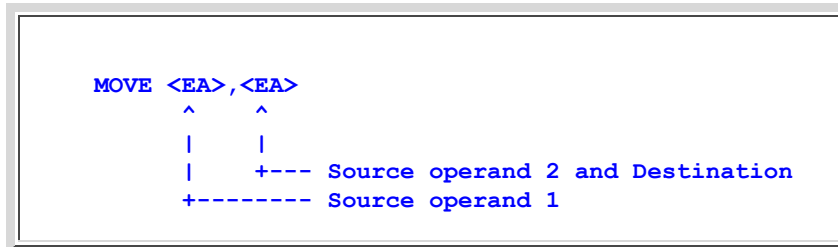
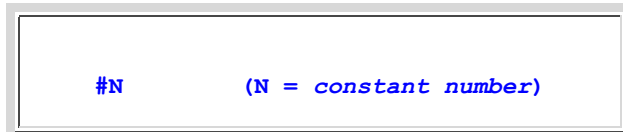


M68000 Immediate Addressing Mode

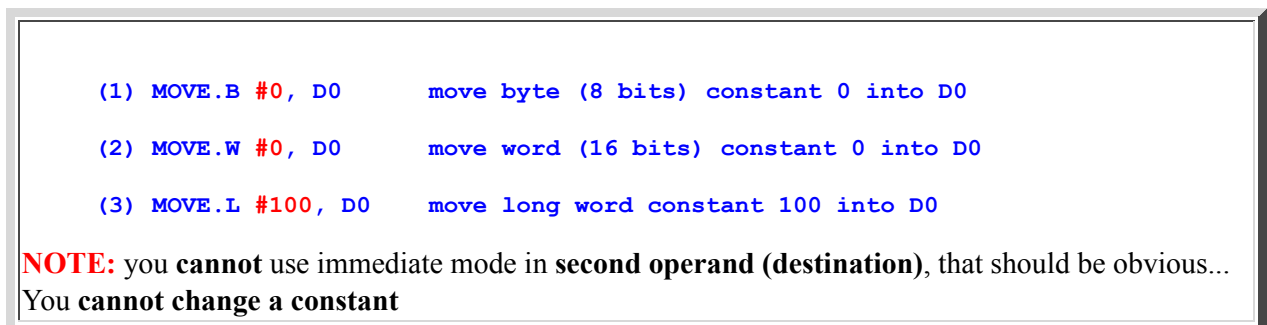
- Operand specified with the immediate addressing mode is a constant operand
- Recall the context that the address mode is used within the MOVE instruction:



- Syntax to specify the *immediate* addressing mode:



- Semantics (meaning):
 - The operand is the constant number *n*
- Examples:



- Constants in other number base systems:
 - \$-prefix indicates a hexadecimal constant, e.g.: \$FA1F
 - @-prefix indicates a octal constant, e.g.: @70167
 - %-prefix indicates a binary constant, e.g.: %11011011

- Examples:

```

MOVE.B #%10101010, D0  move byte binary number 10101010 into D0
MOVE.W #$FFFF, D0     move word constant FFFF (hex) into D0
  
```

- Advanced examples:

```
(1) MAX: EQU 100
      ...
      MOVE.L #MAX,D0      move 100 into reg. D0
```

- You must remember that **every** symbolic name will be replaced by its corresponding number.
- **MAX** is defined to be equal to **100**
- So, the **MOVE** instruction will become (after replacing **MAX** by **100**):

```
MOVE.L #MAX,D0  --> MOVE.L #100,D0
```

- Result: move 100 into reg. D0

```
(2)      MOVE.L #A,D0      move the address of the variable A in D0
      ...
      A: DS.L 10
```

- You must remember that:

```
A: DS.L 10
```

equates the symbolic name **A** to the **address of the memory location** where the (array) variable is defined.

- So the symbolic name **A** will be replaced by this address

- **Example Program:** (Demo above code)

Example

- Prog file: [click here](#)

- **The magic of symbolic names revealed:**

```
006000          org $6000
000007      MAX:  EQU 7
006000 203C      move.l #-1, d0
           FFFF
           FFFF      FFFFFFFF = -1 (in 32 bits)
006006 303C      move.w #1, d0
```

```

0001                                0001 = 1 (in 16 bits)
00600A 103C        move.b #MAX, d0
0007                                MAX is replaced by 7 !
00600E 207C        move.l #A, a0
0000
601A                                A is replaced by 0000601A ! (addr is 32 bits)
006014 227C        move.l #B, a1
0000
6042                                B is replaced by 00006042 !

00601A    A:      ds.l 10    <---- array of 10 int at A
006042    B:      ds.l 10    <---- array of 10 int at B
00606A                                end

```

- You can see that the symbolic name **A** is equated to the hexadecimal number **00601A**

The "**move.l #A, a0**" replaces **A** by **0000601A**

- You can also see that the symbolic name **B** is equated to the hexadecimal number **006042**

The "**move.l #B, a1**" replaces **B** by **00006042**

- **So:**

YOU write	Assembler knows that:	Final assembler instr:
<code>move.b #MAX, d0</code>	<code>MAX == 7 (dec)</code>	<code>move.b #7, d0</code>
<code>move.l #A, d0</code>	<code>A == 601A (hex)</code>	<code>move.l #601A, d0</code>
<code>move.l #B, d0</code>	<code>A == 6042 (hex)</code>	<code>move.l #6042, d0</code>