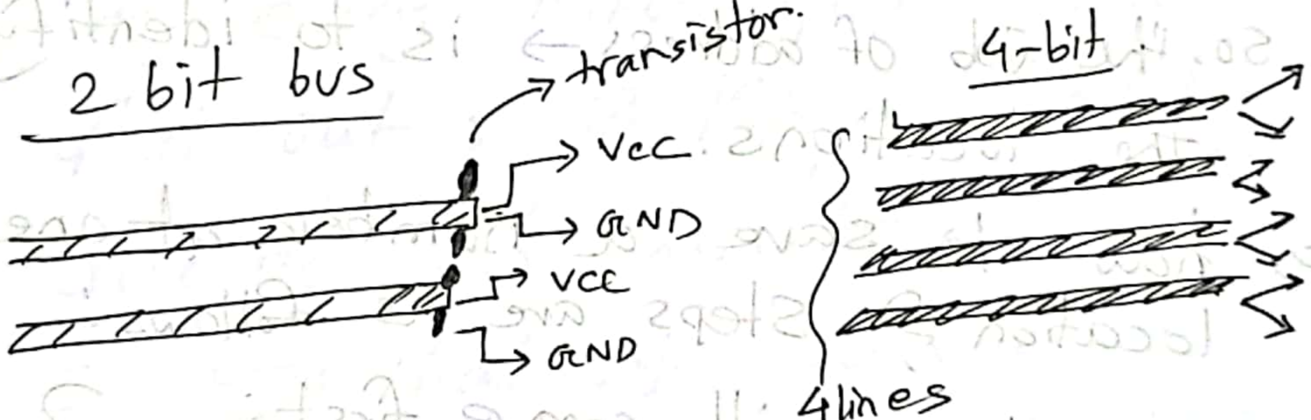
1) Memory ⇒ stores prog & data

2) MPU → fetch
→ decode
→ execute

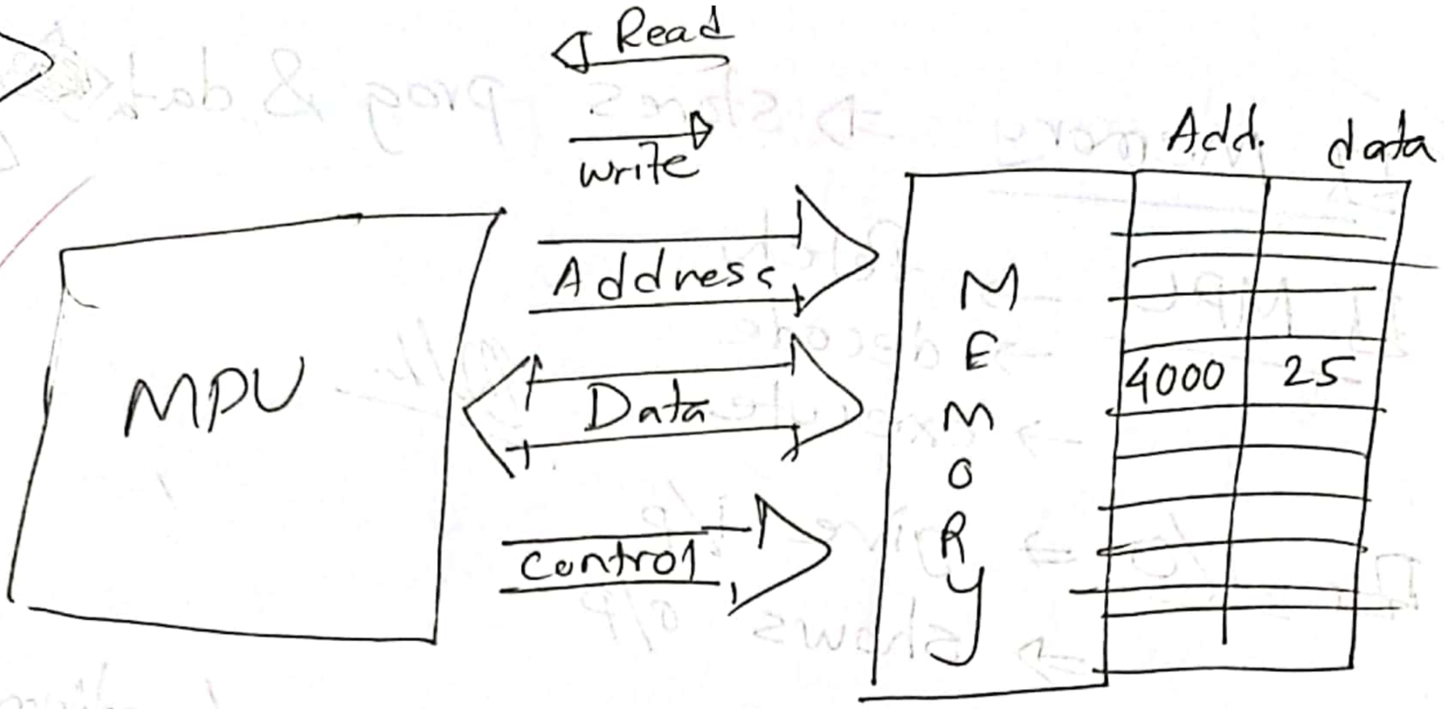
[illegible]

3) I/O ⇒ give i/p
⇒ shows o/p

4) Buses ⇒ set of lines/mediums.
↳ Address.
↳ data.
↳ control.



It is always a trade off: since the more lines the more info will come, so the cost will be higher & also the space will be an issue.



Since there are millions of locations every location has its own unique address. (during admission test you can remember).

// So, the Job of address → is to identify the locations.

So, how to save a number at one location? Steps are as follows:

1) Address bus will come first; will give the address - 4000

2) data bus will carry 35;

3) Control bus will WRITE 35 in location 4000

} WRITE operation

→ Read & write always mention with respect to MPU. (permitted by MPU)

[illegible] [illegible] // 8086 Architecture = [illegible] [illegible]

→ 16 bit MPU ; so always deals with 16-bit transmission.   EU-2

→ two units

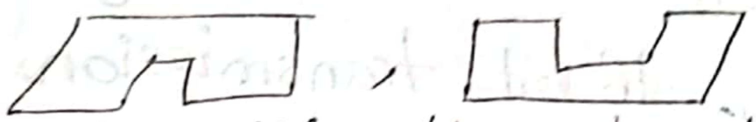
→ BIU & EU.

→ for every MPU, you need to find out : 3 things → fetch, decode & execute.
So, ~~you~~ that you can trace out every MPU. since all MPU does the 3 above tasks.

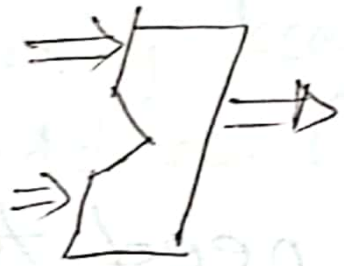
II 8086 Architecture

→ 8086 has 16-bit data bus.

→ 1st thing: every thing is rectangular in shape except:



Arithmetic circuit



→

Physical address calculation

segment × 10 + offset

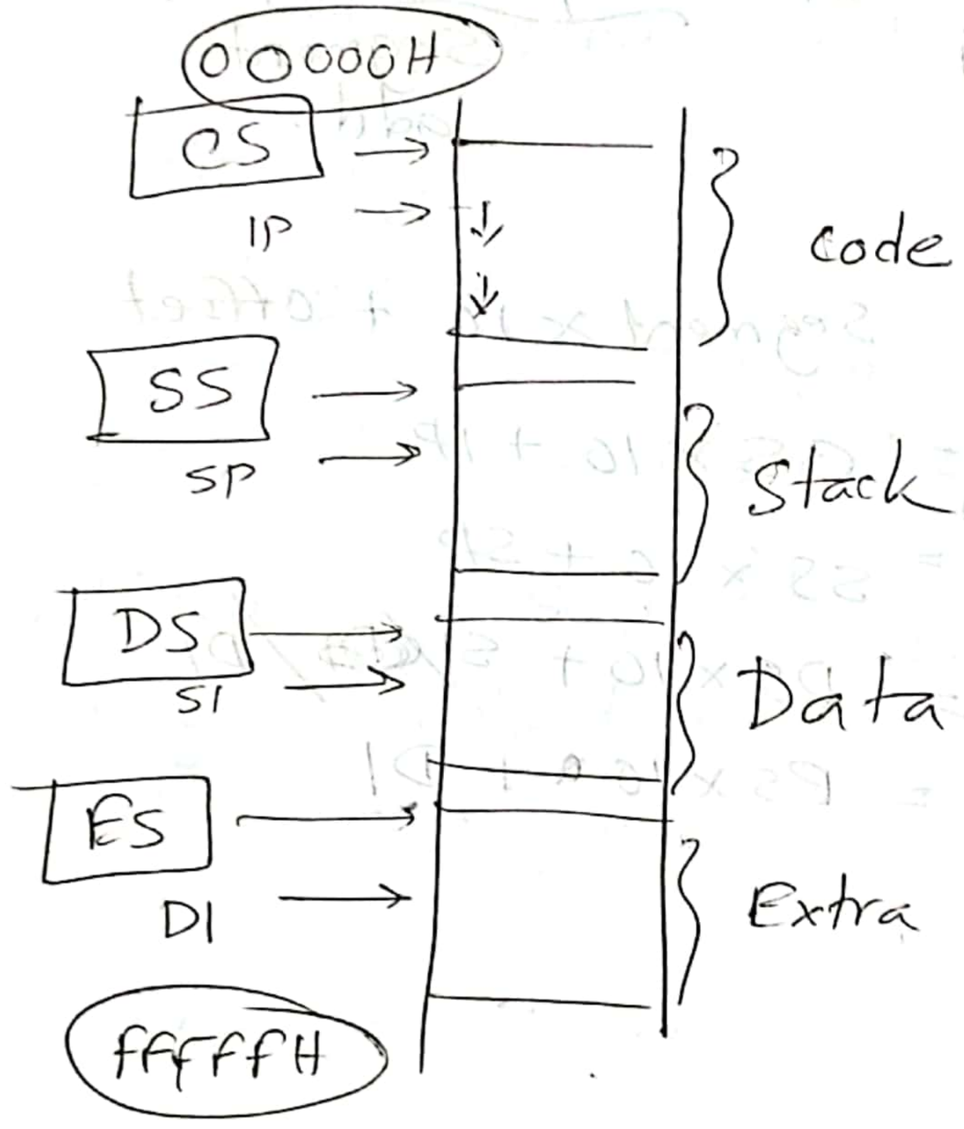
= 20 bit address.

∘ 1 TB → inside HD

- Folder name
- file name

} virtual address

{ in reality there are 1 trillion location



Memory is divided into four segments.

Assume : you want to find out page 564.

So, if all the chaptors are 100 page, you will go to ~~chapter 5 a~~

$\underbrace{\text{pg - 64}}_{\text{offset add.}}$ of $\underbrace{\text{chapter 5.}}_{\text{Segment add.}}$

Physical = Segment × 10 + offset

Calculated by MPU

$$\left\{\begin{aligned} &= CS \times 10 + IP \\ &= SS \times 10 + SP \\ &= DS \times 10 + SI / DI \\ &= ES \times 10 + DI \end{aligned}\right.$$

[] Who will fetch the instrutions?

→ the BIU ; So the physical address calculation should be done by BIU for fetching the next instruction, while EU is executing.

// CS/DS/ES/SS → are not segments;
Segments are present in memory.

// these are segment registers they contain the starting addresses of all the segments, (16-bit Reg)

// After calculating the physical address, instructions gets fetched through the data bus

/ Address bus → locaties the location where to go.

/ data bus → instructions/ data comes in/out.

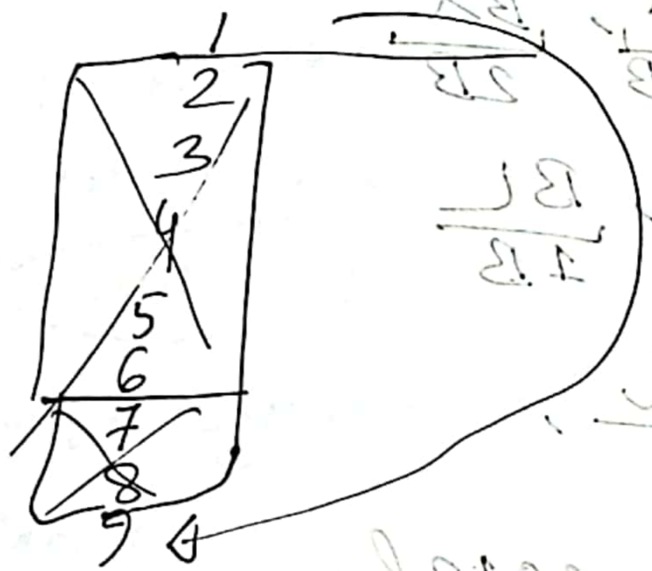
// control bus → gives $\left.\begin{array}{l}\overline{RD} = \text{Read} \\ \overline{WR} = \text{write}\end{array}\right]$ signal.

/ After fetching : we are not going to execute right now, ∵ bec. some execution is currently going on.

⇒ the next $\boxed{\text{6-bytes}}$ of the program will be fetched.

Pipelining fails when there is a branch. so at that time the 6-Byte instructions, what had been fetched should flushed out.

→ µP assumes the program will go in a seavencial manner.



will the pipelining stops?

⇒ No, the pipeline will again start from instructions 9

☑ What are functions?

(1) Calculate $\overset{\text{20 bit}}{\text{physical}}$ address.

(2) prefetch instructions from memory.

(3) Manage the queue of 6Byte.

$\underset{2B}{\underline{\text{MOV}}}\quad \underset{2B}{\underbrace{\text{AX}}},\ \underset{2B}{\underbrace{\text{BX}}}$

$\underset{2B}{\underline{\text{MOV}}}\quad \underset{1B}{\underline{\text{AL}}},\ \underset{1B}{\underline{\text{BL}}}$

$\underset{2B}{\underline{\text{MOV}}}\ \underset{1B}{\underline{\text{AL}}},\ \underset{1B}{\underline{\text{4Ch}}},$

$\underset{2B}{\underline{\text{MOV}}}\quad \underset{2B}{\underline{\text{BX}}},\ \underset{2B}{\underline{\text{001Ch}}}$

$\underset{\text{registers.}}{\underline{\text{CS/DS/ES/SS}}} \Rightarrow$ segment

IP, SP, SI, DI ⇒ holds the offset address of the next instruction.

II Execution Unit

→ control system = decodes the instructions

⇒ we write, AND BL, CL
but what has come is opcode of AND BL, CL ⇒ 0111101111

⇒ Control system releases the control signal.

So, steps are :
1. fetched
2. decoded
3. control signals $\overline{RD}/\overline{WR}$ are released.
4. Execution

□ AX, BX, CX, DX

General purpose Register.

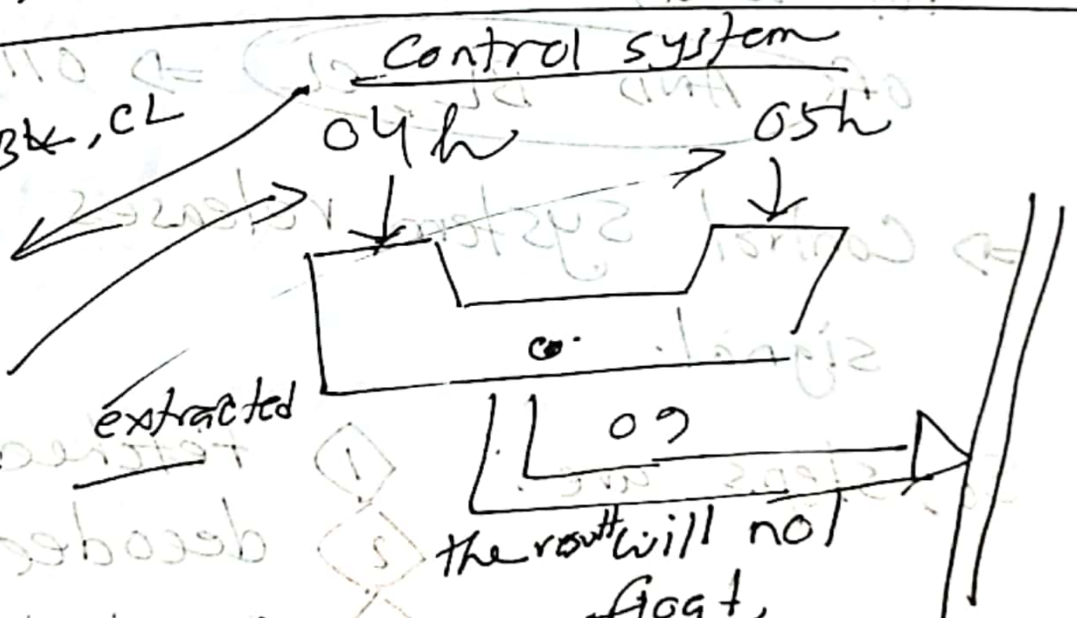
→ are assigned for ~~regi~~ a programmer.

X ⇒ means combination of two.

Mov CL, 34H ;
mov CH, 12H ;
MOV CX, 1234H ; → single instruction.

---

tells BL, CL to release the value 04 & 05 respectively.



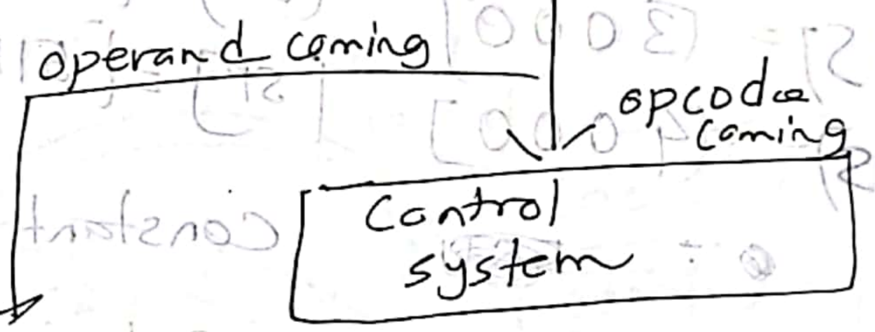
Mov BL, 04h
Mov CL, 05h
ADD BL, CL

the result will not float, since the control system will ~~tell~~ inform the BIU to capture the result of addition.

$\underbrace{\text{MOV BL}}_{\text{opcode}}, \underbrace{04h}_{\text{operand}}$

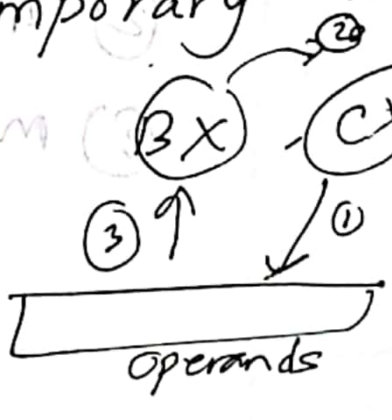
$\underbrace{\text{ADD BL , CL}}_{\text{opcode}}$

We decode the opcode
We add the operand.



operands ⇒ its a temporary register.
→ not available to the programmers
→ used by the up. only; for storing temporary values.

~~XCH~~ XCHG



done by MP.

II Flags → ~~gives the~~
→ has various flags
→ each flag gives some status about the current result.

Sec-1 / Quiz-1 16th march

Sec-2 / Quiz 15th March

[DS] = [10000]
[ES] = [20000]
[SS] = [3000]
[CS] = [4000]

[IP] = [3451h]
[SI] = [45AB h]
[DI] = [61AC h]
[SP] = [51FF h]

constant 09h.

If the queue's 2Byte is empty, BIU will refill the queue.

Q when it will do it?

~~Ans when~~

BIU does : ① calculate physical add
② Transfer memory fetch instructions
③ manage the queue 6 Byte.
?
bec. the biggest instruction can be of 6Byte.