UNIT-IV

Parallel Processing

Parallel processing can be described as a class of techniques which enables the system to achieve simultaneous data-processing tasks to increase the computational speed of a computer system.

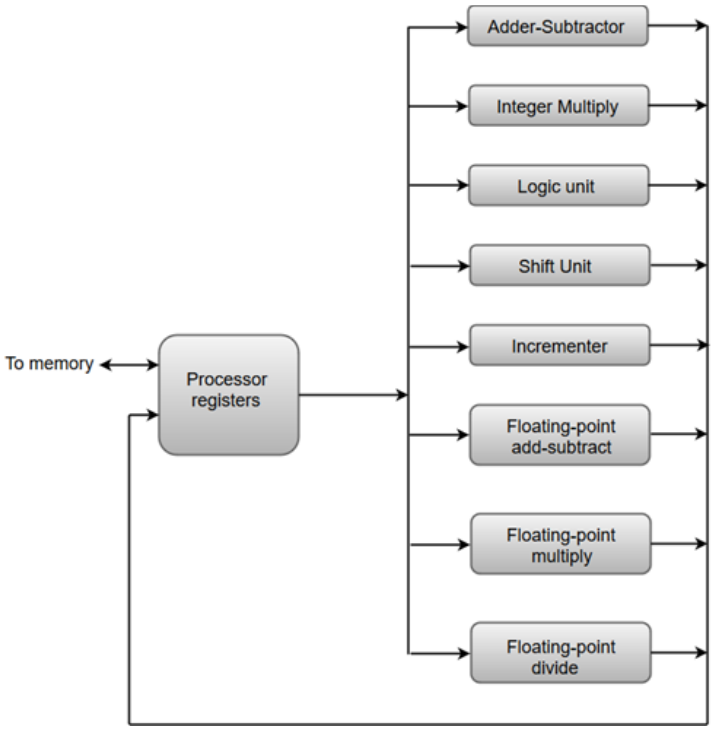
A parallel processing system can carry out simultaneous data-processing to achieve faster execution time. For instance, while an instruction is being processed in the ALU component of the CPU, the next instruction can be read from memory.

The primary purpose of parallel processing is to enhance the computer processing capability and increase its throughput, i.e. the amount of processing that can be accomplished during a given interval of time.

A parallel processing system can be achieved by having a multiplicity of functional units that perform identical or different operations simultaneously. The data can be distributed among various multiple functional units.

The following diagram shows one possible way of separating the execution unit into eight functional units operating in parallel.

The operation performed in each functional unit is indicated in each block if the diagram:



- The adder and integer multiplier performs the arithmetic operation with integer numbers.
- The floating-point operations are separated into three circuits operating in parallel.

The logic, shift, and increment operations can be performed concurrently on different data. All units are independent of each other, so one number can be shifted while another number is being incremented.

COA Pipelining

The term Pipelining refers to a technique of decomposing a sequential process into sub-operations, with each sub-operation being executed in a dedicated segment that operates concurrently with all other segments.

The most important characteristic of a pipeline technique is that several computations can be in progress in distinct segments at the same time. The overlapping of computation is made possible by associating a register with each segment in the pipeline. The registers provide isolation between each segment so that each can operate on distinct data simultaneously.

The structure of a pipeline organization can be represented simply by including an input register for each segment followed by a combinational circuit.

Let us consider an example of combined multiplication and addition operation to get a better understanding of the pipeline organization.

The combined multiplication and addition operation is done with a stream of numbers such as:

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

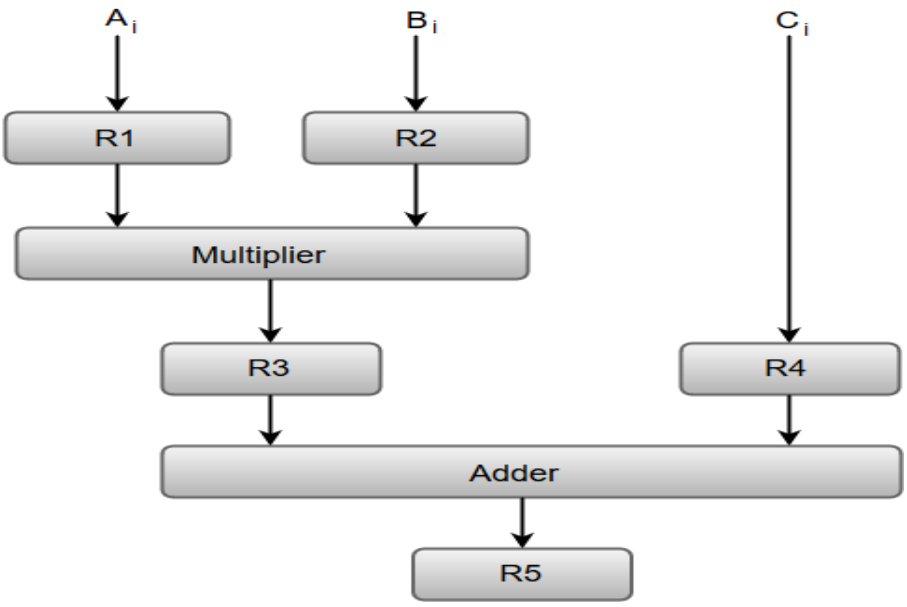
The operation to be performed on the numbers is decomposed into sub-operations with each sub-operation to be implemented in a segment within a pipeline.

The sub-operations performed in each segment of the pipeline are defined as:

$R1 \leftarrow A_i$, $R2 \leftarrow B_i$ *Input* A_i, *and* B_i
$R3 \leftarrow R1 * R2$, $R4 \leftarrow C_i$ *Multiply, and input* C_i
$R5 \leftarrow R3 + R4$ *Add* C_i *to product*

The following block diagram represents the combined as well as the sub-operations performed in each segment of the pipeline.

Pipeline Processing:



Registers R1, R2, R3, and R4 hold the data and the combinational circuits operate in a particular segment.

The output generated by the combinational circuit in a given segment is applied as an input register of the next segment. For instance, from the block diagram, we can see that the register R3 is used as one of the input registers for the combinational adder circuit.

In general, the pipeline organization is applicable for two areas of computer design which includes:

1. Arithmetic Pipeline
2. Instruction Pipeline

We will discuss both of them in our later sections.

We are going to discuss these below briefly for a general idea.

1. Arithmetic Pipeline:

An arithmetic pipeline is a technologically shaped processing pipeline designed to accelerate the implementation of arithmetic operations. It's an ideal part of the general processor figure, particularly specializing in improving the overall performance of mathematical computations.

Components

- **Addition Stage:** In this stage, the pipeline plays the addition operation. It's a crucial mathematical operation and is frequently broken down into sub-parts for efficient processing.
- **Multiplication Stage:** For more complicated mathematics operations, which encompass multiplication, an intense level is covered in the pipeline. Multiplication consists of a sequence of partial products, and an arithmetic pipeline can simplify this device.

- **Division Stage:** Division is any other arithmetic operation that can take advantage of pipelining. Dividing various involves more than one step, and breaking down the approach into pipeline ranges can decorate the general tempo of execution.

Advantages:

- **Parallelism in Arithmetic Operations:** Arithmetic pipelines take advantage of parallelism by breaking down complex operations into small parts. This allows the concurrent execution of a couple of arithmetic operations, considerably enhancing throughput.
- **Optimized Resource Utilization:** The pipeline structure allows for the best usage of processing resources. While one arithmetic operation is within the multiplication stage, every other can be within the addition stage, maximizing the performance of the processor.
- **Enhanced Computational Speed:** By dividing arithmetic operations into smaller, feasible phases, the overall pace of computation is expanded. This is mainly critical in programs in which mathematical calculations are a large element, which includes medical computing or photograph processing.

2. Instruction Pipeline:

An Instruction Pipeline is a key component of a processor's structure designed to facilitate the concurrent execution of a couple of commands. It breaks down the execution of instructions into different phases, allowing one-of-a-type spans to function simultaneously on unique instructions.

Components:

- **Instruction Fetch (IF):** The first stage entails fetching the instruction from memory. The software program counter is used to decide the address of the following approach.
- **Instruction Decode (ID):** In this phase, the fetched instruction is decoded to determine the operation to be completed and to understand the operands involved.
- **Execution (EX):** The actual computation or operation through the instruction takes place in this stage. It might also additionally contain mathematics or logical operations.
- **Memory Access (MEM):** If instruction requires access to memory, this stage is wherein data is analyzed from or written to memory.
- **Write Back (WB):** The final phase includes registering the results once more to report or memory and finishing the execution of these.

Advantages:

- **Improved Throughput:** The instruction pipeline allows for a continuous drift of commands through the processor, enhancing the usual throughput. While one instruction is within the execution phase, every other may be within the decoding phase, resulting in better resource utilization.
- **Faster Program Execution:** By overlapping the execution of instructions, the time taken to execute a series of commands is reduced. This outcome in faster software execution is a vital element in enhancing the general performance of a PC system.
- **Effective Resource Management:** Instructional pipelining allows powerful manipulation of sources by permitting tremendous levels of the pipeline to operate concurrently. This contributes to a good and streamlined execution of commands.

Conclusion

In short, pipelining stands as a cornerstone of processor layout, presenting a systematic and effective technique for enhancing typical overall performance via parallelism. Its application stages range from simple practice pipelines to the modern superscalar architectures seen in current CPUs. The evolution of pipelining techniques, coupled with improvements in memory hierarchy and ILP, continues to enhance power in computer structures, pushing the bounds of computational competencies. In the future, the principles of pipelining are possibly crucial to the continued exploration of faster and more trusting computing structures.

Arithmetic Pipeline

Arithmetic Pipelines are mostly used in high-speed computers. They are used to implement floating-point operations, multiplication of fixed-point numbers, and similar computations encountered in scientific problems.

To understand the concepts of arithmetic pipeline in a more convenient way, let us consider an example of a pipeline unit for floating-point addition and subtraction.

The inputs to the floating-point adder pipeline are two normalized floating-point binary numbers defined as:

$$X = A * 2^a = 0.9504 * 10^3$$
$$Y = B * 2^b = 0.8200 * 10^2$$

Where **A** and **B** are two fractions that represent the mantissa and **a** and **b** are the exponents.

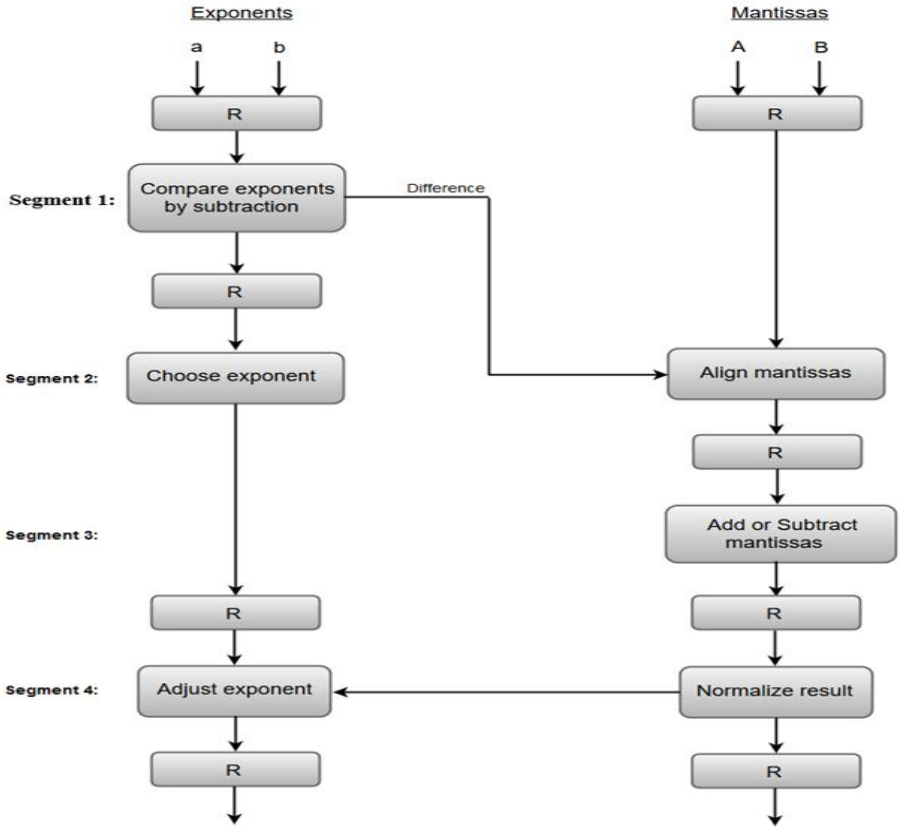
The combined operation of floating-point addition and subtraction is divided into four segments. Each segment contains the corresponding suboperation to be performed in the given pipeline. The suboperations that are shown in the four segments are:

1. Compare the exponents by subtraction.
2. Align the mantissas.
3. Add or subtract the mantissas.
4. Normalize the result.

We will discuss each suboperation in a more detailed manner later in this section.

The following block diagram represents the suboperations performed in each segment of the pipeline.

Pipeline organization for floating point addition and subtraction:



Note: Registers are placed after each suboperation to store the intermediate results.

1. Compare exponents by subtraction:

The exponents are compared by subtracting them to determine their difference. The larger exponent is chosen as the exponent of the result.

The difference of the exponents, i.e., **3 - 2 = 1** determines how many times the mantissa associated with the smaller exponent must be shifted to the right.

2. Align the mantissas:

The mantissa associated with the smaller exponent is shifted according to the difference of exponents determined in segment one.

$X = 0.9504 * 10^3$
$Y = 0.08200 * 10^3$

3. Add mantissas:

The two mantissas are added in segment three.

$$Z = X + Y = 1.0324 * 10^3$$

4. Normalize the result:
After normalization, the result is written as:

$$Z = 0.1324 * 10^4$$

Instruction Pipeline

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

Most of the digital computers with complex instructions require instruction pipeline to carry out operations like fetch, decode and execute instructions.

In general, the computer needs to process each instruction with the following sequence of steps.

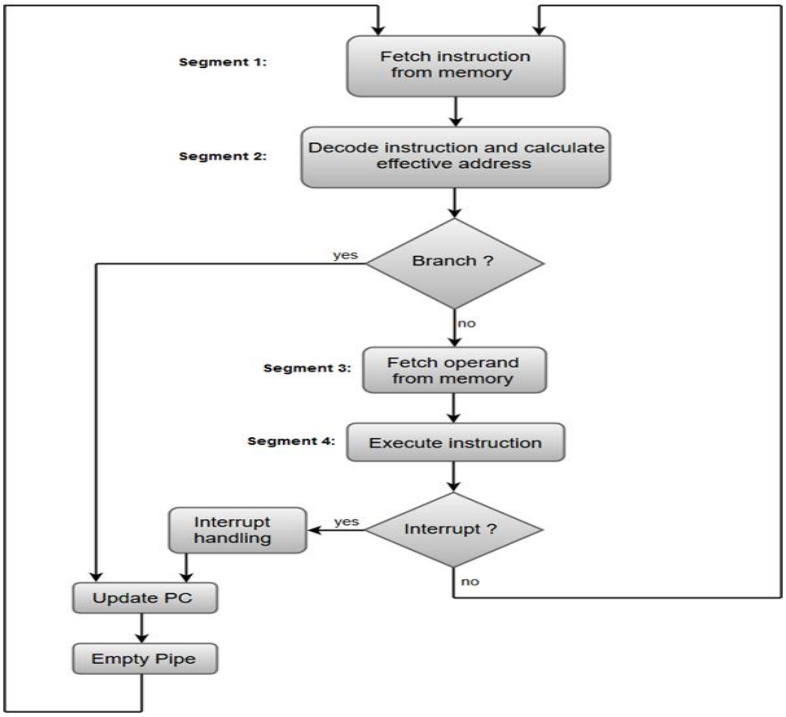
1. Fetch instruction from memory.
2. Decode the instruction.
3. Calculate the effective address.
4. Fetch the operands from memory.
5. Execute the instruction.
6. Store the result in the proper place.

Each step is executed in a particular segment, and there are times when different segments may take different times to operate on the incoming information. Moreover, there are times when two or more segments may require memory access at the same time, causing one segment to wait until another is finished with the memory.

The organization of an instruction pipeline will be more efficient if the instruction cycle is divided into segments of equal duration. One of the most common examples of this type of organization is a **Four-segment instruction pipeline.**

A **four-segment instruction** pipeline combines two or more different segments and makes it as a single one. For instance, the decoding of the instruction can be combined with the calculation of the effective address into one segment.

The following block diagram shows a typical example of a four-segment instruction pipeline. The instruction cycle is completed in four segments.



Segment 1:
The instruction fetch segment can be implemented using first in, first out (FIFO) buffer.

Segment 2:
The instruction fetched from memory is decoded in the second segment, and eventually, the effective address is calculated in a separate arithmetic circuit.

Segment 3:
An operand from memory is fetched in the third segment.

Segment 4:
The instructions are finally executed in the last segment of the pipeline organization

Flynn's Classification of Computers

M.J. Flynn proposed a classification for the organization of a computer system by the number of instructions and data items that are manipulated simultaneously.

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

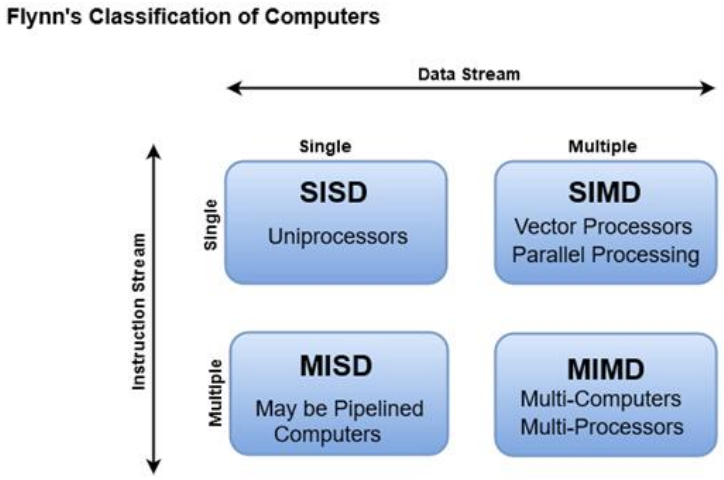
The operations performed on the data in the processor constitute a **data stream**.

Note: The term 'Stream' refers to the flow of instructions or data.

Parallel processing may occur in the instruction stream, in the data stream, or both.

Flynn's classification divides computers into four major groups that are:

1. Single instruction stream, single data stream (SISD)
2. Single instruction stream, multiple data stream (SIMD)
3. Multiple instruction stream, single data stream (MISD)
4. Multiple instruction stream, multiple data stream (MIMD)



What is SISD architecture?

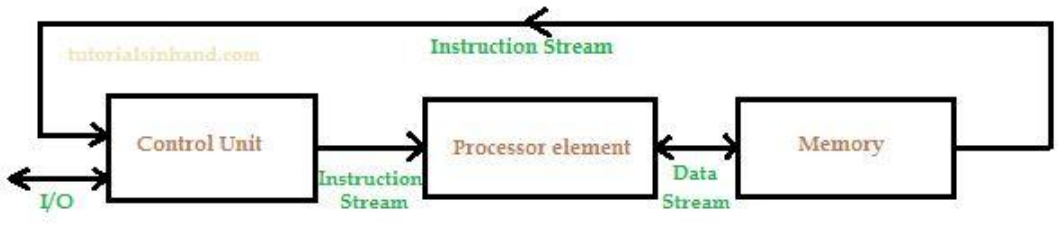
*Full form of **SISD*** is **Single Instruction stream-Single Data stream**.

It is one of the type of flynn's classification of computer.

*In **SISD computer architecture**, instructions are executed sequentially but may be overlapped in their execution stages. In other words pipelining technique can be used in CPU.*
Modern day *SISD* uniprocessor systems are mostly pipelined. An SISD computer may have more than one functional unit in them, but all are under supervision of one control unit. SISD architecture computers can process only scalar type instructions.

Most serial computers available today follow in *SISD architecture*.

Given below is the *sisd architecture diagram*.



SISD Computer

SIMD Architecture

In this chapter of **COA tutorial**, we will learn about:

- *simd architecture*

- *simd architecture diagram*
- *simd architecture example*

In the last chapter we have studied about **SISD architecture**. Lets move ahead and look at another type of flynn's classification of computer.

What is SIMD architecture?

SIMD full form is **Single Instruction stream-Multiple Data stream**.

SIMD computer is one among the four Flynn's classification of computer. Other three are SISD, MISD, MIMD computer.

In ***SIMD computer***, we can see from the below diagram there are multiple processing elements supervised by the common control unit.

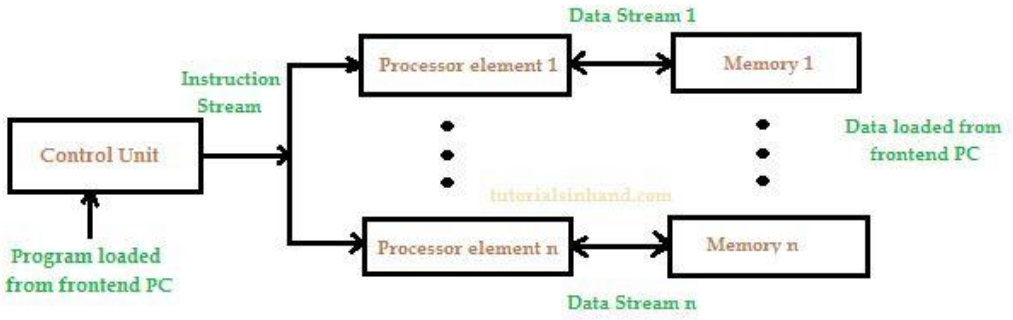
Important points about SIMD architecture

- All the processing elements, which are ALUs, receive the same instruction broadcast from the control unit
- but operate on different data sets from distinct data streams.
- As seen in the diagram, control unit sends common instruction stream to each processor element.
- The shared memory sub-system containing multiple modules is very essential.

SIMD architecture is used in applications

- that requires multimedia operations like image, audio or video processing where same operation like filtering needs to be applied on large set of data pixels.
- that needs matrix multiplication or simulations.

Given below is ***SIMD architecture diagram***:



SIMD Computer

MISD Architecture

In this chapter of **COA tutorial**, we are going to learn about:

- *misd architecture*
- *misd architecture diagram*
- *misd architecture example*

What is MISD architecture?

MISD full form is **Multiple Instruction stream-Single Data stream**. It indicates that multiple instructions are applied to the same set of data.

*MISD architecture is also known as **systolic arrays**.*

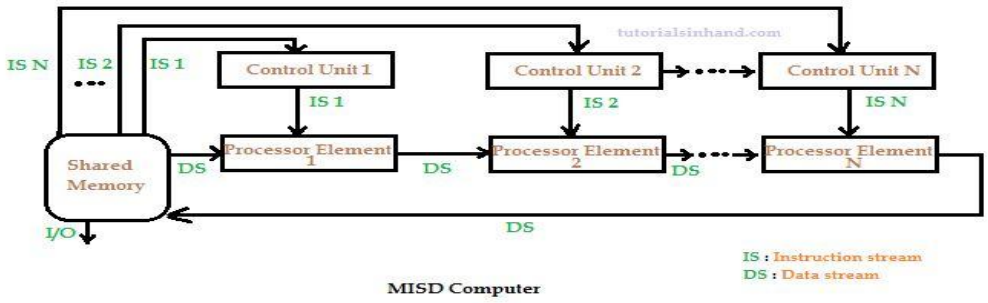
MISD is one among the four machine that falls under the Flynn's classification of computer. Other three are SISD, SIMD and MIMD computers.

In *MISD computer architecture*:

- there are n processor elements.
- Each processor elements receives distinct instructions to execute on the same data stream and its derivatives.
- Here the output of one processor element becomes the input of the next processor element in the series.

MISD architecture is a theoretical architecture and rarely applied in practical systems.

Given below is the ***diagram of MISD architecture***:



MISD Computer

MIMD Architecture

In this chapter of **COA tutorial**, we are going to learn about:

- *mimd architecture*
- *mimd architecture diagram*
- *advantages of mimd architecture*
- *disadvantages of mimd architecture*
- *mimd architecture example*

What is mimd architecture?

MIMD full form is **Multiple Instruction stream-Multiple Data stream**.

MIMD computer is one among four of the *Flynn's classification of computer*. Other three are SIMD, SISD, MISD computer.

MIMD computer category covers multiple computer system and multiprocessor systems.

MIMD architecture involves multiple processors that execute different instructions on different data sets thus exhibiting parallel computing. Thus we can also say that MIMD belongs to class of parallel computing architecture.

MIMD architecture provides great scalability and high performance as well.

MIMD architecture is used in:

- multi-core processors
- supercomputers
- distributed systems

MIMD computer is of two types:

- tightly coupled or Uniform Memory Access (UMA),
- loosely coupled or Non-Uniform Memory Access (NUMA)

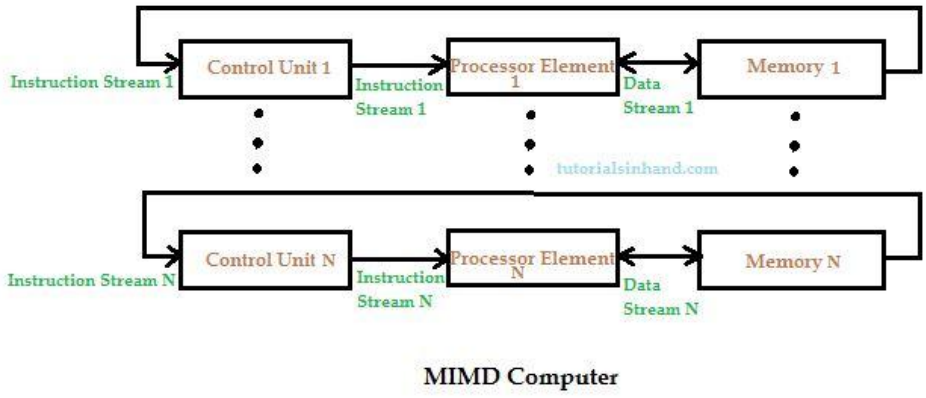
What is Uniform Memory Access?

MIMD computer is called tightly coupled or Uniform Memory Access (UMA) if the degree of interaction among the processor is high.

What is Non-Uniform Memory Access?

MIMD computer is called loosely coupled or Non-Uniform Memory Access (NUMA) if the degree of interaction among processors is low.

Given below is the ***MIMD computer architecture diagram***:



MIMD Computer

Characteristics of Multiprocessor

There are the major characteristics of multiprocessors are as follows −

- **Parallel Computing** − This involves the simultaneous application of multiple processors. These processors are developed using a single architecture to execute a common task. In general, processors are identical and they work together in such a way that the users are under the impression that they are the only users of the system. In reality, however, many users are accessing the system at a given time.
- **Distributed Computing** − This involves the usage of a network of processors. Each processor in this network can be considered as a computer in its own right and have the capability to solve a problem. These processors are heterogeneous, and generally, one task is allocated to a single processor.
- **Supercomputing** − This involves the usage of the fastest machines to resolve big and computationally complex problems. In the past, supercomputing machines were vector computers but at present, vector or parallel computing is accepted by most people.

- **Pipelining** − This is a method wherein a specific task is divided into several subtasks that must be performed in a sequence. The functional units help in performing each subtask. The units are attached serially and all the units work simultaneously.
- **Vector Computing** − It involves the usage of vector processors, wherein operations such as 'multiplication' are divided into many steps and are then applied to a stream of operands ("vectors").
- **Systolic** − This is similar to pipelining, but units are not arranged in a linear order. The steps in systolic are normally small and more in number and performed in a lockstep manner. This is more frequently applied in special-purpose hardware such as image or signal processors
- .

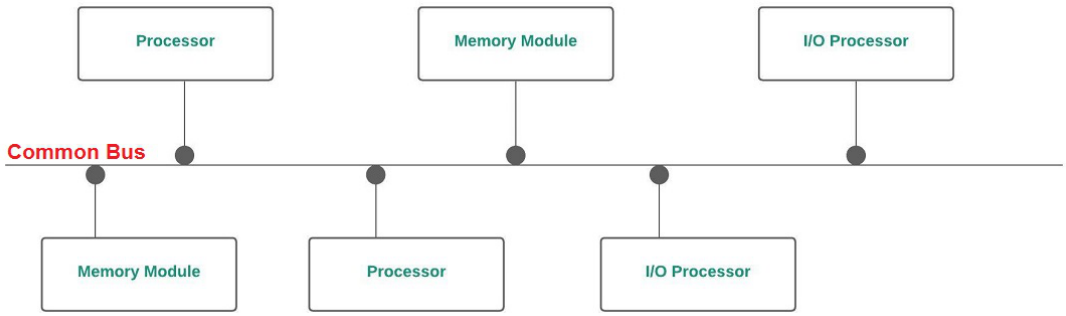
Interconnection structures :

The processors must be able to share a set of main memory modules & I/O devices in a multiprocessor system. This sharing capability can be provided through interconnection structures. The interconnection structure that are commonly used can be given as follows –

1. Time-shared / Common Bus
2. Cross bar Switch
3. Multiport Memory
4. Multistage Switching Network (Covered in 2nd part)
5. Hypercube System

In this article, we will cover Time shared / Common Bus in detail.

1. Time-shared / Common Bus (Interconnection structure in Multiprocessor System) : In a multiprocessor system, the time shared bus interconnection provides a common communication path connecting all the functional units like processor, I/O processor, memory unit etc. The figure below shows the multiple processors with common communication path (single bus).



Single-Bus Multiprocessor Organization

To communicate with any functional unit, processor needs the bus to transfer the data. To do so, the processor first need to see that whether the bus is available / not by checking the status of the bus. If the bus is used by some other functional unit, the status is busy, else free.

A processor can use bus only when the bus is free. The sender processor puts the address of the destination on the bus & the destination unit identifies it. In order to communicate with any functional unit, a command is issued to tell that unit, what work is to be done. The other processors at that time will be either busy in internal operations or will sit free, waiting to get bus.

We can use a bus controller to resolve conflicts, if any. (Bus controller can set priority of

different functional units)
This Single-Bus Multiprocessor Organization is easiest to reconfigure & is simple. This interconnection structure contains only passive elements. The bus interfaces of sender & receiver units controls the transfer operation here.
To decide the access to common bus without conflicts, methods such as static & fixed priorities, First-In-Out (FIFO) queues & daisy chains can be used.

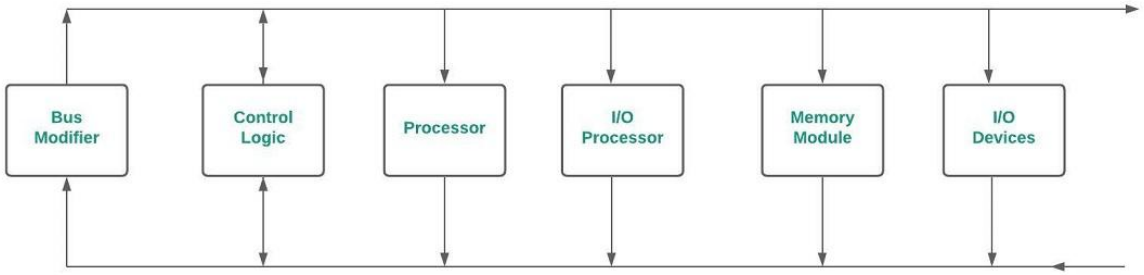
Advantages –

- Inexpensive as no extra hardware is required such as switch.
- Simple & easy to configure as the functional units are directly connected to the bus .

Disadvantages –

- Major fight with this kind of configuration is that if malfunctioning occurs in any of the bus interface circuits, complete system will fail.
- **Decreased throughput** — At a time, only one processor can communicate with any other functional unit.
- **Increased arbitration logic** — As the number of processors & memory unit increases, the bus contention problem increases.

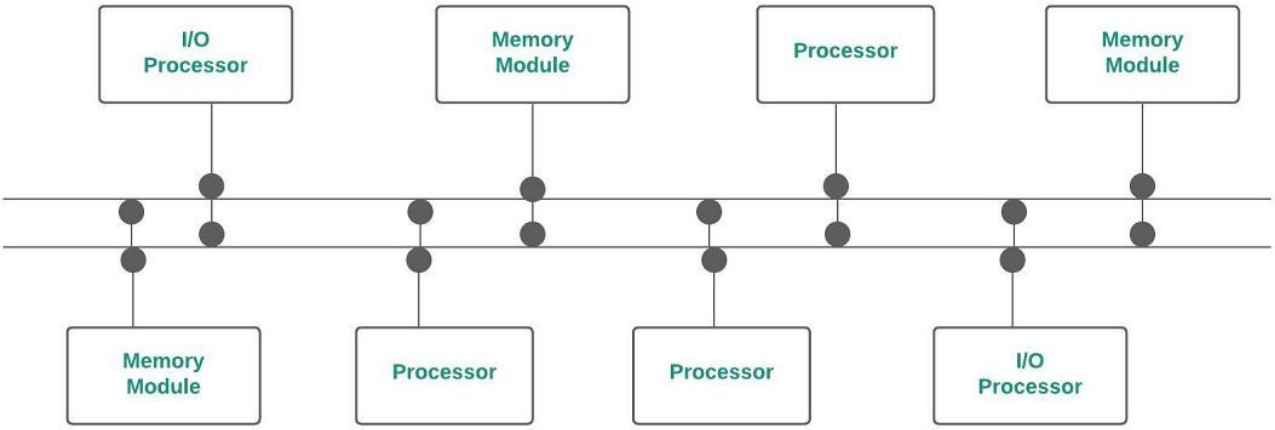
To solve the above disadvantages, we can use two uni-directional buses as :



Multiprocessor System with unidirectional buses

Both the buses are required in a single transfer operation. Here, the system complexity is increased & the reliability is decreased, The solution is to use multiple bi-directional buses.

Multiple bi-directional buses :
The multiple bi-directional buses means that in the system there are multiple buses that are bi-directional. It permits simultaneous transfers as many as buses are available. But here also the complexity of the system is increased.



Multiple Bi-Directional Multiprocessor System

Apart from the organization, there are many factors affecting the performance of bus. They are –

- Number of active devices on the bus.
- Data width
- Error Detection method
- Synchronization of data transfer etc.

Advantages of Multiple bi-directional buses –

- Lowest cost for hardware as no extra device is needed such as switch.
- Modifying the hardware system configuration is easy.
- Less complex when compared to other interconnection schemes as there are only 2 buses & all the components are connected via that buses.

Disadvantages of Multiple bi-directional buses –

- System Expansion will degrade the performance because as the number of functional unit increases, more communication is required but at a time only 1 transfer can happen via 1 bus.
- Overall system capacity limits the transfer rate & If bus fails, whole system will fail.
- Suitable for small systems only.

2. Crossbar Switch :

A point is reached at which there is a separate path available for each memory module, if the number of buses in common bus system is increased. Crossbar Switch (for multiprocessors) provides separate path for each module.

3.Multiport Memory :

In Multiport Memory system, the control, switching & priority arbitration logic are distributed throughout the crossbar switch matrix which is distributed at the interfaces to the memory modules.

4.Hypercube Interconnection :

This is a binary n-cube architecture. Here we can connect 2n processors and each of the processor here forms a node of the cube. A node can be memory module, I/O interface also, not necessarily processor. The processor at a node has communication path that is direct goes to n other nodes (total 2n nodes). There are total 2n distinct n-bit binary addresses.

What is Inter Process Communication?

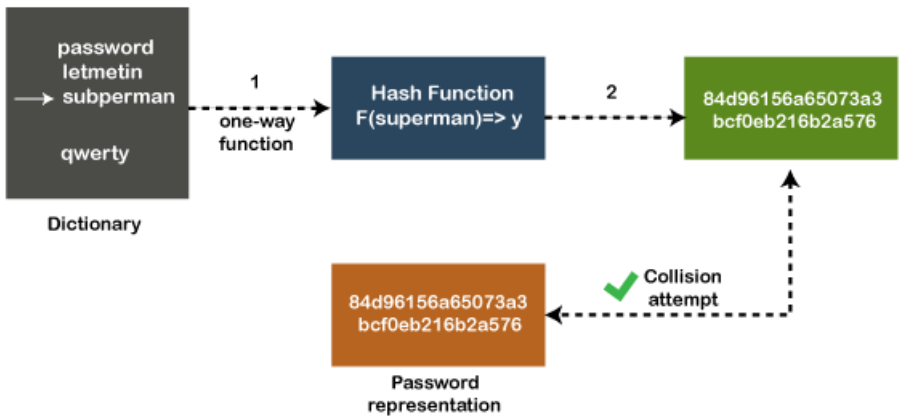
In general, Inter Process Communication is a type of mechanism usually provided by the operating system (or OS). The main aim or goal of this mechanism is to provide communications in between several processes. In short, the intercommunication allows a process letting another process know that some event has occurred.

Let us now look at the general definition of inter-process communication, which will explain the same thing that we have discussed above.

Definition

"Inter-process communication is used for exchanging useful information between numerous threads in one or more processes (or programs)."

To understand inter process communication, you can consider the following given diagram that illustrates the importance of inter-process communication:



Role of Synchronization in Inter Process Communication

It is one of the essential parts of inter process communication. Typically, this is provided by interprocess communication control mechanisms, but sometimes it can also be controlled by communication processes.

These are the following methods that used to provide the synchronization:

1. **Mutual Exclusion**
2. **Semaphore**
3. **Barrier**
4. **Spinlock**

Mutual Exclusion:-

It is generally required that only one process thread can enter the critical section at a time. This also helps in synchronization and creates a stable state to avoid the race condition.

Semaphore:-

Semaphore is a type of variable that usually controls the access to the shared resources by several processes. Semaphore is further divided into two types which are as follows:

1. Binary Semaphore
2. Counting Semaphore

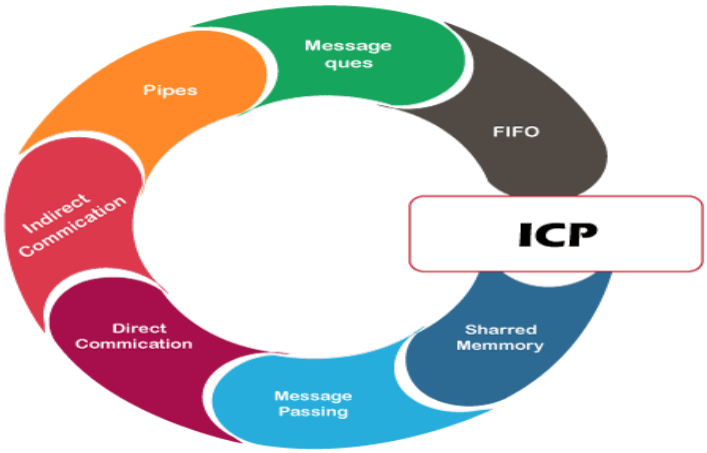
Barrier:-

A barrier typically not allows an individual process to proceed unless all the processes does not reach it. It is used by many parallel languages, and collective routines impose barriers.

Spinlock:-

Spinlock is a type of lock as its name implies. The processes are trying to acquire the spinlock waits or stays in a loop while checking that the lock is available or not. It is known as busy waiting because even though the process active, the process does not perform any functional operation (or task).

Approaches to Interprocess Communication
We will now discuss some different approaches to inter-process communication which are as follows:



These are a few different approaches for Inter- Process Communication:

1. **Pipes**
2. **Shared Memory**
3. **Message Queue**
4. **Direct Communication**
5. **Indirect communication**
6. **Message Passing**
7. **FIFO**

To understand them in more detail, we will discuss each of them individually.

Pipe:-

The pipe is a type of data channel that is unidirectional in nature. It means that the data in this type of data channel can be moved in only a single direction at a time. Still, one can use two-

channel of this type, so that he can able to send and receive data in two processes. Typically, it uses the standard methods for input and output. These pipes are used in all types of POSIX systems and in different versions of window operating systems as well.

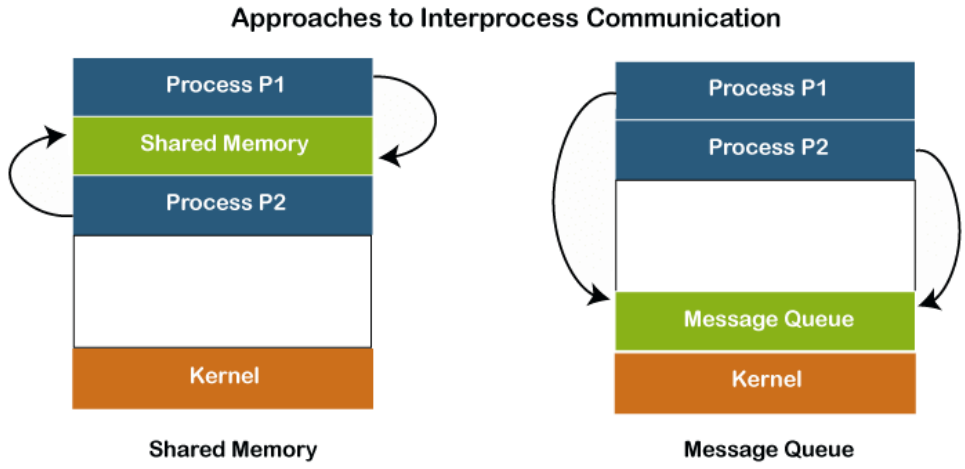
Shared Memory:-

It can be referred to as a type of memory that can be used or accessed by multiple processes simultaneously. It is primarily used so that the processes can communicate with each other. Therefore the shared memory is used by almost all POSIX and Windows operating systems as well.

Message Queue:-

In general, several different messages are allowed to read and write the data to the message queue. In the message queue, the messages are stored or stay in the queue unless their recipients retrieve them. In short, we can also say that the message queue is very helpful in inter-process communication and used by all operating systems.

To understand the concept of Message queue and Shared memory in more detail, let's take a look at its diagram given below:



Message Passing:-

It is a type of mechanism that allows processes to synchronize and communicate with each other. However, by using the message passing, the processes can communicate with each other without restoring the hared variables.

Usually, the inter-process communication mechanism provides two operations that are as follows:

- send (message)
- received (message)

Note: The size of the message can be fixed or variable.

Direct Communication:-

In this type of communication process, usually, a link is created or established between two communicating processes. However, in every pair of communicating processes, only one link can exist.

Indirect Communication

Indirect communication can only exist or be established when processes share a common mailbox, and each pair of these processes shares multiple communication links. These shared links can be unidirectional or bi-directional.

FIFO:-

It is a type of general communication between two unrelated processes. It can also be considered as full-duplex, which means that one process can communicate with another process and vice versa.

Some other different approaches

- **Socket:-**

It acts as a type of endpoint for receiving or sending the data in a network. It is correct for data sent between processes on the same computer or data sent between different computers on the same network. Hence, it used by several types of operating systems.

- **File:-**

A file is a type of data record or a document stored on the disk and can be acquired on demand by the file server. Another most important thing is that several processes can access that file as required or needed.

- **Signal:-**

As its name implies, they are a type of signal used in inter process communication in a minimal way. Typically, they are the massages of systems that are sent by one process to another. Therefore, they are not used for sending data but for remote commands between multiple processes.

Usually, they are not used to send the data but to remote commands in between several processes.

Why we need interprocess communication?

There are numerous reasons to use inter-process communication for sharing the data. Here are some of the most important reasons that are given below:

- It helps to speedup modularity
- Computational
- Privilege separation
- Convenience
- Helps operating system to communicate with each other and synchronize their actions as well.