 (IACR1)
006	Data Access Control Register 0 (DACR0)
007	Data Access Control Register 1 (DACR1)

NOTES:

1. Any other code causes an illegal instruction exception
2. For the MC68020 and MC68030 only.

# MOVES

## Move Address Space (MC68010, MC68020, MC68030, MC68040, CPU32)

# MOVES

**Operation:** If Supervisor State  
Then Rn → Destination [DFC] or Source [SFC] → Rn  
Else TRAP

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

**Attributes:** Size = (Byte, Word, Long)

**Description:** This instruction moves 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 it moves 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 that data register, depending on the size of the operation. 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	4	3	2	1	0
0	0	0	0	1	1	1	0	SIZE		EFFECTIVE ADDRESS					
				REGISTER		dr	0	0	0	MODE			REGISTER		
A/D	REGISTER			dr	0	0	0	0	0	0	0	0	0	0	0

# MOVES

## Move Address Space (MC68010, MC68020, MC68030, MC68040, CPU32)

# MOVES

### Instruction Fields:

Size field—Specifies the size of the operation.

- 00—Byte Operation
- 01—Word Operation
- 10—Long Operation

Effective Address field—Specifies the source or destination location within the alternate address space. Only memory alterable addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	—	—
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
—(An)	100	reg. number:An
(d <sub>16</sub> ,An)	101	reg. number:An
(d <sub>8</sub> ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	—	—
(d <sub>16</sub> ,PC)	—	—
(d <sub>8</sub> ,PC,Xn)	—	—

### MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn] ,od)	110	reg. number:An
([bd,An],Xn ,od)	110	reg. number:An

(bd,PC,Xn)*	—	—
([bd,PC,Xn] ,od)	—	—
([bd,PC],Xn ,od)	—	—

\*Available for the CPU32.

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 > .



# MOVES

## Move Address Space (MC68010, MC68020, MC68030, MC68040, CPU32)

# MOVES

### NOTE

The value stored is undefined for either of the two following examples with the same address register as both source and destination.

```
MOVES.x An,(An) +
MOVES.x An,D(An)
```

The current implementations of the MC68010, MC68020, MC68030, and MC68040 store the incremented or decremented value of An. Check the following code sequence to determine what value is stored for each case.

```
MOVEA.L #$1000,A0
MOVES.L A0,(A0) +
MOVES.L A0,D(A0)
```

Because the MC68040 implements a merged instruction and data space, the MC68040's integer unit into data references (SFC/DFC = 5 or 1) translates MOVES accesses to the OinstructionO address spaces (SFC/DFC = 6 or 2). The data memory unit handles these translated accesses as normal data accesses. If the access fails due to an ATC fault or a physical bus error, the resulting access error stack frame contains the converted function code in the TM field for the faulted access. To maintain cache coherency, MOVES accesses to write the OinstructionO address space must be preceded by invalidation of the instruction cache line containing the referenced location.

# ORI to SR

## Inclusive-OR Immediate to the Status Register (M68000 Family)

# ORI to SR

**Operation:** If Supervisor State  
Then Source V SR → SR  
Else TRAP

**Assembler**

**Syntax:** ORI # < data > ,SR

**Attributes:** Size = (Word)

**Description:** Performs an inclusive-OR operation of the immediate operand and the status register's contents and stores the result in the status register. All implemented bits of the status register are affected.

**Condition Codes:**

X	N	Z	V	C
*	*	*	*	*

X—Set if bit 4 of immediate operand is one; unchanged otherwise.

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.

**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
16—BIT WORD DATA															

# RESET

## Reset External Devices (M68000 Family)

# RESET

**Operation:** If Supervisor State  
                   Then Assert  $\overline{\text{RESET}}$  ( $\overline{\text{RSTO}}$ , MC68040 Only) Line  
                   Else TRAP

**Assembler  
 Syntax:** RESET

**Attributes:** Unsized

**Description:** Asserts the  $\overline{\text{RSTO}}$  signal for 512 (124 for MC68000, MC68EC000, MC68HC000, MC68HC001, MC68008, MC68010, and MC68302) clock periods, resetting all external devices. 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

# RTE

## Return from Exception (M68000 Family)

# RTE

**Operation:** If Supervisor State  
 Then (SP) → SR; SP + 2 → SP; (SP) → PC; SP + 4 → SP; Restore State and Deallocate Stack According to (SP)  
 Else TRAP

**Assembler**

**Syntax:** RTE

**Attributes:** Unsized

**Description:** Loads the processor state information stored in the exception stack frame located at the top of the stack into the processor. The instruction examines the stack format field in the format/offset word to determine how much information must be restored.

**Condition Codes:**

Set according to the condition code bits in the status register value restored from the stack.

**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	1

**Format/Offset Word (in Stack Frame):**

MC68010, MC68020, MC68030, MC68040, CPU32

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

**Format Field of Format/Offset Word:**

Contains the format code, which implies the stack frame size (including the format/offset word). For further information, refer to **Appendix B Exception Processing Reference**.

# STOP

## Load Status Register and Stop (M68000 Family)

# STOP

**Operation:** If Supervisor State  
                   Then Immediate Data → SR; STOP  
                   Else TRAP

**Assembler Syntax:** STOP # < data >

**Attributes:**            Unsize

**Description:** Moves the immediate operand into the status register (both user and supervisor portions), advances the program counter to point to the next instruction, and stops the fetching and executing of instructions. A trace, interrupt, or reset exception causes the processor to resume instruction execution. A trace exception occurs if instruction tracing is enabled (T0 = 1, T1 = 0) when the STOP instruction begins execution. If an interrupt request is asserted with a priority higher than the priority level set by the new status register value, an interrupt exception occurs; otherwise, the interrupt request is ignored. External reset always initiates 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.