Module - IV

Introduction To 8086/88 Mp.

* 8086 Mp is the enhanced version of Intel 8085 Mp
* Designed by Intel in 1978.
* 8086 Mp is a 16-bit Mp.
* Has 16-bit data bus, so it can read data from / write data to memory & ports.
* Has 20 bit address bus.
* 8086 supports 2 modes of oprations —
  a) min mode
  b) max mode

→ Diffrence b/3 8085 & 8086 Mp =

8085 Mp	8086 Mp
* 8-bit Mp	16-bit Mp
* 16-bit ad. line	20-bit ad. line.
* memory capacity is 64KB	memory capacity is 1 MB.
* Have 5 flags	Has 9 flags.
* Does not support pipeline	Supports pipeline.
* Does not support memory Segmentation	~~Does not~~ Support memory Segmentation

⇒ 8086 / 88 Mp.

* They are part of the X86 family of processors, which is still widely used today in personal comp & other embedded systems.

→ Internal Architecture of 8086 | 88 =

* Includes several functional units, including ALU, the EU (Execution unit), BIU (Bus Interface Unit) & Segmnt R)
* BIU responsible for managing memory and I/o operation.
* uses a segmnted memory architecture, which allows it to address more memory than 16-bit architecture.

* Features	8086	8088
* Data bus	16-bit	8-bit
* Address bus	20-bit	20-bit
* max memory address	1 MB	1 MB (bt only 640kB accessible)
* (I) set	same as 8088	same as 8086
* Pin count	40	40
* Applications	high performnce PCs	low-cost PCs.

* 8086 Mp is a higher-perfomance processor than 8088 Mp

I BIU (Bus Interface Unit) = (4)

* fetch the (I)
* write data to memory
* write data to port
* Read data frm port.

II EU (Execution Unit) = (3)

* To tell the BIU where to fetch the (I).
* To decode the (I)
* To execute the (I).

III General purpose (R) = (16-bit)

✓ a) AX : [Acc (R)] → Stores the rslt.
✓ b) BX : [Base (R)] → Stores the starting base loc. of memory.
✓ c) CX : [Counter (R)] → used in loop (I).
✓ d) DX : [Data (R)] → used to contain I/o port address for I/o (I).

* Instruction pointer (Ip) :
  16-bit R) that holds the memory adr- of nxt (I) to be executed.

* Instruction Queue :
  It is a buffer that holds upto 6 (I)s of the crnt (I) being executed.

* Segment (R) :
  used to specify the starting adr- of a segment in memory.

↓

BIU ()s

* BIU 3 ()s → (I) pointer, segmnt (R) and (I) queue.

IV Segmnt (R) = (4 types)

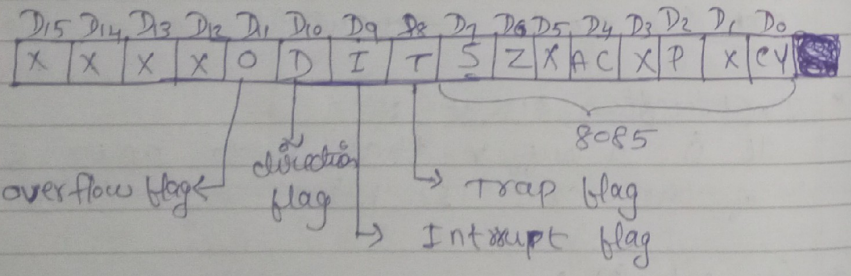
a) CS (Code Segmnt) =
used for addressing a m. loc where the prgms are stored.

b) DS (Data Segmnt) =
Contain data which is used by the prgm.

c) SS (Stack Segmnt) =
Defined as area of memory used for stack.

d) ES (extra segmnt) =
Similar to data segmnt where additional data is stored.

V Flag (R) = (9 flag (R))



* overflow flag :
OF = 1 → if the rslt is too long the nom too small the nom in ACC.

* Direction flag :
if DF = 1 → auto increment mode

if DF = 1 → auto decremnt mode. used for string manipulation (I).

* Interrupt flag :
IF = 0 → maskable Interrupts are recognised.

* Trap flag :
TF = 0 → if it is set, the processor will enter in single step execution mode.

⇒ Segmentation =

* It is memory managmnt technique used in comp system to divide memory into segmnts, each with its own unique ad. & size
* can be used to separate diffrnt types of memory like code & data.
* In 8086 Mp, segmntation is implemnted using Segmnt (R).
* 2 types — code segmnt & Data segmnt.

	Code Segmnt	Data Segmnt
a) purpose	contains Prgm (I)	contains prgm data.
b) (R) used	CS (Code Sgmnt)	DS (Data Sgmnt)
c) eg	machine (I)	variables, arrays
d) size	varies based on prgm size	varies based on prgm data.

8	4	2	1	
1	0	0	0	→ 8
0	1	0	0	→ 4

⇒ Physical address calculation =

* Segmnt (R) = contains 16-bit address which points to the starting ~~address~~ loc. of each segmnt.
* Offset address = within a segmnt particular loc. is pointed by offset ad.
* Physical address = 20-bit ad. obtained by adding segmnt (R) & offset ad.
* segmnt (R) is shifted to left by adding 0H to segmnt ad.
* eg → ①

Segmnt ad = 1034 H
offset ad = 4404 H.

1034H x 10H → 10340H (16)

4404 → 0100 0100 0000 0100.
10340H →
1000 0000 0011 0100 0000 +
1000 0100 0111 0100 0100

8 4 7 4 4 → PA

∴ 84744H is physical ad (PA).

② CS → 4042H.
value of offset ad → BX : 2025H
IP : 0580H
DI : 4247H.

= 4042H x 10H + 0580H.
= 40420 H + 0580H
= 41000 H

IP (R) → holds offset address

② CL → counter low → ~~CX (R) is 16 bit (R)~~ CL (R) is lower 8 bit of CX (R) (16 bit)
CH → counter Higher (8 bit)

⇒ Addressing modes of 8086 :

1) ~~Immediate A mode~~ =

Addressing modes	Description	eg.
1) Immediate A. mode	operand value is directly specified in the (I).	* MOV AX, 1234H, * MVI CL, 12H. (b)
2) Register A. mode	Operand value is in a (R)	* ADD AX, BX., * MOV AX, BX.
3) Direct memory A. mode	operand value is in memory at the address specified.	* MOV AX, [1234H] * MOV CL, [4321H] [4321-ad 2222 data-now CL-copy mov]
4) Register based indirect	operand value is in memory with the address specified by the contents of a (R)	* MOV AX, [BX] [16-bit data (BX) into AX] * MOV AL, [BX] (b)
5) Register relative	operand value is in memory with the ad- specified by the contents of a (R) + a displacmnt	* MOV AX, [BX+2H]. * MOV CL, [BX+04H]
6) Base Indexed	operand value is in memory with the	* MOV AX, [BX+SI]

6) AL (Acc Low) → 8 bit lower of AX (R)
AH (Acc High)

	add. Specified by the sum of the cntnts of 2 (R).	SI → Source index (R)
7) Relative based indexed A. mode	operand value is in memory with the add Specified by the sum of the cntnts of 2 (R) + a displmnt	MOV AX, [BX+SI+2]
8) Implied A. mode	No operand is Specified in (I)	HLT, STC.

~~* MOV AX, BX~~

* GPR'S (continution)

a) AX [Acc (R)]
b) BX [Base (R)]
c) CX [counter (R)]
d) DX [Data (R)]
e) SI [Source Index (R)] =
used as pointer to Source data in memory for string operations.
f) DI [Destination Index (R)] =
used as pointer to the destination data in memory for string operations.
g) BP [Base pointer (R)] =
used as base pointer for stack oprtion.
h) SP [Stack pointer (R)] = used as Stack pointer for stack oprtions.

* Index (R) =

* Also → Base (R) | pointer (R).
* 16-bit (R) that holds a memory add. used as a base / offset for memory access.