Introduction of Computer

Bus structure: A bus is set of lines.
- One line can transfer 1 bits

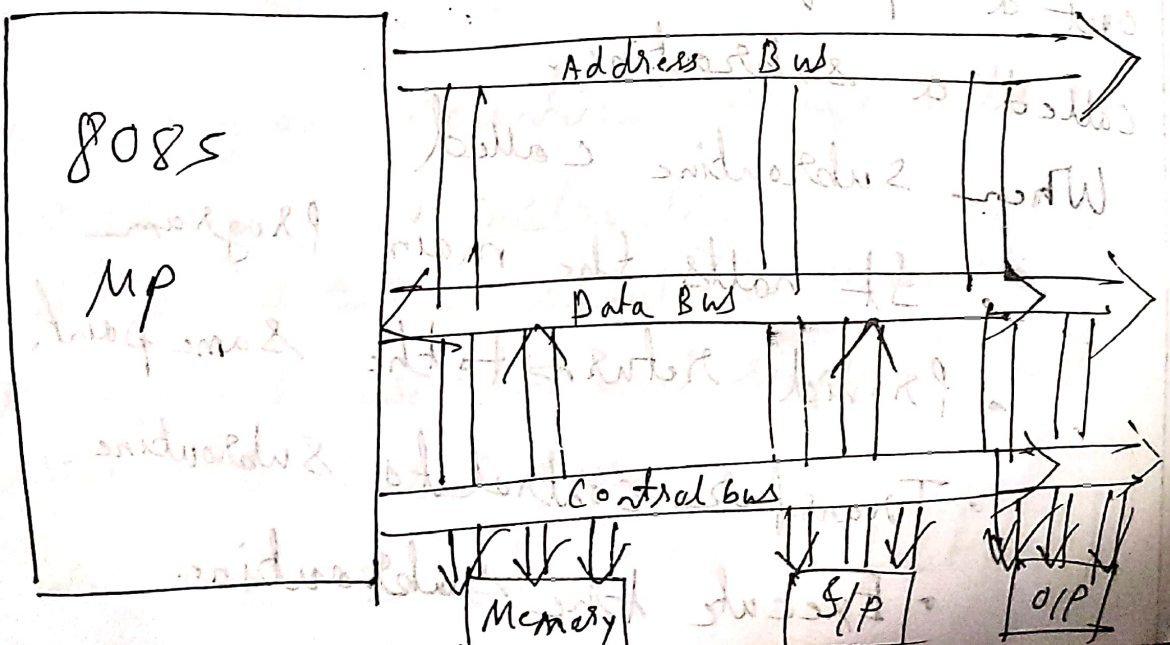
There is mainly 3 buses in CPU (C/I)

(i) Address Bus
- Unidirectional →
- if address bus is of n bits then total Max capacity will be 2^n bits. 16 bit address bus
  Memory location- $0 - 2^{16} - 1$

(ii) Data bus:
- Bidirectional

Input devices to µp →
µp to memory ←

(iii) Control bus: Controlling signal to peripheral device.



Basic I/O :=>

Input devices are used for taking input from user:

(i) Keyboard
(ii) Mouse
(iii) Joystick
(iv) Touch Screen
(v) Light Pen
(vi) Magnetic ink character Reader
(vii) OCR
(viii) Scanner

Output devices are used for producing output.

(i) Monitor
(ii) Printer
(iii) Speaker
(iv) Projector

Subroutine :=>

(assembly)

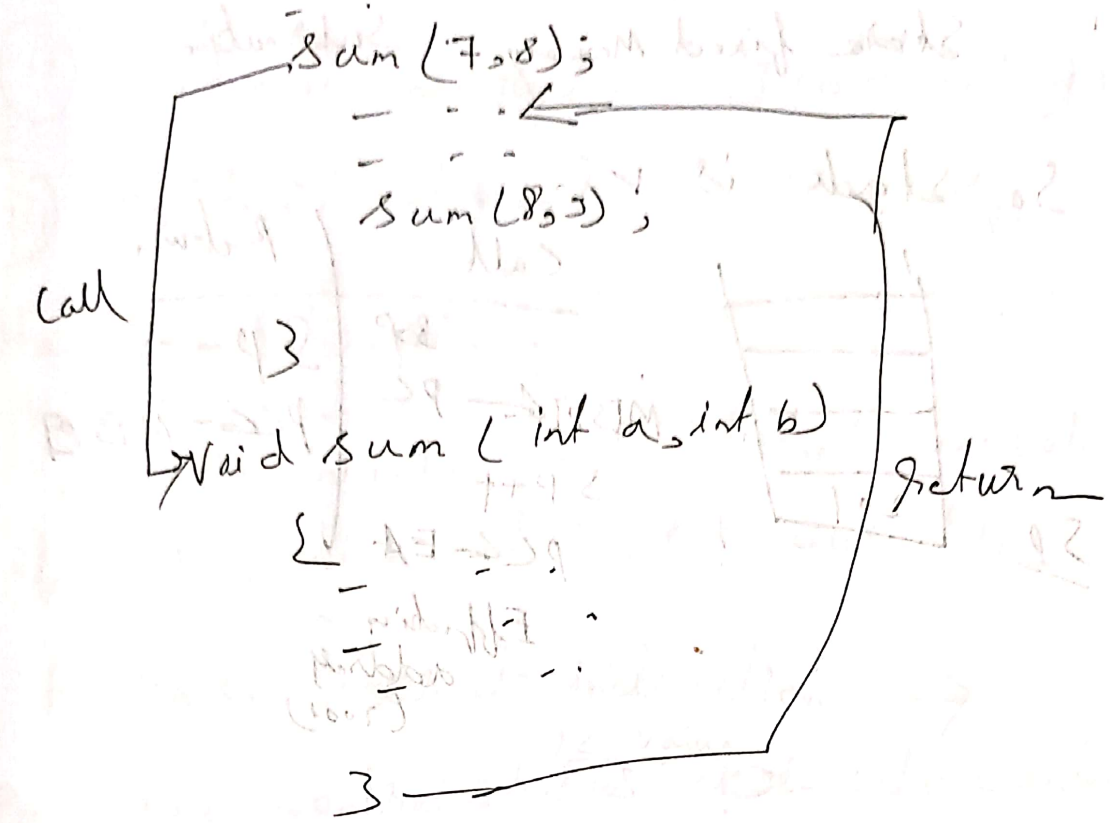
- Subroutine is like function (C/C++ Java)
- It is written and stored separately.

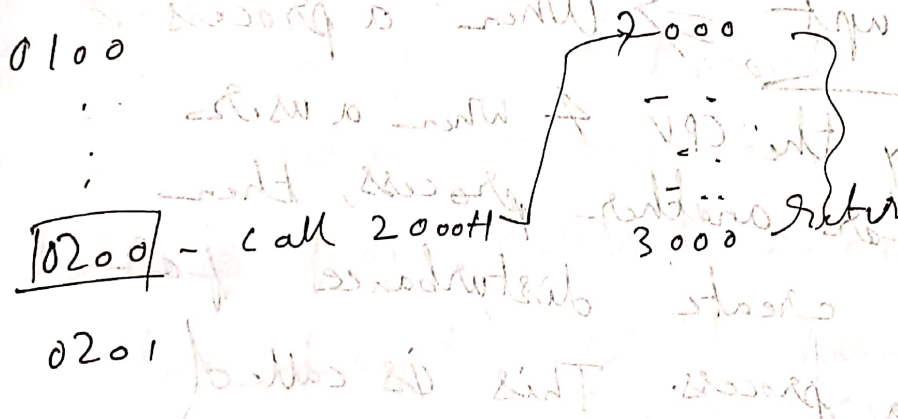
The block of instruction which carries out a specific well defined task is called a subroutine.

When subroutine called

- It halts the main Program
- Provid return to the same point.
- Transfer control to subroutine.
- Execute the subroutine.

Ex- main()
{
.
.
sum(7,8);
.
.
sum(8,3);
.
}
void sum(int a, int b)
{
.
.
}





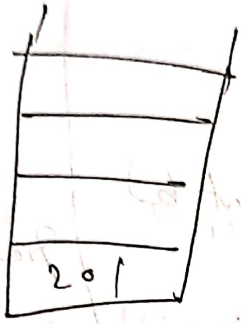
PC (Program counter) always hold the address of next instruction.
After return, PC should be in 201.
but we have to use the most simplest and correct way for return.
We use Stack (Last in fast out)

What we else can we do

- Store in Reg
- Store in subroutine
- Store fixed Memory

→ not worked if another subroutine is available in Subroutine

So Stack is ✓



Call	Ret-urn
SP	
M[SP] ← PC	SP - -
SP++	PC ← M[SP]
PC ← EA	

Effective address (200)

Interrupt ⇒ When a process is executed by the CPU & when a user request for another process, then this will create disturbance for the running process. This is called interrupt.

[Subroutine needs call but Interrupt happens automatically by user // Software hardware]

3 types of interrupt

(i) Internal Interrupt (divided by 0, Reg overflow

(ii) External Interrupt - (I/O devices + instruction

(iii) Software interrupt - (System call

Handling Interrupt: =>

CPU continously check interrupt pin value if it is 1 then interrupt occur Then

Finish the current instruction → Push the control status PC value onto control stack → Load the interrupt subroutine address on PC → Execute interrupt Service routine → Pops the process status from control stack.

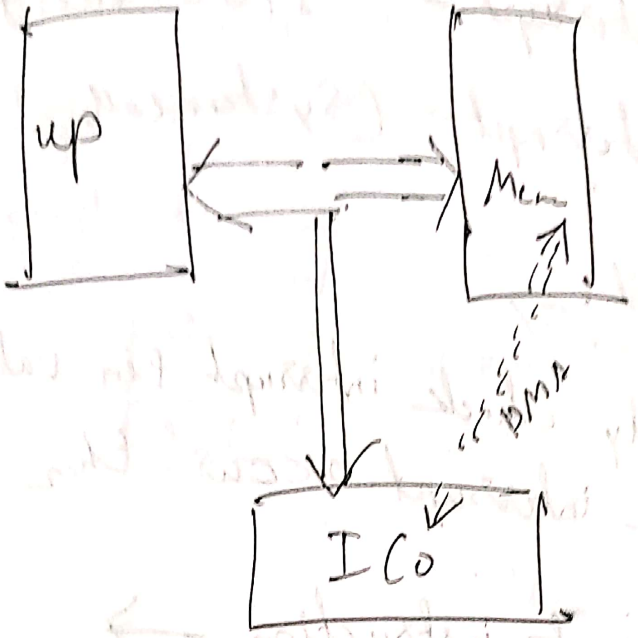
Vector interrupt — fixed address

X8 = Hexadecimal TRAP RST 5.5, 6.5, 7.5

non maskable 0030 H

Maskable interrupt — Interrupt that can be disabled (ignored)

DMA :→ DMA Stands for Direct Memory Access



Mem → up
up → mem
I/o → up
up → I/o
Mem → I/o ✗
I/o → Mem ✗

Transferring data between Memory and I/o without involvement of up is called DMA.

- It is very fast as up is not involved
- up is free for other work. (Sometimes, not for all)

DMAC (DMA controller) Control this data transfer.

DMAC takes the charge of bus master and Release the control signal.

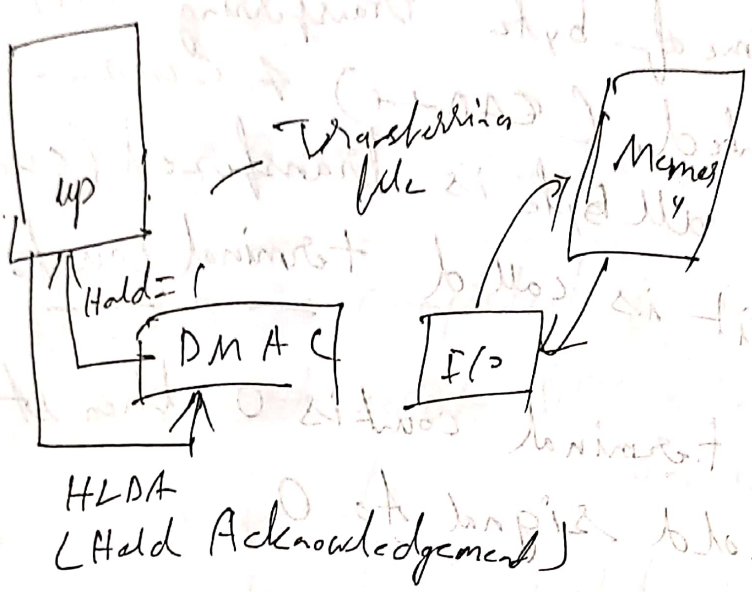
In times of up it takes & and then then send other side but DMAC will do

directly.

Why DMAC fast?

⇒ As DMAC is made exclusively for DMA transfer thats why it is preprogrammed. It does not need any instruction. So fast.

On the other side as up can do a lot of thing it needs instruction for everything. Th- ats why it is comparatively slow.



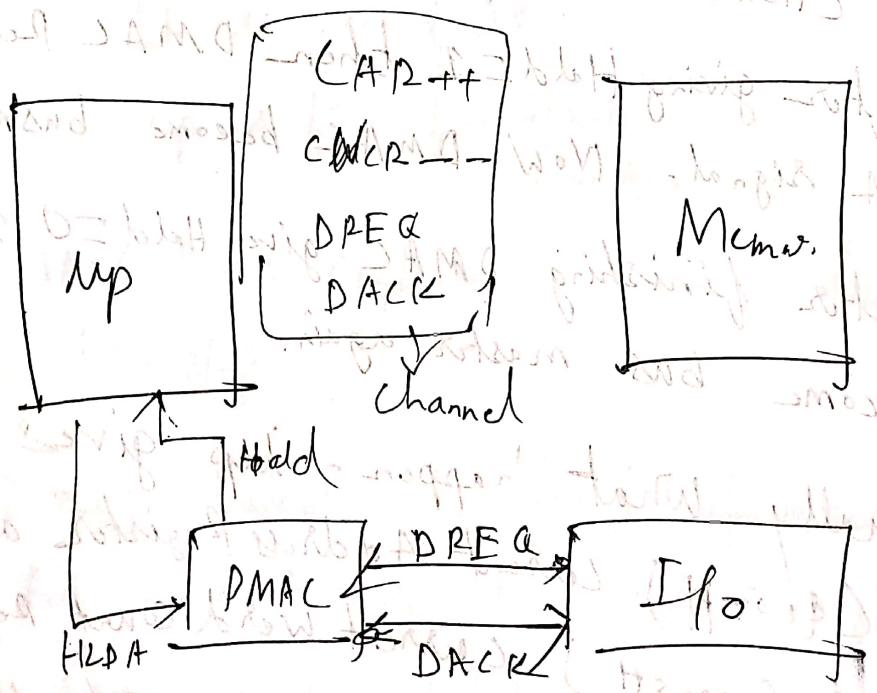
After giving Hold=1 then DMAC Received HLDA signal. Now DMAC become bus master. Then after finishing DMAC give Hold=0 & up become bus master again.

(*) Actually what happen - up gives CAR (4000H) current Address Register and CWCR (0005H) current word count Register Address 4000H, have to transfer 5 bytes.

First DMAC check whether I/O devices available / not. ~~If I/O available~~ it ~~gives~~ checks DREQ (Data Request) signal, if DREQ = 1 then its available. Then DMAC gives Hold, µp release bus & give HLDA. Before transfer Data DMAC gives DACK signal to I/O. So that I/O devices be ready to get the data in meantime.

- In time of byte transferring CAR is incremented. (CAR++) & CWCR--
  When all byte is transferred (CWCR = 0) it is called terminal count.

→ When terminal count is 0 then it gives hold signal to 0.



DMA has 4 channel means 4 I/O devices can be connected.

If more than 1 channel give request then priority comes to place

Channel 0 — High
" 1
" 2
" 3 — low

Transfer Mode —

(i) Block / Burst transfer :→ Untill the transfer is completed, DMAC remains bus master. DREQ become 1 only one time.

Advan. Super fast

Disad. Mp bc in hold state

(ii) Cycle stealing :→ After transfering 1 byte DMAC give control to Mp

- We can do everything in time of transferring.

We use this type of DMAC but in advanced way.

(*) May be we think we are using PC but Up becomes idle as in 1 sec it can done a billion operation.

Nowadays DMAC always look up what our Up is doing. if Up is idle then it tries to steal cycle more.

Demand transfer :=> I/o gives DREQ = 1, 0 (interchangably) to control the flow of data.

Printer wait to load Page, in this mean time DREQ = 0,

Transparent Mode :=> When Up is idle then DMAC do its work.

DMA has 3 Reg
- Starting address
- Word count

After completation The status of operation Processor get interrupt that it is completed.

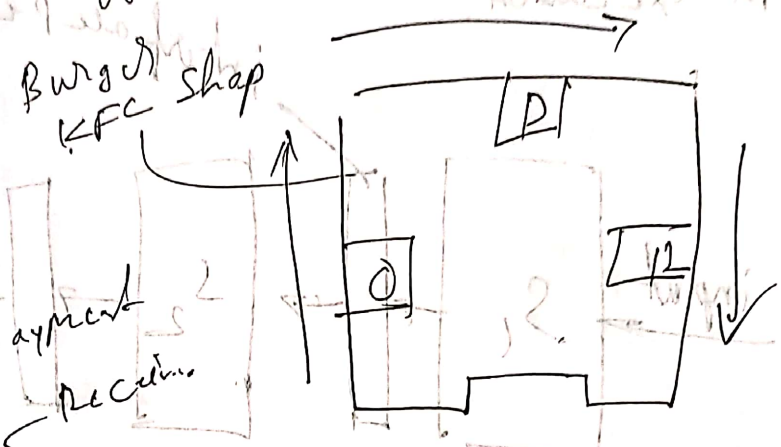
Pipelining :=>

We want to do everything very fast.

In order to do that

(i) ⊗ circuit change : to upgrade CPU it is very costly

(ii) Arranging CPU like a Pipe where there is different stages



order payment receiv...

Car1 - O P R

Car2 - - - O P R

Car3 - - - - O P R

non-pipelining

total 9 min

Car1 O P R

Car2 - O P R

Car3 - - O P R

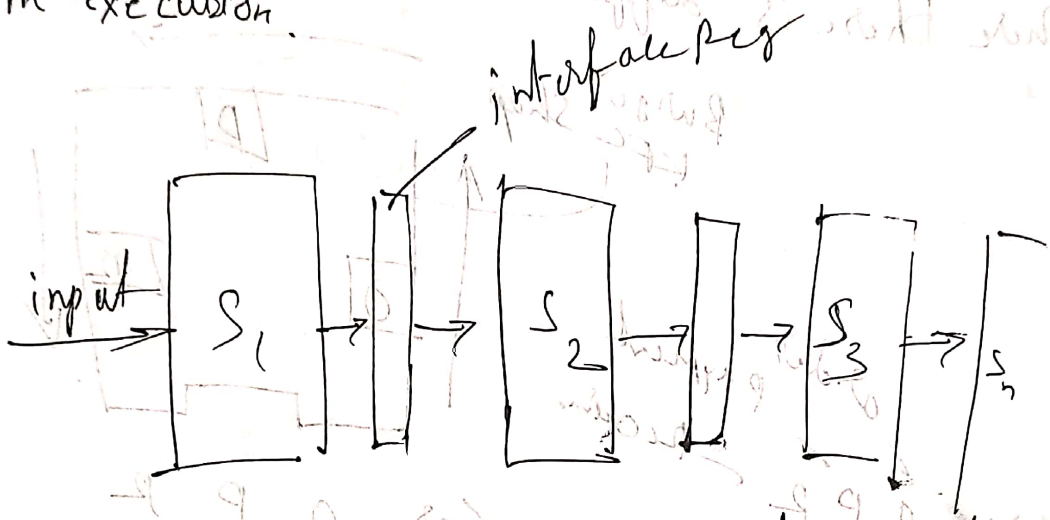
Pipelining

total - 5 min

Performance better

Pipeline is a process of arrangement of hardware elements of CPU such that overall performance is increased.

- Simultaneously more than one instruction takes place in pipelined processor.
- Multiple Processes are overlapped in execution.



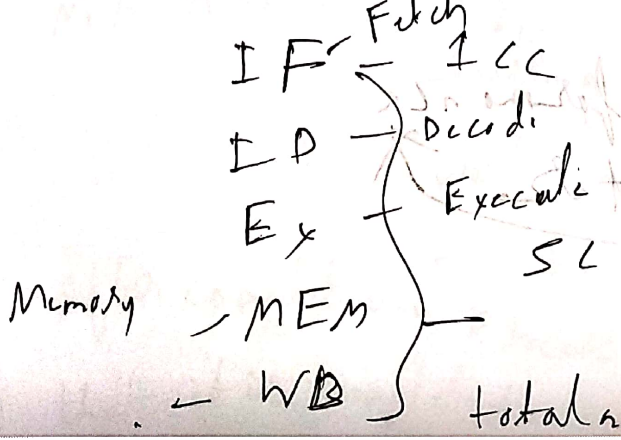
interface register store intermediate result.

All are added with a clock.

Non pipelining:

(X) Suppose there is 8 instruction.

And 1 instruction has 5 stages



Pipelining : There is 8 instruction and 5 stages then

Clock cycle

	1	2	3	4	5
S_1	I_1	I_2	I_3	I_4	I_5
S_2		I_1	I_2	I_3	I_4
S_3			I_1	I_2	I_3
S_4				I_1	I_2
S_5					I_1

In this way every instruction will completed

I_1 completed

no of stages $= k$

no of instruction $= n$

Total clock cycle $= n + k - 1$

1st instruction completed = (no of stages) clock cycle

$P = 8$ $I_1 (I_1 - I_8)$ 5 stages

$= 5 + 8 - 1 = 12$

Speed up = $\frac{NP}{p} = \frac{40}{12} = 3.33\ldots$

Efficiency / Utilization = $\frac{\text{How many blocks used}}{\text{Total block}}$

$= \frac{40}{60} = \frac{2}{3}$

Hazard In Pipelining ⇒

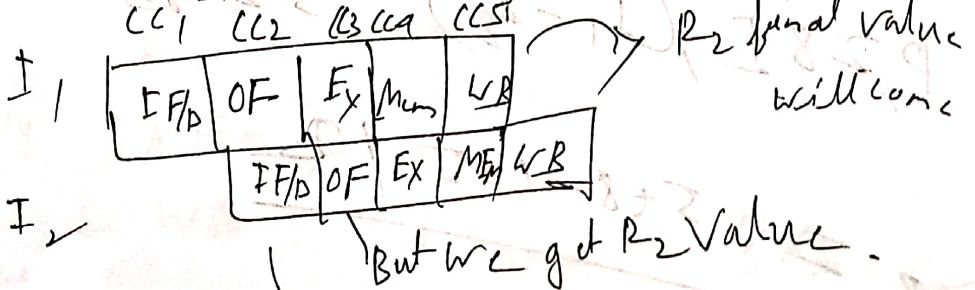
CPI = 1 → In every clock Pulse a instruction is completed

Hazards are the problem that come in achieving CPI = 1

- Data Hazard
  - RAW
  - WAR
  - WAW
- Structural Hazard
- Control Hazard

I_1 $R_2 \leftarrow R_1 + R_3$

I_2 $R_5 \leftarrow R_2 + R_4$



RAM →

Random Access Memory

• Volatile memory: Power supply off data also gone

• The tasks currently performed by CPU are stored in RAM

⊗ Ram is a part of CPU

• It is very fast

• It can be directly accessed by CPU

• It is part of Primary memory

Types of RAM:

(i) SRAM: Static RAM

Data remains in SRAM as long as there is power supply.

• It is also used as Cached Memory

• It doesn't have not to be refreshed.

• It is faster & Costly, consume more Power

(ii) DRAM → Dynamic RAM

• Data will be stored only when it is refreshed frequently

• The Main Memory of Most computers are DRAM

SRAM is mostly used in Specialized Applications

ROM ⇒ Read Only Memory

ROM data is permanently stored during creation of information.

It is called non volatile Memory.

Types –

(i) PROM – Programmable ROM

(ii) EPROM – Erasable PROM (UV erases ROM)

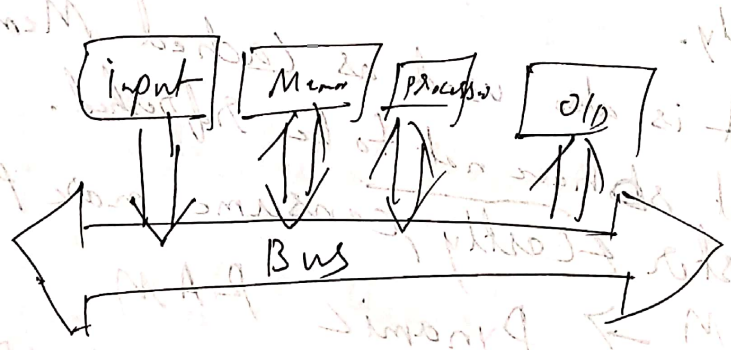
(iii) EEPROM – Electrically Erasable PROM

Bus structure two type –

(i) Single Bus structure

(ii) Multi bus structure

(i)



only 1 bus is used

Single bus does one transfer at a time So only 2 units can actively use the bus at a given time (input – Memory
Mem – output)

advantage
- Simple, low cost
- flexible to add device

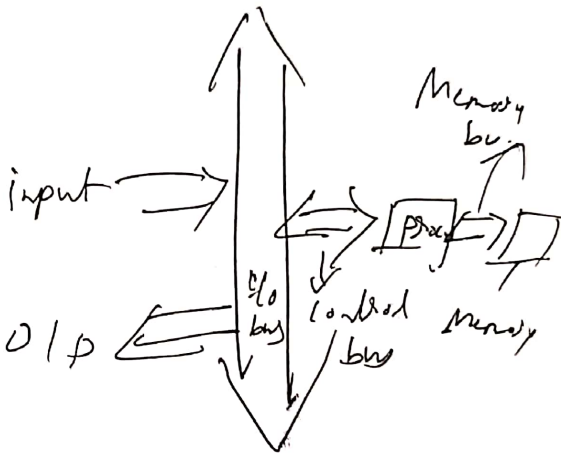
drawback
- Slow speed
- Efficient transfer Mechanism

needed → add buffer register: (first data come to buffer register then when CPU is idle operation is done)

(ii) Multibus :=

it improves performance input

achieve parallelism.



Disadvantage — High price

Advan — One bus for fetch instruction & other fetch data.