Addressing modes

24 August 2020 10:11

The **control transfer instructions**, on the other hand, transfer control to some predefined address or the address somehow specified in the instruction, after their execution. For example, INT, CALL, RET and JUMP instructions fall under this category.

The addressing modes for sequential control transfer instructions are:

1. **Immediate:** In this type of addressing, immediate data is a part of instruction and appears in the form of successive byte or bytes.

Ex: MOV AX, 0005H

In the above example, 0005H is the immediate data. The immediate data may be 8-bit or 16-bit in size.

2. **Direct:** In the direct addressing mode a 16-bit memory address (offset) is directly specified in the instruction as a part of it.

Ex: MOV AX, [5000H]

Here, data resides in a memory location in the data segment, whose effective address may be completed using 5000H as the offset address and content of DS as segment address. The effective address here, is 10H * DS + 5000H.

3. **Register:** In register addressing mode, the data is stored in a register and is referred using the particular register. All the registers, except IP, may be used in this mode.

Ex: MOV BX, AX

4. **Register Indirect:** Sometimes, the address of the memory location, which contains data or operand, is determined in an indirect way, using the offset register. This mode of addressing is known as register indirect mode. In this addressing mode, the offset address of data is in either BX or SI or DI register. The default segment is either DS or ES. The data is supposed to be available at the address pointed to by the content of any of the above registers in the default data segment.

Ex: MOV AX, [BX]

Here, data is present in a memory location in DS whose offset address is in BX. The effective address of the data is given as 10H * DS+[BX].

5. **Indexed:** In this addressing mode, offset of the operand is stored in one of the index registers. DS and ES are the default segments for index registers, SI and DI respectively. This is a special case of register indirect addressing mode.

Ex: MOV AX, [SI]

Here, data is available at an offset address stored in SI in DS. The effective address, in this case, is computed as 10*DS+[SI].

6. **Register Relative:** In this addressing mode, the data is available at an effective address formed by adding an 8-bit or 16-bit displacement with the content of any one of the registers BX, BP, SI and DI in the default (either DS or ES) segment.

Ex: MOV AX, 50H[BX]

Here, the effective address is given as 10H *DS+50H+[BX]

7. **Based Indexed:** The effective address of data is formed, in this addressing mode, by adding content of a base register (any one of BX or BP) to the content of an index register (any one of SI or DI). The default segment register may be ES or DS.

Ex: MOV AX, [BX][SI]

Here, BX is the base register and SI is the index register the effective address is computed as 10H * DS + [BX] + [SI].

8. **Relative Based Indexed:** The effective address is formed by adding an 8 or 16-bit displacement with the sum of the contents of any one of the base register (BX or BP) and any one of the index register, in a default segment.

Ex: MOV AX, 50H [BX] [SI]

Here, 50H is an immediate displacement, BX is base register and SI is an index register the effective address of data is computed as

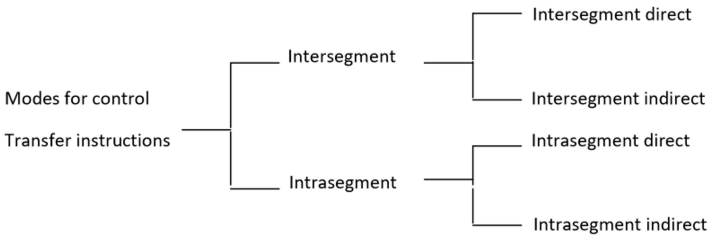
10H * DS + [BX] + [SI] + 50H

For control transfer instructions, the addressing modes depend upon whether the destination is within the same segment or different one. It also depends upon the method of passing the destination address to the processor.

Basically, there are two addressing modes for the control transfer instructions, **intersegment** addressing and **intrasegment** addressing modes.

If the location to which the control is to be transferred lies in a different segment other than the current one, the mode is called intersegment mode.

If the destination location lies in the same segment, the mode is called intrasegment mode.



Addressing modes for Control Transfer Instructions

9. **Intrasegment Direct Mode:** In this mode, the address to which the control is to be transferred lies in the same segment in which the control transfer instruction lies and appears directly in the instruction as an immediate displacement value. In this addressing mode, the displacement is computed relative to the content of the instruction pointer IP.

The effective address to which the control will be transferred is given by the sum of 8 or 16-bit displacement and current content of IP. In the case of jump instruction, if the signed displacement (d) is of 8-bits (i.e –128<d<+128) we term it as short jump and if it is of 16-bits (i.e-32, 768<d<+32,768) it is termed as long jump.

10. **Intrasegment Indirect Mode:** In this mode, the displacement to which the control is to be transferred, is in the same segment in which the control transfer instruction lies, but it is passed to the instruction indirectly. Here, the branch address is found as the content of a register or a memory location. This addressing mode may be used in unconditional branch instructions.

11. **Intersegment Direct:** In this mode, the address to which the control is to be transferred is in a different segment. This addressing mode provides a means of branching from one code segment to another code segment. Here, the CS and IP of the destination address are specified directly in the instruction.

12. **Intersegment Indirect:** In this mode, the address to which the control is to be transferred lies in a different segment and it is passed to the instruction indirectly, i.e contents of a memory block containing four bytes, i.e IP (LSB), IP(MSB), CS(LSB) and CS (MSB) sequentially. The starting address of the memory block may be referred using any of the addressing modes, except immediate mode.

The 8086 Addressing Mode

When the 8086 executes an instruction, it performs the specified function on data. Thee 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 1(a, b).

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 15_H 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 2(a, b).

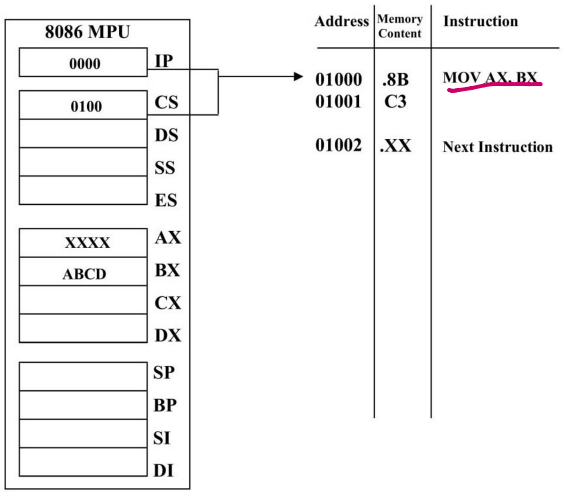


Figure 1(a): Register addressing mode before execution.

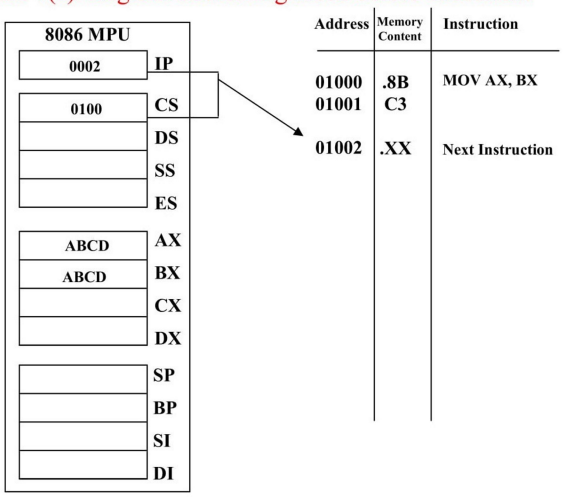


Figure 1(b): Register addressing mode after execution.

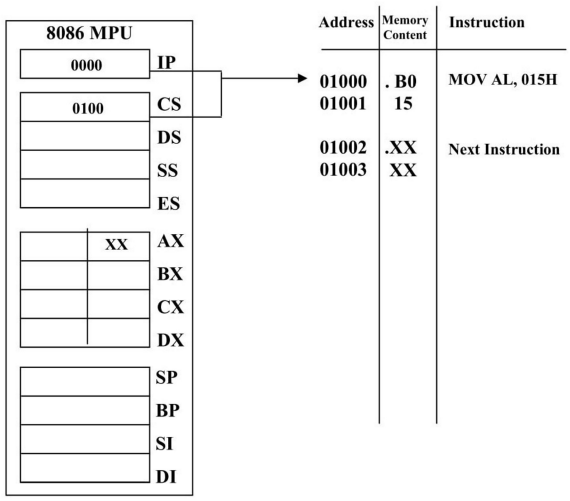


Figure 2 (a): Immediate addressing mode before execution.

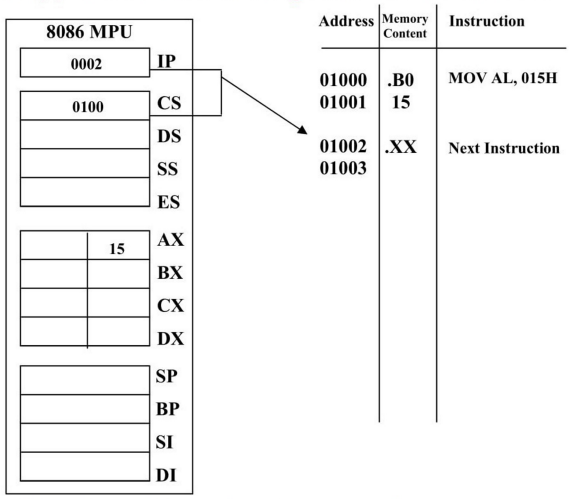


Figure 2 (b): Immediate addressing mode after execution.

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 3(a, b). Notice that the value assigned to constant BETA is 1234_H.

$$PA = 02000_H + 1234_H$$
$$= 03234_H$$

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 4 (a, b). SI contains 1234_H and DS contains 0200_H.

$$PA = 02000_H + 1234_H$$
$$= 03234_H$$

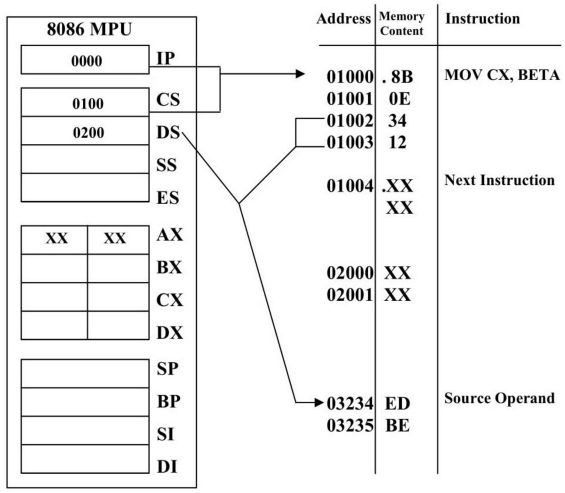


Figure 3 (a): Direct Addressing mode before execution.

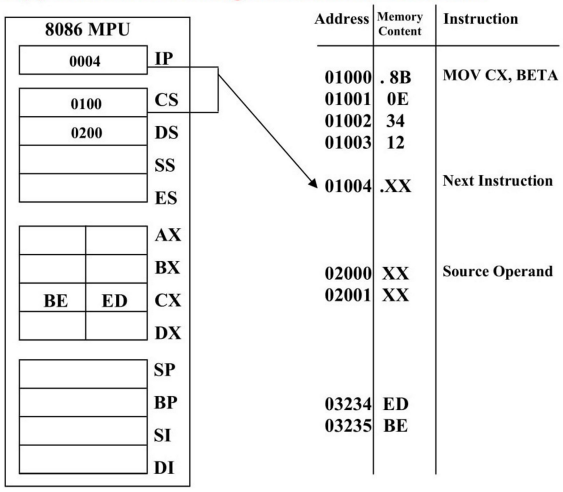


Figure 3 (b): Direct Addressing mode after execution.

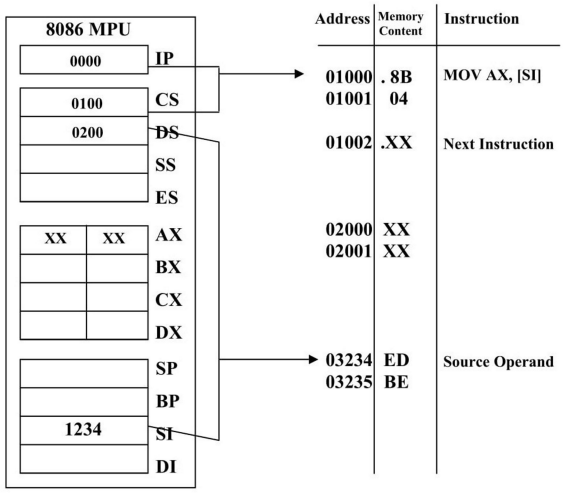


Figure 4 (a): Register Indirect Addressing before execution.

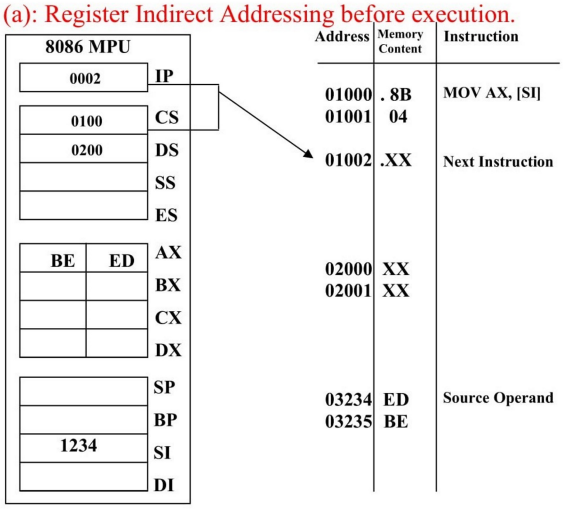


Figure 4 (b): Register Indirect Addressing mode after execution.

5. Based Addressing Mode

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 5(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

$$PA = 02000_H + 1000_H + 1234_H$$
$$= 04234_H$$

6. Indexed Addressing Mode

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 6 (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.

$$PA = 02000_H + 2000_H + 1234_H$$
$$= 05234_H$$

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

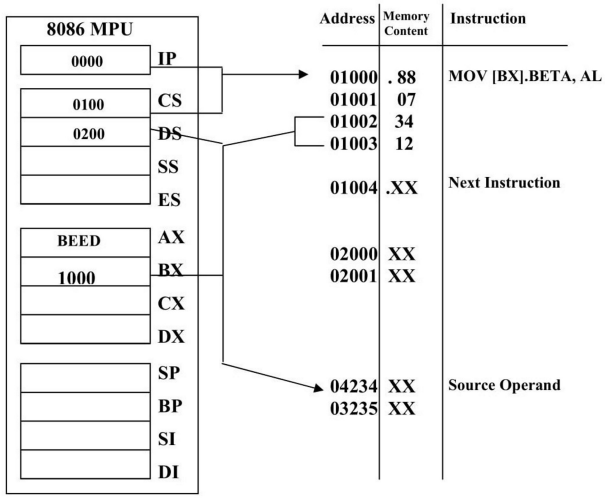


Figure 5 (a): Based Addressing before execution.

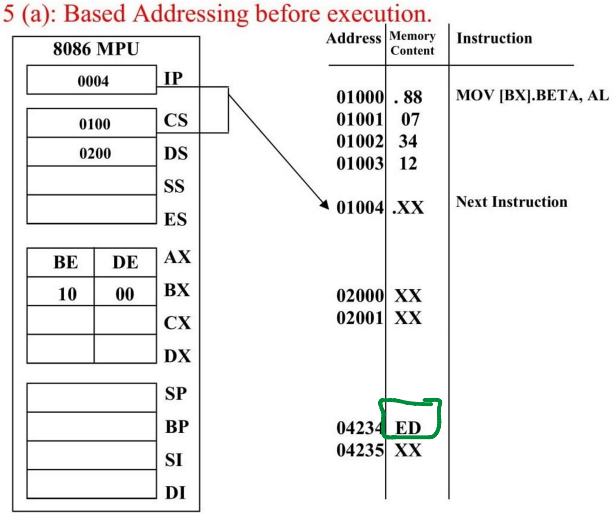


Figure 5 (b): Based Addressing mode after execution.

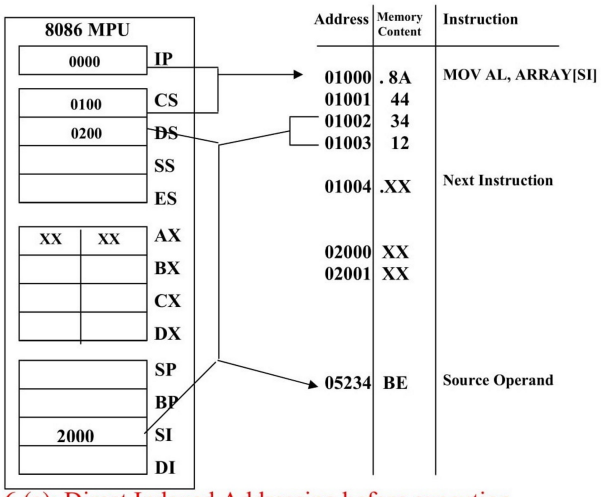


Figure 6 (a): Direct Indexed Addressing before execution.

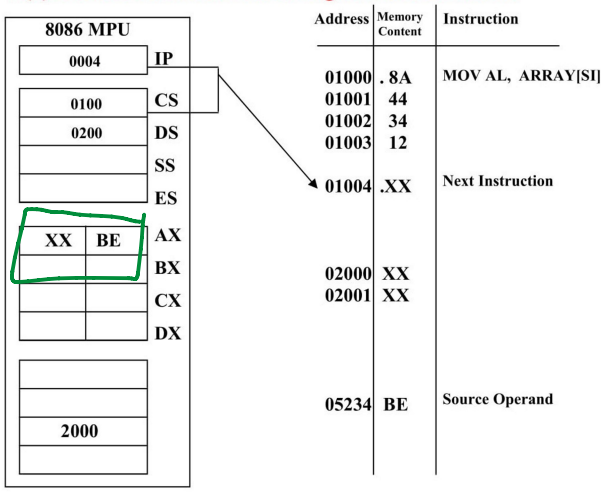


Figure 6 (b): Direct Indexed Addressing mode after execution.

7. Based Indexed Addressing Mode

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 7 (a,b). The address of the source operand is calculated as

$$PA = 02000_H + 1000_H + 1234_H + 2000_H$$
$$= 06234_H$$

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

8. String Addressing Mode

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.

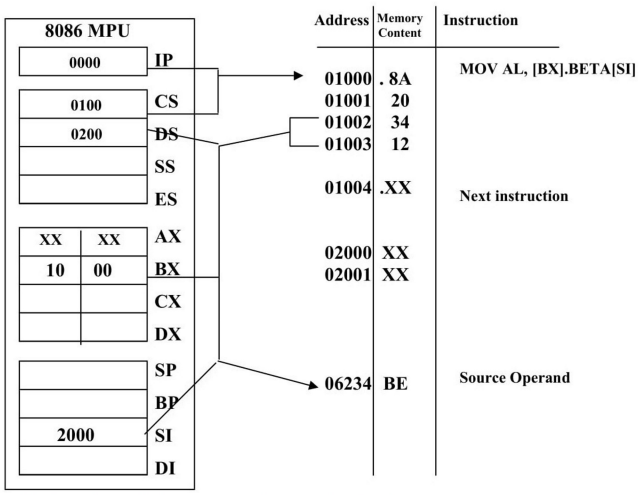


Figure 7 (a): Based Indexed Addressing before execution.

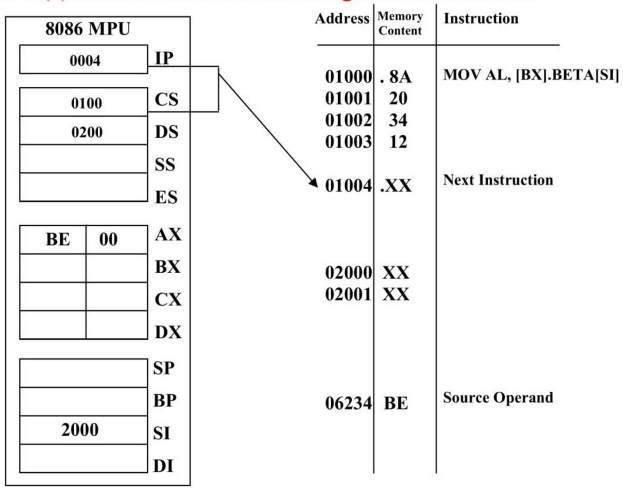


Figure 7 (b): Based Indexed Addressing mode after execution.

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, 15_H

This stands for input the data from the byte wide input port at address 15_H of the I/O address space to register AL.

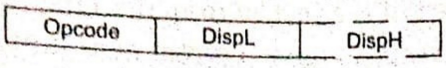
Next, let us consider another example. Using **Indirect port addressing** for the source operand in an IN instruction, we get:

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 1234_H the contents of the port at this I/O address are loaded into AL.

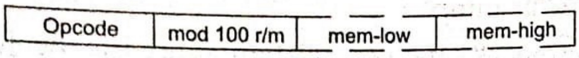
JMP = JMP

Within segment or group, IP relative—near and short



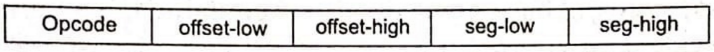
Opcode	Clocks	Operation
E9	15	IP ← IP + Disp16
EB	15	IP ← IP + Disp8 (Disp8 sign-extended)

Within segment or group, Indirect



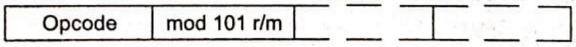
Opcode	Clocks	Operation
FF	11	IP ← Reg16
FF	18 + EA	IP ← Mem16

Inter-segment or group, Direct



Opcode	Clocks	Operation
EA	15	CS ← sefbase IP ← offset

Inter-segment or group, Indirect



Opcode	Clocks	Operation
FF	24 + EA	CS ← segbase IP ← offset

Fig. 4.7 *8086 Unconditional Jump instructions. (Intel Corporation)*