Chapter 6: 68K Instruction Decomposition

Topic

- 68000 Instruction Set Architecture
  - Instruction Set Decomposition

  - 68K manual
  - Chapter 8 (Berger)
  - Chapter 2, 3 (Clements)

Instruction Decomposition

- **Assemble** (decompose): Translate assembly code to machine code
  - Why need assembling?
    - The memory system can store binary numbers only
    - Basically, all assembly languages should be translated into a set of binary numbers to be stored or understood by the computer system

- **Disassemble**: Translate machine code to assembly code
  - Why need disassembling?
    - It helps you understand assembly programming
    - You may need it in your future work
      - Software engineer in software security related area

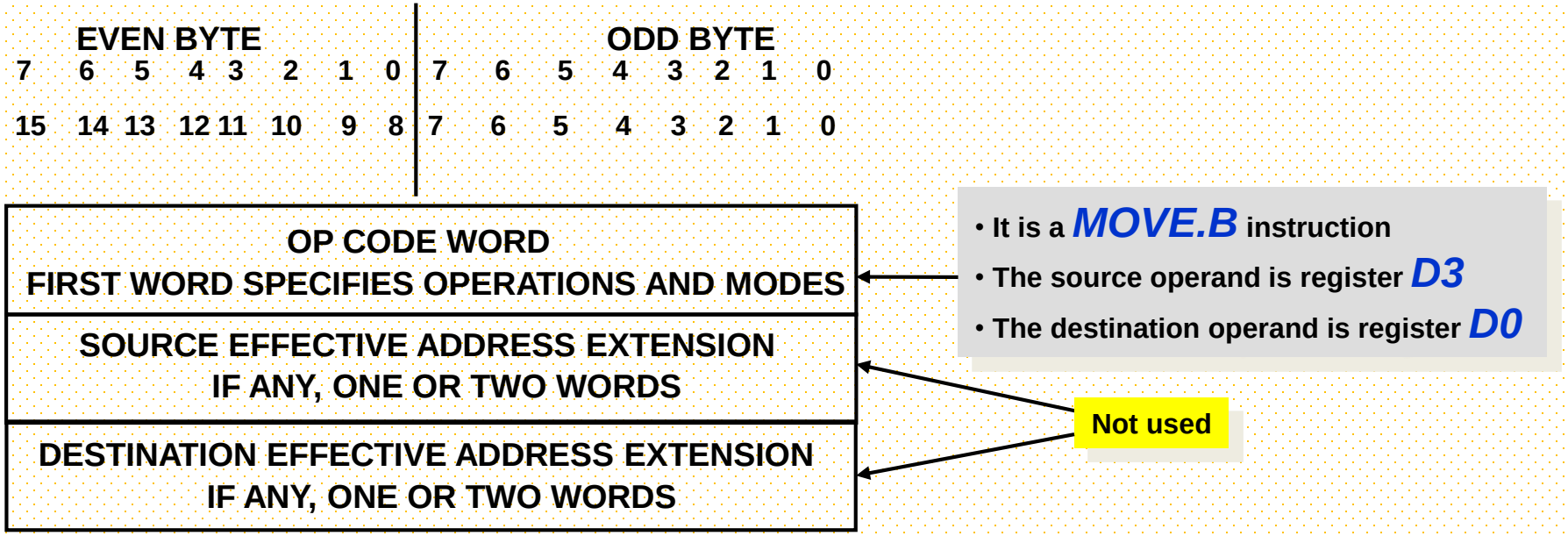
68000 Instruction Format

- Instruction set in a memory
  - Assembly codes are assembled into binary numbers and stored in memory
  - One instruction code can be **up to 5 words – 80 bits**
    - 10 = 2 + 4 + 4

  For example,
  - MOVE.B #14, D0 → 103C 000E (machine code in hex)
  - MOVE.W $0010AA00,$00103000 → 33F9 0010AA00 00103000
- The **first 16-bit** (e.g., $103C, $33F9) of an instruction is called the ***Opcode Word***
  - contains ***all of the information needed to decode the rest of the instruction***
  - contains the ***Opcode*** and ***Effective Address*** *(EA) fields*
- The complete instruction in memory **must contain the opcode word** and ***may contain additional words to complete the instruction***

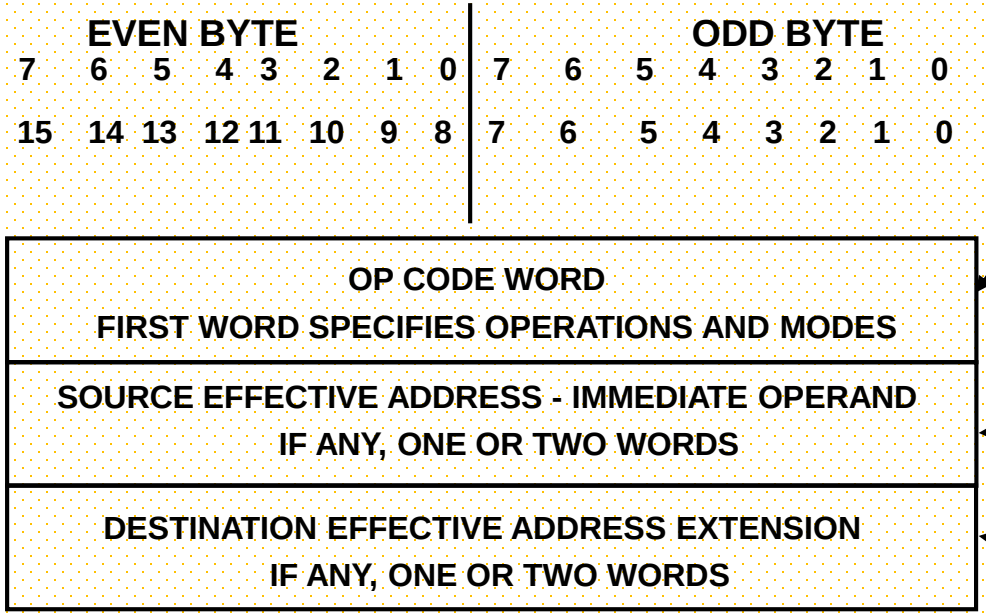
Instruction Format in Memory

- Example: Only op-code word
  - Generally represented as **OPCODE**
  - Example: **MOVE.B D3, D0 → 1003**



Instruction Format in Memory (2)

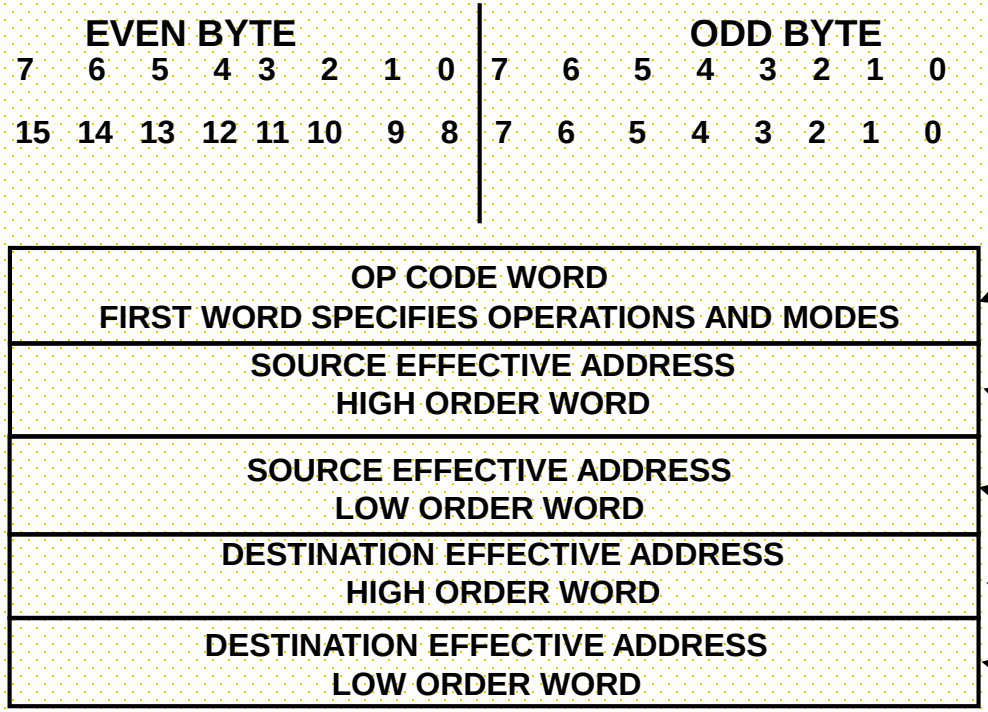
- Example: an ***immediate operand*** is the actual data value
  - Generally represented as **OPCODE *#DATA* (min. unit is word)**
  - Example: **MOVE.B #14, D0 → 103C *000E***



Instruction Format in Memory (3)

- Example: an ***absolute operand*** is the actual data value
  - Generally represented as **OPCODE *source EA, dest EA***

MOVE.W $0010AA00,$00103000 → 33F9 *0010AA00 00103000*



- It is a ***MOVE.W*** instruction
- The source operand is ***absolute address***
- The destination operand is ***absolute address***

- $0010
- $AA00
- $0010
- $3000

Effective Address

- The effective address, EA, determines how the operands of an instruction are **to be accessed by the processor**
- Different types of EA's determine the processor's *addressing modes*
- In 68K manual, each instruction has different codes for each EA mode:
  - **Dn**: data register direct: D0, D1, …, D7
  - **An**: address register direct : A0, A1, …, A6
  - **(An)**: address register indirect: (A0), (A1), …, (A6)
  - **(An)+**: address register indirect with post-increment
  - **-(An)**: address register indirect with pre-decrement
  - **(d_{16}, An)**: address register indirect with displacement
    (EA = (An)+ d_{16})
  - **(d_8, An, Xn)**: address register indirect with index
    (EA = (An)+(Xn)+ d_8)
  - **(xxx).W**: Absolute addressing (word)
  - **(xxx).L**: Absolute addressing (long-word)
  - **#<data>**: Immediate Addressing
  - **(d_{16}, PC)**: Program counter with displacement (EA = (PC)+ d_{16})
  - **(d_8, PC, Xn)**: Program counter with index (EA = (PC)+(Xn)+ d_8)

Assemble manually?

The processor manual tells you what you need to know!

Format of the 68000 Instructions Set (1)

- **Two Operands Operation**

  E.g., MOVE, MOVEA

- **MOVE** instruction: Move the contents at the memory location specified by the **src EA** to the memory location specified by the **dst EA**
- It's ***opcode word*** is shown below:

  - This information is encoded in the 16 bits of the opcode word
    - OPCODE/SIZE = Bit15-12
    - Destination Effective Address = Bit11-6
    - Source Effective Address = Bit5-0
- May have to retrieve **additional words** from memory to complete the instruction
  - Note: *Not all instructions have the same form as the MOVE instruction*
  - *Refer the 68K manual*

68K Manual

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

Operation: Source → Destination

Assembler Syntax: MOVE < ea > , < ea >

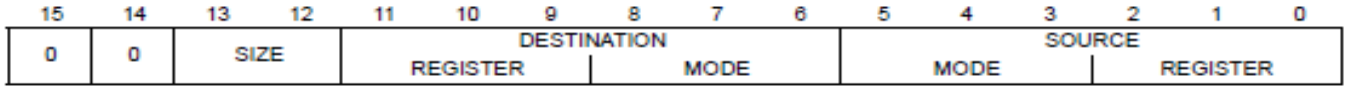
Attributes: Size = (Byte, Word, Long)

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:



Instruction Fields:

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

68K Manual

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.

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.

68K Manual

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.

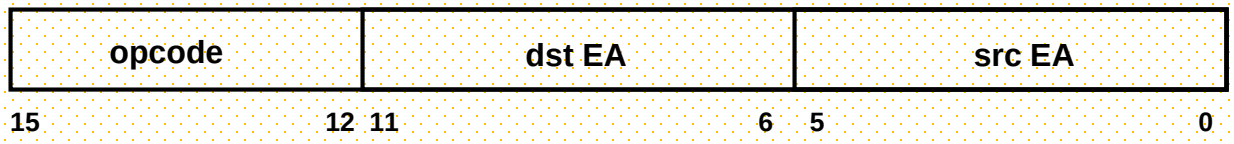
NOTE

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

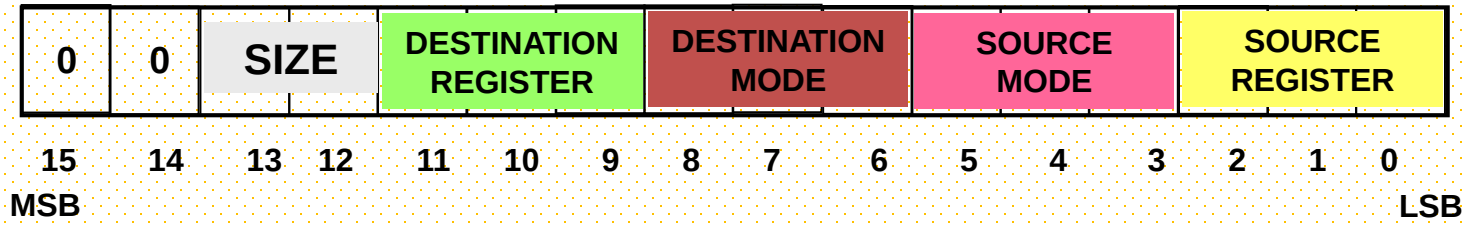
MOVEQ can be used to move an immediate 8-bit value to a data register.

Decomposing the **MOVE** Instruction

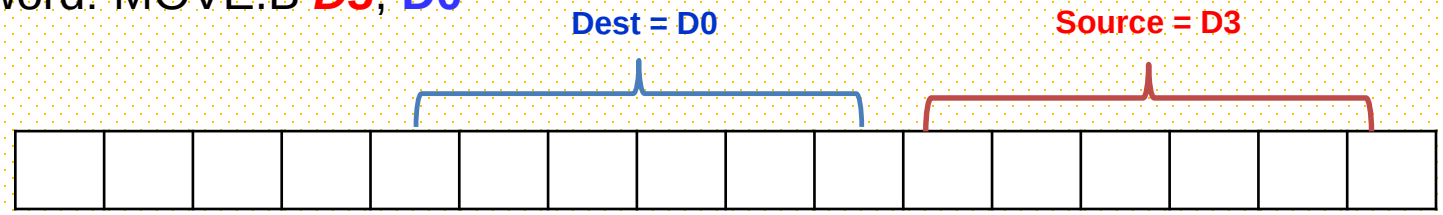
- Refer the 68K manual for instruction format
- Recall that the MOVE instruction format was shown as:



- According to the 68K manual, we can decompose this further to:



- Opcode word: MOVE.B ***D3***, **D0**



68K Manual

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.

68K Manual

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.

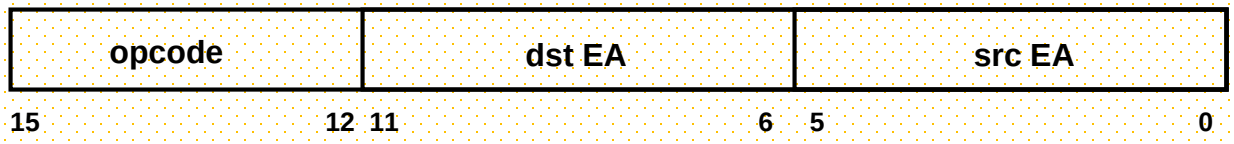
NOTE

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

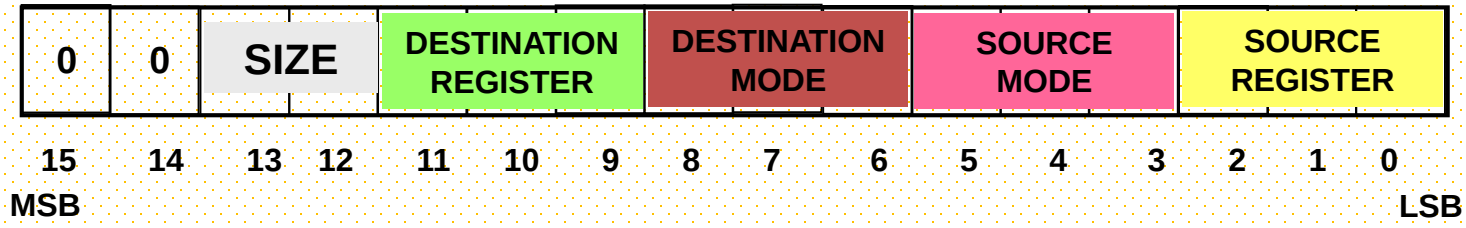
MOVEQ can be used to move an immediate 8-bit value to a data register.

Decomposing the **MOVE** Instruction

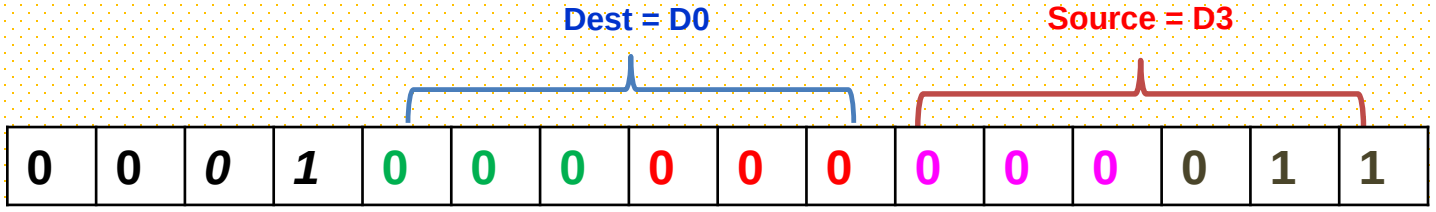
- Refer the 68K manual for instruction format
- Recall that the MOVE instruction format was shown as:



- According to the 68K manual, we can decompose this further to:



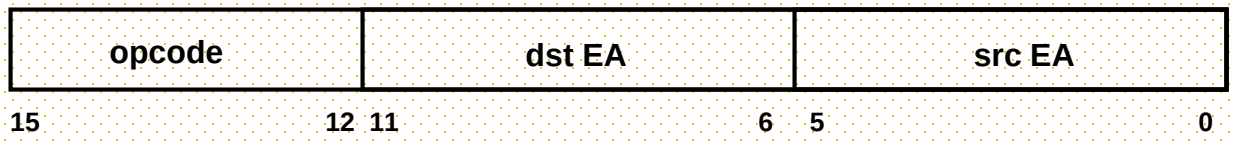
- Opcode word: MOVE.B ***D3***, D0



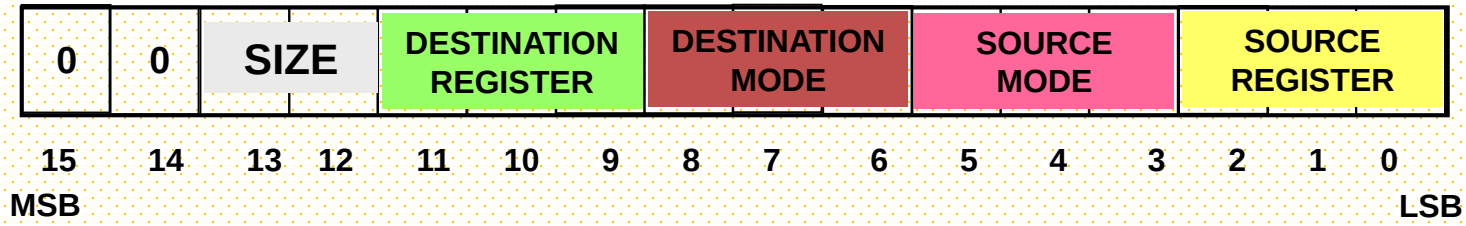
- **1003**

Decomposing the **MOVE** Instruction (2)

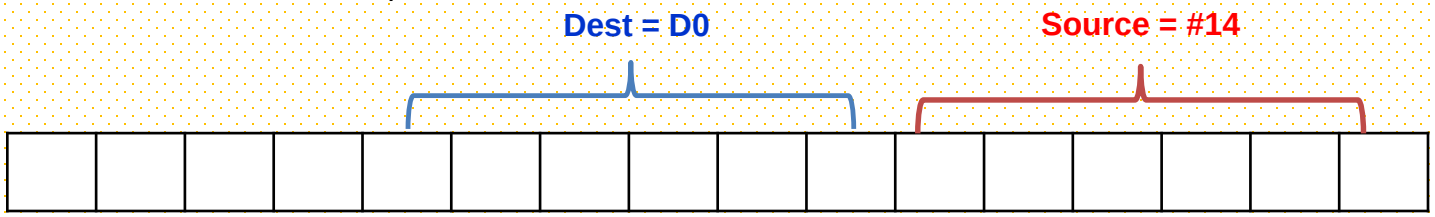
- Refer the 68K manual for instruction format
- Recall that the MOVE instruction format was shown as:



- According to the 68K manual, we can decompose this further to:



- Opcode word: MOVE.B ***#14***, **D0**



68K Manual

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.

68K Manual

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.

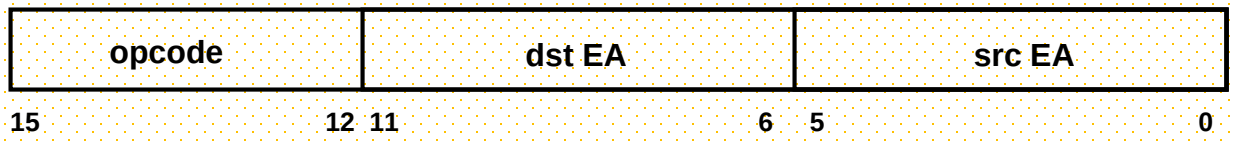
NOTE

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

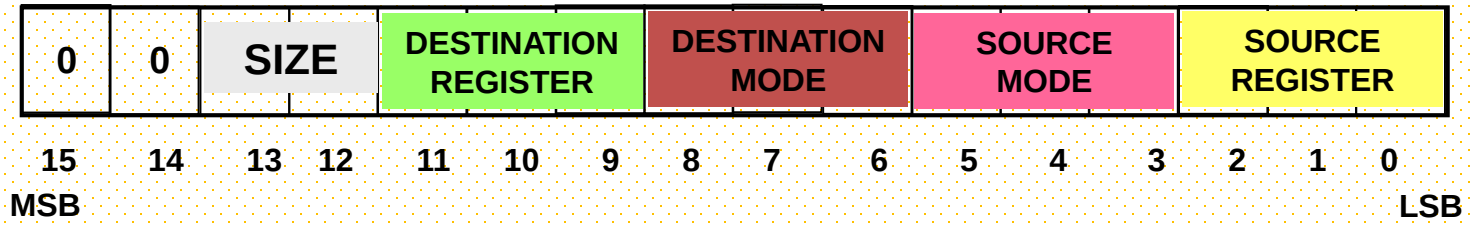
MOVEQ can be used to move an immediate 8-bit value to a data register.

Decomposing the **MOVE** Instruction (2)

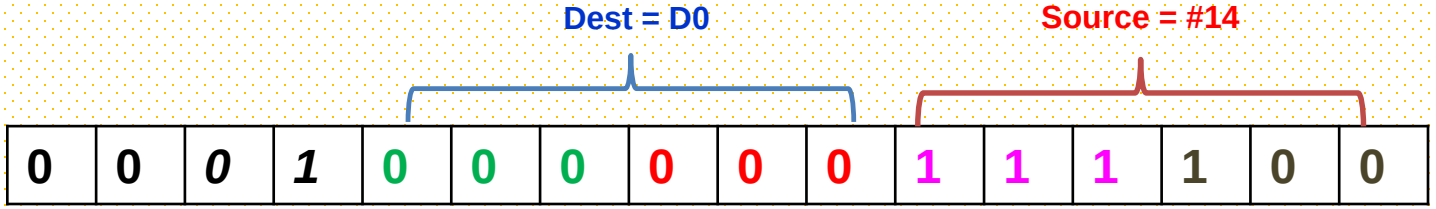
- Refer the 68K manual for instruction format
- Recall that the MOVE instruction format was shown as:



- According to the 68K manual, we can decompose this further to:



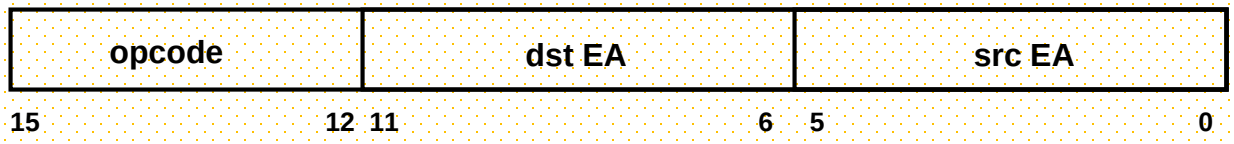
- Opcode word: MOVE.B ***#14***, **D0**



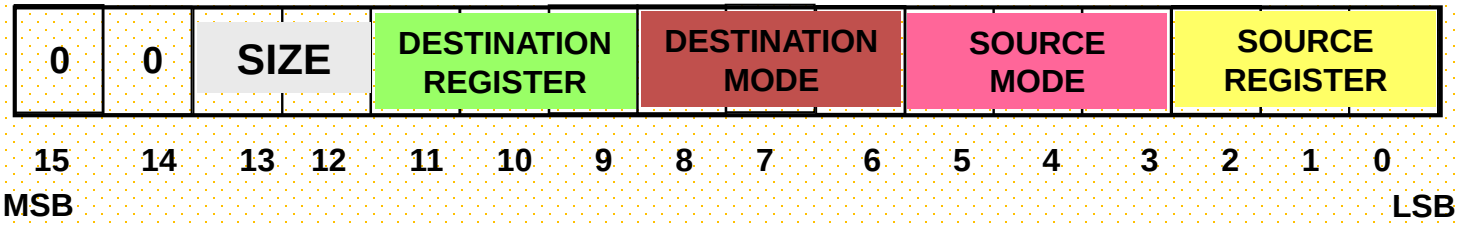
- **Finished?**

Decomposing the **MOVE** Instruction (2)

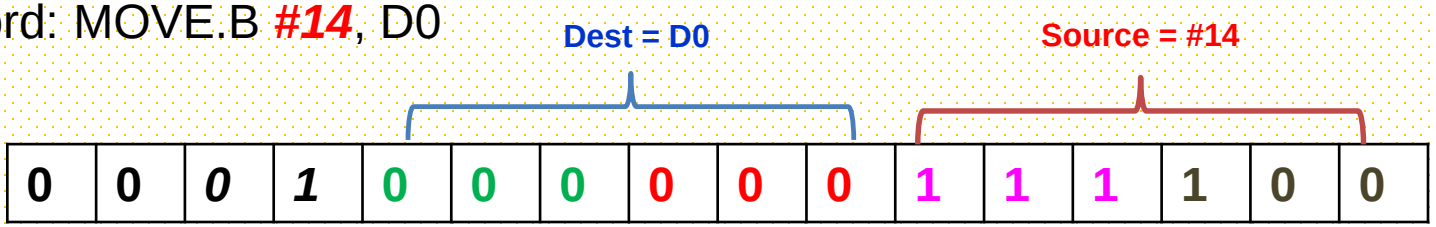
- Refer the 68K manual for instruction format
- Recall that the MOVE instruction format was shown as:



- According to the 68K manual, we can decompose this further to:



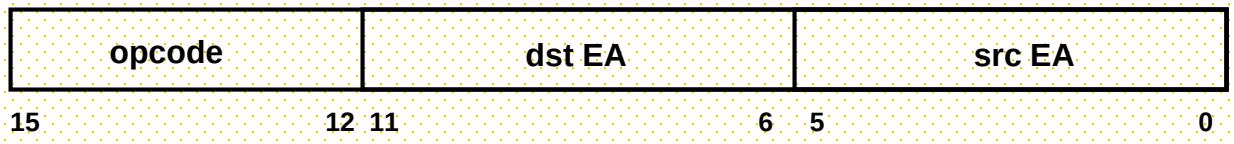
- Opcode word: MOVE.B ***#14***, D0



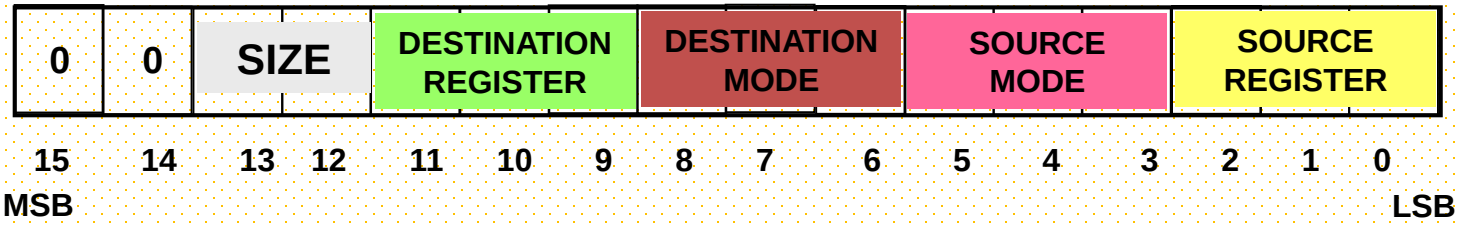
- **Finished?** **No, you need one more word for the number #14**

Decomposing the **MOVE** Instruction (2)

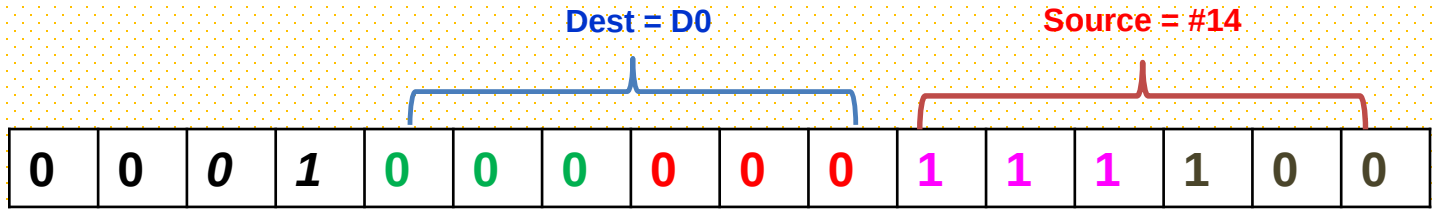
- Refer the 68K manual for instruction format
- Recall that the MOVE instruction format was shown as:



- According to the 68K manual, we can decompose this further to:



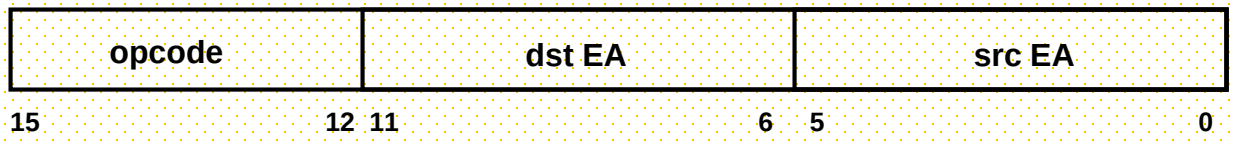
- Opcode word: MOVE.B ***#14***, D0



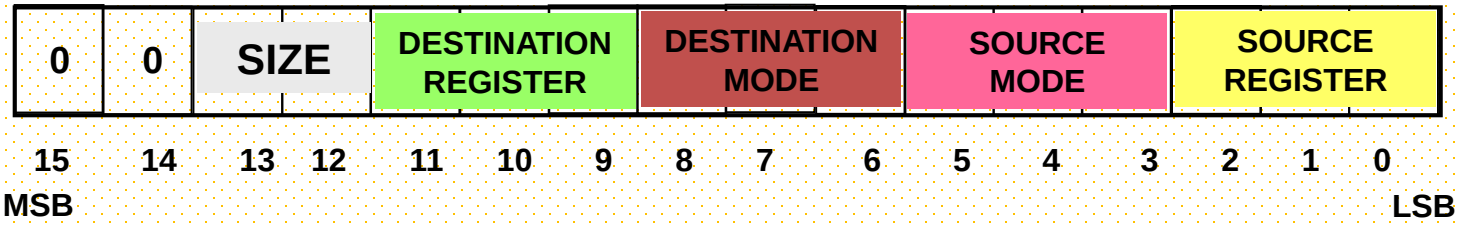
- **103C 000E**

Decomposing the **MOVE** Instruction (3)

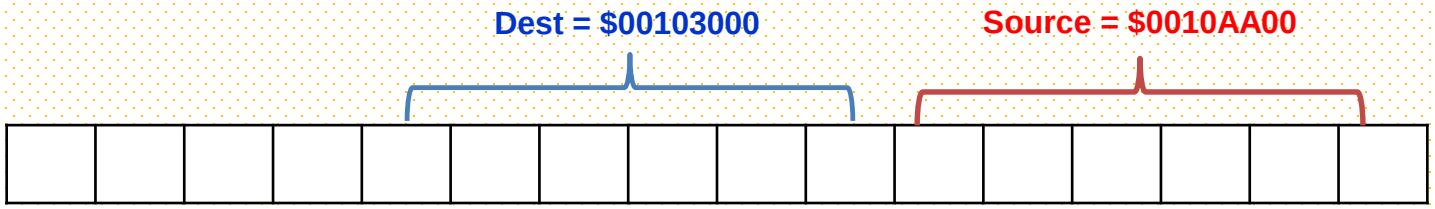
- Refer the 68K manual for instruction format
- Recall that the MOVE instruction format was shown as:



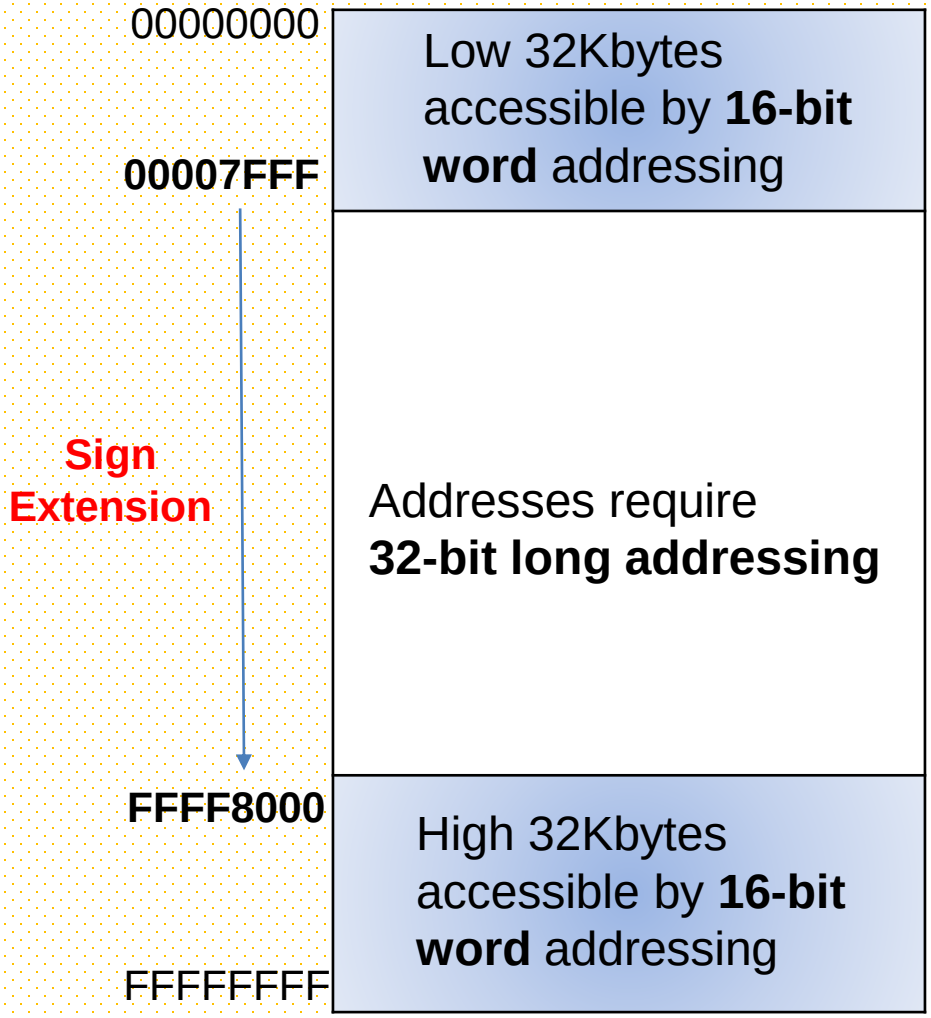
- According to the 68K manual, we can decompose this further to:



- Opcode word: MOVE.W $0010AA00, $00103000



Absolute Addressing Range



68K Manual

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.

68K Manual

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.

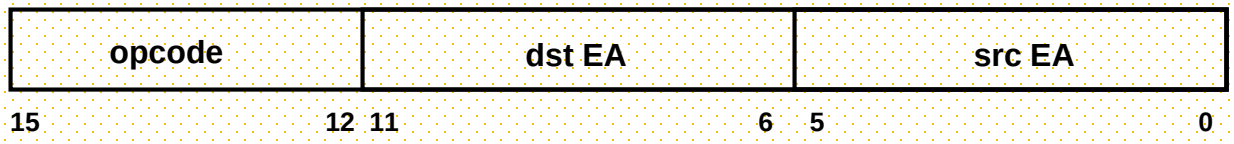
NOTE

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

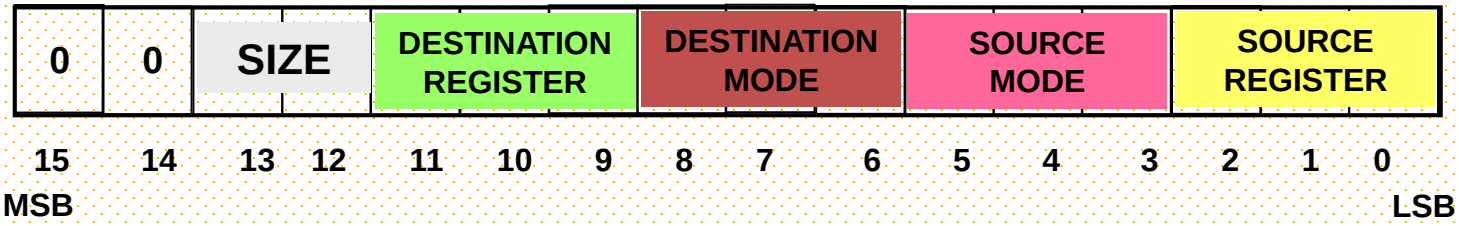
MOVEQ can be used to move an immediate 8-bit value to a data register.

Decomposing the **MOVE** Instruction (3)

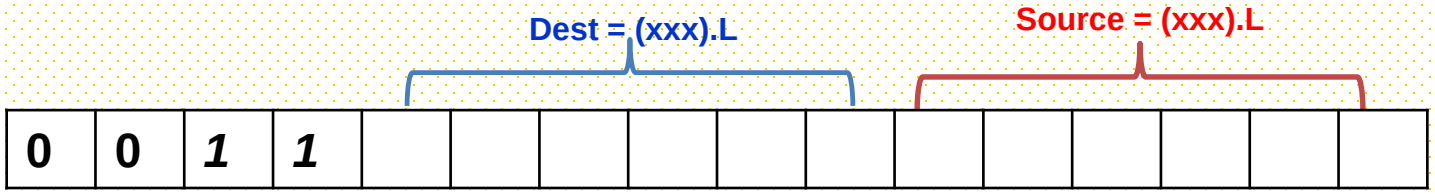
- Refer the 68K manual for instruction format
- Recall that the MOVE instruction format was shown as:



- According to the 68K manual, we can decompose this further to:

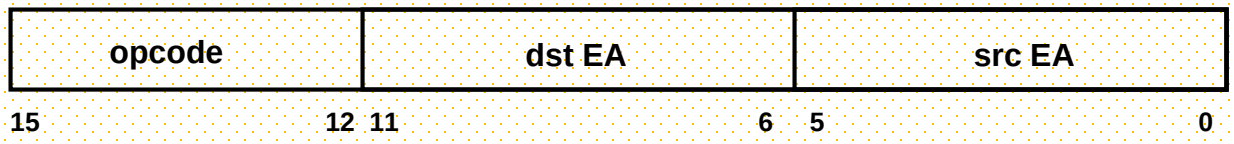


- Opcode word: MOVE.W $0010AA00, $00103000

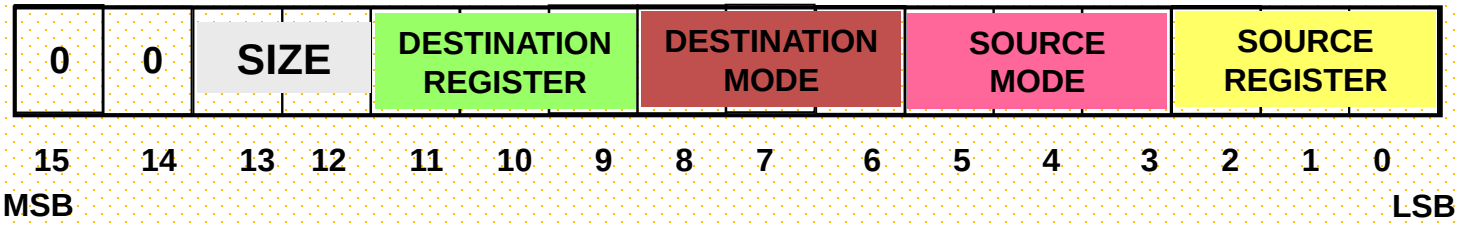


Decomposing the **MOVE** Instruction (3)

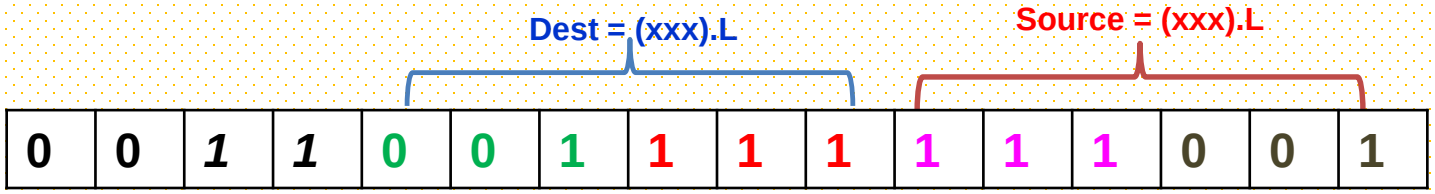
- Refer the 68K manual for instruction format
- Recall that the MOVE instruction format was shown as:



- According to the 68K manual, we can decompose this further to:

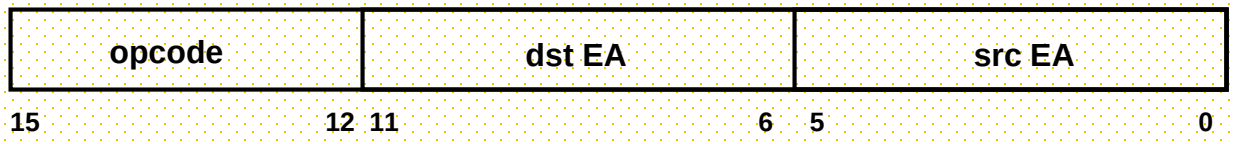


- Opcode word: MOVE.W $0010AA00, $00103000

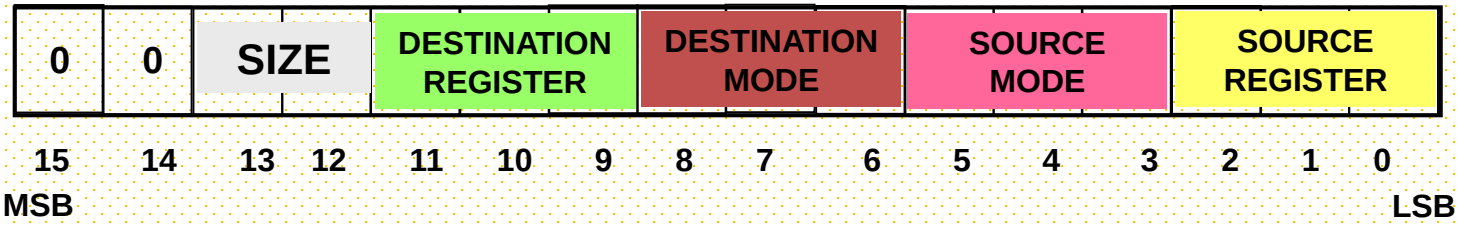


Decomposing the **MOVE** Instruction (3)

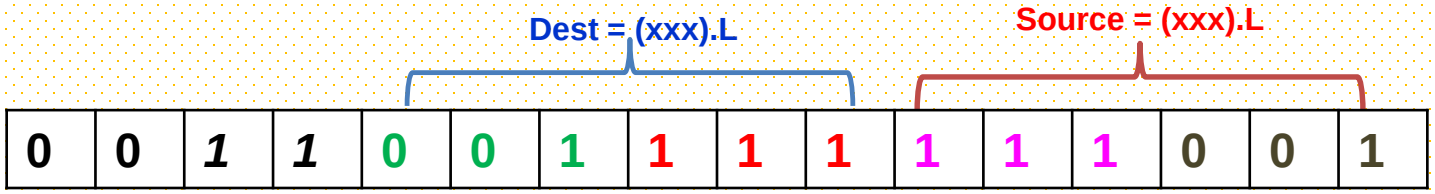
- Refer the 68K manual for instruction format
- Recall that the MOVE instruction format was shown as:



- According to the 68K manual, we can decompose this further to:



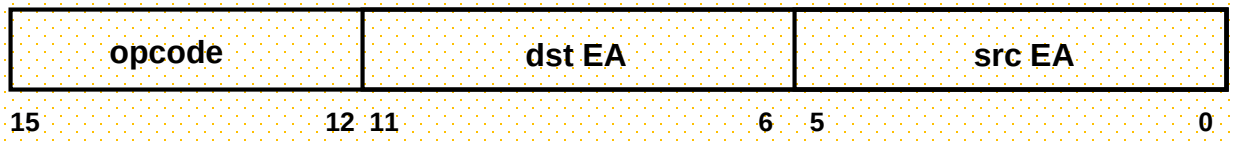
- Opcode word: MOVE.W $0010AA00, $00103000



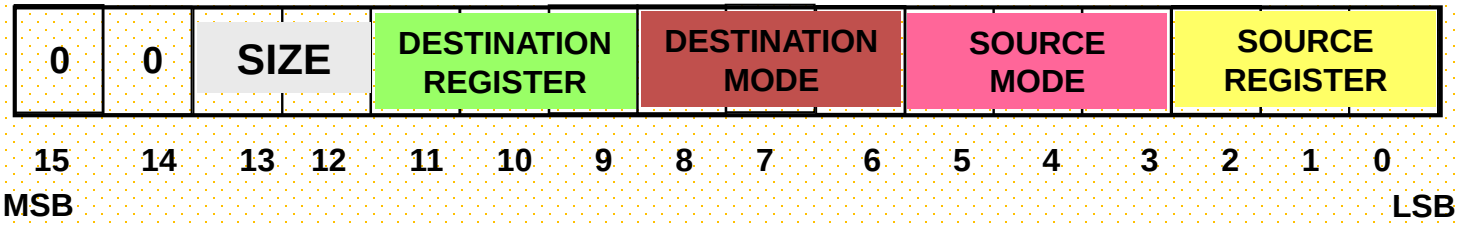
- **Finished?**

Decomposing the **MOVE** Instruction (3)

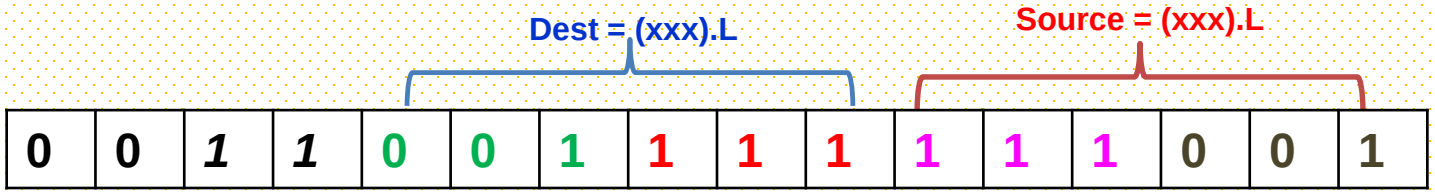
- Refer the 68K manual for instruction format
- Recall that the MOVE instruction format was shown as:



- According to the 68K manual, we can decompose this further to:



- Opcode word: MOVE.W $0010AA00, $00103000



- **33F9 0010AA00 00103000**

MOVEA

Move Address (M68000 Family)

MOVEA

Operation: Source → Destination

Assembler Syntax: MOVEA < ea > ,An

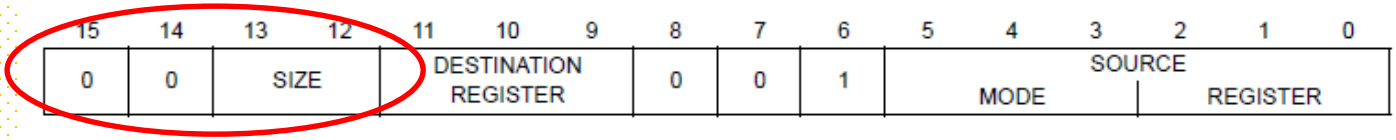
Attributes: Size = (Word, Long)

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

Condition Codes:

Not affected.

Instruction Format:



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.

- The **MOVEA** instruction is a special form of the MOVE instruction and is used **if the destination (dst) is an Address Register**
- **MOVEA.W and MOVE.W have the same opcodes**

MOVEA Notes

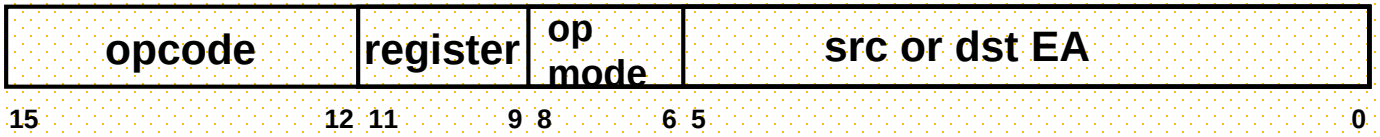
- Why providing both MOVEA and MOVE?
  - In a 68K processor, **word accesses must be aligned on word boundaries**
  - If the same instruction was used to store data in an address register, it would be possible to store an odd address value and ***cause a non-aligned access to occur***
  - ***A runtime error will happen!***

- MOVE.W #1110, D1 -> Valid in 68K!

- MOVE.W #1110, A1 -> **Invalid** in 68K!, use MOVEA instead!
  - The destination register of MOVE operation cannot be Address Register
  - **The Easy68K simulator has a bug that can accept this operation!**

Format of the 68000 Instructions Set (2)

- Use an **internal register** as **the source or destination** of the operation (i.e., ADD, AND, CMP)



ADD **Add** **ADD**
(M68000 Family)

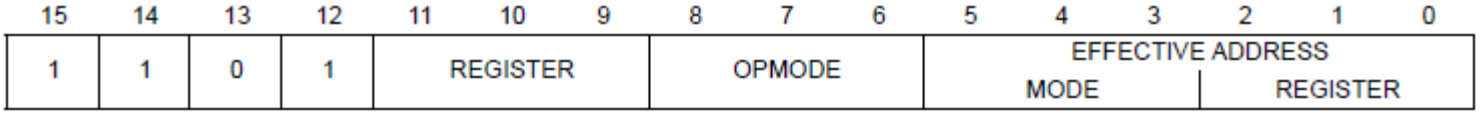
Operation: Source + Destination → Destination

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.

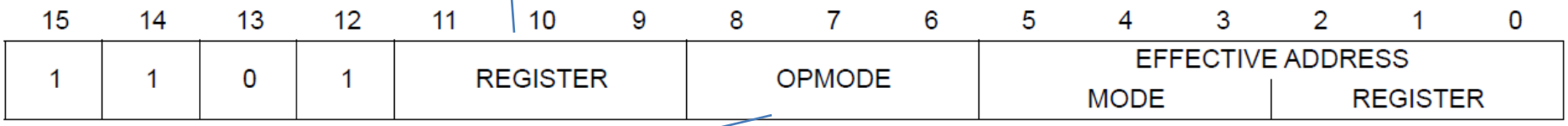
Instruction Format:



Operation: Source + Destination → Destination

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

Specifies any of the eight data registers



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

Effective Address field—

If the location specified is a **source operand, all addressing modes can be used**

If the location specified is a **destination operand, only memory alterable addressing modes can be used**

Exercise

1. Assemble by hand the following assembly language MOVE and ADD instructions

- MOVE.B (A0),D7
- MOVE.L $1234,D7
- MOVEA.W D7,A0
- ADD.W D0,D7

Exercise - Illegal Instructions

2. Explain why the following codes are illegal

- MOVE.W $2233,A5

Answer: A5 is an Address register

- MOVE.B #$2233,D6

Answer: #$2233 is a word (2 bytes)

- ADD.W D0,#$1000

Answer: #$1000 is an immediate number, which cannot be a destination in ADD operation

- MOVEA.B D7,A0

Answer: MOVEA can only operate on Word or Long, but not Byte

- MOVEA.W A0,$1234

Answer: Destination must be an Address register for MOVEA operation

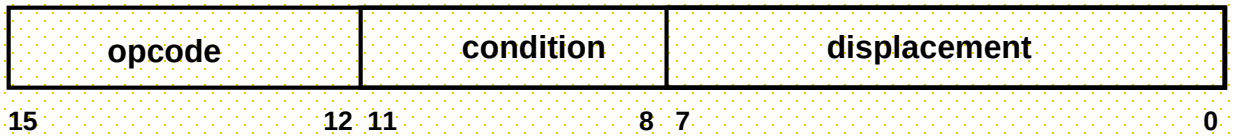
Format of the 68000 Instructions Set (3)

- **Single operand instructions**, i.e., CLR (clear the contents of the dst EA)
  - An opcode word + rest of instruction (sometimes do not have)

- JMP (Jump) and JSR (Jump to Sub-Routine) are single-operand instructions
  - JMP: Change the value of the Program Counter (PC)
  - Next instruction is fetched from <PC>
- JSR is a special type of jump instruction
  - Replaces <PC> with operand but also saves the current <PC> on the stack
  - Can return to starting point with RTS
  - Used for interrupt subroutines or ISR's and function calls

Format of the 68000 Instructions Set (4)

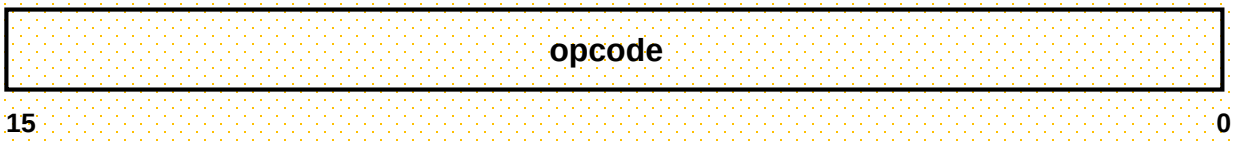
- **Branch instructions**
  - Change the program counter value to a new value if a test condition is true
  - Test conditions are represented by the state of the *flags* in the *Condition Code Register (CCR)*
    - Tests can be, zero, overflow, carry or borrow, negative



- Destination of the branch is calculated by adding the current value of the program counter to the displacement value in the instruction
  - Uses 2's complement, signed addition

Format of the 68000 Instructions Set (5)

- **Inherent addressing**: The effect (dst or src) of the opcode is inherently contained in the function of the opcode
  - RTS: ReTurn from Subroutine:
    - JSR instruction PUSH the return location on the stack
    - RTS only needs to POP the <PC> in order to get back from the subroutine



Some Representative Instructions

- **CLR.B ddst** **0100001000dddddd**
  - Clear (set to zero) the byte of the data destination operand – ddst
- **BNE displacement** **01100110dddddddd**
  - Branch if the result is Not Equal to zero (Zero flag = 0)
- **BEQ displacement** **01100111dddddddd**
  - Branch if the result is EQual to zero (Zero flag = 1)
- **JMP cdst** **0100111011dddddd**
  - Jump to the address defined by control destination operand, cdst
- **RTS** **0x4E75**
  - Notice that the RTS instruction does not require an operand