Microprocessor 8086.

20 address lines, 16 data lines.
Operations like multiplication & division can be done directly.

→ General purpose registers
→ Special purpose registers } 16-bit

1) General Data Registers
2) Segment registers
3) Flag/PSW registers
4) Pointers and index register.

(1)

AX	AH	AL
BX	BH	BL
CX	CH	CL
DX	DH	DL

• Register Cx is used as a default counter in case of string and loop ~~counters~~ operations. [string → set of data bytes stored in consecutive locations.]

• Ax → 16 bit accumulator.
• Bx → Bx is used as an offset storage for forming physical address of memory in case of certain addressing mode.
• Dx → Dx is the general purpose register which may be used as an implicit operand/destination in case of few instructions/operations

(2) 4-segment registers →

Code segment register (CS)
Data " " (DS)
Extra " " (ES)
Stack " " (SS)

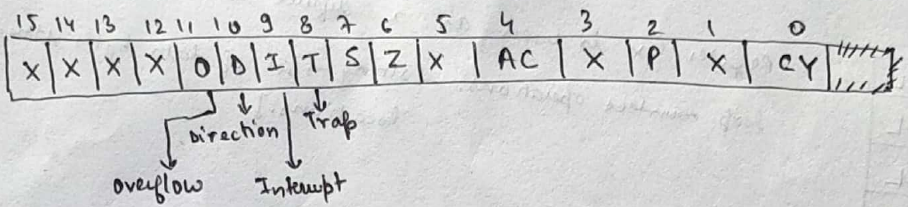
→ CS is used in the code segment of the memory.
→ DS points to the memory where data is resided.
→ ES is also a portion where data is saved.
→ SS is used for addressing stack segment of memory.

(3) Flag/PSW registers:
Divided into 2 parts:
(i) Condition code/Status flag → Consists of lower order bytes of 16-bit register along with overflow flag.
(ii) Machine Control flag → Consists of higher order byte of 16-bit flag register of 8086. & consists of 3 flags:-

(a) Direction flag → Used by string manipulation instructions. If this flag bit is zero, the string is processed beginning from the lowest address to the highest address in auto-incrementing mode. Otherwise, string is processed in auto-decrementing mode.

(b) Interrupt flag → If this flag is set, maskable interrupts are recognised by CPU otherwise ignored.

(c) TRAP flag → If this flag is set, processor enters single execution mode. Processor executes current instruction and control is transferred to the TRAP Interrupt Service Service Subroutine (ISS).



20 bit ⇒ 1 MB ≡ 16 segments (64K)

④ $\underbrace{\text{SP, BP}}_{\text{stack}}$, SI, DI and IP

SP, BP → Contains off-set address for stack.

→ Register SI is used to store offset of the data segment while register DI is used to store off set of extra segment.

→ IP contains off-set address for code segment of memory.

Complex architecture of 8086 is divided into 2 parts:

1) BIU (Bus Interface Unit):-

BIU contains ckt for physical address calculations and a predecoding instruction byte queue which is 6 byte long.

2) Execution Unit (EU):-

While BIU is operating, an opcode is fetched after completion of execution, done by EU.

entiate b/w interrupts and polling.

Interrupt.	Polling.
1) Here, device which is in service sends a notification to the controller by sending an interrupt signal.	1) Here, microcontroller continuously monitors the status of a given device.
2) Upon receiving notification, the controller finishes its current instruction and serves the device using ~~x~~ ISS.	2) Here if status conditions are met, then only controller serves the device.
3) Priorities can be assigned to the interrupts & service is also rendered acc. to priority.	3) Polling method cannot assign priority since it checks all devices in a round-robin fashion.
4) The controller can mask a device request for service.	4) This is not possible in polling method.
5) More effective method of servicing interrupts as time is efficiently used to service the device which is in urgent need.	5) This method wastes much time of the controller by polling devices that do not need much service. (tying down controller)

6 interrupts in 8051 :

i) Reset → when it is activated, 8051 jumps to location 0000H.

2) 2 interrupts are set aside for the timer — 1 for timer 0 & one for timer 1.

3) 2 interrupts are set aside for external hardware interrupts. Pin nos. 12 & 13 in port 3 are assigned for the external hardware interrupts. INT 0 also known as EX1, and INT 1/EX2 are assigned.

4) Serial communication has a single interrupt that belongs to both receive & transmit.

<u>Enabling and disabling an interrupt</u> →

The interrupts must be enabled by software. There is a register called <u>interrupt enable (IE)</u> that is responsible for enabling & disabling interrupts.

D7	D6	D5	D4	D3	D2	D1	D0
EA	–	ET_2	ES	ET_1	EX_1	ET_0	EX_0

(cont.....)

IE.7 → Disable all interrupts in EA = 0. If EA = 1, each Interrupt source is individually enabled/diabled by setting/clearing its enable bit.

IE.6 → D6 is not implemented.

IE.5 → D5 enables/diables timer 2 overflow.

IE.4 → ~~D4~~ Enables/disables serial port interrupt.

IE.3 → Enables/diables timer 1 overflow.

IE.2 → ~~&~~ " extend interrupt 1.

IE 1 → " timer 0 overflow interrupt.

IE 0 → " extend interrupt 0.