## Unit 5

## Topic Name : Pipelining

**Table of Contents**


- Pipelining Definition
- Advantages and disadvantages of pipelining
- Instruction Pipelining
- Pipeline hazards.
- References.

**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. A pipeline can be visualized as a collection of processing segments through which binary information flows. Each segment performs partial processing dictated by the way the task is partitioned. The result obtained from the computation in each segment is transferred to the next segment in the pipeline. The final result is obtained after the data have passed through all segments.

The best example of pipelining is the assembly line in industries. In industrial plants multiple platforms are used. In each platform partial processing is done and at the last platform the complete produced is produced. The pipelining is also used in automobile industries for assembling of automobile.

Suppose an examiner has to check 100 copies. There are 5 questions in each copy. One question takes 1 min to check, so a complete copy will take 5 minutes to check. After checking one copy, examiner will pick next copy.

So 100 copies multiplied by 5 minutes

100 X5 = 500 Minutes, is the total time for copy checking.

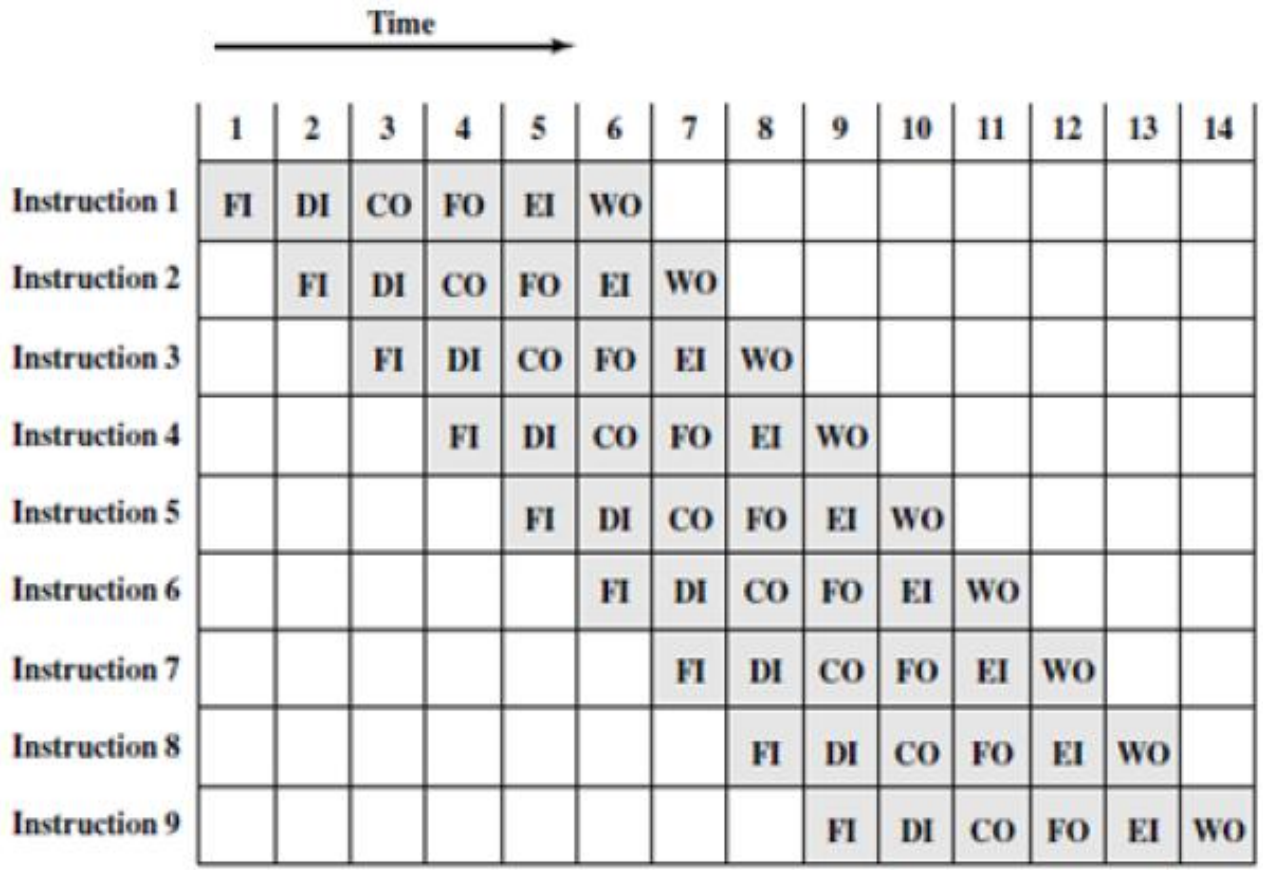
Now, there are 5 examiner E1, E2, E3, E4 and E5. Examiner E1 will check question no. 1 and E2 will check question no 2, E3 will check question no 3, E4 will check question no 4 and E5 will check question no 5. After checking 1 question, copy will be passed to the next examiner and examiner will receive copy from previous examiner. In this way Examiner E5 is placing a completely checked copy after every 1 minute.

Using this multiple examiner technique, we can reduce to total time of copy checking.

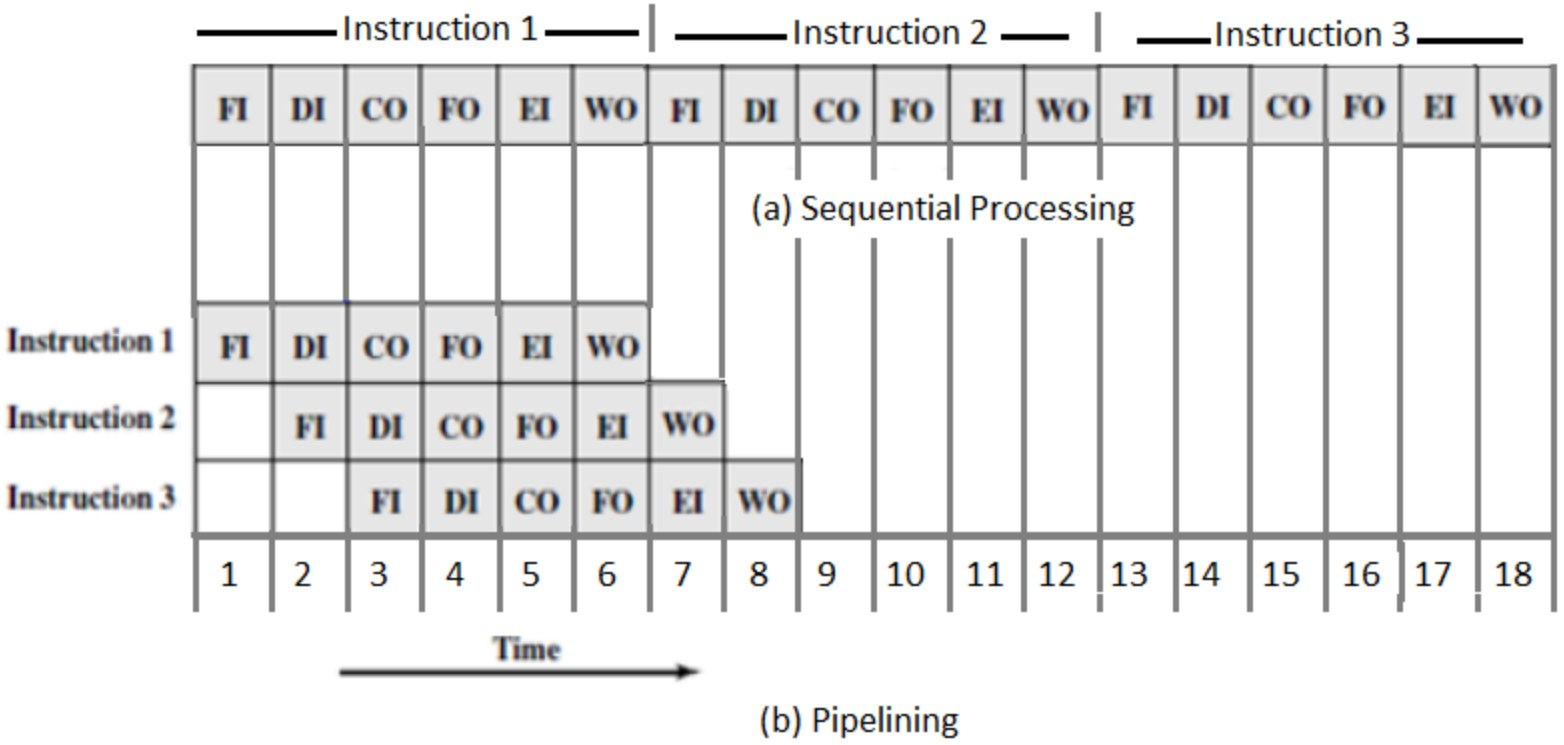
**3. Instruction Pipelining:-** This is a is a technique for implementing instruction-level parallelism within a single processor. Pipelining attempts to keep every part of the processor busy with some instruction by dividing incoming instructions into a series of sequential steps performed by different processor units with different parts of instructions processed in parallel. It allows faster CPU throughput than would otherwise be possible at a given clock rate, but may increase latency due to the added overhead of the pipelining process itself.

Let us consider the following decomposition of the instruction processing.

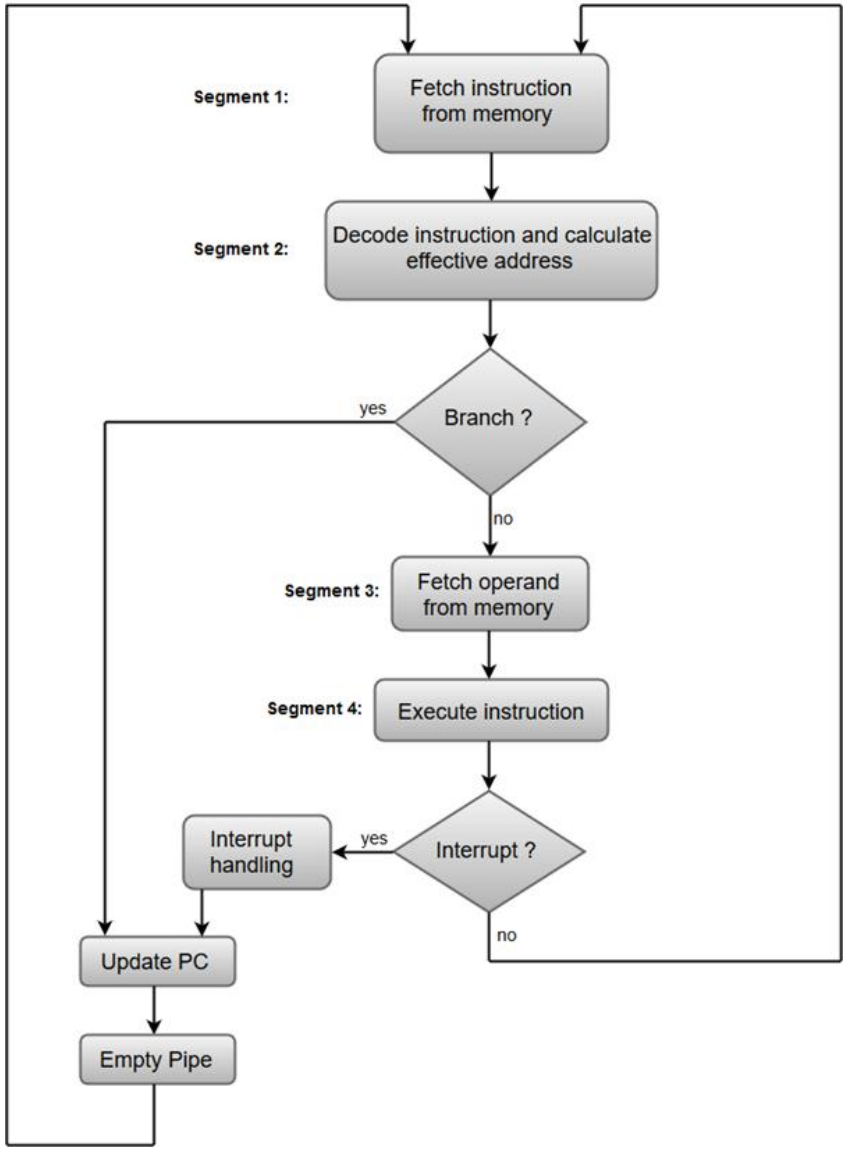
- **Fetch instruction (FI):** Read the instruction from memory.
- **Decode instruction (DI):** Determine the opcode and the operand specifiers.
- **Calculate operands (CO):** Calculate the effective address of each source operand.
- **Fetch operands (FO):** Fetch each operand from memory.
- **Execute instruction (EI):** Perform the indicated operation and store the result, if any, in the specified location.
- **Write operand (WO):** Store the result in memory.




Timing diagram for instruction pipelining




# Pipelining versus sequential process




Four Segment CPU Pipeline

**Arithmetic Pipeline:**

The principles used in instruction pipelining can be used in order to improve the performance of computers in performing arithmetic operations such as add, subtract, and multiply. In this case, these principles will be used to realize the arithmetic circuits inside the ALU.

An arithmetic pipeline divides an arithmetic problem into various sub problems for execution in various pipeline segments. It is used for floating point operations, multiplication and various other computations. Here, the multiple arithmetic logic units are built in the system to perform the parallel arithmetic computation in various data format.

**Pipeline Conflicts**

There are some factors that cause the pipeline to deviate its normal performance. Some of these factors are given below:

**1. Timing Variations**

All stages cannot take same amount of time. This problem generally occurs in instruction processing where different instructions have different operand requirements and thus different processing time.

**2. Data Hazards**

When several instructions are in partial execution, and if they reference same data then the problem arises. We must ensure that next instruction does not attempt to access data before the current instruction, because this will lead to incorrect results.

**3. Branching**

In order to fetch and execute the next instruction, we must know what that instruction is. If the present instruction is a conditional branch, and its result will lead us to the next instruction, then the next instruction may not be known until the current one is processed.

**4. Interrupts**

Interrupts set unwanted instruction into the instruction stream. Interrupts effect the execution of instruction.

**5. Data Dependency**

It arises when an instruction depends upon the result of a previous instruction but this result is not yet available.

**Advantages of Pipelining**

- The cycle time of the processor is reduced.
- It increases the throughput of the system
- It makes the system reliable.

**Disadvantages of Pipelining**

- The design of pipelined processor is complex and costly to manufacture.
- The instruction latency is more.

# Multiprocessor System and Interconnection Structure

**Table of Contents**

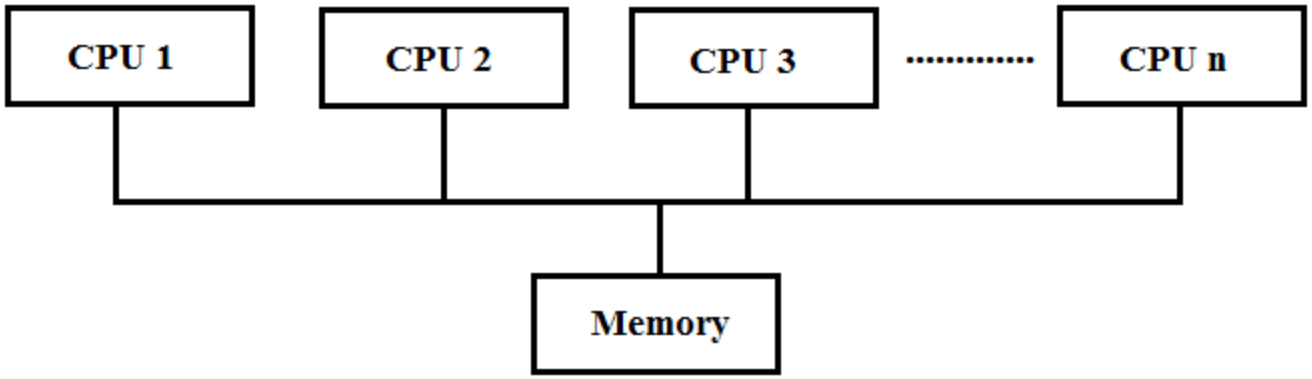

- Multiprocessor System
- Benefits of Multiprocessor System
- Characteristics of Multiprocessors
- Interconnection Structure
- Types of Interconnection Structure
- References

**Multiprocessor System:** A Multiprocessor is a computer system with two or more central processing units (CPUs) share full access to a common RAM. The main objective of using a multiprocessor is to boost the system's execution speed, with other objectives being fault tolerance and application matching.

There are two types of multiprocessors, one is called shared memory multiprocessor and another is distributed memory multiprocessor. In shared memory multiprocessors, all the CPUs shares the common memory but in a distributed memory multiprocessor, every CPU has its own private memory.

**Benefits of a Multiprocessor**

- Enhanced performance.
- Multi-tasking inside an application.
- High throughput and responsiveness.
- Hardware sharing among CPUs.




**Multiprocessor Architecture**

**Characteristics of multiprocessors :**

1. A multiprocessor system is an interconnection of two or more CPUs with memory and input-output equipment.
2. The term “processor” in multiprocessor can mean either a central processing unit (CPU) or an input-output processor (IOP).
3. Multiprocessing improves the reliability of the system.
4. The benefit derived from a multiprocessor organization is an improved system performance.
5. Multiprocessor are classified by the way their memory is organized.
    - 6.1. A multiprocessor system with common shared memory is classified as a shared-memory or **tightly coupled** multiprocessor.
    - 6.2. Each processor element with its own private local memory is classified as a **loosely coupled** system.

**interconnection structure:** A computer consists of a set of components or modules of three basic types (processor, memory, I/O) that communicate with each other. The collection of paths connecting the various modules is called the interconnection structure.

An interconnection network is used for exchanging data between multiple processors in a multiprocessor system. In multiprocessor systems, the performance will be severely affected in case the data exchange between processors is delayed. The multiprocessor system has one global shared memory and each processor has a small local memory. The processors can access data from memory associated with another processor or from shared memory using an interconnection network. Thus, interconnection networks play a central role in determining the overall performance of the multiprocessor systems.

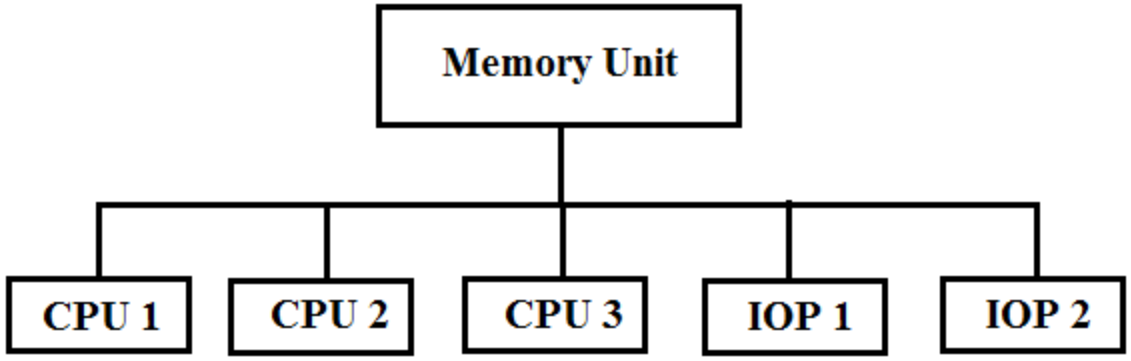
**Interconnection Structures :** The interconnection between the components of a multiprocessor System can have different physical configurations depending on the number of transfer paths that are available between the processors and memory in a shared memory system and among the processing elements in a loosely coupled system. Commonly used interconnection structures are as follows.

- Time-Shared Common Bus
- Multiport Memory
- Crossbar Switch
- Multistage Switching Network
- Hypercube System

**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.

**Disadvantages:**

- Only one processor can communicate with the memory or another processor at any given time.
- As a consequence, the total overall transfer rate within the system is limited by the speed of the single path




**Time Shared Common Bus System**

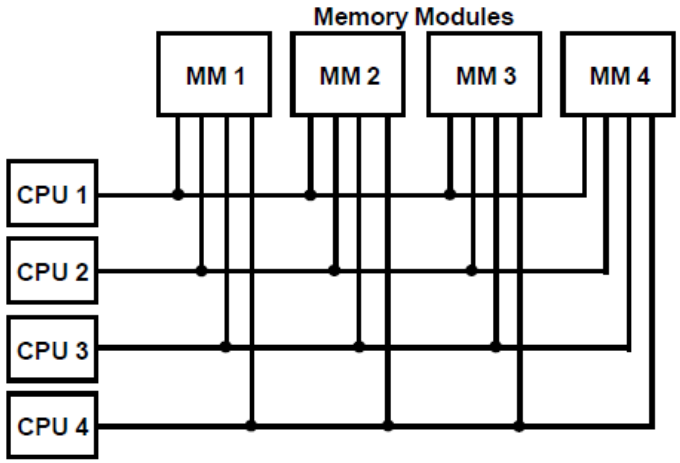
**2. Multiport Memory:** A multiport memory system employs separate buses between each memory module and each CPU. The module must have internal control logic to determine which port will have access to memory at any given time. Memory access conflicts are resolved by assigning fixed priorities to each memory port.

**Advantages:**

The high transfer rate can be achieved because of the multiple paths.

**Disadvantages:**

It requires expensive memory control logic and a large number of cables and connections

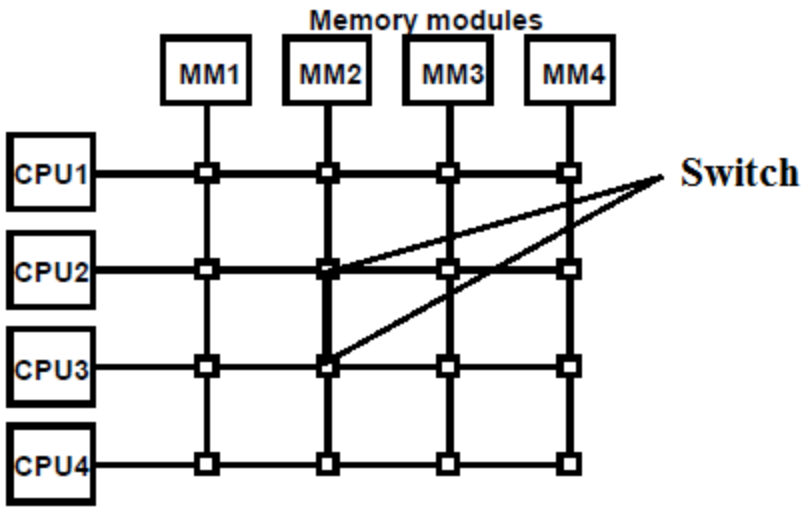



Multiport Memory

**3. Crossbar Switches:** Each switch point has control logic to set up the transfer path between a processor and a memory. It also resolves the multiple requests for access to the same memory on the predetermined priority basis.

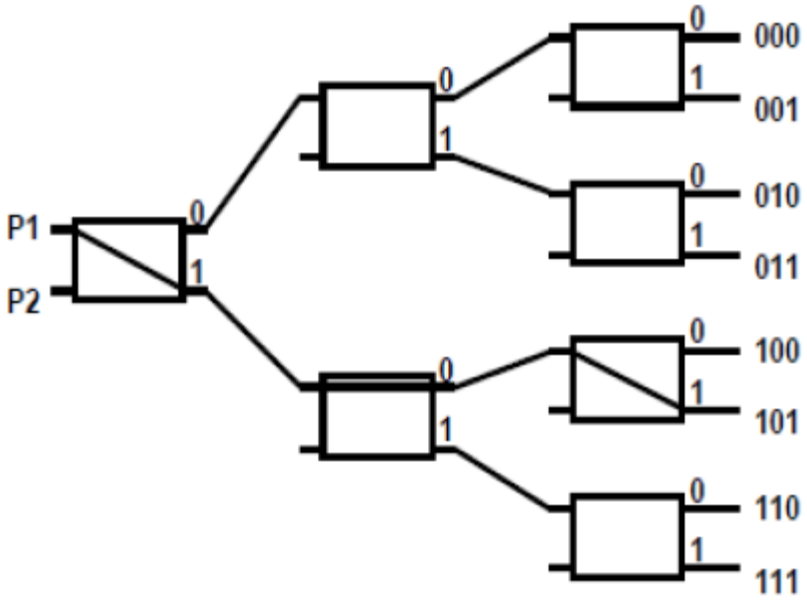
**Advantage:** Though this organization supports simultaneous transfers from all memory modules because there is a separate path associated with each Module.

**Disadvantage:** The H/w required to implement the switch can become quite large and Complex.



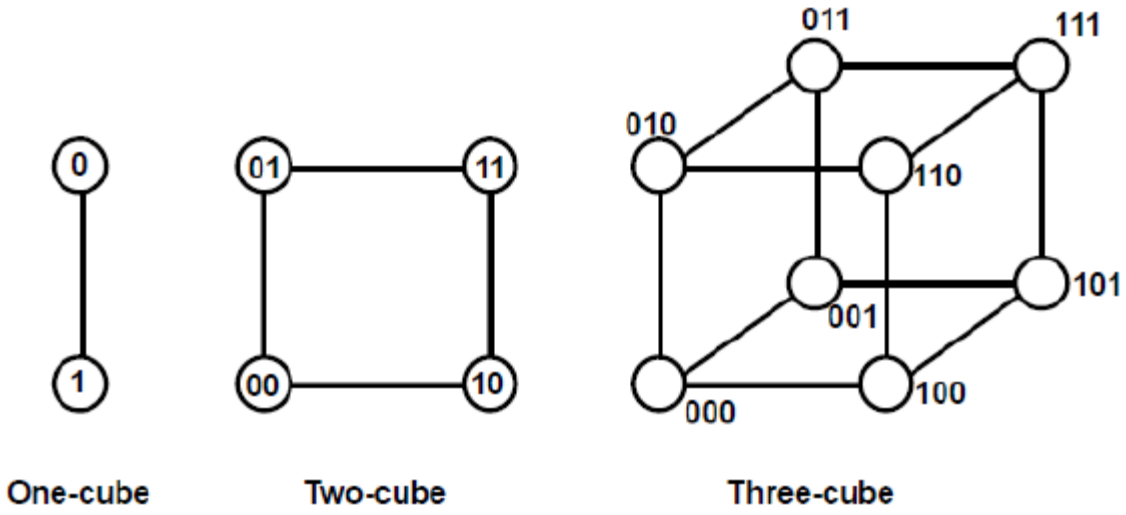

Crossbar Switch

**4. Multistage Switching Network:** The basic component of a multi stage switching network is a two-input, two output interchange switch. Using the 2x2 switch as a building block, it is possible to build a multistage network to control the communication between a number of sources and destinations.




**Binary Tree with 2X2 Switches**

**5. Hypercube System:** 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. Number of edges connected with a node is called node degree (n). processors are conceptually on the corners of a n-dimensional hypercube, and each is directly connected to the n neighboring nodes




Hypercube structures for n=1,2,3

## Topic Name : Interprocessor Arbitration:

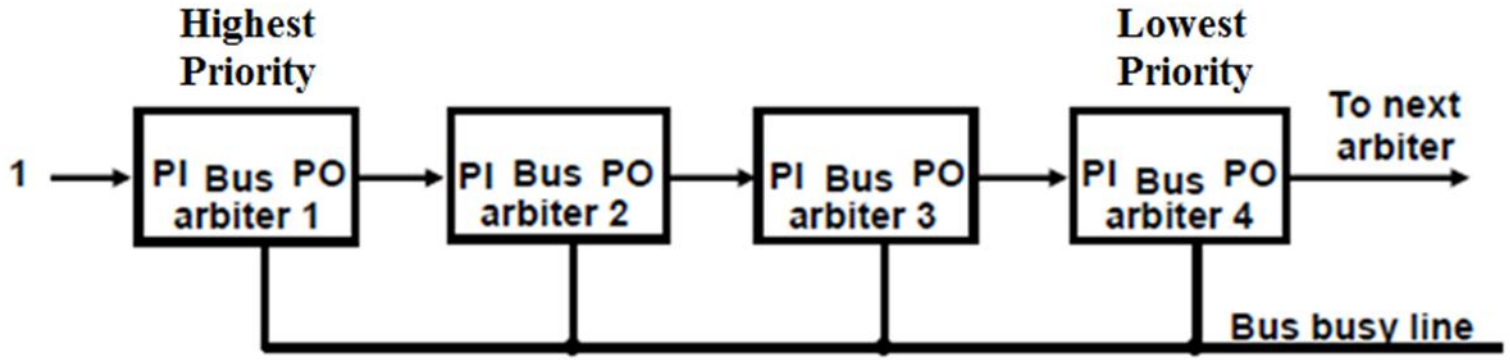
**Table of Contents**


- Interprocessor Arbitration
- Serial Arbitration Procedure
- Parallel Arbitration
- Dynamic Arbitration Algorithms
- References.

**Interprocessor Arbitration:** Computer system needs buses to facilitate the transfer of information between its various components. There are buses that transfer data between the CPUs and memory. These are called memory buses. The arbitration procedure comes into picture whenever there are more than one processors requesting the services of the bus. The conflict is resolved by arbitration procedure.

A processor, in a multiprocessor system, requests the access of a component through the system bus. In case there is no processor accessing the bus at that time, it is given the control of the bus immediately. If there is a second processor utilizing the bus, then this processor has to wait for the bus to be freed. If at any time, there is request for the services of the bus by more than one processor, then the arbitration is performed to resolve the conflict. A bus controller is placed between the local bus and the system bus to handle this.

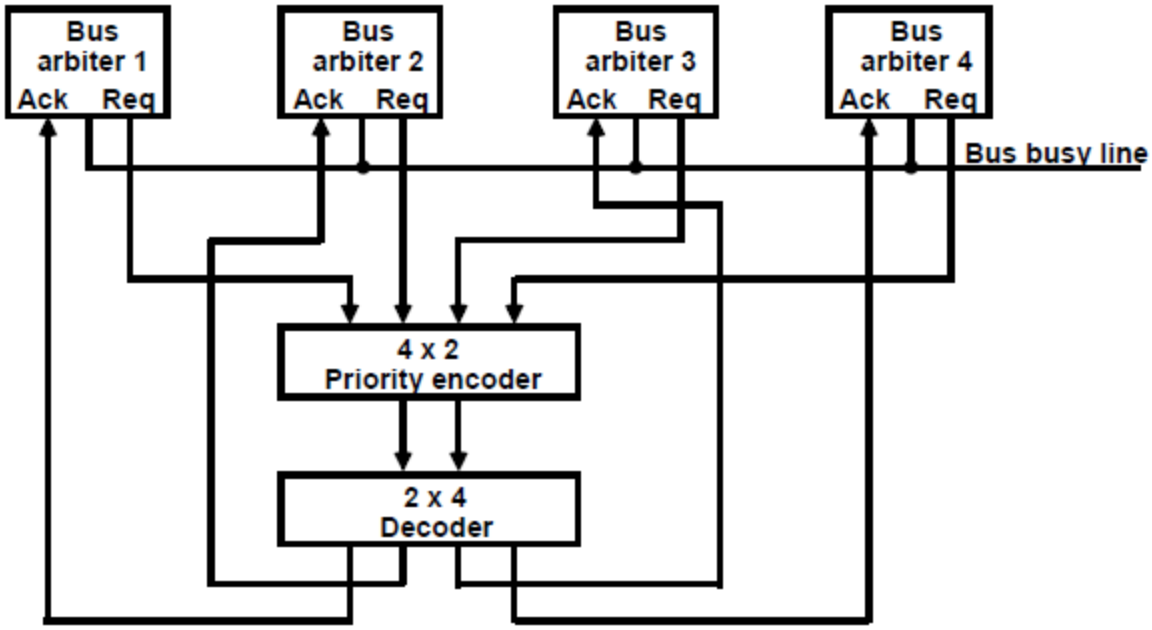
**1. Serial Arbitration Procedure:** The arbitration procedure comes into picture whenever there are more than one processors requesting the services of the bus. The conflict is resolved by assigning priorities to the processors. In serial arbitration the processors/bus arbitors are connected to each other in a serial fashion by an arbitration signal. Whenever any of the processors requests the services of the bus, it sends a request in the form of an interrupt. The interrupt is acknowledged, by asking the first processor in the queue, "have you interrupted ? ".




Serial (daisy chain) arbitration

let us assume that the processor connected to arbitor 3 requests for the services of the bus. The bus request is activated. The bus, if it is free, sends an acknowledgement signal. This signal is received by the PI line of the arbitor 1 corresponding to processor 1. The PI line goes high. Now, as the processor connected to arbitor 1 had not requested for the services of the bus, the PO line of the arbitor also goes high. As PO line of arbitor 1 is connected to the PI line of arbitor 2, it. activates the PI line of arbitor 2. The processor connected to arbitor 2 had also not requested the services of the bus, so in the manner as before, the acknowledgement is rippled to the third arbitor. The third arbitor has PO as 0, therefore the acknowledgement is received by the third arbitor and is not rippled further. Now Processor corresponding to arbitor 3 will use the bus.

**2. Parallel Arbitration Logic:** The parallel arbitration logic uses an external priority encoder and a decoder. Arbitor has a bus request line and a bus acknowledge line. The arbitor activates the bus request line whenever the processor attached to it requests for the services of the bus. If the bus is not currently busy (bus busy line is low), its acknowledge line is enabled.

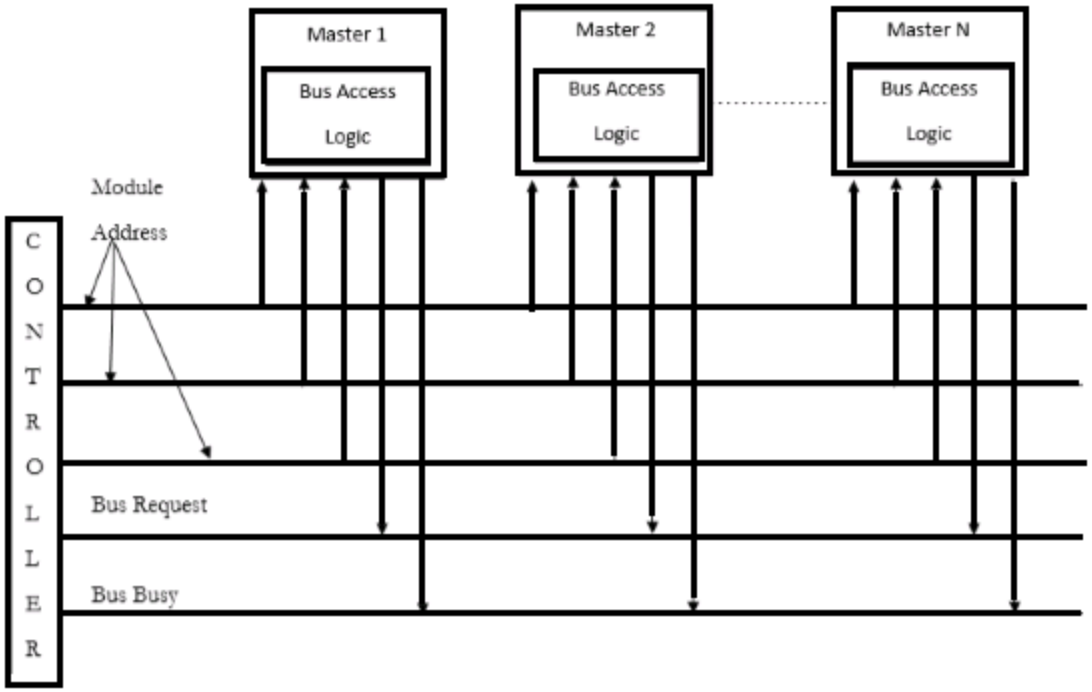



**Parallel Arbitration**

**Dynamic Arbitration Algorithms:** The bus arbitration logics discussed in the previous sections were both static in nature, in the sense, that the priorities to the various arbitors were allocated through hardware and could not be easily changed through software during the execution of the tasks. In dynamic arbitration the priorities, assigned through software, which could be changed easily, during the execution of the tasks. Following the commonly used dynamic arbitration algorithms.

- **Time Slice:** Time slice algorithm allocates a fixed time interval to each processor in a round-robin fashion
- **Least recently used:** The Least Recently Used priority scheme gives the control to the device which has not used the services of the buses for the longest time.
- **First in first out:** In the First In First Out scheme a queue of processors is maintained. After a processor is serviced it gets to the tail of the queue, and the processor next in the queue gets the highest priority.

- **Polling:** In polling, bus controller advances the address to identify the requesting unit. When processor that requires the access recognizes its address, it activates the bus busy line and then accesses the bus. After a number of bus cycles, the polling continues by choosing a different processor.




**Polling bus arbitration**

## Topic Name : Cache Coherence

**Table of Contents**

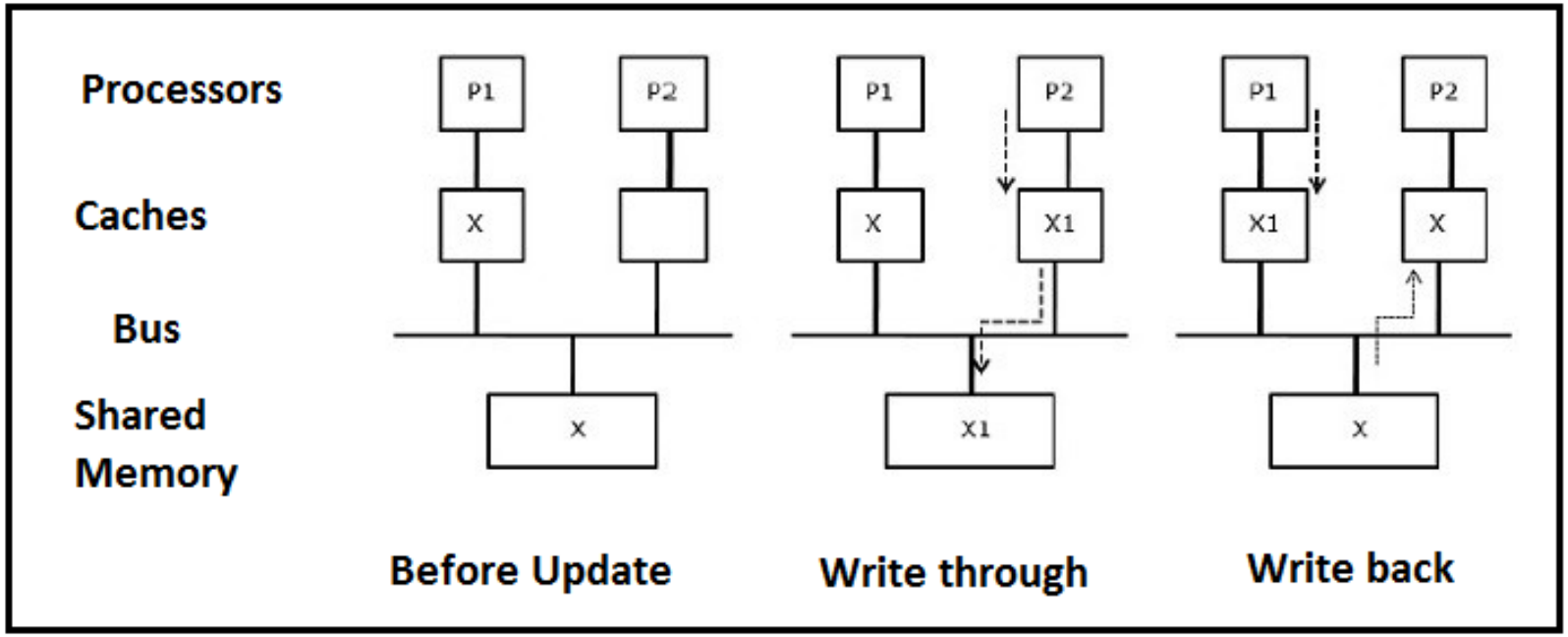

- Cache Coherence
- Cache Write Policies
- Write back
- Write through
- RISC
- CISC
- References.

**Cache Coherence:** In a multiprocessor system, data inconsistency may occur among adjacent levels or within the same level of the memory hierarchy. For example, the cache and the main memory may have inconsistent copies of the same object.

As multiple processors 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 Write Policies:** There two main cache write policies.

1. **Write back** : Write operations are usually made only to the cache. Main memory is only updated when the corresponding cache line is flushed from the cache.
2. **Write through** : All write operations are made to main memory as well as to the cache, ensuring that main memory is always valid.




Cache Coherence

**Some solution of Cache Coherence problem**

- Disallow private cache
- Access time delay
- Read-Only Data are Cacheable
- Private Cache is for Read-Only data
- Shared Writable Data are not cacheable
- Compiler tags data as cacheable and non cacheable

**Reduced instruction set computers (RISC):**

RISC (reduced instruction set computer) is a microprocessor that is designed to perform a smaller number of types of computer instructions so that it can operate at a higher speed (perform more millions of instructions per second, or MIPS).

**Characteristics of RISC**

- Relatively few instructions.
- Relatively few addressing modes.
- Memory access limited to load and store instructions.
- All operations done within the registers of the CPU.
- Single cycle instruction execution.
- Hardwired than micro programmed.

**Complex instruction set computers (CISC):**

Complex instruction set computing (**CISC)** is a processor design where single instructions can execute several low-level operations (such as a load from memory, an arithmetic operation, and a memory store) or are capable of multi-step operations or addressing modes within single instructions.

**Characteristics of CISC**

- A large number of instructions – typically from 100 to 240 instructions.
- A large variety of addressing modes.
- Variable length instruction format.
- Instruction that manipulate operands in memory

## References


- Mano Morris, “Computer System Architecture”, PHI
- William Stalling, “Computer Organization & Architecture”, Pearson education Asia
- Hamacher vranesic zaky, “Computer Organization”, McGraw Hill
- B. Ram, “Computer Fundamental Architecture & Organization”, New Age.
- Tannenbaum, “Structured Computer Organization”, PHI.