Page No.
Date
YOUVA

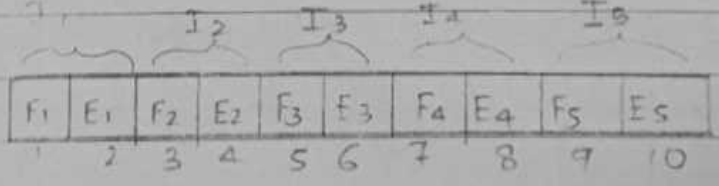
Pipelining -

- Overlapping of different stages of instructions is called pipelining.
- An instruction requires several steps which mainly involves - fetching, decoding and execution
- If these steps are performed one after another they will take long time.
- As processor becomes faster, several of these steps started getting overlapped resulting in faster processing.
- This is done by a mechanism called pipelining.

Two stages of pipelining :-
(8086 processor)

- Here the instruction process is divided into 2 stages - fetching and execution.
- Fetching of the next instruction takes place while current instruction is being executed.
- Hence 2 instructions are being processed at any point of time.

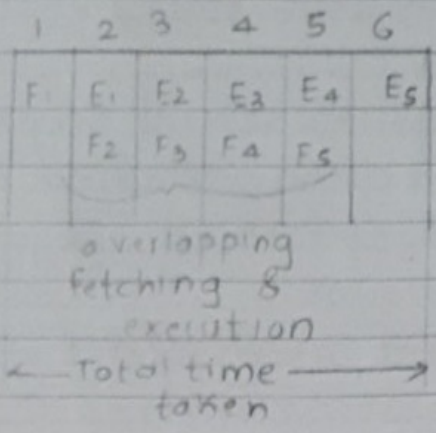
Non pipelining processor (eg · 8085)
Assume 5 instructions for 2 stage pipeline.



Total no. of clk cycles = $n \times k = 5 \times 2 = 10$

where n - no. of instructions
k - no. of stages.

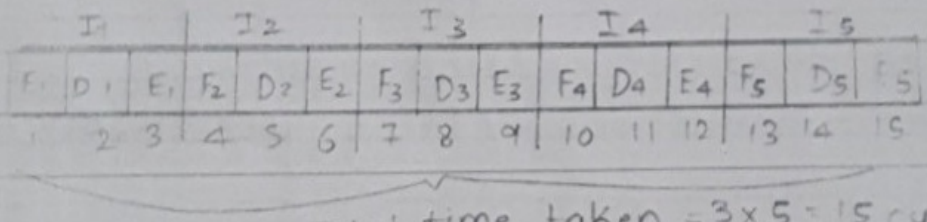
Pipelined processor (eg. 8086)



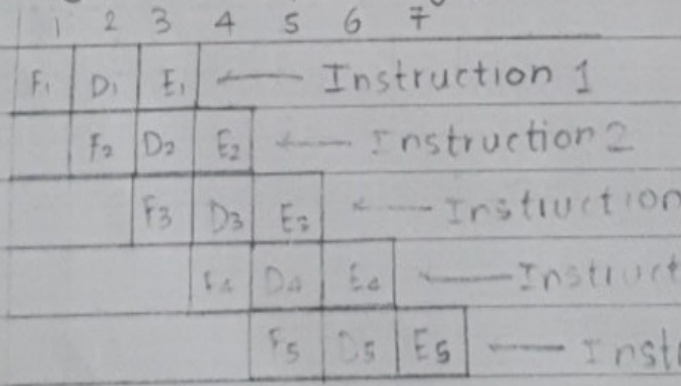
Total no. of clk cycles = n+k-1
= 5+2-1 = 6

3stage pipelining:-
(Fetch + decode + execute)

Non pipeline processor-



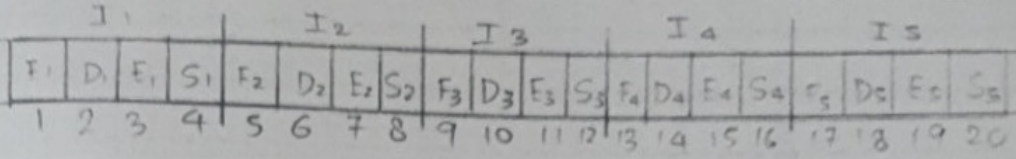
3 stages of pipelining [80386/ Arm processor]



Total time taken → k+n-1
= 3+5-1
= 7

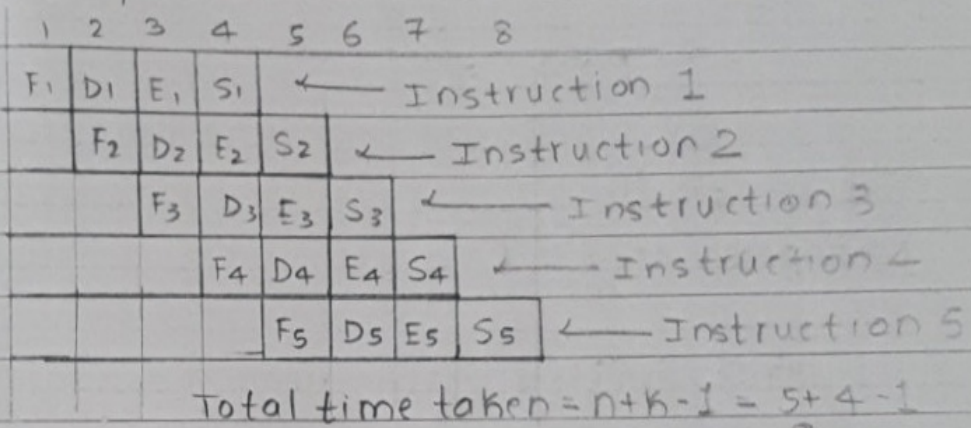
4 stage pipelining:-
(Fetch + decode + execute + store)

Non pipeline processor -



Total time taken = 4 × 5 = 20 cycles

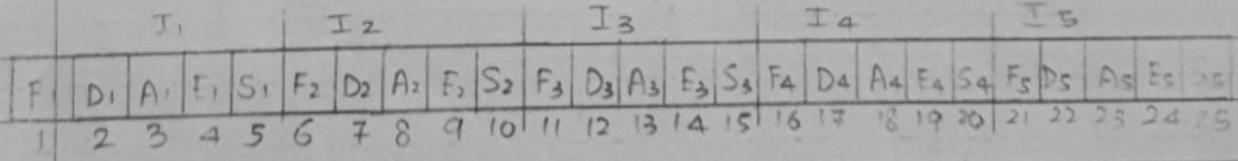
Pipeling processor -



Total time taken = n + k - 1 = 5 + 4 - 1
= 8

5 stage pipelining:-
(Fetch + decode + Address calculation + execute + store)

Non pipeline processor -



Total time taken = 5 × 5 = 25 cycles

	2	3	4	5	6	7	8	9
F_1	D_1	A_1	E_1	S_1				
	F_2	D_2	A_2	E_2	S_2			
		F_3	D_3	A_3	E_3	S_3		
			F_4	D_4	A_4	E_4	S_4	
				F_5	D_5	A_5	E_5	S_5

No. of cycles = $n+K-1 = 5+5-1$
$= 9$

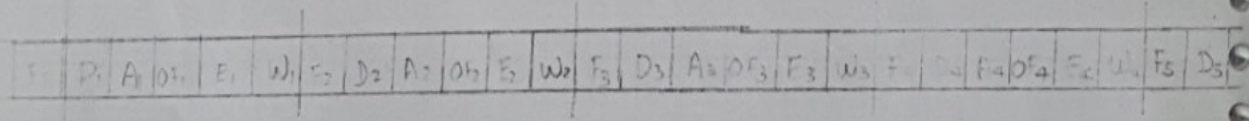
6 stages pipelining (eg. pentium pro):-

Stages

Instruction fetch + Instruction decode + Address generation + operand fetch
(IF) (ID) (AG) (OF)

Execute + Write Back
(Ex) (WB)

Non-pipeline process -



No of cycles = $nk = 5 \times 5 = 25$

1	2	3	4	5	6	7	8	9	10
F_1	D_1	A_1	OF_1	E_1	W_1				
	F_2	D_2	A_2	OF_2	E_2	W_2			
		F_3	D_3	A_3	OF_3	E_3	W_3		
			F_4	D_4	A_4	OF_4	E_4	W_4	
				F_5	D_5	A_5	OF_5	E_5	W_5

No. of cycles = $n + k - 1 = 5 + 6 - 1$
= 10

Advantages -

- The obvious advantge of pipelining is that it increases the performance as shown by the various examples above. Deeper the pipelining more is the level of parallelism.
Hence the processor becomes more faster.

Drawback / Hazard of pipelining:-

There are various hazards of pipelining which cause a drop in the performance of the processor. These hazards become even more prominent as the no. of pipelining stages increases.
They may occur due to the following reasons -

1) Data Hazard / Data dependancy hazard-

- Data Hazard is caused when the result of one instruction becomes the operand of the next instruction.
- Consider 2 instructions I_1 and I_2.

I_1 : INC [4000H]
I_2 : MOV BL [4000H]

- From above eg. it is clear that in I_2 BL should get the incremented value of location 4000H.
- But this can only happen once I_1 has completely finished execution and also has returned the result at 4000H.
- In a multistage pipeline I_2 may reach execution stage before I_1 has finished storing the result at location 4000H.
- Hence I_2 will get wrong value of data.
- This is called data dependency hazards.

Solution:-
- It is solved by inserting NOP (No operation) instructions between such data dependent instructions.

2] Control Hazard / Code Hazard -
- Pipeline assumes that the program will always flow in sequential manner.
- Hence it performs various stages of the forthcoming instructions before hand, while the current instruction is still being executed.
- While the programs are sequential most of the times, it is not true always.
- Sometimes branches do occur in program.
- In such an event all the forthcoming instructions that have been fetched, decoded, etc. have to be flushed or discarded and the process has to start all over again from the branch address.
- This causes pipeline bubbles i.e. time of processor is wasted.

eg. Branch instruction/JMP.

```
start
JMP DOWN
INC BL
MOV CL, BL
:
:
:
DOWN : DEC CH
```

- In this before completely executing the JMP instruction, processor has already started processing the next instructions.
- Hence the processor time is wasted.
- when JMP finished execution, it understands that it has to start processing from DOWN.
- The time wasted in execution of the next instructions is called pipeline bubbles.

Solution-

- The problem of branching is solved in higher processors by method called branch prediction algorithm.
- It was introduced by pentium processor.
- It relies on previous history of the instruction.
- It then makes a prediction whether the branch will be taken or not.
- Hence it puts correct instruction in pipelining.

3) structural hazards -

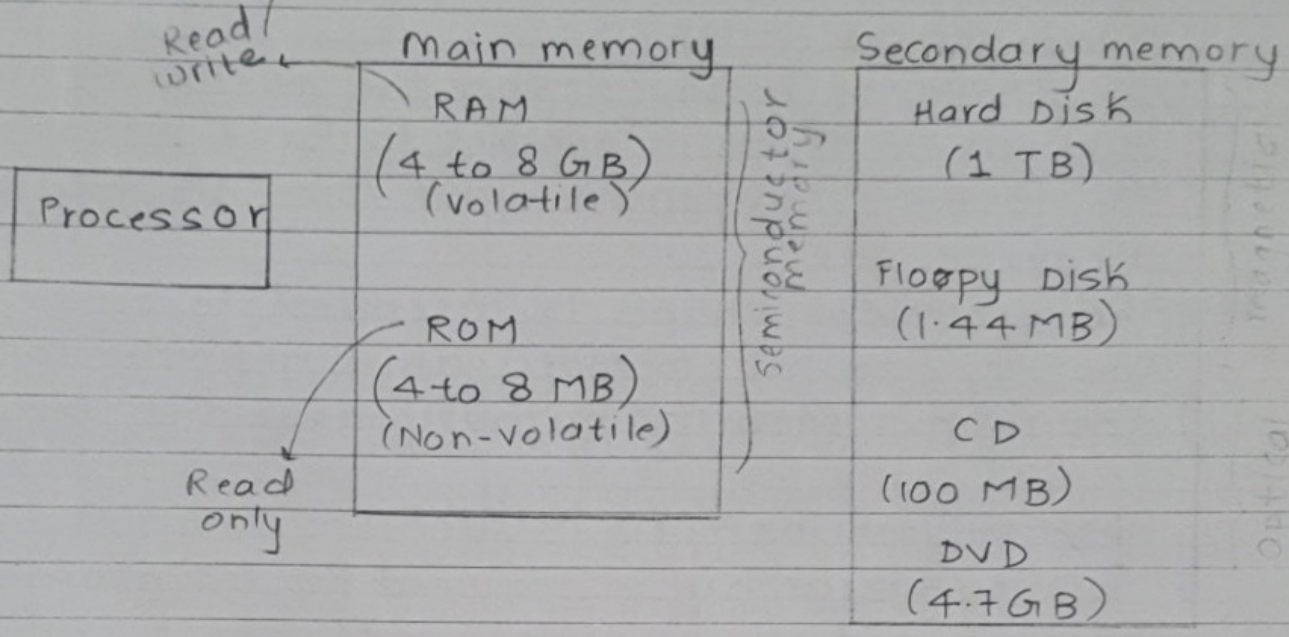
- structural hazards are caused by physical constraint in the architecture like buses.
- Even in the most basic form of pipelining, we want to execute one instruct^n and fetch the next one.
- Now as long as execut^n only involves registers, pipelining is possible.
- But if execut^n requires to read or write data from the memory then system bus will be used.
- which means fetching cannot take place at the same time.
- So the fetching unit will have to wait and hence pipeline bubble is caused.

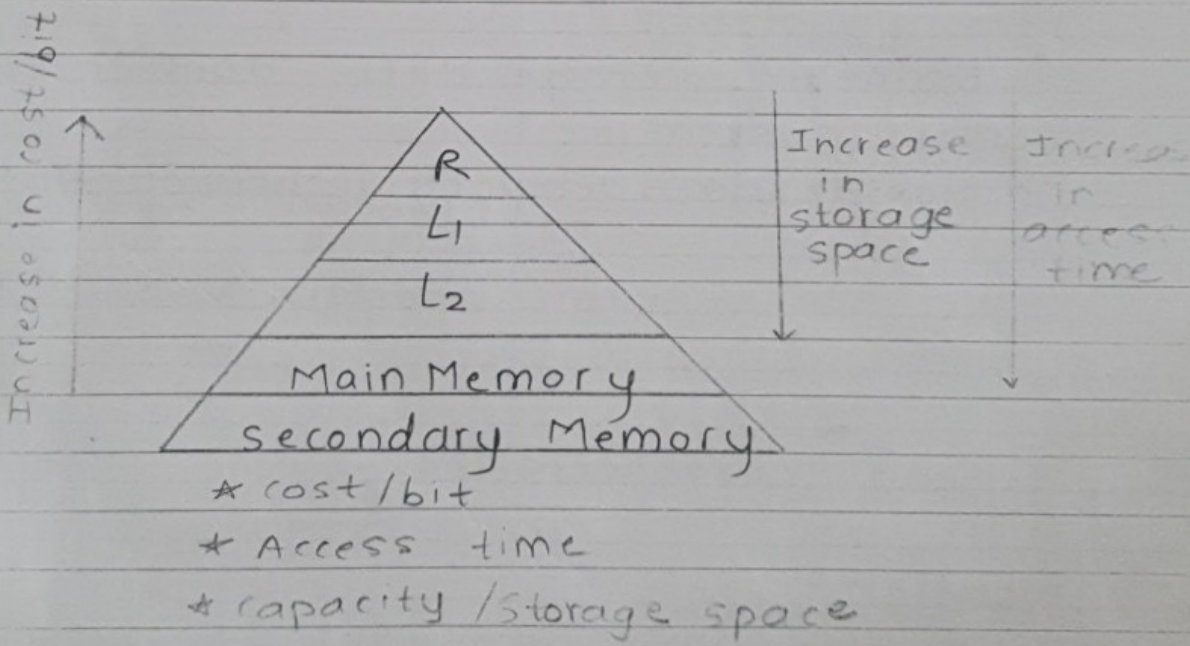
Solution:-

- Harvard architecture is the solution for this as it can have 2 simultaneous fetches.

4. Memory Organisation

Memory Hierarchy –



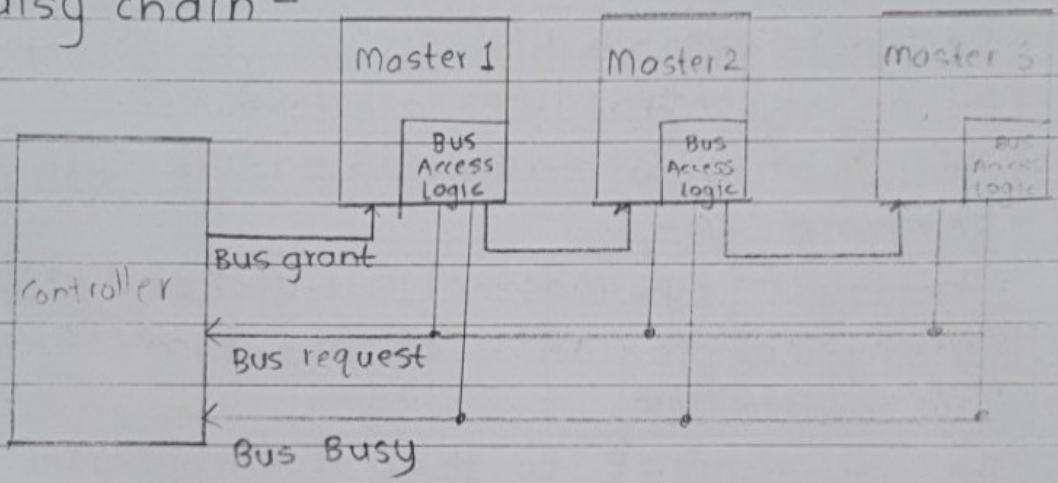


* cost/bit
* Access time
* capacity / Storage space

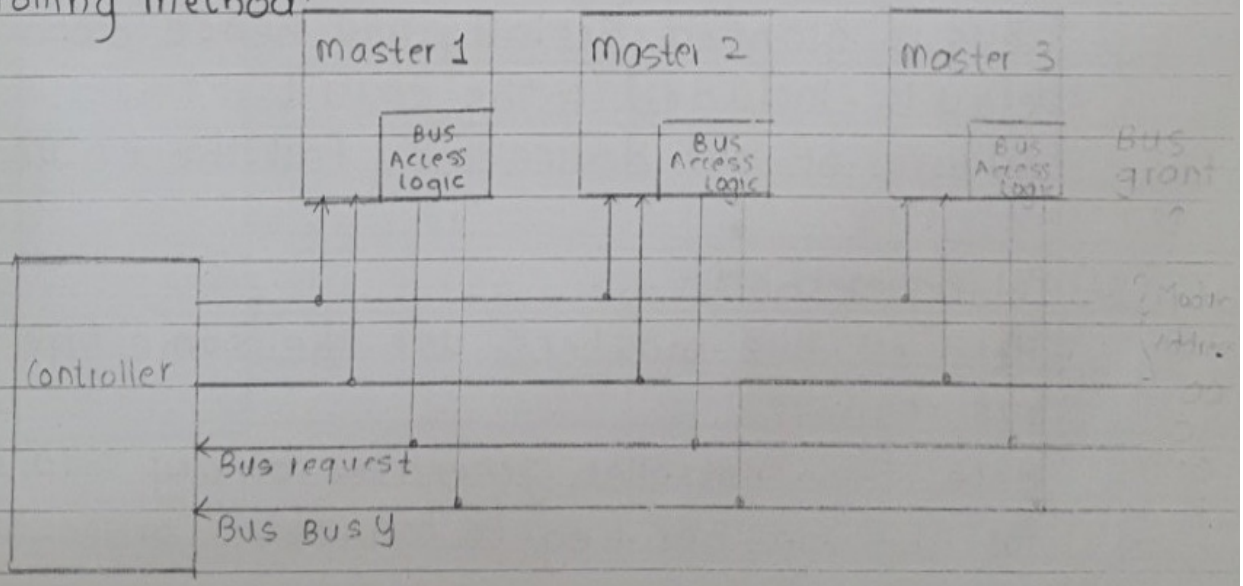
Bus Arbitration-

- In multiprocessor systems (like LAN) there are more than one processor that requires control of system bus at a time.
- Hence an appropriate priority resolving mechanism is required to decide which processor should get control of the bus.
- This is called bus contention.
- The following 3 methods are used to resolve bus contention —

1. Daisy chain -



2. Polling method:-



(1) Daisy chain -

- All the bus masters use the same line for bus request.
- If the bus busy line is inactive the controller gives bus grant signal.
- Bus grant signal is propogated serially through all the masters starting from the nearest one.
- The bus master which requires the system bus, stops the signal, activates the bus busy line and takes the control of system bus.

Advantage -

- Design is simple.
- The no. of control lines are less.
- Adding of new master is easy.

Disadvantage -

- Priority of bus master is rigid and depends on physical proximity of bus master with bus controller.
- The one nearest to the bus controller gets the highest priority.
- Bus is granted serially and hence propogation delay is included in the circuit.
- Failure of one device may fail the entire system.

(2) Polling method -

- Here all bus masters use the same line for bus request.
- Here the controller generates binary address for the master (eg. to connect 4 processors 2 address lines are required.)
- In respond to bus request the controller

polls the bus master by sending a sequence of bus master address on the address line.

eg. 00, 01, 10

- When a requesting master recognises its address it activates the bus busy line and takes the control of system bus.

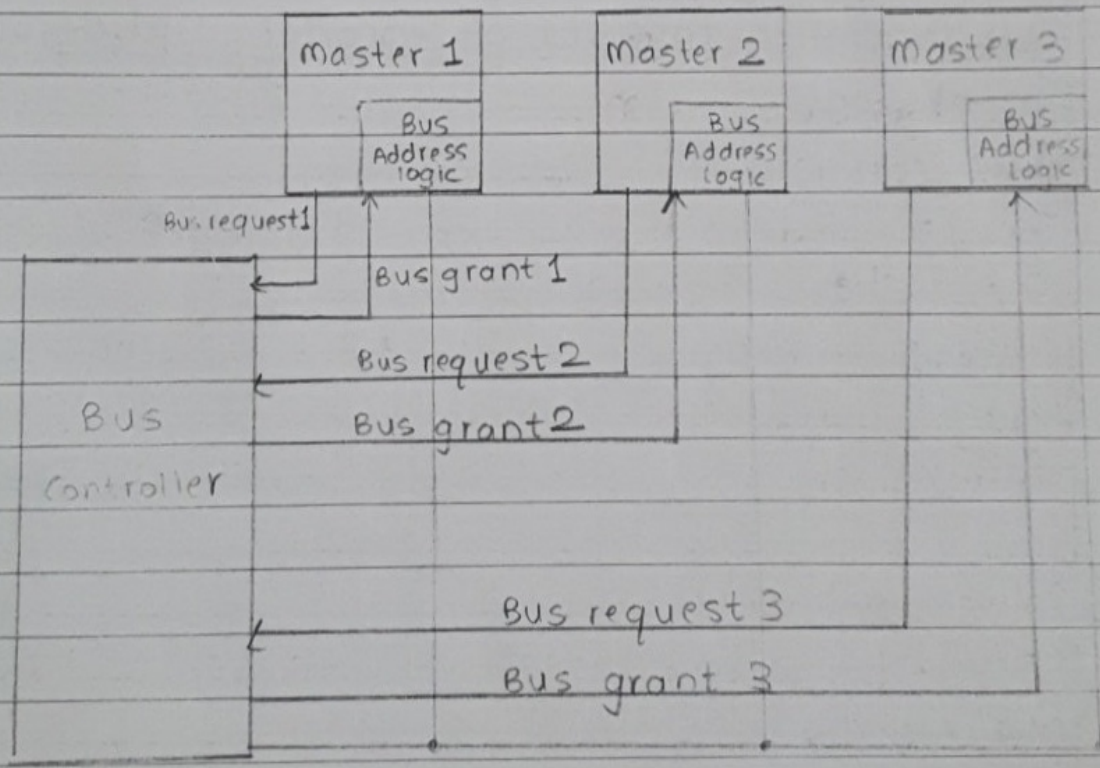
Advantage-

- The priority is flexible and can be easily changed by changing the polling sequence.
- ~~If~~ failure of one processor will not fail entire system.

Disadvantage-

- Adding more bus maters is difficult as it increases the no. of address lines

(3.) Independent method :-



- In this each master has a separate bus request and bus grant line.
- The bus busy line is common.
- This is the most efficient bus arbitration method.

Advantage-

- In this the Bus controller understands which master has requested.
- Time required is very less as the controller directly grants request to the master. It is independent of no. of processo

Disadvantage-

- In case of more no. of processors, hardware required will increase. Hence connecting large no of bus masters is difficult
- 2n lines are required for n processors.
- Priority of masters is pre-defined hence if 2 simultaneous requests arrive, control is given acc. to priority
- The controller consists of encoder and decoder logic.

Flyn's Classification:-

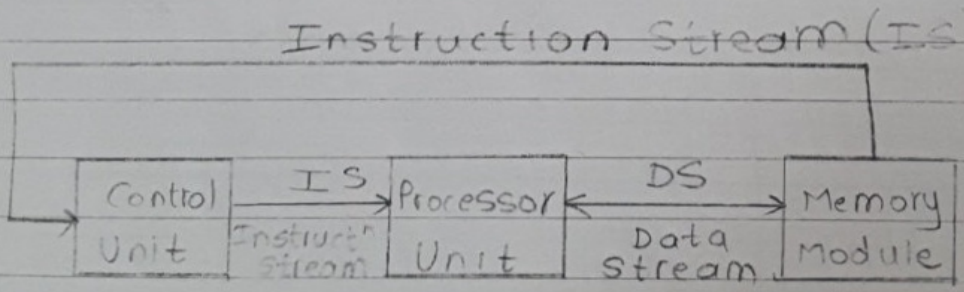
In 1966 Michelle Flyn has made an informal and widely used classification of processor parallelism based on no. of simultaneous instructions and data stream seen by proces during program execution.

- Let M_I and M_D denote the minimum no. o instructions and data streams respectively.
- The classification made by Flyn divided computers into 4 major groups based on values M_I and M_D.

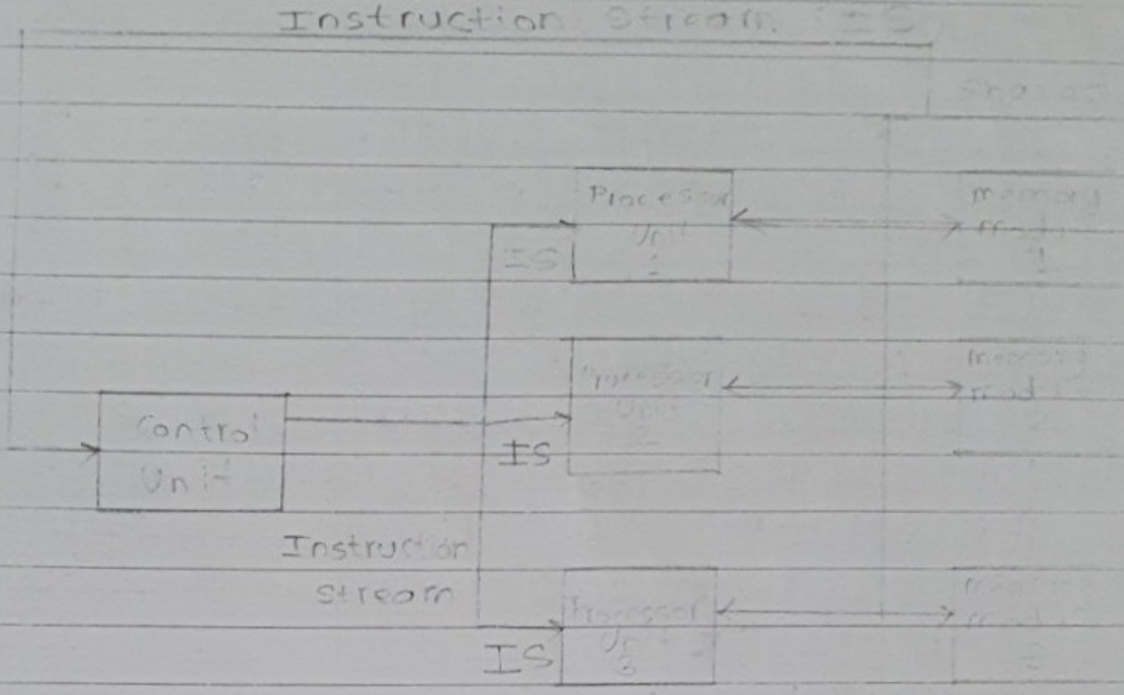
1] Single Instruction, Single Data Stream
2] Single Instruction Multiple Data Stream
3] Multiple Instruction stream Single Data Stre
4] Multiple Instruction multiple data stream.

1] SISD -

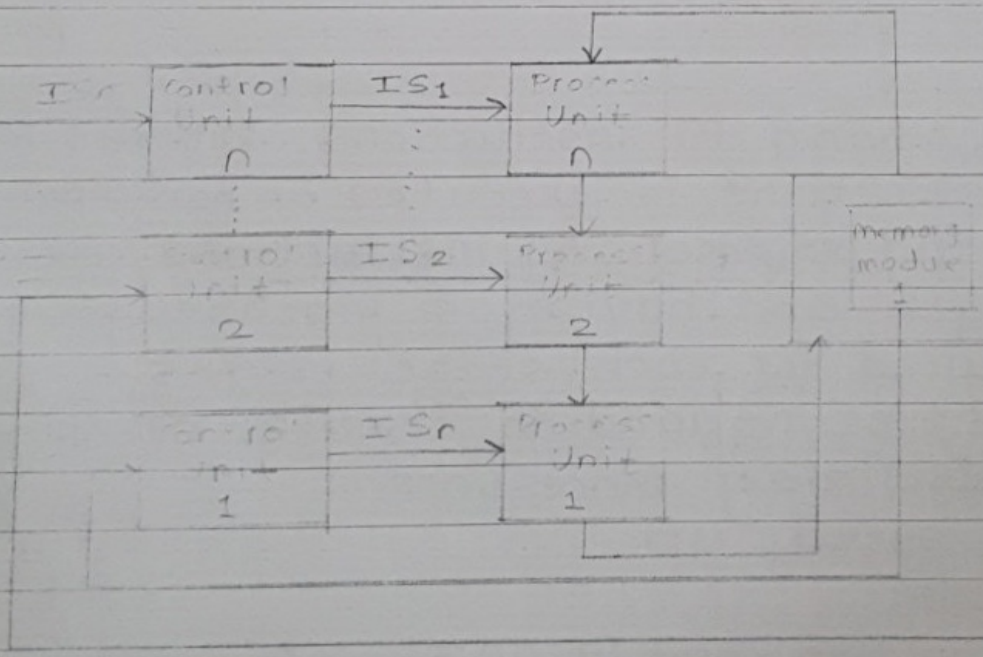
$M_I = M_D = 1$



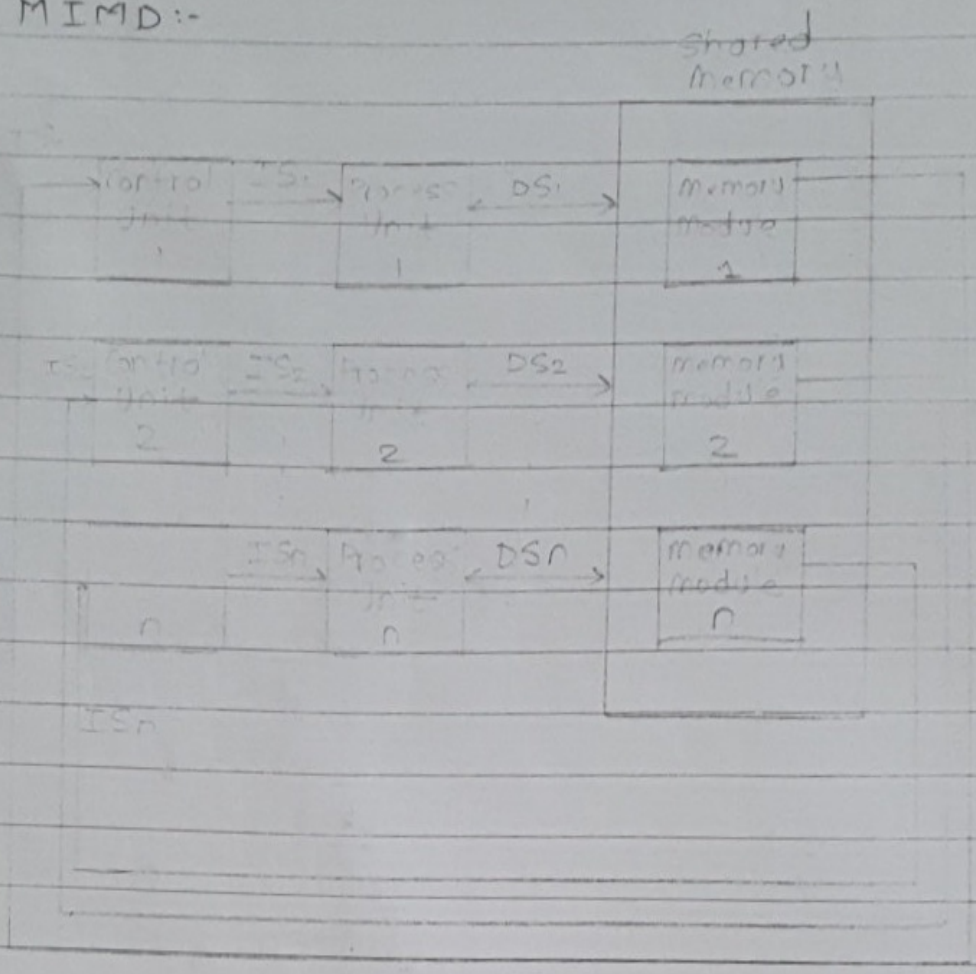
2.] SIMD :-



3.] MISD -



4] MIMD:-



Q] A program having 10 instructions without branch or call instruction is executed on non pipelining and pipeling processor. All instructions are of same length and having 4 pipeline stages. Time required for each stage is 1ns.

- Calculate time required to execute on pipeling and non pipelining processors.
- Calculate speed up.

Soln:-

Non-pipe lining = $n \times k$

$= 4 \times 10$

Time = 40×1

Time = 40 ns

Pipelining = n+k-1
= 4+10-1
= 13
Time = 13 x 1 = 13ns

Q.] Consider a cache memory of 16 words. Each block consists of 4 words. Size of main memory is 256 bytes. Draw associative mapping and calculate tag and word size.

Soln:-

tag

$\frac{256}{4}$ = No. of blocks