PIPELINE AND VECTOR PROCESSING

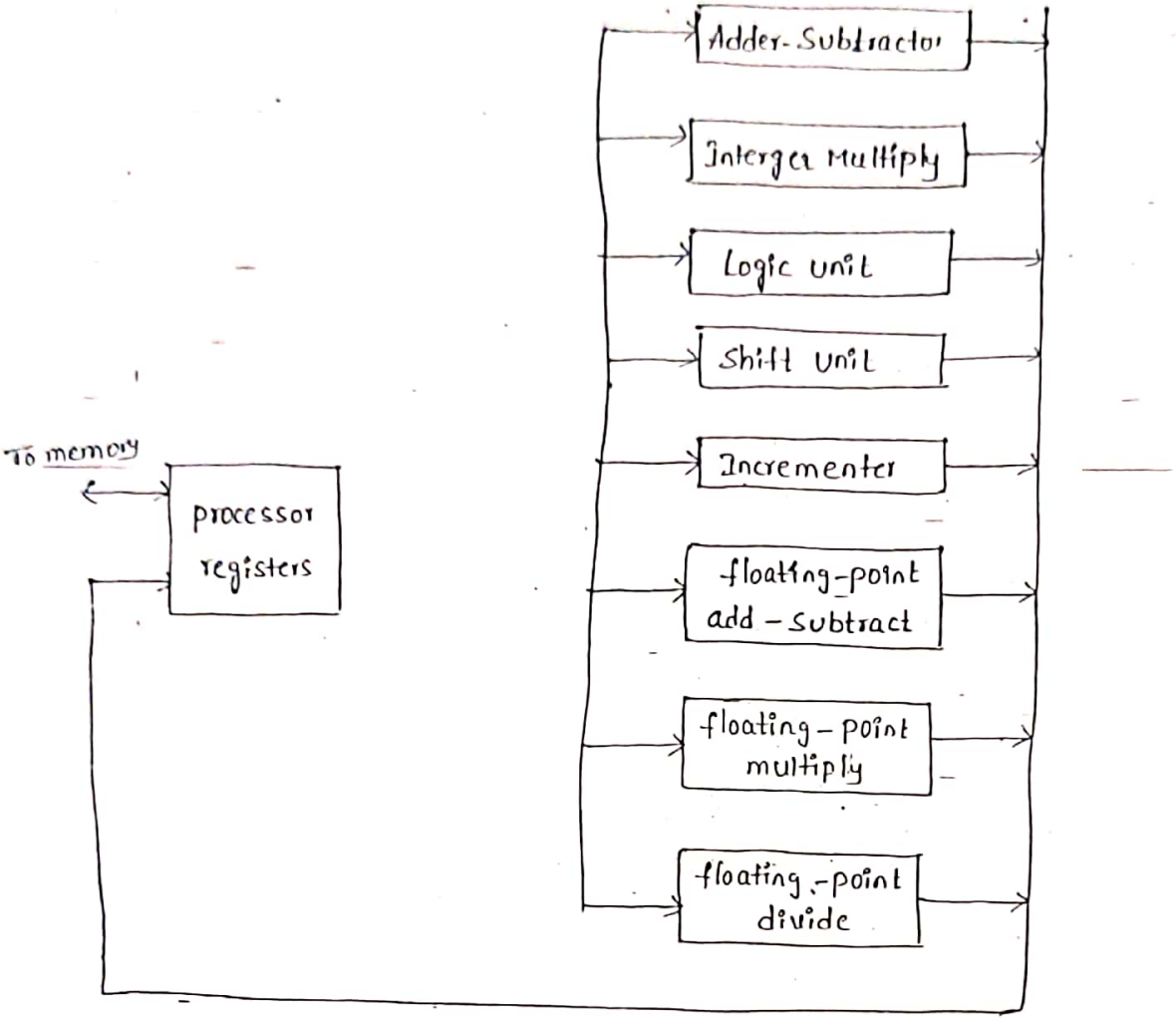
Contents

- parallel processing
- Pipelining
- Arithmetic pipeline
- Instruction pipeline
- RISC pipeline Vector processing
- Array processors
- Multiprocessors
  - Characteristics of Multiprocessors
  - Interconnection Structures
  - Interprocessor Arbitration
  - Interprocessor communication and synchronization
  - Cache coherence.

parallel processing:-

- A parallel processing system is able to perform concurrent data processing to achieve faster Execution time.
- For example, while an instruction is being executed in the ALU, the next instruction can be read from memory.
- The system may have two or more processors operating concurrently and able to Execute two or more instructions at the same time.
- The purpose of parallel processing is to speed up the computer processing capability and increase its throughput, i.e., The amount of processing that can be accomplished during a given interval of time.

→ Processor with multiple functional units is depicted as,



→ It shows the one possible way of seperating the Execution unit into eight functional units opeing in parallel.

→ The operands in the registers are applied to one of the units depending on the operation specified by the instruction associated with the operands.

→ The sequence of instructions read from memory constitutes an instruction stream.

→ The operations performed on the data in the processor constitutes a data stream.

→ parallel processing may occur in the instruction stream, in the data stream, (or) in both.

Flynn's classification :-

According to Flynn's classification divides computer into four major groups :-

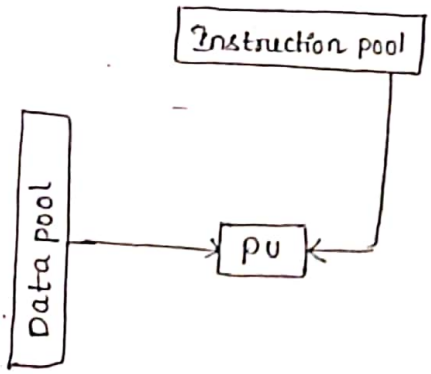
(1) SISD (Single instruction Stream, Single Data Stream)

(2) SIMD (Single instruction Stream, Multiple Data Stream)

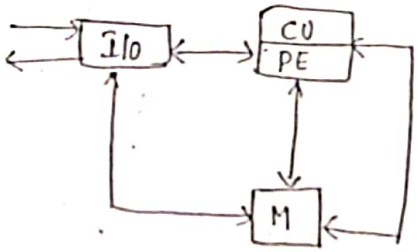
(3) MISD (Multiple instruction Stream, Single Data Stream)

(4) MIMD (Multiple instruction Stream, Multiple Data stream)

SISD :- SISD represents the organization of a Single computer containing a control unit, a processor unit, and a memory unit.



(or)



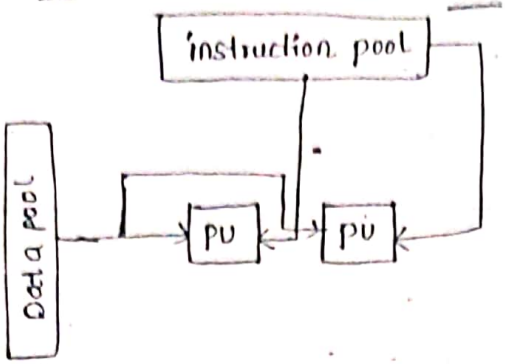
CU - Control unit PE = processing Element,
M = memory

- SISD corresponds to the traditional mono processor (von Neumann computer). A Single data stream is being processed by one instruction Stream

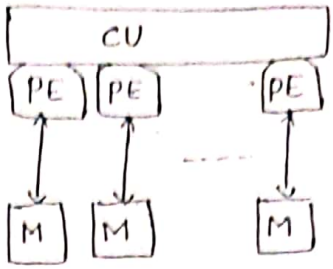
(or)

- A Single processor computer (uni - processor) in which a Single Stream of instructions is generated from the program.

SIMD :- SMD represents the organization of a Single computer containing a control unit, a processor unit, and a that includes many processing units under the supervision of a common control unit.

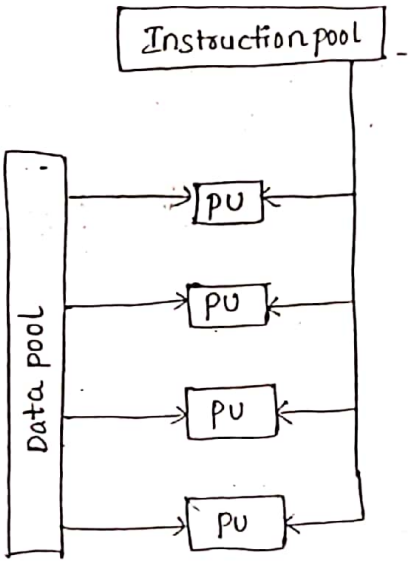


(or)

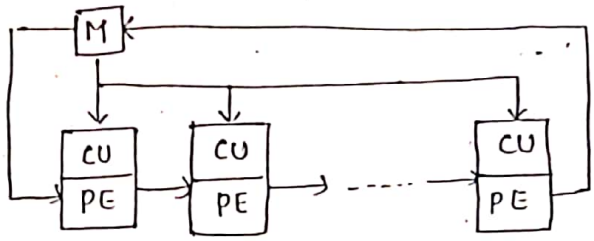


- Each instruction is executed on a different set of data by different processors. i.e., multiple processing units of the same type process on multiple data streams.
- This group is dedicated to array processing machines.
- Sometimes, vector processors can also be seen as a part of this group.

MISD:- MISD structure is only of theoretical interest since no practical system has been constructed using this organization.



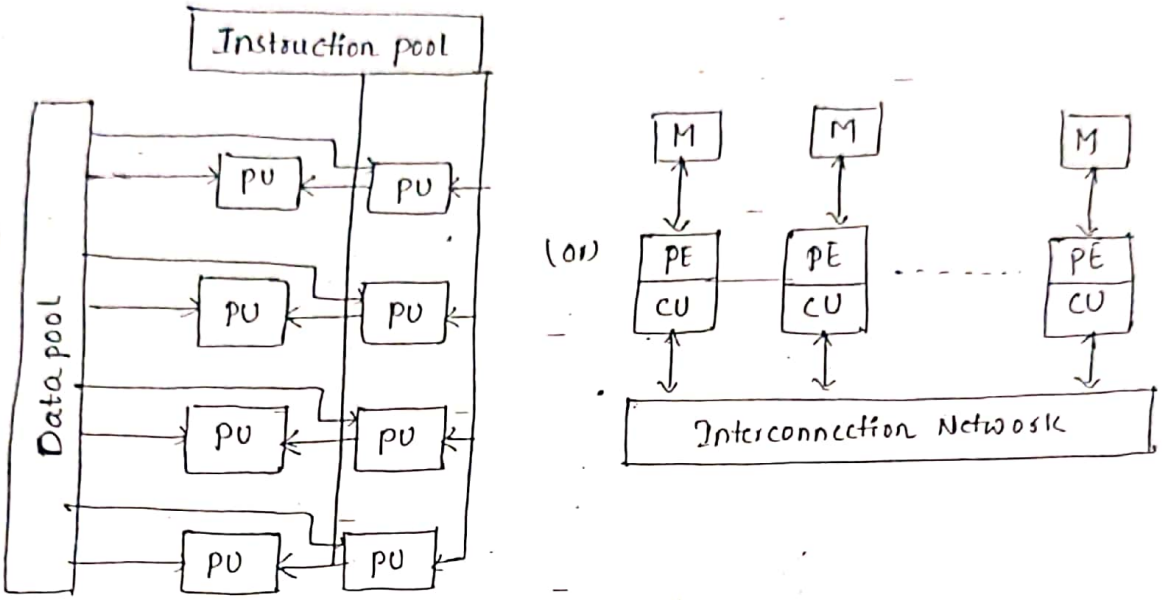
(or)



- Each processor executes a different sequence of instructions.
- In case of MISD computers, multiple processing units operate on one single data stream.
- In practice, this kind of organization has never been used.

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

→ Multiprocessors are best Example.



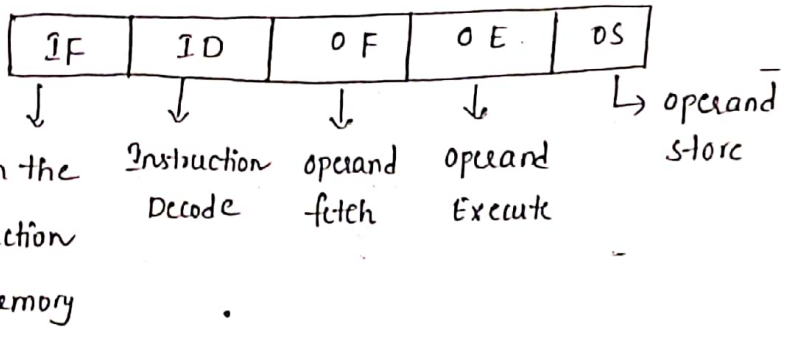
- Each processor has a separate program.
- An instruction stream is generated from each program.
- Each instruction operates on different data.
- MIMD builds the group for the traditional multi processors.

Several processing units operate on multiple-data streams.

Pipelining :- Set of data processing elements which are connected in series.

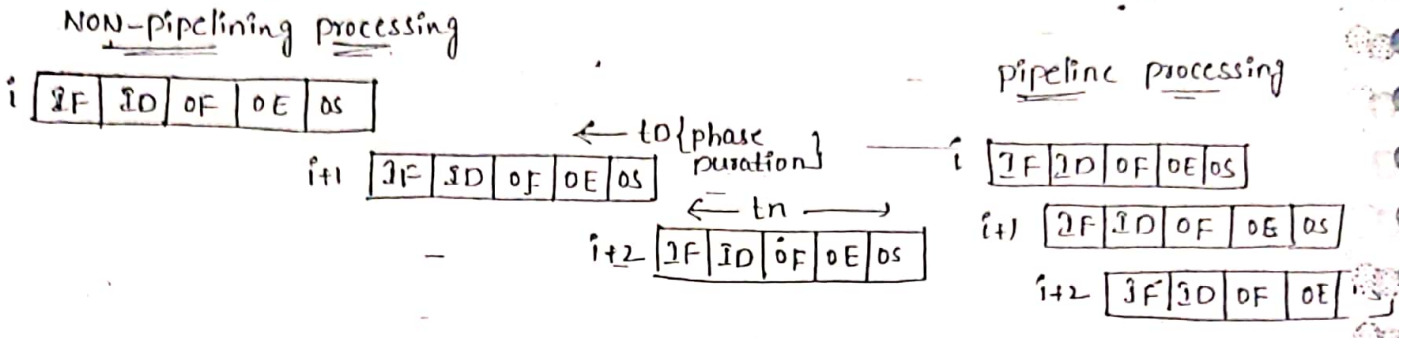
output of the one element = Input of the next instruction.

While Executing the program in memory, involves in 5 phases.



→ pipelining is a technique of decomposing a sequential process into suboperations, with each subprocess being executed in a special dedicated segment that operates concurrently with all other segments.

→ A pipeline can be visualized as a collection of processing segments through which binary information flows.



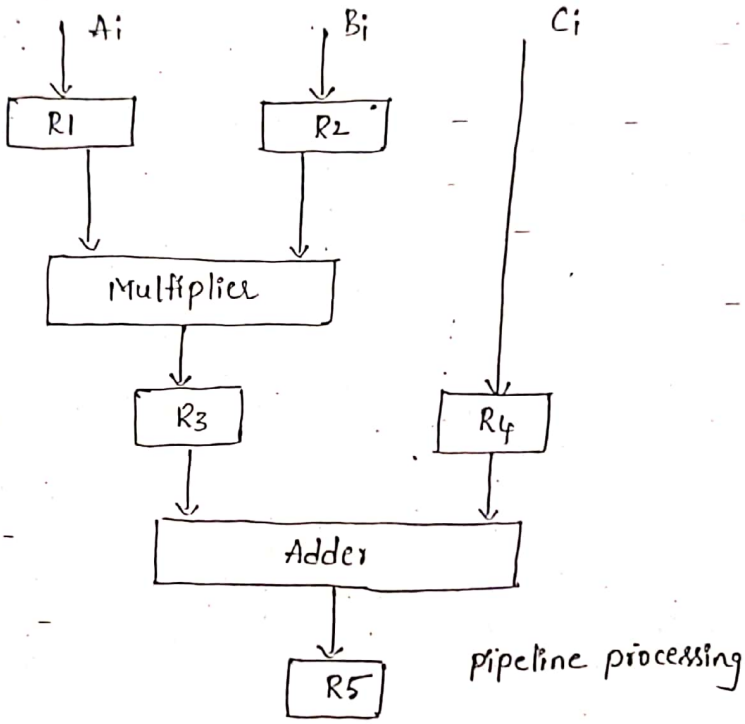
Non-pipeling	pipelining
1. All the actions (fetching, decoding, Executing) of instructions and writing the results are grouped into one step.	1. Multiple Instructions are Execut parallelly.
2. Low throughput (no.of instructions are Executed per cycle)	2. It has high throughput. (amount of instructions Execute per unit time)
3. Only one instruction is Executed per unit time and Execution process requires more number of cycles.	3. In pipelining many instruction are executed at the same time an Execution is completed in fewer cyc
4. It is not dependent on cpu Schedular.	4. The effecting of pipeling sy depends upon the effectiveness of cpu schedular.

→ Suppose we want to perform the combined multiply and add operations with a stream of numbers.

Ex:- $A_i * B_i + C_i$ for $i = 1, 2, 3 \ldots 7$

→ Each suboperation is to be implemented in a segment within a pipeline.

→ Each segment has one or two registers and a combinational circuit is depicted as.



pipeline processing

→ The suboperations performed in each segment of the pipeline are as follows:

$R_1 \leftarrow A_i$, $R_2 \leftarrow B_i$ (Input A_i and B_i)

$R_3 \leftarrow R_1 * R_2$, $R_4 \leftarrow C_i$ (Multiply and Input C_i)

$R_5 \leftarrow R_3 + R_4$ (Add C_i to product)

→ The five registers are loaded with new data every clock pulse.

→ The effect of each clockpulse, the content of registers in pipeline is depicted in a table as,

Clock Pulse Number	Segment 1		Segment 2		Segment 3
	R1	R2	R3	R4	R5
1	A_1	B_1	—		—
2	A_2	B_2	$A_1 * B_1$	C_1	—
3	A_3	B_3	$A_2 * B_2$	C_2	$A_1 * B_1 + C_1$
4	A_4	B_4	$A_3 * B_3$	C_3	$A_2 * B_2 + C_2$
5	A_5	B_5	$A_4 * B_4$	C_4	$A_3 * B_3 + C_3$
6	A_6	B_6	$A_5 * B_5$	C_5	$A_4 * B_4 + C_4$
7	A_7	B_7	$A_6 * B_6$	C_6	$A_5 * B_5 + C_5$
8	—	—	$A_7 * B_7$	C_7	$A_6 * B_6 + C_6$

Arithmetic pipeline :-

→ pipeline Arithmetic units are usually found in very high-speed computers.

→ They are used to implement floating-point operations, multiplication of fixed-point numbers, and similar computations encountered in specific problems.

→ An example of pipeline unit for floating-point addition and subtraction.

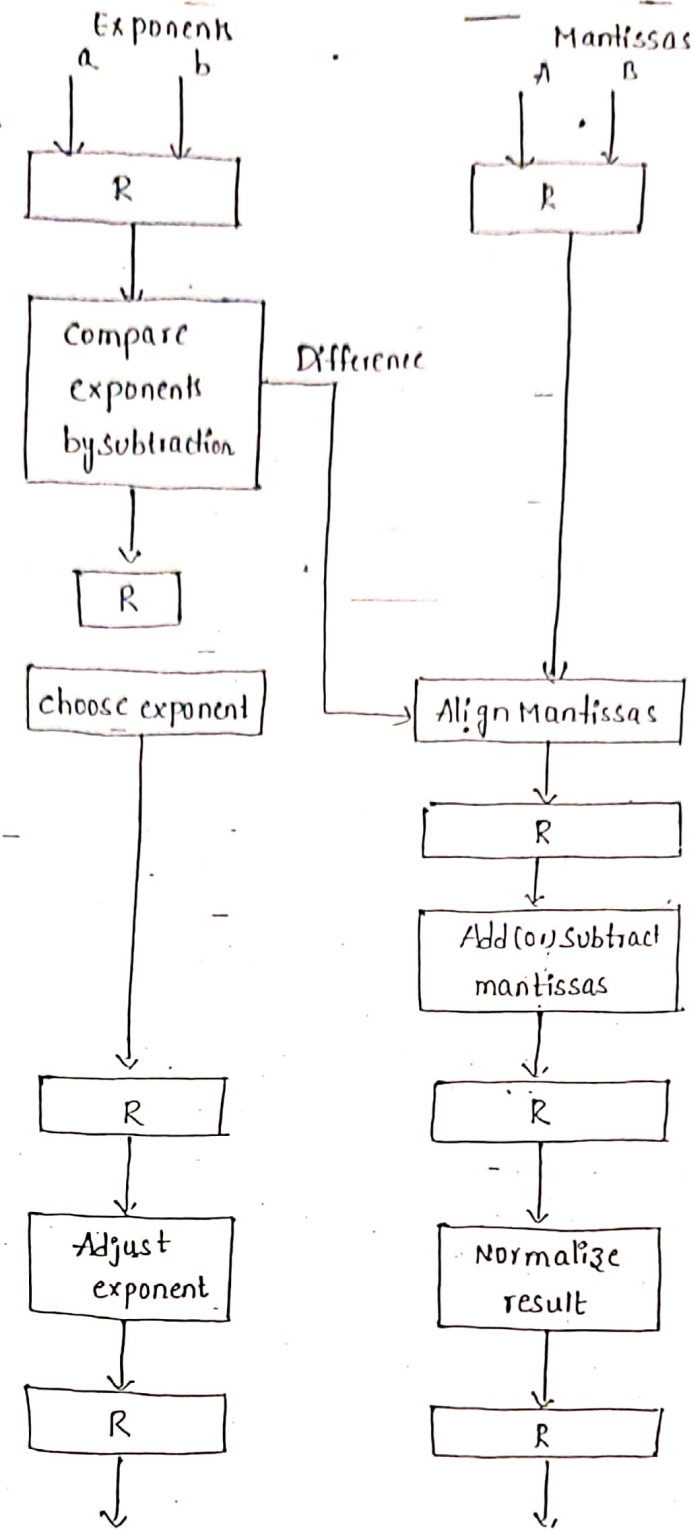
→ The inputs to the floating-point adder pipeline are two normalized floating-point binary numbers.

$$X = A \times 2^a$$

$$Y = B \times 2^b$$

→ A and B are two fractions that represent mantissas and a and b are the exponents.

→ The floating-point addition and subtraction can be performed in four segments, is depicted as



Pipeline for floating-point addition and Subtraction

→ The registers labeled R are placed between the segments to store intermediate results.

→ The suboperations that are performed in the four segments are,

- Compare exponents
- Align the mantissas
- Add or Subtract the mantissas
- Normalize the result.

Ex:- floating-point addition

$X = 0.9504 \times 10^3$ —Same→ 0.9504×10^3

$Y = 0.8200 \times 10^2$ —Adjust→ 0.0820×10^3

Addition = 1.0324×10^3 —Adjust (Normalize result)→ 0.10324×10^4

Instruction pipeline :-

→ pipeline processing can occur not only in the data stream but in the instruction stream as well.

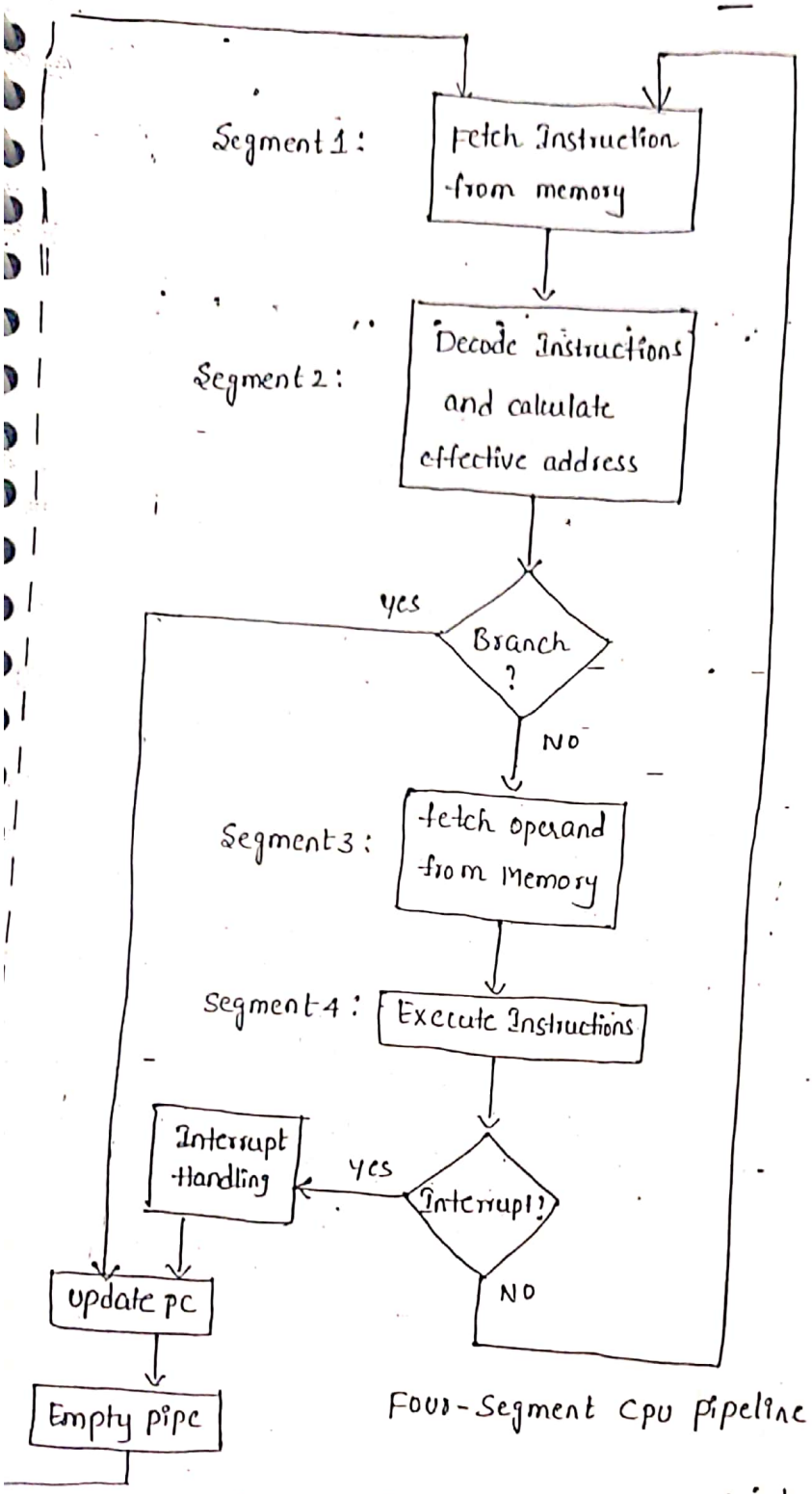
→ An instruction pipeline reads consecutive instructions from memory while previous instructions are being executed in other segments.

→ The computer needs to process each instruction with the following sequence of steps.

- Fetch the Instruction from memory
- Decode the Instruction
- Calculate the effective address
- Fetch the operands from memory
- Execute the instruction
- Store the result in the proper place.

EX:- Four-Segment Instruction pipeline :

→ The operation of the four-segment Instruction pipeline is depicted as



Four-Segment CPU pipeline

→ The time in the horizontal axis is divided into steps of equal duration.

→ The four segments are represented in the diagram with an abbreviated symbol

(i) FI is the segment that fetches an instruction.

(ii) DA is the segment that decodes the instruction and calculates the effective address.

(iii) FO is the segment that fetches the operand.

(iv) EX is the segment that execute the instruction.

→ Timing of instruction pipeline is depicted as.

Steps:	1	2	3	4	5	6	7	8	9	10	11	12	13
Instruction 1	FI	DA	FO	EX									
2		FI	DA	FO	EX								
3			FI	DA	FO	EX							
4				FI	-	-	FI	DA	FO	EX			
5					-	-	-	FI	DA	FO	EX		
6	-		-						FI	DA	FO	EX	
7	-									FI	DA	FO	EX

→ In general, there are three major conflicts (difficulties) that cause the instruction pipeline to deviate from its normal operation.

(i) Resource conflicts

- caused by access to memory by two segments at the same time.
- Most of these conflicts can be resolved by using seperate instruction and data memories.

(ii) Data Dependency conflicts

- arise when an instruction depends on the result of a previous instruction, but this result is not yet available.

(iii) Branch Difficulties

- arise from branch and other instructions that change the value of PC.

RISC pipeline :-

→ Reduced Instruction set computing (RISC)

→ Among the characteristics attributed to RISC is its ability to use an efficient instruction pipeline.

→ The simplicity of the instruction set can be utilized to implement an instruction pipeline using a small number of suboperations, with each being executed in one clock cycle.

→ Because of fixed-length instruction format, the decoding of the operation can occur at the same time as the register selection.

EX: Three-Segment Instruction pipeline :

→ The instruction cycle can be divided into three suboperations and implemented in three segments :

I : Instruction fetch

A : ALU operation

E : Execute instruction.

Delayed load :-

→ consider operation of the following four instructions :

LOAD : R1 ← M[address 1]

LOAD : R2 ← M[address 2]

ADD : R3 ← R1 + R2

STORE : M[address 3] ← R3

→ pipeline timing with data conflict (three-segment pipe...

is depicted in a table as,

clock cycles	1	2	3	4	5	6
Load R1	I	A	E			
Load R2		I	A	E		
Add R1+R2			I	A	E	
Store R3				I	A	E

→ pipeline timing with delayed load is depicted in a table as

clock cycles	1	2	3	4	5	6	7
Load R1	I	A	E				
Load R2		I	A	E			
No-operation			I	A	E		
Add R1+R2				I	A	E	
Store R3					I	A	E

→ The concept of delaying the use of the data loaded from memory is referred to as delayed load.

Delayed Branch :-

→ The method used in most RISC processors is to rely on the compiler to redefine the branches so that they take effect at the proper time in the pipeline. This method is referred to as delayed branch.

Ex :- - load from memory to R1
- Increment R2
- Add R3 to R4
- Subtract R5 from R6
- Branch to address x.

→ An example of delayed branch is depicted as.

Clock cycles	1	2	3	4	5	6	7	8	9	10
Load	I	A	E							
Increment		I	A	E						
Add			I	A	E					
Subtract				I	A	E				
Branch to X					I	A	E			
No operation						I	A	E		
No operation							I	A	E	
Instruction in X								I	A	E

(a) Using No-operation Instructions

Clock cycles	1	2	3	4	5	6	7	8	9
Load	I	A	E						
Increment		I	A	E					
Branch to X			I	A	E				
add				I	A	E			
Subtract					I	A	E		
Instruction in X						I	A	E	

(b) Rearranging the instructions

Vector processing :-

→ In many Science and engineering applications, the problems can be formulated interms of vectors and matrices.

→ Different application areas of vector processing are,

- long-range weather forecasting
- petroleum explorations
- Seismic data analysis
- Medical diagnosis
- Aerodynamics and space flight Simulations
- Artificial Intelligence and expert systems
- Image Processing

Vector operations :-

→ A vector is an ordered set of a one-dimensional array of data items.

→ A vector V of length n represented as a row vector by

$$V = [V_1, V_2 \cdots V_n]$$

EX :- FORTRAN DO loop

```
      DO   20   I=1, 100
20    _    C(I) = B(I) + A(I)
```

- This is a program for adding two vectors A and B of length
- 100 to produce a vector C.
- This is implemented in machine language as,

```
      Initialize  I=0
20    Read  A(I)
      Read  B(I)
      Store  C(I) = A(I) + B(I)
      Increment  I = I+1
      If  I ≤ 100  go to 20
      Continue :
```

- This constitutes a program loop that reads a pair of operands from arrays A and B and performs a floating-point addition.
- The loop control variable is then updated and the steps repeat 100 times.

Matrix Multiplication

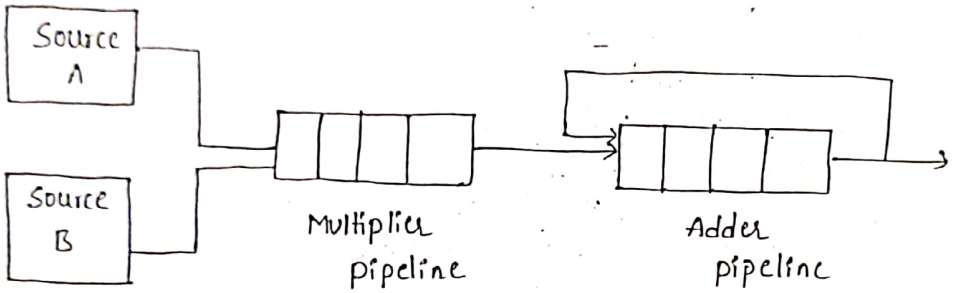
→ The multiplication of two 3×3 matrices A and B

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}$$

→ The product matrix C is a 3×3 matrix whose elements are related to the elements of A and B by the inner product.

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \times b_{kj}$$

→ The pipeline for calculating an inner product is depicted as

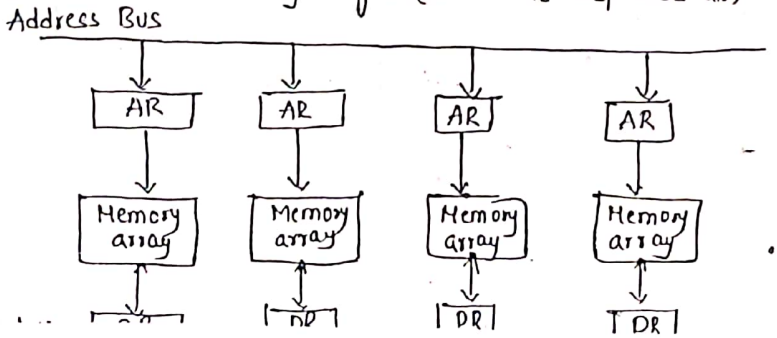


Memory Interleaving :-

→ Instead of using two memory buses for simultaneous access, the memory can be partitioned into a number of modules connected to a common memory address and data buses.

→ A memory module is a memory array together with its own address and data register.

→ Multiple module memory organization is depicted as,



→ The advantage of a modular memory is that it allows the use of a technique called interleaving.

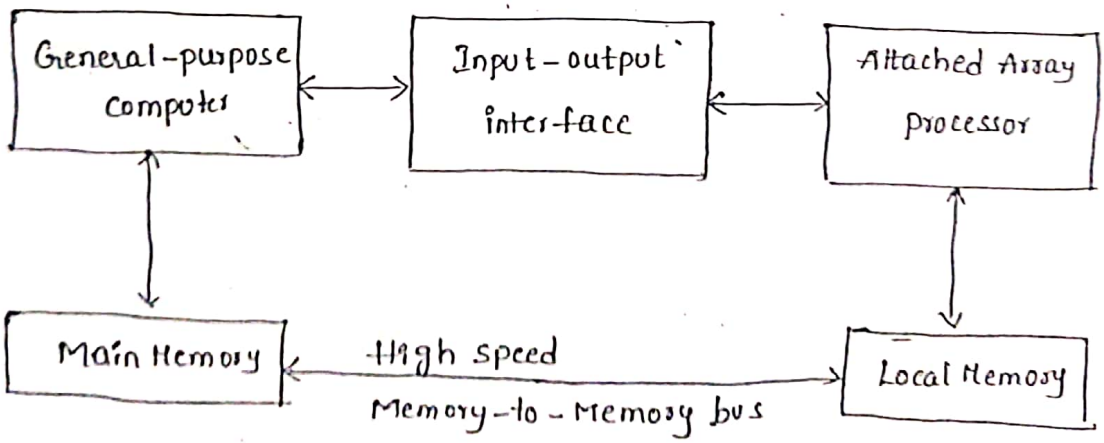
→ In an interleaved memory, different sets of addresses are assigned to different memory modules.

Array processor :-

- An array processor is a processor that performs the computations on large arrays of data.
- There are two different types of array processor :
  - Attached Array processor
  - SIMD Array processor

Attached Array processor :-

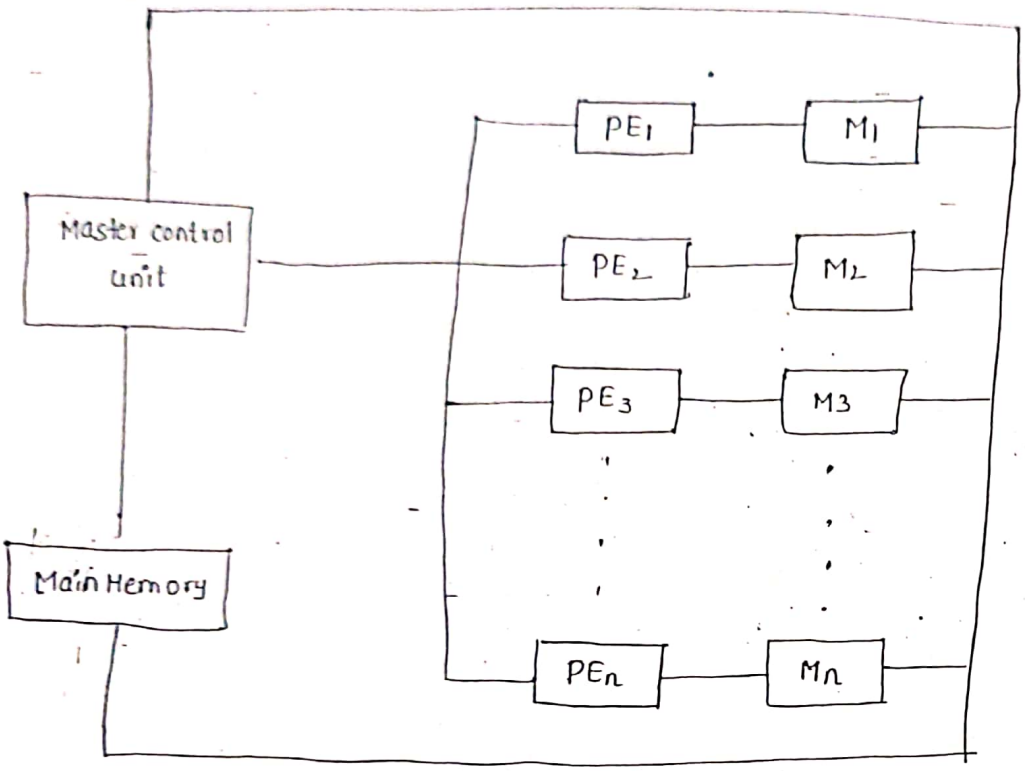
- It is designed as a peripheral for a conventional host computer.
- Its purpose is to enhance the performance of the computer by providing vector processing.
- It achieves high performance by means of parallel processing with multiple functional units.



Attached Array processor with host computer

SIMD Array processor :-

- It is processor which consists of multiple processing unit operating in parallel.
- The processing units are synchronized to perform the same task under control of common control unit.
- Each processor elements (PE) includes an ALU, a floating point arithmetic unit and working registers.



SIMD array processor organization

Multiprocessors :-

- A Multiprocessor is an interconnection of two or more cpus with memory and input-output equipment.
- The term processor in multiprocessor system can mean either a central processing unit (cpu) or an input-output processor (IOP).
- Multiprocessors are classified as multiple instruction stream multiple data stream (MIMD) systems.

- The main distribution between a multicomputer and a multiprocessor system is, in multicomputer system computers are interconnected with each other to form a network and they may or may not communicate with each other.
- A multiprocessor system is controlled by one operating system that provides interaction between processors and all the components of the system cooperate in the solution of a problem.
- Multiprocessing improves the reliability of the system so that a failure in one part has a limited effect on the rest of the system.
- Multiprocessors provide "fault-tolerent", If a fault causes one processor to fail, another processor can be assigned to perform the functions of disable processor.
- Multiprocessor organization improves System performance.
- Multiprocessing can improve performance by decomposing a program into parallel executable tasks.
  - This can be achieved in two ways.
  - The user can explicitly declare that certain tasks to be executed in parallel.
  - The other efficient way is to provide a compiler that can automatically detect parallelism in a user's program.
  - The compiler checks for data dependency in the program.
  - If a program depends on data generated in another part, the part yielding the needed data must be executed first.

- Two parts of a program that do not use data generated by each can run concurrently.
- Multiprocessors are classified by the way their memory is organized.
- A multiprocessor system with common shared memory is classified as a shared memory or "tightly coupled multiprocessor".
- Information can be passed by placing that in common global memory
- Another type is the distributed memory or loosely-coupled system.
- Each processor element in a loosely coupled system has it's own private local memory.

Interconnections Structures :-

- The components that form a multiprocessor system are CPU's IOPs connected to input output devices, and a memory unit.
- The interconnection between the components can have different physical configurations, depending on the number of transfer paths that are available.
  - Between the processors and memory in a shared memory system.
  - Among the processing elements in a loosely coupled system.
- there are several physical forms available for establishing an interconnection network.
  - Time shared common bus
  - Multiport Memory
  - Crossbar Switch
  - Multistage switching network
  - Hypercube system Time shared common bus.

1. Time shared common Bus:-

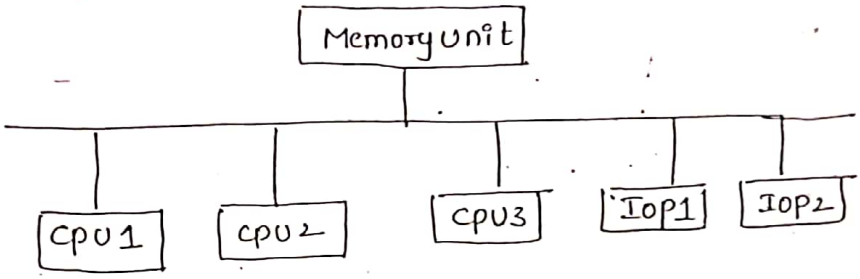
A common bus multiprocessor system consists of a number of processors connected through a common path to a Memory unit.

Disadvantage:-

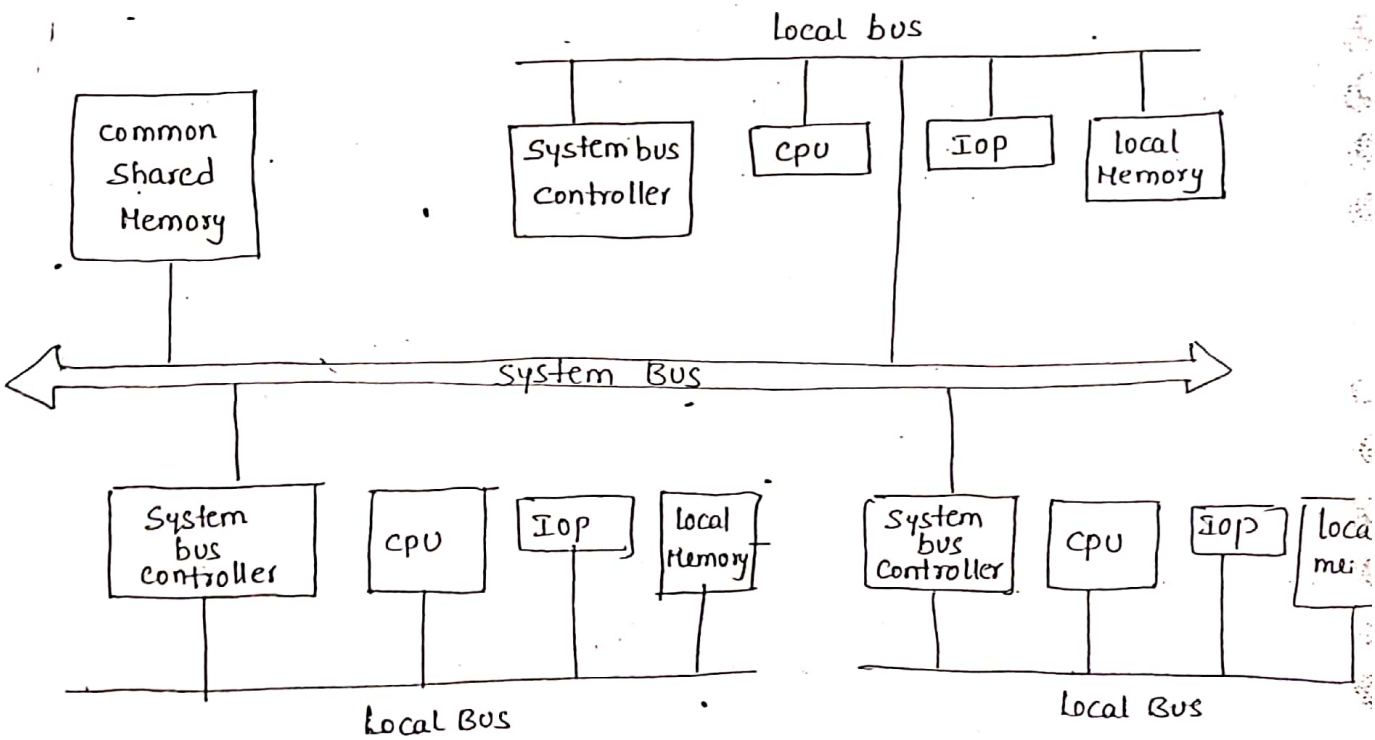
1. only one processor can communicate with the memory or another processor at any given time.

2. The overall transfer rate within the system is limited by the speed of the single path.

3. part of the local memory may be designed as a cache memory attached to the cpu.



Time shared common bus organization



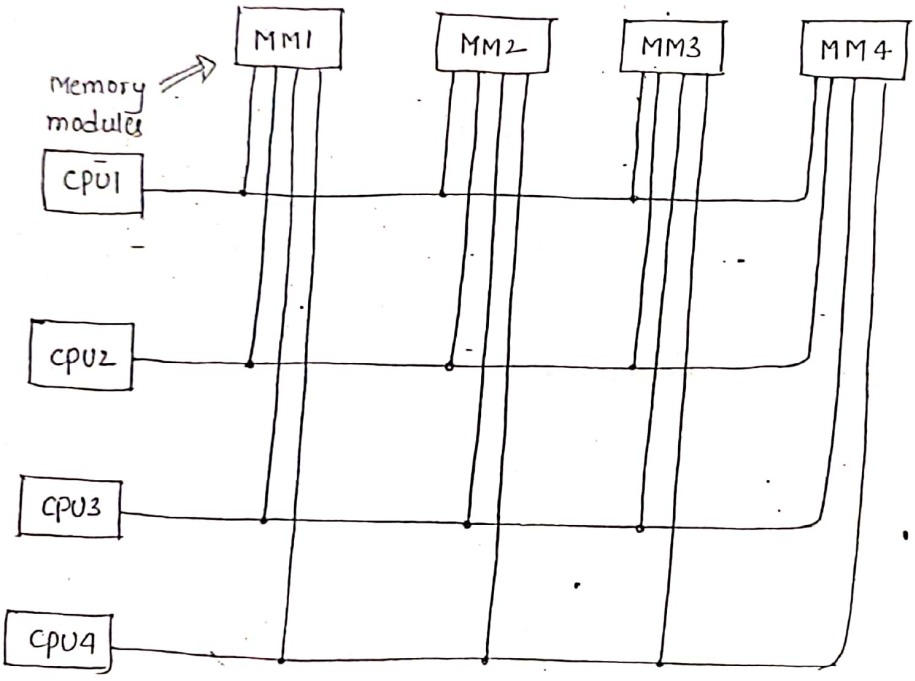
System bus structure for multiprocessors.

2. Multiport Memory :-

- A multiport memory system employs separate buses between each memory module and each cpu.
- The module must have internal control logic to determine which port will have access to memory at any given time.
- Memory access conflict are resolved by assigning fixed priorities to each memory port.

Advantage :- The higher transfer rate can be achieved because of the multiple paths.

Disadvantage :- It requires expensive memory control unit logic and large no. of cables & connections.



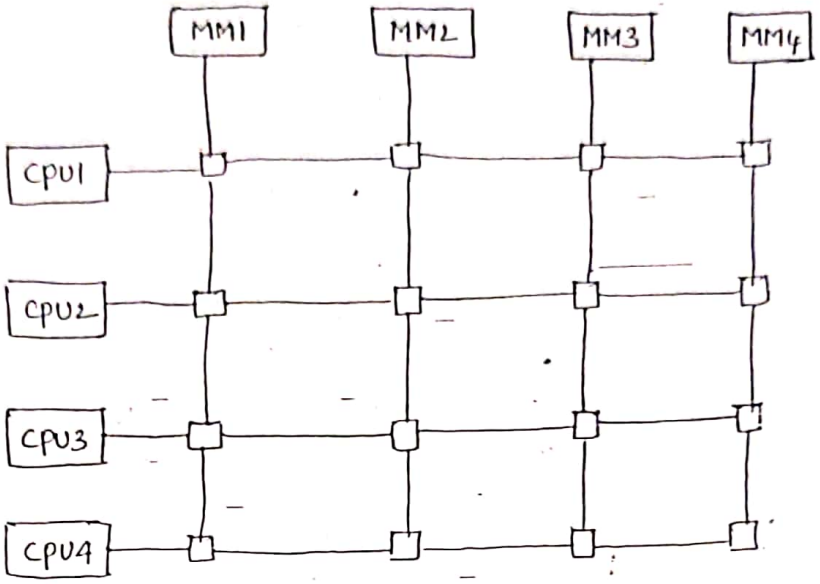
Multiport Memory organization

③ Crossbar Switch :-

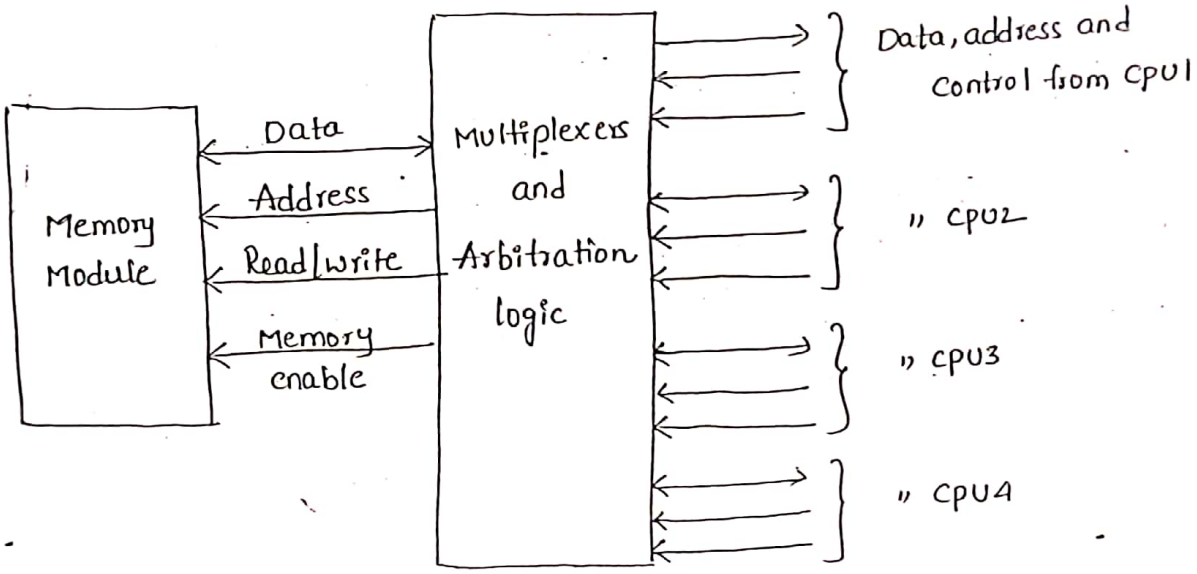
- Consists of a no. of crosspoints that are placed at intersections between processor buses and memory modular paths.
- The small square in each crosspoint is a switch that determines the path from a processor to a memory module.

advantage :- supports simultaneous transfers from all memory modules.

Disadvantage :- The hardware required to implement the switch can become quite large and complex.



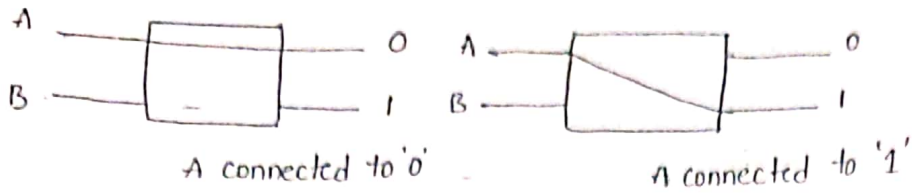
Crossbar Switch



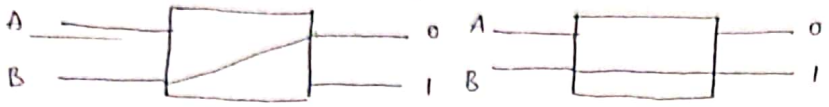
Block diagram of crossbar switch

(4) Multistage Switching Network :-

- The basic component of multistage network is a two-input, two-output interchange switch.



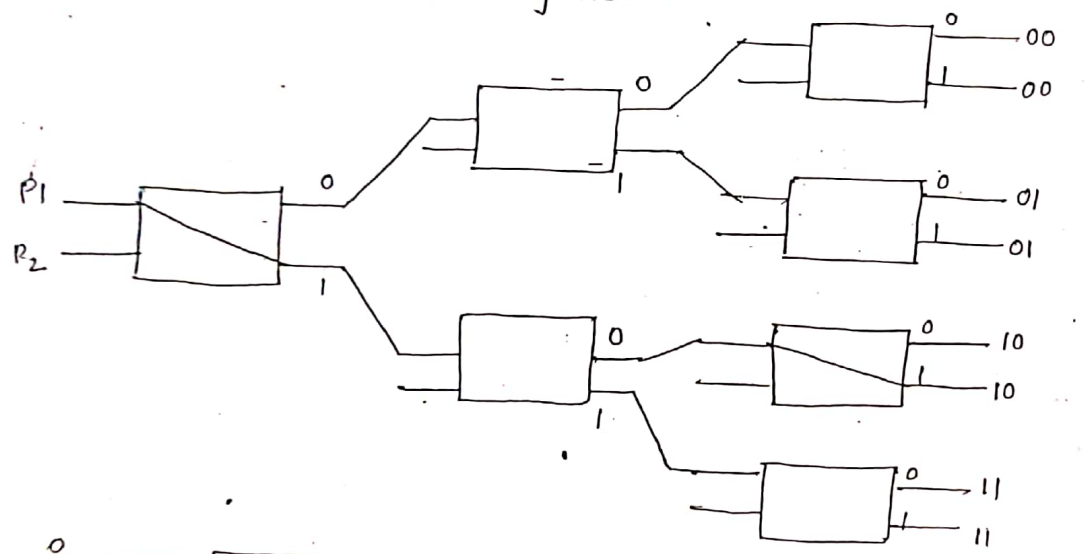
A connected to '0' A connected to '1'

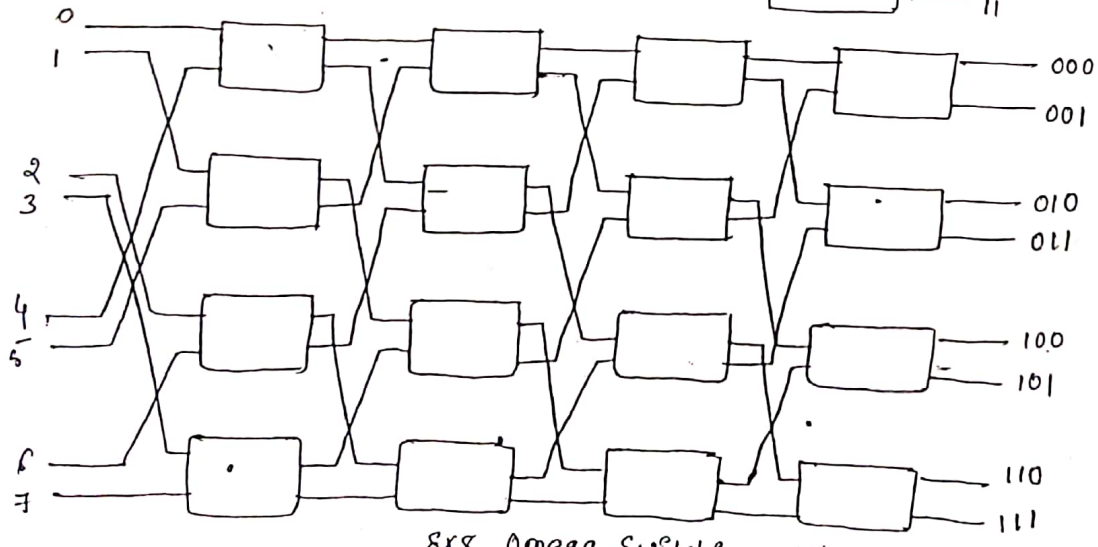


B connected to '0' B connected to '1'

- Using the 2x2 switch as a building block, it is possible to build a multistage network to control the communication between the no. of sources and destinations.

consider the binary tree





8x8 Omega Switching Network

• Some request patterns can't be connected simultaneously.
i.e., any two sources can't be connected simultaneously to destination.
000 and 001.

• In a tightly coupled multiprocessor system, the source is a processor and destination is memory module.

• Set up the path ⇒ transfer the address into memory to transfer the data.

• In a loosely coupled multiprocessor system, both the source and destination are processing elements.

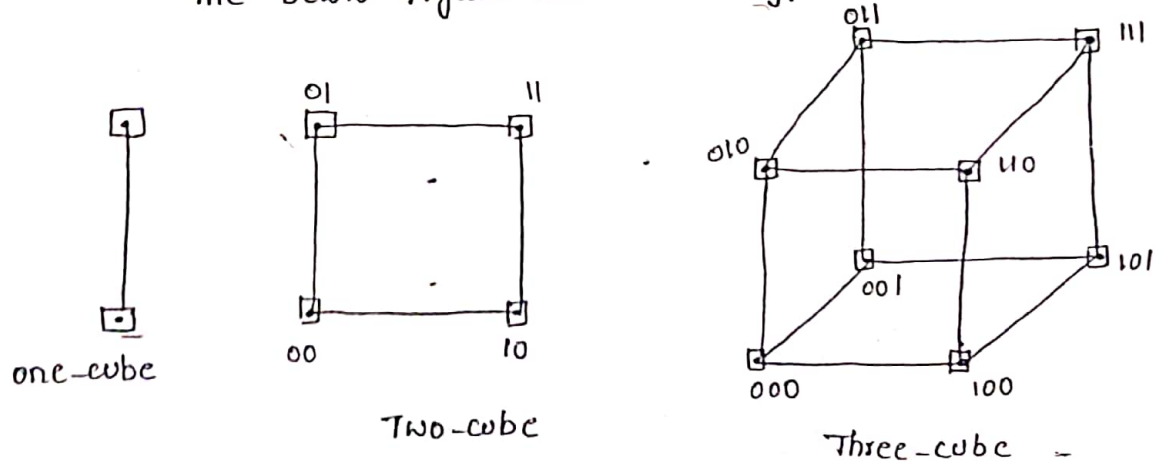
⑤. Hypercube Systems:-

• The Hypercube or binary n-cube multiprocessor structure is a loosely coupled system composed of $N = 2^n$ processors, interconnected in an n-dimensional binary cube.

• Each processor forms a node of the cube, in effect it contains not only a cpu but also local memory and I/O interface.

• Each processor address differs from that of each of its n neighbours by exactly one bit position.

• The below figure shows the hypercube structure for n = 1,2,3 —



Hypercube structure for n = 1,2,3

- Routing messages through an n-cube structure may take from one to n links from a source code to a destination node.
- A routing procedure can be developed by computing the exclusive-OR of the source node address with the destination node address.

Interprocessor Arbitration :-

Arbitration :- It is deciding who gets access to the bus for dividing the transaction. It decided centrally, by "bus master" or "bus controller". Always only one master device per bus.

Why We need?

Bus arbitration scheme usually try to balance

→ Bus priority

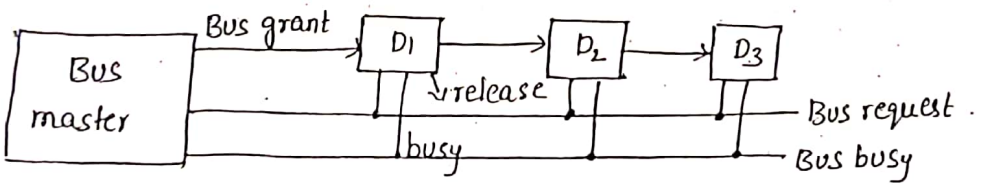
→ fairness

There are three arbitration mechanisams are available

* Daisy chain
* polling
* Independent Request

} Each uses special control lines for arbitration.

Daisy chain :-



Step1 :- If bus not busy, make bus request, Sends the bus request to the bus master.

Step2 : Master activates bus grant by checks the priority which devices have heighest priority using bus grant Signal.

- Which device receives the grant Signal it is set it as busy i.e., it is not available for the other devices.

Step3:- After completion of work it releases the bus for D2 will take the bus request and so on.

Advantages:- (1) Simple

(2) only three external bus lines required

Disadvantages:- (1) poor fault tolerance

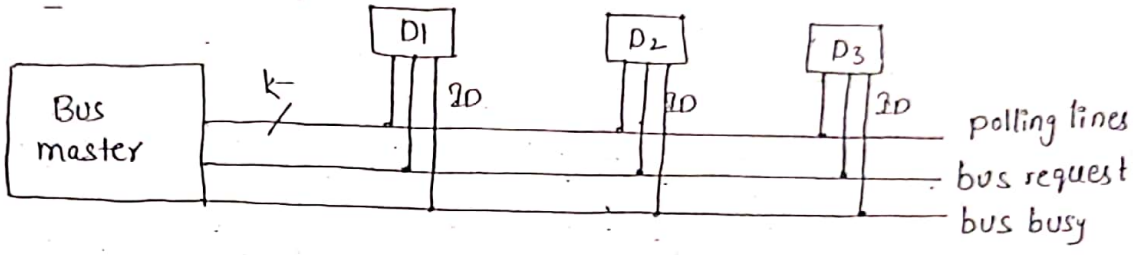
(2) Hardware priority

(3) cannot assure fairness

(i.e., less priority device needs to wait finite amount of time.)

(4) It is slower.

(2) polling :-



Steps:-

1. If bus not busy makes bus request to the master.
2. Master polls by checking each device which have height priority by placing device ID's on polling lines.
3. If device gets bus grant can be marked as bus busy.

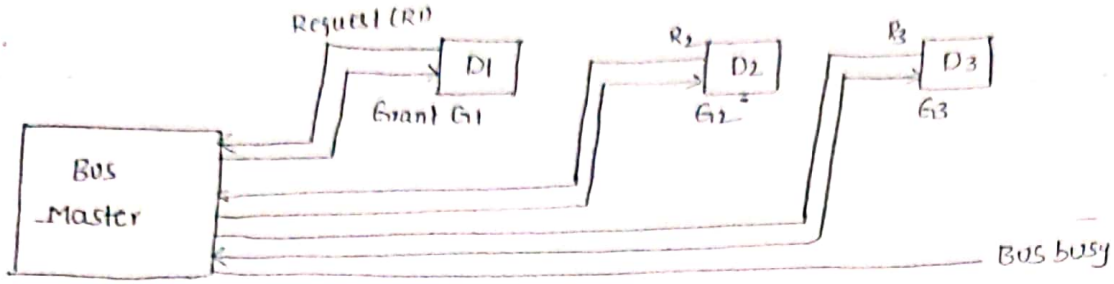
Advantages:-

1. No disadvantages of Daisy chain
2. Dynamic priority is possible
3. It is faster.

Disadvantages:-

1. Extra poll lines should be required
2. polling is delay.

③ Independent Request :-



Steps:- Same as above three steps

Advantages :- 1) It is fast

2) dynamic priority is possible

Disadvantage :- 1) 2n lines - for n devices.

Interprocess communication & Synchronization

What is Ipc:-

- processes within a system may be independent or cooperating.
- Mechanism whereby one process can communicate or exchange data, with another process.
- Set of techniques for the exchange of data among multiple threads in one or more process.
- processes may be running on one or more computers connected by a network.

Why Ipc ?

- Information Sharing
- Resource sharing
- Computation Speedup
- Synchronization
- Modularity
- convenience

Ipc Mechanisms:-

→ • Message passing :- send messages to other processes or receive messages from them.

- Establish a communication link
  - physical
  - logical
- Exchange messages two primitive operations for fixed or variable sized message passing
  - Send
  - receive
- Message passing systems
  - Direct or indirect communications
  - Symmetric or Asymmetric communications
  - Automatic or Explicit buffering

→ Synchronization :- Synchronization refers to the special case where the data used to communicate between processors is control information.

- Synchronization is needed to enforce the correct sequence of processors and to ensure mutually exclusive access to shared writable data.

- In multiprocessors system usually include various mechanisms to deal with the synchronization of resources, and one of the technique is

→ "Mutual exclusion with semaphore"

Mutual Exclusion :- If a process is executing its critical section then no other process can execute in their critical section.

Critical Section :- A critical section is a piece of code that accesses a shared resource that must not be concurrently accessed.

Critical Section problem :- It is a code segment that can be accessed by one process at a time.

→ it contains Shared variable (Every one can use).

Processes	P1	P2	P3
Resources	A	B	C

B's c are must be wait until they get chance.

```
do
{
    Entry Section => controls the entry into critical section and
                     gets lock on required resources.
    Critical Section => nothing but code
    Exit Section => removes the lock on resources and
                    others know that it's critical section over
    Remainder Section
} while(true);
```

Semaphores :- It is synchronization tool that does not require busy waiting. It is denoted by 's' and it is an integer value. It has two control signals "wait(s)", "signal(s)".

```
wait(s)
{
    while (s<=0) do no operation;
    s:=s-1;
    block();
}
signal(s)
{
    s:=s+1;
    if (s>=0)
    {
        wake up();
    }
```

Semaphore s=1 then process enter into it's critical section s=0, exit.

Semaphore Types

(1) Counting semaphore :- It can take non-negative integer values. Int value can range over an unrestricted domain.

(2) Binary semaphore :- Int value can range over between '0' & '1'.
'0' → Locked or unavailable
'1' → unlock or available

** Cache coherence :-

In a shared memory multiprocessor with a separate cache memory for each processor. So that,

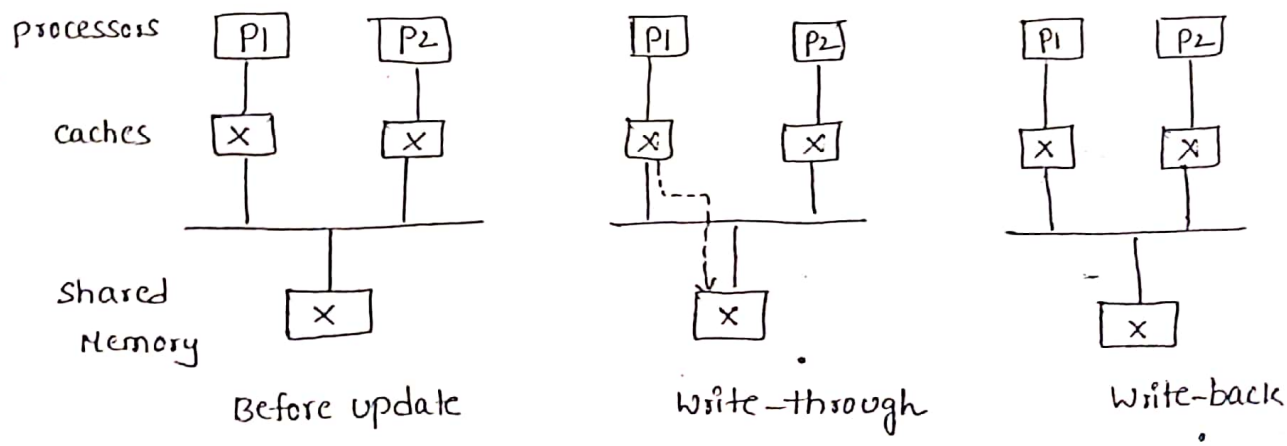
- Multiple copy of the same data can exist in the different caches simultaneously,
- and if processors allowed to update their own copies freely, an inconsistent view of memory can result.

Write policies :- Write back, Write through

→ In a write back policy only cache is updated and the location marked so that it can be copied later into main memory.

→ In a write through policy cache and main memory are updated with every write operation.

So, cache coherence is a problem of keeping the data in these cache consistent.



Solutions to cache coherence

There are two solutions

- Hardware
- Software

Hardware solution :- In Hw solution the cache controller specify designed to allow it to monitor all bus requests from CPU's and IOP's.

Hardware solution uses two protocols

1. Directory protocol
2. Snoopy protocol

1. Directory protocol :-

- It collects & maintain the information about copies of lines reside.
- contain the information about content of various local caches.
- Keeping the information up-to-date.-
- Manage the information which caches copy which line.

Drawback :- only for less buses not large scale systems.

2. Snoopy cache protocol :-

→ Distributed responsibility for maintaining cache coherence among all of the cache controller in the multiprocessor.

Basic approach is : Write invalid & Write update

Write Invalid protocol :- There can be multiple readers but only one writer at a time, only one cache can write to the line.

Write update protocol :- There can be multiple writers as well as multiple readers.

→ When processor wishes to update a shared line, the word to be [illegible] and all caches update it.

Software cache solution:-

→ In the slw based protocol based on the compiler & OS.

→ Compiler based coherence mechanisam performed an analysis on the code to determine which data items become unsafe for caching, and mark them those item accordingly.

→ The OS prevent any non-cacheable items from being cached.

→ software approaches are attractive because to overhead of deleting potential problems is transferred runtime to compiletime.

Important Questions

1. What is meant by parallel processing ? Explain flynn's classification ?
2. What is meant by pipelining ? Explain Arithmetic pipeline.
3. Explain Instruction pipeline ; What are the conflicts occured in Instruction pipeline?
4. Explain the phases of Risc pipeline ?
5. Explain the concepts of (i) vector processing
   (ii) Array processing
6. What is meant by multiprocessor ? Explain characteristics of M.p's ?
7. Explain the Interconnection structures in M.p's ?
8. What is meant by Interprocess Arbitration ? Explain ?
9. Explain (i) Interprocess-communication
   (ii) Ip Synchronization
10. Explain Cache coherence problem with it's Solutions: