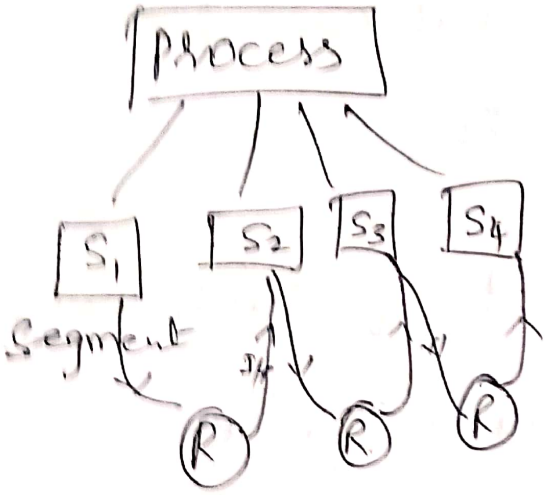
UNIT-IV

Pipelining: A process is divided into several sub-operations



We can Execute all the segments Concurrently

Why it is called Pipelining?

→ Passing information from $S_1 \rightarrow S_2$

Ex $A_i * B_i + C_i$ for $i = 1, \ldots 7$

$R_1 \leftarrow A_i$, $R_2 \leftarrow B_i$ ⇒ Segment 1

$R_3 \leftarrow R_1 * R_2$, $R_4 \leftarrow C_i$ ⇒ Segment 2

$R_5 \leftarrow R_3 + R_4$ ⇒ Segment 3

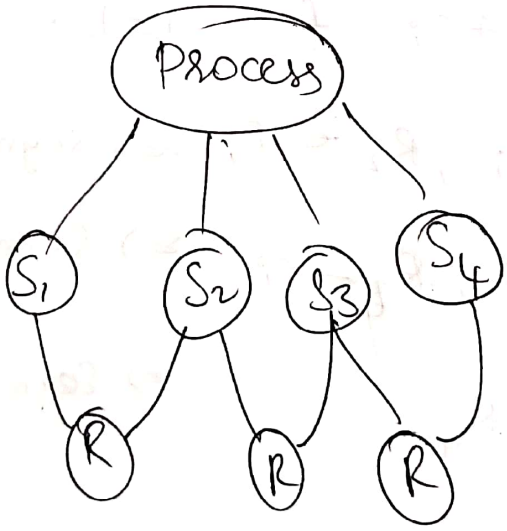
Pipelining

→ A process is divided into several sub operations

Where each suboperation is represented with segments.

→ Here output of each segment is stored in a Registes

→ The output of each segment is passed to as an input to Next segment

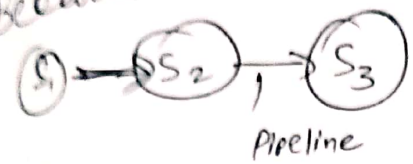


→ Here all the segments Executes Independently

→ We can Execute all the segments Concurrently

→ Why it is called pipelining.

Because we are transferring information



Ex:

$A_i * B_i + C_i$ for $i = 1, 2, \ldots 7$

(1) $A_i * B_i$

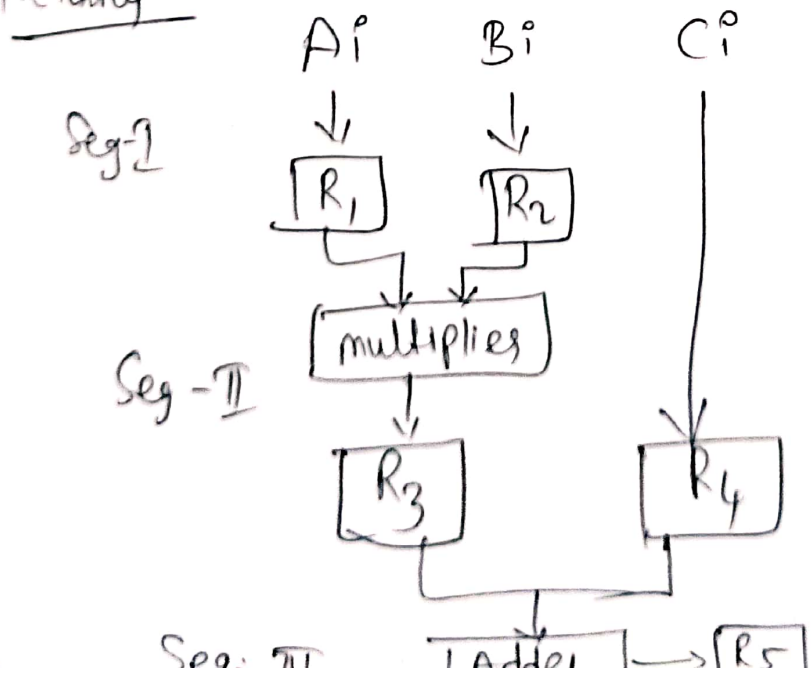
(2) $A_i * B_i + C_i$

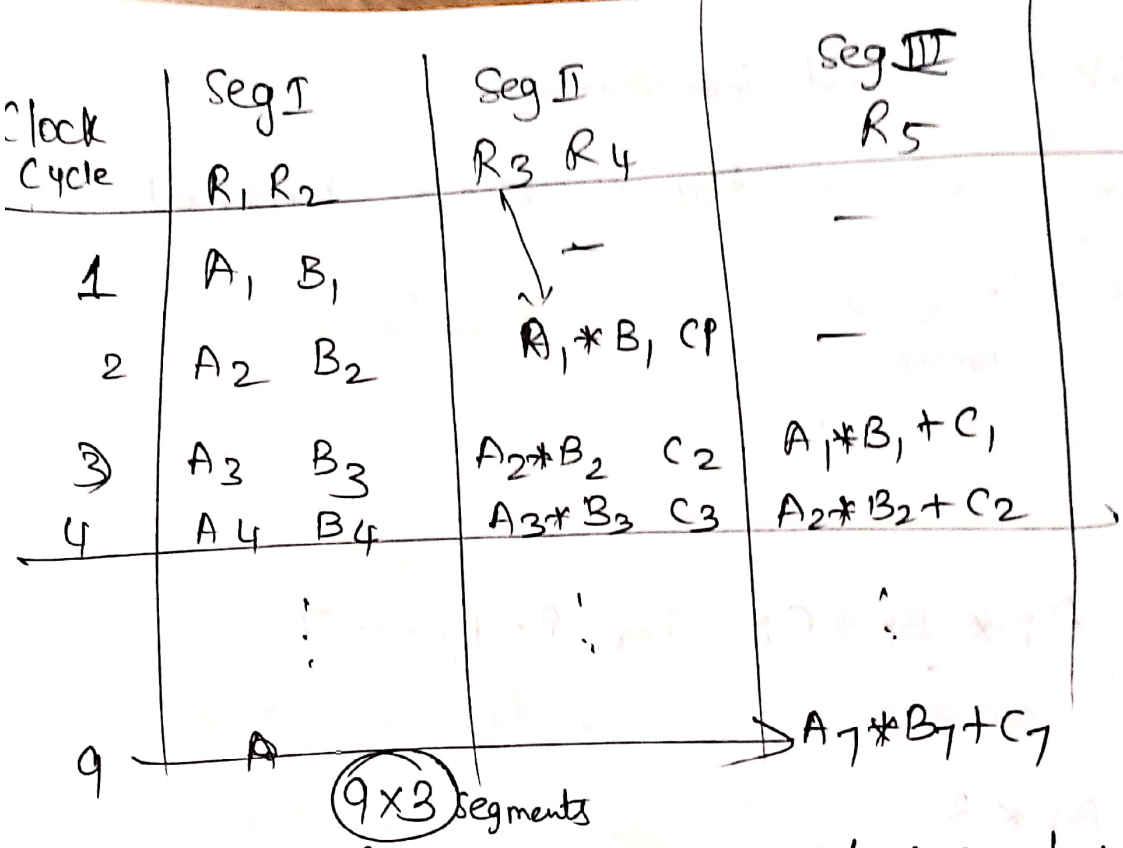
Segment I ⇒ $R_1 \leftarrow A_i$, $R_2 \leftarrow B_i$

Segment II ⇒ $R_3 \leftarrow R_1 * R_2$, $R_4 \leftarrow C_i$

Segment III ⇒ $R_5 \leftarrow R_3 + R_4$

Pictorial





It takes 27 clock cycles to give the Result

3 clock cycles for one output

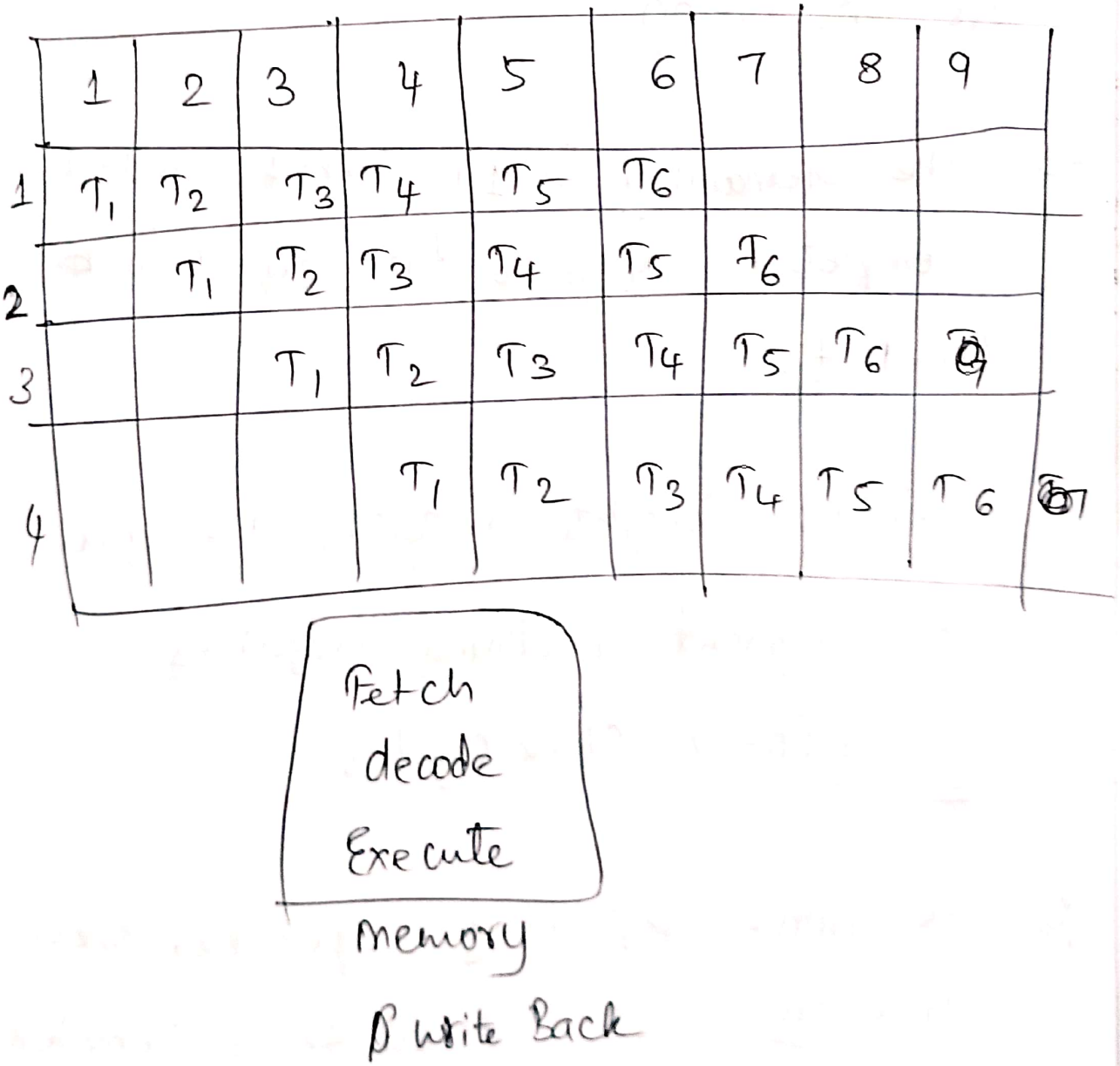
$\Rightarrow$ 9 x 3 = 27 clock cycles

→ To Increase performance

→ To decrease Time

Performance of pipelining

The Space-time diagram of a four-segment pipeline is demonstrated in following figure



Performance of pipelining

→ Where a (k) Segment pipeline with a clock cycle time (tp) is used to Execute n tasks

→ The first task T_1 requires a time Equal to (k)(tp) ⇒ $k \cdot t_p$ to complete its operation.

→ The remaining (n-1) tasks will be Completed after a time Equal to

$(n-1)\,t_p$

∴ To complete n tasks Using a k - Segment pipelining requires

$k + (n-1)$ clock cycles

for a non - pipelining operation takes time t_n (i.e) total task completes at $n \cdot t_n$ ⇒ $n \times t_n$

Speed up:

The Speed up of a pipeline processing over an Equivalent non-pipeline processing is defined by the Ratio

$$S = \frac{n\,t_n}{(k+n-1)\,t_p} \Rightarrow \frac{n \cdot k\,t_p}{(k+n-1)\,t_p} \Rightarrow \frac{n \cdot k}{(k+n-1)}$$

As the Numbes of tasks increase the speedup becomes

$$S = \frac{t_n}{t_p}$$

∵ If we Assume that the time it takes to process a task is the same in the pipeline and non-pipeline Circuits

(i,e) $t_n = k\,t_p$

The speed up reduces to

$$S = \frac{k\,t_p}{t_p} = k \Rightarrow \boxed{S = k}$$

This shows that the theoretical Maximum speedup that a pipeline can provide is k, where k is the Number of Segments in the pipeline.

Ex:

Cycle time $(t_p) = 20ns$

No. of segments $\boxed{k = 4}$

Number of tasks $\boxed{n = 100}$

The pipelining system will take

$k + (n-1)\, t_p \Rightarrow 4 + (100-1)\, 20ns$

$\Rightarrow 2060\ ns$

* Assuming $t_n = k\, t_p \Rightarrow 4 * 20 = 80ns$

A. Non pipelining system requires $n\, k\, t_p \Rightarrow 100 \times 80 \times 20$

$\Rightarrow 8000ns$

Speedup Ratio is $8000/2060 = 3.88$

Throughput :

Throughput of a processor is the rate at which Operations get Executed

→ Latency is the Amount of time that a single Operation takes to Execute

→ In non-pipelined Computer

throughput = 1 / latency

→ In pipelined Computer

throughput > 1 / latency

Since Execution of one Instruction is overlapped.

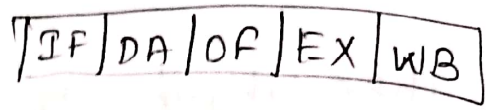
$$\text{Throughput} = \frac{\text{No. of Instructions}}{\text{Total time to complete Instructions}}$$

$$\therefore \text{Throughput} = \frac{n}{(k+n-1)\,t_p} \quad \text{for pipeline}$$

$$= \frac{1}{t_n} \longrightarrow \text{for non-}$$

* Pipeline Hazards

Instruction Pipeline



→ The Execution of Instructions blocked/ Stopped due to some problem in the Ppeline is Called pipeline Hazard.

* Situation that prevent the next instruction from being Executing during its designated clock cycle.

⇒ (1) Structural Hazard / Resource Conflict
(2) Data Hazard / Data dependency
(3) Control Hazard / Branch Difficulty

(1) Structural Hazard:

2 different segments try to use Same Resource at same time

(or)

If a Resource is needed by 2 Instructions simultaneously

(1) Ex

MUL => 2 cycles
ADD => 1 cycle

MUL	IF DA OF EX EX WB
ADD	IF DA OF _ EX

Structural Hazard
Can't Execute two Instructions at a time

Solution : (1) Increase Number of resources
(2) Wait

(2) Data Hazard :

Result of an instruction is used as input in next

Ex

$i = R_1 \leftarrow R_2 + R_3$
$i+1 = R_5 \leftarrow R_1 * R_4$

IF	DA	OF	EX	WB	
	IF	DA	OF	EX	WB

(X)

* Here in the first instruction until the value of R_1 is updated in the memory by WB (WriteBack).

→ The value of R_1 in second instruction can't be fetched (i,e) (OF : Operand fetch)

Solution : A general instruction pipeline cannot detect data dependency

→ Then solution of Data dependency is provided by compiler. (i,e) delayed Load

Ex $R_1 \leftarrow R_2 + R_3$

↓ ≡ | Independent Instructions Executed

$R_5 \leftarrow R_1 + R_4$

wait until Instruction get Executed

Ex $R_1 \leftarrow R_2 + R_3$ → IF | DA | OF | EX | WB

Independent Instruction ≡ ↓ delayed

IF | DA | OF | EX | WB

IF | DA | OF | EX | WB

IF | DA | OF | EX | WB

$R_5 \leftarrow R_1 + R_4$ →

Parallel Processing

* Parallel processing is a term used to denote a large class of techniques that are used to provide simultaneous data-processing tasks for the purpose of increasing the computational speed of computer system.

* The purpose of parallel processing is to Speedup the computer processing capability and increase its throughput

that the Amount of processing that can be accomplished during a given interval of time

* The Amount of hardware increases with parallel processing and with it, the Cost of the system increases.

* parallel processing can be viewed from various levels of complexity

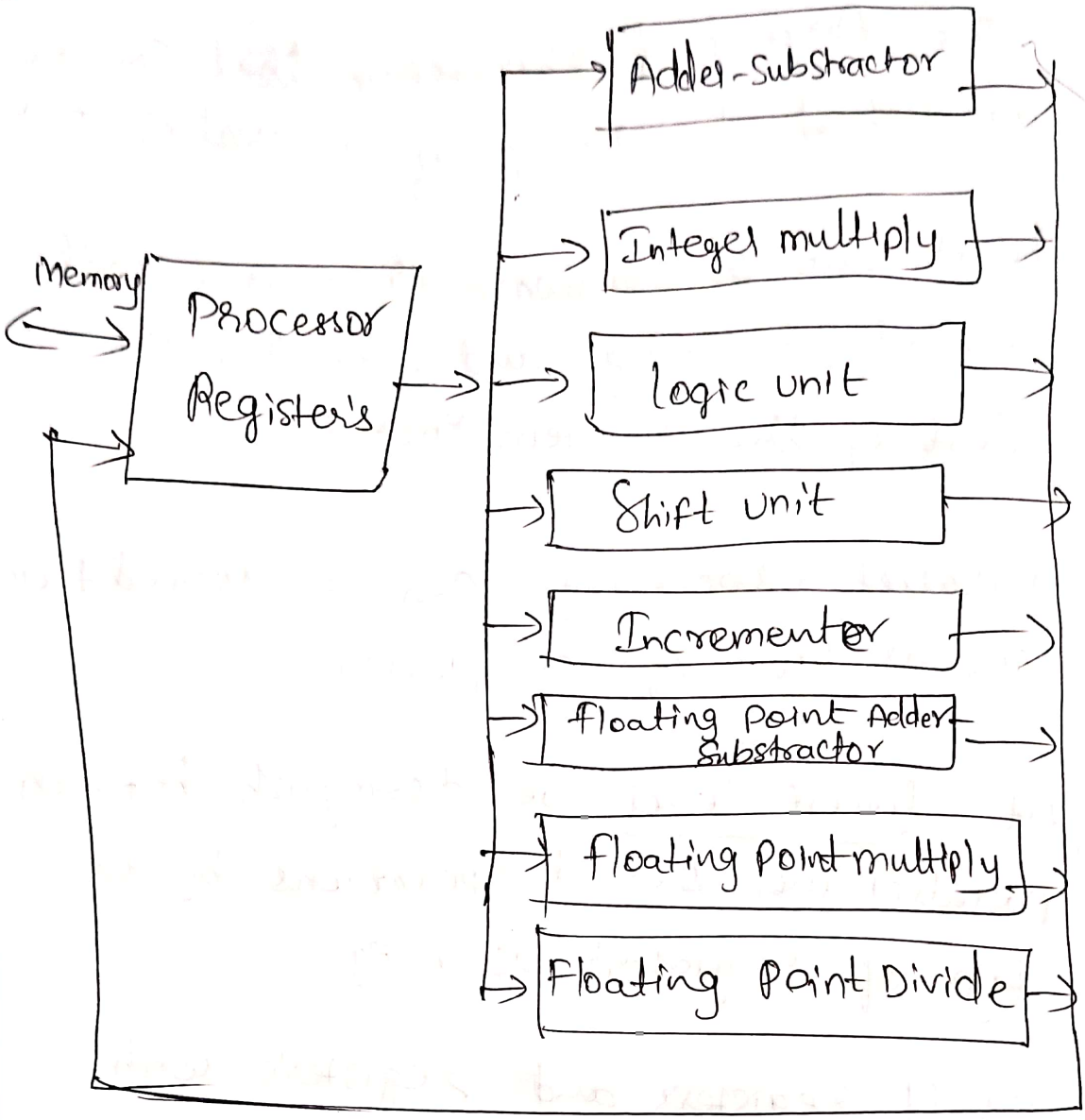
6

1) At lowest level we distinguish between parallel and serial Operations by the type of registers used eg.

Shift registers and registers with parallel load.

2) At the higher level, it can be achieved by having a multiplicity of functional units that perform identical (or) different operations simultaneously

* A multifunctional organization is usually associated with complex control unit to coordinate all the activities among various components



In the Above figure we can see that the data stored in the processor register's is being sent to separate devices basing on the operation needed on the data

→ If the data inside the processor register is requesting for an arithmetic operation then the data will be sent to the arithmetic unit and if in the same time another data is requested in the logic unit then the data will be sent

to logic unit. for logical operations.

* Now in the same time both arithmetic operations and logical operations are Executing in parallel.

This is called parallel processing

* The Computers are Classified into (4) Types based on the

(1) Instruction streams and

(2) Data Stream

These (4) Classifications are called as ~~Instruction stream~~

Flynn's Classification of Computers.

(1) Instruction Stream: The sequence of Instructions read from the memory is called as an Instruction Stream

(2) Data Stream: The Operations performed on the data in the processor are Called as Data Stream

M.J. Flynn's Classifications

M.J. flynn Considers the organisation of a Computer System by the Number of instructions and data items that are manipulated Simultaneously.

1. Single Instruction Stream, Single data Stream (SISD)
2. Single Instruction Stream, Multiple data Stream (SIMD)
3. Multiple Instruction Stream, Single data Stream (MISD)
4. Multiple Instruction Stream, Multiple data Stream (MIMD)

1. SISD

→ Represents the organization of a single computer Containing a control unit, a processor unit, and a memory unit.

→ Instructions are Executed Sequentially and the system may (or) may not have internal parallel processing Capabilities.

→ Parallel processing may be Achieved by means of multiple functional units (or) by pipeline processing

SIMD :

→ Represents an organization that includes many processing units under the Supervision of a common control unit

→ All processors receive the same instruction from the control unit but operate on different data items.

→ The shared memory unit must contain multiple modules so that it can communicate with all the processors simultaneously

~~MIMD~~ MISD :

MISD structure is only of theoretical interest since NO practical system has been constructed using this organisation.

MIMD : This organization refers to a computer system capable of processing several programs at the same time.

Ex Multiprocessor and multicomputer system

One type of parallel processing that does not fit flynn's classifications is **pipelining**

We consider parallel processing under the following main topics.

① Pipeline processing:

It is an Implementation technique. Where arithmetic sub operations (or) the phases of a computer instruction cycle overlap in Execution

② Vector processing:

Deals with Computations involving large Vectors and matrices.

③ Array processing:

Perform Computations on large Array's of data

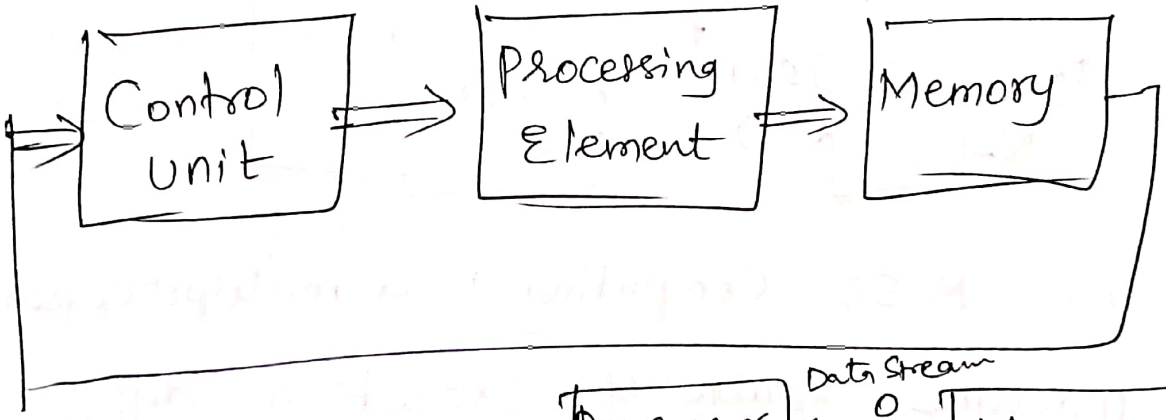
SISD

Single Instruction : ~~String~>

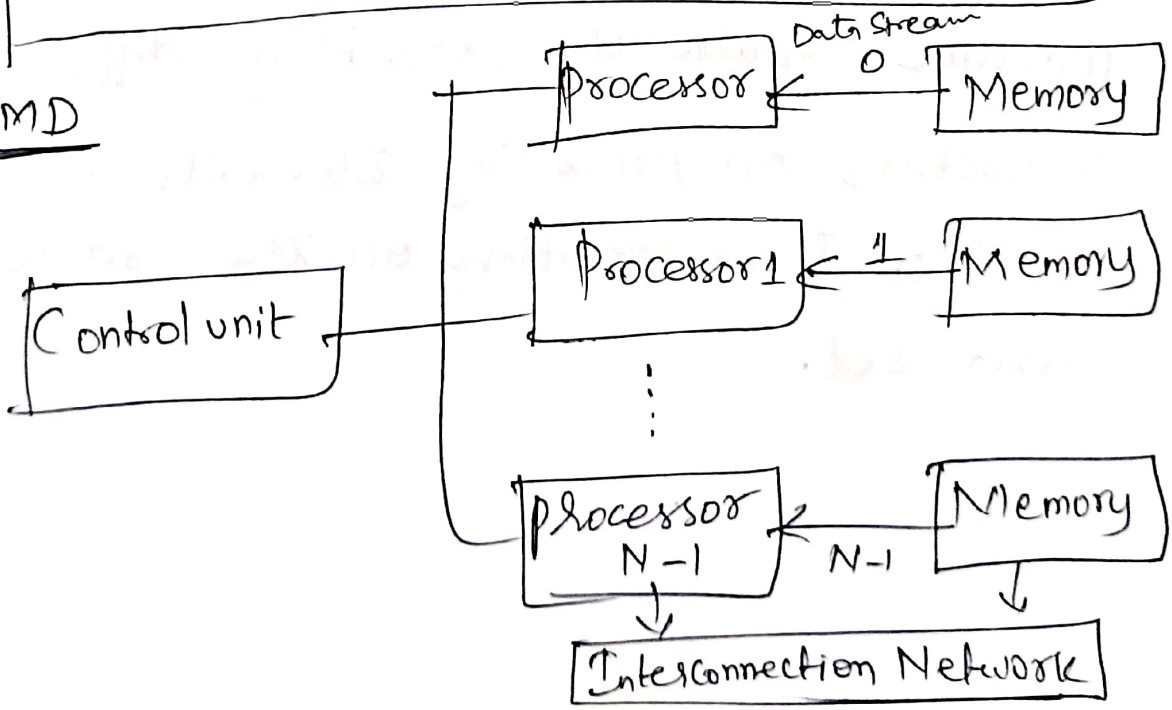
Only one instruction stream is being acted (or) Executed by CPU during one clock cyd.

Single data: Only one data stream is used as input during one clock cycle.

Most Conventional Computers have SISD architecture Where all the instruction and data to be processed have to be stored in primary memory.

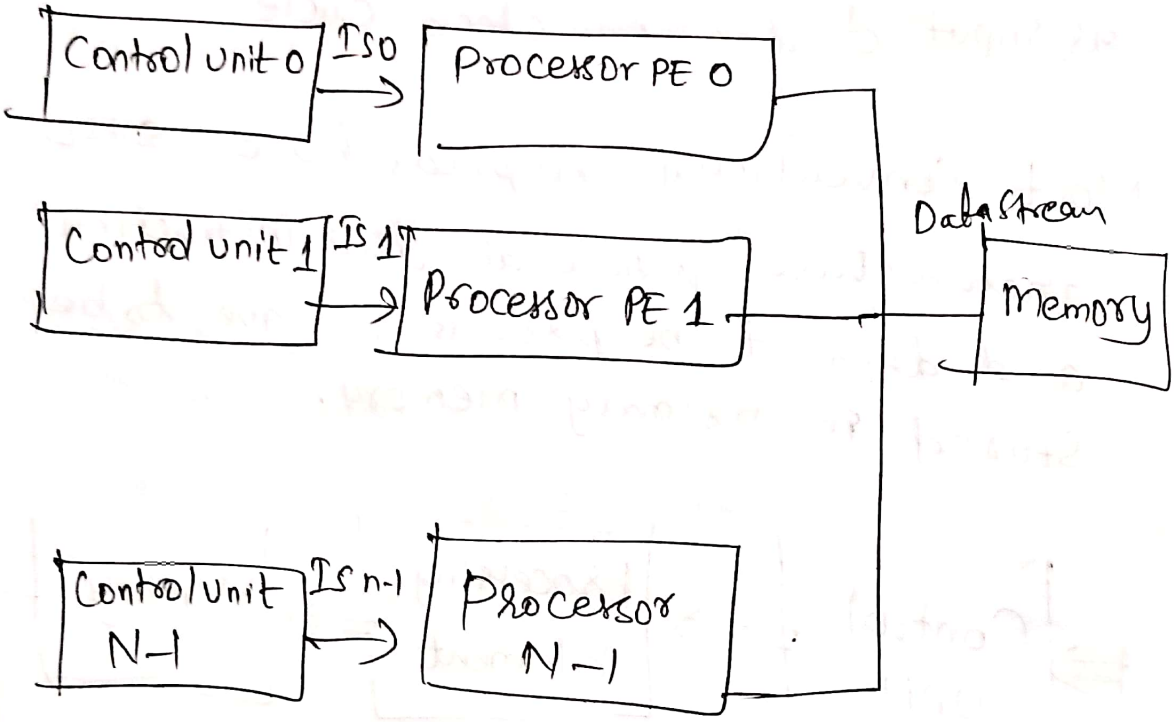


SIMD



A SIMD System is a multiprocessor machine Capable of executing the same instruction on all the CPU's but operating on the different data stream.

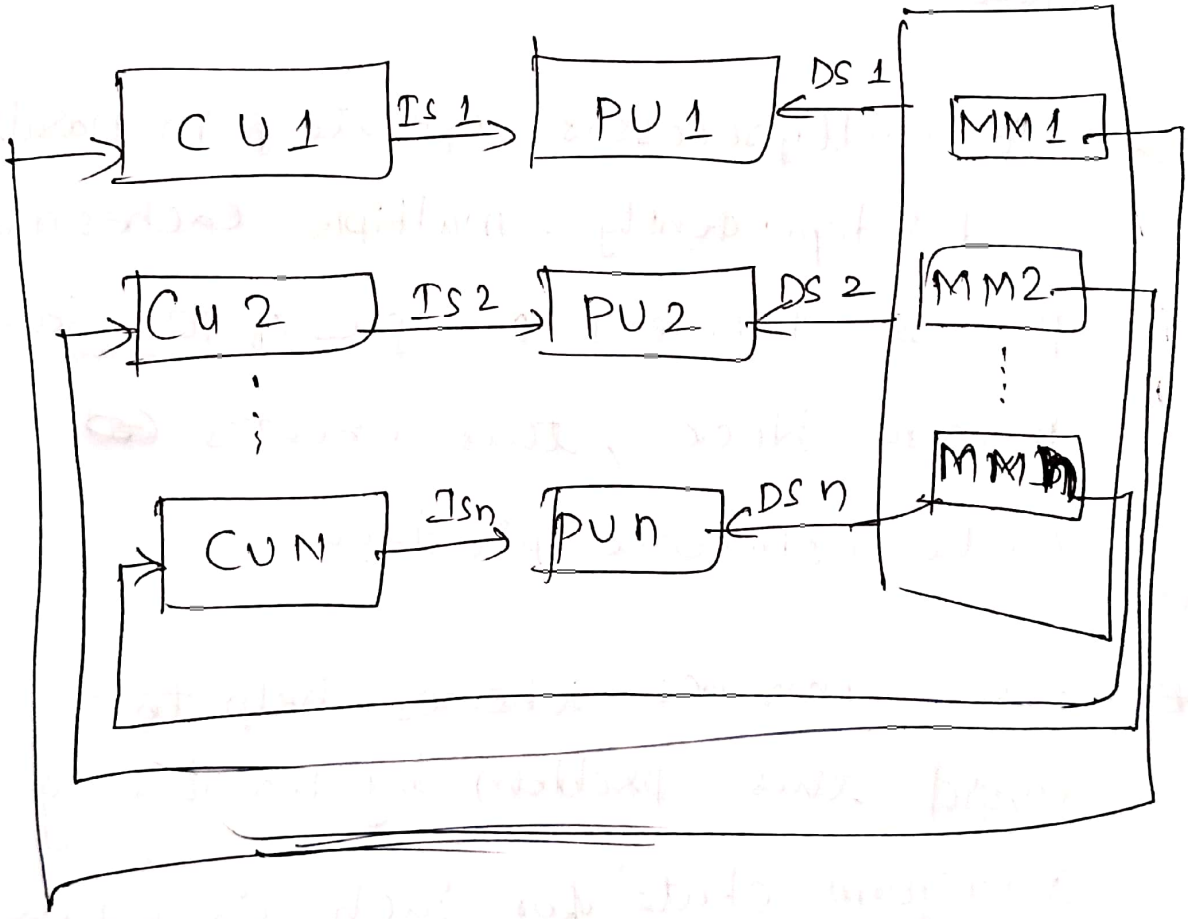
MISD



An MISD Computing is a multiprocessor machine Capable of Executing different instructions on processing Elements but all of them operating on the same data set.

MIMD

A MIMD system is a multiprocessor machine that is capable of executing multiple instructions over multiple data streams. Each processing element has a seperate instruction stream and data stream.



Cache Coherence

In a multiprocessor system, data inconsistency may occur among adjacent levels (or) within the same level of memory hierarchy

For Example: the cache and the main memory may have inconsistant copies of the same object.

→ As multiprocessor operate in parallel and independently, multiple caches may possess different copies of the same memory block, this creates Cache coherence problem.

* Cache Coherence Schemes help to avoid this problem by maintaining a uniform state for each cached block of data

Cache coherence

→ Let X be an Element of Shared data which has been referenced by two processors P_1 and P_2

→ In the beginning three copies of X are Consistant

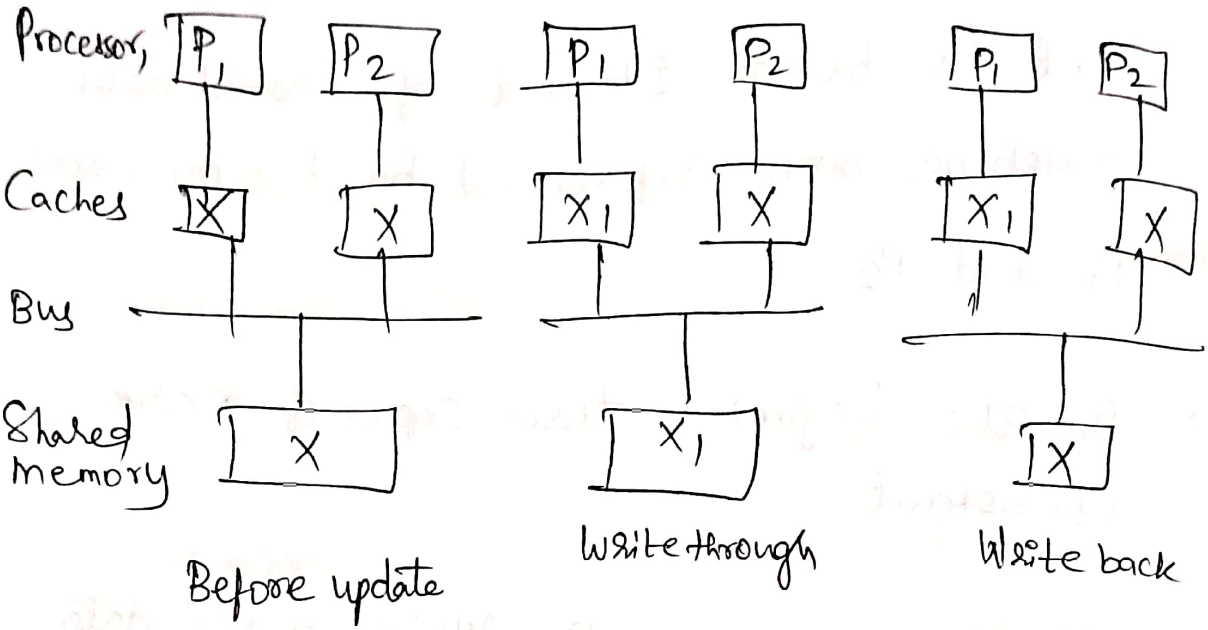
→ If the processor P_1 writes a new data X_1 into the cache,

By using Write-through policy, the same copy will be written immediately into shared memory.

→ In this Case, inconsistency occurs between Cache memory and the main memory

When a write-back policy is used the main memory will be updated when the modified data in the Cache is replaced (or) invalidated.

→ If Cache is full then the value is Replaced (or) modified.

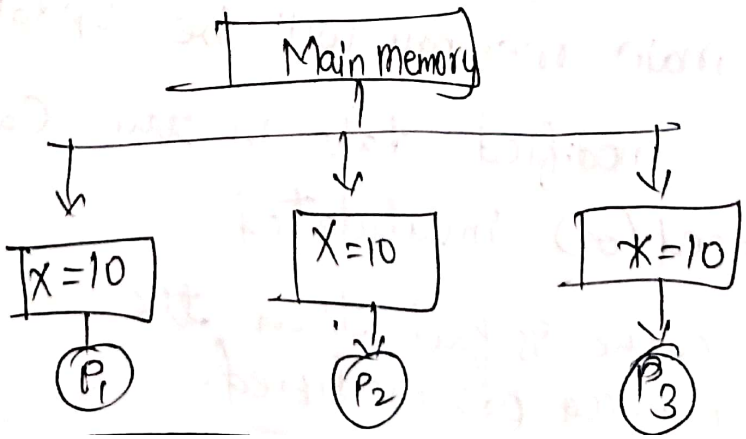


1) Write update :

When Ever the value of (X) is Changed in Cache of (P_1) then it is updated as same in (P_2) and (P_3)

2) Write through :

When (X) is modified in Cache and Shared memory then it is updated in main memory also.



* Write Invalidate :

When Ever the value of (X) is modified (or) Changed in Cache then other Cache of (P_2) and (P_3) values are invalidated.

(2) protocols

(1) Write update → update in memory at a time of modification

(2) Write Invalidate

↳ If any updating of data done then Other Cache values will be invalidated.