Instruction format

Converting Assembly Language Instructions to Machine Code in 8086 microprocessor:-

To convert an assembly language program to machine code, we must convert each assembly language instruction to its equivalent machine code instruction. The machine code for an instruction specifies things like:

- what operation is to be performed,
- what operand or operands are to be used,
- whether the operation is performed on byte or word data,
- whether the operation involves operands that are located in registers or a register and a storage location in memory,
- and if one of the operands is in memory, how its address is to be generated.

All of this information is encoded into the bits of the machine code for the instruction. The machine code instructions of the 8086 vary in the number of bytes used to encode them. Some instructions can be encoded with just 1 byte, others can be done in 2 bytes, and many require even more. The maximum number of bytes might take is 6.

- Single-byte instructions generally specify a simpler operation with a register.

  Ex: Complement carry (**CMC**) is specified by the machine code byte 11110101**B**, which equals F5**H**,. That is,

  CMC = 11110101B = F5H

- The machine code for instructions can be obtained by following the formats used in encoding the instructions of the 8086 microprocessor. Most multi-byte instructions use the general instruction format shown in Fig. 1.

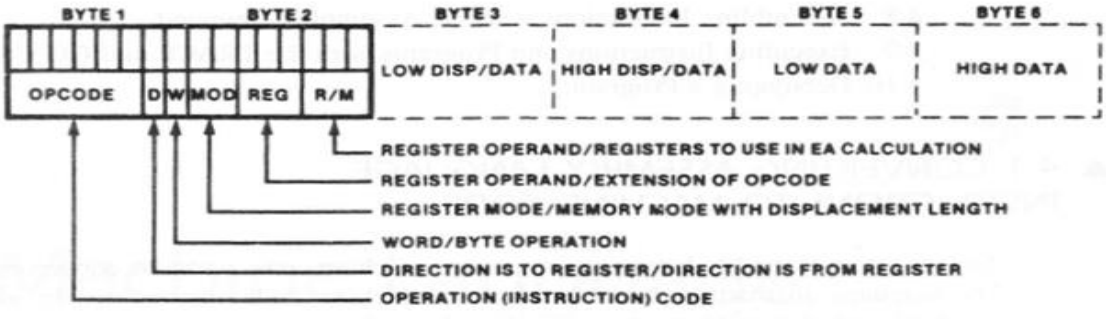

Figure -1- General instruction format. (Reprinted by permission of Intel Corp. Copyright/Intel Corp. 1979)

Byte 1 contains three kinds of information:

Opcode	D	W

- Opcode field (6 bits) specifies the operation (add, subtract, move)
- Register Direction Bit (D bit) Tells the register operand in REG field in byte 2 is source or destination operand

  1: destination 0: source
- Data Size Bit (W bit) Specifies whether the operation will be performed on 8-bit or 16-bit data

  0: 8 bits 1: 16 bits

Byte 2 has three fields:

MOD	REG	R/M

- Register field (REG) used to identify the register for the first operand , which is the one that was defined as the source or destination by the D bit in byte 1. Table below shows the encoding for each of the 8086's registers. Here we find that the 16-bit register AX and the 8-bit register AL are specified by the same binary code. Note that the decision whether to use AX or AL is made based on the setting of the operation size (W) bit in byte 1.

REG	W = 0	W = 1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

- Mode field (MOD)
- Register/memory field (R/M field)

The 2-bit MOD field and 3-bit R/M field together specify the second operand. Encoding for these two fields is shown in Figs. -2-(a) and (b), respectively. MOD indicates whether the

operand is in a register or memory. Note that in the case of a second operand in a register, the MOD field is always 11.The R/M field, along with the W bit from byte 1, selects the register.

CODE	EXPLANATION
00	Memory Mode, no displacement follows*
01	Memory Mode, 8-bit displacement follows
10	Memory Mode, 16-bit displacement follows
11	Register Mode (no displacement)

Fig-2-
(a)

MOD = 11			EFFECTIVE ADDRESS CALCULATION			
R/M	W = 0	W = 1	R/M	MOD = 00	MOD = 01	MOD = 10
000	AL	AX	000	(BX) + (SI)	(BX) + (SI) + D8	(BX) + (SI) + D16
001	CL	CX	001	(BX) + (DI)	(BX) + (DI) + D8	(BX) + (DI) + D16
010	DL	DX	010	(BP) + (SI)	(BP) + (SI) + D8	(BP) + (SI) + D16
011	BL	BX	011	(BP) + (DI)	(BP) + (DI) + D8	(BP) + (DI) + D16
100	AH	SP	100	(SI)	(SI) + D8	(SI) + D16
101	CH	BP	101	(DI)	(DI) + D8	(DI) + D16
110	DH	SI	110	DIRECT ADDRESS	(BP) + D8	(BP) + D16
111	BH	DI	111	(BX)	(BX) + D8	(BX) + D16

Fig-2- (b)

Ex: The instruction

MOV BL, AL

"move the byte contents from source register AL to destination register BL."
Using the general format in Fig. -1-, show how to encode the instruction in machine code. Assume that the 6-bit opcode for the move operation is 100010.

Solution: In byte 1, the first six bits specify the move operation and thus must be 100010.

OPCODE = 100010

The next bit, D, indicates whether the register specified by the REG part of byte 2 is a source or destination operand. Let us say that we will encode **AL in the REG field** of byte 2; therefore, D is set equal to 0 for source operand.

D = 0

The last bit (W) in byte 1 must specify a byte operation. For this reason, it is also set to 0.

W = 0

This leads to

Byte 1 = 10001000 B

In byte 2, the source operand, specified by the REG field, is AL. The corresponding code from table is

REG = 000

Since the second operand is also a register, the MOD field is made 11. The R/M field specifies that the destination register is BL, for which the code (from Fig. -2-(b)) is 011. This gives

MOD = 11

R/M = 011

Therefore, byte 2 is

Byte 2 = 11000011 B

Thus, the hexadecimal machine code for the instruction is given by

MOV BL,AL =88 C3 H

Ex:

Tie instruction

ADD AX, [SI]

"add the 16-bit contents of the memory location indirectly specified by **SI** to be contents of **AX**." Encode the instruction in machine code.

The opcode for add is **000000**.

Solution: To specify a 16-bit add operation with a register as the destination, the first byte of machine code will be

Byte 1 = 00000011B = 03H

The **REG** field bits in byte 2 are **000** to select **AX** as the destination register. The other operand is in memory, and its address is specified by the contents of **SI** with no displacement, In Figs. -2-(a) and (b), we find that for indirect addressing using SI with no displacement, **MOD** equals **00** and **R/M** equals **100**. That is,

MOD = 00

R/M = 100

This gives

Byte 2 = 00000100B = 04H

Thus, the machine code for the instruction is

ADD AX, [SI] = 0304H

Ex: What is the machine code for the instruction

XOR CL, [1234H]

"exclusive-OR the byte of data at memory address 1234H with the byte contents of CL." The opcode for exclusive-OR is 001100.

Solution: Using 001100 as the opcode bits, **1** to denote the register as the **destination operand**, and **0** to denote **byte data**, we get

BYTE1 = 00110010B = 32H

The **REG** field has to specify **CL**, which makes it equal to **001**. In this case, a direct address has been specified for operand 2. This requires **MOD = 00** and **R/M = 110**. Thus,

BYTE2 = 00001110 B = 0E H

To specify the address 1234H, we must use **byte 3** and **byte 4**.
The least significant byte of the address is encoded first, followed by the most significant byte. This gives

BYTE 3 = 34H

and

BYTE 4 = 12H

Thus. the machine code form of the instruction is given by

XOR CL, 1234H = 32 0E 34 12 H