Addressing Modes in 8086

Introduction:

An instruction consists of an opcode and an operand. The operand may reside in the accumulator, or in a general purpose register or in a memory location. The manner in which an operand is specified (or referred to) in an instruction is called ***addressing mode***. In this lecture, the **MOV** (move data) instruction is used to describe the operand-addressing modes. Fig. 3-1 shows the MOV instruction.

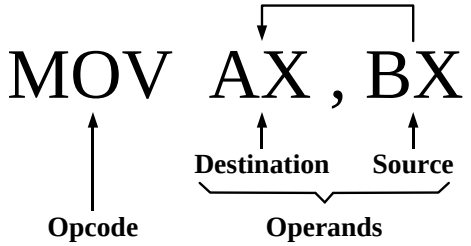

Fig. 3-1: The MOV instruction.

Register Addressing:

With this type of addressing, the operand to be accessed is specified as residing in an internal register of 8086. All registers of 8086 can be used except flag register. Table 3-1 shows many variations of register move instructions. A segment-to-segment register MOV instruction is the only type of MOV instruction not allowed. Note that the code segment register is not normally changed by a MOV instruction because the address of the next instruction is found in both IP and CS. If only CS were changed, the address of the next instruction would be unpredictable. Therefore, changing the CS register with a MOV instruction is not allowed.

Table 3-1: Examples of the register-addressed instructions.

Assembly Language	Size	Operation
MOV AL , BL	8-bits	Copies BL into AL
MOV CH , CL	8-bits	Copies CL into CH
MOV AX , CX	16-bits	Copies CX into AX
MOV SP , BP	16-bits	Copies BP into SP
MOV DS , AX	16-bits	Copies AX into DS
MOV SI , DI	16-bits	Copies DI into SI
MOV BX , ES	16-bits	Copies ES into BX
MOV CX , BX	16-bits	Copies BX into CX
MOV SP , DX	16-bits	Copies DX into SP

MOV ES , DS		Not allowed (segment-to-segment)
MOV BL , DX		Not allowed (mixed sizes)
MOV CS , AX		Not allowed (the code segment register may not be the destination register)

Immediate Addressing:

The term ***immediate*** implies that data immediately follow the hexadecimal opcode in the memory. Table 3-2 shows many variations of immediate move instructions.

Table 3-2: Examples of the immediate-addressed instructions.

Assembly Language	Size	Operation
MOV BL , 44	8-bits	Copies a 44 decimal into BL
MOV AX , 44H	16-bits	Copies a 0044H into AX
MOV SI , 0	16-bits	Copies a 0000H into SI
MOV CH , 100	8-bits	Copies a 100 decimal into CH
MOV AL , 'A'	8-bits	Copies an ASCII A into AL
MOV AX , 'AB'	16-bits	Copies an ASCII BA into AX
MOV CL , 11001110B	8-bits	Copies a 44 decimal into BL

Memory Addressing:

To reference an operand in memory, the 8086 must calculate the physical address (PA) of the operand and then initiate a read or write operation of this storage location. The 8086 MPU is provided with a group of addressing modes known as the memory operand addressing modes for this purpose. Physical address can computed from a segment base address (SBA) and an effective (offset) address (EA). SBA identifies the starting location of the segment in memory, and EA represents the offset of the operand from the beginning of this segment of memory.

$$\begin{aligned} PA &= SBA\ (segment) : EA\ (offset) \\ &= Segment : Base + Index + Displacement \end{aligned}$$

$$PA = \begin{Bmatrix} CS \\ DS \\ SS \\ ES \end{Bmatrix} : \begin{Bmatrix} BX \\ BP \end{Bmatrix} + \begin{Bmatrix} SI \\ DI \end{Bmatrix} + \begin{Bmatrix} 8-bit\ displacement \\ 16-bit\ displacement \end{Bmatrix}$$

Generally, not all these elements are always used in the effective address calculation. In fact, a number of memory addressing modes are

defined by using various combinations of these elements. Next, we will examine each of the memory operand addressing modes in detail.

⋖ 1. Direct Addressing:

Direct addressing mode is similar to immediate addressing in that information is encoded directly into the instruction. However, in this case, the instruction opcode is followed by an effective address, instead of the data, as shown below:

$$\text{PA} = \begin{Bmatrix} \text{CS} \\ \text{DS} \\ \text{SS} \\ \text{ES} \end{Bmatrix} : \{\text{Direct Address}\}$$

↑
Default is DS

Direct addressing with a MOV instruction transfer data between a memory location, located within the data segment, and the AL, CL, AX or CX register.

Example 3-1:

The instruction:

MOV AL , DS : [2000 H] or MOV AL , [2000 H]

move the contents of the memory location with offset 2000 in the current data segment into internal register AL. The symbol [ ] mean contents of the memory.

⋖ 2. Register Indirect Addressing:

This mode is similar to the direct address except that the effective address held in any of the following registers: BP, BX, SI, and DI, as shown below:

$$\text{PA} = \begin{Bmatrix} \text{CS} \\ \text{DS} \\ \text{SS} \\ \text{ES} \end{Bmatrix} : \begin{Bmatrix} \text{BX} \\ \text{BP} \\ \text{SI} \\ \text{DI} \end{Bmatrix}$$

↑
Default is DS

Example 3-2:

The instruction:

MOV AX , DS : [SI] or MOV AX , [SI]

move the contents of the memory location that is offset from the beginning of the current data segment by the value of effective address in register SI into internal register AX. For instance, if DS contains 0200H and SI contains 1234H, the result produced by executing the instruction is that the contents of the memory location at address:
PA = DS × 10 + SI = 0200H ×10 + 1234H = 02000H + 1234H = 03234H
are moved to the AX register.

➢ 3. Based Addressing:

In the based addressing mode, the effective address of the operand is obtained by adding a direct or indirect displacement to the contents of either base register BX or base pointer register BP. The physical address is calculated as shown below:

$$PA = \begin{Bmatrix} CS \\ DS \\ SS \\ ES \end{Bmatrix} : \begin{Bmatrix} BX \\ BP \end{Bmatrix} + \begin{Bmatrix} 8-\text{bit displacement} \\ 16-\text{bit displacement} \end{Bmatrix}$$

Example 3-3:

If DS = 0600H, BX = 1000H, and AL = E0H, for the following instruction:

MOV DS : [BX] + 1234H , AL

EA = BX + 1234H = 1000H + 1234H = 2234H

PA = DS × 10 + EA = 0600H × 10 + 2234H = 06000H + 2234H = 08234H

so it writes the contents of source operand AL (E0H) into the memory location 08234H.

If BP is used instead of BX, the calculation of the physical address is performed using the contents of the stack segment (SS) register instead of DS. This permits access to data in the stack segment of memory.

➢ 4. Indexed Addressing:

In the Indexed addressing mode, the effective address of the operand is obtained by adding a direct or indirect displacement to the contents of either SI or DI register. The physical address is calculated as shown below:

$$PA = \begin{Bmatrix} CS \\ DS \\ SS \\ ES \end{Bmatrix} : \begin{Bmatrix} SI \\ DI \end{Bmatrix} + \begin{Bmatrix} 8-\text{bit displacement} \\ 16-\text{bit displacement} \end{Bmatrix}$$

Example 3-4:

If DS = 3000H, SI = 6633H, and AL = A0H, for the following instruction:

MOV DS : [SI] + 0012H , AL

EA = SI + 0012H = 6633H + 0012H = 6645H

PA = DS × 10 + EA = 3000H × 10 + 6645H = 30000H + 6645H = 36645H

so it writes the contents of source operand AL (A0H) into the memory location 36645H.

➢ 5. Based-Indexed Addressing:

By combining the based addressing mode and the indexed addressing mode results in a new, more powerful mode known as based-indexed addressing mode. This addressing mode can be used to access complex data structures such as two-dimensional arrays. As shown below, this mode can be used to access elements in an (m × n) array of data. Notice that the displacement, which is a fixed value, locates the array in memory. The base register specifies the m coordinate of the array, and the index register identifies the n coordinate. Simply by changing the values in the base and index registers permits access to any element in the array.

$$\text{PA} = \begin{Bmatrix}\text{CS}\\ \text{DS}\\ \text{SS}\\ \text{ES}\end{Bmatrix} : \begin{Bmatrix}\text{BX}\\ \text{BP}\end{Bmatrix} + \begin{Bmatrix}\text{SI}\\ \text{DI}\end{Bmatrix} + \begin{Bmatrix}8-\text{bit displacement}\\ 16-\text{bit displacement}\end{Bmatrix}$$

Example 3-5:

If DS = A000H, BX =1000H, and SI = 1234H, for the following instruction:

MOV AL , DS : [BX] [SI] + 0012H

EA = BX + SI + 0012H = 1000H + 1234H + 0012 = 2246H

PA = DS × 10 + EA = A000H × 10 + 2246H = A0000H + 2246H = A2246H

so it writes the contents of the memory location A6645H into the destination operand AL register.