8086 instruction set and Assembly language program:

An **instruction** is a command to the mp to perform a given task on specified data
Each instruction has two parts: -

1- The task to be performed, called the operation code (**opcode**)
2- The data to be operated on, called the **operand**

- The entire group of instructions, called the **instruction set**
- The 8086 instruction set contains of 117 basic instructions
- The instructions are organized into groups ,These groups consist of
  - Data transfer instructions
  - Arithmetic instructions
  - Logic instructions
  - Shift instructions
  - Rotate instructions
  - Flag control instructions
  - Compare instruction
  - Control flow and jump instructions
  - Loop instruction
  - String instruction

MOV INSTRUCTION

The move instruction is one of the instructions in the data transfer group of the 8086 instruction set.
The MOV instruction is common and flexible to describe the direction of flow of data. And The format of this instruction, as shown in Figure below, is written in general as

MOV D, S

Mnemonic	Meaning	Format	Operation	Flags affected
MOV	Move	MOV D,S	(S) → (D)	None

Its operation is described in general as

(S) —> (D)

That is, execution of the instruction transfers a byte or a word of data from a source location to a destination location. These data locations can be internal registers and storage locations in memory. Figure below shows the valid source and destination variations.

Destination	Source
Memory	Accumulator
Accumulator	Memory
Register	Register
Register	Memory
Memory	Register
Register	Immediate
Memory	Immediate
Seg-reg	Reg16
Seg-reg	Mem16
Reg16	Seg-reg
Memory	Seg-reg

This large choice of source and data locations results in many different move instructions. Looking at this list, we see that data can be moved between general-purpose registers, between a general-purpose register and a segment register, between a general-purpose register or segment register and memory, or between a memory location and the accumulator.

Example**:** **MOV AX,BX**

The flow of data takes place from right to left register i.e., from BX register (source register) to AX register (destination register). It is read as copy the contents of BX register into AX register. The destination and source registers are called **operands.**

Addressing modes:

When the 8086 executes an instruction, it performs the specified function on data. These data, called operands, may be part of the instruction, may reside in one of the internal registers of the microprocessor, may be stored at an address in memory, or may be held at an I/O port.

To access these different types of operands, the 8086 is provided with various addressing modes.

An **addressing mode** is a method of specifying an operand.

The addressing modes are categorized into three types:

- register operand addressing,
- immediate operand addressing, and
- memory operand addressing.

➢ Register operand addressing mode

With register operand addressing mode, the operand to be accessed is specified as residing in an internal register.

EX:

MOV AX, BX

Fig -1- below shows the memory and registers before and after the execution of this instruction:

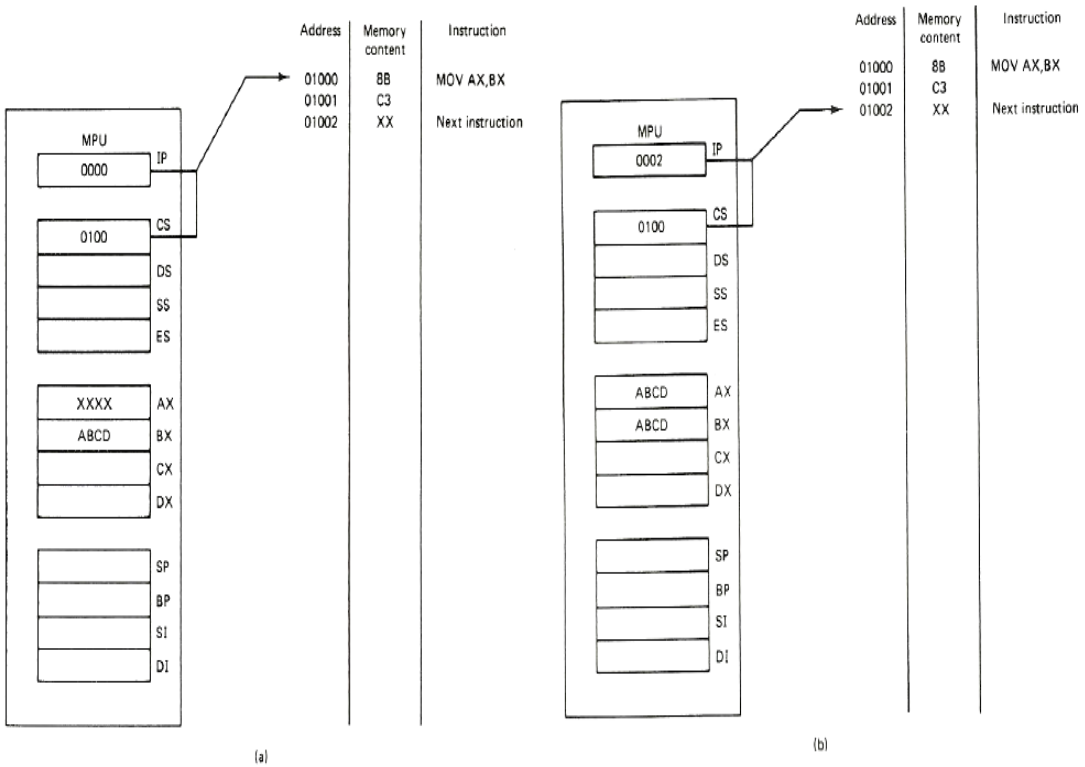


Fig-1-(a) before fetching and execution (b) after execution

- **Immediate operand addressing mode**

If an operand is part of the instruction instead of the contents of a register or memory location, it represents what is called an immediate operand. The operand, which can be 8 bits (Imm8) or 16 bits (Imml6) in length, is encoded as part of the instruction. This addressing mode can only be used to specify a source operand.

EX: In the instruction

MOV AL, 15 H

the source operand 15 H is an example of a byte-wide immediate source operand.

Figures -2- (a) and (b) illustrate the fetch and execution of this instruction. Here we find that the immediate operand 15 H is stored in the code segment of memory in the byte location immediately following the opcode of the instruction. This value is fetched, along with the opcode for MOV, into the instruction queue within the 8086. When it performs the move operation, the source operand is fetched from the queue, not from the memory, and no external memory operations are performed. Note that the result produced by executing this instruction is that the immediate operand, which equals 15 H, is loaded into the lower-byte part of the accumulator (AL).

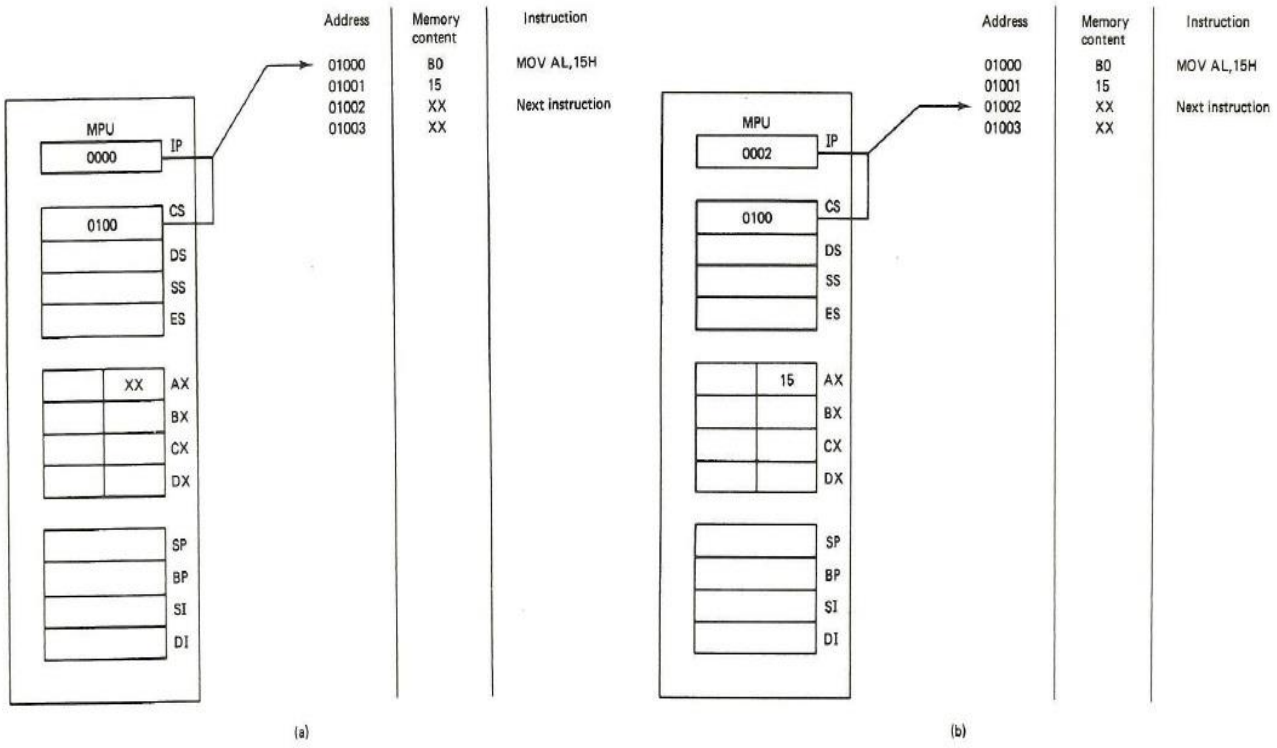


Fig-2-(a) before fetching and execution (b) after execution

Ex:

What is the result of MOV CX,7 ?

Because register CX is specified, the immediate value 7 is coded as the 2-byte number 00 07.

	CH	CL
CX	00	07

Ex:

MOV AX, 1234H

In this example, the data 1234H is moved to the AX register.

- **Memory Operand addressing modes:**

The 8086 use this mode to reference an operand in memory. The 8086 must calculate the physical address of the operand and then initiate a read or write operation of this storage location.

As shown in fig-a- the physical address of the operand is calculated from a *segment base address* (**SBA**) and an *effective address* (**EA**). :

PA = SBA : EA

PA = Segment base : Base + Index + Displacement

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

Fig-a-

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. Fig-a- shows that an effective address can be made up from as many as three elements: the base, index, and displacement. Using these elements, the effective address calculation is made by the general formula

EA = Base + Index + Displacement

This Figure also identifies the registers that can be used to hold the values of the segment base, base, and index. For example, it tells us that any of the four segment registers can be the

source of the segment base for the physical address calculation and that the value of base for the effective address can be in either the base register (BX) or base pointer register (BP) also this Figure identifies the sizes permitted for the displacement.

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. These modes includes five types:-

1- Direct addressing:

Direct addressing made 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. Fig-b- show this effective address is used directly as the 16-bit offset of the storage location of the operand from the location specified by the current value in the selected segment register.

Fig-b-

The default segment register is **DS**. Therefore, the 20-bit physical address of the operand in memory is normally obtained from logical address **DS:EA**.

Example: an instruction that uses direct addressing mode for its source operand is

MOV CX, [1234H]

This stands for "move the contents of the memory location with offset 1234 H in the current data segment into internal register CX." The offset is encoded as part of the instruction's machine code.

In Fig-3-a, we find that the offset is stored in the two byte locations that follow the instruction's opcode. As the instruction is executed, the 8086 combines 1234H with 0200H to get the physical address of the source operand as follows:

PA = 02000 H + 1234 H = 03234H

Then it reads the word of data starting at this address, which is BEED H, and loads it into the CX register. This result show in Fig- 3-b.

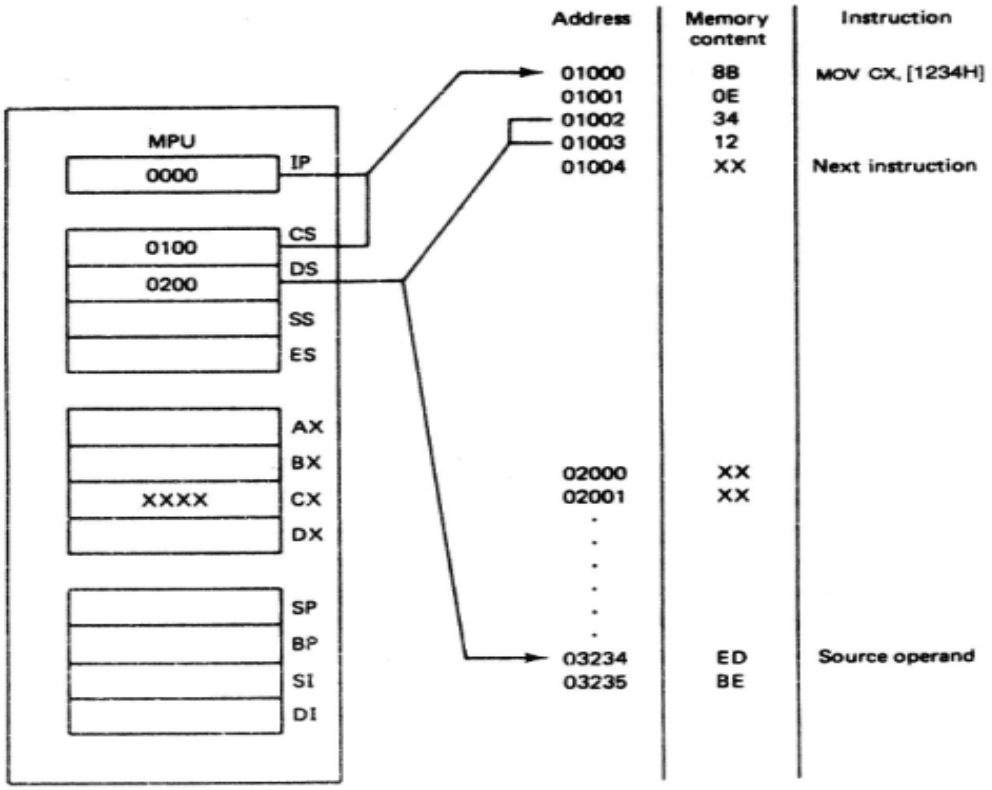


Fig-3- (a)

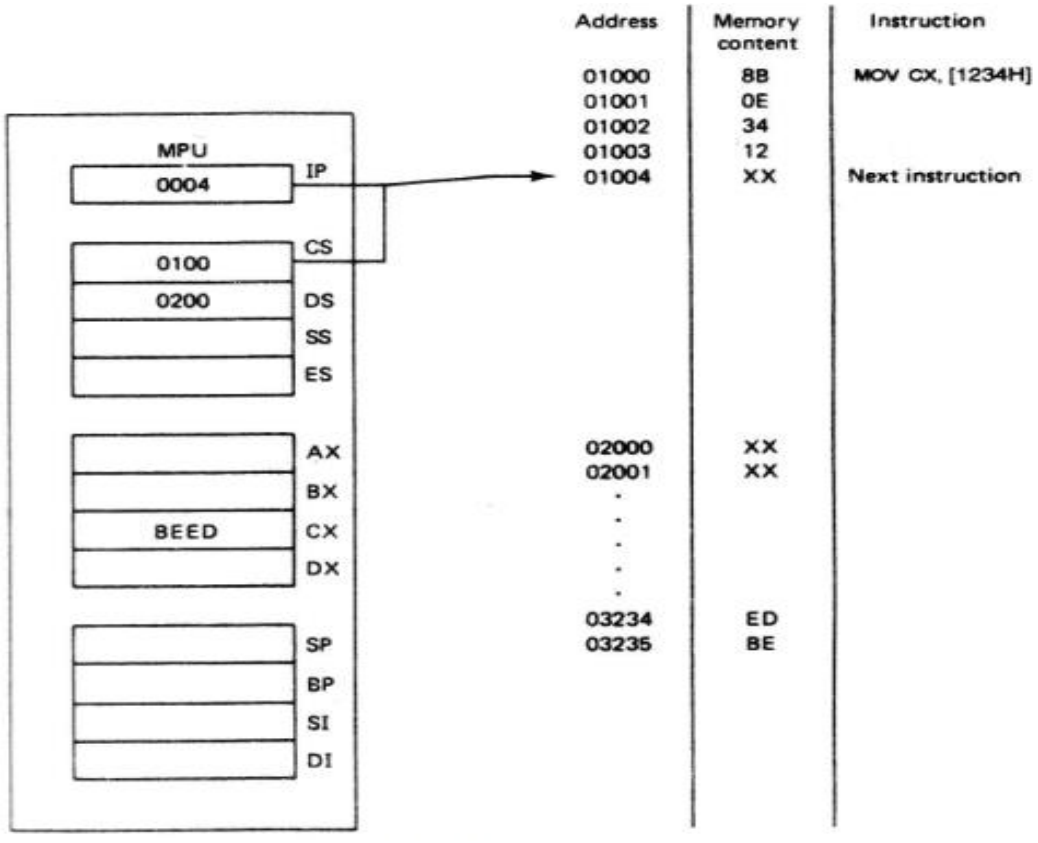


Fig-3- (b)

Fig-3-(a) before fetching and execution (b) after execution

2- Register indirect addressing:

this mode is similar to the direct addressing but the offset is specified in a base register (BX), base pointer (BP) or an index register (SI or DI) within the 8086. Fig -4-below shows the memory and registers before and after the execution of instruction:

MOV AX, [SI]

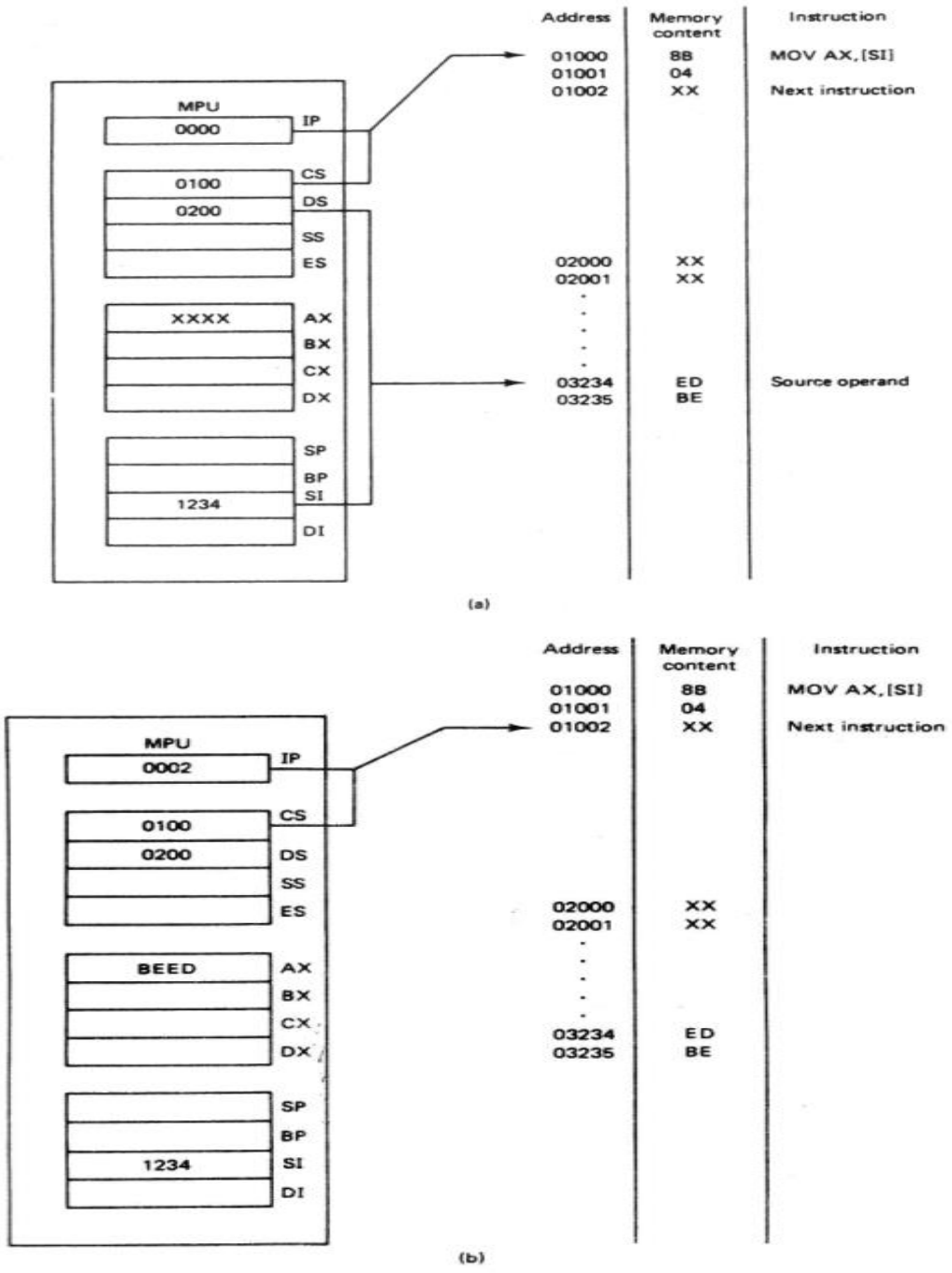


Fig-4-(a) before fetching and execution (b) after execution

3- Based addressing:

this mode, the effective address is obtained by adding a direct or indirect displacement to the contents of either base register **BX** or Base pointer register **BP**. Fig -5-below shows the memory and registers before and after the execution of instruction:

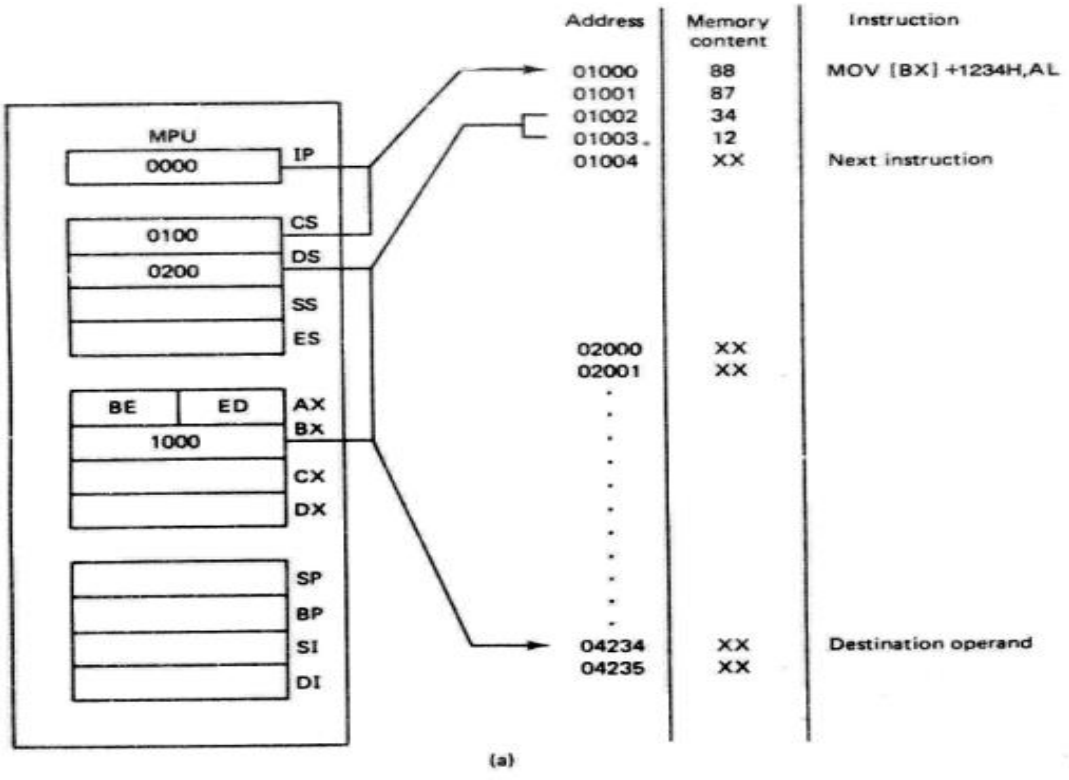
MOV [BX]+1234H, AL

This instruction uses base register BX and direct displacement 1234H to derive the EA of the destination operand. The based addressing mode is implemented by specifying the base register in brackets followed by a + sign and the direct displacement. The source operand in this example is located in byte accumulator AL.

the fetch and execution of this instruction cause the 8086 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

Then it writes the contents of source operand AL into the storage location at 04234H. The result is that ED H is written into the destination memory location.



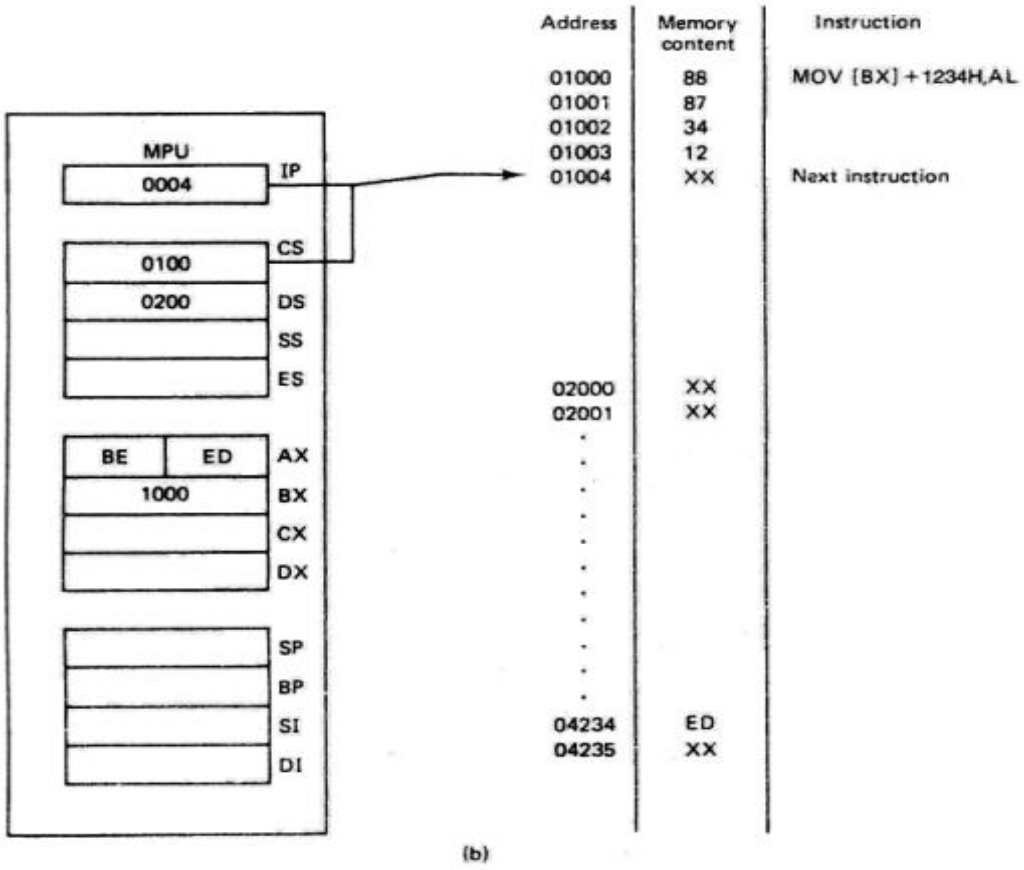


Fig-5-(a) before fetching and execution (b) after execution

Note that :- 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.**

Note that:- The displacement could be **8 bits** or **16 bits**

4- Indexed addressing:

this mode, work in similar manner to that of the based addressing mode but the effective address is obtained by adding the displacement to the value in an index register (**SI** or **DI**). Fig -6-below shows the memory and registers before and after the execution of instruction:

MOV AL, [SI]+1234H

the physical address of the source operand is calculated from the contents of DS, SI, and the direct displacement.

PA = 02000H + 2000H + 1234H = 05234H

Then the byte of data stored at this location, BE H, is read into the lower byte (AL) of the accumulator register.

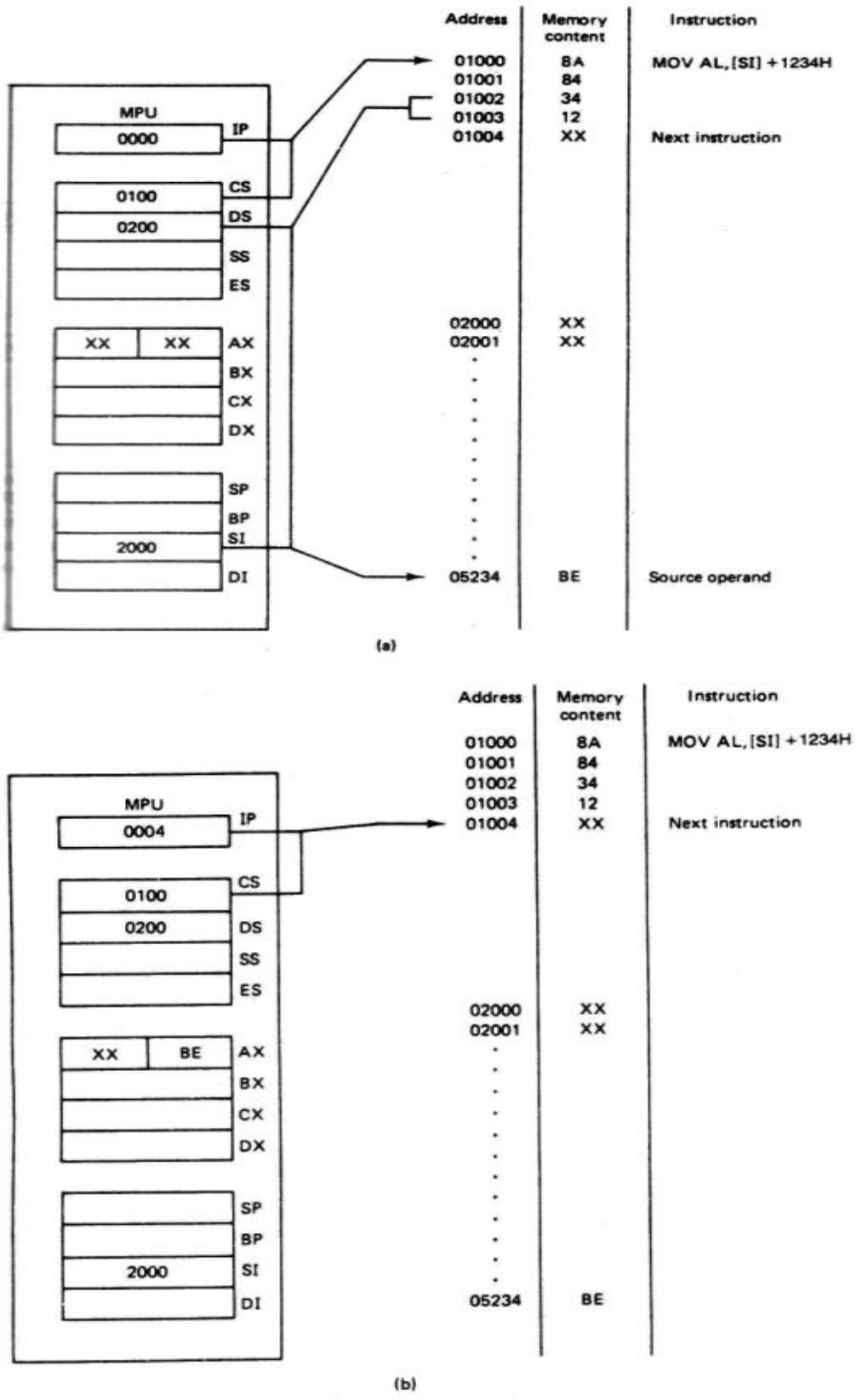


Fig-6-(a) before fetching and execution (b) after execution

Note that:- The displacement could be **8 bits** or **16 bits**

5- Based-Indexed addressing:

this mode combines the based addressing mode and indexed addressing mode. Fig-7- below shows the memory and registers before and after the execution of instruction:

MOV AH, [BX][SI]+1234H

Note that the source operand is accessed using based-indexed addressing mode. Therefore, the effective address of the source operand is obtained as

EA = (BX) + (SI) + 1234H

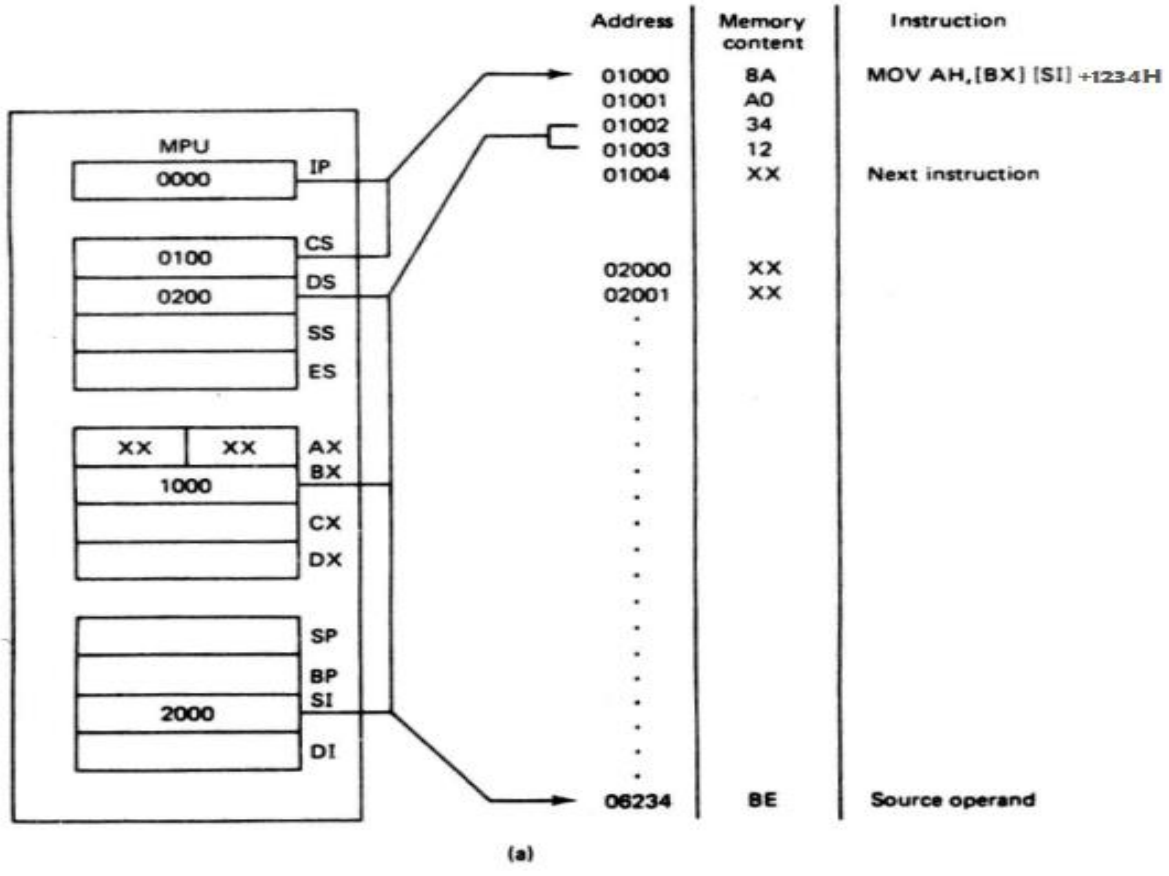
and the physical address of the operand is computed from the current contents of DS and the calculated EA.

PA = DS:(BX) + (SI) + 1234H

Using the contents of the various registers in the example, the address of the source operand is calculated as

PA = 02000H + 1000H + 2000H + 1234H = 06234H

Execution of the instruction causes the value stored at this location in memory to be read into AH.



(a)

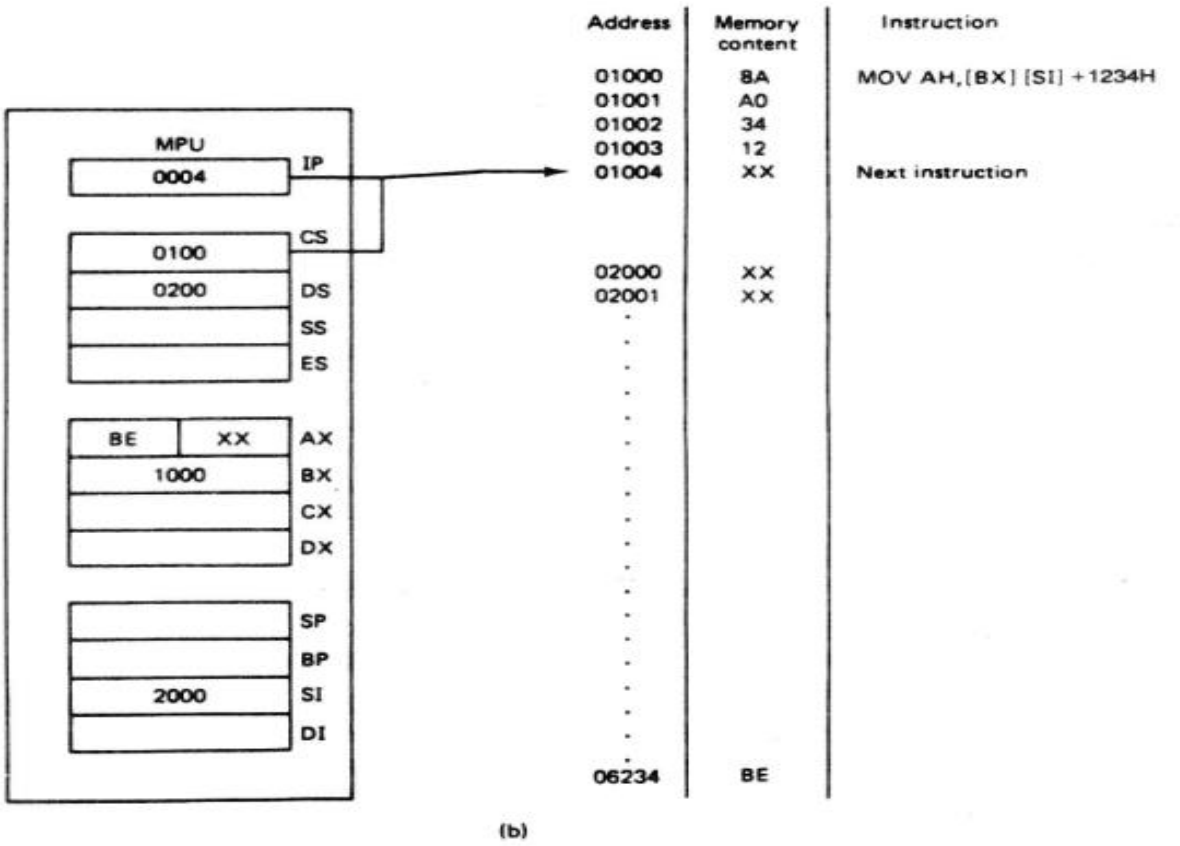


Fig-7-(a) before fetching and execution (b) after execution

Note that:- 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.**

Note that:- The displacement could be **8 bits** or **16 bits**