8086 Instruction Set and Assembler Directives

→ 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:

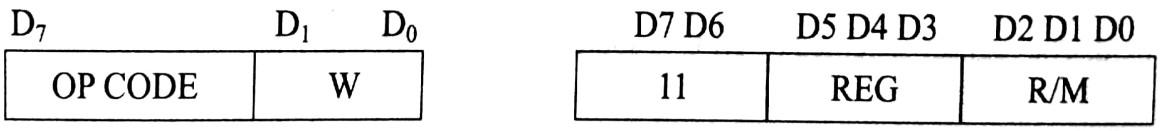
(1) One byte instruction:

→ This format is only one byte long and may have implied data or register operands.

The LSB 3-bits of opcode are used to specify the register operand

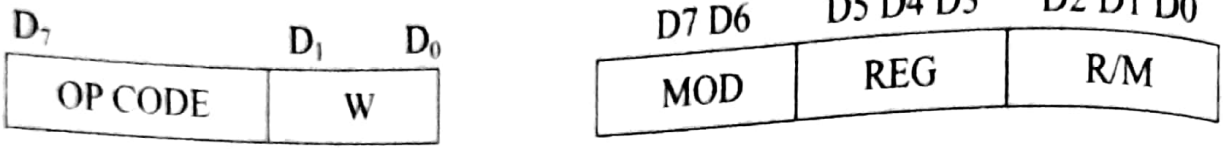
Otherwise all 8 bits form an opcode and operands are implied.

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.



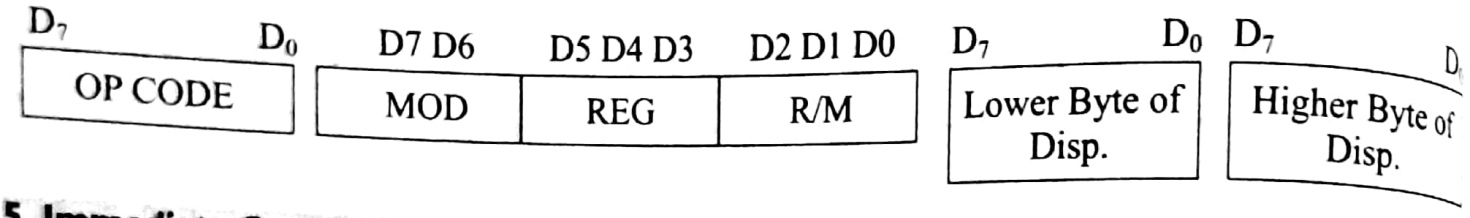
location, i.e. the other operand.

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.

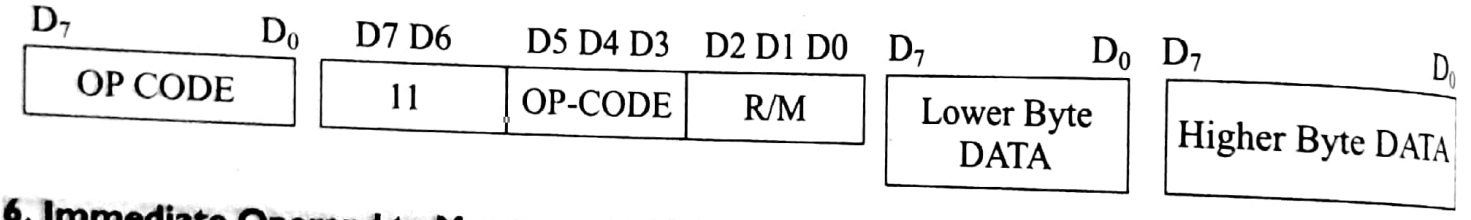


The MOD field shows the mode of addressing. The MOD, R/M, REG and the W fields are decided as Table 2.2.

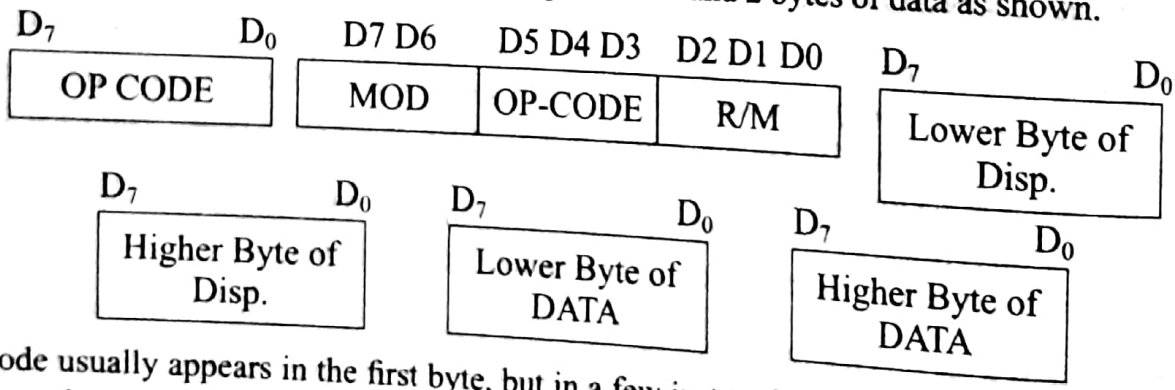
4. Register to/from Memory with Displacement This type of instruction format contains one or two additional bytes for displacement along with 2-byte the format of the register to/from memory without displacement. The format is as shown below.



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 appears 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 significances are given as follows:

W-bit This indicates whether the instruction is to operate over an 8-bit or 16-bit data/operands. If W bit is 0, the operand is of 8-bits and if W is 1, 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 if D = 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 1, 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 2.1.

Table 2.1 *Assignment of Codes with Different Registers*

W	*Register Address (code)*	*Registers*	*Segment 2 bit bit Register (code)*	*Segment Register*
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	001	CX		
1	010	DX		
1	011	BX		
1	100	SP		
1	101	BP		
1	110	SI		
1	111	DI		

Table 2.2 *Addressing Modes and the Corresponding MOD, REG and R/M Fields*

Operands	*Memory Operands*			*Register Operands*	
MOD	*No Displacement*	*Displacement 8-bit*	*Displacement 16-bit*	*11*	
R/M	*00*	*01*	*10*	*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	(BP) + (DI) + D16	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 and 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.

Addressing Modes of 8086:

→ Addressing mode indicates a way of locating data or operands.

→ Depending upon the data types used in the instruction and the memory addressing mode, any instruction may belong to one or more addressing mode or some instruction may not belong to any of the addressing modes.

→ Addressing modes describe the types of operands and the way they are accessed for executing an instruction.

→ According to the flow of instruction execution, the instructions may be categorized as
(i) Sequential control flow instructions
(ii) control transfer instructions.

Sequential control flow instruction are instructions, which after execution, transfer control to the next instruction appearing immediately after it (in the sequence) in the program.

Ex:- Arithmetic, logical, data transfer and processor control instructions are sequential control flow instructions.

control transfer instructions, transfer control to some predefined address or the address somehow specified in the instruction, after their execution

Ex:- INT, CALL, RET, JUMP instruction

The various addressing modes are explained as follows:

(1) Immediate :

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

Ex:- MOV AX, 0005H

MOV BL, 06H

(2) Direct :

→ In this addressing mode, a 16 bit memory address (offset) or an I/O address is directly specified in the instruction as a part of it.

Ex:- MOV AX, [5000H]

IN 80H

→ In the first instruction, data available in memory location whose offset value 5000H is copied into AX.

The effective addressing of the memory location is computed using 5000H as the offset and content of DS as segment address.

EA = 10H * DS + 5000H

→ In the second instruction 80H is I/O address.

(3) Register :

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

Ex:- MOV BX, AX

ADC AL, BL

(4) Register Indirect :

We can give the address of the memory location which contains data or operand in an indirect way, using offset registers. This mode of addressing is known as register Indirect addressing mode In this mode, the offset address of data is in either BX or SI, or DI register.
The default segment is either DS or ES

Ex:- MOV AX, [BX]

∴ Effective address of data is

10H x DS + [BX].

Data from above address copied into AX register.

(5) Indexed:

In this addressing mode, offset of the operand is stored in one of the index registers.

DS is the default segment for index registers, SI, and DI.

In case of string instructions, DS and ES are default segments for SI and DI respectively.

Ex:—

MOV AX, [SI]

Effective address = 10H x DS + [SI]

data from the above effective address is moved into AX register.

MOV CX, [DI]

Effective address = 10H x DS + [SI]

data from the above effective address will be transferred into CX register.

(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, DI in the default (DS or ES) segment.

Ex:- MOV AX, 50H [BX]

Effective address = 10H x DS + 50H + [BX]

data available from the above Effective address is transferred to AX register.

MOV 10H [SI], DX

Effective address = 10H x DS + 10H + [SI]

data from DX register is transferred into a memory location whose effective address is computed as above.

(7) Based Indexed:

In this addressing mode, the effective address of data is formed, by adding the contents 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]
```

effective address = 10H x DS + [BX] + [SI]

data from the above effective address of memory location is transferred into AX register

```
MOV [BX] [DI], AX
```

effective address = 10H x DS + [BX] + [DI]

data from AX register is copied into a memory location whose effective address is calculated as above.

(8) Relative Based Indexed :

In this mode, the effective address is formed by adding an 8 bit or 16 bit displacement with the sum of contents of any of the base registers (BX or BP) and any one of the index registers, in a default segment.

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

Effective address = 10H x DS + [BX] + [SI] + 50H

moves the data from a memory location whose effective address is computed as above into AX register.

Ex:- ADD 50H [BX][SI], BP

Effective address = 10H x DS + [BX] + [SI] + 50H.

The above instruction adds the contents of BP with data in memory location whose effective address is computed as above. The result is stored in the same memory location.

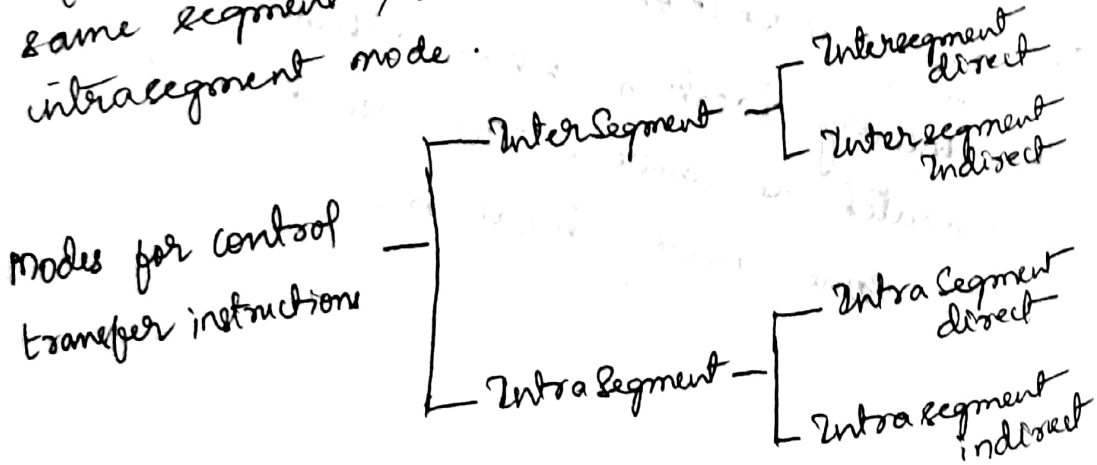
→ For control transfer instructions, the addressing modes depend upon whether the destination is within the same segment or in a different one. It also depends on the method of passing the destination address to the processor.

Basically, there are two addressing modes for control transfer instructions:

(1) Inter segment addressing mode
(2) Intra segment addressing mode.

If the location to which the control is 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.



(4) Intra Segment 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.
The displacement is computed relative to the content of 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 a jump instruction, if the signed displacement (d) is 8 bit (ie. $-128 < d < 127$) we term it as short jump and if it is 16 bits (ie $-32768 \leq d < 32767$), it is termed as long jump.

Ex:- JMP SHORT LABEL ; LABEL lies b/w -128 & 127. from current IP.

Here SHORT LABEL is an 8 bit signed displacement.

A 16 bit target address of a label indicates it lies within -32768 to 32767. But a problem arises, when one requires a forward jump at a relative address greater than 32767 or backward jump at relative address -32768.

If IP = 5000 then forward jump allowed IP + DISP = FFFFH
or DISP = FFFFH - 5000H
= AFFFH.
∴ forward jump range displacement 0000H - AFFFH.

If displacement exceeds AFFFH i.e. from B000H to FFFFH, Then all such jumps are treated as Backward jumps.

All such jumps are called NEARPTR jumps and coded as JMP NEAR PTR LABEL.

(10) Intrasegment Indirect mode :

In this mode, the displacement by 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, branch address is found as the content of a register or memory location. This addressing mode is used in unconditional branch instructions.

Ex:- JMP [BX] ; Jump to the effective address stored in BX.

(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

Ex:- JMP 5000H : 2000H

Jump to an effective address 2000H in segment 5000H.

(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 ie. contents of a memory block containing four bytes ie. IP (LSB), IP (MSB), CS (LSB), CS (MSB) sequentially. The starting address of the memory block may be referred using any of the addressing modes, except immediate.

Ex:- JMP [2000H]; jump to an address in other segment at effective address 2000H in DS.

Ex:- The contents of different registers are given below form effective address for different addressing modes.

offset (displacement) = 5000H

[AX] – 1000H, [BX] – 2000H, [SI] – 3000H,
[DI] – 4000H, [BP] – 5000H, [SP] – 6000H
[DS] – 1000H, [ES] – 2000H, [IP] – 7000H

Shifting a no. 4 times is equivalent to multiplying by 16_{10} or 10_{16}.

(1) Direct addressing mode

MOV AX, [5000H]

DS : offset = 1000H : 5000H

10H * DS = 10000

offset = + 5000

15000 → Effective address

(2) Register indirect :

MOV AX, [BX]

DS : BX = 1000H : 2000H

∴ Effective address = 10H × DS + [BX]
= 10000 + 2000
= 12000H.

(3) Register relative

MOV AX, 5000 [BX]

DS: [5000 + BX] = 1000 : [5000 + 2000H]

∴ effective address = 10H x 1000 + 5000 + 2000

= 10000 + 5000 + 2000

= 17000 H

(4) Based Indexed

MOV AX, [BX] [SI]

DS : [BX + SI] = 1000 : [2000H + 3000H]

∴ effective address = 10H x 1000 + 2000H + 3000H

= 15000H.

(5) Relative based indexed

MOV AX, 5000 [BX] [SI]

DS : [BX + SI + 5000] = 1000 + [2000 + 3000 + 5000]

∴ effective address = 10H x 1000 + 2000 + 3000 + 5000

= 10000 + A000

= 1A000

Address formation in control transfer instructions

Ex:-

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 BX contains the offset of the subroutine i.e. BX = 0050H.

∴ CALL [BX] → intra segment indirect addressing mode.

[CS] → 1000H

10H × [CS] = 10000H

[BX] = 0050

+

∴ Effective address = 10050H

Ex:- let us assume that subroutine ~~don't~~ resides in another segment where CS = 2000H.

CALL 2000H : 0050H is an example of inter segment direct addressing mode.

Effective address = 10H × 2000H + 0050H

= 20000H + 0050H

= 20050H.