Chapter Three Addressing Mode

The 8086 Addressing Mode

The 8086 Addressing Mode When the 8086 executes an instruction, it performs the specified function on data. The data are called its operands and may be part of the instruction reside in one of the internal registers of the 8086, stored at an address in memory, or held at an I/O port. To access these different types of operands, the 8086 is provided with various addressing modes:

1. **Register Addressing Mode** :- With the register addressing mode, the operand to be accessed is specified as residing in an internal register of the 8086, an example of an instruction that uses this addressing mode is

MOV AX, BX

This stands for move the contents of BX, the source operand, to AX, the destination operand. Both the source and destination operands have been specified as the content of the internal registers of the 8086. See Figure 10(a, b).

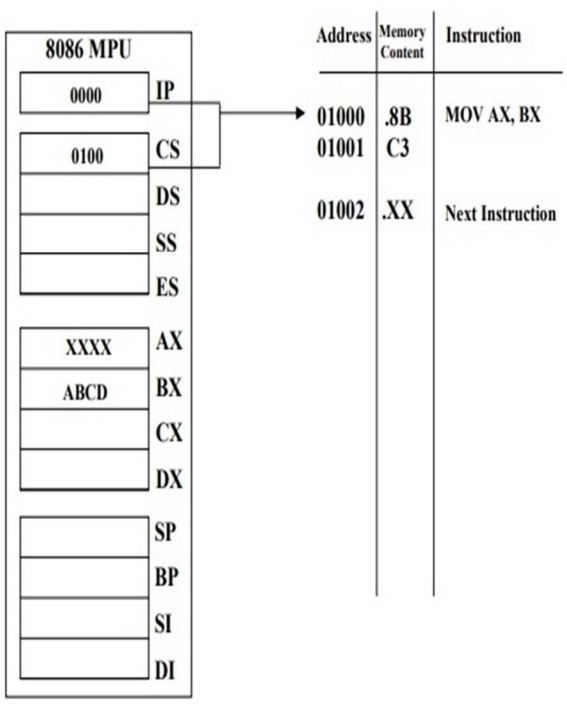

Figure 10(a): Register addressing mode before execution

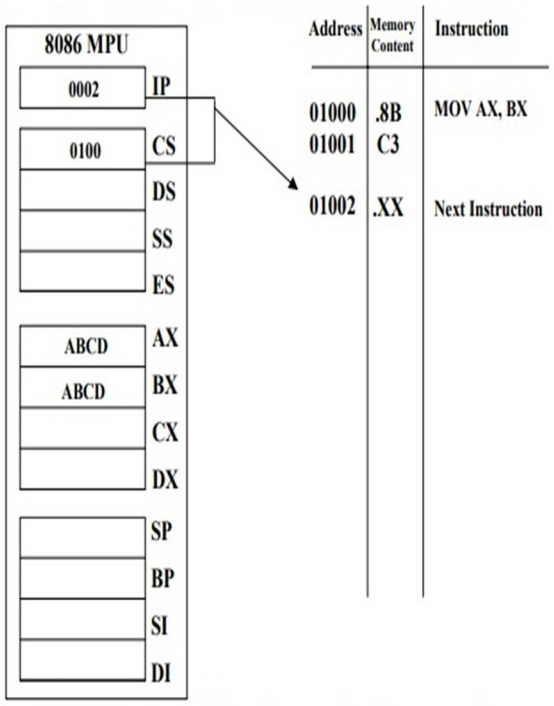

Figure 10(b): Register addressing mode after execution.

2. **Immediate Addressing Mode**:- If a source operand is part of the instruction instead of the contents of a register or memory location, it represents what is called an immediate operand and is accessed using the immediate addressing mode. Typically, immediate operands represent constant data. Immediate operands can be either a byte or word of data. In the instruction

MOV AL, 015H

The source operand 15H is an example of a byte-wide immediate source operand. Note that the value of the immediate operand must always be preceded by a zero. See Figure 11(a, b).

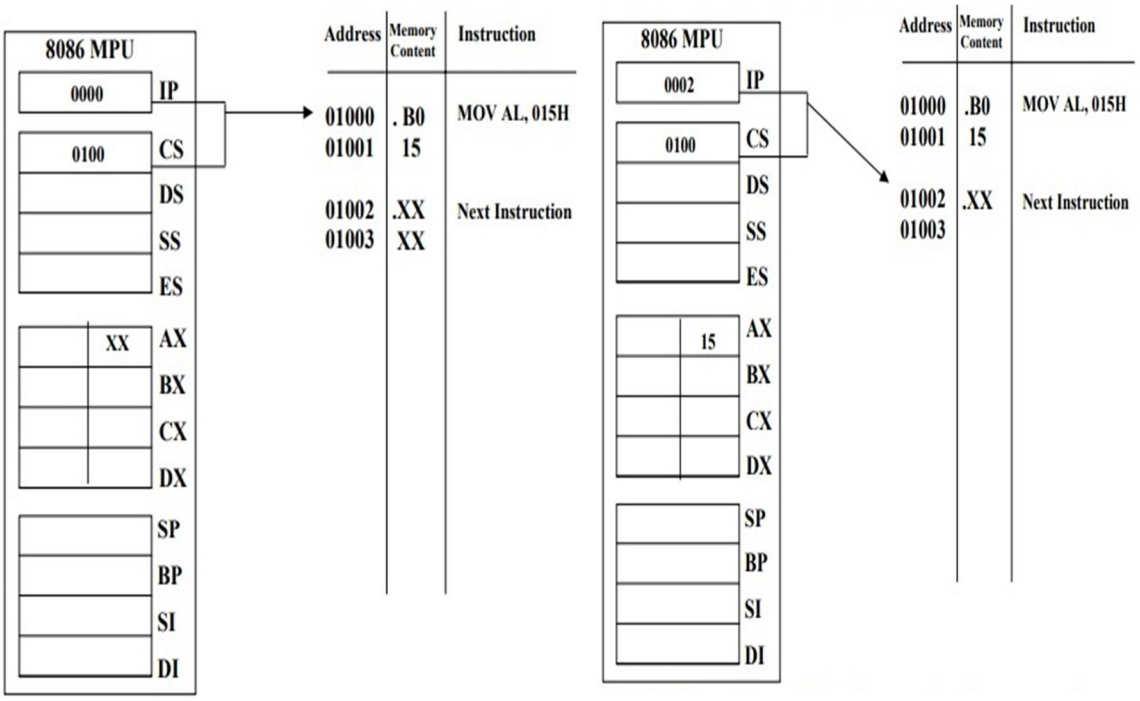

Figure 11(a): Immediate addressing mode before execution. **Figure 11(b): Immediate addressing mode after execution.**

3. Direct Addressing Mode:- Direct addressing differs from immediate addressing in that the locations following the instruction opecode hold an **effected memory address (EA)** instead of data. This effective address is a 16-bit offset of the storage location of the operand from the current value in the

data segment (DS) register. EA is combined with the contents of DS in the BIU to produce the **physical address** for its source operand is

MOV CX, BETA

This stands for move the contents of the memory location which is offset by BETA from the current value in DS into internal register CX. See Figure 12(a, b). Notice that the value assigned to constant BETA is 1234H.

PA = 02000H + 1234H

= 03234H

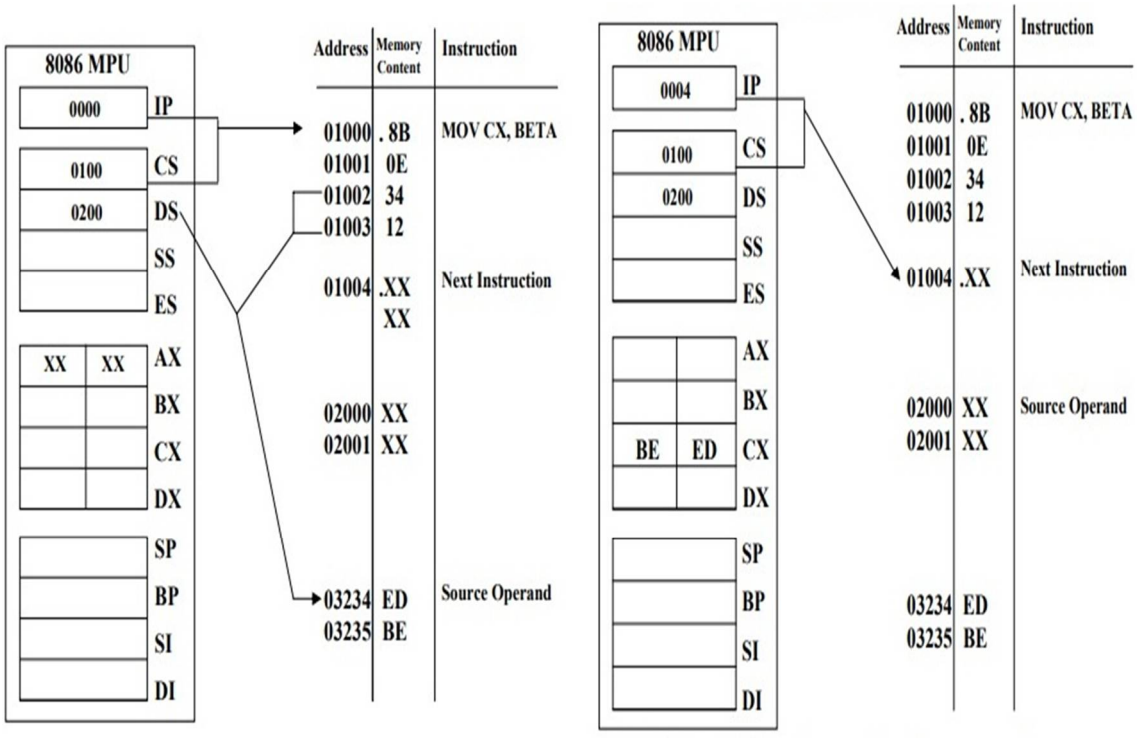

Figure 12(a): Direct Addressing mode before execution. **Figure 12(b): Direct Addressing mode after execution.**

4. **Register Indirect Addressing Mode**:- Register indirect addressing is similar to direct addressing in that an effective address is combined with the contents of DS to obtain a physical address. However, it differs in the way the offset is

specified. This time EA resides in either a pointer register or index register within the 8086. The pointer register can be either BX or BP and the index register can be SI or DI.

MOV AX, [SI]

This instruction moves the contents of the memory location offset by the value of EA in SI from the current value in DS to the AX register. See Figure 13(a, b). SI contains 1234H and DS contains 0200H.

PA = 02000H + 1234H

= 03234H

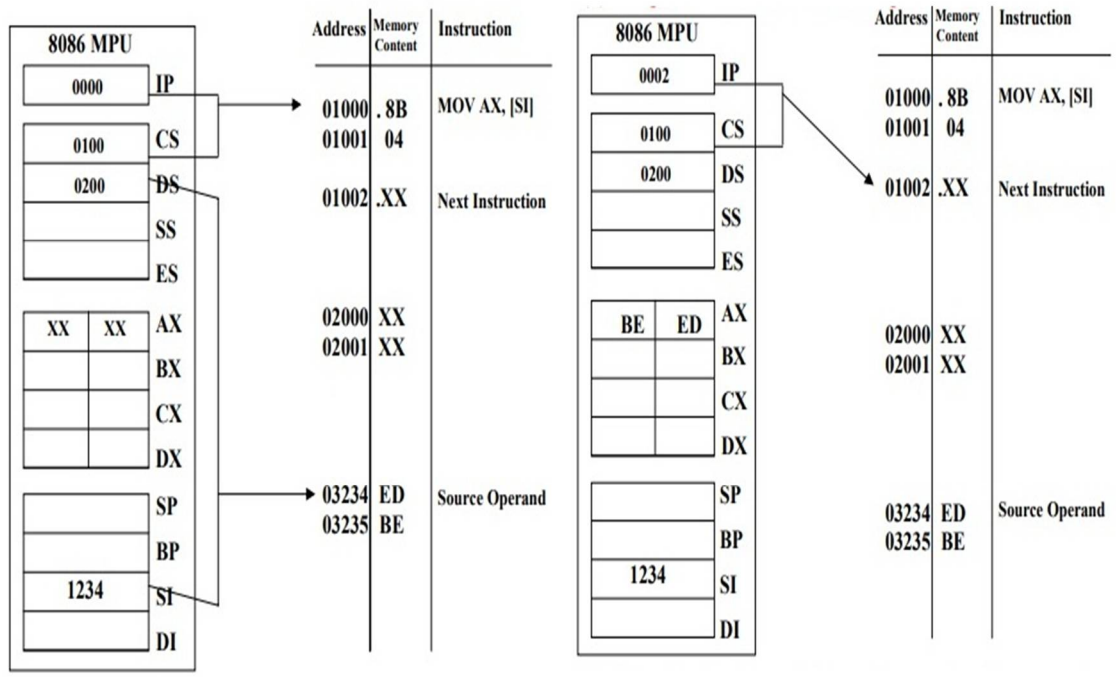

Figure 13(a): Register Indirect Addressing before execution. **Figure 13(b): Register Indirect Addressing mode after execution.**

5. <u>Based Addressing Mode</u>:- In the based addressing mode, the physical address of the operand is obtained by adding a direct or indirect displacement to the contents of either BX or BP and the current value in DS and SS,

respectively. A MOV instruction that uses based addressing to specify the location of its destination operand is as follows:

MOV [BX].BETA, AL

As shown in Figure 14(a, b) the fetch and execution of this instruction causes the BIU to calculate the physical address of the destination operand from the contents of DS, BX, and the direct displacement. The result is

PA = 02000H + 1000H + 1234H

= 04234H

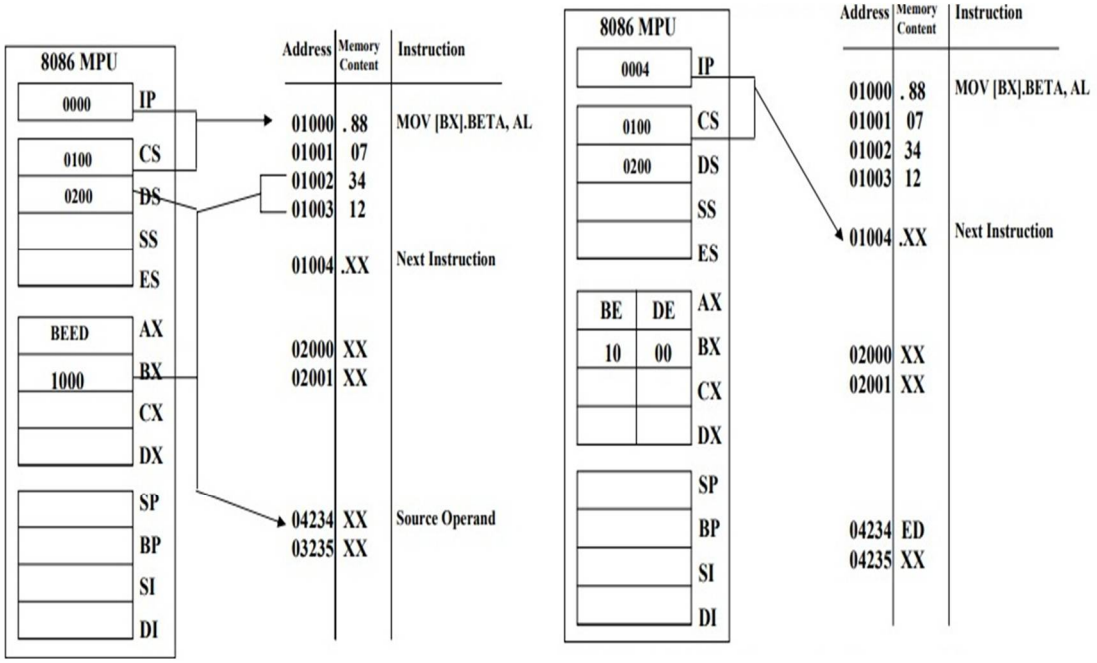

Figure 14(a): Based Addressing before execution. **Figure 14(b): Based Addressing mode after execution.**

6. <u>Indexed Addressing Mode:-</u> Indexed addressing works identically to the based addressing, it uses the contents of one of the index registers, instead of BX or BP, in the generation of the physical address, here is an example:

MOV AL, ARRAY[SI]

The example in Figure 15(a,b) shows the result of executing the MOV instruction. First the physical address for the source operand is calculated from DS, SI, and the direct displacement.

PA = 02000H + 2000H + 1234H

= 05234H

Then the byte of data stored at this location, which is BEH is read into lower byte AL of the accumulator register.

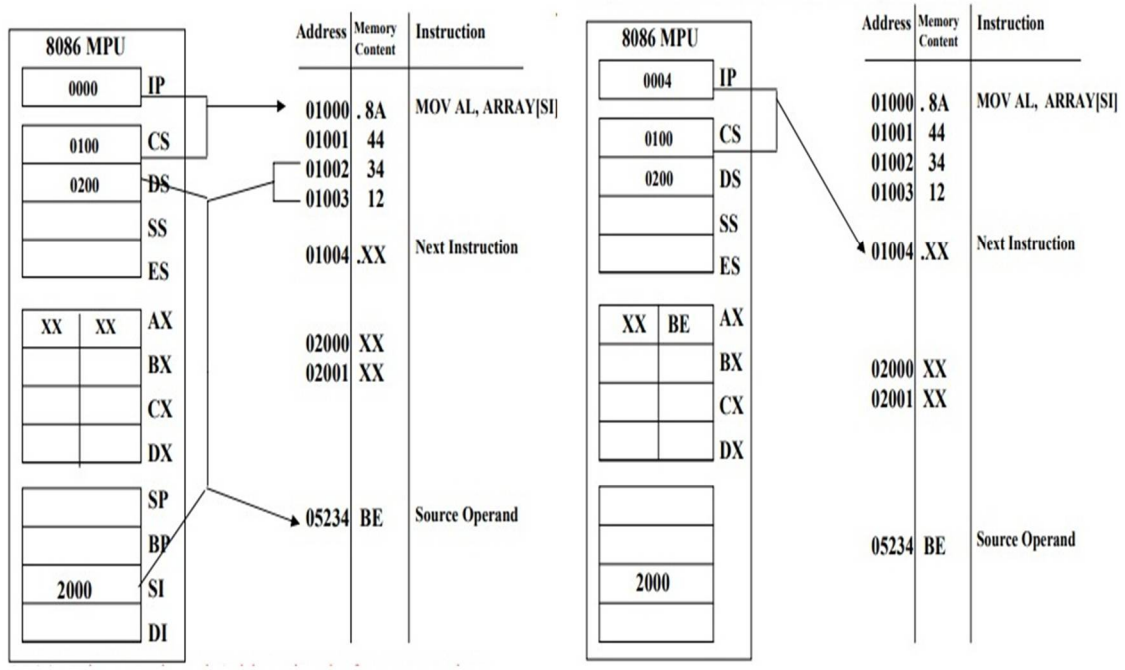

Figure 15(a):Direct Indexed Addressing before execution. Figure 15(b):Direct Indexed Addressing mode after execution.

7. <u>Based Indexed Addressing Mode</u>:- Combining the based addressing mode and the indexed addressing mode together results in a new, more powerful mode known as based indexed addressing. Let us consider an example of a MOV instruction using this type of addressing.

MOV AH, [BX].BETA[SI]

An example of executing this instruction is illustrated in Figure 16(a,b). The address of the source operand is calculated as

PA = 02000H + 1000H + 1234H + 2000H

= 06234H

Execution of this instruction causes the Value stored at this location to be written into AH.

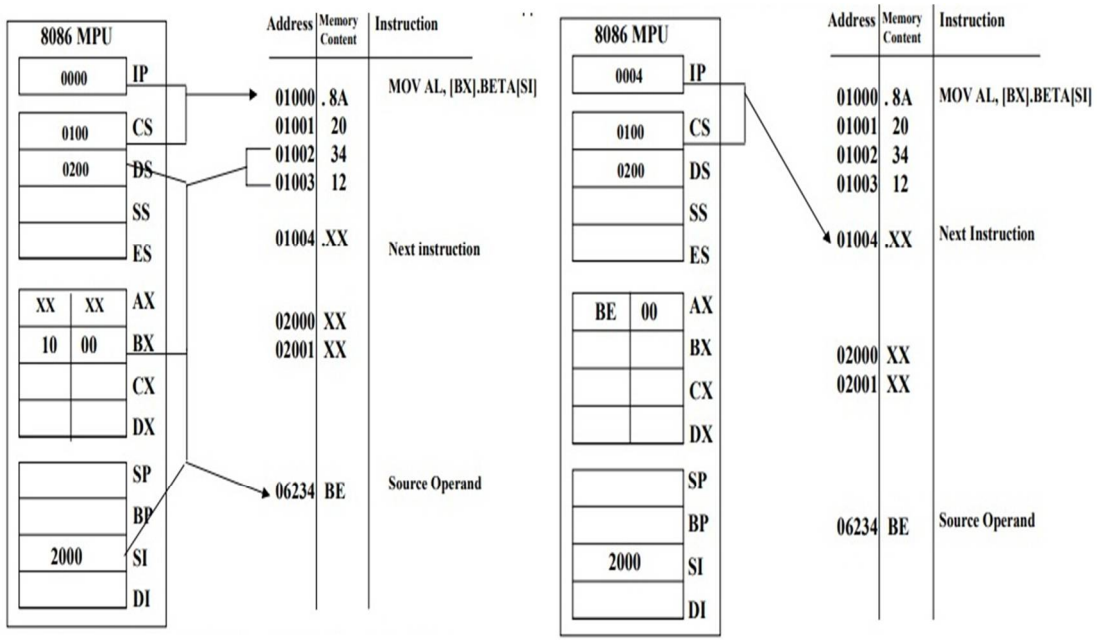

Figure 16(a):Based Indexed Addressing before execution. Figure 15(b):Based Indexed Addressing mode after execution.

8. <u>String Addressing Mode:-</u> The string instructions of the 8086's instruction set automatically use the source and destination index registers to specify the effective addresses of the source and destination operands, respectively. The move string instruction

MOVS

is an example. Notice that neither SI nor DI appears in the string instruction, but both are used during its execution.

9. Port Addressing Mode:- Port addressing is used in conjunction with the IN and OUT instructions to access input and output ports. Any of the memory addressing modes can be used for the port address for memory mapped ports. For ports in the I/O address space, only the **Direct addressing mode** and **an Indirect addressing mode** using DX are available. For example, **Direct addressing** of an input port is used in the instruction

IN AL, 15H

This stands for input the data from the byte wide input port at address 15H of the I/O address space to register AL. Next, let us consider another example. Using **Indirect port addressing** for the source operand in an IN instruction, we get:

IN AL, DX

It means input the data from the byte wide input port whose address is specified by the contents of register DX. For instance, if DX equals 1234H the contents of the port at this I/O address are loaded into AL.

Problems

- Which register holds a count for some instruction?
- What is the purpose of the IP register?
- The carry flag bit is set by which arithmetic operation?
- A number that contain 3 one bit said to have parity?
- Find the memory address of the next instruction execute by the micro processor for the following CS:IP combinations:

a. CS=1000H and IP=2000H

b. CS=2000H and IP=1000H

- Which register or registers are used as an offset address for string instruction destination in the microprocessor?
- The stack memory is addressed by a combination of the segment plus offset.
- Which registers of the 8086 are used in memory segmentation?
- Categorize each flag bit of the 8086 as either a control flag or as a flag to monitor the effect of instruction execution.
- identify the three part of an assembly language instruction in each of the following statement:

```
AGAIN: ADD AX, CX; ADD THE REGISTERS
MOV BX, AX; SAVE RESULT
```

- Identify the source and destination operand for each of the statements in 10.