DLDM unit-4

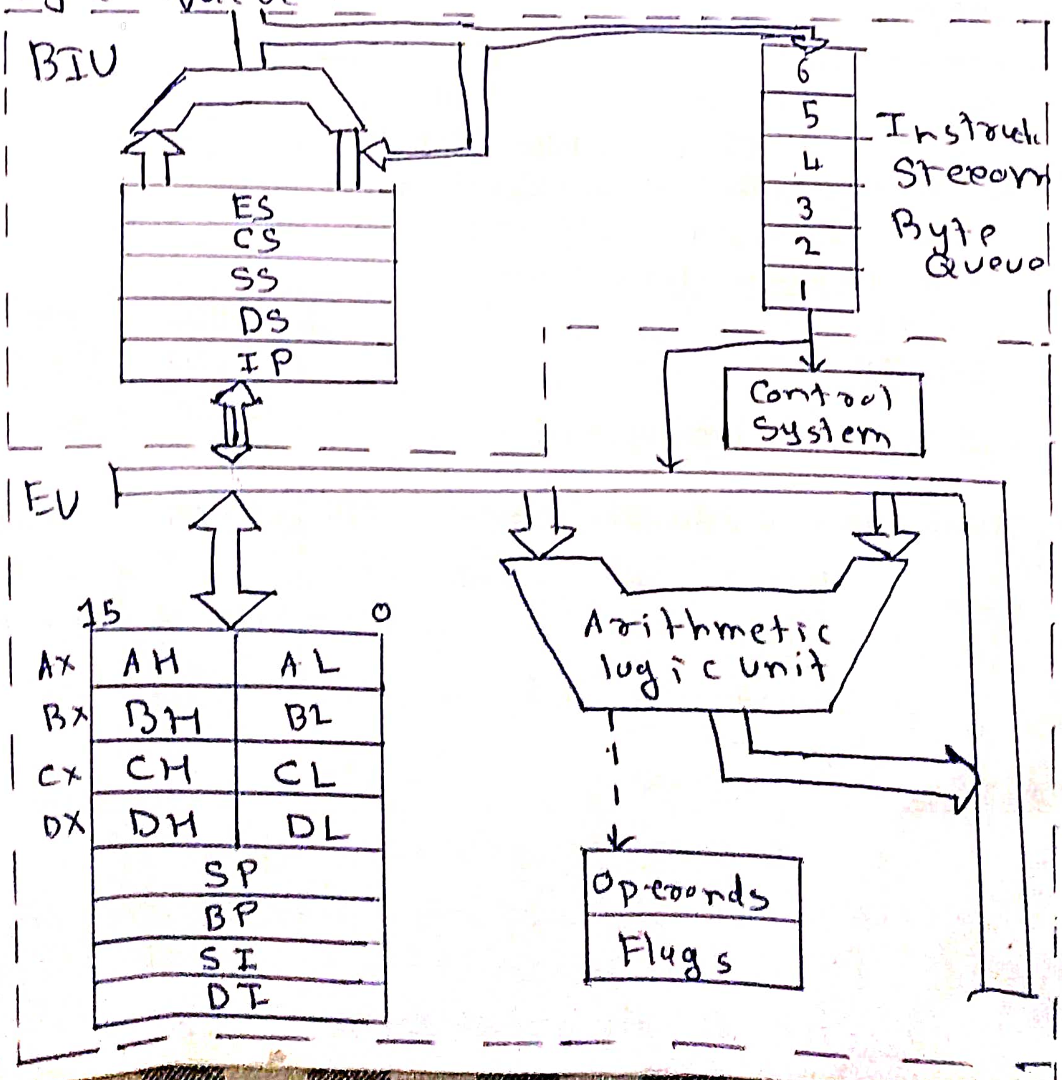
8086 Microprocessor & Assembly lang

★ Architecture of 8086

- 16 bit Microprocessor.
- 20 address lines & 16 datalines.
- has a storage upto 1 MB (memory)
- 8086 has two main blocks:-

① Bus Interface Unit (BIU):-
Bus Interface Unit performs all operation related to systembus (add bus, data bus, control bus) like instruction fetching reading and writing operands and calculating the address of memory operands.

② Execution Unit (CEU):-
execution unit executes instructions from the instr system byte queue.



- Both units operate asynchronously to give 8086 an overlapping instruction fetch and execution mechanism which is called PIPELINING
- Fetching the next instruction while the current instruction executes is called pipelining.

A] Bus Interface Unit (BIU) :-

- Provides 16 bit data bus & 20 bit address bus.
- responsible for all external bus operations like
  1. Instruction fetch
  2. Instruction queuing
  3. Operand fetch & storage.
  4. Address relocation
  5. Bus Control.
- BIU uses a mechanism known as instruction stream queue to implement pipline architecture.
- This queue permits prefetch of upto six bytes of instruction code.
- These inst^n are held in it's FIFO queue.
- with it's 16 bit data bus, the BIU fetches 2 inst^n bytes in a single memory cycle.
- after a byte is loaded at the I/P end of the queue, it automatically shift's up.
- EU access the queue from the O/P end it reads one instruction byte after the other from the output of the queue.
- Intervals of no bus activity, which may occure between bus cycles are known as IDLE State.
- If the BIU is alredy in the process of fetching an instruction when the EU request it to read or write operands from memory or I/O, the BIU first complete the inst^n fetch. before read Start

- BIU contains an address adder which is used to generate ~~[illegible]~~ 20 bit physical add. that is o/p on the address bus.
- This address is formed by adding or appended 16 bit segment address & 16 bit off set address.
- BIU is also responsible for generating control signals like memory read/write, I/O/read/write

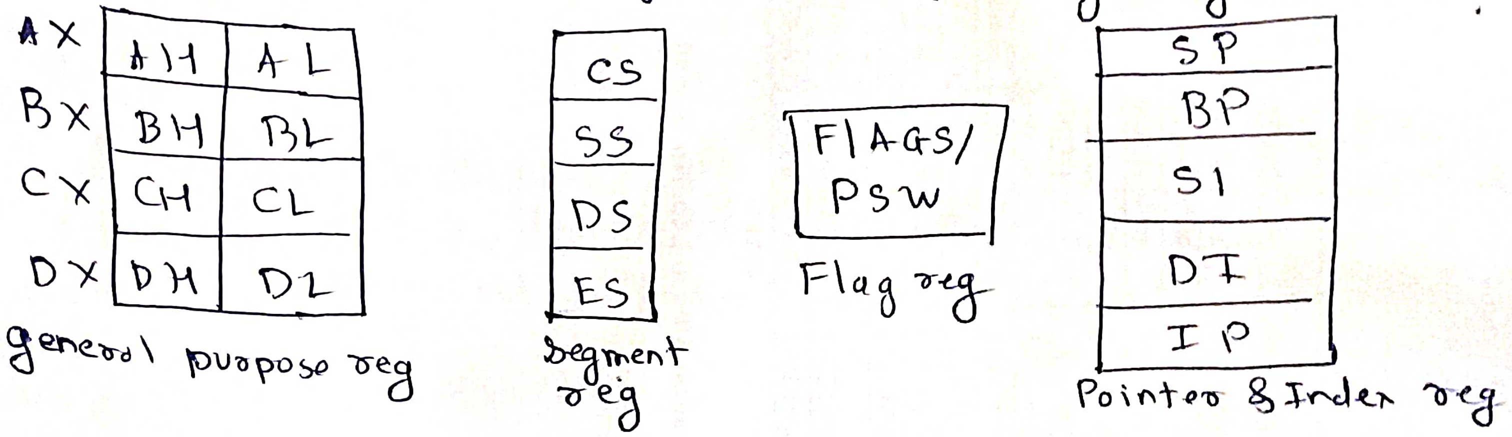
B] Execution Unit

- responsible for decoding & executing all instructions
- EU extracts inst from the top of the queue, decodes them, generate operands, passes them to BIU and request it to perform read or write operation & perform operation specified by instⁿ.
- EU tests the status & control flags and updates them based on the results.
- queue is empty, EU waits for next instⁿ byte to be fetched & shifted to top of the queue.
- when EU executes branch or jump instⁿ it transfers control to a location, corresponding to another set of sequential instⁿ.
- BIU automatically reset the queue & then begin to fetch the instⁿ from this new location.

★ Registers of 8086

- 8086 up has 16 bit general purpose & special purpose register.
- The register are as fallows.

1] General Data Registers 2] Segment Registers
3] Pointers & Index Registers 4] Flag Register.



A] General Data Registers

- These register can be used as either 8-bit reg or 16 bit reg.
- They may be either used for holding data, variables & intermediate results temporarily
- and storing offset address for some particular addressing mode etc.

AX	AH & AL	→	Accumulator
BX	BH & BL	→	offset storage.
CX	CH & CL	→	counter reg.
DX	DH & DL	→	Destination operand.

B] Segment Registers

- Segment registers are use to store 16 bit of starting address of a particular segment
- 1 MB is divided into 16 segments having 64k memory
- The content of segment reg is seg Base address
1] Code Segment Register (CS) ⇒ holds the address where executable program
2] Data Segment Register (DS) ⇒ holds where data is stored
3] Extra Segment Register (ES) ⇒ is segments & also contain where is data
4] Stack Segment Register (SS) ⇒ holds stack segment of memory.

C· Pointer & Index register

- Pointer · contain offsets .
- physical address = Base addr + offset addr .

IP → Instruction pointer → store memory location of next instructions to be executed

BP → Base pointer → store offset of ~~code~~ Data .

SP → Stack pointer → store offset of stack segment .

- Index reg are use as general purpose reg as well as offset store .

SI → Source Index → store offset of source data

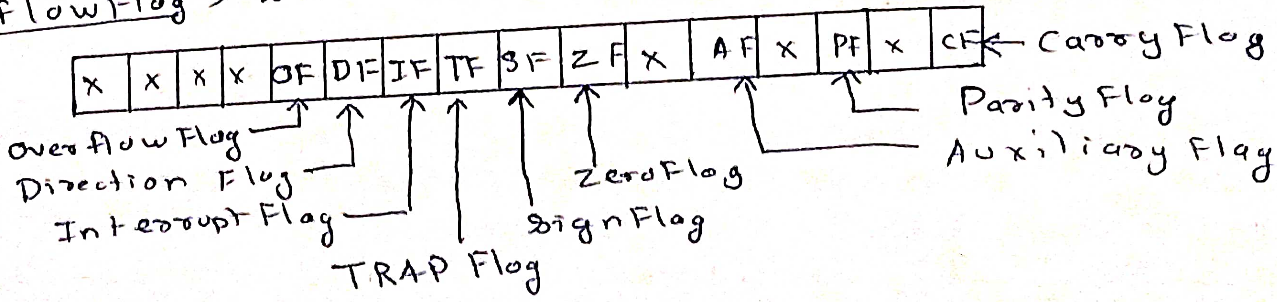
DI → Destination Index → store offset of destinat. data .

D] FLAG Register

- 8086 up show the result in Flag reg after operation
- 6 Status Flag and 3 Control Flag .

set = 1
reset = 0

1] Carry Flag → if carry set to 1 or reset
2] Parity Flag → 1 = even parity   0 = odd parity
3] Auxiliary Flag → 1 = carry from lower to higer else 0
4] Zero Flag → if result ⇒ 0 set to 1 else set to 0
5] Sign Flag → set if MSB is 1 else 0
6] Trap Flag → set if single stepping else reset .
7] Interrupt Flag → set if Enable interrupt else reset .
8] Direction Flag → set if auto Decrement else AutoIncrement .
9] Overflow Flag → set if overflow occure else reset .


X
X
X
X
OF
DF
IF
TF
SF
ZF
X
AF
X
PF
X
CF
Carry Flag
Parity Flag
Auxiliary Flag
Over flow Flag
Direction Flag
Interrupt Flag
TRAP Flag
Sign Flag
Zero Flag