

00

**MOVE
MOVEA
ORI
BCLR
CMPI**

**WEEK 1
11/9 - 11/15**

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.

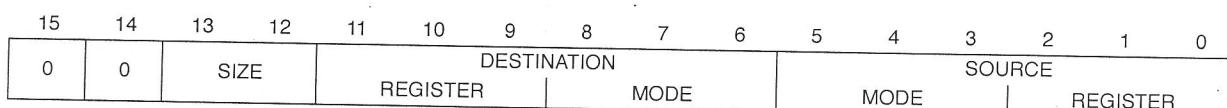
N — Set if the result is negative; cleared otherwise.

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

V — Always cleared.

C — Always cleared.

Instruction Format:



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 ₁₆ ,An)	101	reg. number:An
(d ₈ ,An,Xn)	110	reg. number:An

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

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

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

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												
16-BIT WORD DATA								MODE		REGISTER												
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
Dn	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
-(An)	100	reg. number:An
(d ₁₆ ,An)	101	reg. number:An
(d ₈ ,An,Xn)	110	reg. number:An

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

BCLR**Test a Bit and Clear
(M68000 Family)****BCLR**

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

Assembler Syntax: BCLR Dn, <ea>
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 ₁₆ ,An)	101	reg. number:An
(d ₈ ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d ₁₆ ,PC)	—	—
(d ₈ ,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			
0	0	0	0	0	0	0	0			MODE		REGISTER			

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 ₁₆ ,An)	101	reg. number:An
(d ₈ ,An,Xn)	110	reg. number:An

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

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					
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
Dn	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
-(An)	100	reg. number:An
(d ₁₆ ,An)	101	reg. number:An
(d ₈ ,An,Xn)	110	reg. number:An

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

*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.

11

**ADD
ADDA
MULS
LSL, LSR
ASL, ASR
ROL, ROR**

**WEEK 2
11/16 – 11/22**

ADD**Add**
(M68000 Family)**ADD****Operation:** Source + Destination → Destination**Assembler Syntax:** ADD < ea >, Dn
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
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 ₁₆ ,An)	101	reg. number:An
(d ₈ ,An,Xn)	110	reg. number:An

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

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

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

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 ₁₆ ,An)	101	reg. number:An
(d ₈ ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	111	100
(d ₁₆ ,PC)	111	010
(d ₈ ,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									
0	REGISTER DI				1	SIZE	0	0	0	0	0	0	0	0	REGISTER				
															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 ₁₆ ,An)	101	reg. number:An
(d ₈ ,An,Xn)	110	reg. number:An

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

LSL, LSR

Logical Shift (M68000 Family)

LSL, LSR

Operation: Destination Shifted By Count → Destination

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

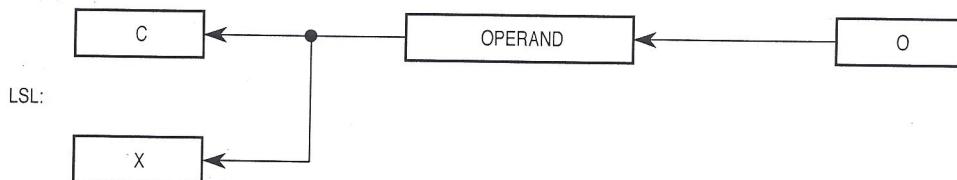
Attributes: Size = (Byte, Word, Long)

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:

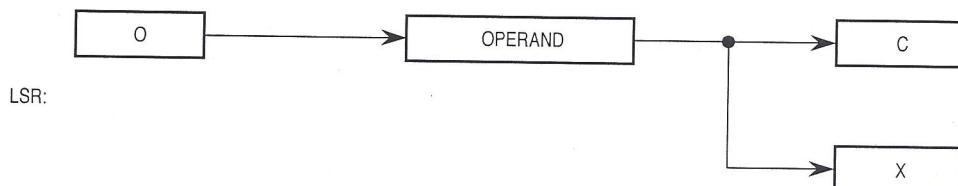
1. Immediate—The shift count (1 – 8) is specified in the instruction.
2. Register—The shift count is the value in the data register specified in the instruction modulo 64.

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	REGISTERS					

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 ₁₆ ,An)	101	reg. number:An
(d ₈ ,An,Xn)	110	reg. number:An

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



ASL, ASR

Arithmetic Shift (M68000 Family)

ASL, ASR

Operation: Destination Shifted By Count → Destination

Assembler Syntax:

- ASd Dx,Dy
- ASd # < data >,Dy
- ASd < ea >
- where d is direction, L or R

Attributes: Size = (Byte, Word, Long)

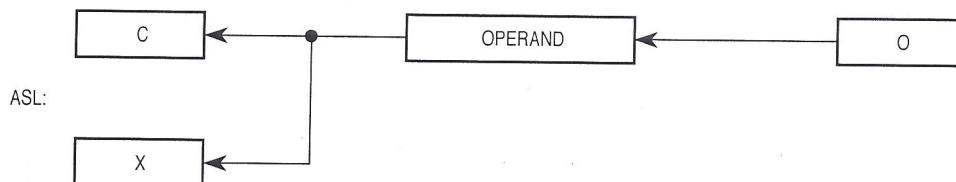
Description: Arithmetically shifts the bits of the operand in the direction (L or R) specified.

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.

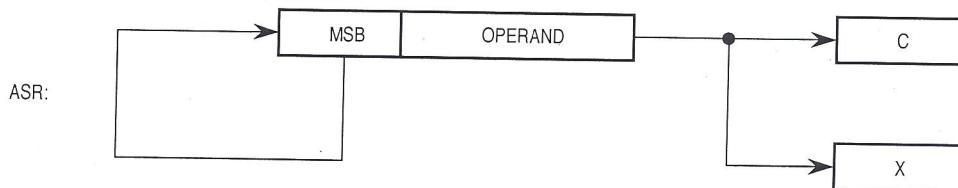
The size of the operation can be specified as byte, word, or long. An operand in memory can be shifted one bit only, and the operand size is restricted to a word.

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 ₁₆ ,An)	101	reg. number:An
(d ₈ ,An,Xn)	110	reg. number:An

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

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

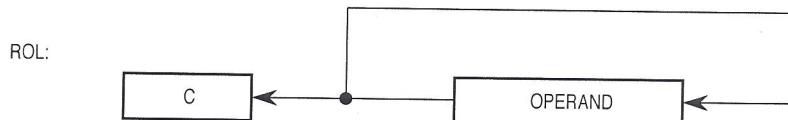
Attributes: Size = (Byte, Word, Long)

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:

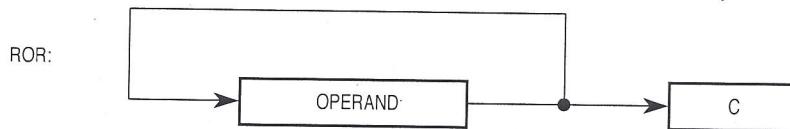
1. Immediate—The rotate count (1 – 8) is specified in the instruction.
2. Register—The rotate count is the value in the data register specified in the instruction, modulo 64.

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:**

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

Instruction Fields:

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 ₁₆ ,An)	101	reg. number:An
(d ₈ ,An,Xn)	110	reg. number:An

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

01

**MOVEM
LEA
NEG
Bcc
BRA
JSR
RTS**

**WEEK 3
11/23 – 11/29**

MOVEM**Move Multiple Registers
(M68000 Family)****MOVEM**

Operation: Registers → Destination; Source → Registers

Assembler Syntax: MOVEM < list > , < ea >
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 ₁₆ ,An)	101	reg. number:An
(d ₈ ,An,Xn)	110	reg. number:An

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

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

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 ₁₆ ,An)	101	reg. number:An
(d ₈ ,An,Xn)	110	reg. number:An

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

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 ₁₆ ,An)	101	reg. number:An
(d ₈ ,An,Xn)	110	reg. number:An

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

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
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).

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 ₁₆ ,An)	101	reg. number:An
(d ₈ ,An,Xn)	110	reg. number:An

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

RTS**Return from Subroutine
(M68000 Family)****RTS****Operation:** $(SP) \rightarrow PC; SP + 4 \rightarrow SP$ **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

10

SUB
DIVS
OR
EOR
CMP

WEEK 4
12/30 - 12/6

SUB

Subtract
(M68000 Family)

SUB

Operation: Destination – Source → Destination

Assembler Syntax: SUB < ea >, Dn
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	Dn - < ea > → Dn
100	101	110	< ea > - Dn → < ea >

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
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 ₁₆ ,An)	101	reg. number:An
(d ₈ ,An,Xn)	110	reg. number:An

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

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

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.

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 ₁₆ ,An)	101	reg. number:An
(d ₈ ,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	111	100
(d ₁₆ ,PC)	111	010
(d ₈ ,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
0		REGISTER Dq		1	SIZE	0	0	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 ₁₆ ,An)	101	reg. number:An
(d ₈ ,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 ₁₆ ,PC)	111	010
(d ₈ ,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

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.

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

Operation: Source V Destination → Destination

Assembler Syntax: OR < ea >, Dn
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 ₁₆ , An)	101	reg. number:An
(d ₈ , An, Xn)	110	reg. number:An

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

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 ₁₆ ,An)	101	reg. number:An
(d ₈ ,An,Xn)	110	reg. number:An

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

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.

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:

WORD

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 any of the eight data registers.

Opmode field

Byte	Word	Long	Operation
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 ₁₆ ,An)	101	reg. number:An
(d ₈ ,An,Xn)	110	reg. number:An

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

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 ₁₆ ,An)	101	reg. number:An
(d ₈ ,An,Xn)	110	reg. number:An

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