ANNEXE

Jeu d’instructions du microprocesseur 68000 Motorola

INSTRUCTION SET DETAILS

This appendix contains detailed information about each instruction in the M68000 instruction set. Instruction descriptions are arranged in alphabetical order with the mnemonic heading set in large bold type for easy reference.

B.1 ADDRESSING CATEGORIES

Effective address modes can be categorized by the ways in which they are used. The following classifications are used in the instruction definitions.

Data — If an effective address mode is used to refer to data operands, it is considered a data addressing effective address mode.

Memory — If an effective address mode is used to refer to memory operands, it is considered a memory addressing effective address mode.

Alterable — If an effective address mode is used to refer to alterable (writeable) operands, it is considered an alterable addressing effective address mode.

Control — If an effective address mode is used to refer to memory operands without associated sizes, it is considered a control addressing effective address mode.

Table B-1 shows the categories of each of the effective address modes.

Table B-1. Effective Address Mode Categories

Address Modes	Mode	Register	Data	Memory	Control	Alterable	Assembler Syntax
Data Register Direct	000	reg. no.	X	—	—	X	Dn
Address Register Direct	001	reg. no.	—	—	—	X	An
Address Register Indirect	010	reg. no.	X	X	X	X	(An)
Address Register Indirect with Postincrement	011	reg. no.	X	X	—	X	(An) +
Address Register Indirect with Predecrement	100	reg. no.	X	X	—	X	− (An)
Address Register Indirect with Displacement	101	reg. no.	X	X	X	X	(d_{16},An) or $d_{16}(An)$
Address Register Indirect with Index	110	reg. no.	X	X	X	X	(d_8,An,Xn) or $d_8(An,Xn)$
Absolute Short	111	000	X	X	X	X	(xxx).W
Absolute Long	111	001	X	X	X	X	(xxx).L
Program Counter Indirect with Displacement	111	101	X	X	X	—	(d_{16},PC) or $d_{16}(PC)$
Program Counter Indirect with Index	111	011	X	X	X	—	(d_8,PC,Xn) or $d_8(PC,Xn)$
Immediate	111	100	X	X	—	—	#(data)

These categories can be combined to define additional, more restrictive classifications. For example, the instruction descriptions use such classifications as memory alterable and data alterable. Memory alterable memory refers to addressing modes that are both alterable and memory addresses. Data alterable refers to addressing modes that are both data and alterable.

B.2 INSTRUCTION DESCRIPTION

The instruction descriptions in this section contain detailed information about the instructions. The format of these descriptions is shown in Figure B-1.

B.3 OPERATION DESCRIPTION DEFINITIONS

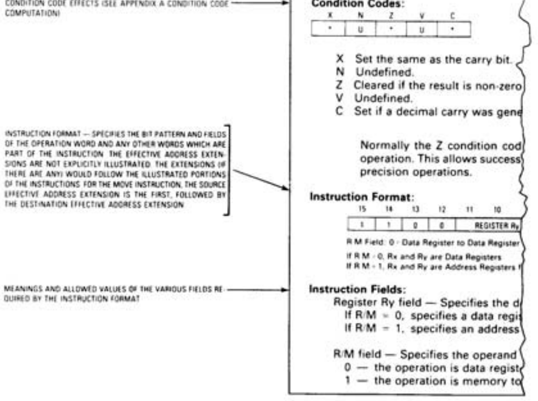
The following notation is used in the instruction descriptions.

OPERANDS

An	—Address register
Dn	—Data register
Rn	—Any data or address register
PC	—Program counter
SR	—Status register
CCR	—Condition codes (low-order byte of status)
SSP	—Supervisor stack pointer
USP	—User stack pointer
SP	—Active stack pointer (equivalent to A7)
X	—Extend operand condition code
N	—Negative condition code
Z	—Zero condition code
V	—Overflow condition code
C	—Carry condition code
Immediate data	—Immediate data from the instruction
d	—Address displacement
Source	—Source contents
Destination	—Destination contents
Vector	—Location of exception vector
ea	—Any valid effective address

SUBFIELDS AND QUALIFIERS

⟨bit⟩ of ⟨operand⟩	Selects a single bit of the operand
(⟨operand⟩)	The contents of the referenced location
$\langle operand\rangle_{10}$	The operand is binary coded decimal; operations are to be performed in decimal
(⟨address register⟩) − (⟨address register⟩) (⟨address register⟩) +	The register indirect operator, which indicates that the operand register points to the memory location of the instruction operand
#xxx or #⟨data⟩	Immediate data operand from the instruction



INSTRUCTION NAME
OPERATION DESCRIPTION
ASSEMBLER SYNTAX FOR THIS INSTRUCTION
TEXT DESCRIPTION OF INSTRUCTION OPERATION
CONDITION CODE EFFECTS (SEE APPENDIX A CONDITION CODE COMPUTATION)
INSTRUCTION FORMAT — SPECIFIES THE BIT PATTERN AND FIELDS OF THE OPERATION WORD AND ANY OTHER WORDS WHICH ARE PART OF THE INSTRUCTION. THE EFFECTIVE ADDRESS EXTENSIONS ARE NOT EXPLICITLY ILLUSTRATED. THE EXTENSIONS (IF THERE ARE ANY) WOULD FOLLOW THE ILLUSTRATED PORTIONS OF THE INSTRUCTIONS. FOR THE MOVE INSTRUCTION, THE SOURCE EFFECTIVE ADDRESS EXTENSION IS THE FIRST, FOLLOWED BY THE DESTINATION EFFECTIVE ADDRESS EXTENSION
MEANINGS AND ALLOWED VALUES OF THE VARIOUS FIELDS REQUIRED BY THE INSTRUCTION FORMAT

ABCD Add

Operation: $Source_{10}$ + Destination

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

Attributes: Size = (Byte)

Description: Adds the source opera
and stores the result in the destinat
decimal arithmetic. The operands,
different ways:

1. Data register to data register:
in the instruction.
2. Memory to memory: The op
mode using the address

This operation is a byte operation

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 non-zero
V Undefined.
C Set if a decimal carry was gene

Normally the Z condition cod
operation. This allows success
precision operations.

Instruction Format:

15	14	13	12	11	10
1	1	0	0	REGISTER Ry	

R/M Field: 0 - Data Register to Data Register
If R/M - 0, Rx and Ry are Data Registers
If R/M - 1, Rx and Ry are Address Registers f

Instruction Fields:
Register Ry field — Specifies the d
If R/M = 0, specifies a data regi
If R/M = 1, specifies an address

R/M field — Specifies the operand
0 — the operation is data regist
1 — the operation is memory to

FIGURE B.1 Instruction Description Format

BINARY OPERATIONS

These operations are written ⟨operand⟩ ⟨op⟩ ⟨operand⟩, where ⟨op⟩ is one of the following:

→	The left operand is moved to the right operand
↔	The two operands are exchanged
+	The operands are added
−	The right operand is subtracted from the left operand
*	The operands are multiplied
/	The first operand is divided by the second operand
Λ	The operands are logically ANDed
V	The operands are logically ORed
⟨	Relational test, true if the left operand is less than the right operand
⟩	Relational test, true if the left operand is greater than the right operand
shifted by rotated by	The left operand is shifted or rotated by the number of positions specified by the right operand

UNARY OPERATIONS

~⟨operand⟩	The operand is logically complemented
⟨operand⟩sign extended	The operand is sign extended; bits equal to the high-order bit of the operand are inserted to extend the operand to the left
⟨operand⟩tested	The operand is compared to zero; the condition codes are set to the result.

OTHER OPERATIONS

TRAP — Equivalent to: SSP - 2 → SSP; format/offset word → (SSP); SSP - 4 → SSP; PC → (SSP); SSP - 2 → ; SR → (SSP); (vector) → PC

STOP — Enter the stopped state, waiting for interrupts

if ⟨condition⟩ then ⟨operations⟩ else ⟨operations⟩;

The condition is tested. If true, the operations after "then" are performed. If the condition is false and the optional "else" clause is present, the "else" clause operations are performed. If the condition is false and the "else" clause is omitted, the instruction performs no operation.

The semicolon is used to separate operations and terminate the if/then/else operation.

ABCD — Add Decimal with Extend — ABCD

Operation: Source$_{10}$ + Destination$_{10}$ + X ♦ Destination

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

Attributes: Size = (Byte)

Description: 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 BCD 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 non-zero. 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.

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 — 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			OP-MODE			EFFECTIVE ADDRESS					
										MODE			REGISTER		

Instruction Fields:
Register field — Specifies any of the eight data registers.
Op-Mode field —

Byte	Word	Long	Operation
000	001	010	⟨ea⟩+⟨Dn⟩ ♦ ⟨n⟩
100	101	110	⟨Dn⟩+⟨ea⟩ ♦ ⟨ea⟩

ADD — Add — ADD

Effective Address Field — Determines addressing mode:

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

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

*Word and Long only.

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

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)	—	—

Notes:
1. The Dn mode is used when the destination is a data register; the destination ⟨ea⟩ mode is invalid for a data register.
2. ADDA is used when the destination is an address register. ADDI and ADDQ are used when the source is immediate data. Most assemblers automatically make this distinction.

ADDA — Add Address — ADDA

Operation: Source + Destination ♦ Destination

Assembler Syntax: 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			OP-MODE			EFFECTIVE ADDRESS					
										MODE			REGISTER		

Op-Mode Field:

Word	Long	Operation
011	111	(⟨ea⟩) + (⟨An⟩) ♦ ⟨An⟩

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

Addressing Mode	Mode	Register
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

ADDI — Add Immediate — 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		
WORD DATA (16 BITS)								BYTE DATA (8 BITS)							
LONG DATA (32 BITS)															

Instruction Fields:

Size field — Specifies the size of the operation:
00 — Byte operation
01 — Word operation
10 — Long operation

ADDI — Add Immediate — ADDI

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

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)	—	—

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 — ADDQ

Operation: Immediate Data + Destination → Destination

Assembler Syntax: ADDQ #⟨data⟩,⟨ea⟩

Attributes: Size = (Byte, Word, Long)

Description: Adds an immediate value of 1 to 8 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		

Instruction Fields:

Data field — Three bits of immediate data, 0-7 (with the immediate value 0 representing a value of 8).
Size field — Specifies the size of the operation:
00 — Byte operation
01 — Word operation
10 — Long operation

ADDQ — Add Quick — ADDQ

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

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

*Word and Long only.

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#⟨data⟩	—	—
(d_{16},PC)	—	—
(d_8,PC,Xn)	—	—

ADDX Add Extended ADDX

Operation: Source + Destination + X ♦ Destination

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

Attributes: Size = (Byte, Word, Long)

Description: Adds the source operand to the destination operand along with the extend bit 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 non-zero. 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.

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		

ADDX Add Extended ADDX

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 AND

Operation: Source∧Destination ♦ Destination

Assembler Syntax:
AND ⟨ea⟩,Dn
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
										EFFECTIVE ADDRESS					
1	1	0	0	REGISTER			OP-MODE			MODE			REGISTER		

Instruction Fields:

Register field — Specifies any of the eight data registers
Op-Mode field —

Byte	Word	Long	Operation
000	001	010	(⟨ea⟩)∧(⟨Dn⟩) ♦ Dn
100	101	110	(⟨Dn⟩)∧(⟨ea⟩) ♦ ea

AND And Logical AND

Effective Address field — Determines addressing mode:
If the location specified is a source operand only data addressing modes are allowed as shown:

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

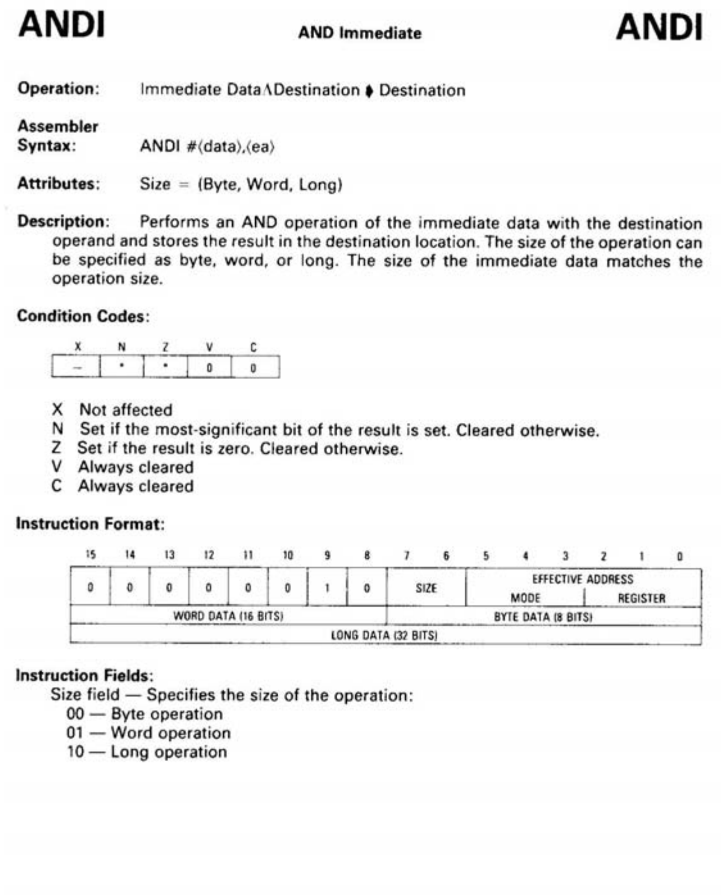
If the location specified is a destination operand only memory alterable addressing modes are allowed as shown:

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)	—	—

Notes:

1. The Dn mode is used when the destination is a data register; the destination ⟨ea⟩ mode is invalid for a data register.
2. Most assemblers use ANDI when the source is immediate data.



ANDI — AND Immediate — ANDI

Operation: Immediate Data∧Destination ➧ 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

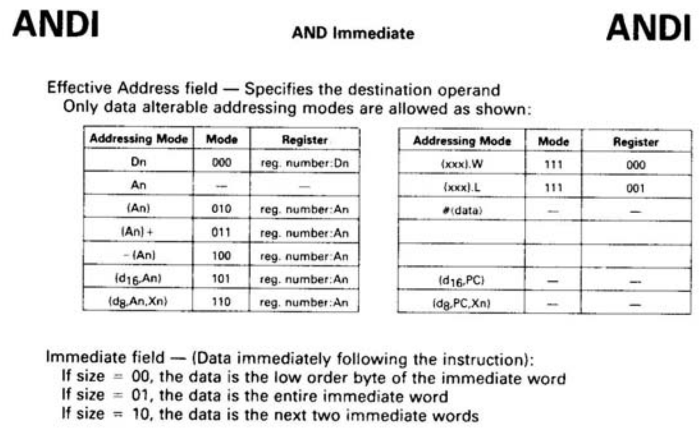
X Not affected
N Set if the most-significant bit of the result is set. Cleared otherwise.
Z Set if the result is zero. Cleared otherwise.
V Always cleared
C Always cleared

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		
WORD DATA (16 BITS)								BYTE DATA (8 BITS)							
LONG DATA (32 BITS)															

Instruction Fields:

Size field — Specifies the size of the operation:
00 — Byte operation
01 — Word operation
10 — Long operation



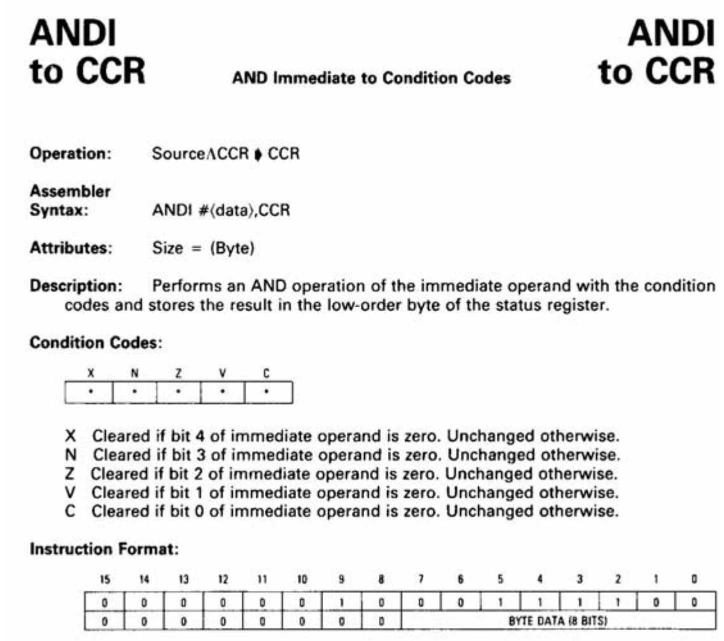
ANDI — AND Immediate — ANDI

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

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)	—	—

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 — AND Immediate to Condition Codes — ANDI to CCR

Operation: Source∧CCR ➧ 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.

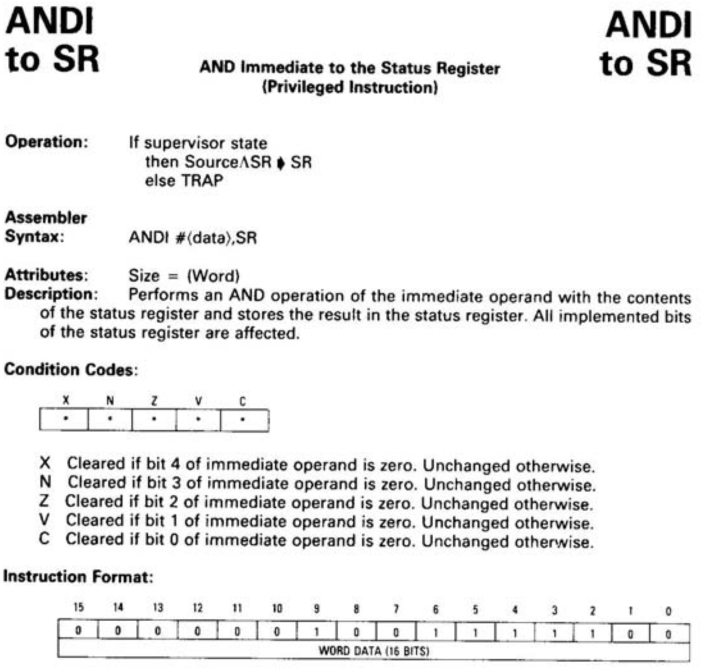
Condition Codes:

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
0	0	0	0	0	0	0	0	BYTE DATA (8 BITS)							



ANDI to SR — AND Immediate to the Status Register (Privileged Instruction) — ANDI to SR

Operation: If supervisor state
then Source∧SR ➧ SR
else TRAP

Assembler Syntax: ANDI #⟨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:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	0	0	1	1	1	1	1	0	0
WORD DATA (16 BITS)															

Operation: Destination Shifted by ⟨count⟩ ➧ Destination

Assembler Syntax:
ASd Dx,Dy
ASd #⟨data⟩,Dy
ASd ⟨ea⟩
where d is direction, L or R

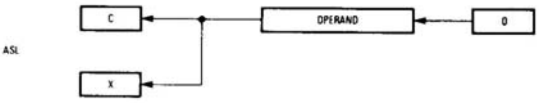
Attributes: Size = (Byte, Word, Long)

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:

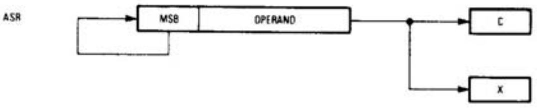
1. Immediate — The shift count is specified in the instruction (shift range, 1-8).
2. Register — The shift count is the value in the data register specified in instruction modulo 64.

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.



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 (Register Shifts):

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; value of zero represents a count of 8.
 If i/r = 1, this field specifies the data register that 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

Instruction Format (Memory Shifts):

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

Instruction Fields (Memory Shifts):

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

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

Addressing Mode	Mode	Register
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)	—	—

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

Assembler Syntax: Bcc ⟨label⟩

Attributes: Size = (Byte, Word)

Description: If the specified condition is true, program execution continues at location (PC) + displacement. The PC contains the address of the instruction word of the Bcc instruction plus two. The displacement is a twos complement integer that represents the relative distance in bytes from the current PC to the destination PC. If the 8-bit displacement field in the instruction word is zero, a 16-bit displacement (the word immediately following the instruction) is used. Condition code cc specifies one of the following conditions:

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

Condition Codes:
Not affected

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	CONDITION				8-BIT DISPLACEMENT							
16-BIT DISPLACEMENT IF 8-BIT DISPLACEMENT = $00															

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.

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 BCHG

Operation: ~(⟨number⟩ of Destination) ⧫ Z;
~(⟨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

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 (Bit Number Dynamic):

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

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

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)	—	—

*Long only; all others are byte only.

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 (Bit Number Static):

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

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)	—	—

*Long only; all others are byte only.

Bit Number field — Specifies the bit number

BCLR Test a Bit and Clear BCLR

Operation: ~(⟨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

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 (Bit Number Dynamic):

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

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

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)	—	—

*Long only; all others are byte only.

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 (Bit Number Static):

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

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)	—	—

*Long only; all others are byte only.

Bit Number field — Specifies the bit number

BKPT Breakpoint BKPT

Operation: Execute breakpoint acknowledge bus cycle;
Trap as illegal instruction

Assembler Syntax: BKPT #⟨data⟩

Attributes: Unsized

Description: This instruction is used to support the program breakpoint function for debug monitors and real-time hardware emulators, and the operation will be dependent on the implementation. Execution of this instruction will cause the MC68010 to run a breakpoint acknowledge bus cycle (all function codes driven high) and zeros on all address lines.

Whether the breakpoint acknowledge bus cycle is terminated with $\overline{\text{DTACK}}$, $\overline{\text{BERR}}$, or $\overline{\text{VPA}}$, the processor always takes an illegal instruction exception. During exception processing, a debug monitor can distinguish eight different software breakpoints by decoding the field in the BKPT instruction.

For the MC68000, MC68HC000, and MC68008, this instruction causes an illegal instruction exception but does not run the breakpoint acknowledge bus cycle.

Condition Codes:
Not affected

Instruction Format:

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

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

BRA Branch Always BRA

Operation: PC + d → PC

Assembler Syntax: BRA ⟨label⟩

Attributes: Size = (Byte, Word)

Description: Program execution continues at location (PC) + displacement. The PC 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 PC to the destination PC. If the 8-bit displacement field in the instruction word is zero, a 16-bit displacement (the 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															

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.

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 BSET

Operation: ~(⟨bit number⟩ of Destination) → Z;
1 → ⟨bit number⟩ of Destination

Assembler Syntax: BSET Dn,⟨ea⟩
BSET #⟨data⟩,⟨ea⟩

Attributes: Size = (Byte, Long)

Description: 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

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		

BSET Test a Bit and Set BSET

Instruction Fields (Bit Number Dynamic):
Register field — Specifies the data register that contains the bit number
Effective Address field — Specifies the destination location. Only data alterable addressing modes are allowed as shown:

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)	—	—

*Long only; all others are byte only.

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 (Bit Number Static):
Effective Address field — Specifies the destination location. Only data alterable addressing modes are allowed as shown:

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)	—	—

*Long only; all others are byte only.

Bit Number field — Specifies the bit number

BSR Branch to Subroutine BSR

Operation: SP − 4 → SP; PC → (SP); PC + d → PC

Assembler Syntax: BSR ⟨label⟩

Attributes: Size = (Byte, Word)

Description: Pushes the long word address of the instruction immediately following the BSR instruction onto the system stack. The PC 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 PC to the destination PC. If the 8-bit displacement field in the instruction word is zero, a 16-bit displacement (the 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															

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

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 BTST

Operation: − (⟨bit number⟩ of Destination) → Z;

Assembler Syntax: BTST Dn,⟨ea⟩
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

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
										EFFECTIVE ADDRESS					
0	0	0	0	REGISTER			1	0	0	MODE			REGISTER		

BTST Test a Bit BTST

Instruction Fields (Bit Number Dynamic):
Register field — Specifies the data register that contains the bit number
Effective Address field — Specifies the destination location. Only data addressing modes are allowed as shown:

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

*Long only; all others are byte only.

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
										EFFECTIVE ADDRESS					
0	0	0	0	1	0	0	0	0	0	MODE			REGISTER		
0	0	0	0	0	0	0	0	BIT NUMBER							

Instruction Fields (Bit Number Static):
Effective Address field — Specifies the destination location. Only data addressing modes are allowed as shown:

Addressing Mode	Mode	Register
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

Bit Number field — Specifies the bit number

CHK Check Register Against Bounds CHK

Operation: If Dn < 0 or Dn > Source then TRAP

Assembler Syntax: CHK ⟨ea⟩,Dn

Attributes: Size = (Word)

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 Dn < 0; cleared if Dn > 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
										EFFECTIVE ADDRESS					
0	1	0	0	REGISTER			1	1	0	MODE			REGISTER		

Instruction Fields:
Register field — Specifies the data register that contains the value to be checked

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

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

CLR — Clear an Operand — 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		

Instruction Fields:

Size field — Specifies the size of the operation
- 00 — Byte operation
- 01 — Word operation
- 10 — Long operation

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

Addressing Mode	Mode	Register
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)	—	—

NOTE

In the MC68000, MC68HC000, and MC68008 a memory destination is read before it is cleared.

CMP — Compare — 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			OP-MODE			EFFECTIVE ADDRESS					
										MODE			REGISTER		

Instruction Fields:

Register field — Specifies the destination data register
Op-Mode field —

Byte	Word	Long	Operation
000	001	010	⟨⟨Dn⟩⟩ – ⟨⟨ea⟩⟩

CMP — Compare — CMP

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

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

*Word and Long only.

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

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 — CMPA

Operation: Destination – Source

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:

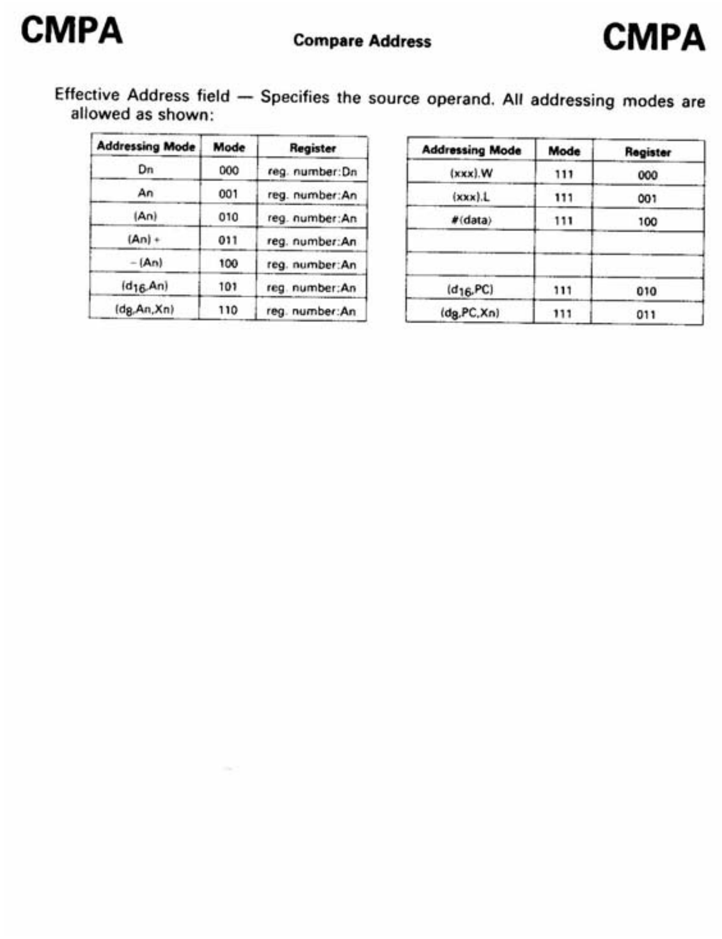
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	REGISTER			OP-MODE			EFFECTIVE ADDRESS					
										MODE			REGISTER		

Op-Mode Field:

Word	Long	Operation

Instruction Fields:

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

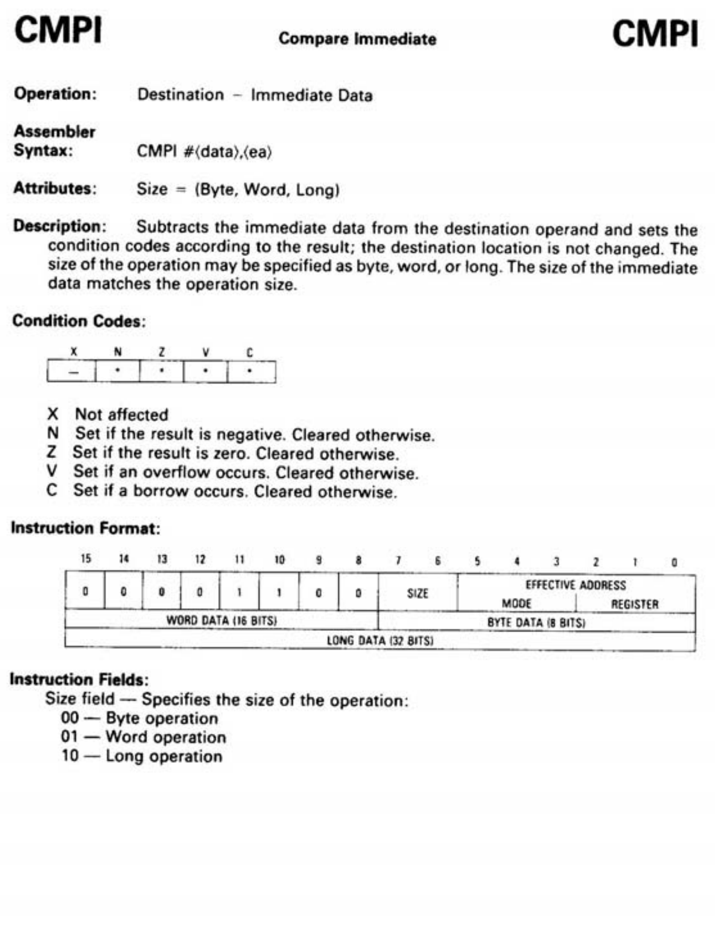


CMPA Compare Address CMPA

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

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



CMPI Compare Immediate CMPI

Operation: Destination − Immediate Data

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
—	*	*	*	*

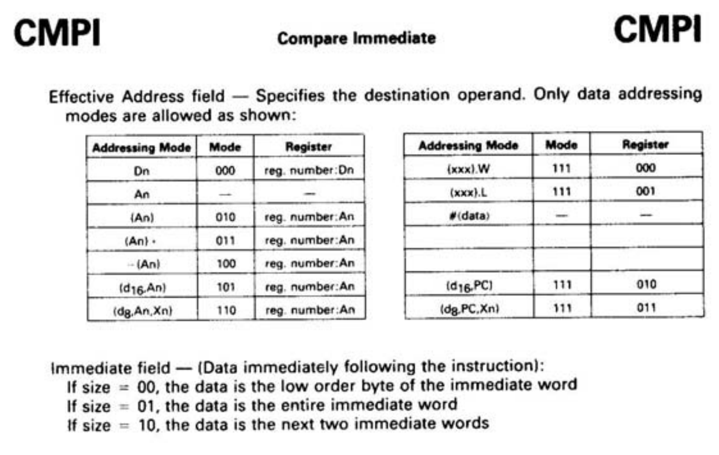
X Not affected
N Set if the result is negative. Cleared otherwise.
Z Set if the result is zero. Cleared otherwise.
V Set if an overflow occurs. Cleared otherwise.
C Set if a borrow occurs. Cleared otherwise.

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		
WORD DATA (16 BITS)								BYTE DATA (8 BITS)							
LONG DATA (32 BITS)															

Instruction Fields:

Size field — Specifies the size of the operation:
00 — Byte operation
01 — Word operation
10 — Long operation



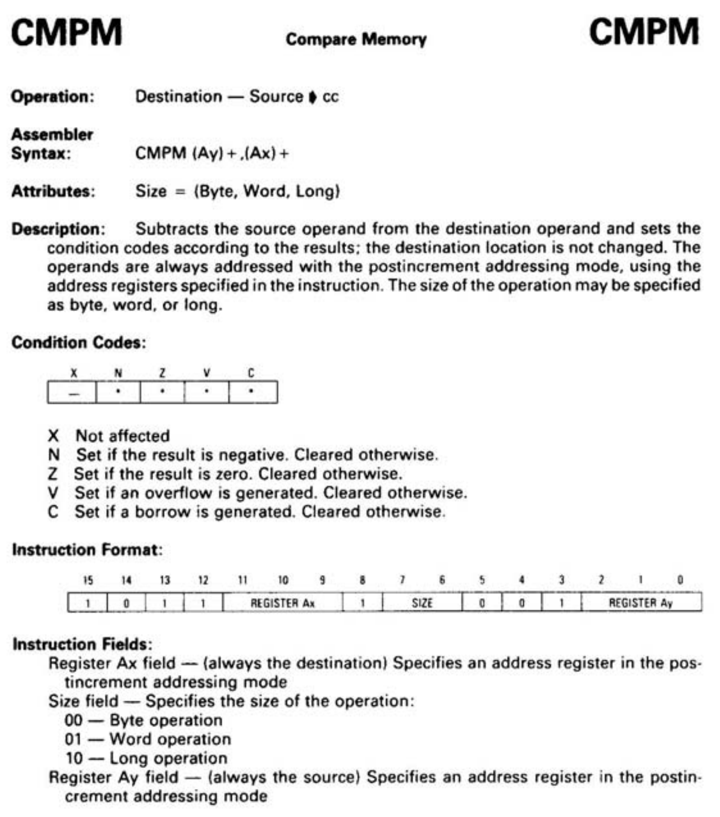
CMPI Compare Immediate CMPI

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

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

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 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 — DBcc

Operation: If condition false then (Dn − 1 → Dn;
If Dn ≠ − 1 then PC + d → 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 are decremented 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 PC plus the sign-extended 16-bit displacement. The value in the PC 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 PC to the destination PC.

Condition code cc specifies one of the following conditions:

CC	carry clear	0100	$\overline{C}$	LS	low or same	0011	$C+Z$
CS	carry set	0101	C	LT	less than	1101	$N \cdot \overline{V} + \overline{N} \cdot V$
EQ	equal	0111	Z	MI	minus	1011	N
F	never equal	0001	0	NE	not equal	0110	$\overline{Z}$
GE	greater or equal	1100	$N \cdot V + \overline{N} \cdot \overline{V}$	PL	plus	1010	$\overline{N}$
GT	greater than	1110	$N \cdot V \cdot \overline{Z} + \overline{N} \cdot \overline{V} \cdot \overline{Z}$	T	always true	0000	1
HI	high	0010	$\overline{C} \cdot \overline{Z}$	VC	overflow clear	1000	$\overline{V}$
LE	less or equal	1111	$Z + N \cdot \overline{V} + \overline{N} \cdot V$	VS	overflow set	1001	V

Condition Codes:
Not affected

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	CONDITION				1	1	0	0	1	REGISTER		
DISPLACEMENT (16 BITS)															

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

Notes:

1. 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".
2. Most assemblers accept DBRA for DBF for use when only a count terminates the loop (no condition is tested).
3. 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 — Signed Divide — DIVS

Operation: Destination/Source → Destination

Assembler Syntax: DIVS.W ⟨ea⟩,Dn 32/16 → 16r:16q

Attributes: Size = (Word)

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

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

Instruction Format (word form):

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

Instruction Fields:
Register field — Specifies any of the eight data registers. This field always specifies the destination operand.
Effective Address field — Specifies the source operand. Only data addressing modes are allowed as shown:

Addressing Mode	Mode	Register
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

NOTE

Overflow occurs if the quotient is larger than a 16-bit signed integer. The instruction checks for overflow at the start of execution. If the upper word of the dividend is greater than or equal to the divisor, the overflow bit is set in the condition codes, and the instruction terminates with the operands unchanged.

DIVU Unsigned Divide DIVU

Operation: Destination/Source ♦ Destination

Assembler Syntax: DIVU.W ⟨ea⟩,Dn 32/16 ♦ 16r:16q

Attributes: Size = (Word)

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

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

Instruction Format (word form):

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

DIVU Unsigned Divide DIVU

Instruction Fields:

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

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

Addressing Mode	Mode	Register
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

NOTE

Overflow occurs if the quotient is larger than a 16-bit signed integer. The instruction checks for overflow at the start of execution. If the upper word of the dividend is greater than or equal to the divisor, the overflow bit is set in the condition codes, and the instruction terminates with the operands unchanged.

EOR Exclusive OR Logical EOR

Operation: Source ⊕ Destination ♦ 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 form):

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

Instruction Fields:

Register field — Specifies any of the eight data registers
Op-Mode field —

Byte	Word	Long	Operation
100	101	110	(⟨ea⟩) ⊕ (⟨Dn⟩) ♦ ⟨ea⟩

EOR Exclusive OR Logical EOR

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

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)	—	—

NOTE

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

EORI — Exclusive OR Immediate — EORI

Operation:

Immediate Data ⊕ Destination ➧ 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		
WORD DATA (16 BITS)								BYTE DATA (8 BITS)							
LONG DATA (32 BITS)															

Instruction Fields:

Size field — Specifies the size of the operation:
00 — Byte operation
01 — Word operation
10 — Long operation

EORI — Exclusive OR Immediate — EORI

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

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)	—	—

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 — EORI to CCR

Operation:

Source ⊕ CCR ➧ 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
0	0	0	0	0	0	0	0	BYTE DATA (8 BITS)							

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

Operation: If supervisor state
then Source ⊕ SR ➧ SR

else TRAP

Assembler Syntax: EORI #⟨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
WORD DATA (16 BITS)															

EXG — Exchange Registers — EXG

Operation: Rx ↔ Ry

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

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	OP-MODE					REGISTER Ry		

Instruction Fields:

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

Op-Mode field — Specifies 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 — Sign Extend — EXT

Operation: Destination Sign-Extended → Destination

Assembler Syntax:

EXT.W Dn	Extend byte to word
EXT.L Dn	Extend word to long word

Attributes: Sizes (Word, Long)

Description: Extends a byte in a data register to a 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.

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	OP-MODE			0	0	0	REGISTER		

Instruction Fields:

Op-Mode field — Specifies the size of the sign-extension operation:
010 — Sign-extend low-order byte of data register to word
011 — Sign-extend low-order word of data register to long

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

ILLEGAL — Take Illegal Instruction Trap — ILLEGAL

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

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.

Only the MC68010 stores a four-word exception stack frame by first writing the exception vector offset and format code to the system stack. All processors write the PC, followed by the SR, to the system stack.

Condition Codes:
Not affected

Instruction Format:

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

JMP — Jump — 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:

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

Instruction Fields:

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

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

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

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

LEA — Load Effective Address — 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 are allowed as shown:

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

LINK — Link and Allocate — LINK

Operation: Sp − 4 ♦ Sp; An ♦ (SP);
SP ♦ An; SP + d ♦ SP

Assembler Syntax: LINK An, #⟨displacement⟩

Attributes: Size = Unsized

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 16-bit sign-extended displacement operand to the stack pointer. 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:

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 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 — 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

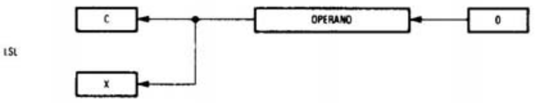
Attributes: Size = (Byte, Word, Long)

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:

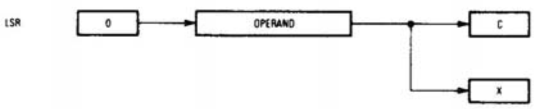
1. Immediate — The shift count (1-8) is specified in the instruction.
2. Register — The shift count is the value in the data register specified in the instruction modulo 64.

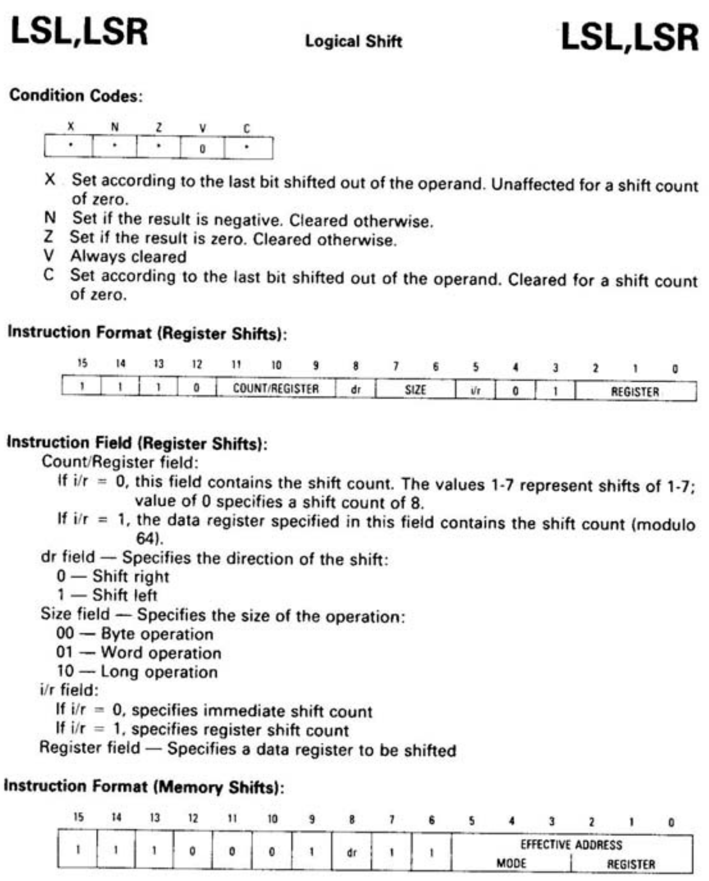
The size of the operation for register destinations may be specified as byte, word, or long. The contents of memory, ⟨ea⟩, can be shifted one bit only, and the operand size is restricted to a word.

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 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.

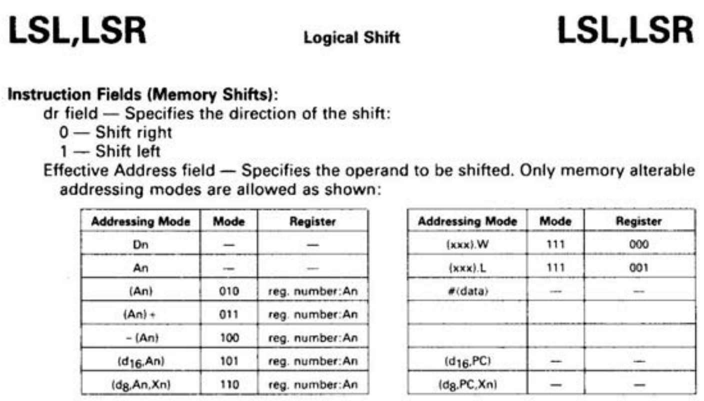
Instruction Format (Register Shifts):

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 Field (Register Shifts):
Count/Register field:
If i/r = 0, this field contains the shift count. The values 1-7 represent shifts of 1-7; value of 0 specifies a shift count of 8.
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

Instruction Format (Memory Shifts):

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

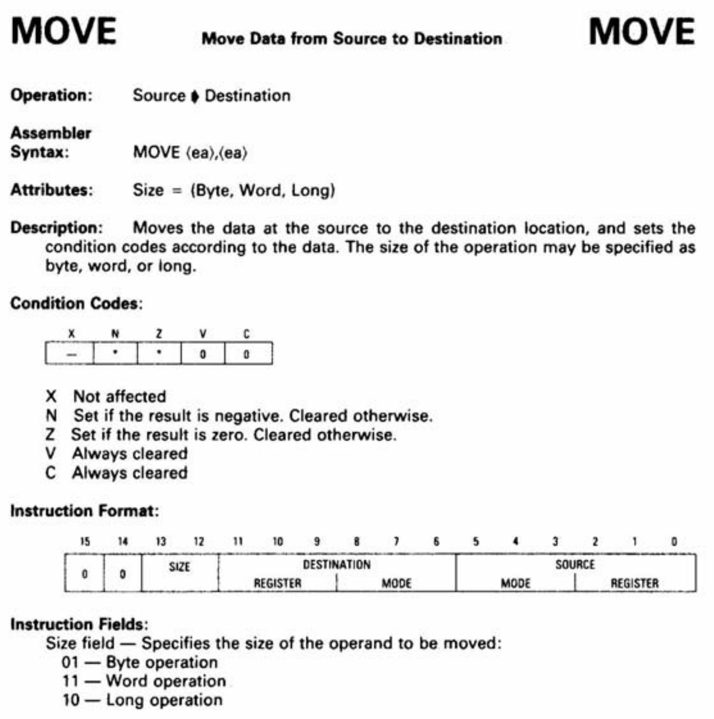


LSL,LSR Logical Shift LSL,LSR

Instruction Fields (Memory Shifts):
dr field — Specifies the direction of the shift:
0 — Shift right
1 — Shift left
Effective Address field — Specifies the operand to be shifted. Only memory alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register
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)	—	—



MOVE Move Data from Source to Destination 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:

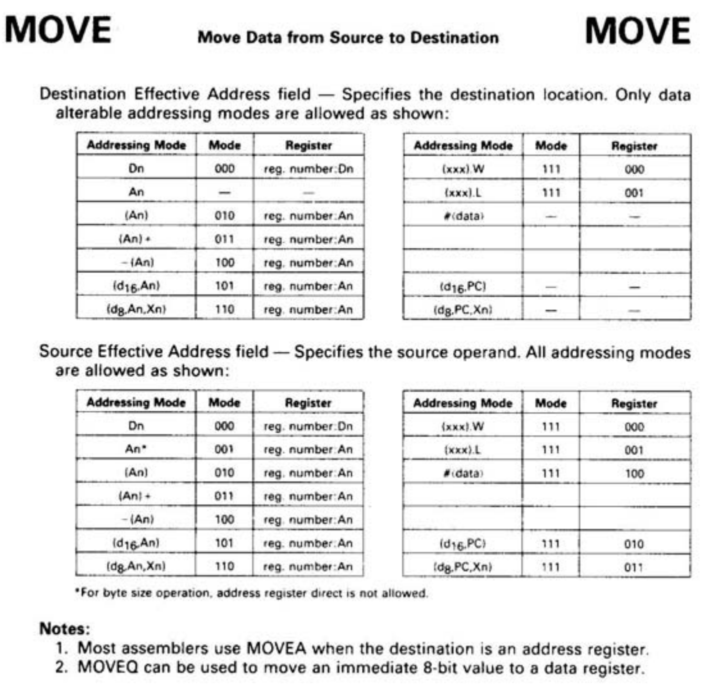
X	N	Z	V	C
—	*	*	0	0

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
				DESTINATION						SOURCE					
0	0	SIZE		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 MOVE

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

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)	—	—

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

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

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

Notes:
1. Most assemblers use MOVEA when the destination is an address register.
2. MOVEQ can be used to move an immediate 8-bit value to a data register.

MOVEA Move Address 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	SOURCE					
										MODE			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

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

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

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

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		

Instruction Fields:

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

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)	—	—

NOTE

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

MOVE from SR Move from the Status Register (Privileged Instruction – MC68010 Only) MOVE from SR

Operation: SR → Destination
MC68010 only:
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		

Instruction Fields:

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

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)	—	—

NOTE

Use the MOVE from CCR instruction to access only the condition codes. In the MC68000, MC68HC000, and MC68008, memory destination is read before it is written to.

MOVE to CCR Move to Condition Codes 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		

Instruction Fields:

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

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

NOTE

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

MOVE to SR

Move to the Status Register
(Priviledged Instruction)

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		

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

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	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_{16},An)	101	reg. number:An	(d_{16},PC)	111	010
(d_8,An,Xn)	110	reg. number:An	(d_8,PC,Xn)	111	011

MOVE USP

Move User Stack Pointer
(Privileged Instruction)

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 USP
1 — Transfer the USP to the address register
Register field — Specifies the address register for the operation

MOVEC

Move Control Register
(Privileged Instruction)

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
Register field — Specifies the register number
Control Register field — Specifies the control register

Hex	Control Register
000	Source Function Code (SFC) register
001	Destination Function Code (DFC) register
800	User Stack Pointer (USP)
801	Vector Base Register (VBR)

Any other code causes an illegal instruction exception.

MOVEM

Move Multiple Registers

Operation: Registers ♦ Destination
Source ♦ Registers

Assembler Syntax: MOVEM register list,‹ea›
MOVEM ‹ea›,register list

Attributes: Size = (Word, Long)

Description: 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 data register 0 to data register 7, then from address register 0 to address register 7.

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 address register 7 to address register 0, then from data register 7 to data register 0. When the instruction has completed, the decremented address register contains the address of the last operand stored.

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.

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															

MOVEM Move Multiple Registers MOVEM

Instruction Field:

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 are allowed as shown:

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)	—	—

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

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

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, both for control modes and for the 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

MOVEM Move Multiple Registers MOVEM

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

NOTE

An extra read bus cycle occurs for memory operands. This accesses an operand at one address higher than the last register image required.

MOVEP Move Peripheral Data MOVEP

Operation: Source ♦ Destination

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

Attributes: Size = (Word, Long)

Description: Moves data between a data register and alternate bytes within the address space (typically assigned to a peripheral), starting at the location specified and incrementing by two. This instruction is designed for 8-bit peripherals on a 16-bit data bus. 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. If the address is even, all the transfers are to or from the high order half of the data bus; if the address is odd, all the transfers are to or from the low order half of the data bus. The instruction also accesses alternate bytes on an 8-bit bus.

Example: Long transfer to/from an even address

Byte Organization in Register

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

Byte Organization in Memory (Low Address at Top)

15 8	7 0
HI-ORDER	
MID-UPPER	
MID-LOWER	
LOW-ORDER	

Example: Word transfer to/from an odd address

Byte Organization in Register

31 24	23 16	15 8	7 0
		HI-ORDER	LOW-ORDER

Byte Organization in Memory (Low Address at Top)

15 8	7 0
	HI-ORDER
	LOW-UPPER

MOVEP Move Peripheral Data 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			OP-MODE			0	0	1	ADDRESS REGISTER		
DISPLACEMENT (16 BITS)															

Instruction Fields:

Data Register field — Specifies the data register for the instruction

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

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

Displacement field — Specifies the displacement used in the operand address

MOVEQ Move Quick 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 — 8 bits of data, which are sign extended to a long operand

MOVES Move Address Space (Privileged Instruction) 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: 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, 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					
										MODE			REGISTER		
A/D	REGISTER			dr	0	0	0	0	0	0	0	0	0	0	0

Instruction Fields:

Size field — Specifies the size of the operation:
00 — Byte operation
01 — Word operation
10 — Long operation

MOVES Move Address Space (Privileged Instruction) MOVES

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

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)	—	—

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⟩

NOTE

For either of the two following examples with the same address register as both source and destination
MOVES.x An,(An)+
MOVES.x An,−(An)
the value stored is undefined. The current implementation of the MC68010 stores the incremented or decremented value of An.

MULS Signed Multiply MULS

Operation: Source * Destination → Destination

Assembler Syntax: MULS.W ⟨ea⟩,Dn 16 × 16 → 32

Attributes: Size = (Word)

Description: Multiplies two signed operands yielding a signed result. 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.

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 (word form):

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 addressing modes are allowed as shown:

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

MULU Unsigned Multiply MULU

Operation: Source * Destination → Destination

Assembler Syntax: MULU.W ⟨ea⟩,Dn 16 × 16 → 32

Attributes: Size = (Word)

Description: Multiplies two unsigned operands yielding an unsigned result. 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.

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 (word form):

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 are allowed as shown:

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

NBCD Negate Decimal with Extend NBCD

Operation: 0 – $(\text{Destination}_{10})$ – X → 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 BCD 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 non-zero. 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.

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 are allowed as shown:

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)	—	—

NEG Negate 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		

Instruction Fields:

Size field — Specifies the size of the operation:
00 — Byte operation
01 — Word operation
10 — Long operation

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

Addressing Mode	Mode	Register
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)	—	—

NEGX Negate with Extend NEGX

Operation: 0 – (Destination) – X → 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 non-zero. 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.

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

NEGX — Negate with Extend — NEGX

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

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)	—	—

NOP — No Operation — 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.

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 — 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		

Instruction Fields:
Size field — Specifies the size of the operation:
00 — Byte operation
01 — Word operation
10 — Long operation
Effective Address field — Specifies the destination operand. Only data alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register
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)	—	—

OR — Inclusive OR Logical — 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			OP-MODE			EFFECTIVE ADDRESS					
										MODE			REGISTER		

Instruction Fields:
Register field — Specifies any of the eight data registers
Op-Mode field —

Byte	Word	Long	Operation
000	001	010	(⟨ea⟩) V (⟨Dn⟩) → ⟨Dn⟩
100	101	110	(⟨Dn⟩) V (⟨ea⟩) → ⟨ea⟩

OR Inclusive OR Logical OR

Effective Address field — If the location specified is a source operand, only data addressing modes are allowed as shown:

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

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

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)	—	—

Notes:

1. If the destination is a data register, it must be specified using the destination Dn mode, not the destination ‹ea› mode.
2. Most assemblers use ORI when the source is immediate data.

ORI Inclusive OR 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:

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td rowspan="2">0</td><td rowspan="2">0</td><td rowspan="2">0</td><td rowspan="2">0</td><td rowspan="2">0</td><td rowspan="2">0</td><td rowspan="2">0</td><td rowspan="2">0</td><td colspan="2" rowspan="2">SIZE</td><td colspan="6">EFFECTIVE ADDRESS</td></tr>
<tr><td colspan="3">MODE</td><td colspan="3">REGISTER</td></tr>
<tr><td colspan="8">WORD DATA (16 BITS)</td><td colspan="8">BYTE DATA (8 BITS)</td></tr>
<tr><td colspan="16">LONG DATA (32 BITS)</td></tr>
</table>

Instruction Fields:

Size field — Specifies the size of the operation.
00 — Byte operation.
01 — Word operation.
10 — Long operation.

ORI Inclusive OR ORI

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

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)	—	—

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 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:

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td colspan="8">BYTE DATA (8 BITS)</td></tr>
</table>

ORI to SR — Inclusive OR Immediate to the Status Register (Privileged Instruction) — ORI to SR

Operation: If supervisor state
then Source V SR → SR
else TRAP

Assembler Syntax: ORI #⟨data⟩,SR

Attributes: Size − (Word)

Description: Performs an inclusive OR operation of the immediate operand and 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 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
WORD DATA (16 BITS)															

PEA — Push Effective Address — PEA

Operation: Sp − 4 → SP; ⟨ea⟩ → (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 word address.

Condition Codes:
Not affected

Instruction Format:

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

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

Addressing Mode	Mode	Register
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

RESET — Reset External Devices (Privileged Instruction) — RESET

Operation: If supervisor state
then Assert $\overline{\text{RESET}}$ Line
else TRAP

Assembler Syntax: RESET

Attributes: Unsized

Description: Asserts the $\overline{\text{RESET}}$ signal for 124 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

ROL ROR — Rotate (Without Extend) — ROL ROR

Operation: Destination Rotated by ⟨count⟩ → Destination

Assembler Syntax: ROd Dx,Dy
ROd #⟨data⟩,Dy
ROd ⟨ea⟩
where d is direction, L or R

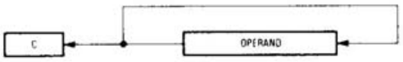
Attributes: Size = (Byte, Word, Long)

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:

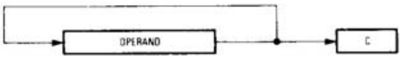
1. Immediate — The rotate count (1-8) is specified in the instruction.
2. Register — The rotate count is the value in the data register specified in the instruction, modulo 64.

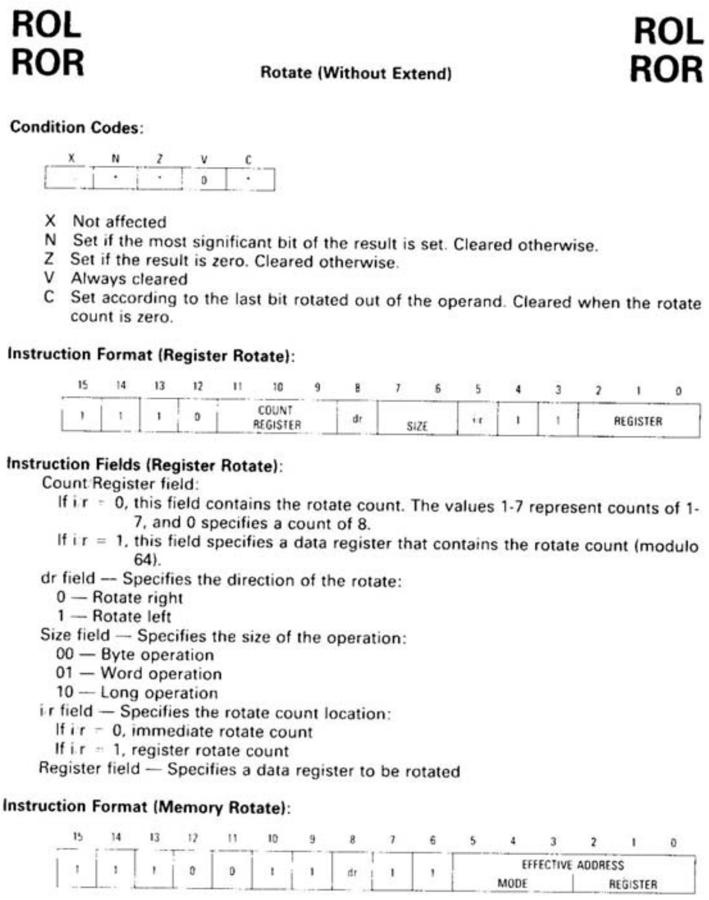
The size of the operation for register destinations is specified as byte, word, or long. The contents of memory, ⟨ea⟩; can be rotated one bit only, and operand size is restricted to a word.

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)

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):

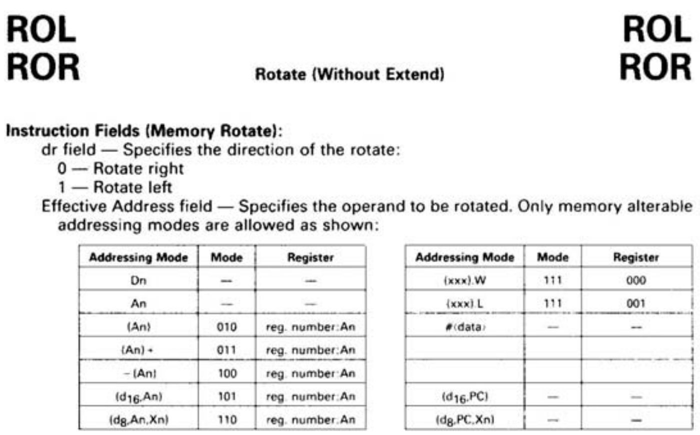
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	COUNT REGISTER			dr	SIZE		i/r	1	1	REGISTER		

Instruction Fields (Register Rotate):

Count/Register field:
If i/r = 0, this field contains the rotate count. The values 1-7 represent counts of 1-7, and 0 specifies a count of 8.
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

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		



ROL ROR

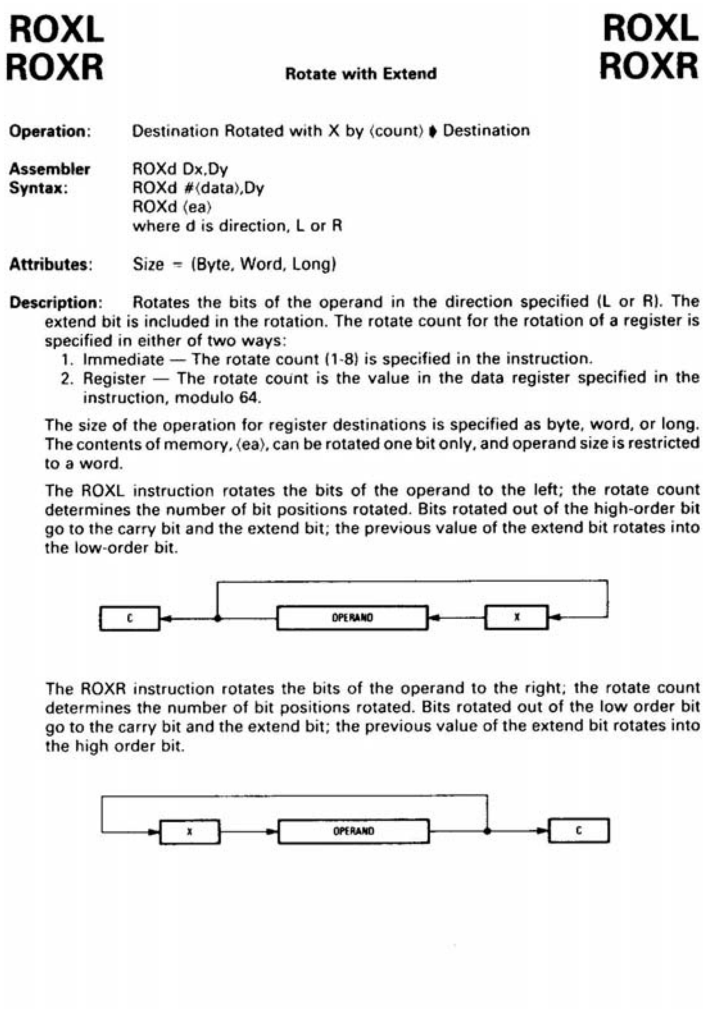
Rotate (Without Extend)

Instruction Fields (Memory Rotate):

dr field — Specifies the direction of the rotate:
0 — Rotate right
1 — Rotate left
Effective Address field — Specifies the operand to be rotated. Only memory alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register
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)	—	—



ROXL ROXR

Rotate with Extend

Operation: Destination Rotated with X by ⟨count⟩ → Destination

Assembler Syntax:
ROXd Dx,Dy
ROXd #⟨data⟩,Dy
ROXd ⟨ea⟩
where d is direction, L or R

Attributes: Size = (Byte, Word, Long)

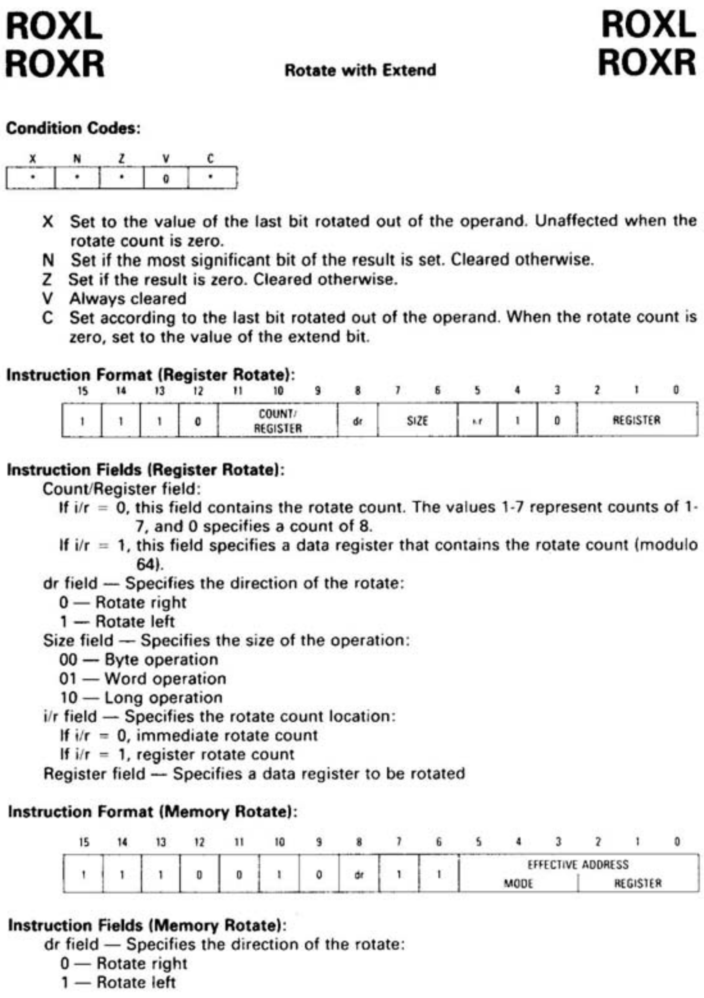
Description: Rotates the bits of the operand in the direction specified (L or R). The extend bit is included in the rotation. The rotate count for the rotation of a register is specified in either of two ways:

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

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 (Register Rotate):

Count/Register field:
If i/r = 0, this field contains the rotate count. The values 1-7 represent counts of 1-7, and 0 specifies a count of 8.
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

Instruction Format (Memory Rotate):

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

Instruction Fields (Memory Rotate):

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

ROXL ROXR

Rotate with Extend

ROXL ROXR

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

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)	—	—

RTD

Return and Deallocate

RTD

Operation: (SP) → PC; SP + 4 + d → 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
DISPLACEMENT (16 BITS)															

Instruction Field:
Displacement field — Specifies the twos complement integer to be sign extended and added to the stack pointer

RTE

Return from Exception
(Privileged Instruction)

RTE

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

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):
0000 — Short Format, removes four words. Loads the status register and the program counter from the stack frame.
1000 — MC68010 Long Format, removes 29 words
Any other value in this field causes the processor to take a format error exception.

RTR

Return and Restore Condition Codes

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 codes 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 — 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 — SBCD

Operation: $Destination_{10} - Source_{10} - X$ ♦ Destination

Assembler Syntax:
SBCD Dx,Dy
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 BCD 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 non-zero. 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.

Instruction Format:

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

Instruction Fields:
Register Ry 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 Rx 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 — 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 conditions:

CC	carry clear	0100	$\overline{C}$	LS	low or same	0011	C + Z
CS	carry set	0101	C	LT	less than	1101	$N \cdot \overline{V} + \overline{N} \cdot V$
EQ	equal	0111	Z	MI	minus	1011	N
F	never true	0001	0	NE	not equal	0110	$\overline{Z}$
GE	greater or equal	1100	$N \cdot V + \overline{N} \cdot \overline{V}$	PL	plus	1010	$\overline{N}$1
GT	greater than	1110	$N \cdot V \cdot \overline{Z} + \overline{N} \cdot \overline{V} \cdot \overline{Z}$	T	always true	0000	$\overline{V}$
HI	high	0010	$\overline{C} \cdot \overline{Z}$	VC	overflow clear	1000	V
LE	less or equal	1111	$Z + N \cdot \overline{V} + \overline{N} \cdot V$	VS	overflow set	1001	

Condition Codes:
Not affected

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										EFFECTIVE ADDRESS					
0	1	0	1	CONDITION				1	1	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 are allowed as shown:

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)	—	—

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, MC68HC000, and MC68008 a memory destination is read before it is written to.

STOP — Load Status Register and Stop (Privileged Instruction) — STOP

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

Assembler Syntax: STOP #⟨data⟩

Attributes: Unsized

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 instructions execution. A trace exception occurs if instruction tracing is enabled 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

SUB — Subtract — 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			OP-MODE			EFFECTIVE ADDRESS					
										MODE			REGISTER		

Instruction Fields:

Register field — Specifies any of the eight data registers
Op-Mode field —

Byte	Word	Long	Operation
000	001	010	(⟨Dn⟩) − (⟨ea⟩) ⧫ ⟨Dn⟩
100	101	110	(⟨ea⟩) − (⟨Dn⟩) ⧫ ⟨ea⟩

SUB — Subtract — SUB

Effective Address field — Determines the addressing mode. If the location specified is a source operand, all addressing modes are allowed as shown:

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

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

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

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)	—	—

Notes:

1. If the destination is a data register, it must be specified as a destination Dn address, not as a destination ⟨ea⟩ address.
2. Most assemblers use SUBA when the destination is an address register, and SUBI or SUBQ when the source is immediate data.

SUBA — Subtract Address — 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 size 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			OP-MODE			EFFECTIVE ADDRESS					
										MODE			REGISTER		

Op-Mode Field:

Word	Long	Operation
011	111	(⟨An⟩) − (⟨ea⟩) ⧫ ⟨An⟩

Instruction Fields:

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

Addressing Mode	Mode	Register
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

SUBI — Subtract Immediate SUBI — 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		
WORD DATA (16 BITS)								BYTE DATA (8 BITS)							
LONG DATA (32 BITS)															

Instruction Fields:

Size field — Specifies the size of the operation:
00 — Byte operation
01 — Word operation
10 — Long operation
Effective Address field — Specifies the destination operand. Only data alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register
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)	—	—

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 — 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 are allowed with address registers, and the condition codes are not affected. When subtracting from address registers, the entire destination address register is used, regardless of 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		

Instruction Fields:

Data field — Three bits of immediate data; 1-7 represent immediate values of 1-7, and 0 represents 8
Size field — Specifies the size of the operation:
00 — Byte operation
01 — Word operation
10 — Long operation
Effective Address field — Specifies the destination location. Only alterable addressing modes are allowed as shown:

Addressing Mode	Mode	Register
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

*Word and Long only.

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#⟨data⟩	—	—
(d_{16},PC)	—	—
(d_8,PC,Xn)	—	—

SWAP — Swap Register Halves — 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 Fields:

Register field — Specifies the data register to swap

SUBX — Subtract with Extend — SUBX

Operation: Destination - Source - X ♦ Destination

Assembler Syntax: SUBX Dx,Dy
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 non-zero. Unchanged otherwise.
V Set if an overflow occurs. Cleared otherwise.
C Set if a carry 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.

Instruction Format:

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

Instruction Fields:

Register Ry 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 Rx 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

TAS — Test and Set an Operand — 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 read-modify-write memory cycle that completes the operation without interruption. 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		

Instruction Fields:

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

Addressing Mode	Mode	Register
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)	—	—

TRAP — Trap — TRAP

Operation: 1 ♦ S bit of SR
SSP − 2 ♦ SSP; Format/Offset ♦ (SSP); — MC68010 only
SSP − 4 ♦ SSP; PC ♦ (SSP); SSP − 2 ♦ SSP;
SR ♦ (SSP); Vector Address ♦ PC

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 — TRAPV

Operation: If V then TRAP

Assembler Syntax: TRAPV

Attributes: Unsized

Description: If the overflow condition is set, causes a TRAPV exception (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 — 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
										EFFECTIVE ADDRESS					
0	1	0	0	1	0	1	0	SIZE		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. If the operation size is word or long, all addressing modes are allowed. If the operation size is byte, only data addressing modes are allowed as shown:

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

UNLK — Unlink — 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 Fields:
Register field — Specifies the address register for the instruction