Memory Addressing and Instruction Formats

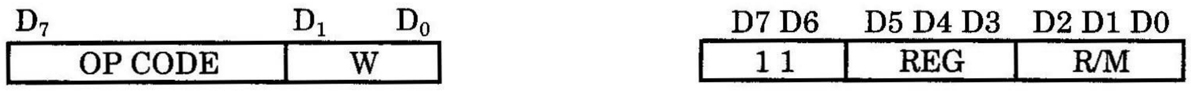
A machine language instruction format has one or more number of fields associated with it. The first field is called as operation code field or opcode field, which indicates the type of the operation to be performed by the CPU. The instruction format also contains other fields known as operand fields. The CPU executes the instruction using the information which reside in these fields.

There are six general formats of instructions in 8086 instruction set. The length of an instruction may vary from one byte to six bytes. The instruction formats are described as follows:

1. One byte Instruction: This format is only one byte long and may have the implied data or register operands. The least significant 3-bits of the opcode are used for specifying the register operand, if any.Otherwise, all the 8-bits form an opcode and the operands are implied.

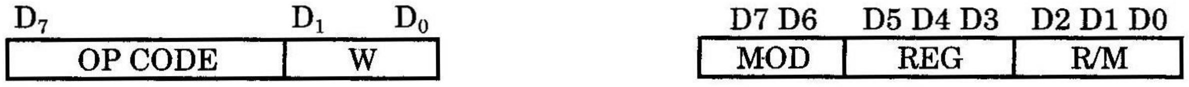
2. Register to Register: This format is 2 bytes long. The first byte of the code specifies the operation code and width of the operand specified by w bit. The second byte of the code shows the register operands and R/M field, as shown below.

The register represented by the REG field is one of the operands. The R/M field specifies another



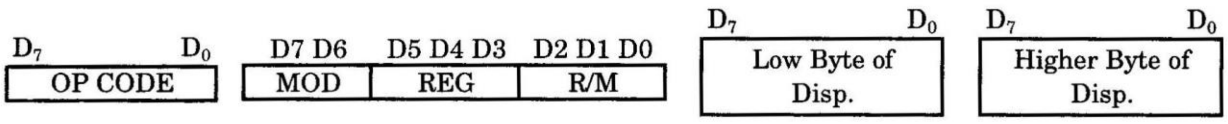
register or memory location, i.e. the other operand.

3. Register to/from Memory with no Displacement: This format is also 2 bytes long and similar to the register-to-register format except for the MOD field as shown.

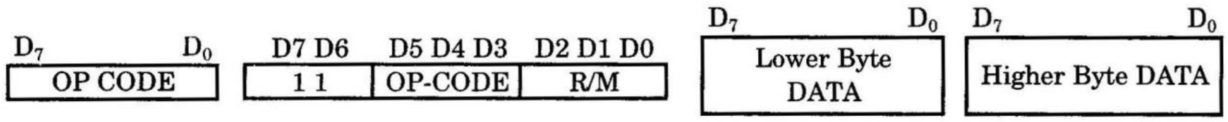


The MOD field shows the mode of addressing. The MOD, R/M, REG and the W fields are decided in Table 3.2

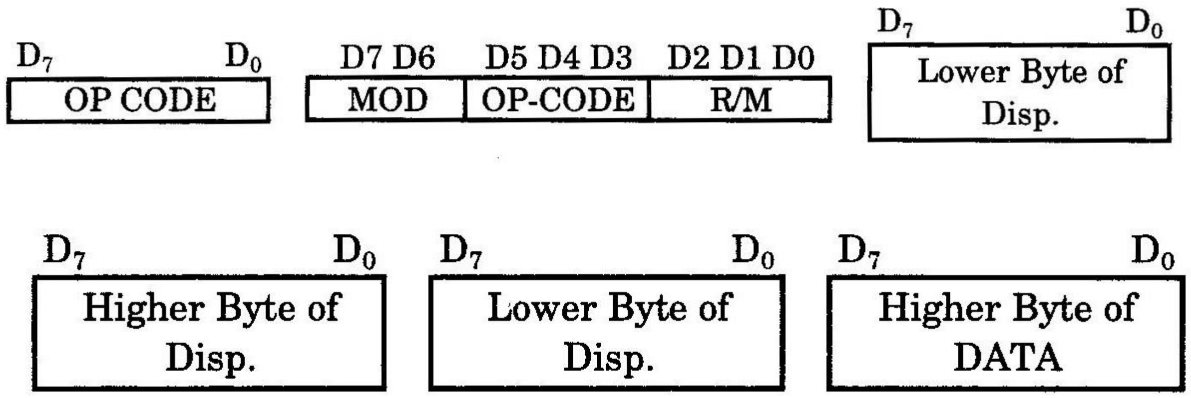
4. Register to/from Memory with Displacement: This type of instruction format contains one or two additional bytes for displacement along with 2-byte the format of the register to/from memory without displacement. The format is as shown below.



5. Immediate Operand to Register: In this format, the first byte as well as the 3-bits from the second byte which are used for REG field in case of register to register format are used for opcode. It also contains one or two bytes of immediate data. The complete instruction format is as shown below.



6. Immediate Operand to Memory with 16-bit Displacement: This type of instruction format requires 5 or 6 bytes for coding. The first 2 bytes contain the information regarding OPCODE, MOD, and R/M fields. The remaining 4 bytes contain 2 bytes of displacement and 2 bytes of data as shown.



The opcode usually aooears in the first byte. but in a few instructions, a register destination is in the first byte and few other instructions may have their 3-bits of opcode in the second byte. The opcodes have the single bit indicators. Their definitions and significance’s are given as follows:
W-bit: This indicates whether the instruction is to operate over an 8-bit or 16-bit data/operands. If bit is 0,the operand is of8-bits and if W is I, the operand is of 16-bits.
D-bit: This is valid in case of double operand instructions. One of the operands must be a register specified by the REG field. The register specified by REG is source operand ifD=0, else, it is a destination operand.
S-bit: This bit is called as sign extension bit. The S bit is used along with W-bit to show the type of the operation. For, example

8-bit operation with 8-bit immediate operand is indicated by S = 0, W = 0;

16-bit operation with 16-bit immediate operand is indicated by S = 0, W = 1 and

16-bit operation with a sign extended immediate data is given by S = 1, W = 1

V-bit: This is used in case of shift and rotate instructions. This bit is set to 0, if shift count is 1 and is set to 1, if CL contains the shift count.

Z-bit: This bit is used by REP instruction to control the loop. If Z bit is equal to I, the instruction with REP prefix is executed until the zero flag matches the Z bit.

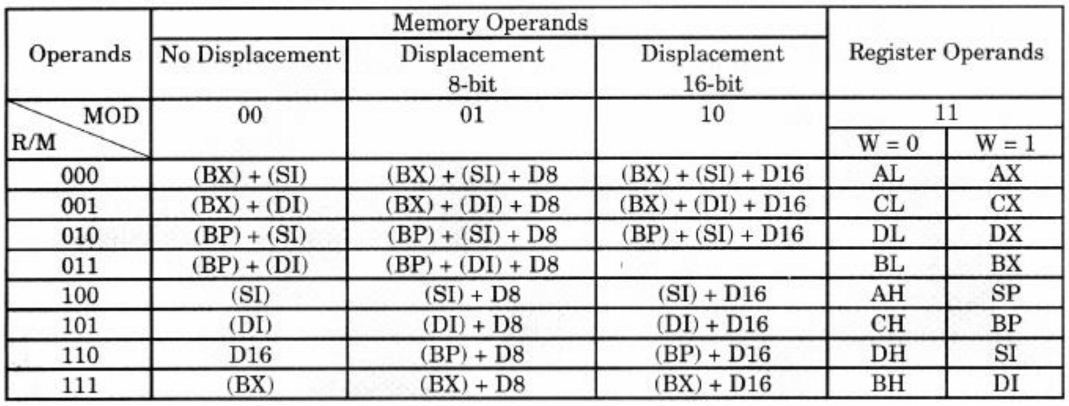
The REG code of the different registers (either as source or destination operands) in the opcode byte are assigned with binary codes. The segment registers are only 4 in number hence 2 binary bits will be sufficient to code them. The other registers are 8 in number, so at least 3-bits will be required for coding them. To allow the use of 16-bit registers as two 8-bit registers they are coded with W bit as shown in Table 3.1.

Table 3.1: Assignment of Codes with Different Registers

W bit	Register Address (code)	Registers	Segment 2 bit	Segment Registter
0	000	AL		
0	001	CL	00	ES
0	010	DL	01	CS
0	011	BL	10	SS
0	100	AH	11	DS
0	101	CH		
0	110	DH		
0	111	BH		
1	000	AX		
1	010	CX		
1	100	SP		
1	101	BP		
1	110	SI		
1	111	DI		

Please note that usually all the addressing modes have DS as the default data segment. However, the addressing modes using BP and SP have SS as the default segment register.
To find out the MOD and R/M fields of a particular instruction, one should first decide the addressing mode of the instruction. The addressing mode depends upon the operands and suggests how the effective address may be computed for locating the operand, if it lies in memory. The different addressing modes of the 8086 instructions are listed in Table 3.2. The R/M column and addressing mode row element specifies the R/M field, while the addressing mode column specifies the MOD field.

Table 3.2: Addressing Modes and the Corresponding MOD, REG and R/M Fields



Operands	Memory Operands			Register Operands	
	No Displacement	Displacement 8-bit	Displacement 16-bit		
MOD / R/M	00	01	10	11	
				W = 0	W = 1
000	(BX) + (SI)	(BX) + (SI) + D8	(BX) + (SI) + D16	AL	AX
001	(BX) + (DI)	(BX) + (DI) + D8	(BX) + (DI) + D16	CL	CX
010	(BP) + (SI)	(BP) + (SI) + D8	(BP) + (SI) + D16	DL	DX
011	(BP) + (DI)	(BP) + (DI) + D8		BL	BX
100	(SI)	(SI) + D8	(SI) + D16	AH	SP
101	(DI)	(DI) + D8	(DI) + D16	CH	BP
110	D16	(BP) + D8	(BP) + D16	DH	SI
111	(BX)	(BX) + D8	(BX) + D16	BH	DI

Note: 1. D8 & D16 represent 8 and 16 bit displacements respectively.

2. The default segment for the addressing modes using BP and SP is SS, For all other addressing modes the default segments are DS or ES.

When a data is to be referred as an operand, DS is the default data segment register. CS is the default code segment register for storing program codes (executable codes). SS is the default segment register for the stack data accesses and operations. ES is the default segment register for the destination data storage. All the segments available (defined in a particular program) can be read or written as data segments by newly defining the data segment as required. There is no physical difference in the memory structure or no physical separation between the segment areas. They may or may not overlap with each other.

Addressing Modes of 8086

Addressing mode indicates a way of locating data or operands. Depending upon the data types used in theinstruction and the memory addressing modes, any instruction may belong to one or more addressing modes, or some instruction may not belong to any of the addressing modes. Thus the addressing modes describe the types of operands and the way they are accessed for executing an instruction. Here, we will present the addressing modes of the instructions depending upon their types. According to the flow of instruction execution, the instructions may be categorised as (i) Sequential control flow instructions and (ii)Control transfer instructions. Sequential control flow instructions are the instructions which after execution, transfer control to the next instruction appearing immediately after it (in the sequence) in the program. For example, the arithmetic, logical, data transfer and processor control instructions are sequential control flow instructions. 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 explained as follows:

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

Example 1: 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 theinstruction as a part of it.

Example 2: MOV AX, [5000H]

Here, data resides in a memory location in the data segment, whose effective address may be computed 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 it is referred using the particular register. All the registers, except IP, may be used in this mode.

Example 3: 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 registers. 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 Dl 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.

Example 4: 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 Dl respectively.

This mode is a special case of the above discussed register indirect addressing mode.

Example 5: 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 10H*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 Dl in the default (either DS or ES) segment. The example given below explains this mode.

Example 6: 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 Dl). The default segment register may be ES or DS.

Example 7: 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 contents of any one of the base registers (BX or BP) and any one of the index registers, in a default segment.

Example 8: MOV AX, 5 OH [BX][SI]

Here, 50H is an immediate displacement, BX is a base register and SI is an index register. The effective address of data is computed as 160H*DS+[BX]+[SI]+50H.

For the control transfer instructions, the addressing modes depend upon whether the destination

location is within the same segment or a 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, viz. intersegment 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. Figure 3.3 shows the modes for control transfer instructions.

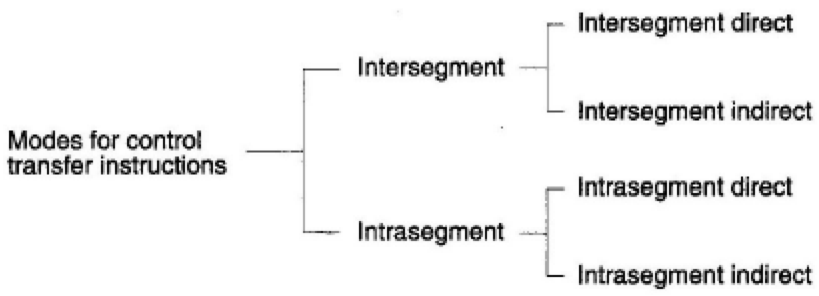


Figure 3.3: Addressing Modes for Control Transfer Instructions

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 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. $–32768 < d < +32768$), 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.

Forming the Effective Addresses: The following examples explain forming of the effective addresses in the different modes.

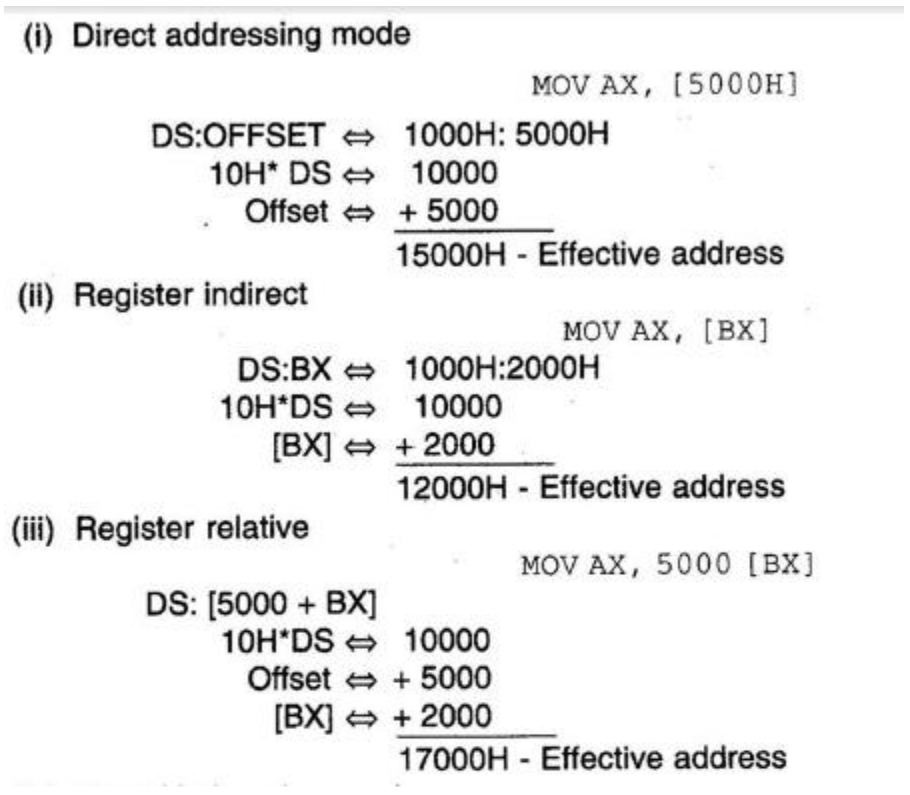
Example 9: The contents of different registers are given below. Form effective addresses for different addressing modes.

Offset (displacement) = 5000H

[AXJ-1000H, [BXJ-2000H, [SI]-3000H, [DIJ-4000H, [BP]-5000H,

[SPJ-6000H, [CSJ-0000H, [DSJ-1000H, [SSJ-2000H, [IPJ-7000H.

Shifting a number four times is equivalent to multiplying it by 160 or 10,.



(i) Direct addressing mode

```
                         MOV AX, [5000H]
DS:OFFSET ⇔  1000H: 5000H
   10H* DS ⇔   10000
    Offset ⇔  + 5000
             ------
             15000H - Effective address
```

(ii) Register indirect

```
                         MOV AX, [BX]
    DS:BX ⇔  1000H:2000H
  10H*DS ⇔   10000
     [BX] ⇔  + 2000
             ------
             12000H - Effective address
```

(iii) Register relative

```
                         MOV AX, 5000 [BX]
DS: [5000 + BX]
  10H*DS ⇔   10000
   Offset ⇔  + 5000
     [BX] ⇔  + 2000
             ------
             17000H - Effective address
```

(iv) Based indexed

```
MOV AX, [BX] [SI]
```

DS:[BX + SI]

```
10H*DS ⇔  10000
  [BX] ⇔+ 2000
  [SI] ⇔+ 3000
          15000H - Effective address
```

(v) Relative based indexed

```
MOV AX, 5000 [BX] [SI]
```

DS: [BX + SI + 5000]

```
10H*DS ⇔  10000
  [BX] ⇔ + 2000
  [SI] ⇔ + 3000
Offset ⇔ + 5000
          1A000 - effective address
```

Below, we present examples of address formation in control transfer instructions.

Example 10: Suppose our main program resides in the code segment where CS=1000H. The main program calls a subroutine which resides in the same code segment. The base register contains offset of the subroutine, i.e. BX= 0050H . Since the offset is specified indirectly, as the content of BX, this is indirect addressing. The instruction CALL [BX] calls the subroutine located at an address 10H*CS+[BX]=10050H, i.e. in the same code segment. Since the control goes to the subroutine which resides in the same segment, this is an example of intrasegment indirect addressing mode.

Example 11: Let us now assume that the subroutine resides in another code segment, where CS=2000H. Now CALL 2000H:0050H is an example of intersegment direct addressing mode, since the control now goes to different segment and the address is directly specified in the instruction. In this case, the address of the subroutine is 20050H.