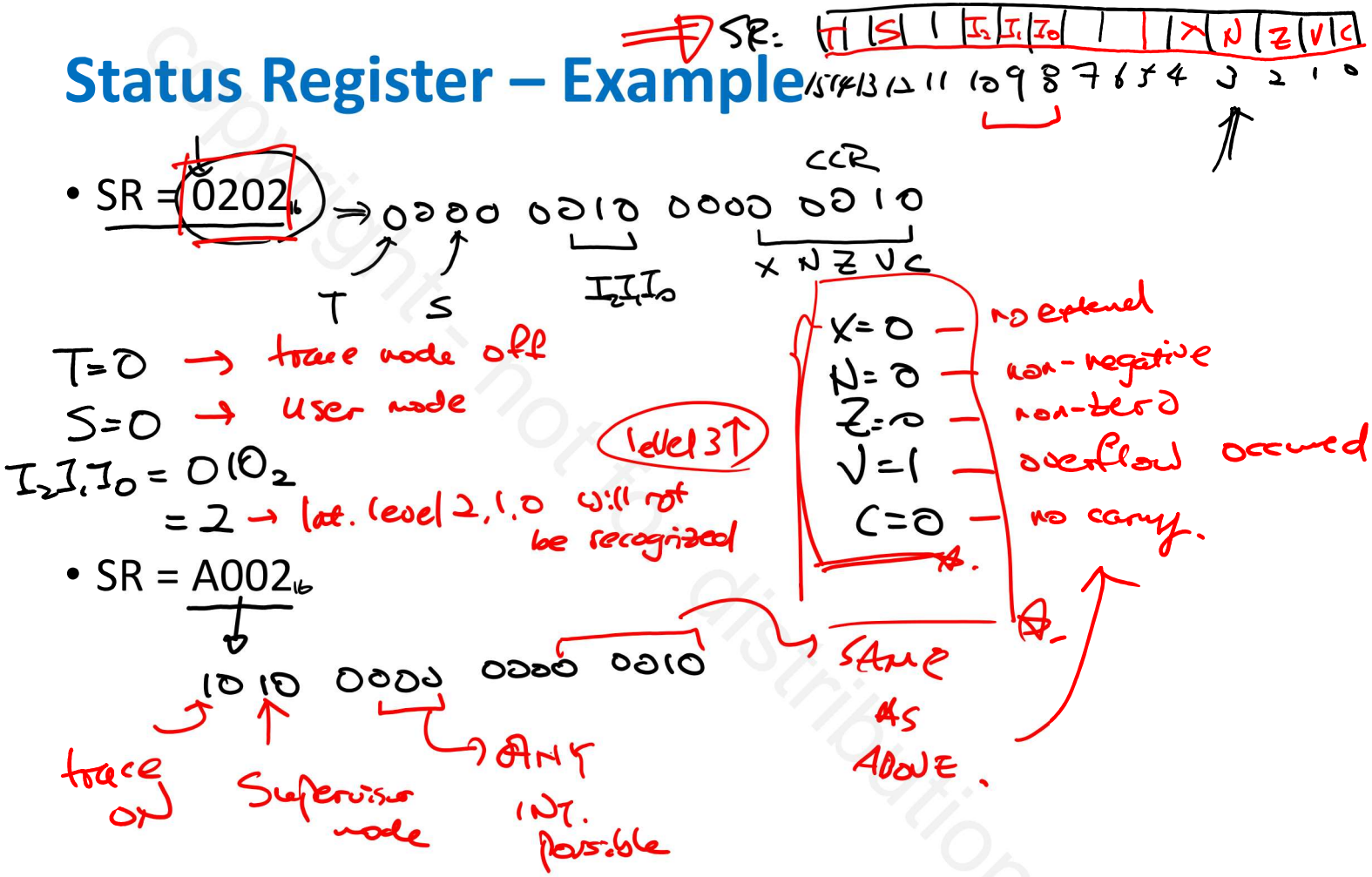
ECE 242
Digital Computers and Computing

Lecture 5. Intro to MC68K – Part 2
Spring 2021

Won-Jae Yi

Status Register – Example



MC68000 Instruction Set

- Information can be found in Appendix IV of the textbook
- Also, can be found under Motorola MC68000 Programmer's Reference Manual (Section 4)
  - https://www.nxp.com/files-static/archives/doc/ref_manual/M68000PRM.pdf
- Programmers refer to this manual when programming in assembly/machine language

MC68000 Instruction Set

- Symbolic Instruction to add two 16-bit operands

ADD.W X,Y

- (Instruction).(Length) (operand),(operand)

Example – CLR (Clear)

CLR — Clear an Operand — CLR

Operation: 0 → Destination

Assembler Syntax: CLR <ea>

Attributes: Size = (Byte, Word, Long)

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

Condition Codes:

X	N	Z	V	C
—	0	1	0	0

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

Instruction Format:

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

Instruction Fields:

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

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

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

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

Instruction	Contents of the Destination After Instruction Executes
CLR.B (EAd)	FFFF FF00
CLR.W (EAd)	FFFF 0000
CLR.L (EAd)	0000 0000

Note: Destination contains FFFF FFFF before each instruction executes.

Instruction Symbolic Form	After Execution
CLR.B D1	(D1)[7:0] = 0
CLR.W D1	(D1)[15:0] = 0
CLR.B 1000	(1000) = 0
CLR.W 1000	(1000) = 0
	(1001) = 0

Example – MOVE (Move)

MOVE.X <EAs>,<EAd>

- X is B, W or L (length/size of the operand)
- <EAs> : Effective Address source
- <EAd> : Effective Address destination

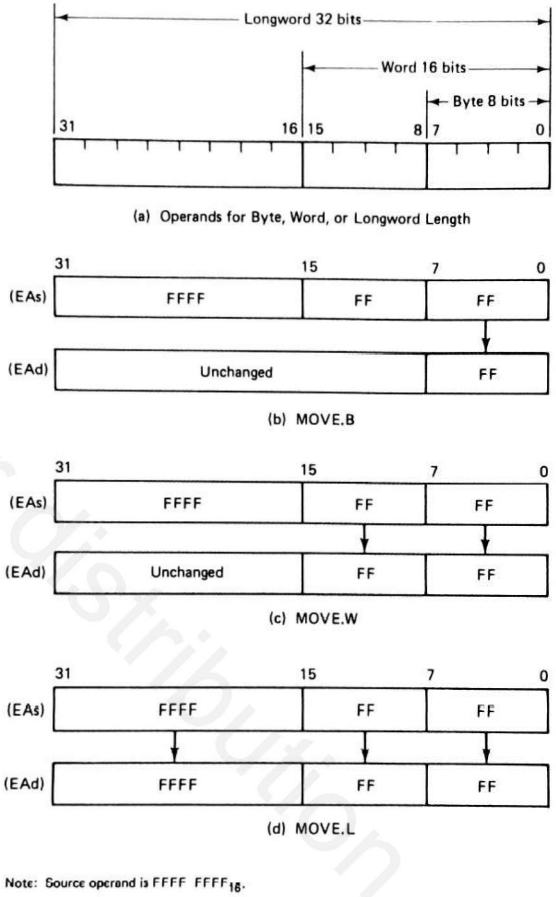
- Can include registers or memory locations

When D2 is 0FFF 0105… Content of D1 for:

- MOVE.B D2, D1

- MOVE.W D2, D1

- MOVE.L D2, D1



Example – MOVE (Move)

MOVE — Move Data from Source to Destination (M68000 Family)

Operation: Source → Destination

Assembler Syntax: MOVE < ea > , < ea >

Attributes: Size = (Byte, Word, Long)

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

X	N	Z	V	C
—	*	*	0	0

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

Instruction Format:

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

Instruction Fields:

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

Integer Instructions

MOVE — Move Data from Source to Destination (M68000 Family)

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

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An

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

MC68020, MC68030, and MC68040 only

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

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

*Can be used with CPU32.

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

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	001	reg. number:An
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011

MC68020, MC68030, and MC68040 only

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

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

*For byte size operation, address register direct is not allowed.
**Can be used with CPU32.

NOTE

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

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

MC68000 Instruction Set

- JMP <EA>

- BRA <disp>

if D2 contains 0FFF 0105, D1 is…

- ADD.B D2, D1
- ADD.W D2, D1
- ADD.L D2, D1

Addressing Modes for MC68K

- Determine the ways in which a processor can reference an operand held in one of its registers or in memory
- Specifies how the processor is to locate or calculate the actual address (effective address) of the operand

- Broad categories of addressing
  - Direct addressing
  - Indirect addressing
  - Relative addressing
  - Immediate addressing

Notation of MC68K Assembly Language

- An absolute address can be expressed in either <decimal address> or $<hexadecimal address>
- Symbol "*" represents current value of PC
- Symbol "#" preceding a number indicates immediate addressing (or a number)
- HEX number in the text itself is preceded by a "$"
  - e.g., $1000 vs 1000
- Immediate value in the text is preceded by a "#"
  - e.g., #$1000 vs $1000 vs #1000 vs 1000
- When writing an MC68K assembly code, must define where your code begins in memory
  - e.g., ORG $8000

Addressing Modes for MC68K

Table 5.7 MC68000 Addressing Modes

Mode	Effective Address Calculation
Register direct addressing	
Data register direct	EA = Dn
Address register direct	EA = An
Absolute data addressing	
Absolute short	EA = (next word)
Absolute long	EA = (next two words)
Indirect addressing	
Address register indirect	EA = (An)
Indirect with postincrement	EA = (An); (An) ← (An) + *N*
Indirect with predecrement	(An) ← (An) − *N*; EA = (An)
Indirect with displacement	EA = (An) + ⟨d_{16}⟩
Indirect with index	EA = (An) + (Xn) + ⟨d_8⟩
Relative addressing	
PC relative with offset	EA = (PC) + ⟨d_{16}⟩
PC relative with index	EA = (PC) + (Xn) + ⟨d_8⟩
Immediate data addressing	
Immediate	DATA = next word(s)
Quick immediate	Inherent data
Implied addressing	
Implied register	EA = SR, USP, SP, or PC

Notes:

1. EA Effective address
   An Address register
   Dn Data register
   Xn Address or data register used as index register
   SR Status register
   PC Program counter
   ⟨d_8⟩ 8-bit offset (displacement)
   ⟨d_{16}⟩ 16-bit offset (displacement)
   ( ) Contents of
   ← Replaces
2. *N* = 1 for byte, 2 for words, and 4 for longwords. If An is the stack pointer and the operand size is byte, *N* = 2 to keep the stack pointer on a word boundary.
3. The designation Ri is also used to indicate a register used for indexing. Motorola literature also uses Xi or ix for an index register as in Appendix IV.

Direct Addressing – Register

- No calculation needed of an effective address
- Operand is in one of the address (Ax) or data (Dx) registers
  - Address register direct and data register direct mode
- Examples:
  - CLR.W D2
  - MOVE.W D1,D2
  - MOVE.L A1,D1
  - MOVEA.L A1,A2

Direct Addressing – Absolute

- Absolute **Short/Word** Addressing Mode
- 16-bit address for an operand in memory
- Used to represent address ranges in the operand of the following:
  - from $00 0000 to $00 7FFF
  - from $FF 8000 to $FF FFFF
- Examples:
  - MOVE.W 10000,D1
  - MOVE.W D1, $9000
  - MOVE.L $2000, D1

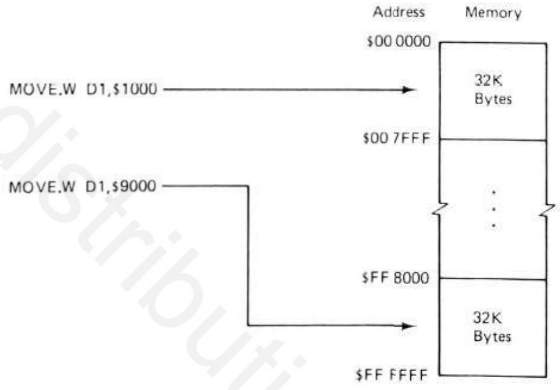


Figure 5.11 Effect of absolute short addressing.

Direct Addressing – Absolute

- Absolute **Long** Addressing Mode
- More than 16-bit address for an operand in memory
- Used to represent address ranges in the operand of the following:
  - from $00 8000 to $FF 7FFF
- Examples:
  - MOVE.W $12000,D1
  - MOVE.B $3FFF,$12000

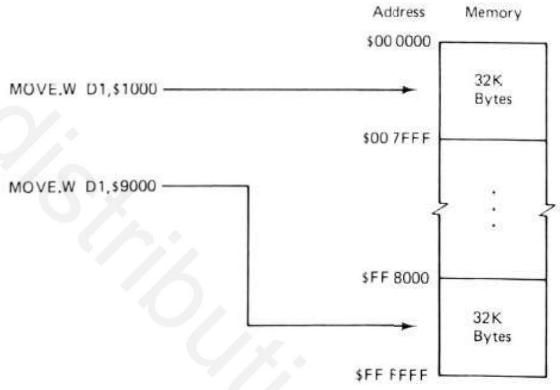


Figure 5.11 Effect of absolute short addressing.

Indirect Addressing

- Use of the contents of an address register as the address of an operand in memory

- MOVE.W (A1), D1

- MOVE.W D1, -(A1)

- MOVE.W (A1)+, D2

Indirect Addressing

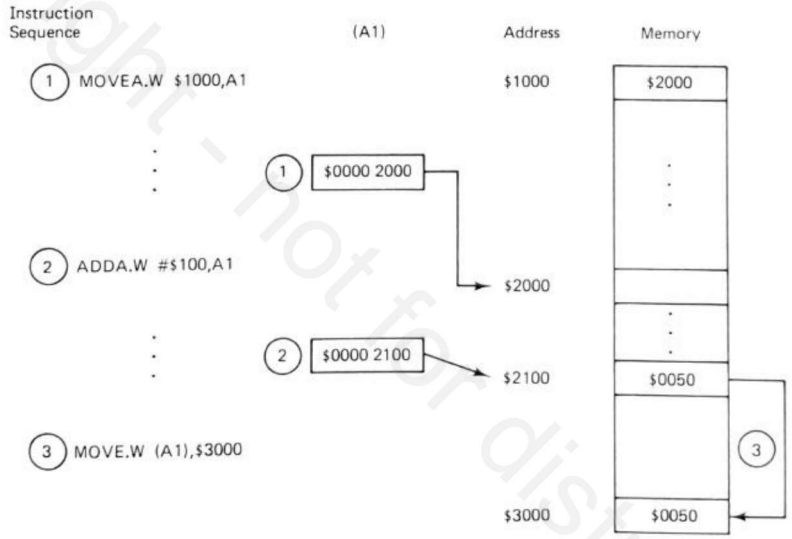


Figure 5.13 Examples of indirect addressing.

Relative Addressing

- An address that the CPU calculates by adding a displacement to the value in the program counter.

EA = (PC) + <displacement>

- BRA *+10

Immediate Addressing

- Used to specify a constant 8, 16 or 32 bits long
- ADD.W #5, D1

- MOVE.L #1234,D1
- MOVE.L #'1234',D1

- MOVE.W #$F0,D1
- MOVE.W $F0,D1

- MOVE.B 1000,#10000
  → Can't be done because destination is an immediate operand

Immediate Addressing

- BSR <address or label>

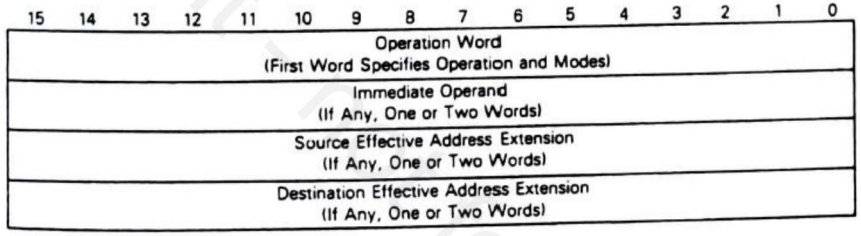
- RTS or RTE (no address needed!)
  - Where is it stored then?

Machine Language for MC68K

- The instructions that the processor executes are coded in machine language format in memory (BINARY!)
  - A programmer can directly code these binary on the memory by modifying the content of the memory
- Assembly program is translated by the assembler from symbolic notation to machine language instructions.
  - Rarely works directly with the machine language program, but knowing its format is necessary to gain full understanding of a processor
- MC68K machine language instructions consist of from 1 to 5 of 16-bit words in memory

Machine Language for MC68K

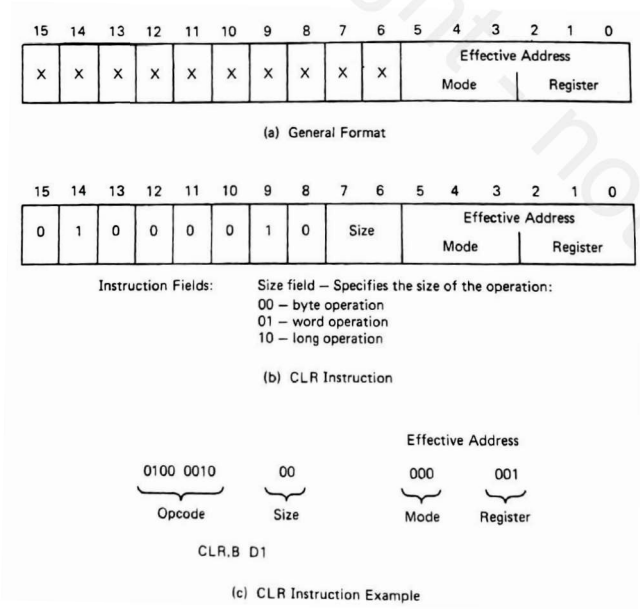
- MC68K machine language instructions consist of from 1 to 5 of 16-bit words in memory



- Other than Operation Word (opword), additional extension words may contain
  - Immediate data
  - Absolute addresses for source/destination operands
    - Short Absolute address = 16 bits → 1 word (Abs.W)
    - Long Absolute address = 32 bits → 2 words (Abs.L)

Single-Address Instructions

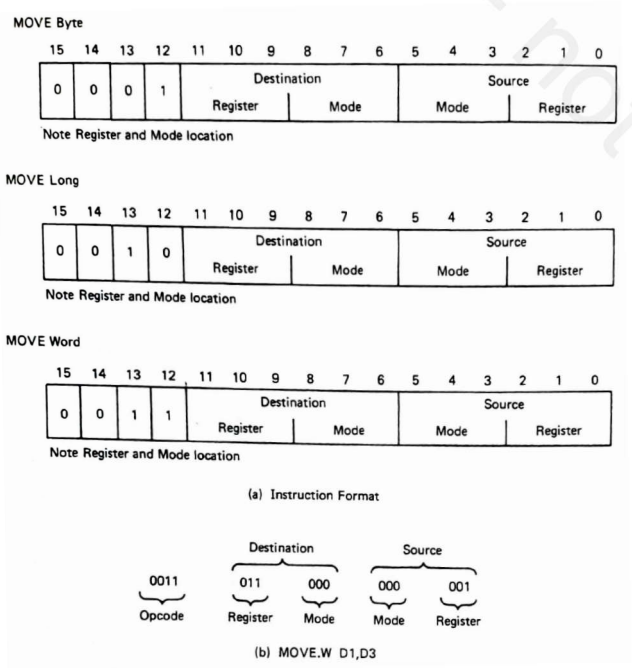
- CLR.B D1



- Refer to Appendix IV of textbook or Page 5 of this slide
- Other instructions such as..
  - NEG (negate)
  - NOT (1's complement)
  - NBCD (negate BCD)
- No need to memorize! Just look at the Instruction Set Details document!

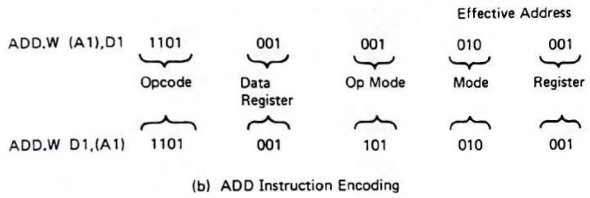
Double-Address Instructions

- An instruction that uses two operands, addressing modes for both (source & destination) must be specified on op-word
- MOVE.W D1, D3
- See Appendix IV MOVE



Double-Address Instructions

- ADD.X <EAs>,Dn or Dn,<EAd>
  - X can be B or W or L



(b) ADD Instruction Encoding

ADD

Add
(M68000 Family)

Operation: Source + Destination → Destination

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

Attributes: Size = (Byte, Word, Long)

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

Condition Codes:

X	N	Z	V	C
*	*	*	*	*

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

Instruction Format:

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

Instruction Fields:

Register field—Specifies any of the eight data registers.

Opmode field

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

ADD

Effective Address field—Determines addressing mode.

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

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	reg. number:Dn	(xxx).W	111	000
An*	001	reg. number:An	(xxx).L	111	001
(An)	010	reg. number:An	#<data>	111	100
(An) +	011	reg. number:An			
– (An)	100	reg. number:An			
(d_{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

MC68020, MC68030, and MC68040 only

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

*Word and long only
**Can be used with CPU32.

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

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	—	—	(xxx).W	111	000
An	—	—	(xxx).L	111	001
(An)	010	reg. number:An	#<data>	—	—
(An) +	011	reg. number:An			
– (An)	100	reg. number:An			
(d_{16},An)	101	reg. number:An	(d_{16},PC)	—	—
(d_8,An,Xn)	110	reg. number:An	(d_8,PC,Xn)	—	—

MC68020, MC68030, and MC68040 only

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

*Can be used with CPU32

NOTE

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

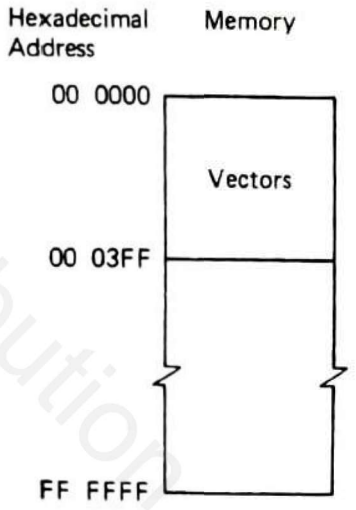
ADDA is used when the destination is an address register. ADDI and ADDQ are used when the source is immediate data. Most assemblers automatically make this distinction.

Assembly Language to Machine Language Examples

- MOVE.B $3FFF,$12000
- MOVE.W $12000,D1
- ADD.L #$1000,D1
- ADD.B $1000,D1

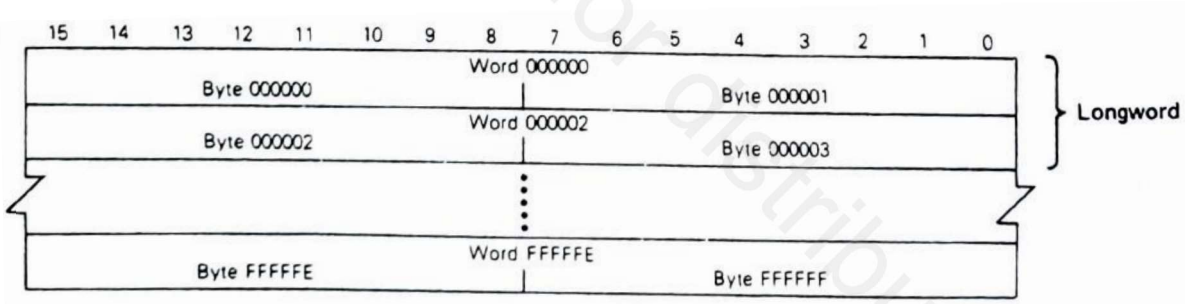
Memory Organization of MC68K

- Data signal lines = 16 bits
- Address signal lines = 24 bits
  - Each address indicates **a byte location in memory or address** of a **location associated with an interface**
- Can be accessed in byte, word or longword
- Addressing space after $3FF_{16}$ can be utilized for system designers
- "Vectors" include addresses that point to routines for servicing interrupts or processing traps
  - *"Vector Tables"*



Memory Organization of MC68K

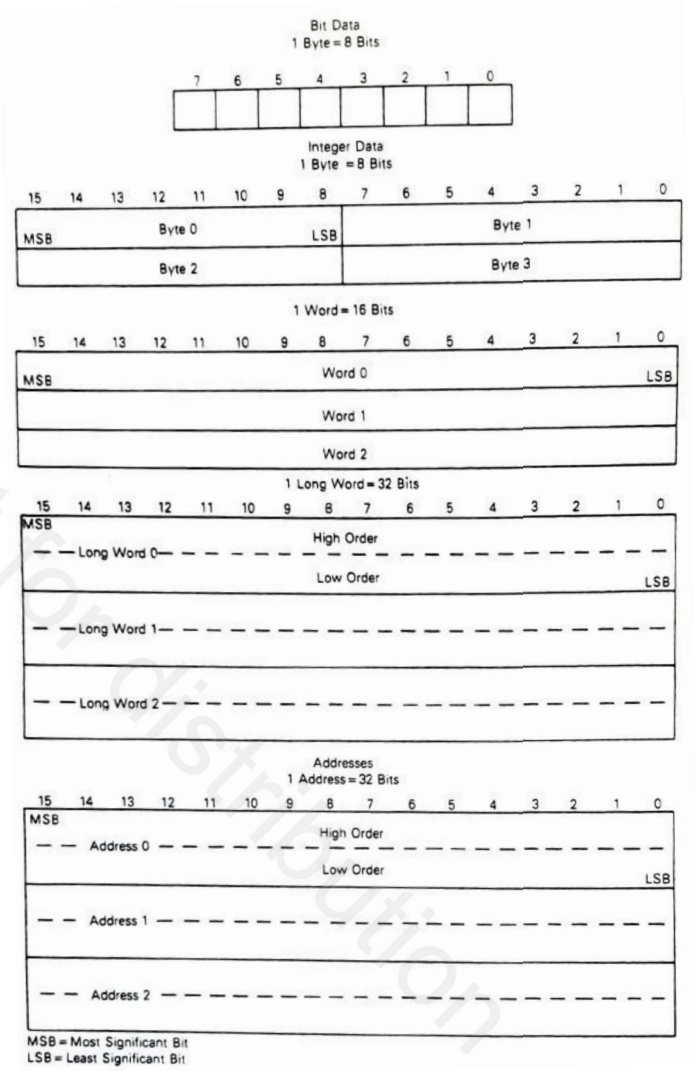
- Although each byte in the word ca be addressed by the processor, **instructions and word operands** must be referenced only at **even addresses**
  - n, n+2, n+4, …. where n is an even address
  - applies to .W and .L operations
  - .B operations can manage odd address



- Example: MOVE.W $1001, D1
  - Causes error, “addressing error”

Data Organization

- Within a byte, bit 0 is the rightmost, bit 7 is the leftmost bit
- Bit 0 represents LSB for a byte or a word value (integer data)
- High-order 16bits in location *n*, low-order 16bits in location *n+2* for longword value (*n* is even integer)



Data Organization of MC68K

- MC68K is a Big-Endian system
  - Most-significant byte at least address of a word
  - c.f. Little-Endian: least-significant byte at least address

- Storing 0002 00F6$_{16}$ at memory location $1000

Memory Address	Big Endian (Motorola)	Little Endian (Intel)
$1000	00	F6
$1001	02	00
$1002	00	02
$1003	F6	00