COMPUTER ORGANIZATION AND ARCHITECTURE

Course code: ACSB07

IV. B.Tech II semester

Regulation: IARE R-18

Prepared by:

Dr.P.L.Srinivasa Murthy
Professor

r

COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

DUNDIGAL, HYDERABAD - 500 043

Course Outcomes

The course should enable the students to:

CO 1	Understand the organization and levels of design in computer architecture and To understand the concepts of programming methodologies.
CO 2	Describe Register transfer languages, arithmetic micro operations, logic micro operations, shift micro operations address sequencing, micro program example, and design of control unit.
CO 3	Understand the Instruction cycle, data representation, memory reference instructions, input-output, and interrupt, addressing modes, data transfer and manipulation, program control. Computer arithmetic: Addition and subtraction, floating point arithmetic operations, decimal arithmetic unit.
CO 4	Knowledge about Memory hierarchy, main memory, auxiliary memory, associative memory, cache memory, virtual memory Input or output Interface, asynchronous data transfer, modes of transfer, priority interrupt, direct memory access.
CO 5	Explore the Parallel processing, pipelining-arithmetic pipeline, instruction pipeline Characteristics of multiprocessors, inter connection structures, inter processor arbitration, inter processor Communication and synchronization

MODULE-V

MULTIPROCESSORS

Course Outcomes

CO 1	Understand the Pipelining and Parallel processing units.
CO 2	Classify the pipelining, arithmetic pipeline and instruction pipeline.
CO 3	Describe the Characteristics of multiprocessors and inter connection structures.
CO 4	Classify the different inter processor arbitration, inter processor communications and synchronization.

Contents

Pipeline and Vector Processing:

- Parallel processing
- Pipelining
- Instruction pipeline
- Vector Processing
- Array Processors

Multiprocessors:

- Characteristics of multiprocessors
- Inter connection structures
- Inter processor arbitration
- Inter processor communication and synchronization.

Multicore Computers:

- Hardware Performance Issues
- Software Performance Issues
- Multicore organization
- Intel Core i7-990x

Pipelining

- Pipelining is a technique of decomposing a sequential process into sub operations,
- with each sub process being executed in a special dedicated segment that operates concurrently with all other segments.
- **Throughput**: The amount of processing that can be accomplished during a given interval of time .

Parallel Processing

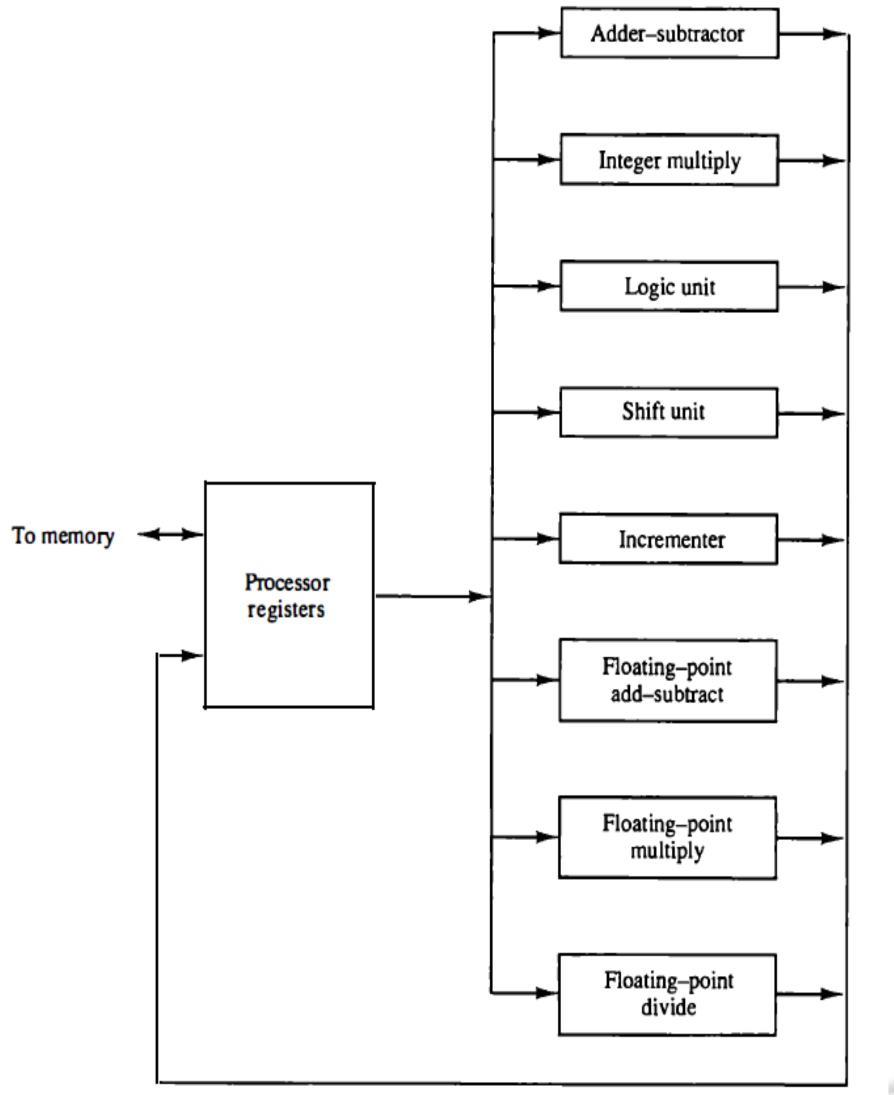
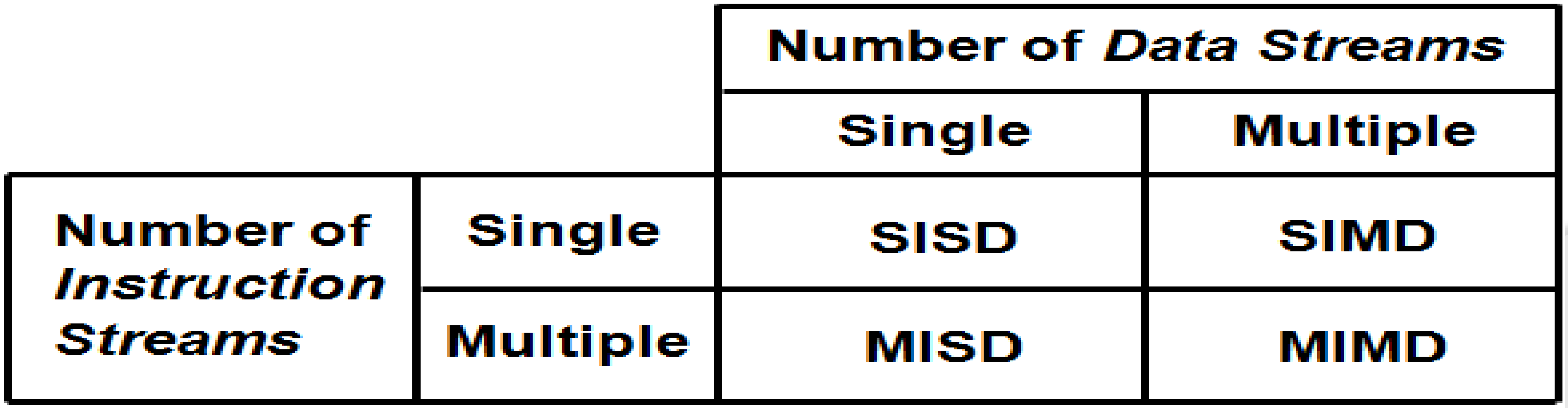


Fig: Processor with Multiple Functional Units

Parallel Processing

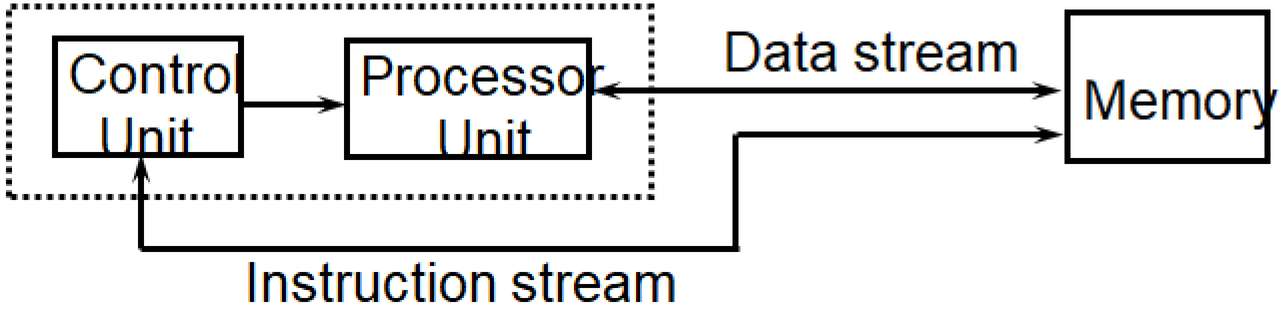
- M.J. Flynn Classify the parallel processing based on the number of instructions and data items that are manipulated simultaneously .
- Instruction Stream
  - Sequence of Instructions read from memory .
- Data Stream
  - Operations performed on the data in the processor .
- Flynn’s Classification divides computers into four major groups:



		Number of *Data Streams*	
		Single	Multiple
Number of *Instruction Streams*	Single	SISD	SIMD
	Multiple	MISD	MIMD

Parallel Processing

SISD Computer Systems

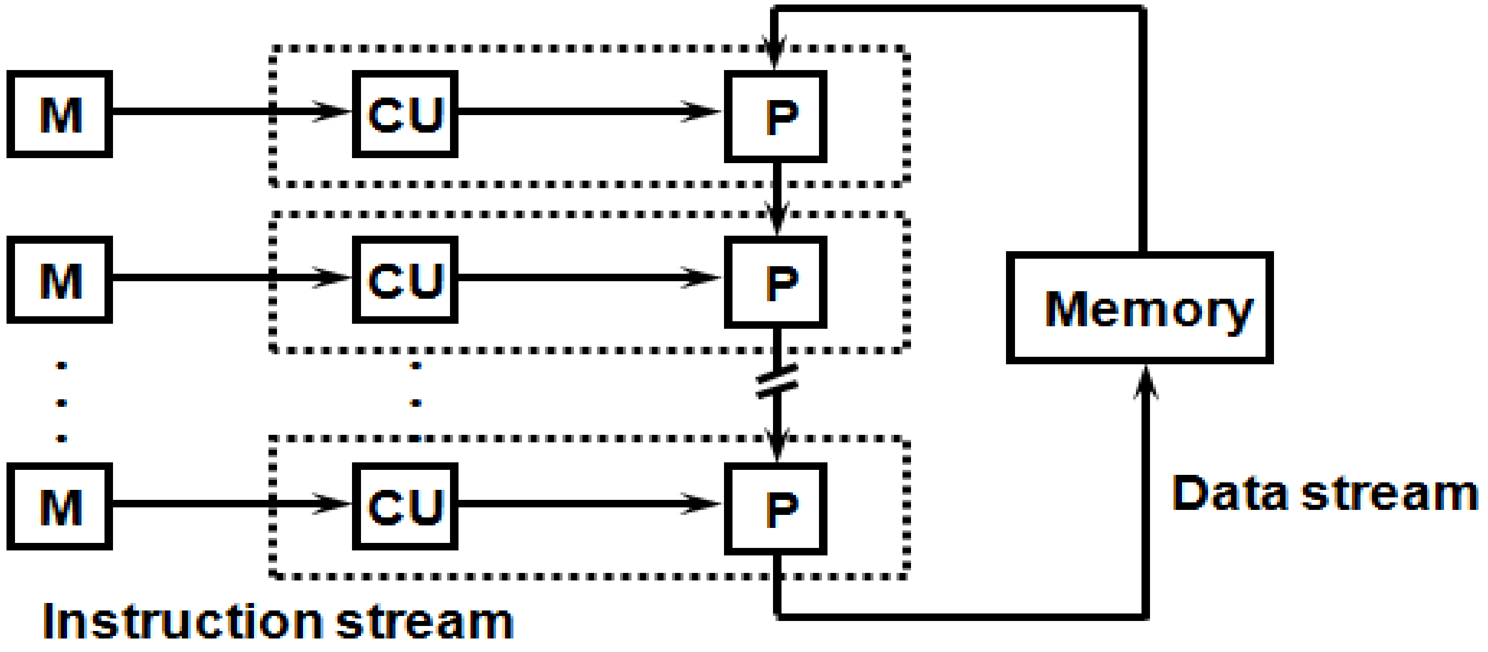


- Characteristics
  - Standard von Neumann machine
  - Instructions and data are stored in memory
  - One operation at a time
  - Parallel processing is achieved by means of multiple functional units or by pipeline.

- Limitations
  - Von Neumann bottleneck
  - Maximum speed of the system is limited by the *Memory Bandwidth*
  - Limitation on *Memory Bandwidth*
  - Memory is shared by CPU and I/O

Parallel Processing

MISD Computer System

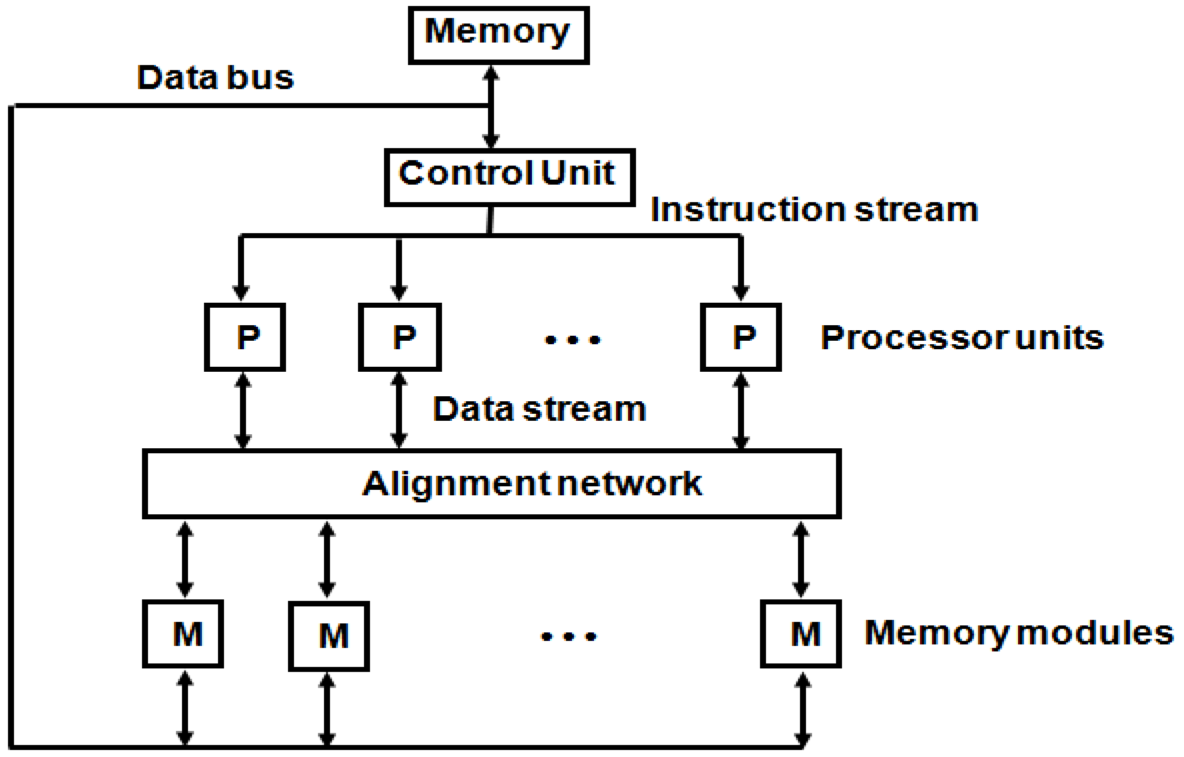


Characteristics

- There is no computer at present that can be classified as MISD .

Parallel Processing

SIMD Computer System

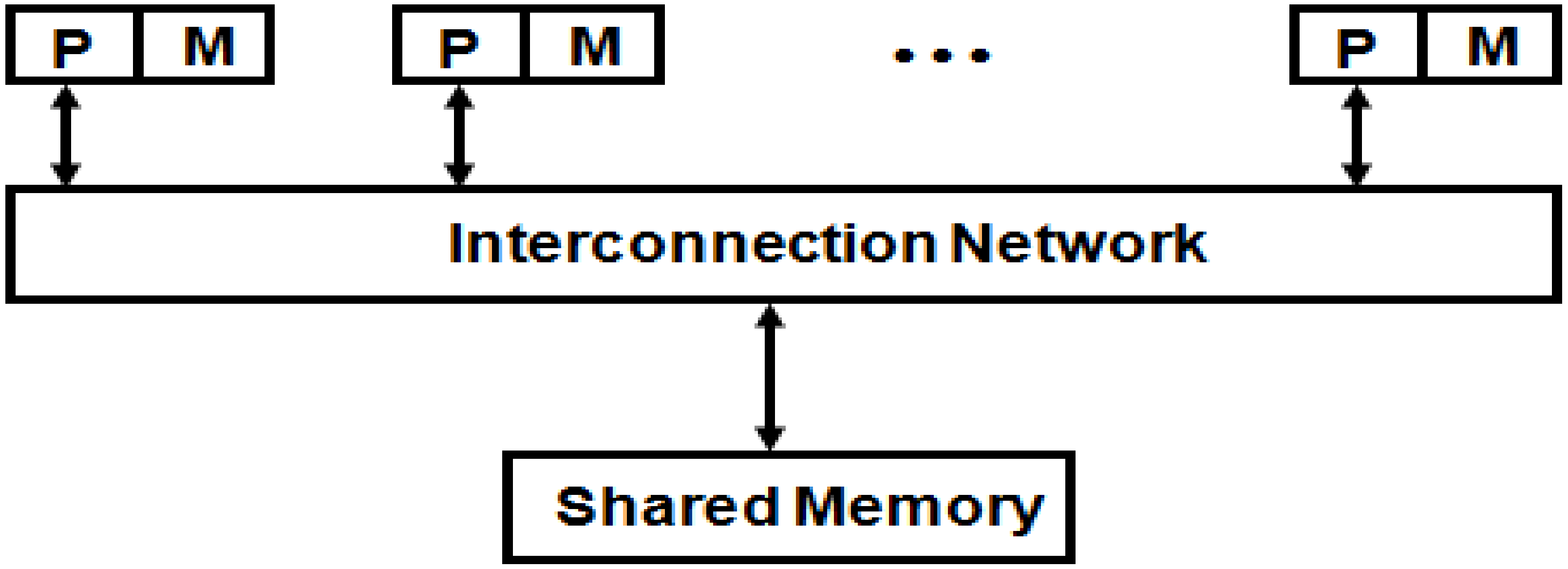


Characteristics

- Only one copy of the program exists
- A single controller executes one instruction at a time

Parallel Processing

MIMD Computer System



Characteristics

- Multiple processing units
- Execution of multiple instructions on multiple data.
- Multiprocessors and multi computers are MIMD computers.

Parallel Processing

- The parallel processing Is achieved by

  1)Pipeline Processing

  2) Vector Processing

  3) Array processors

Pipelining :

- A technique of decomposing a sequential process into sub operations, with each sub process being executed in a special dedicated segment that operates concurrently with all other segments.

Example

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

$R1 \leftarrow A_i,\ R2 \leftarrow B_i$	Load A_i and B_i
$R3 \leftarrow R1 * R2,\ R4 \leftarrow C_i$	Multiply and load C_i
$R5 \leftarrow R3 + R4$	Add

Pipelining

Pipelining

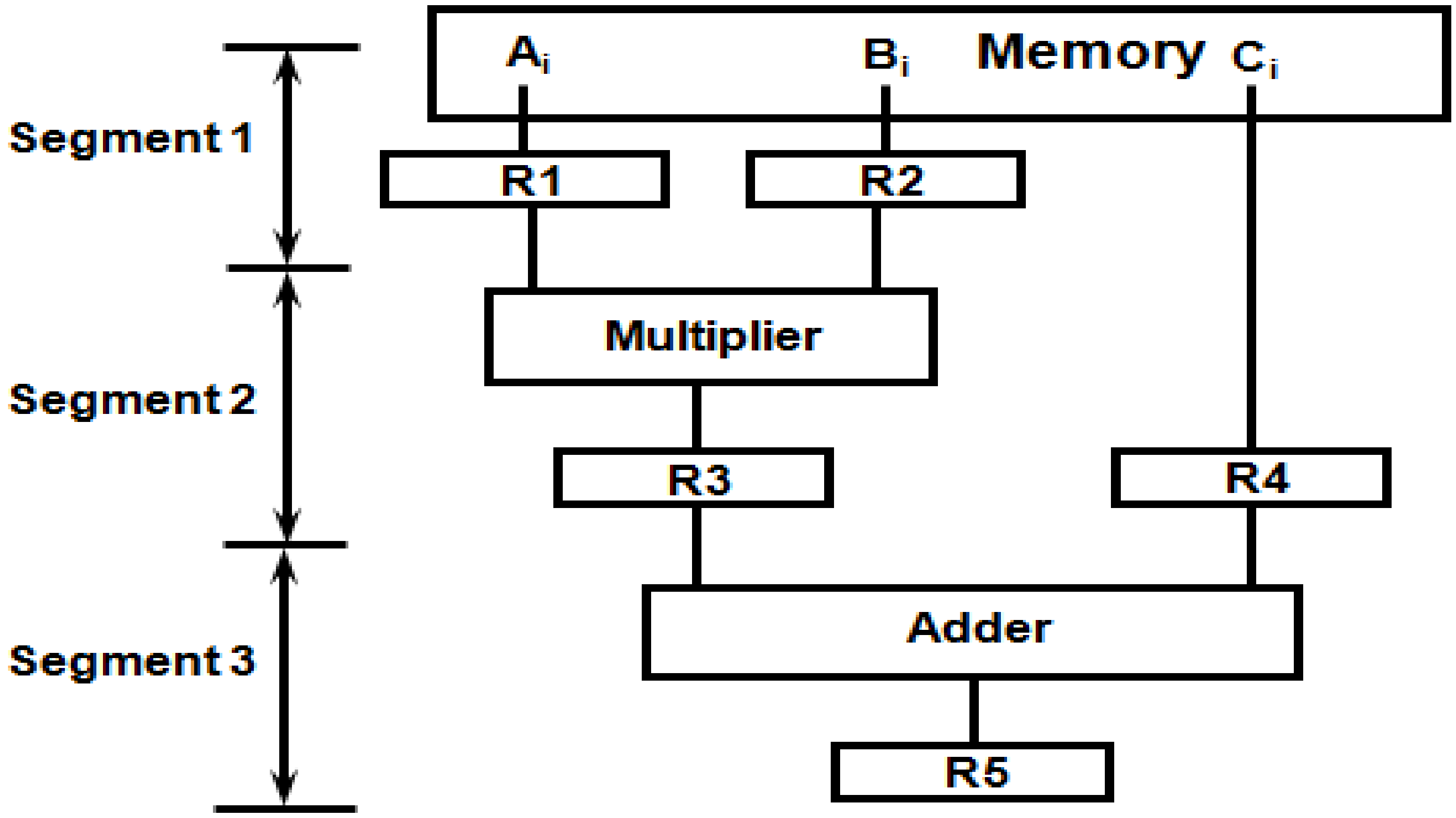
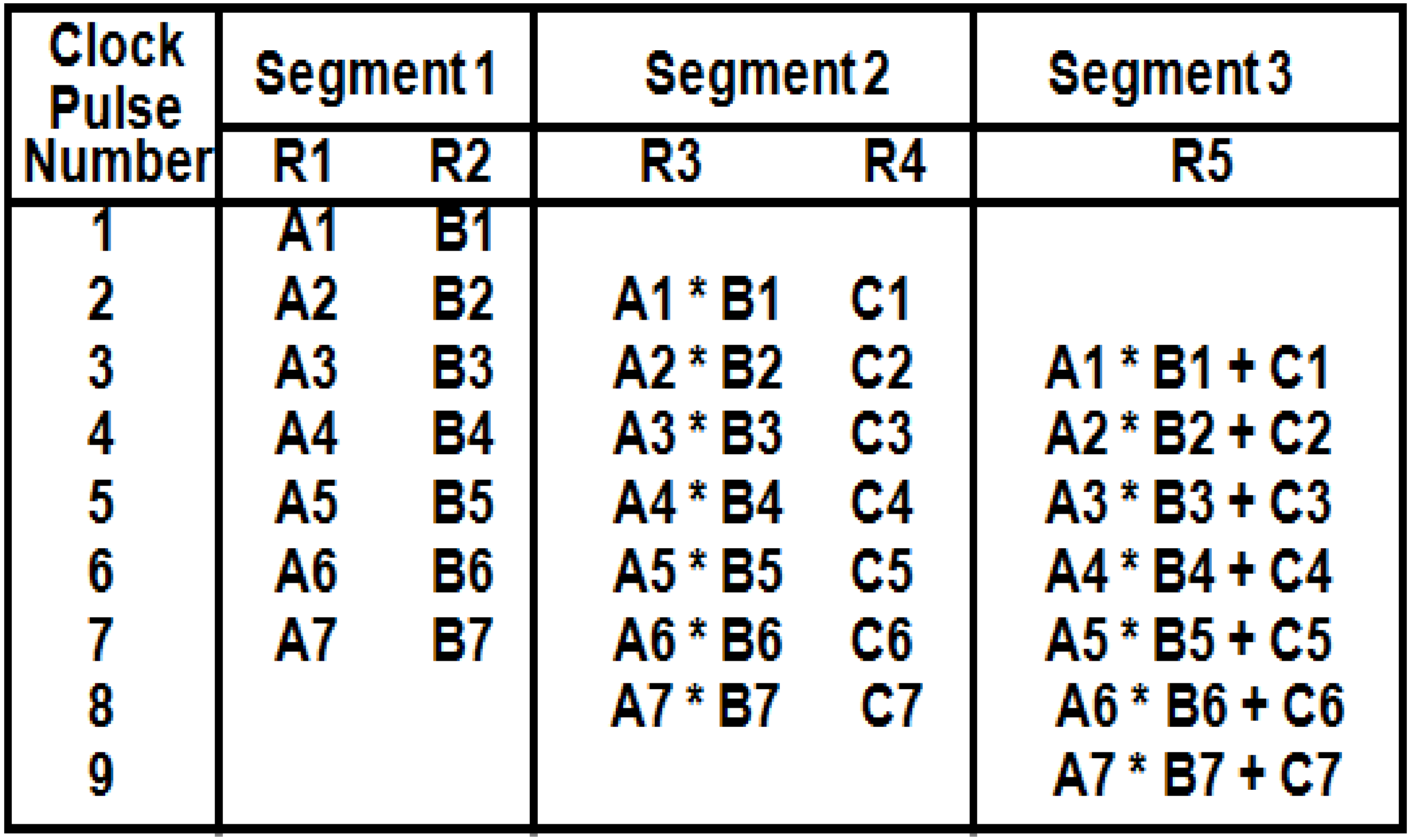


Fig: Pipe Line Processing

Pipelining



Clock Pulse Number	Segment 1		Segment 2		Segment 3
	R1	R2	R3	R4	R5
1	A1	B1			
2	A2	B2	A1 * B1	C1	
3	A3	B3	A2 * B2	C2	A1 * B1 + C1
4	A4	B4	A3 * B3	C3	A2 * B2 + C2
5	A5	B5	A4 * B4	C4	A3 * B3 + C3
6	A6	B6	A5 * B5	C5	A4 * B4 + C4
7	A7	B7	A6 * B6	C6	A5 * B5 + C5
8			A7 * B7	C7	A6 * B6 + C6
9					A7 * B7 + C7

Table: Content of register in pipeline

Pipelining

General Pipeline

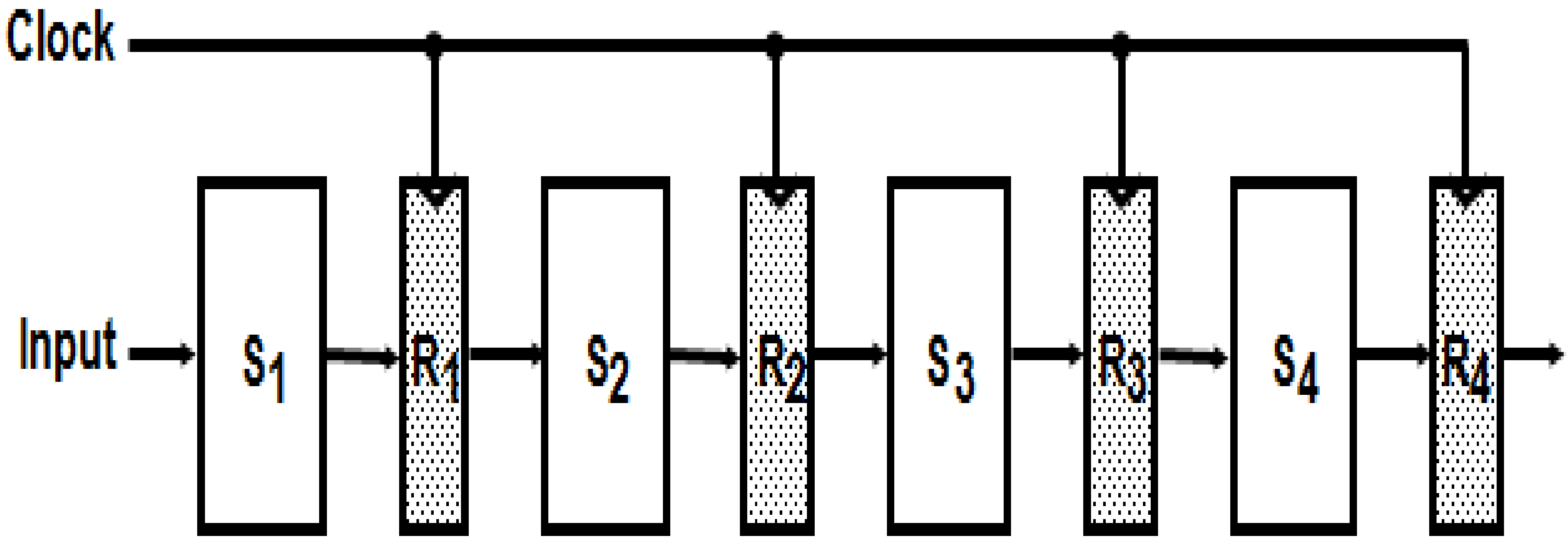
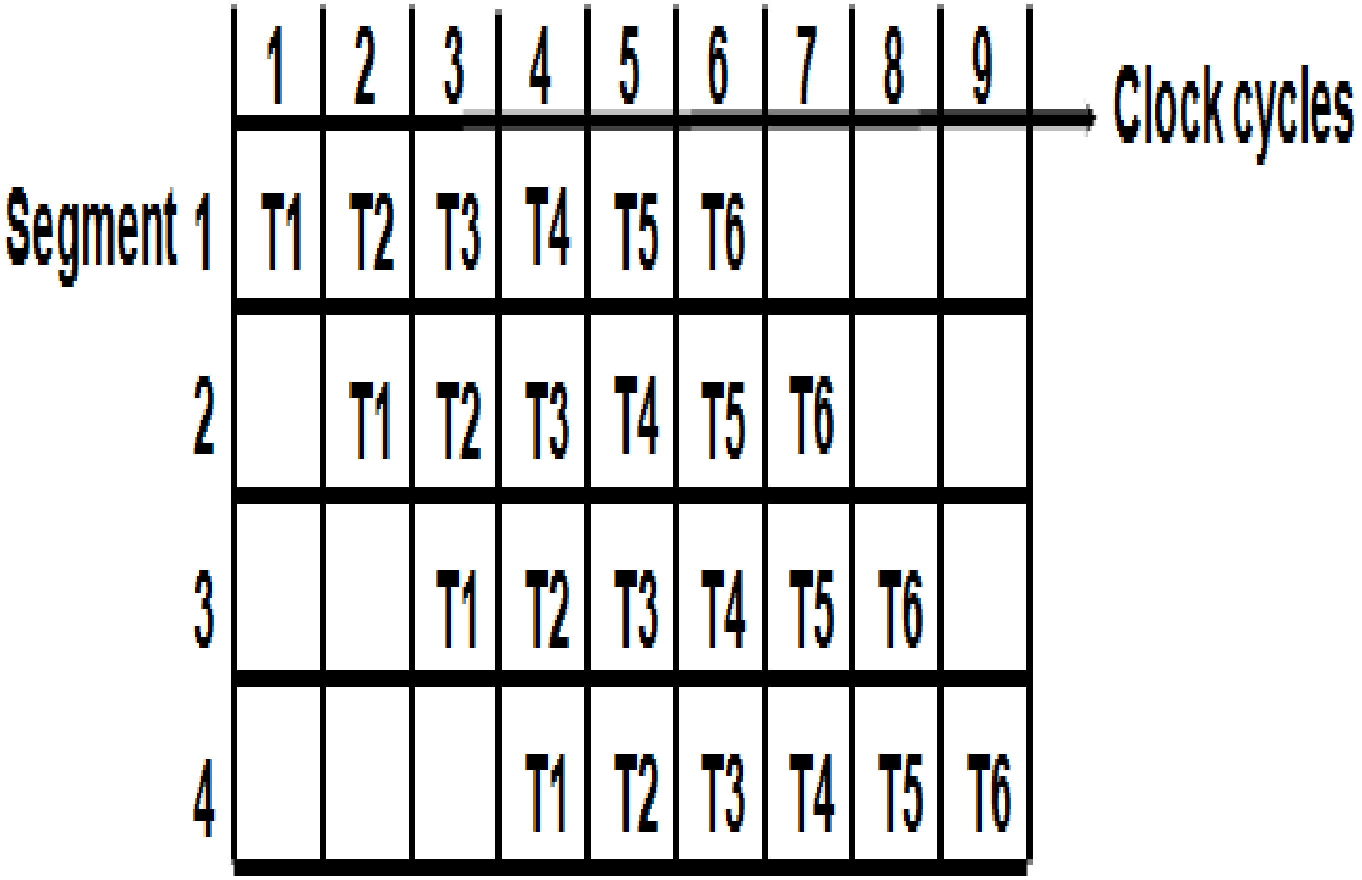


Fig : Four-Segment Pipeline

Pipelining

General Pipeline



Segment	1	2	3	4	5	6	7	8	9
Segment 1	T1	T2	T3	T4	T5	T6			
2		T1	T2	T3	T4	T5	T6		
3			T1	T2	T3	T4	T5	T6	
4				T1	T2	T3	T4	T5	T6

Clock cycles

Fig: Space-Time Diagram for pipeline

Pipelining

Pipeline Speedup

n: Number of tasks to be performed
K: number of segments

Conventional Machine (Non-Pipelined)

t_n: Clock cycle
t_1: Time required to complete the n tasks
$t_1 = n * t_n$

Pipelined Machine (k stages)

t_p: Clock cycle (time to complete each suboperation)
t_k: Time required to complete the n tasks
$t_k = (k + n - 1) * t_p$

Speedup

S_k: Speedup

$S_k = n*t_n / (k + n - 1)*t_p$

Pipelining

Pipeline Speedup

Example

- 4-stage pipeline
- sub opertion in each stage; $t_p = 20nS$
- 100 tasks to be executed
- 1 task in non-pipelined system; 20*4 = 80nS

Pipelined System

$$(k + n - 1)*t_p = (4 + 99) * 20 = 2060nS$$

Non-Pipelined System

$$n*k*t_p = 100 * 80 = 8000nS$$

Speedup

$$S_k = 8000 / 2060 = 3.88$$

4-Stage Pipeline is basically identical to the system with 4 identical function units

Pipelining

General Pipeline

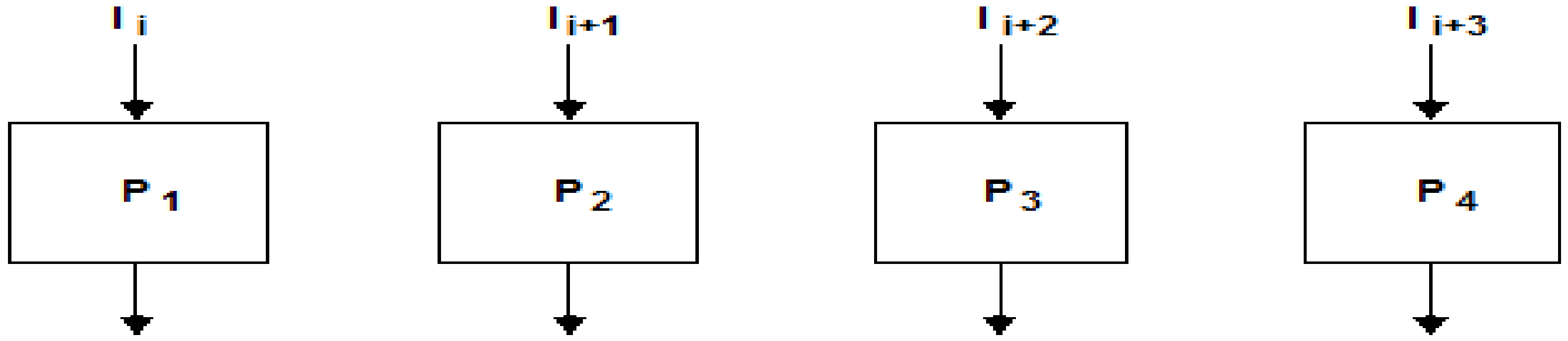


Fig: Multiple Functional Units

There are two are as of computer design where the pipeline organization is use full.

1. Arithmetic Pipeline (Not there in Syllabus)

2. Instruction Pipeline

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.
- This causes the instruction fetch and execute phases to overlap and perform simultaneous operations .
- One disadvantage with this scheme is that an instruction may cause a branch out of sequence. In this case the pipeline must be empted and all instructions read from memory after branch.
- A computer with an instruction fetch unit and an instruction execution unit designed to provide a two segment pipeline.

Instruction Pipeline

- In general any computer system needs six steps to process any instruction .

  1 . Fetch an instruction from memory

  2 . Decode the instruction

  3 . Calculate the effective address of the operand

  4 . Fetch the operands from memory

  5 . Execute the operation

  6 . Store the result in the proper place

- Some instructions skip some phases

  - Effective address calculation can be done in the part of the decoding phase .

  - Storage of the operation result into a register is done automatically in the execution phase .

Instruction Pipeline

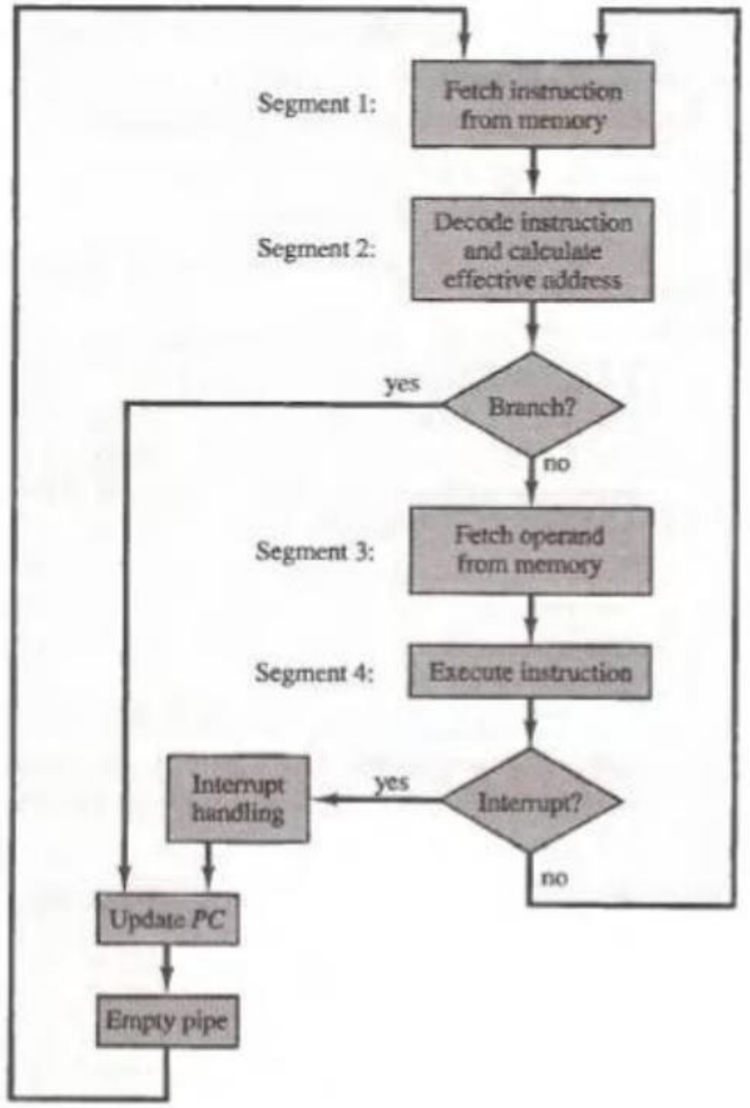


Fig: Four-segment CPU pipeline

Instruction Pipeline

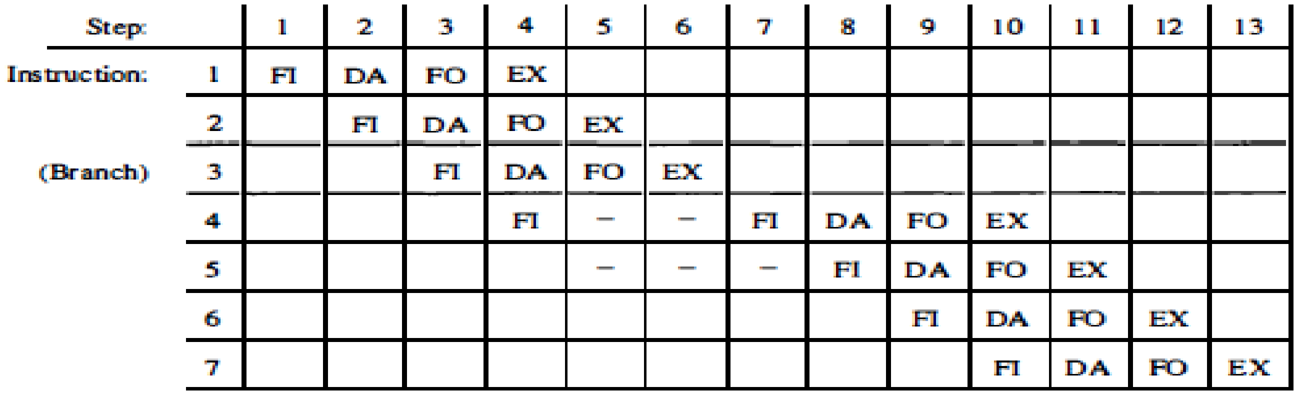
- Reduces the instruction pipeline into four segments. Figure shows how the instruction cycle in the CPU can be processed with a four-segment pipeline. While an instruction is being executed in segment 4, the next instruction in sequence is busy fetching an operand from memory in segment 3.
- The effective address may be calculated in a separate arithmetic circuit for the third instruction, and whenever the memory is available, the fourth and all subsequent instructions can be fetched and placed in an instruction FIFO.
- Thus up to four sub operations in the instruction cycle can overlap and up to four different instructions can be in progress of being processed at the same time.
- Figure shows the operation of the instruction 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.

Instruction Pipeline

1. FI is the segment that fetches an instruction.
2. DA is the segment that decodes the instruction and calculates the effective address.
3. FO is the segment that fetches the operand.
4. EX is the segment that executes the instruction.

- It is assumed that the processor has separate instruction and data memories so that the operation in FI and FO can proceed at the same time. In the absence of a branch instruction, each segment operates on different instructions. Thus, in step 4, instruction 1 is being executed in segment EX; the operand for instruction 2 is being fetched in segment FO; instruction 3 is being decoded in segment DA; and instruction 4 is being fetched from memory in segment FI.
- Assume now that instruction 3 is a branch instruction. As soon as this instruction is decoded in segment DA in step 4, the transfer from FI to DA of the other instructions is halted until the branch instruction is executed in step 6. If the branch is taken, a new instruction is fetched in step 7. If the branch is not taken, the instruction fetched previously in step 4 can be used. The pipeline then continues until a new branch instruction is encountered.

Instruction Pipeline



Step:		1	2	3	4	5	6	7	8	9	10	11	12	13
Instruction:	1	FI	DA	FO	EX									
	2		FI	DA	FO	EX								
(Branch)	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

Figure: Timing of instruction pipeline.

pipeline conflicts:

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

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

2. *Data dependency* conflicts arise when an instruction depends on the result of a previous instruction, but this result is not yet available.

3. *Branch difficulties* arise from branch and other instructions that change the value of PC .

Vector Processing

Vector processing is a central processing unit that can perform the complete vector input in individual instruction.

It is a complete unit of hardware resources that implements a sequential set of similar data elements in the memory using individual instruction.

Features of Vector Processing

There are various features of Vector Processing which are as follows −

A vector is a structured set of elements. The elements in a vector are scalar quantities. A vector operand includes an ordered set of n elements, where n is known as the length of the vector.

Each clock period processes two successive pairs of elements. During one single clock period, the dual vector pipes and the dual sets of vector functional units allow the processing of two pairs of elements.

Vector Processing

- In parallel vector processing, more than two results are generated per clock cycle. The parallel vector operations are automatically started under the following two circumstances –
    1. When successive vector instructions facilitate different functional units and multiple vector registers.
    2. When successive vector instructions use the resulting flow from one vector register as the operand of another operation utilizing a different functional unit. This phase is known as chaining.
- A vector processor implements better with higher vectors because of the foundation delay in a pipeline.
- Vector processing decrease the overhead related to maintenance of the loop-control variables which creates it more efficient than scalar processing.

Array Processor

Array Processor performs computations on large array of data. These are two types of Array Processors: Attached Array Processor, and SIMD Array Processor. These are explained as following below.

1. Attached Array Processor
2. SIMD array processor

Attached Array Processor

To improve the performance of the host computer in numerical computational tasks auxiliary processor is attached to it.

Attached array processor has two interfaces:

1. Input output interface to a common processor.
2. Interface with a local memory.

- Here local memory interconnects main memory. Host computer is general purpose computer. Attached processor is back end machine driven by the host computer.
- The array processor is connected through an I/O controller to the computer & the computer treats it as an external interface.

Array Processor

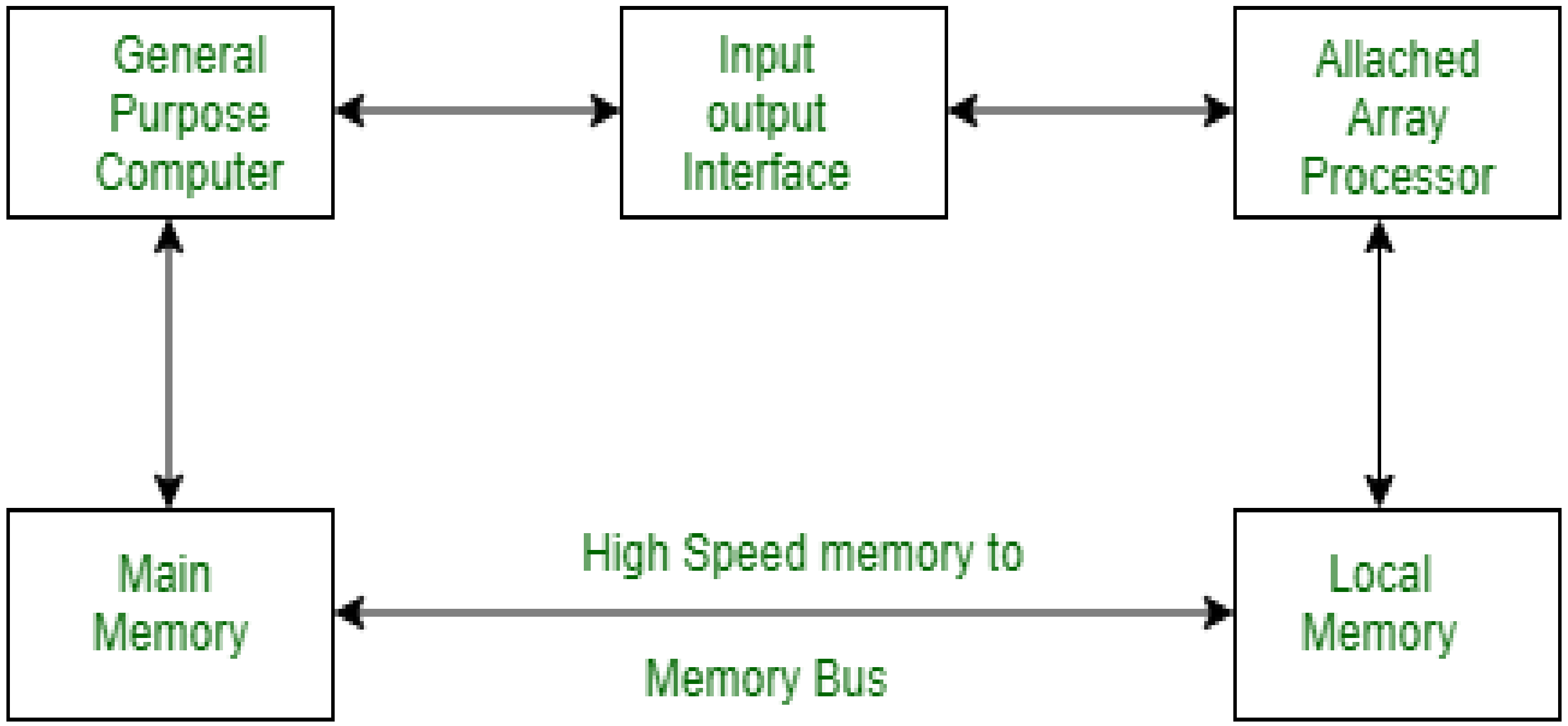


Figure - Interconnection of an attached array Processor to a host computer

Array Processor

SIMD array processor

- This is computer with multiple process unit operating in parallel Both types of array processors, manipulate vectors but their internal organization is different.
- SIMD is a computer with multiple processing units operating in parallel.
- The processing units are synchronized to perform the same operation under the control of a common control unit. Thus providing a single instruction stream, multiple data stream (SIMD) organization.
- As shown in figure, SIMD contains a set of identical processing elements (PES) each having a local memory M.

Array Processor

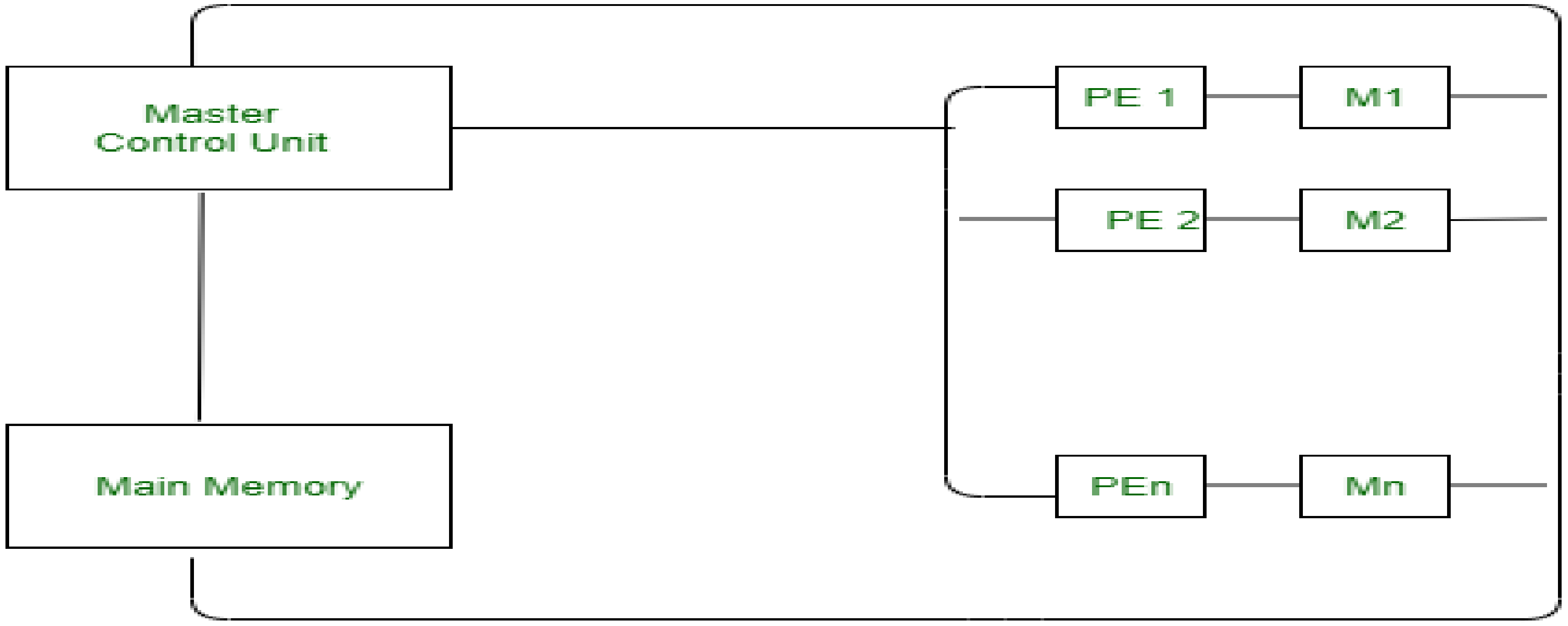


Figure - SIMD Array Processor Organization

Each PE includes –

- ALU
- Floating point arithmetic unit
- Working registers

Multiprocessors:

- Characteristics of multiprocessors
- Inter connection structures
- Inter processor arbitration
- Inter processor communication and synchronization.

Characteristics of Multiprocessors

Parallel Computing

- Simultaneous use of multiple processors, all components of a single architecture, to solve a task.
- Typically processors identical, single user .

Distributed Computing

- Use of a network of processors, each capable of being viewed as a computer in its own right, to solve a problem.
- Processors may be heterogeneous, multiuser, usually individual task is assigned to a single processors .

Concurrent Computing

- All of the above

Characteristics of Multiprocessors

Supercomputing

- Use of fastest, biggest machines to solve big, computationally intensive problems.
- Historically machines were vector computers, but parallel/vector or parallel becoming the norm.

Pipelining

- Breaking a task into steps performed by different units, and multiple inputs stream through the units, with next input starting in a unit when previous input done with the unit but not necessarily done with the task.

Vector Computing

- Use of vector processors, where operation such as multiply broken into several steps, and is applied to a stream of operands ("vectors"). Most common special case of pipelining.

Characteristics of Multiprocessors

Multiprocessor computer

- Execute a number of different application tasks in parallel
- Execute subtasks of a single large task in parallel
- All processors have access to all of the memory – shared-memory multiprocessor
- Cost – processors, memory units, complex interconnection networks

- Reliable, Single OS

Multicomputers

- Each computer only have access to its own memory
- Exchange message via a communication network – message-passing multicomputers

- Reliable ,Different OS

Characteristics of Multiprocessors

Multiprocessors and Multicomputers

- To achieve benefit of multiprocessing the computations can be processed in two ways

  1) multiple independent jobs can be made.

  2) A single job can be partitioned into multiple parallel tasks.

- Multiprocessors are classified into two types based on their memory organization.

1) Tightly Coupled System

 - Tasks and/or processors communicate in a highly synchronized fashion
 - Communicates through a common shared memory
 - Each processor may have cache memory.
 - Shared memory system
 - Efficient when higher degree of interaction between tasks.

Characteristics of Multiprocessors

Multiprocessors and Multicomputers

2) Loosely Coupled System

- Tasks or processors do not communicate in a synchronized fashion
- Each processor has its own private local memory.
- Information is shared among processor through message passing scheme.
- A packet consists of an address, the data content and some error detection code.
- Distributed memory system
- Efficient when the interaction between the tasks are minimal.

Characteristics of Multiprocessors

MEMORY

– Shared (Global) Memory

- A Global Memory Space accessible by all processors
- Processors may also have some local memory

– Distributed (Local, Message-Passing) Memory

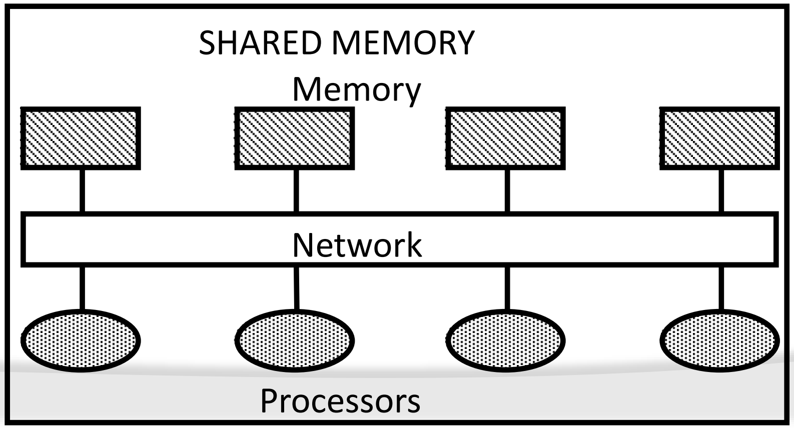
- All memory units are associated with processors
- To retrieve information from another processor's memory a message must be sent there

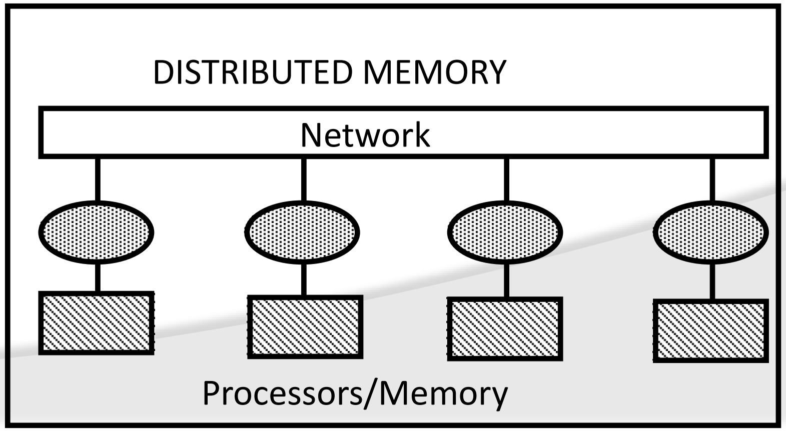
– Uniform Memory

- All processors take the same time to reach all memory locations

– Nonuniform (NUMA) Memory

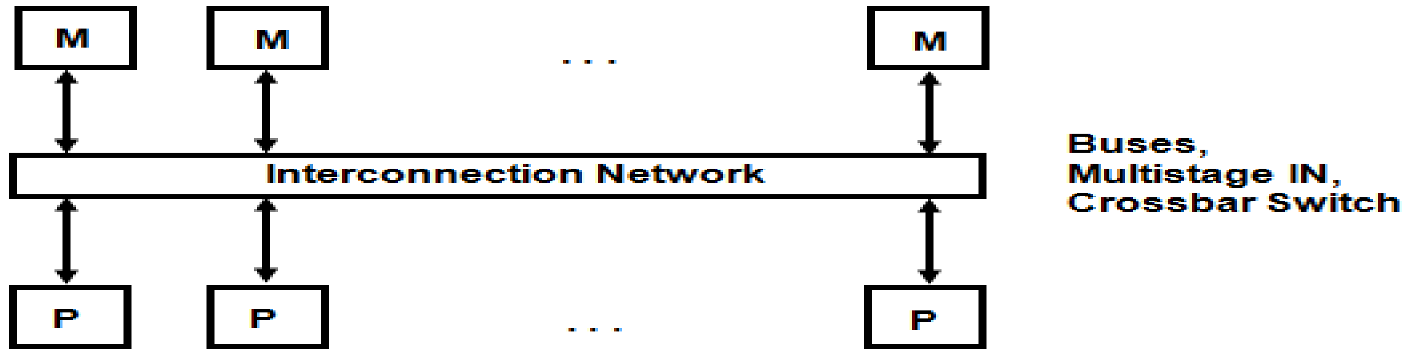
- Memory access is not uniform





Characteristics of Multiprocessors

Shared Memory Multiprocessors



Characteristics

- All processors have equally direct access to one
- large memory address space

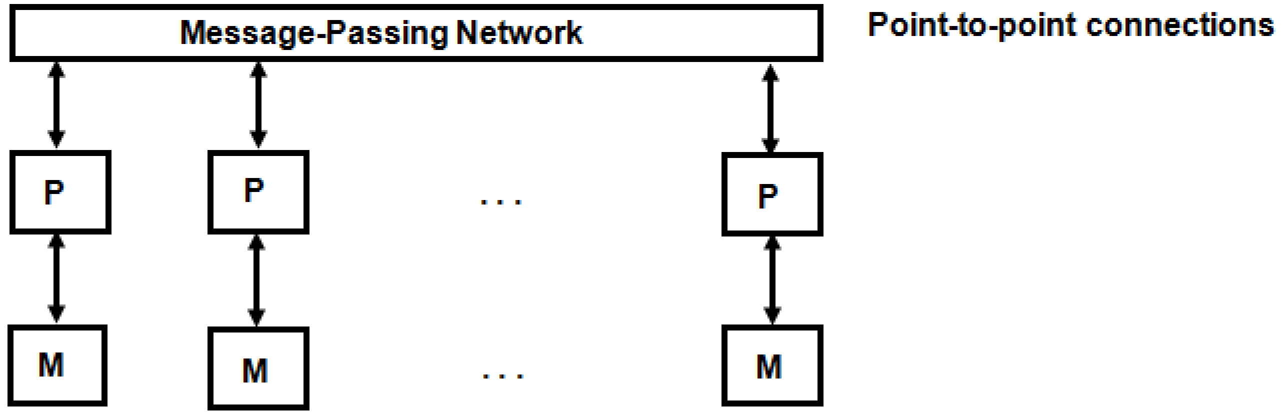
Example systems

- Bus and cache-based systems: Sequent Balance, Encore Multimax
- Multistage IN-based systems: Ultracomputer, Butterfly, RP3, HEP
- Crossbar switch-based systems: C.mmp, Alliant FX/8

Limitations

- Memory access latency; Hot spot problem

Message-Passing Multiprocessors



Characteristics

- Interconnected computers
- Each processor has its own memory, and communicate via message-passing

Example systems

- Tree structure: Teradata, DADO
- Mesh-connected: Rediflow, Series 2010, J-Machine
- Hypercube: Cosmic Cube, iPSC, NCUBE, FPS T Series, Mark III

Limitations

- Communication overhead; Hard to programming

Interconnection Structures

1) Time-Shared Common Bus
2) Multiport Memory
3) Crossbar Switch
4) Multistage Switching Network
5) Hypercube System

Time-Shared Common Bus

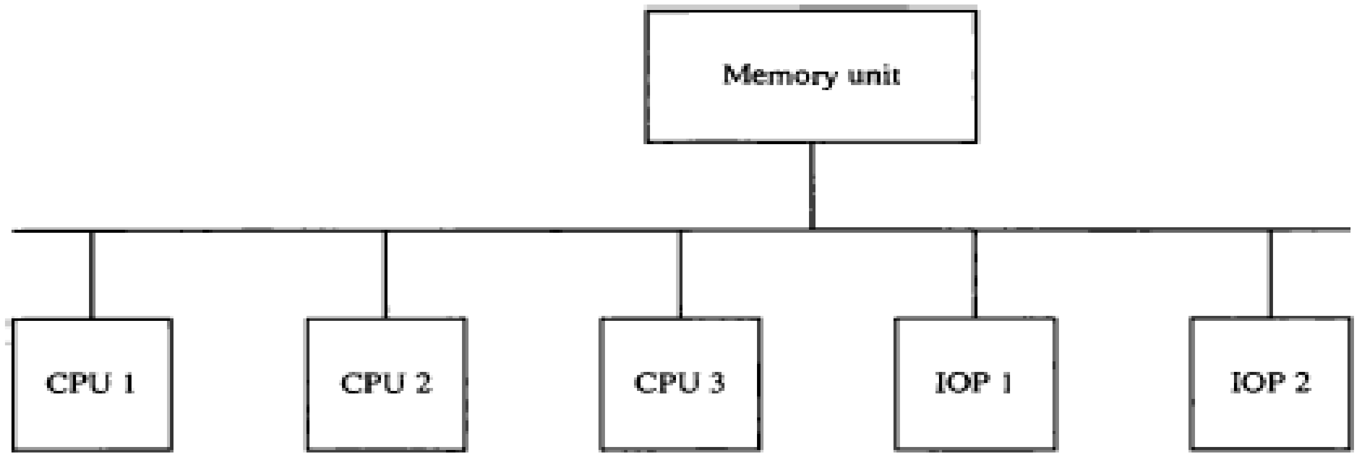


Figure: **Time-shared** common bus organization.

- A common-bus multiprocessor system consists of a number of processors connected through a common path to a memory unit. A time-shared common bus for five processors is shown in Fig. Only one processor can communicate with the memory or another processor at any given time.

Interconnection Structures

Time-Shared Common Bus

- Transfer operations are conducted by the processor that is in control of the bus at the time. Any other processor wishing to initiate a transfer must first determine the availability status of the bus, and only after the bus becomes available can the processor address the destination unit to initiate the transfer.
- A command is issued to inform the destination unit what operation is to be performed . The receiving unit recognizes its address in the bus and responds to the control signals from the sender, after which the transfer is initiated.
- A single common-bus system is restricted to one transfer at a time

Interconnection Structures

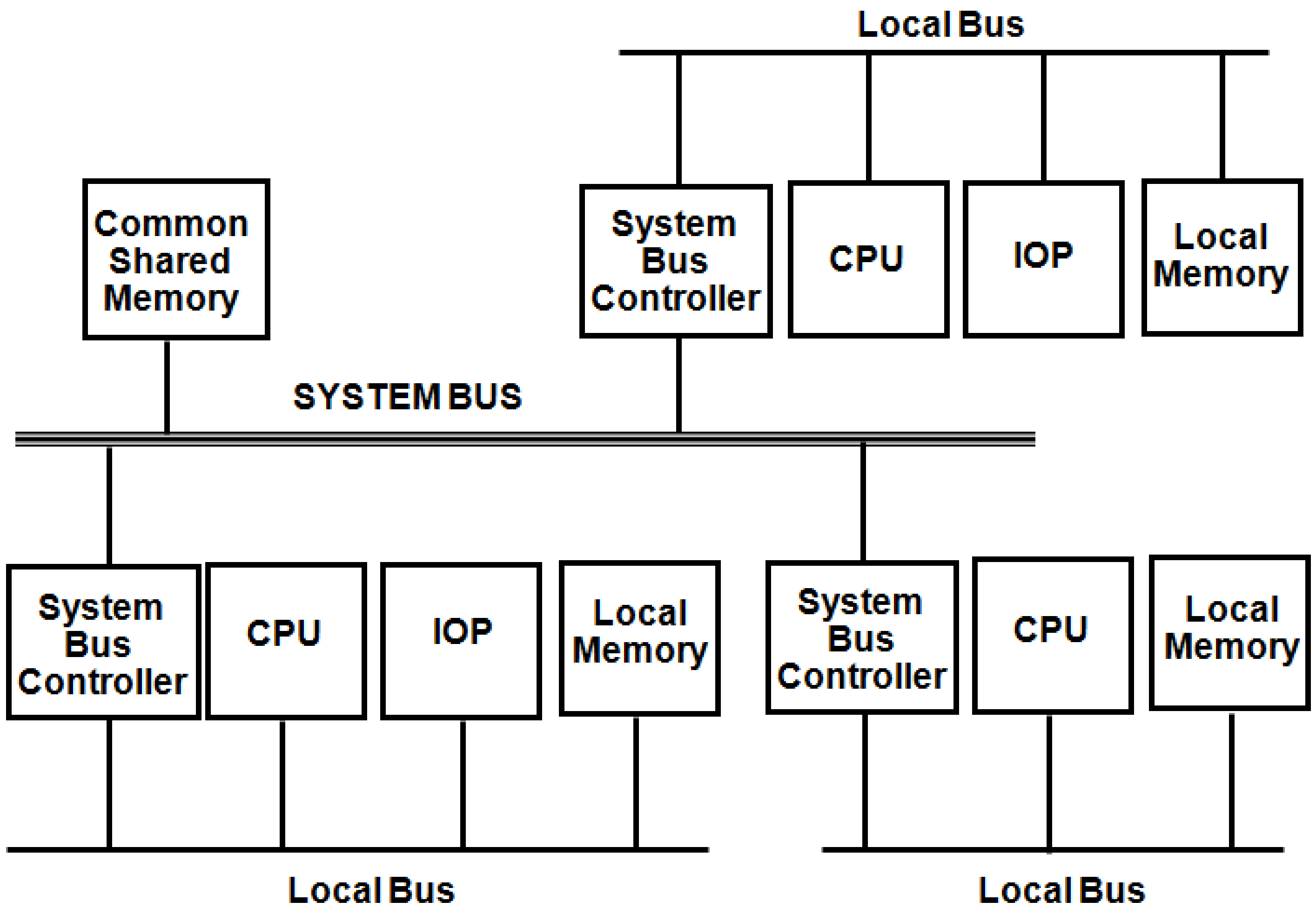


Fig : system bus structure for multiprocessors

Interconnection Structures

- Here we have a number of local buses each connected to its own local memory and to one or more processors. Each local bus may be connected to a CPU, an IOP, or any combination of processors. A system bus controller links each local bus to a common system bus . The IO devices connected to the local IOP, as well as the local memory, are available to the local processor.
- The memory connected to the common system bus is shared by all processors. If an IOP is connected directly to the system bus, the VO devices attached to it may be made available to all processors. Only one processor can communicate with the shared memory and other common resources through the system bus at any given time.
- The other processors are kept busy communicating with their local memory and IO devices. Part of the local memory may be designed as a cache memory attached to the CPU in this way, the average access time of the local memory can be made to approach the cycle time of the CPU to which it is attached.

Interconnection Structures

Multiport Memory

- A multiport memory system employs separate buses between each memory module and each CPU.
- for four CPUs and four memory modules (MMs). Each processor bus is connected to each memory module. A processor bus consists of the address, data, and control lines required to communicate with memory. The memory module is said to have four ports and each port accommodates one of the buses.
- Thus CPU 1 will have priority over CPU 2, CPU 2 will have priority over CPU 3, and CPU 4 will have the lowest priority.
- The advantage of the multi port memory organization is the higher rate that can be achieved because of the multiple paths between processors and memory. The disadvantage is that it requires expensive memory control logic and a large number of cables and connectors.

Interconnection Structures

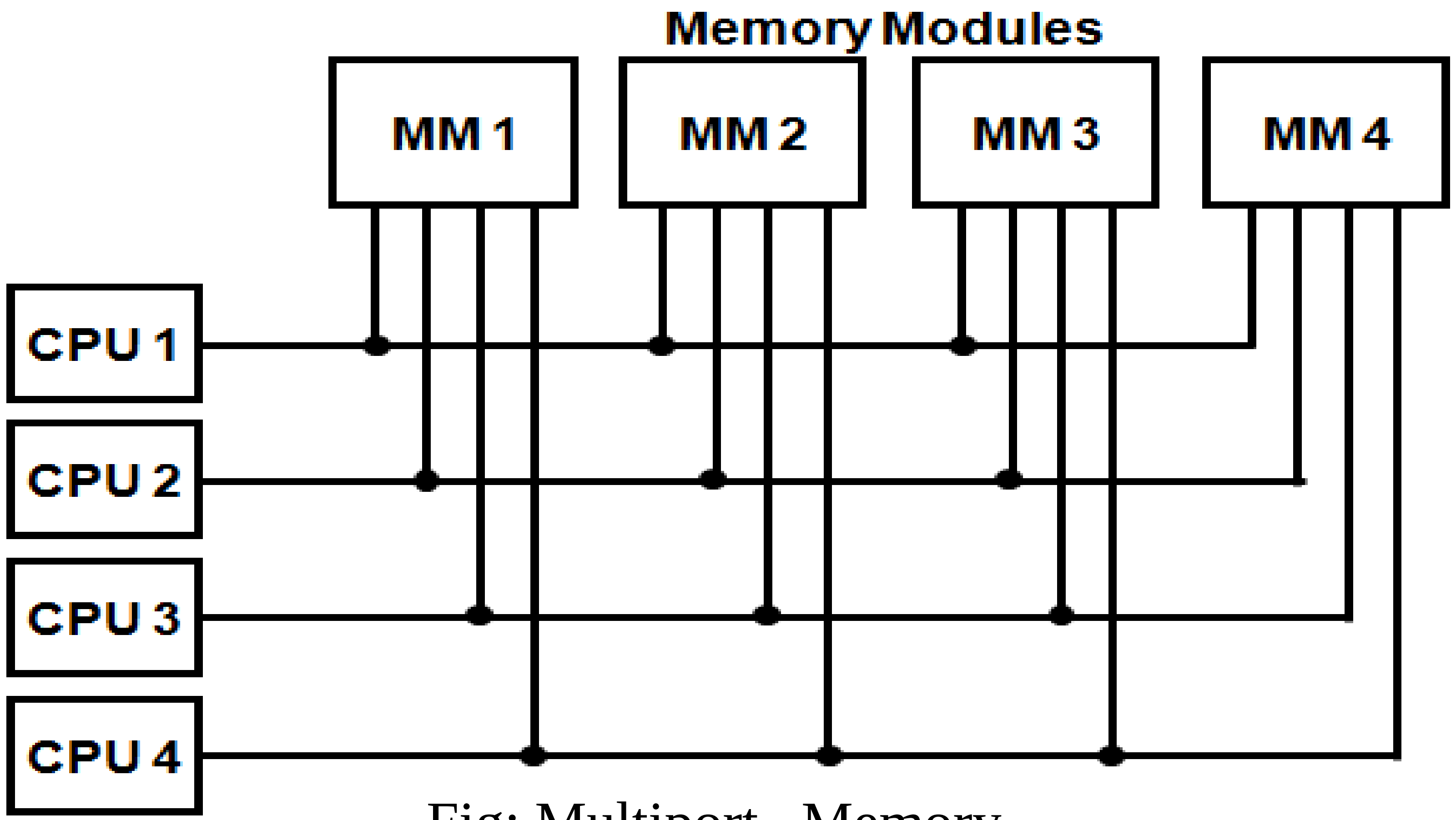


Fig: Multiport Memory

Interconnection Structures

Crossbar Switch

- The cross bar switch consists of a number of cross points that are placed at intersection between processor bus and memory module paths.
- The small square in each cross point is a switch that determines the path from a processor to memory module.
- Each switch contains control logic to set up the transfer path between a processor and memory.
- It determines the address and it resolves the multiple requests for access to the same memory module on a predetermined priority basis.
- A cross bar switch organization support simultaneous transfers from all memory modules.

Interconnection Structures

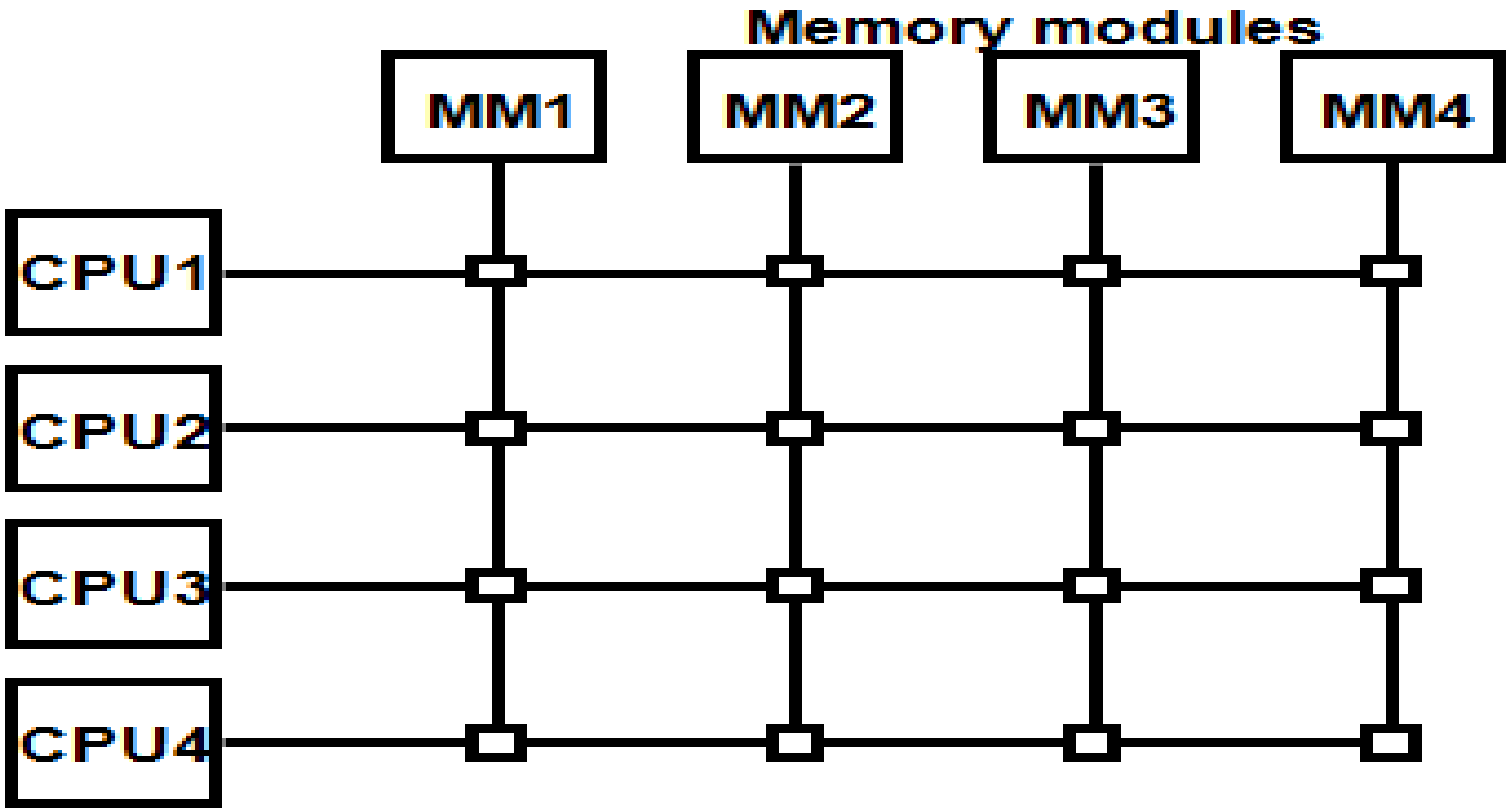


Fig : Crossbar Switch

Interconnection Structures

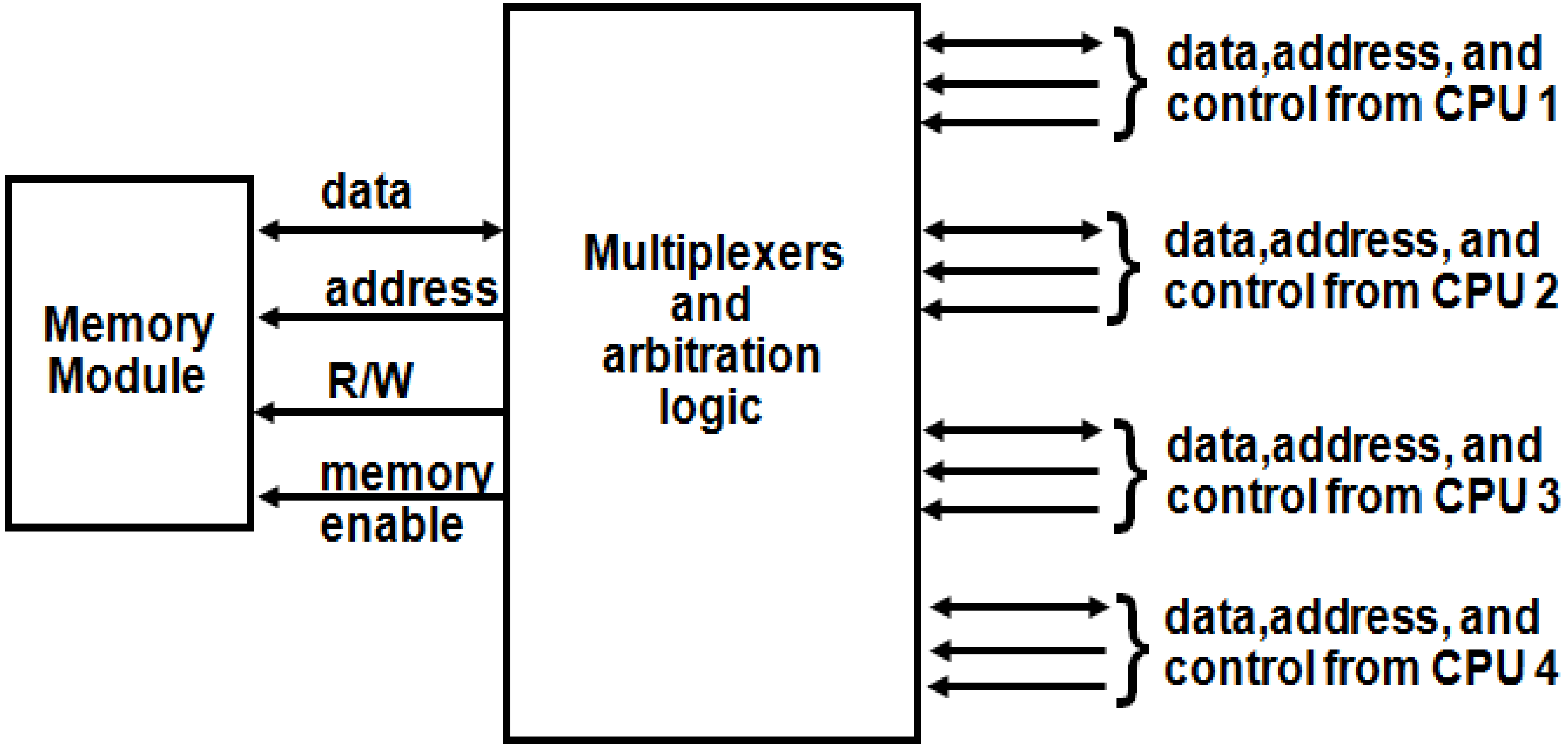

data
address
R/W
memory enable
Memory Module
Multiplexers and arbitration logic
data,address, and control from CPU 1
data,address, and control from CPU 2
data,address, and control from CPU 3
data,address, and control from CPU 4

Fig: Block Diagram of Crossbar Switch

Interconnection Structures

Multistage Switching Network

- The basic component of a multistage network is a two input , two output inter change switch.
- The 2 X 2 Switch has two inputs and A and B and Two outputs 0 and 1.

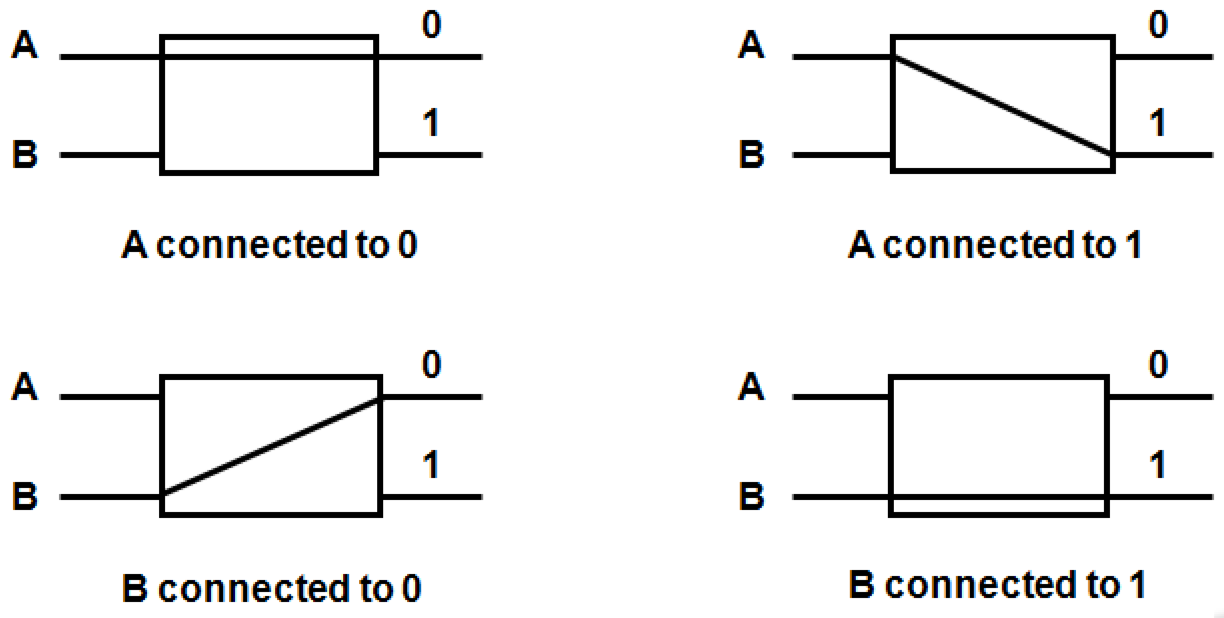
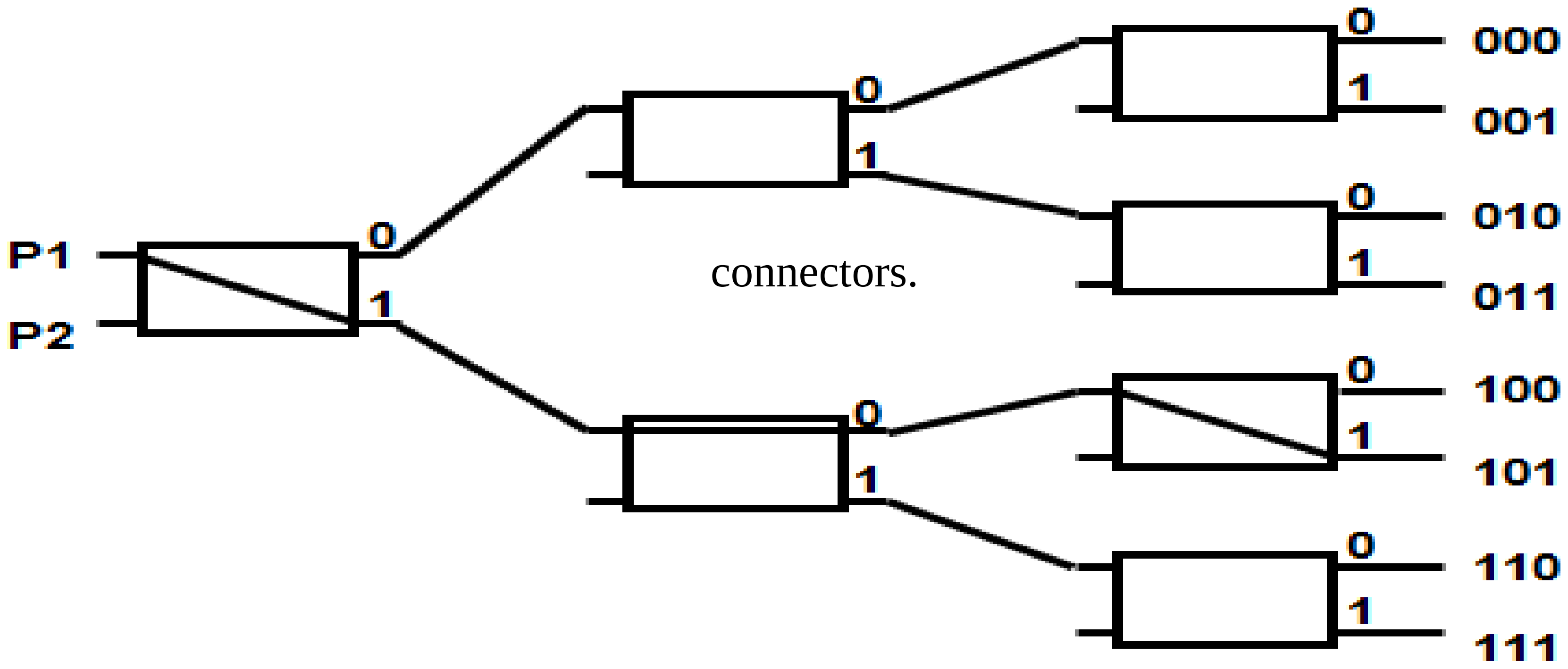


Fig : operation of 2 X 2 Switch

Interconnection Structures



connectors.

Fig : Binary tree with 2 X 2 switch

Interconnection Structures

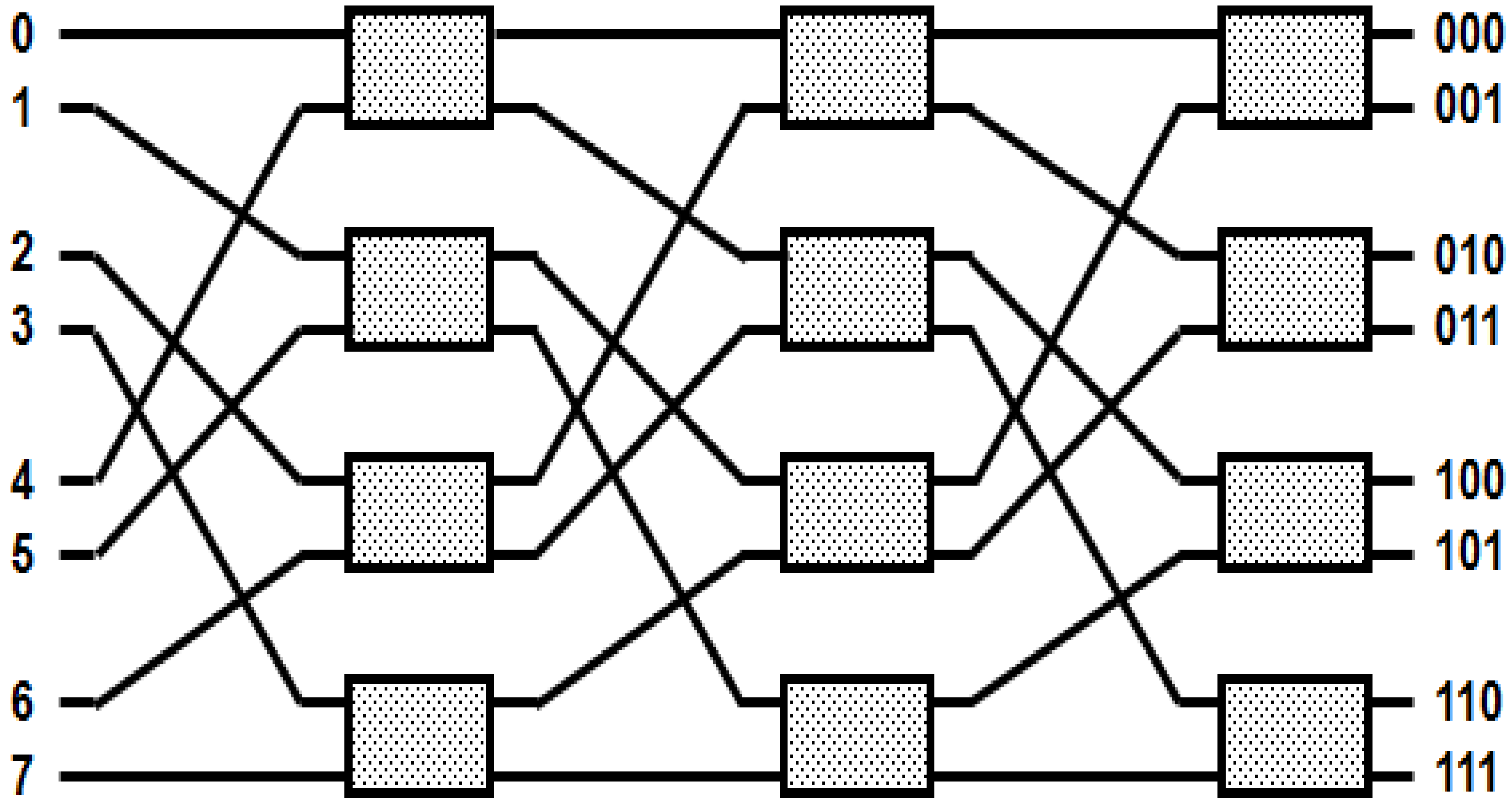


Fig : 8 X 8 Omega Switching Network

Interconnection Structures

Hypercube Interconnection

- 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 each node it contains processor ,memory and I/O Interface.
- Each processor has direct communication paths to n other neighbor processors.
- There are 2^n distinct n-bit binary address that can be assigned to the processors.
- Each processor address differs from that of each of its n neighbors by exactly one bit position.
- The routing procedure developed by using XOR operation.

Interconnection Structures

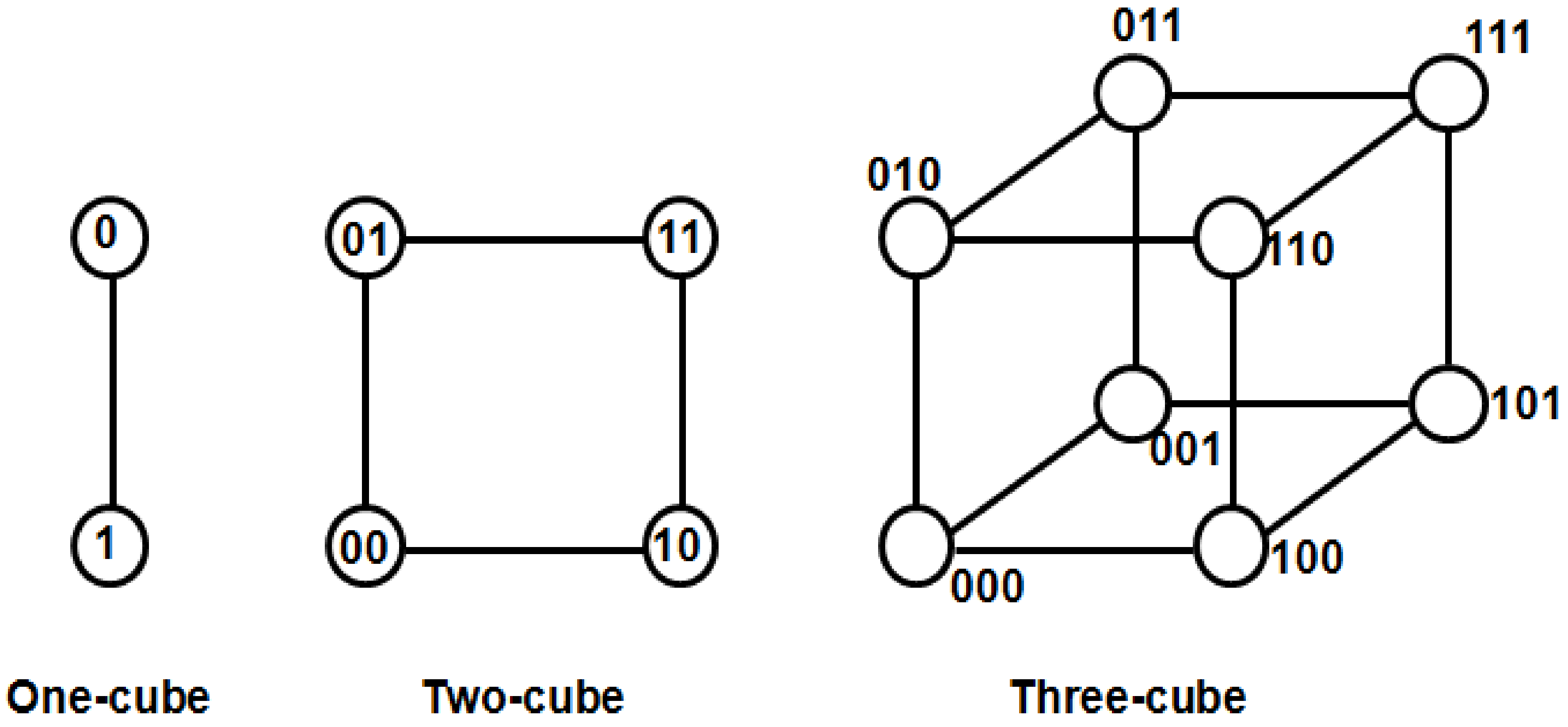


Fig: Hypercube Structure for n=1,2,3

Inter Processor Arbitration

- There are two arbitrations

1. Serial (daisy-chain) arbitration.
2. Parallel Arbitration

Serial (daisy-chain) arbitration.

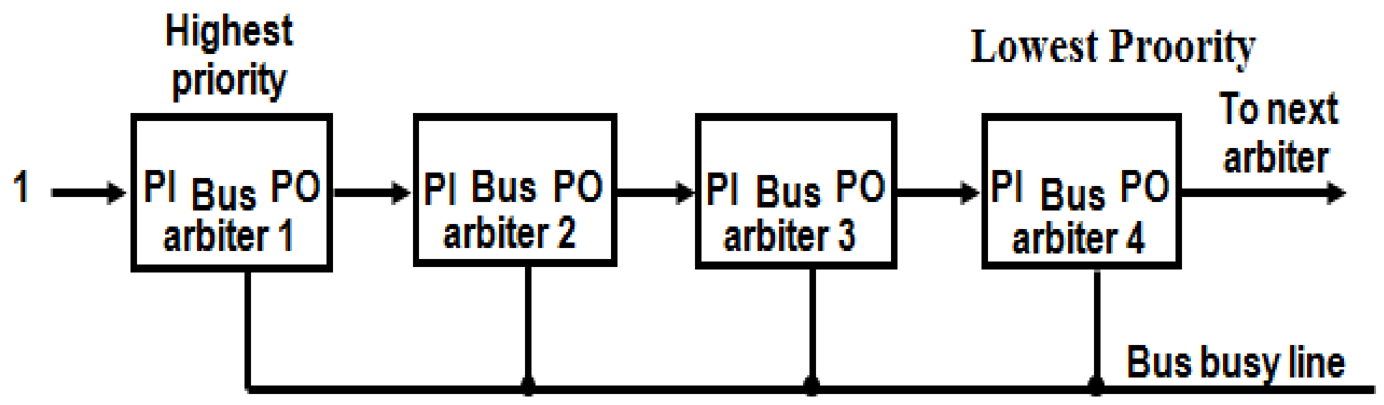


Fig : Serial Arbitration

Inter Processor Arbitration

- The processors connected to the system bus are assigned priority according to their position along the priority control line.
- The device closest to the priority line is assigned the highest priority.
- When multiple devices concurrently request the use of the bus , the device with the highest priority is granted access to it.
- In the above figure shows the daisy chain connection of four arbiters. Each processor contains its own arbiter logic with priority in and priority out .
- The Priority out of each processor (PO) is connected to the priority in (PI) of the next level priority arbiter.
- The PI of the highest-priority unit is maintained at a logic 1 value.

Inter Processor Arbitration

- The PO output for a particular arbiter is equal to 1 if its PI input is equal to 1 and the processor associated with the arbiter logic is not requesting control of the bus.
- If the processor requests control of the bus and the corresponding arbiter finds its PI input equal to 1 , it sets its PO output to 0.
- Lower-priority arbiters receive a 0 in PI and generates a 0 in PO.
- The processor whose arbiter has a PI=1 and PO =0 is the one that is given the control of the system bus.
- A processor may be in the middle of a bus operation when a higher priority processor request the bus. The lower-priority processor must complete its bus operation before it release control of the bus.

Inter Processor Arbitration

Parallel Arbitration Logic

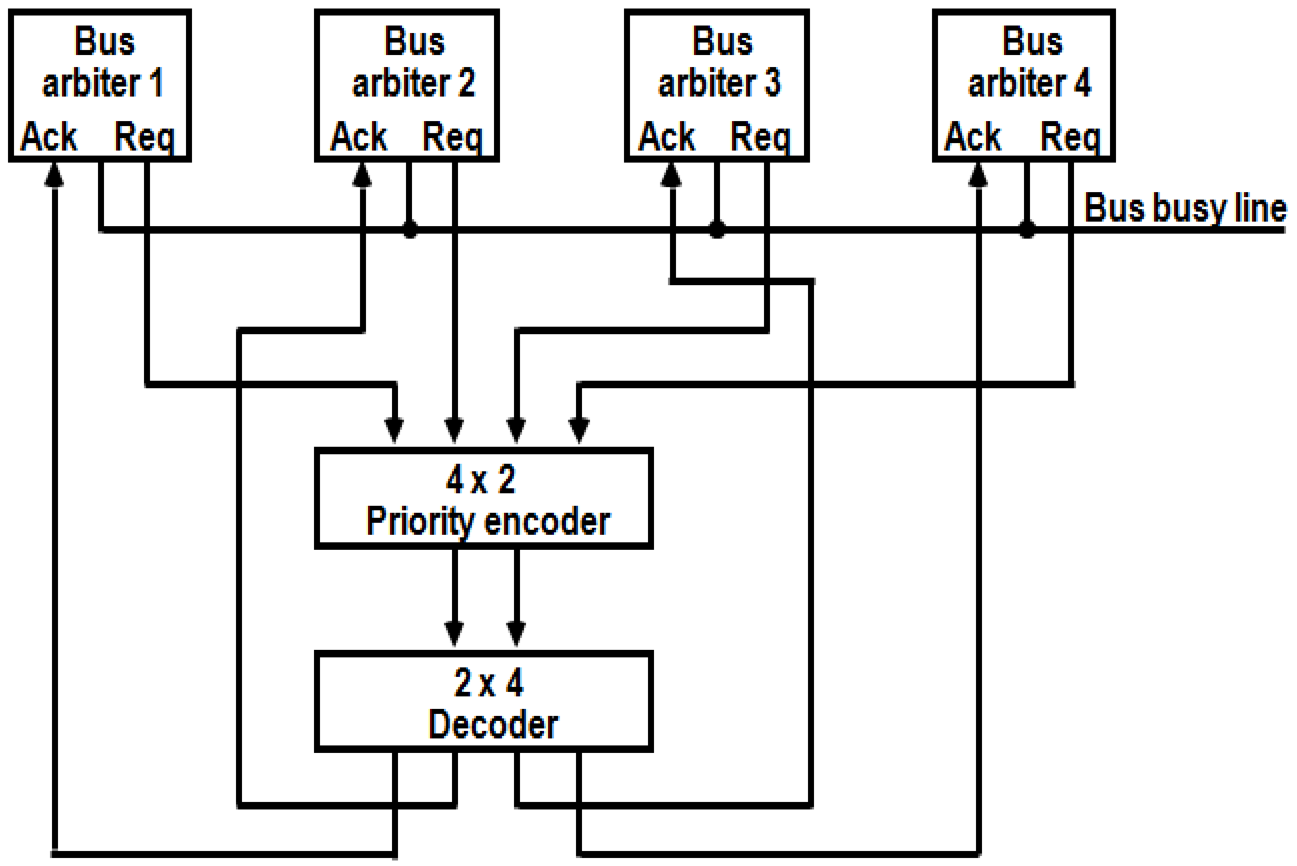


Fig: Parallel Arbitration

Inter Processor Arbitration

- The parallel bus arbitration technique uses an external priority encoder and a decoder.
- Each bus arbiter in the parallel scheme has a bus request output line and a bus acknowledge input line.
- Each arbiter enables the request line when its processor is requesting access to the system bus.
- The processor takes control of the bus if its acknowledge input line is enabled.

Inter processor communication and synchronization

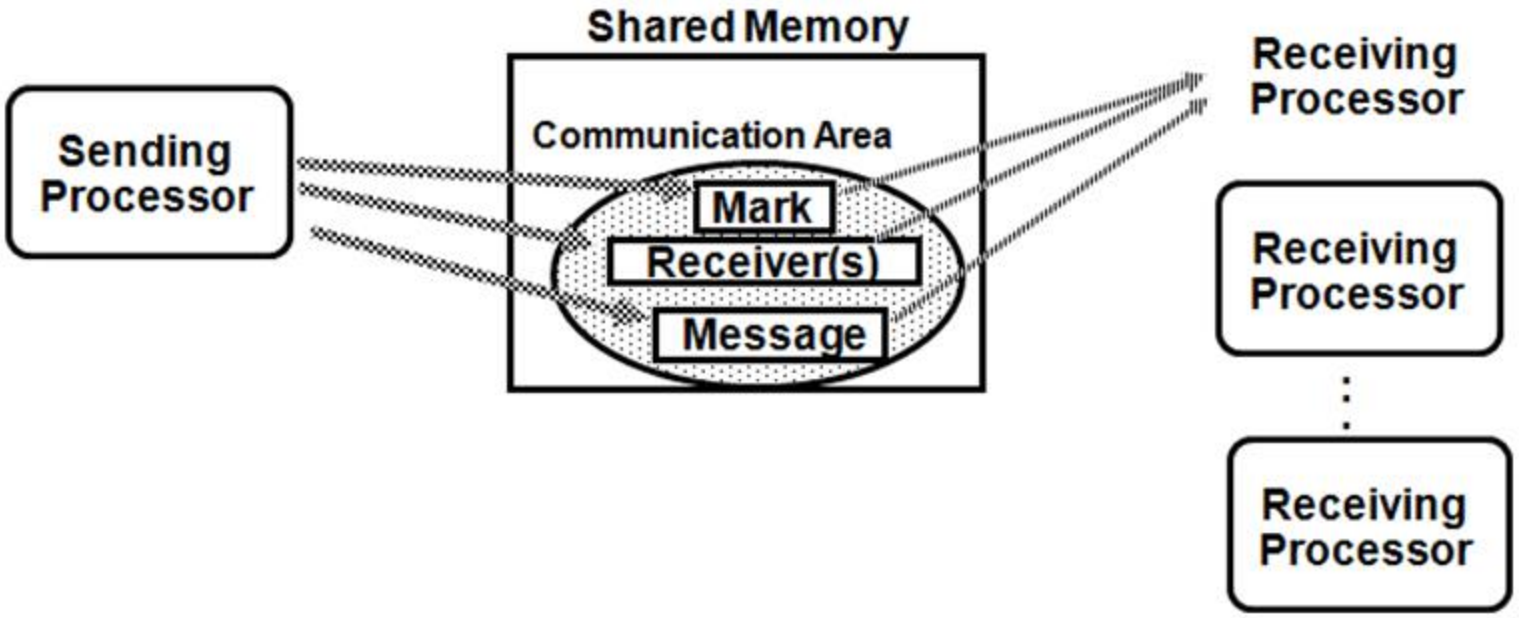


Fig: Shared Memory Communication

Inter processor communication and synchronization

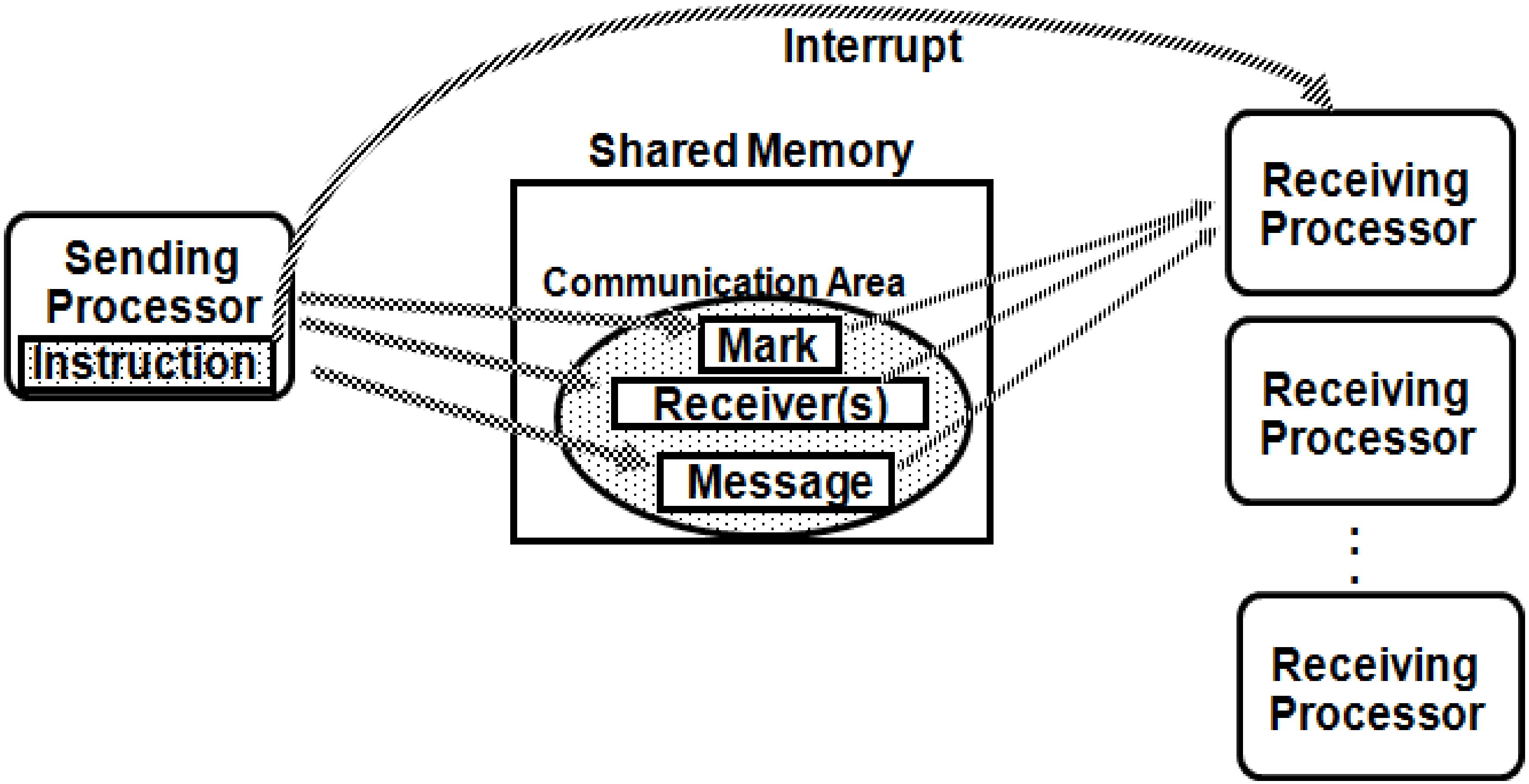


Fig: Shared Memory Communication with Interrupt

Inter processor communication and synchronization

Interprocessor Synchronization

In multiprocessor system communication refers to the exchange of data between different processes.

Synchronization is the process of where the data used to communicate between processor is control information.

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

Mutual Exclusion with a Semaphore

Mutual Exclusion

One processor to exclude or lock out access to shared resource by other processors when it is in a *Critical Section* .

Critical Section

is a program sequence that once begun, must Complete execution before another processor accesses the same shared resource.

Inter processor communication and synchronization

Semaphore

A binary variable

- 1: A processor is executing a critical section, that not available to other processors .
- 0: Available to any requesting processor

Software controlled Flag that is stored in memory that all processors can be access .

Testing and Setting the Semaphore

Avoid two or more processors test or set the same semaphore

May cause two or more processors enter the same critical section at the same time .

Must be implemented with an indivisible operation

```
R <- M[SEM]          / Test semaphore /
M[SEM] <- 1          / Set semaphore /
```

Inter processor communication and synchronization

- These are being done while *lock*ed, so that other processors cannot test and set while current processor is being executing these instructions.

- If R=1, another processor is executing the critical section, the processor executed this instruction does not access the shared memory .

- If R=0, available for access, set the semaphore to 1 and access The last instruction in the program must clear the semaphore .

Inter processor communication and synchronization

- Communication of control information between processors to enforce the correct sequence of processes
- To ensure mutually exclusive access to shared writable data
- Hardware Implementation
- Mutual Exclusion with a Semaphore

Multicore Computers:

- Hardware Performance Issues
- Software Performance Issues
- Multicore organization
- Intel Core i7-990x

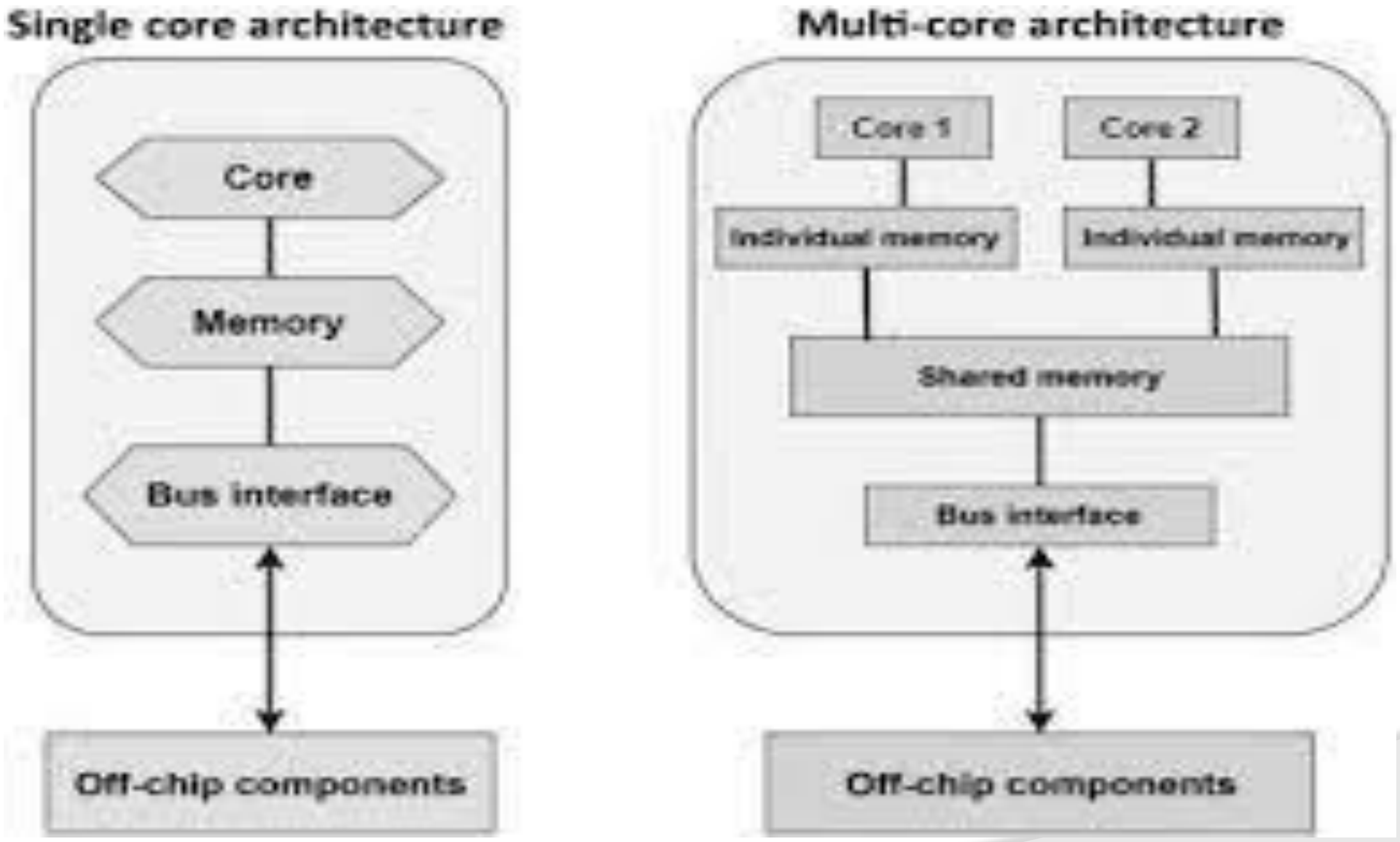
Multicore Computers

What are Multicore Computers?

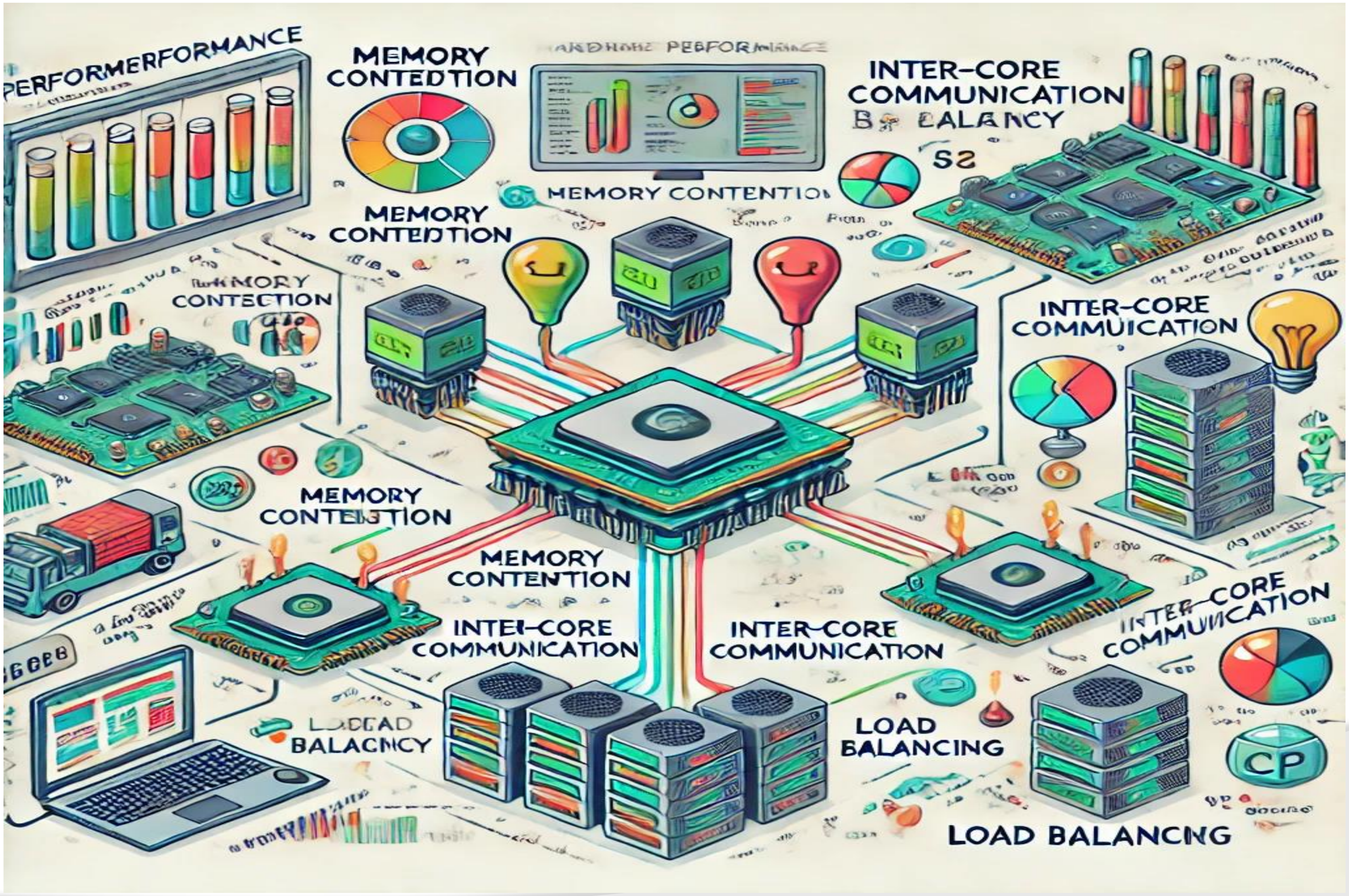
A multicore processor is a type of computer processor that contains multiple independent processing units, known as cores, on a **single integrated circuit (IC) chip**.

Each core can execute instructions independently and concurrently, allowing for increased processing power and improved multitasking capabilities.

Multicore Computers



Hardware Performance Issues



Hardware Performance Issues

Hardware Performance Issues in Multicore Computers

Memory Contention:

- In multicore systems, all CPU cores share the same memory bus and main memory (RAM).
- When multiple cores access memory simultaneously, a bottleneck occurs, slowing down data transfer and reducing performance.

Inter-Core Communication Latency:

- Cores need to communicate and share data, especially in parallel computing tasks.
- This communication happens via caches or interconnects, which can introduce delays due to data synchronization or transfer.

Load Balancing:

- In multicore systems, workload distribution is critical.
- If some cores are idle while others are overloaded, it results in inefficient usage of resources, reducing overall performance.

Hardware Performance Issues

Diagram Description:

Cores (CPU1, CPU2, etc.):

Represented as separate blocks connected to a shared bus.

Shared Memory Bus:

A central pathway connecting cores to the main memory, showing contention as multiple arrows pointing towards it.

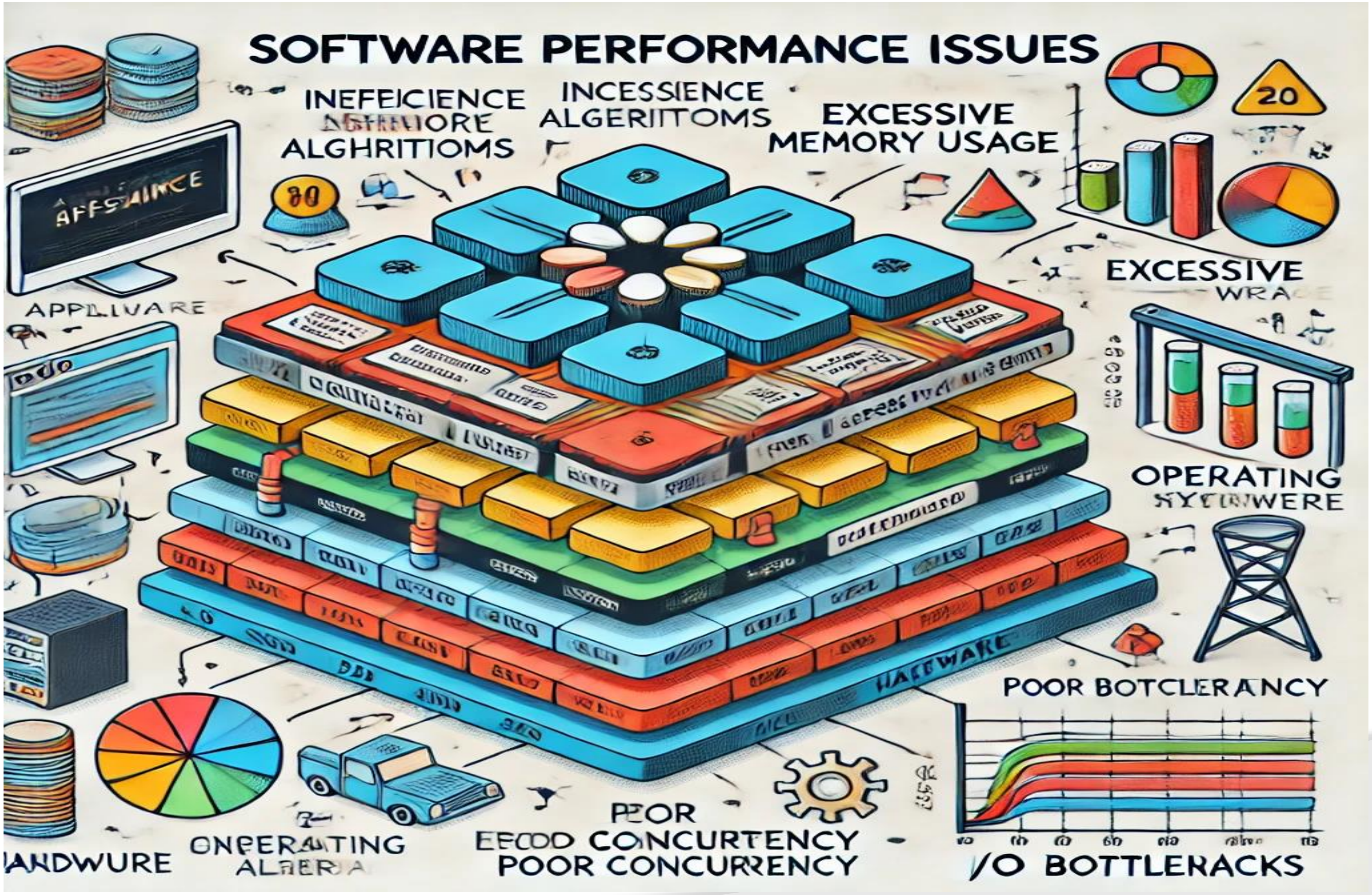
Cache & Interconnects:

Highlighted to show the latency issues in data sharing and communication.

Task Allocation:

Demonstrated with uneven workloads on different cores to represent load balancing problems.

Software Performance Issues



Software Performance Issues

Software Performance Issues

Inefficient Algorithms:

Poorly optimized algorithms can result in high computational complexity, slowing down software execution.

Excessive Memory Usage:

Applications that do not manage memory efficiently can cause excessive use of RAM, leading to performance degradation.

Poor Concurrency:

In multi-threaded applications, inefficient synchronization or improper thread management can cause delays or deadlocks.

I/O Bottlenecks:

Slow input/output operations (e.g., file access, network requests) can limit the application's ability to perform efficiently.

Software Performance Issues

Diagram Features:

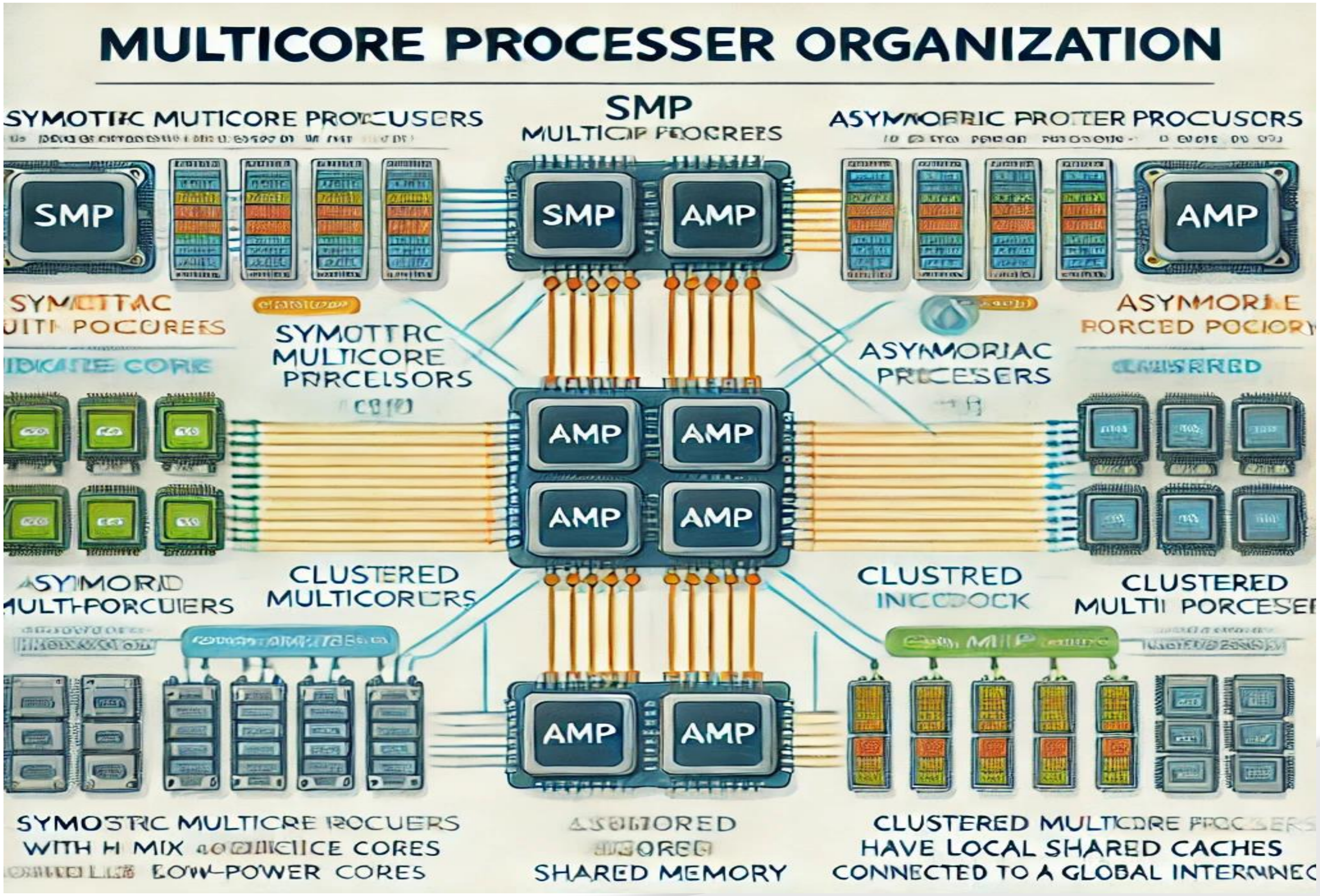
Layers:

Application, Middleware, Operating System, and Hardware layers are shown to illustrate how performance bottlenecks occur at different stages.

Annotations:

Highlight each issue with text and arrows pointing to the relevant layer.

Multicore organization



Multicore organization

Multicore organization refers to how multiple processor cores are arranged and interconnected on a single chip to optimize performance, power consumption, and efficiency. Here's a basic explanation:

Types of Multicore Organization

Symmetric Multicore Processors (SMP):

- All cores are identical in architecture and performance.
- They share memory and peripherals equally.
- Used in general-purpose computing, offering balanced performance for all tasks.

Multicore organization

Asymmetric Multicore Processors (AMP):

- Cores are heterogeneous, with different architectures or performance levels.
- Some cores are high-performance (e.g., for heavy tasks), while others are power-efficient (e.g., for light tasks).
- Common in mobile devices (e.g., ARM's BIG endian LITTLE endian architecture).

Clustered Multicore Processors:

- Cores are grouped into clusters, each with its own cache or memory access.
- Clusters reduce contention for shared memory resources and improve scalability.

Multicore organization

Key Components

Cores:

Independent CPUs that can execute instructions concurrently.

Shared Cache:

A memory level (e.g., L3 cache) accessible by all cores for faster data sharing.

Interconnects:

High-speed pathways (e.g., mesh or ring topology) that enable communication between cores and memory.

Memory:

Shared memory (e.g., RAM) or private memory for each core

Diagram Description

Basic SMP Diagram:

Multiple identical cores connected to a shared memory via an interconnect.

AMP Diagram:

A mix of high-performance and low-power cores connected to shared resources.

Clustered Multicore Diagram:

Groups of cores with localized shared cache and an interconnect connecting clusters..

Intel Core i7-990x

The **Intel Core i7-990X** Extreme Edition is a high-end desktop processor released in 2011 as part of Intel's Gulf town series based on the 32nm process technology. It is designed for enthusiasts and high-performance computing applications. Here are its key features:

Key Specifications

Architecture: Gulf town (part of the Nehalem microarchitecture family)

Cores / Threads: 6 cores and 12 threads (with Hyper-Threading)

Base Clock: 3.46 GHz

Turbo Boost Clock: Up to 3.73 GHz

Cache:

L1 Cache: 64 KB per core

L2 Cache: 256 KB per core

L3 Cache: 12 MB shared

Socket: LGA 1366

TDP: 130W

TDP stands for Thermal Design Power

A CPU with a TDP of 130W would typically require a cooling system capable of handling at least 130 watts of thermal output to maintain safe operating temperatures and prevent thermal throttling.

Intel Core i7-990x

Memory Support: Triple-channel DDR3-1066

Unlocked Multiplier: Designed for overclocking.

Notable Features

Extreme Edition:

Targeted at gamers, content creators, and overclocking enthusiasts.

Comes with an unlocked multiplier for easy overclocking.

6-Core Design:

Among the first Intel desktop processors to feature 6 cores, providing robust multitasking and computing power.

High-End Motherboards:

Compatible with Intel's X58 chipset, requiring premium motherboards for optimal performance.

Legacy and Performance

- The i7-990X was one of the most powerful processors of its time, outperforming most quad-core CPUs in tasks requiring heavy multitasking or computation.
- While it has since been outclassed by newer architectures and processors, it remains a milestone in Intel's lineup of extreme processors.