Microprocessor Instructions

Addressing Modes

1. Introduction to assembly language programming

- Program is a sequence of commands used to tell a microcomputer what to do.
- Each command in a program is an instruction.
- Programs must always be coded in machine language before they can be executed by the microprocessor.
- A program written in machine language is often referred to as *machine code*.
- Machine code is encoded using **0**s and **1**s.
- A single machine language instruction can take up one or more bytes of code.
- In assembly language, each instruction is described with alphanumeric symbols instead of with **0**s and **1**s.
- Instruction set includes :

1- Data transfer instructions.

2- Arithmetic instructions.

3- Logic instructions.

4- String manipulation instructions.

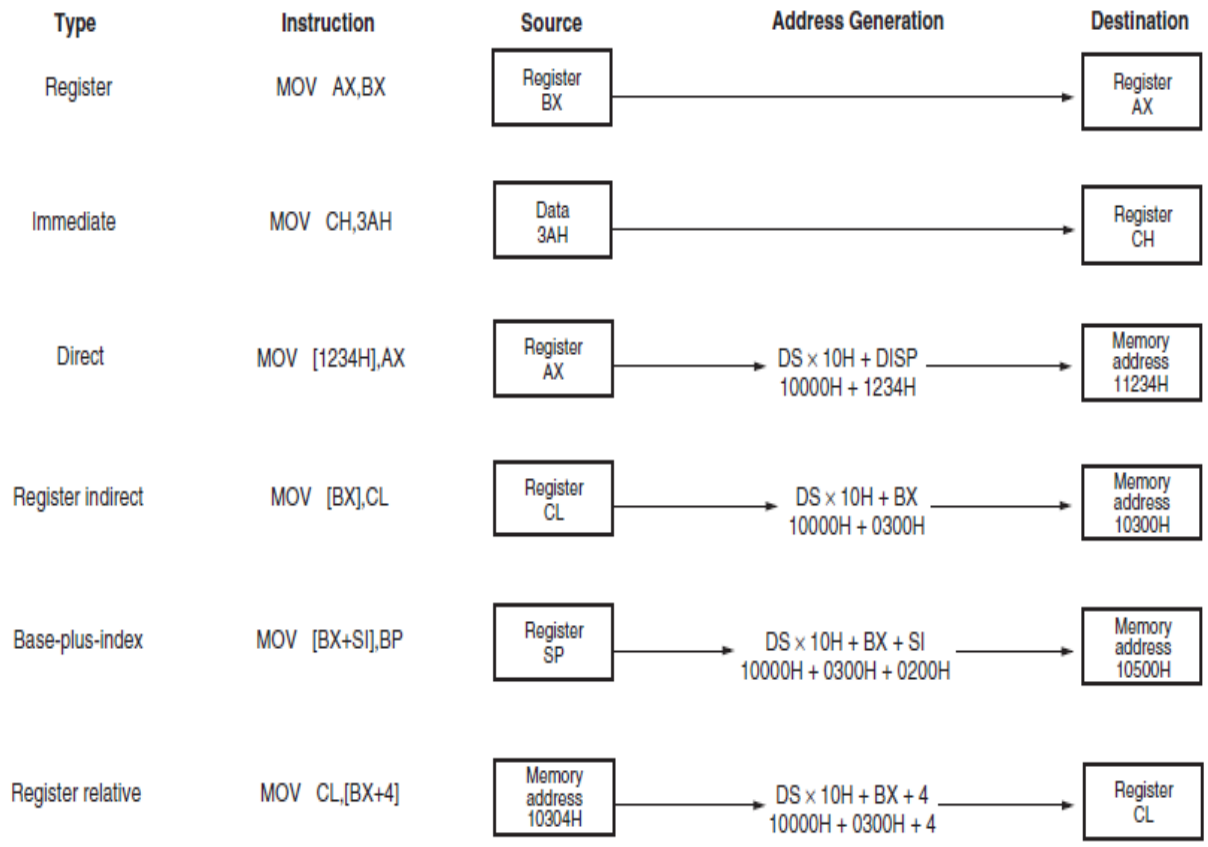
5- Control transfer instructions.

6- Processor control instructions.

1- Data Movement Instruction

A) MOV:

This instruction is used to transfer a byte or a word of data from source operand to destination operand.



Notes: ARRAY = 1000H, and DS = 1000H

8086–data-addressing modes.

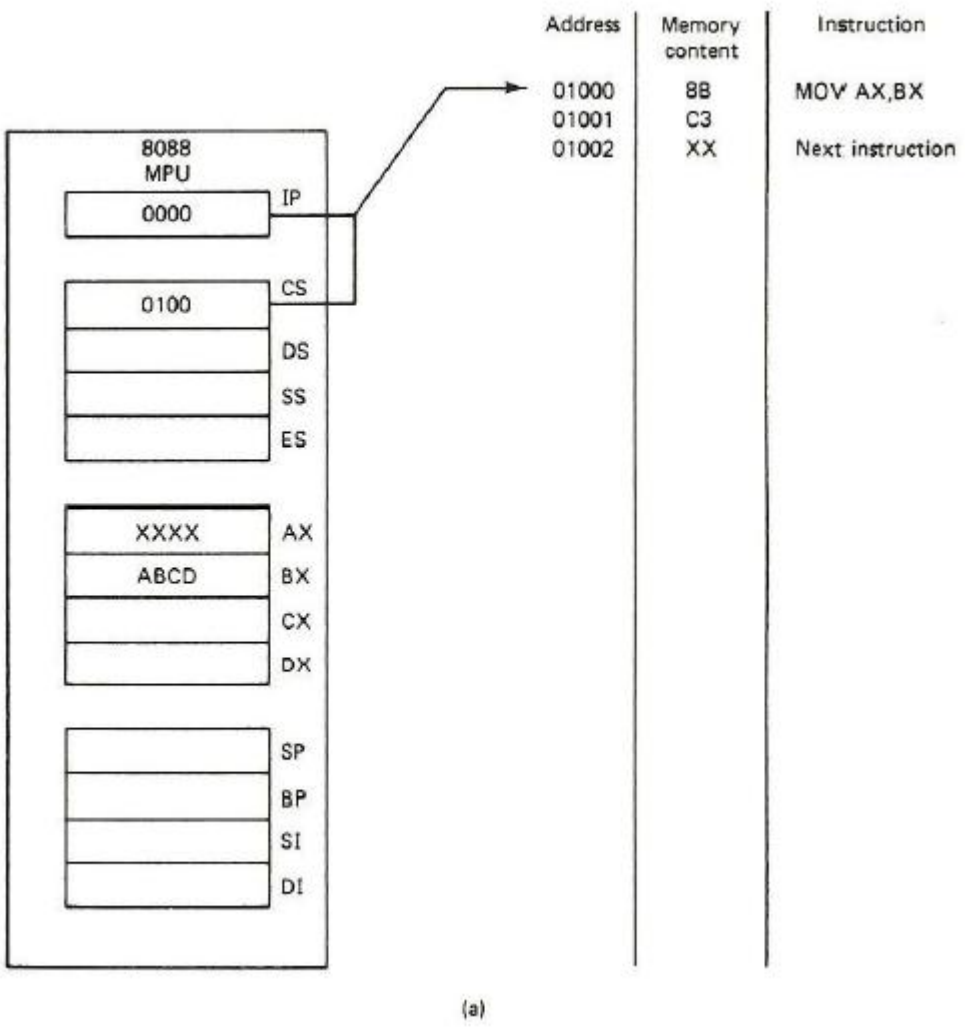
The data-addressing modes are as follows:

- **Register addressing** Register addressing transfers a copy of a byte or word from the source register or contents of a memory location to the destination register or memory location.

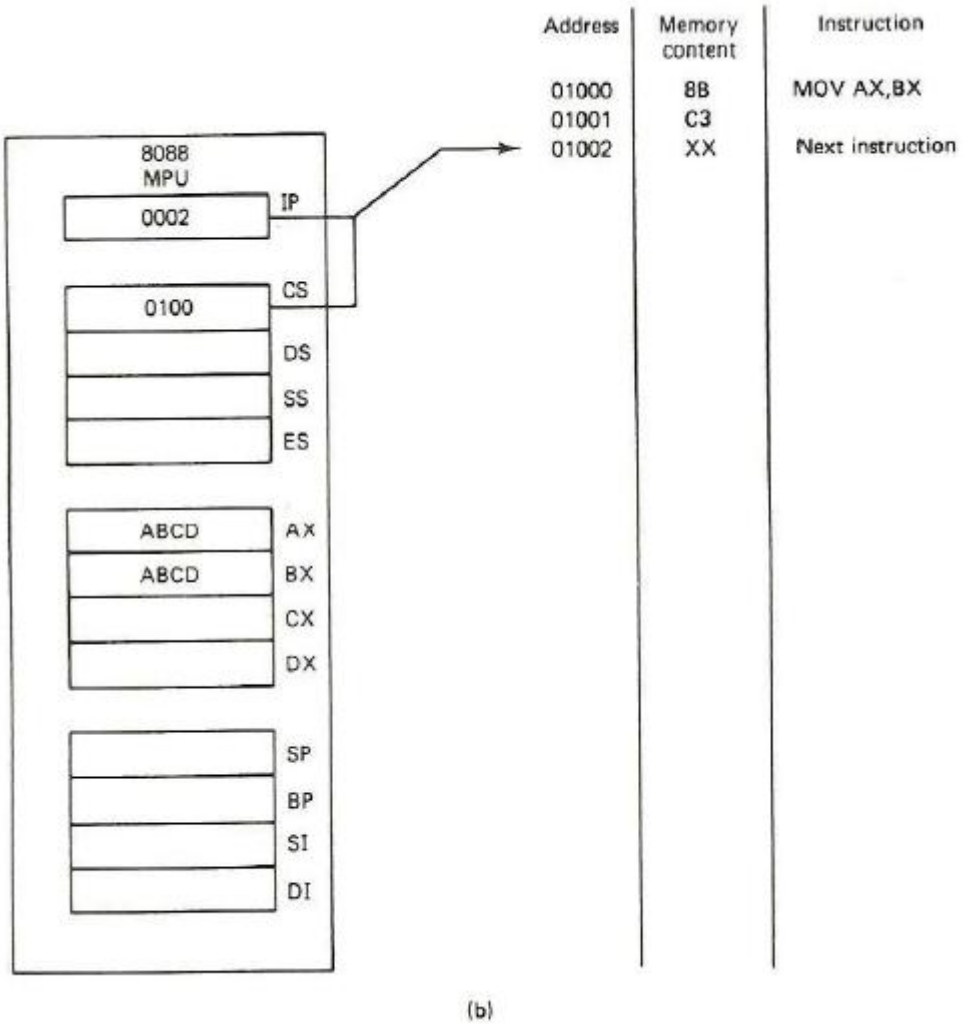
  Example:

  The MOV AX, BX instruction copies the word-sized contents of register BX into register AX. Fig below shows the memory and registers before and after the execution of instruction:

  MOV AX, BX



(a) Before fetching and execution

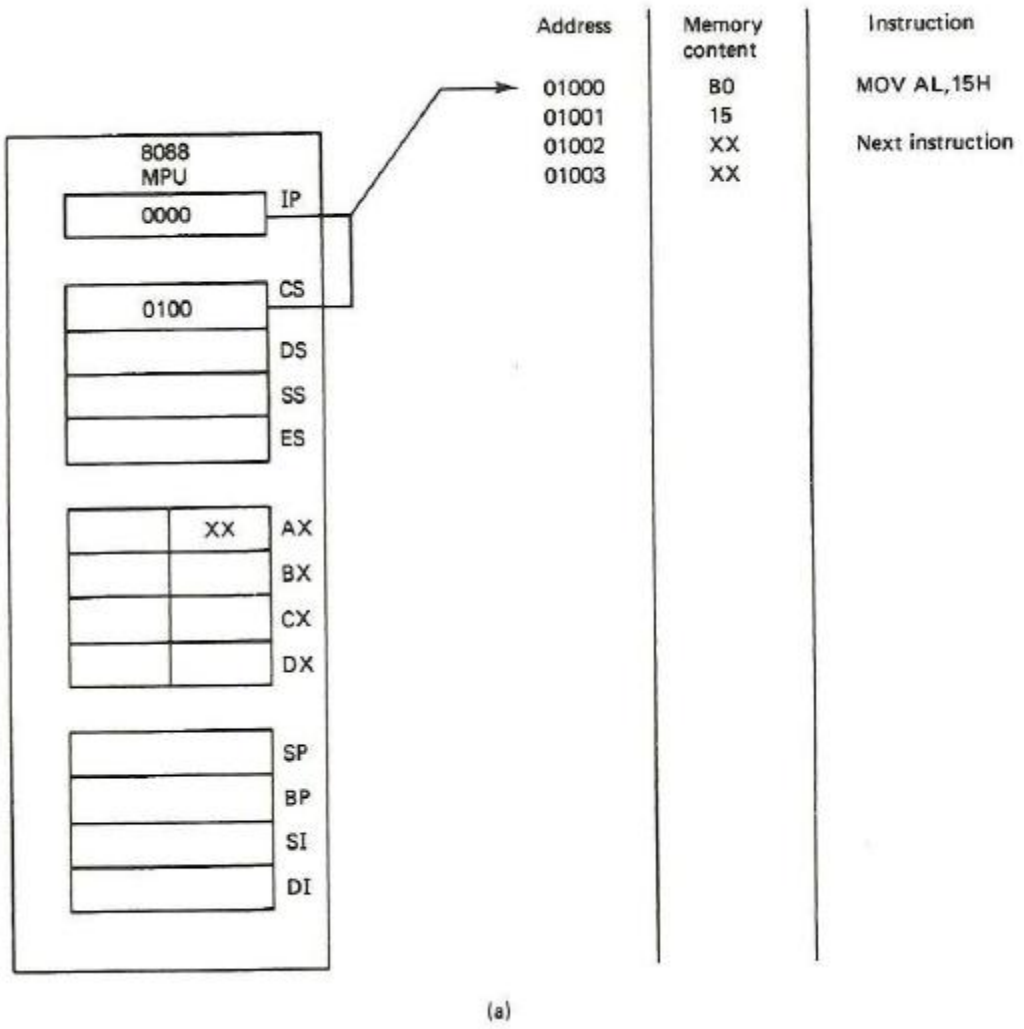


(b) After execution

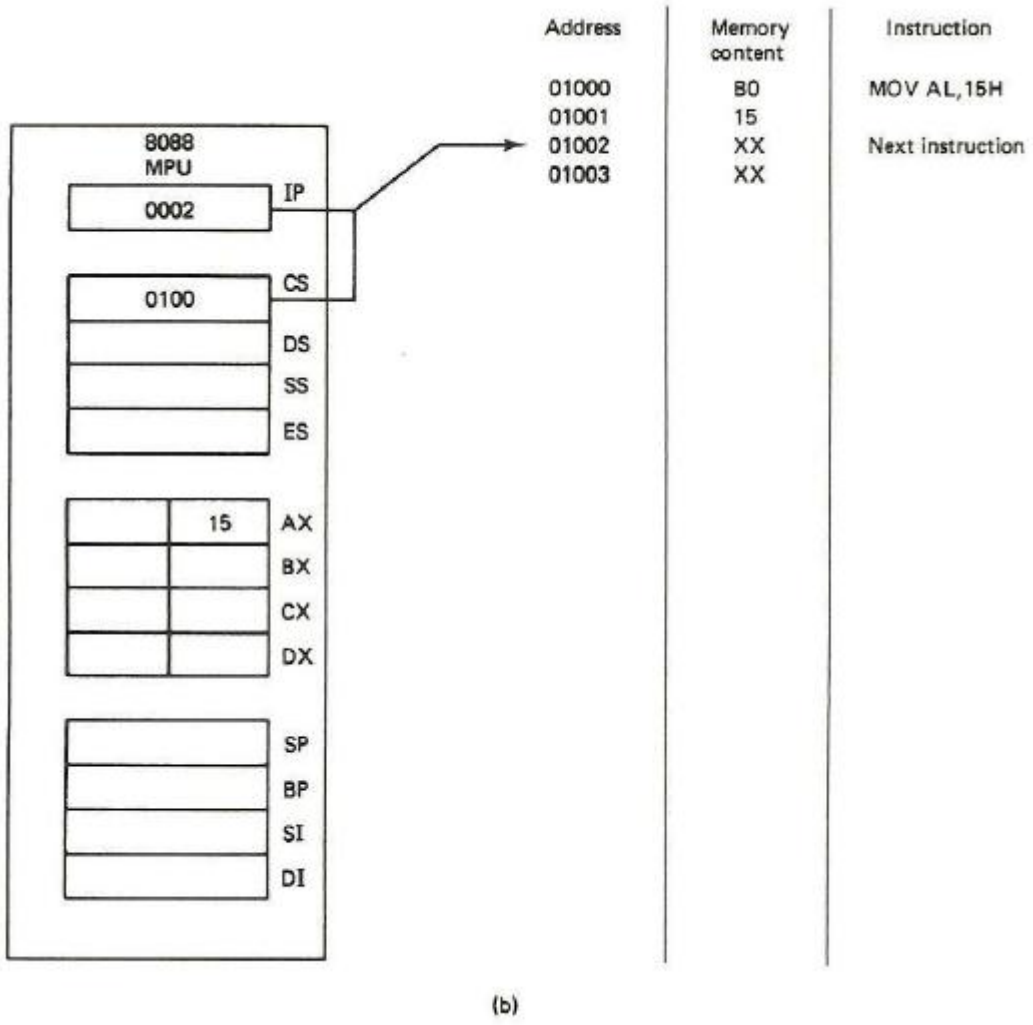
- **Immediate addressing** immediate addressing transfers the source, an immediate byte, word, double word, or quad word of data, into the destination register or memory location.

Example:

The MOV AL, 15H instruction copies a byte-sized 15H into register AL.



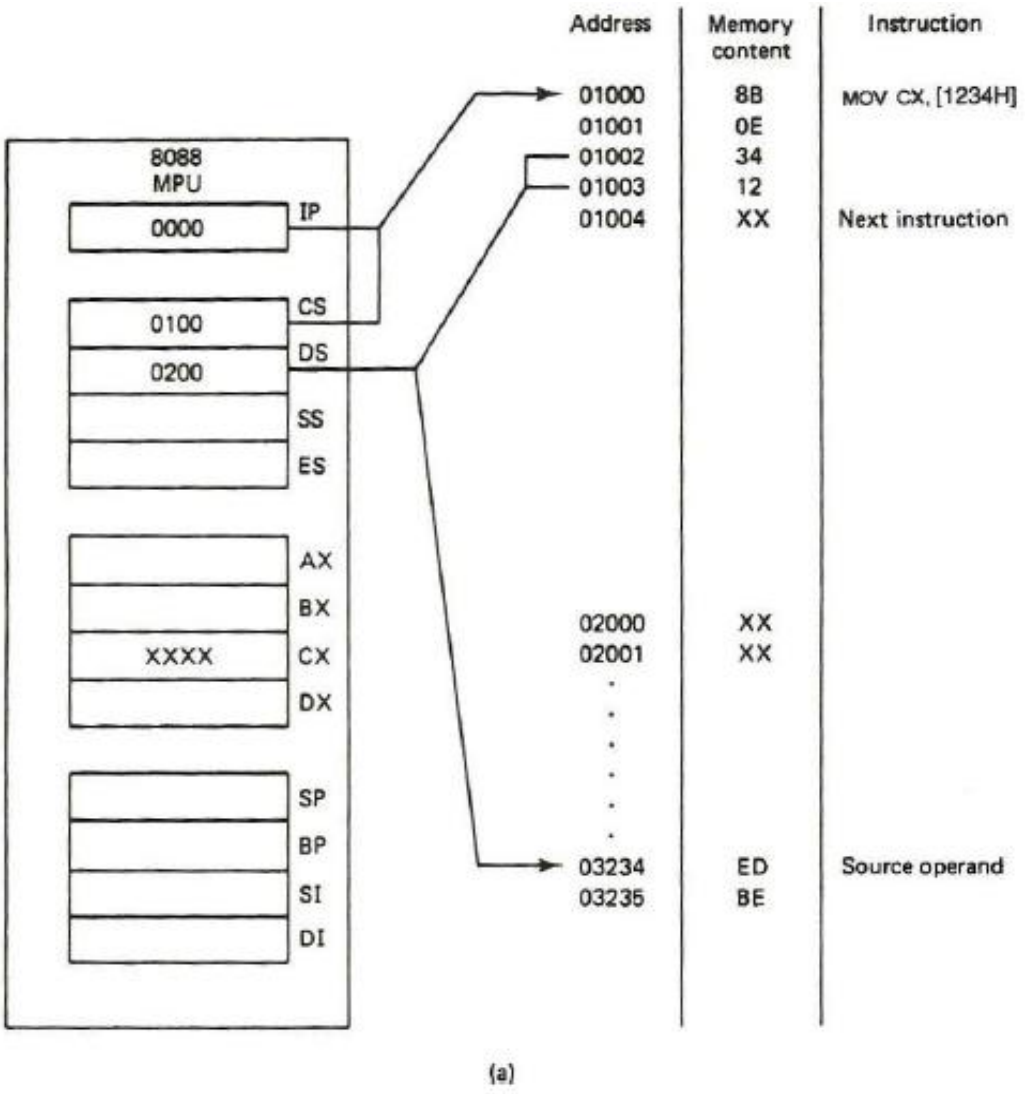
(a) Before fetching and execution



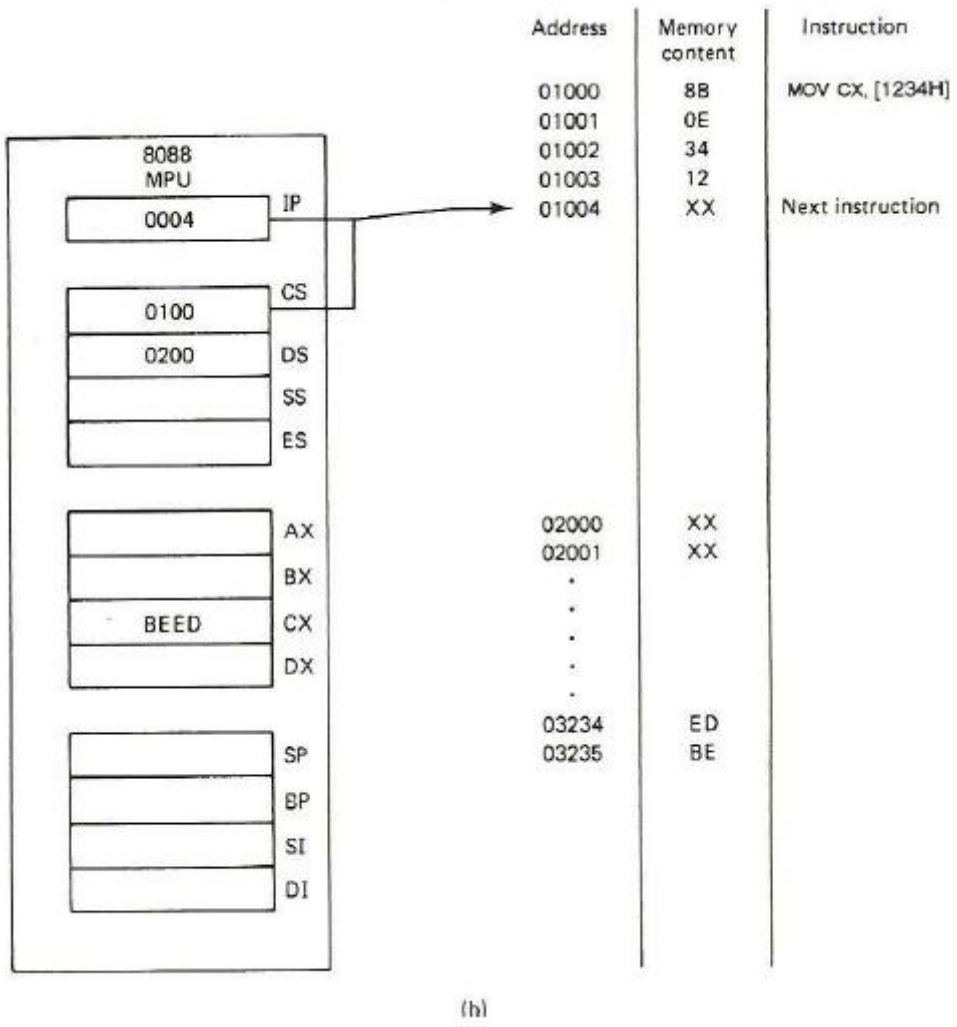
(b) After execution

- **Direct addressing** direct addressing moves a byte or word between a memory location and a register. The instruction set does not support a memory-to memory transfer.

MOV CX, [1234H]



(a) Before fetching and execution

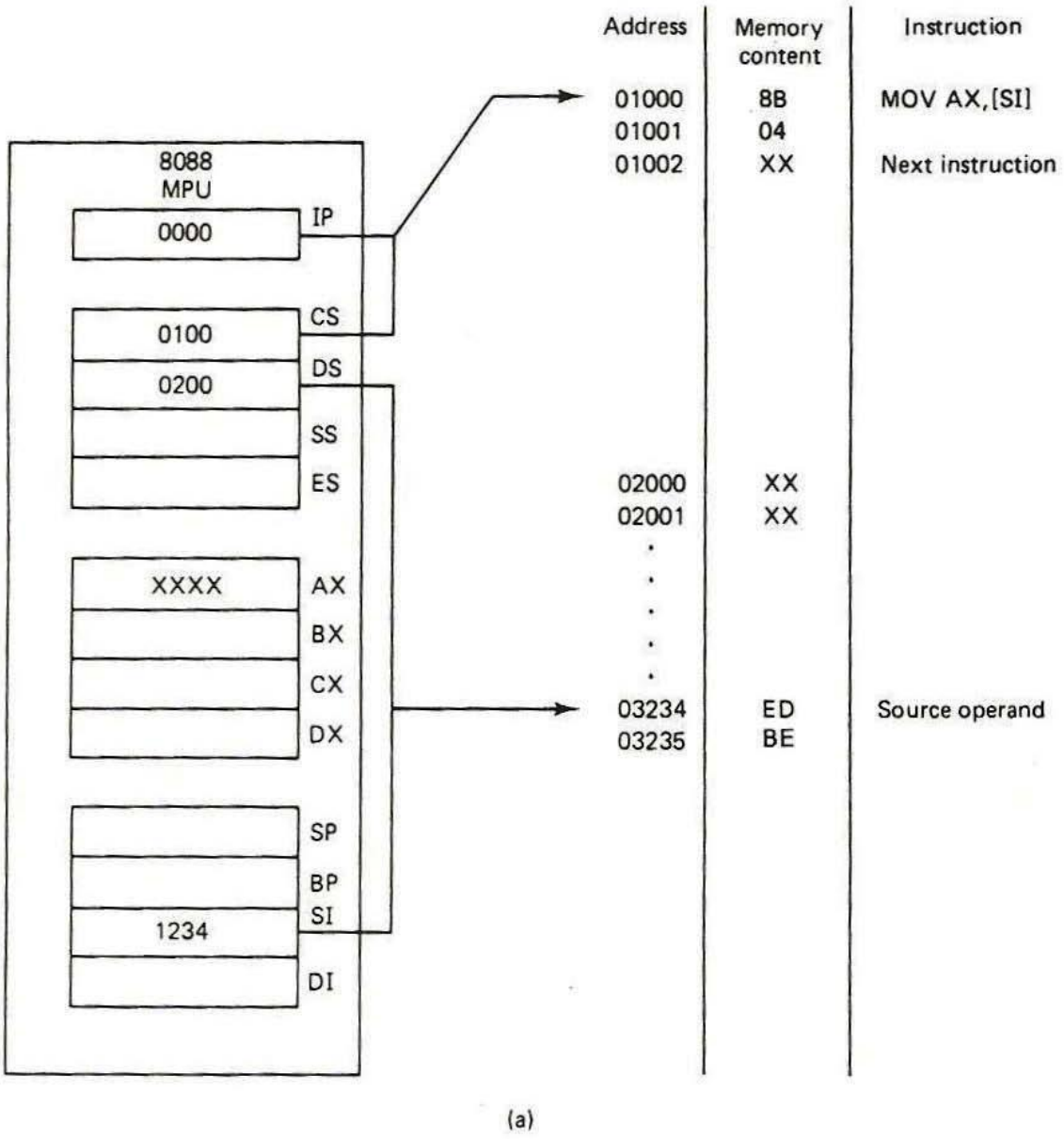


(b) After execution

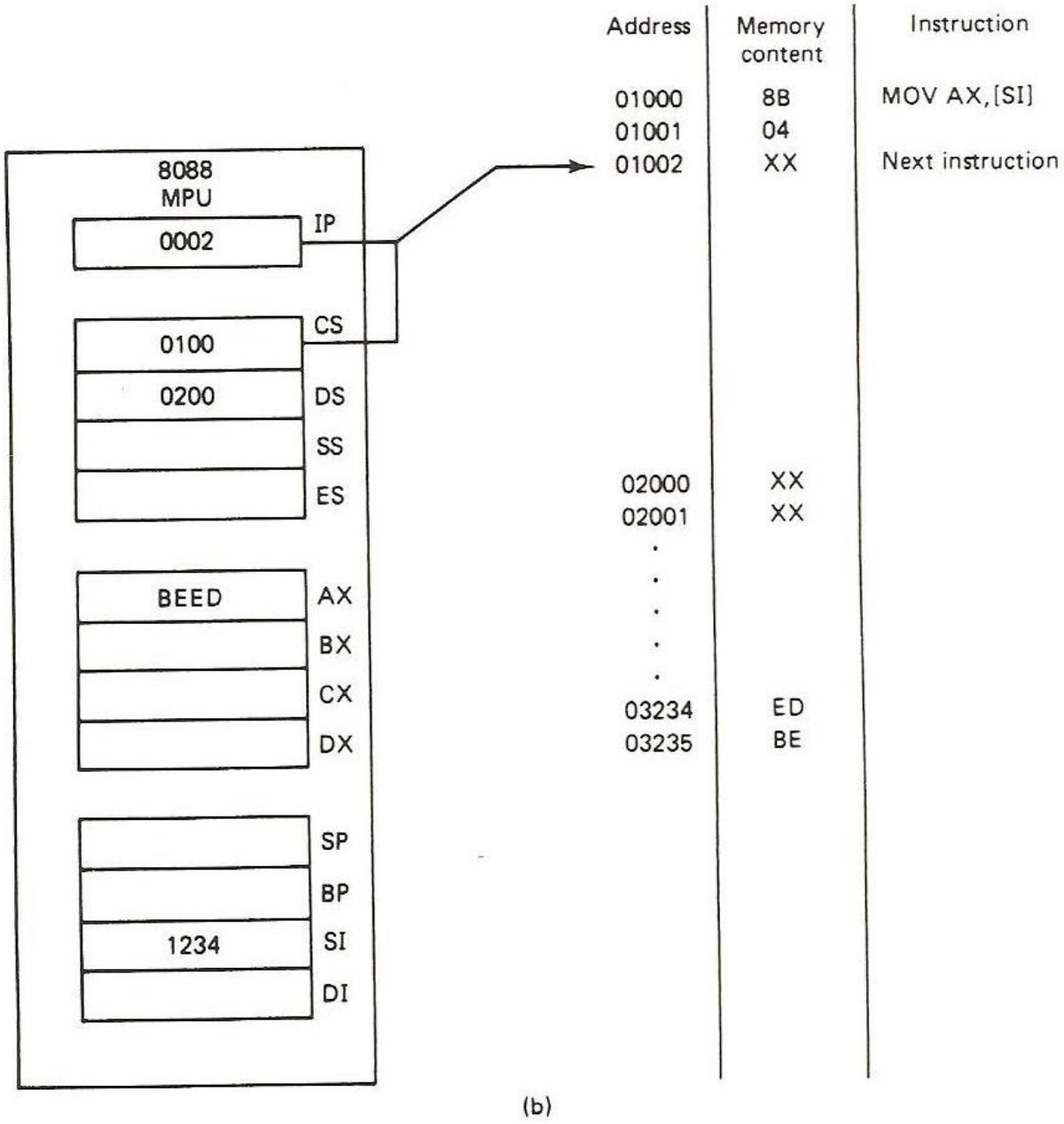
- **Register indirect addressing** Register indirect addressing transfers a byte or word between a register and a memory location addressed by an index or base register. The index and base registers are BP, BX, DI, and S1.

Example:

The MOV AX, [SI] instruction copies the word-sized data from the data segment offset address indexed by SI into register AX.



(a) Before fetching and execution

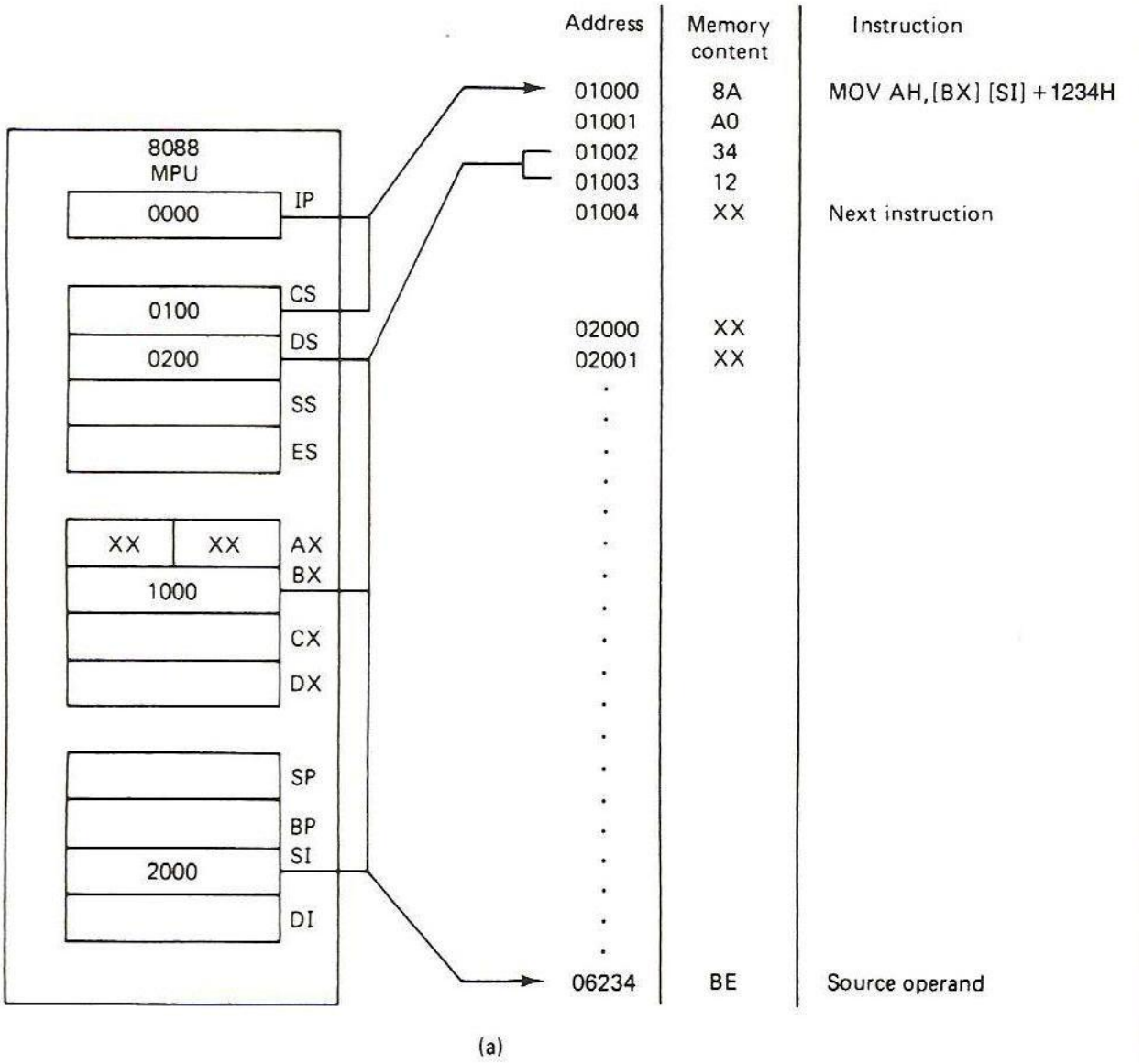


(b) After execution

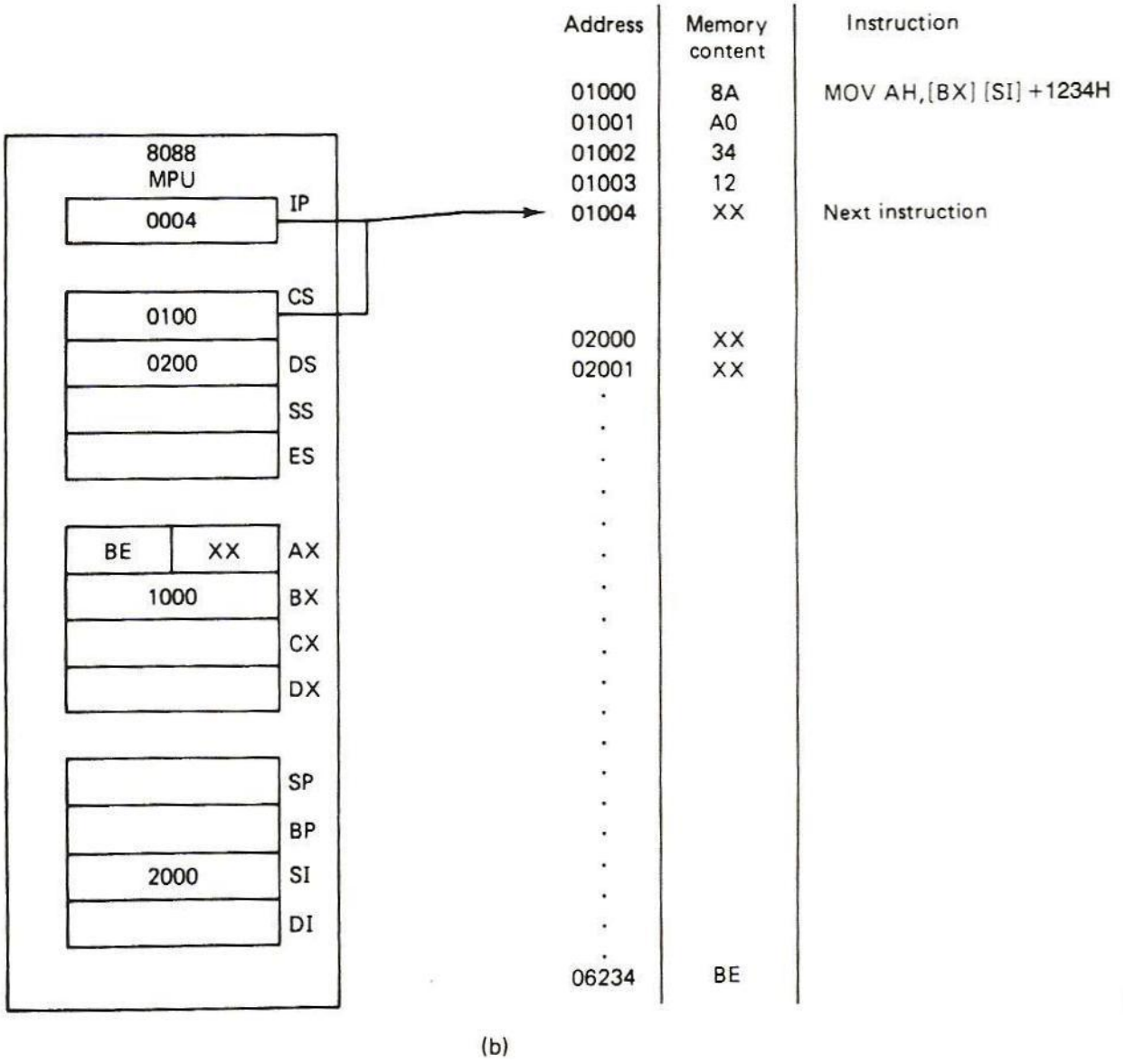
- **Base-plus-index addressing** Base-plus-index addressing transfers a byte or word between a register and the memory location addressed by a base register (BP or BX) plus an index register (DI or SI).

Example:

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



(a)Before fetching and execution



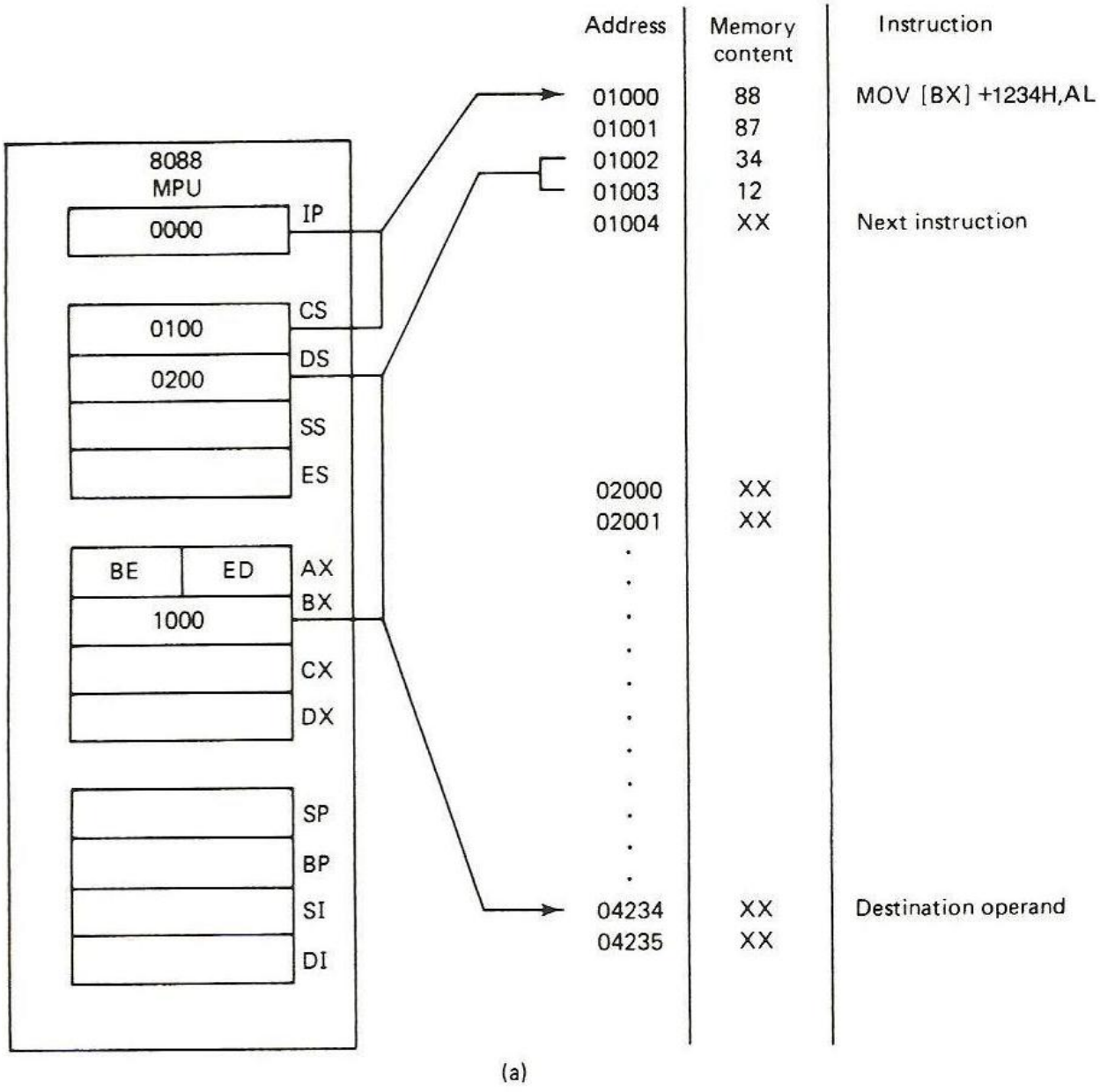
(b)After executing

Register relative addressing Register relative addressing moves a byte or word between a register and the memory location addressed by an index or base register plus a displacement.

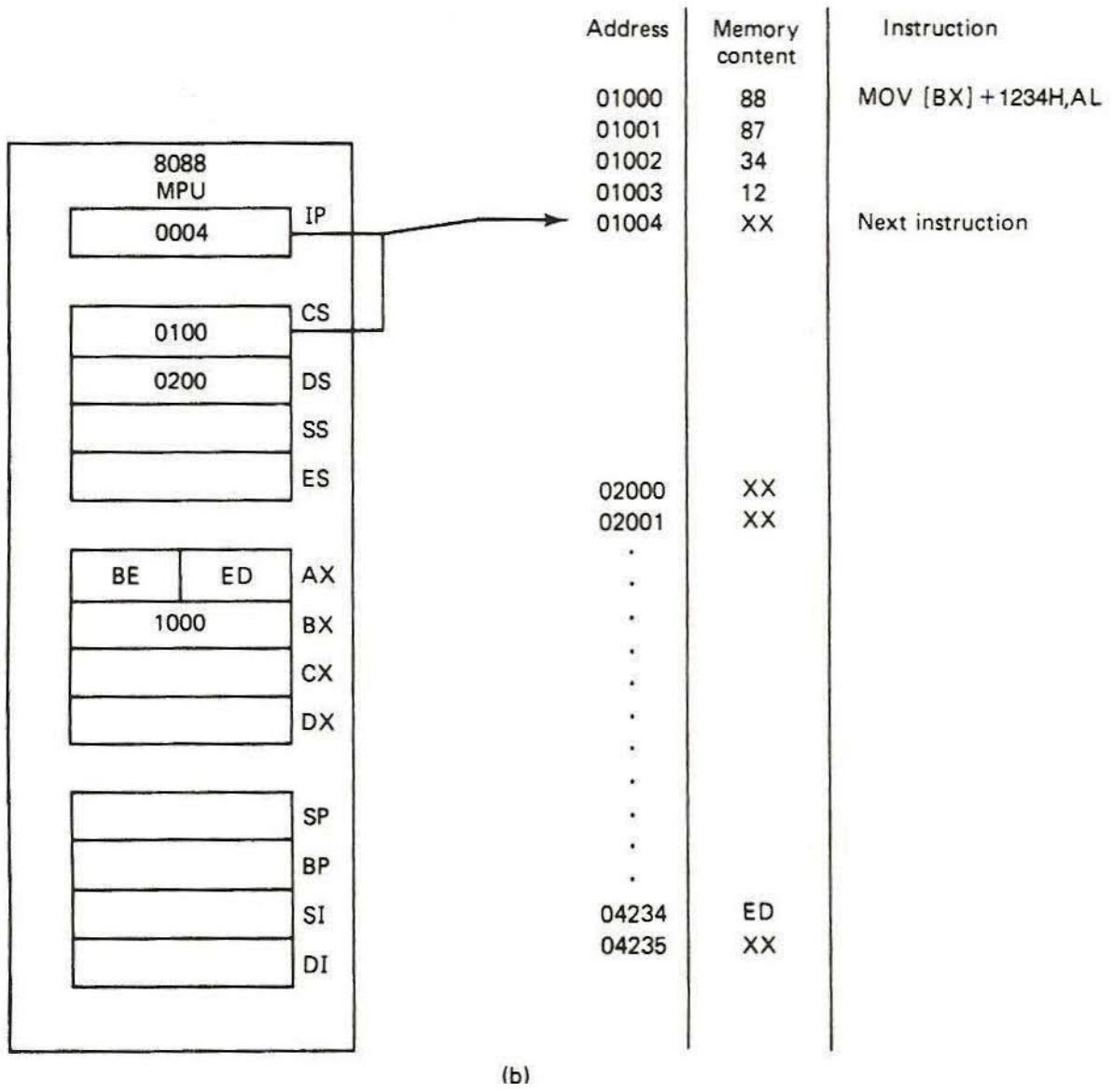
Example:

MOV [BX] +1234H, AL or MOV AX, ARRAY[BX].

The second instruction loads AX from the data segment memory location in ARRAY plus the contents of BX.)



(a) Before fetching and execution



(b) After executing

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**

- **Base relative-plus- index addressing** Base relative-plus-index addressing transfers a byte or word between a register and the memory location addressed by a base and an index register plus a displacement.

Example:

MOV AX, ARRAY[BX+DI] or MOV AX, [BX+DI+4]. These instructions load AX from a data segment memory location. The first instruction uses an address formed by adding ARRAY, BX, and DI and the second by adding BX, DI, and 4.

Register Addressing

Register addressing is the most common form of data addressing and, once the register names are learned, is the easiest to apply.

Examples of 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 DS,CX	16 bits	Copies CX into DS
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

Another data-addressing mode is immediate addressing. The term *immediate* implies that the data immediately follow the hexadecimal opcode in the memory. Also note that immediate data are constant data, whereas the data transferred from a

register or memory location are variable data. Immediate addressing operates upon a byte or word of data.

Examples of immediate addressing using the MOV instruction.

Assembly Language	*Size*	*Operation*
MOV BL,44	8 bits	Copies 44 decimal (2CH) into BL
MOV AX,44H	16 bits	Copies 0044H into AX
MOV SI,0	16 bits	Copies 0000H into SI
MOV CH,100	8 bits	Copies 100 decimal (64H) into CH
MOV AL,'A'	8 bits	Copies ASCII A into AL
MOV AH,1	8 bits	Not allowed in 64-bit mode, but allowed in 32- or 16-bit modes
MOV AX,'AB'	16 bits	Copies ASCII BA* into AX
MOV CL,11001110B	8 bits	Copies 11001110 binary into CL

In summary the valid source and destination variations for MOV instruction

Destination	Source
Memory	Accumulator
Accumulator	Memory
Register	Register
Register	Memory
Memory	Register
Register	Immediate
Memory	Immediate
Seg- Register	Register 16
Seg- Register	Memory 16
Register 16	Seg- Register
Memory	Seg- Register