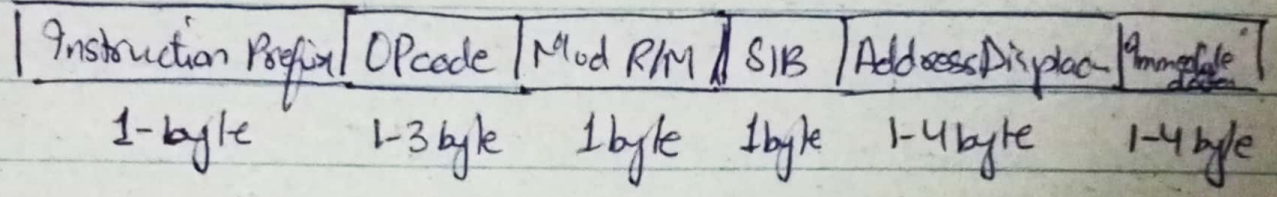
"MACHINE LANGUAGE"

— TRANSLATION —

→ To encod instruction means to convert assembly lang to machine code.

→ decode is vicevesa.

INSTRUCTION FORMAT :



→ Instructions are stored in Little Endian order, so the prefix byte is located at instruction's starting address.

→ Every instruction has opcode (other field are optional).

→ Mostly instr. are of 2-3 bytes

- INSTRUCTION PREFIX :
  overrides default operand sizes

- OPCODE :
  (Operation Code) identifies a specific variant of an instruction
  The ADD instr. for ex has nine diff opcodes, depending on parameter types used.

02 00 000 000
mod AL R/M

- MOD R/M :

6-7 bits	3-5 bits	bit 0-2
Mod	Reg/opcode	R/M

This field identifies the addressing mode & operands. The notation "R/M" stands for register & mode.

- Scale Index Byte (SIB):

bit 6-7	bits 3-5	bits 0-2
Scale	Index	Base

SIB is used to calc. offsets of array indexes.

- Address Displacement:

holds an operand's offset, or it can be added to base & index registers in addressing modes such base-displacement or base-index-displacement.

- Immediate Data:

This field holds constant operands.

SINGLE BYTE INSTRUCTION:

→ Simplest type of instruction with either no operand or an implied operand.

→ Such instr. req only the opcode field.

eg: AAA → 37 , CBW → 98 , XLAT → D7

AAS → 3F , LODSB → AC , INC DX → 42

Some commonly used instructions have been given unique opcodes.

- MOVE IMMEDIATE TO REGISTERS

→ Immediate operands are appended to instr. in Little Endian order (lowest byte first).

The ~~instr~~ encoding format for MOV

(no. 0-7 indicating which reg is used)

B8 + rw dw

opcode byte value — immediate word operand

AX / AL → 0	BP / CH → 5	
CX / CL → 1	SI / DH → 6	TABLE 03
DX / DL → 2	DI / BH → 76	
BX / BL → 3		
SP / AH → 4		

(1) eg: PUSH CX → 51

opcode for PUSH = 50 (with 16-bit register operand is 50)

opcode for CX = 1 (so adding 1 to 50).

(2) eg: MOV AX, 1 → B8 01 00

MOV → B8 (for moving 16-bit to a reg.)

AX → 0 (so adding 0 to B8 = B8)

Immediate operand → 0001 (appending to B8 in little Endian).

B8 00 10 hona chahiye tha lekin little Endian ki wajah se lowest byte 1st hogi to B8 01 00 hogya

eg: MOV BX, 1234h → BB 34 12

MOV → BB

BX → 2 (BB+3 = BB)

1234 → 34 12

- REGISTER-MODE INSTRUCTIONS:

→ In instructions using register operands, the MOD R/M byte contains a 3-bit identifier for each register operand.

R/M	Register	R/M	Register
000	AX or AL, ES	100	SP or AH
001	CX or CL, CS	101	BP or CH
010	DX or DL, SS	110	SI or DH
011	BX or BL or DS	111	DI or BH

→ The choice of 8-bit or 16-bit reg depends on the bit ~~=0~~ of the opcode field:

↳ 1 → 16-bit reg ↳ 0 → 8-bit reg

MOV AX, BX → 89 D8

Opcode for ~~moving~~ a 16-bit ~~value~~ MOV from a reg to any other oper. is 89/r.

/r indicates MOD R/M byte.

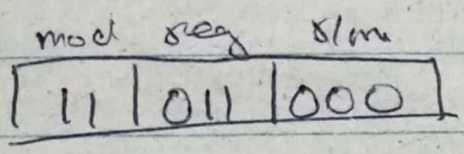
89 k aage MOD R/M wali Byte likhenge.

mod	reg	r/m
bit 6-7	3-5	0-2

→ MOD field identifies addressing mode. See table 12-18.
11 → means R/M field contain a register number

→ Reg field identifies src operand. An ex BX is register 011.

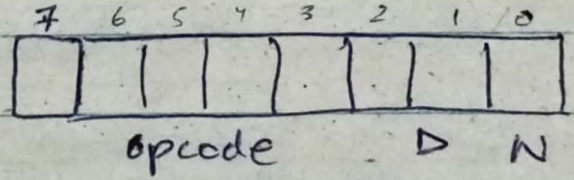
→ r/m field, which identifies destination operand.
↳ AX → 000 so R/m → 000.



89 D8

~~eg:~~ MOV AX, DX 89 11 011 000

OPCODE :



→ The 6-bit opcode identifies operation. The same opcode is used for 16-bit and 8-bit.

→ The size of operand is given by "W" bit:
W=0 → 8-bit data
W=1 → 16-bit data (or 32 bits)
mov ax, bx so W=1 as both are 16-bit
mov al, bl W=0

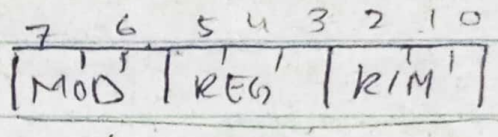
→ Bit marked "D", specifies direction of data transfer.
D=0 → then destination operand is a memory loc (not in reg mod)
eg: add [ebx], al

'D' or 'REG' field apar nein related hai

D=1 → then destination operand is reg (not in reg mod) in "REG" field

- The "D" bit specifies whether the register in "REG" field is a source or destination. If 0 then src if 1 then dest.

- MOD R/M:



→ "R/M" field combined with MOD specifies, either
↳ the second oper in a two-operand instr, or
↳ the only " " " one-oper " like NEG or NOT.
↳ These two are used to let the CPU know if there is some memory operand. if yes, then how to calc offset address. • R/M "mod" field se related hai.

→ The "MOD" field specifies x86 addressing mode

(1) ↳ MOD = 11 (means Register Mode)

(2) ↳ MOD = 00 (means memory mode with no displacement except when R/M = 110 then a 16 bit displacement follows)

(3) ↳ MOD = 01 (means memory, with 8-bit displacement following (D8))

(4) ↳ MOD = 10 (mem " " 16-bit " " (D16))

BYTE 1			BYTE 2			BYTE 3	BYTE 4	BYTE 5	BYTE 6
6 bits	1bit	1bit	2-bits	3-bits	3-bits	LOW DISP	HIGH DISP		
opcode	D	W	MOD.	REG	R/M	DATA	DATA	low data	high data

▲ General Format

→ Instruction size can range from 1–6 bytes.

→ The last 4 bytes are related with the displacement or immediate (data)
If a displacement is given that high part of displ. in Byte 4 & low in 3. If only data is given then also same.
If both displ. & data are given then displ. in 3 & 4 and data in 5 & 6.

○ mov bl, al
→ opcode for mov → 100010 (will be given)
→ REG mein humne "al" rakha hai ab chunke Al source hai to D = 0 rakhenge.
→ REG = AL = 000 → value of AL
→ W = 0 as 8-bit oper.
→ both are reg so MOD = 11
→ R/M = BL = 011 → as BL is destination

100010 00 11 000 011 = 88C3H
opcode D W MOD REG R/M

- MOV CX, DX

MOV → 100010

REG → CX → 001 (= we choose CX and it is dest). so

D = 1

W = 1 (16-bit oper)

MOD = 11 , R/M = DX = 010

100101 1 11 001 010 = 8BCAH

- ADD [BX][DI] + 4567H, DX

→ Opcode → Add → 000000

→ REG = DX → 010 (kyuka humne REG DX choose kra q k left operand memory hai).

→ D = 0 , W = 1

→ MOD = 10 (if it is memory mode with 16 bit displ.)

→ R/M = 001 (ye humne table se dekha hai +4567H)

MOD 10 mein (BX)[DI] + 16 bit ki R/M val find kri we hai 001.

000000	0	1	10	010	001	67H	45H →
Opcod	D	w	MOD	REG	R/M	high disp	low disp

01916745H.

- ADD AX, [BP][SI] + 45H

Opcode → 000000

REG → AX → 000

D = 1 (∵ REG is dest), W = 1 (16 bit oper.)

MOD = 10 (same reason

R/M = 010

000000	1	1	01	000	010	45H → 034245H

Immediate Addressing Mode:

Mov R, I:

- MOV AX, 1234H

→ Immediate to Reg:

1011	W	Reg	Low data	High data

MOV ax, 1234h

opcode → 1011

W = 1 → (16-bit oper)

Reg → AX → 000

DATA = LB = 34, HB = 12

1011 1 000 34h 12h → B8 3412h

ENCODING ONE OPERAND INSTR.

(1) SINGLE OPERAND IN A REGISTER:

There are two encoding patterns

1. Short form → which consist of just one byte and used when reg. operand is a 16 bit reg
2. General form → which consist of opcode byte followed by "mod r/m" byte and is used in any other situation than short form.

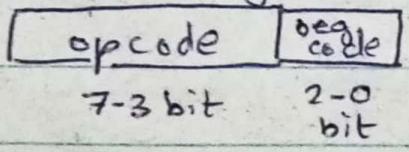
Short form with a 16-bit reg operand (single):

Two formats used for short form.

↳ One for Segment reg

↳ One for 16-bit reg

- For 16-bit Reg other than Segment Reg:



PUSH AX	50h+00	→	50h
PUSH BX	50h +03	→	53h
INC AX	40h +00	→	40h
INC SI	40h + 06	→	46h

opcode reg code → for this refer to TABLE#03

▲ reg code are added in opcodes

- Shortform for Segment Registers:

6-7 bit	5-3 bit	2-0 bit
opcode	reg code	opcode

	opcode	Regcode	Opcode	
PUSH ES	00	000	110	= 06h
PUSH CS	00	001	110	= 0Eh
POP DS	00	011	111	= 1Fh

→ The opcodes for 6-7 bit & 2-0 bit are given in APPENDIX 2.

General form of An Instruction :

	7	6	5	4	3	2	1	0
opcode byte	11		opcode			reg code		

ismein bhi wohi sab hai,

opcode extension for INC for AH the code is 100

	opcode	mod	reg	r/m	
INCAH	FE	11	000	100	→ FEC4

∵ it is reg
so mod = 11

IMUL BX	F7	11	101	011	→ F7EB

▲ Agr single Instruction with Reg ho to "REG" k andar "opcode extension" ayega. Ye Appendix 2 mein given hai.

(2) Singe Operand in a Memory :

There are diff encoding pattern for diff encoding mode.

- Direct Addressing Mode : (No displacement)

- Here mod will always be 00 ↑ it's memory and no displacement.
- R/M will be 110

INC WORD PTR [DS:200h]

opcode	mod	reg	r/m	Offset
FF	00	000	110	00 | 20

FF060002

- Indirect Addressing Mode : (No disp.)
- Here also mod = 00

INC BYTE PTR [BX][SI]

opcode	mod	reg	r/m
FE	00	000	000

→ FE 00

r/m ki value [BX][SI] mein jab mod=00 ho to 000 hoti hai.

opcode extension for INC

- Indirect Addressing Mode : (with Displ.)

- Mod will be either 01 (for 8-bit displ) / 10 (for 16bit disp)
- REG mein opcode extension
- R/M mein reg ki value
- or last mein displacement (Little Endian)

IDIV WORD PTR [DI] + 1A2Bh

opco	mod	reg	r/m	displ.	
F7	10	111	101	2B	1A

↓ opcode ↓ opc extension ↳ [DI] value at mod=10

IDIV WORD PTR [SI + 2Bh]

op	mod	reg	r/m	dis
F7	01	111	100	2B

↓ 8bit reg ↳ [SI] value at mod = 01

(3) SINGLE IMMEDIATE DATA:

When the only operand of the instruc. is an immediate data the machine language instruction is

opcode - immediate data

RET 8 → C2 08 00

Encoding Two Operand Instr.

① Register, imm Oper. (With no 'mod r/m' byte)

ADD AX, 7

opcode byte	imm. data	
05	07	00

Sagr AX, kisi imm
ke saath add hoga to opcode & imm data likhange.
0507 00

② Register, imm. Oper. (With 'mod r/m' byte)

cmp CX, 5

considering 5 as byte

opcode	11	opcode	regud	imm
83	11	111	001	05 00

mod — extension — r/m of CX

only reg is involved

③ Reg, Reg Operands
mod will be 11 as both are reg.

SUB BX, DX

opcode	11	Sec reg	1st reg
29	11	010	011

mod reg r/m

yaha bhyjd opcode mein D = 0
hoga jabki "reg" field mein
src wala reg hai.

29 → 0000 1 0 1

opcode D W → 16 bit reg

④ Reg, Memory in Direct Mode (With 'mod r/m' byte)

MOV [DS:200h], AL

opcode byte	Offset	
A2	02	00

⑤ Reg, Mem in Direct Mode (With "mod r/m byte")

MOV [DS:200h], BL

opcode	mod	reg	r/m	Offset	
88	00	011	110	00	02

mod — value of BL — value at direct address when mod = 00

⑥ Reg, Mem in Indirect Mode (With no displ)

mod = 00 (→ memory & no displ)

MOV [BX][DI], DS

opcode	mod	reg	r/m	
8C	00	011	001	

mod — value for DS — → value of [BX][DI] at mod = 00

⑦ Reg, Mem in Indirect Mode (With displ)

mod = 01 / 10 (depending on displ.)

	opcode	mod	reg	r/m	displ.	
MOV [BX][DI]+5, DS	8C	01	011	001	05	-
ADD AX, [SI]+1A 2Bh	03	10	000	100	2B	1A

val of AX — val of [SI] in mode 10

⑧ Mem Oper. in Direct Mode, imm Data

mod = 00 (→ mem and no displ)

	opcode	mod	opc-ext	r/m	offset		imm	
SUB WORD PTR [DS:20h], 1A2Bh	81	00	101	110	00	20	2B	1A

↓ value of DS

⑨ Mem Oper. in Indirect Mod (No displ), imm data

mod = 00 (→ mem & no displ)

	opcode	mod	opc-ext	r/m	offset	imm
SUB WORD PTR [BX], 5	83	00	101	111	-	05 -

↳ val of BX at mod = 00

⑩ Mem Oper in Indirect (Displ), imm data

Mod = 01 / 10 (dep. on displ)

	opcode	mod	reg	r/m	disp		imm	
SUB WORD PTR [BX]+0E0Fh, 1A2Bh	81	10	101	111	0F	0E	2B	1A

↳ val of BX at mod = 10

(i) ADD AL, [BX+SI]

opcode	mod	reg	r/m	
02	00	000	000	→ 02000
	↓ mem & no disp.	↓ AL	↓ BX+SI	

(ii) INC DX

opcode	mod	reg	r/m	
FF	11	000	010	→ FFC2
		↓ operation	↓ DX val	

(iii) MOV AX, VAR+6

opc	mod	reg	r/m	disp
B8	01	000		06
		↓ val of AX		

B8 → mov reg16, imm16 ⇒ B8 00 08h

(iv) SUB CX, VAR ; OFFSET 0008h

opc	mod	reg	r/m	
2B	00	001	110	→ 2B E
	↓ no disp	↓ val of CX		

(v) MOV [SI+490], BX

89 9C 16 FE

opcode	mod	reg	r/m	displacem	
89	10	011	100	16	FE
	↓ mod	↓ val of BX	↓ val of SI+D16 if mod=10		

1 1 1 0 1 1 0 0

1 1 1 1 1 1 0 0 1 0 0 1 0 0

(i) B9 00 12

MOV reg16, imm16 → B8 + regcd

opcode = B8

regcd = 01 → CX

imm = 1200h

MOV CX, 1200h

(ii) 8C 85 DC 01

8C → MOV reg16/mem16, Sreg

8C = 1000 1100

∴ d = 0 → so "reg" field contains src reg

w = 0

85 → 10 000 101

mod = 10

reg = 000 → ES

r/m = 101 → [DI + D16]

DC 01 → 1111 1111 1000 1001 00 → 2's comp → FE24

MOV [DI + FE24], ES

1000 1110 1010

8B 87 56 78

8B → mov reg16, mem16

8B → 1000 1011

d = 1 → so 'reg' field contain dest. reg

w = 1 → 16-bit operation

87 → 10 000 111

mod = 10 → memory mode 16 bit disp.

reg = 000 → AX

r/m = 111 → BX + D16

7856h → 2's compl → 87AAh

MOV AX, [BX + 87AAh]

87AAh

iv) 28 1D

28 → SUB mem8/reg8, reg8

28 → 00101000

d = 0 → reg field contain ~~reg~~ ~~dest~~ src reg

w = 0 → 8-bit oper.

1D → 00 011 101

mod = 00 → memory mode no disp

reg = 011 → BX

r/m = 101 → DI

SUB [DI], BX