Unit – 1

Introduction to computer organization Architecture & Design

Computer Organization and Architecture is used to design computer systems. Computer Architecture is considered to be those attributes of a system that are visible to the user like addressing techniques, instruction sets, and bits used for data, and have a direct impact on the logic execution of a program, It defines the system in an abstract manner, It deals with What does the system do.

Whereas, Computer Organization is the way in which a system has to structure and It is operational units and the interconnections between them that achieve the architectural specifications, It is the realization of the abstract model, and It deals with How to implement the system.

Computer Architecture can be defined as a set of rules and methods that describe the functionality, management and implementation of computers. To be precise, it is nothing but rules by which a system performs and operates.

Sub-divisions

Computer Architecture can be divided into mainly three categories, which are as follows –

- Instruction set Architecture or ISA – Whenever an instruction is given to processor, its role is to read and act accordingly. It allocates memory to instructions and also acts upon memory address mode (Direct Addressing mode or Indirect Addressing mode).
- Micro Architecture – It describes how a particular processor will handle and implement instructions from ISA.
- System design – It includes the other entire hardware component within the system such as virtualization, multiprocessing.

Role of computer Architecture

The main role of Computer Architecture is to balance the performance, efficiency, cost and reliability of a computer system.

For Example – Instruction set architecture (ISA) acts as a bridge between computer's software and hardware. It works as a programmer's view of a machine.

Computers can only understand binary language (i.e., 0, 1) and users understand high level language (i.e., if else, while, conditions, etc). So to communicate between user and computer, Instruction set Architecture plays a major role here, translating high level language to binary language.

Structure

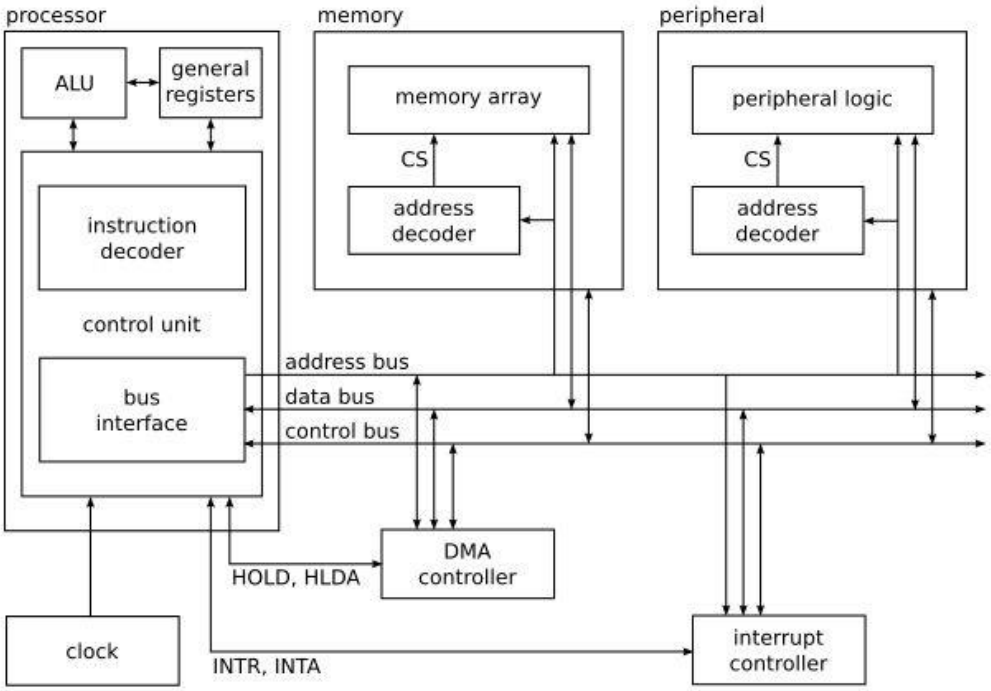
Let us see the example structure of Computer Architecture as given below.

Generally, computer architecture consists of the following –

- Processor
- Memory
- Peripherals

All the above parts are connected with the help of system bus, which consists of address bus, data bus and control bus.

The diagram given below depicts the computer architecture –

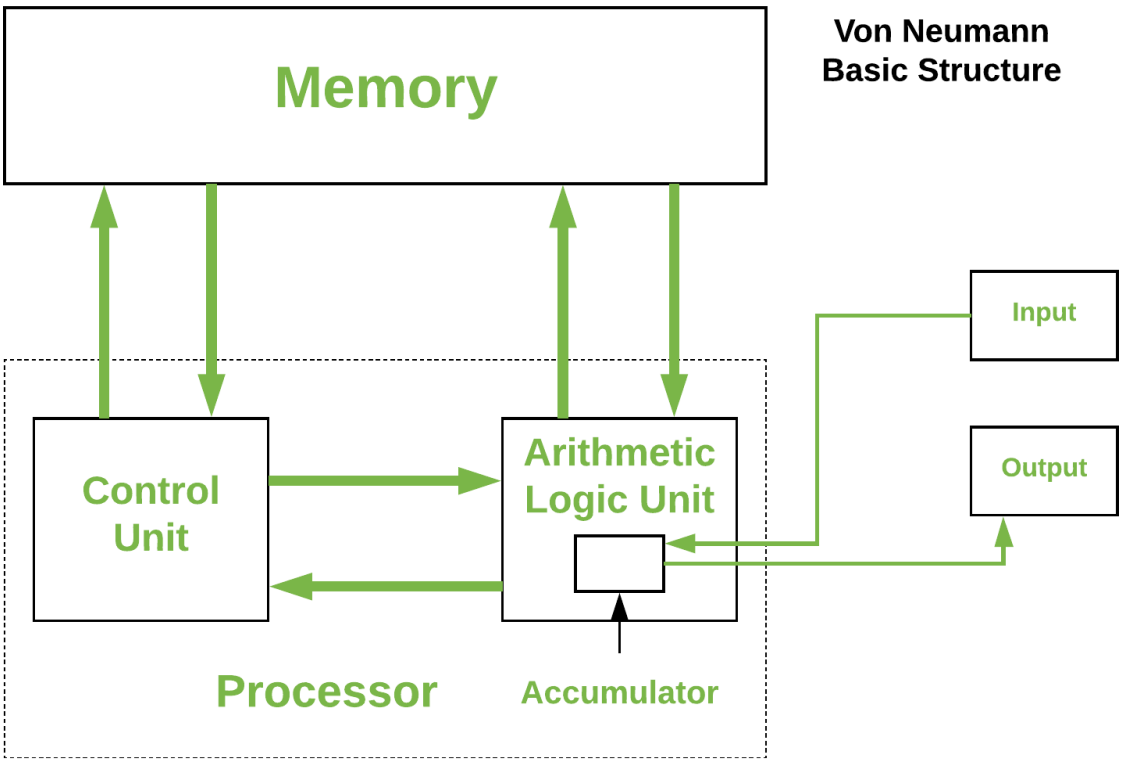


Von-Neumann computer architecture:
Von-Neumann computer architecture design was proposed in 1945.It was later known as Von-Neumann architecture.
Historically there have been 2 types of Computers:

1. **Fixed Program Computers –** Their function is very specific and they couldn't be reprogrammed, e.g. Calculators.
2. **Stored Program Computers –** These can be programmed to carry out many different tasks, applications are stored on them, hence the name.

Modern computers are based on a stored-program concept introduced by John Von Neumann. In this stored-program concept, programs and data are stored in a separate storage unit called memories and are treated the same. This novel idea

meant that a computer built with this architecture would be much easier to reprogram.
The basic structure is like this,



It is also known as **ISA** (Instruction set architecture) computer and is having three basic units:

1. The Central Processing Unit (CPU)
2. The Main Memory Unit
3. The Input/Output Device

Let's consider them in detail.

1. Central Processing Unit-

The central processing unit is defined as the it is an electric circuit used for the executing the instruction of computer program.

It has following major components:

1.Control Unit(CU)

2.Arithmetic and Logic Unit(ALU)

3.variety of Registers

- **Control Unit –**
  A control unit (CU) handles all processor control signals. It directs all input and output flow, fetches code for instructions, and controls how data moves around the system.
- **Arithmetic and Logic Unit (ALU) –**
  The arithmetic logic unit is that part of the CPU that handles all the calculations the CPU may need, e.g. Addition, Subtraction, Comparisons. It performs Logical Operations, Bit Shifting Operations, and Arithmetic operations.

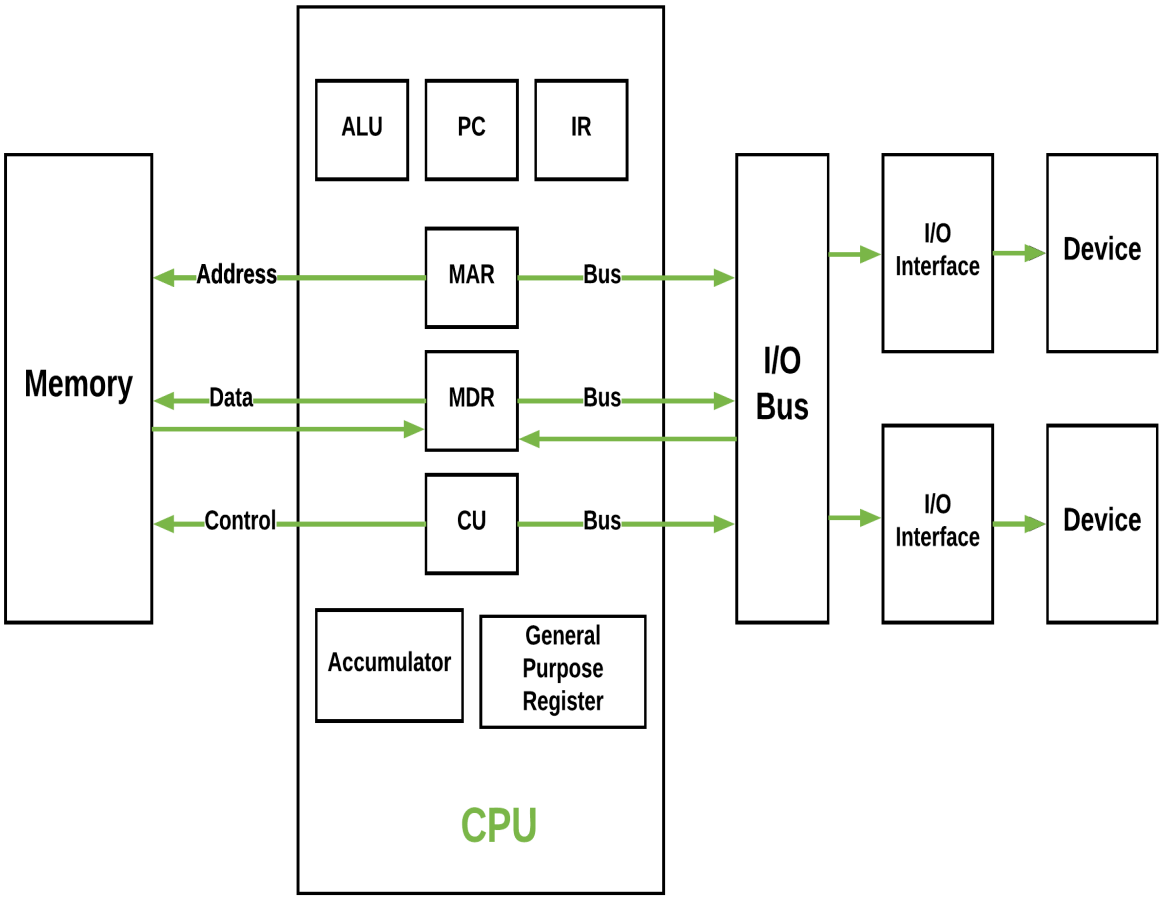


Figure – Basic CPU structure, illustrating ALU

- **Registers –** Registers refer to high-speed storage areas in the CPU. The data processed by the CPU are fetched from the registers. There are different types of registers used in architecture :-
  1. **Accumulator:** Stores the results of calculations made by ALU. It holds the intermediate of arithmetic and logical operatoins.it act as a temporary storage location or device.
  2. **Program Counter (PC):** Keeps track of the memory location of the next instructions to be dealt with. The PC then passes this next address to the Memory Address Register (MAR).
  3. **Memory Address Register (MAR):** It stores the memory locations of instructions that need to be fetched from memory or stored in memory.
  4. **Memory Data Register (MDR):** It stores instructions fetched from memory or any data that is to be transferred to, and stored in, memory.
  5. **Current Instruction Register (CIR):** It stores the most recently fetched instructions while it is waiting to be coded and executed.
  6. **Instruction Buffer Register (IBR):** The instruction that is not to be executed immediately is placed in the instruction buffer register IBR.

- **Buses –** Data is transmitted from one part of a computer to another, connecting all major internal components to the CPU and memory, by the means of Buses. Types:

1. **Data Bus:** It carries data among the memory unit, the I/O devices, and the processor.
2. **Address Bus:** It carries the address of data (not the actual data) between memory and processor.
3. **Control Bus:** It carries control commands from the CPU (and status signals from other devices) in order to control and coordinate all the activities within the computer**.**

- **Input/Output Devices –** Program or data is read into main memory from the *input device* or secondary storage under the control of CPU input instruction. *Output devices* are used to output information from a computer. If some results are evaluated by the computer and it is stored in the computer, then with the help of output devices, we can present them to the user.

Von Neumann bottleneck –
Whatever we do to enhance performance, we cannot get away from the fact that instructions can only be done one at a time and can only be carried out sequentially. Both of these factors hold back the competence of the CPU. This is commonly referred to as the 'Von Neumann bottleneck'. We can provide a Von Neumann processor with more cache, more RAM, or faster components but if original gains are to be made in CPU performance then an influential inspection needs to take place of CPU configuration.
This architecture is very important and is used in our PCs and even in Super Computers.

Register Transfer Language (RTL)

In symbolic notation, it is used to describe the micro-operations transfer among registers. It is a kind of intermediate representation (IR) that is very close to assembly language, such as that which is used in a compiler.The term "Register Transfer" can perform micro-operations and transfer the result of operation to the same or other register.

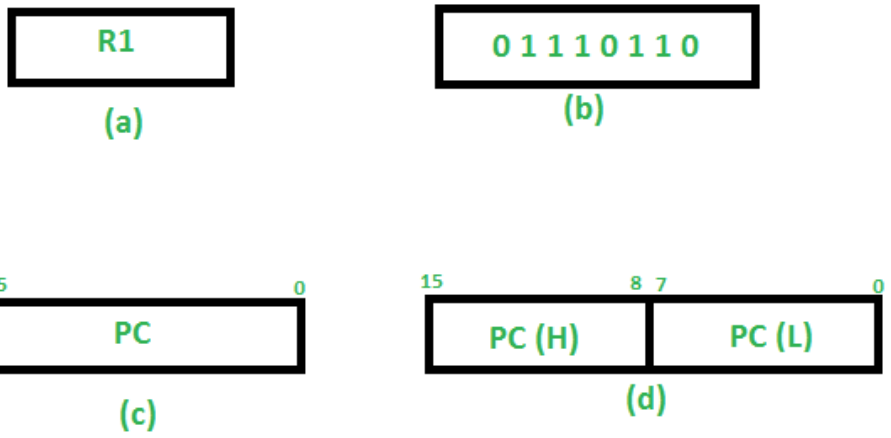
Micro-operations :
The operation executed on the data store in registers are called micro-operations. They are detailed low-level instructions used in some designs to implement complex machine instructions.
Register Transfer :
The information transformed from one register to another register is represented in symbolic form by replacement operator is called Register Transfer.
Replacement Operator :
In the statement, R2 <- R1, **<-** acts as a replacement operator. This statement defines the transfer of content of register R1 into register R2.
There are various methods of RTL –

1. General way of representing a register is by the name of the register enclosed in a rectangular box as shown in (a).

2. Register is numbered in a sequence of 0 to (n-1) as shown in (b).

3. The numbering of bits in a register can be marked on the top of the box as shown in (c).

4. A 16-bit register PC is divided into 2 parts- Bits (0 to 7) are assigned with lower byte of 16-bit address and bits (8 to 15) are assigned with higher bytes of 16-bit address as shown in (d).



Basic symbols of RTL :

Symbol	Description	Example
Letters and Numbers	Denotes a Register	MAR, R1, R2
( )	Denotes a part of register	R1(8-bit) R1(0-7)
<-	Denotes a transfer of information	R2 <- R1
,	Specify two micro-operations of Register Transfer	R1 <- R2 R2 <- R1
:	Denotes conditional operations	P : R2 <- R1 if P=1
Naming Operator (:=)	Denotes another name for an already existing register/alias	Ra := R1

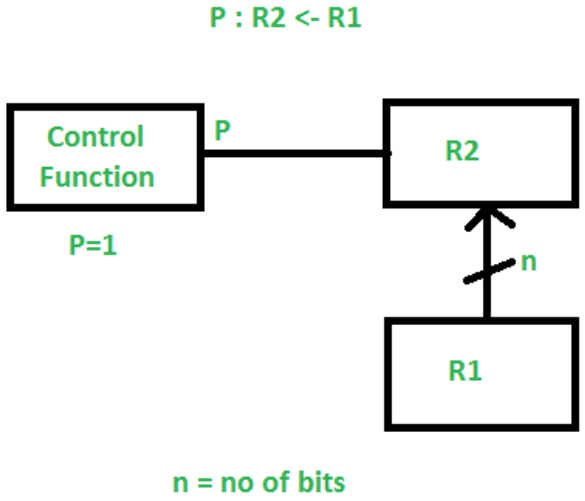
Register Transfer Operations:

The operation performed on the data stored in the registers are referred to as register transfer operations.

There are different types of register transfer operations:

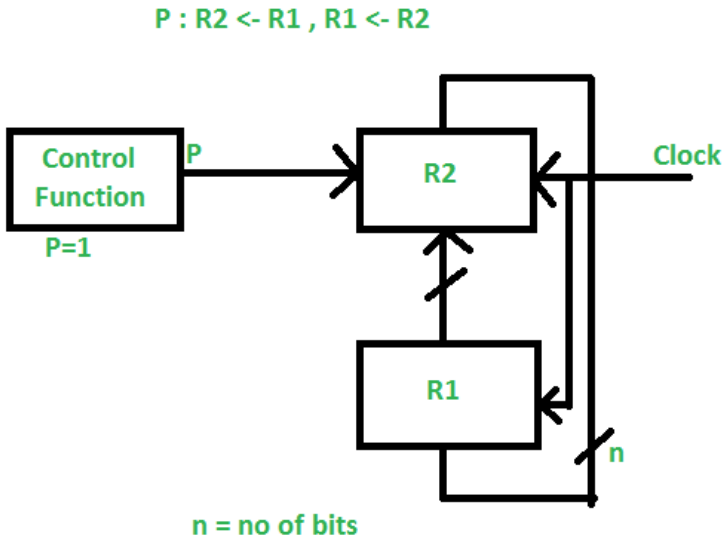
1. Simple Transfer – R2 <- R1
The content of R1 are copied into R2 without affecting the content of R1. It is an unconditional type of transfer operation.

2. Conditional Transfer –



It indicates that if P=1, then the content of R1 is transferred to R2. It is a unidirectional operation.

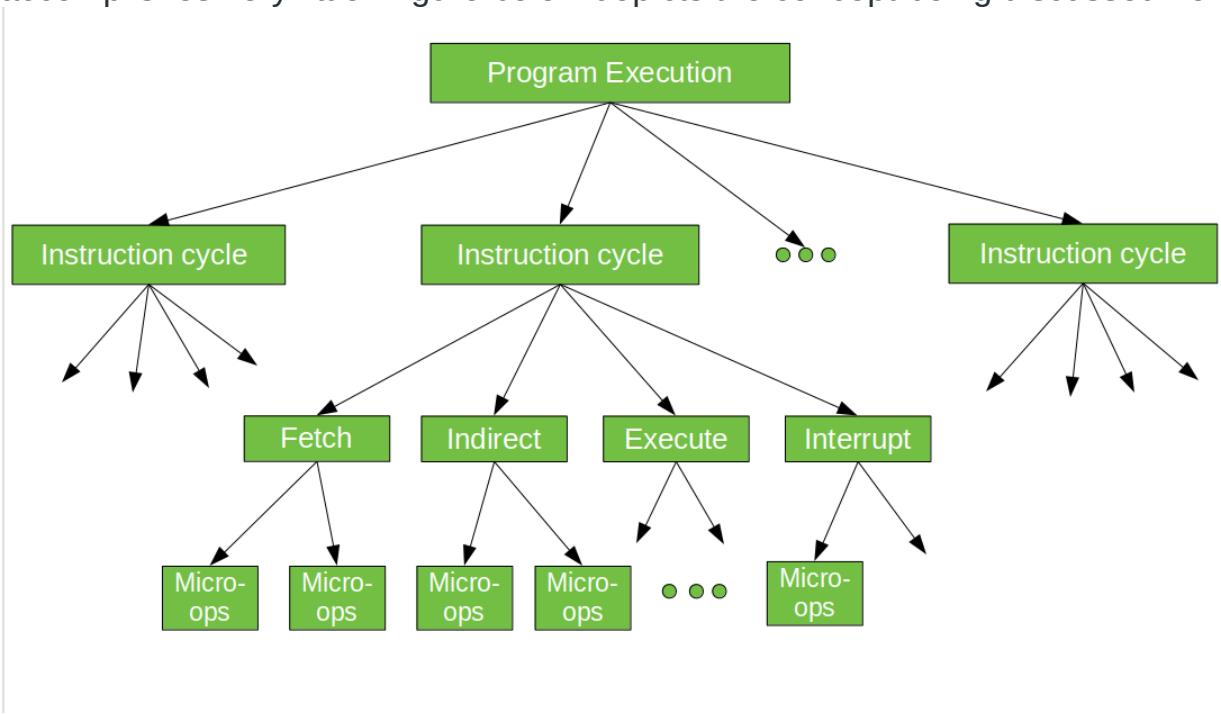
3. Simultaneous Operations –
If 2 or more operations are to occur simultaneously then they are separated with comma **(,)**.



If the control function P=1, then load the content of R1 into R2 and at the same clock load the content of R2 into R1.

Micro-operations

In computer central processing units, **micro-operations** (also known as micro-ops) are the functional or atomic, operations of a processor. These are low level instructions used in some designs to implement complex machine instructions. They generally perform operations on data stored in one or more registers. They transfer data between registers or between external buses of the CPU, also performs arithmetic and logical operations on registers. In executing a program, operation of a computer consists of a sequence of instruction cycles, with one machine instruction per cycle. Each instruction cycle is made up of a number of smaller units – *Fetch, Indirect, Execute and Interrupt cycles.* Each of these cycles involves series of steps, each of which involves the processor registers. These steps are referred as micro-operations. the prefix micro refers to the fact that each of the step is very simple and accomplishes very little. Figure below depicts the concept being discussed here.



Miniature tasks are performed on the information put away in the registers inside the computer chip. They are utilized to perform math and intelligent activities, as well as to move information among registers and memory. A few instances of miniature tasks include:

1.Load: This miniature activity loads information from memory into a register.
2.Store: This miniature activity stores information from a register into memory.
3.Add: This miniature activity adds two qualities and stores the outcome in a register.
4.Subtract: This miniature activity deducts two qualities and stores the outcome in a register.

5.And: This miniature activity plays out a legitimate AND procedure on two qualities and stores the outcome in a register.
6.Or: This miniature activity plays out a legitimate OR procedure on two qualities and stores the outcome in a register.
7.Not: This miniature activity plays out a legitimate NOT procedure on a worth and stores the outcome in a register.
8.Shift: This miniature activity moves the pieces of a worth to the left or right.
9.Rotate: This miniature activity pivots the pieces of a worth to the left or right.
Miniature activities are consolidated to frame more elevated level guidelines and tasks. For instance, an option activity might be executed utilizing various miniature tasks, including a heap activity to stack the qualities into registers, an add activity to play out the option, and a store activity to store the outcome in memory.

Summary: Execution of a program consists of sequential execution of instructions. Each instruction is executed during an instruction cycle made up of shorter sub-cycles(example – fetch, indirect, execute, interrupt). The performance of each sub-cycle involves one or more shorter operations, that is, *micro-operations.* In my next article I will give detailed information of each Instruction Cycle.

Memory Transfer

The transfer of data from a memory word to the external environment is known as a read operation. The read operation in memory transfer is represented as the transfer of data from the address register (AR) with the selected word M for the memory into the memory buffer register (MBR).

[AR]M MBR=Read Operation

The control signal of the read operation starts the read operation. The read operation statement generates the data transfer from the chosen memory register M into the MBR.

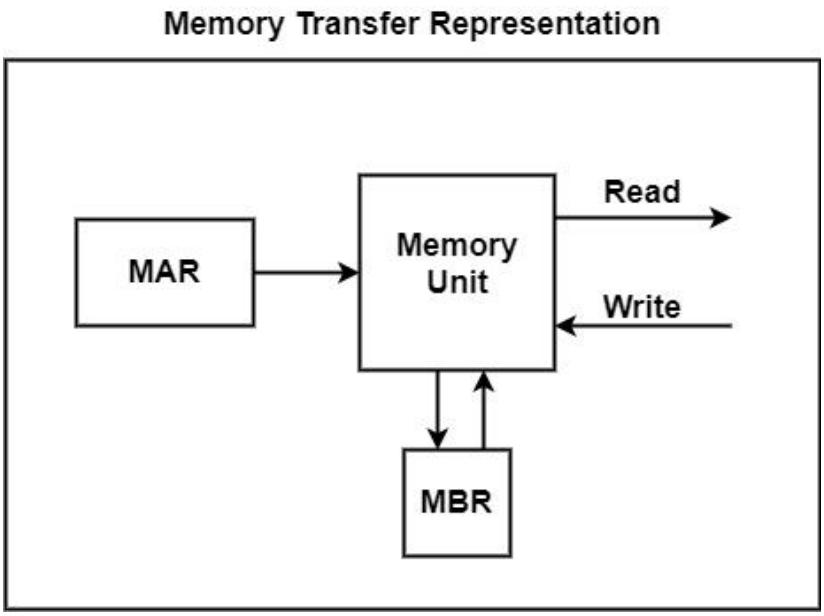
The transfer of new data to be saved into the memory is known as the **write operation**. The memory transfer in the write operation is described as the transfer of data from the memory buffer register (MBR) to the address register (AR) with the chosen word M for the memory.

MBR M [AR] =Write Operation.

The control signal of the write operation starts the write operation. The write operation statement generates the data transfer from the MBR into the chosen memory register through the address shows in the memory M [AR].

It can achieve through a read or write operation, first, the memory register (M) should be selected by a particular address.

The figure shows the memory transfer representation. It demonstrates that the memory unit can transfer the data from the memory address register and memory buffer register to implement read and write operations in the memory transfer.



Bus Transfer

A bus transfer is the most effective method to send data by using a common bus system. It is constructed using common bus registers in multiple registers. The mechanism of the bus includes a collection of lines. These lines are registers of one bit each, which share only one information at a time. The data transfer is contained by the control signals.

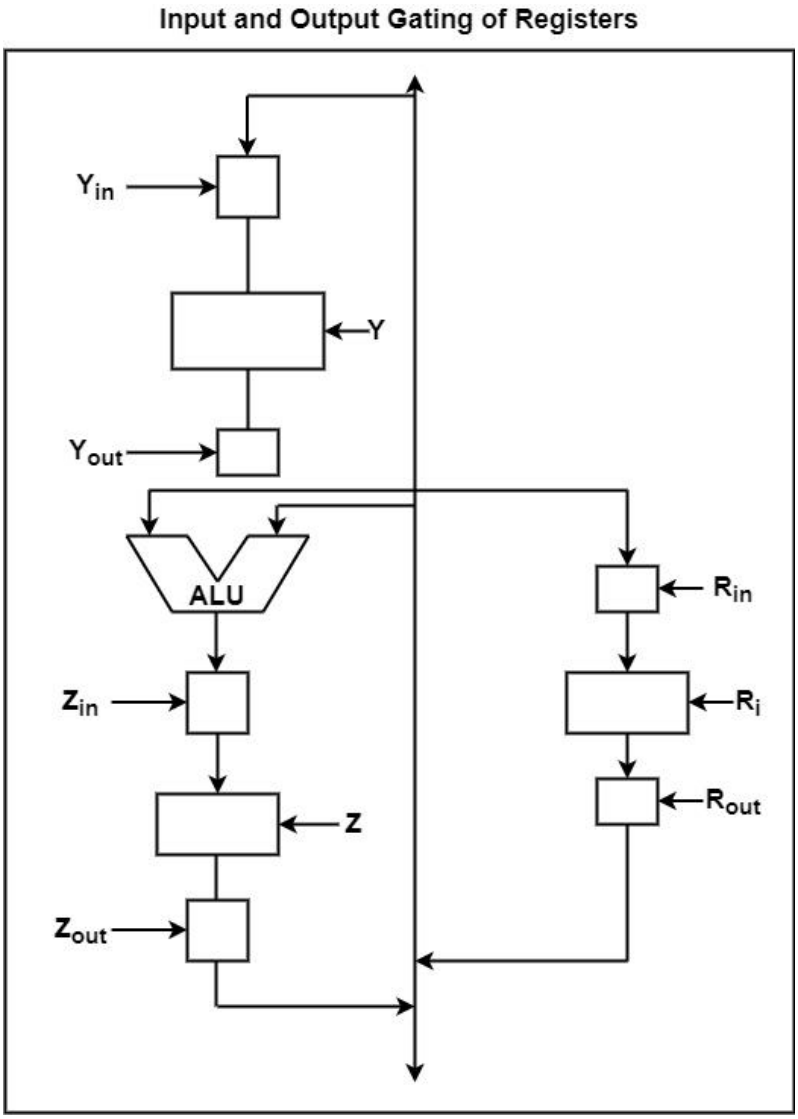
The two methods that can be used in Bus transfer are as follows –

- Using multiplexer
- Using three states bus buffers

Using Multiplexer

A common bus can be generated using a multiplexer. It facilitates in choosing the source register to place the binary data on the bus. The bus register has input and

output gating controlled by control signals. The diagram demonstrates the input and output gating of registers.



- Ri is the register and Rin and Rout are the input and output gating signals of Ri.
- Z is the register and Zin and Zout are the input and output gating signal of register Z.
- Y is the register and Yin and Yout are the input and output signals of Y.

The figure shows input and output gating. The switches are controlled by control signals. Rin and Rout are the input and output gating of the register Ri. When the signal is ON, Ri is set to 1 and when the signal is OFF, Ri is set to 0.

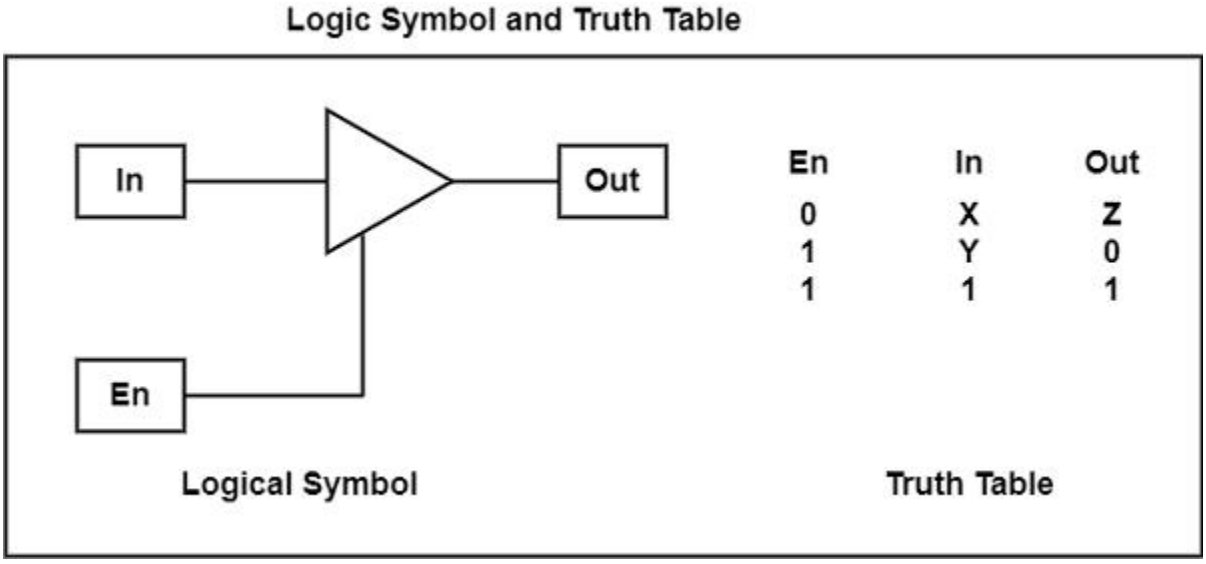
When the input gating Rin is set to 1, the data is loaded into the register bus Ri accessible on the common bus. When Rout is set to 1, the contents of the register Ri are placed on the data bus. It is referred to as input enabled and output enabled signals. The functions that take place inside the processor are in sync with the clock pulse.

Three-State Buffers

Three-state buffers can generate a common bus. The buffer is an area of the memory, which is added in between the other devices to block several interactions and to connect the support. It is established on the three states, 1, 0, and the open circuit. These three states defines are as follows –

- The logic 0 and 1 are the two signals similar to the ones in the conventional gate.
- The high impedance state defines that it does not contain the logic significance and the output is separated.
- These three-state gates can implement any conventional logic AND or NAND, OR, or NOR.

The diagram demonstrates the logic symbols and the associated truth table.



As shown in the figure –

- When the output is allowed and the control input is similar to 1. The logic gate performs as a buffer with the output similar to the input.
- When the input is provided is 0, the gate goes too high impedance state Z and the output is disabled.
- The impedance in three-state buffers linked all the outputs with a cable to produce a common bus line and does not threaten the loading effect.
- The truth table shows that when some input is given and the gate is disabled, it shows in high impedance.
- When the gate is enabled with some input given, then the output results are not in disabled mode.
- When the gate is enabled with input as 1, the output is similar to 1.

Flynn's taxonomy

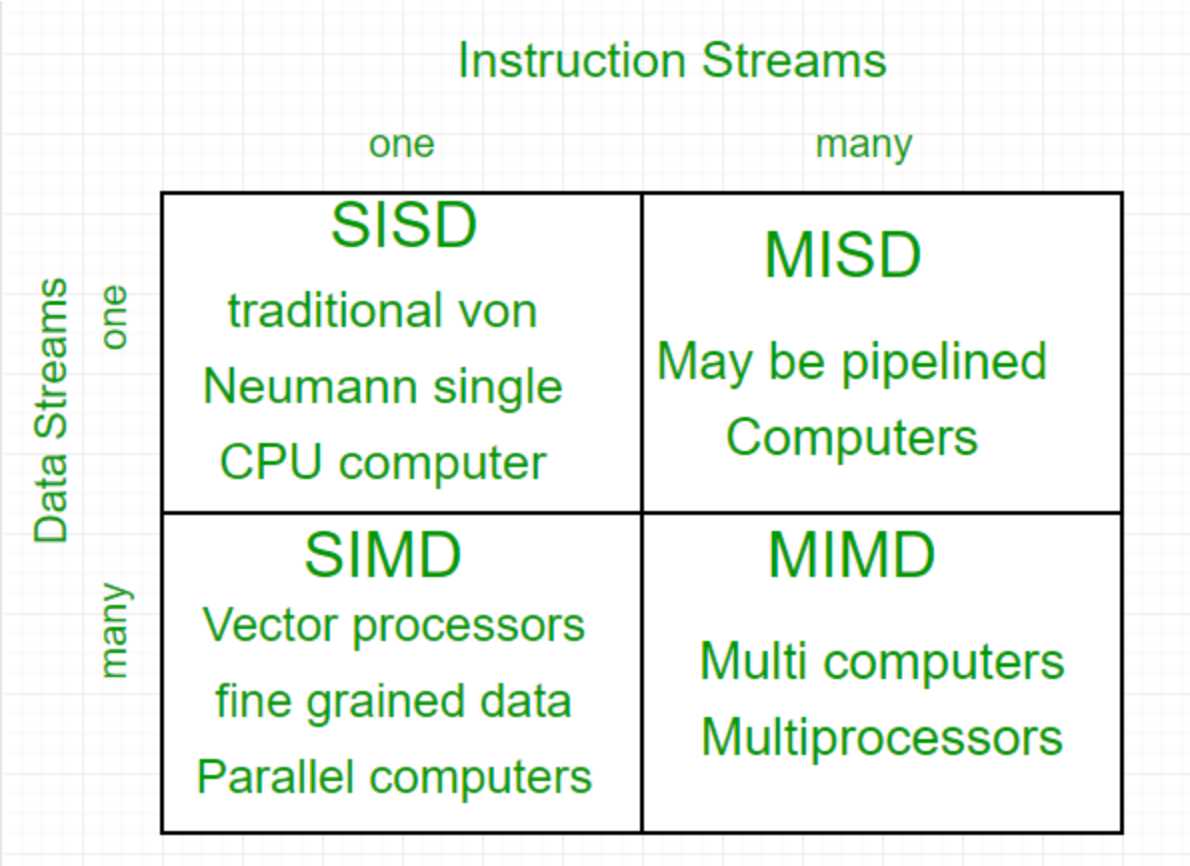
Parallel computing is computing where the jobs are broken into discrete parts that can be executed concurrently. Each part is further broken down into a series of instructions. Instructions from each piece execute simultaneously on different CPUs. The breaking up of different parts of a task among multiple processors will help to reduce the amount of time to run a program. Parallel systems deal with the

simultaneous use of multiple computer resources that can include a single computer with multiple processors, a number of computers connected by a network to form a parallel processing cluster, or a combination of both. Parallel systems are more difficult to program than computers with a single processor because the architecture of parallel computers varies accordingly and the processes of multiple CPUs must be coordinated and synchronized. The difficult problem of parallel processing is portability.

An Instruction Stream is a sequence of instructions that are read from memory. Data Stream is the operations performed on the data in the processor.
Flynn's taxonomy is a classification scheme for computer architectures proposed by Michael Flynn in 1966. The taxonomy is based on the number of instruction streams and data streams that can be processed simultaneously by a computer architecture.
There are four categories in Flynn's taxonomy:

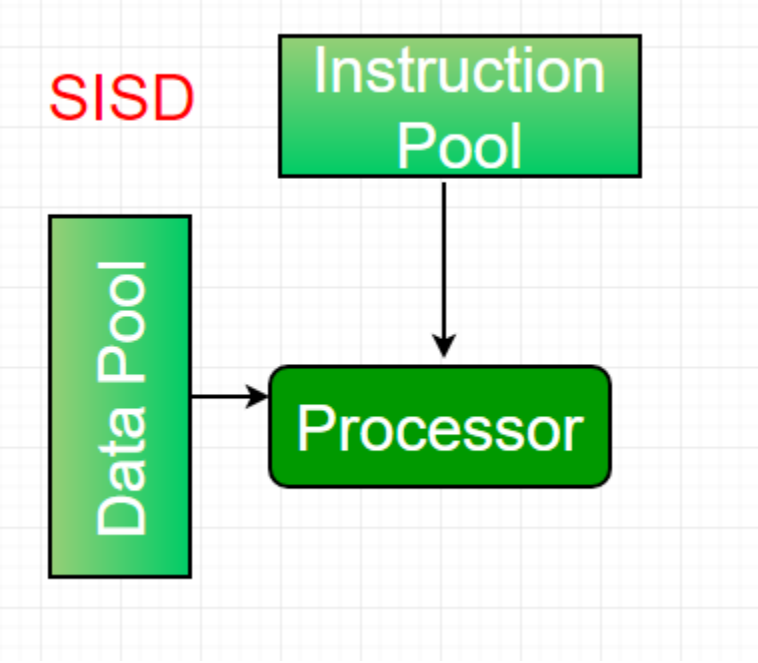
1. Single Instruction Single Data (**SISD**): In a SISD architecture, there is a single processor that executes a single instruction stream and operates on a single data stream. This is the simplest type of computer architecture and is used in most traditional computers.
2. Single Instruction Multiple Data (**SIMD**): In a SIMD architecture, there is a single processor that executes the same instruction on multiple data streams in parallel. This type of architecture is used in applications such as image and signal processing.
3. Multiple Instruction Single Data (**MISD**): In a MISD architecture, multiple processors execute different instructions on the same data stream. This type of architecture is not commonly used in practice, as it is difficult to find applications that can be decomposed into independent instruction streams.
4. Multiple Instruction Multiple Data (**MIMD**): In a MIMD architecture, multiple processors execute different instructions on different data streams. This type of architecture is used in distributed computing, parallel processing, and other high-performance computing applications.

Flynn's taxonomy is a useful tool for understanding different types of computer architectures and their strengths and weaknesses. The taxonomy highlights the importance of parallelism in modern computing and shows how different types of parallelism can be exploited to improve performance.
systems are classified into four major categories:

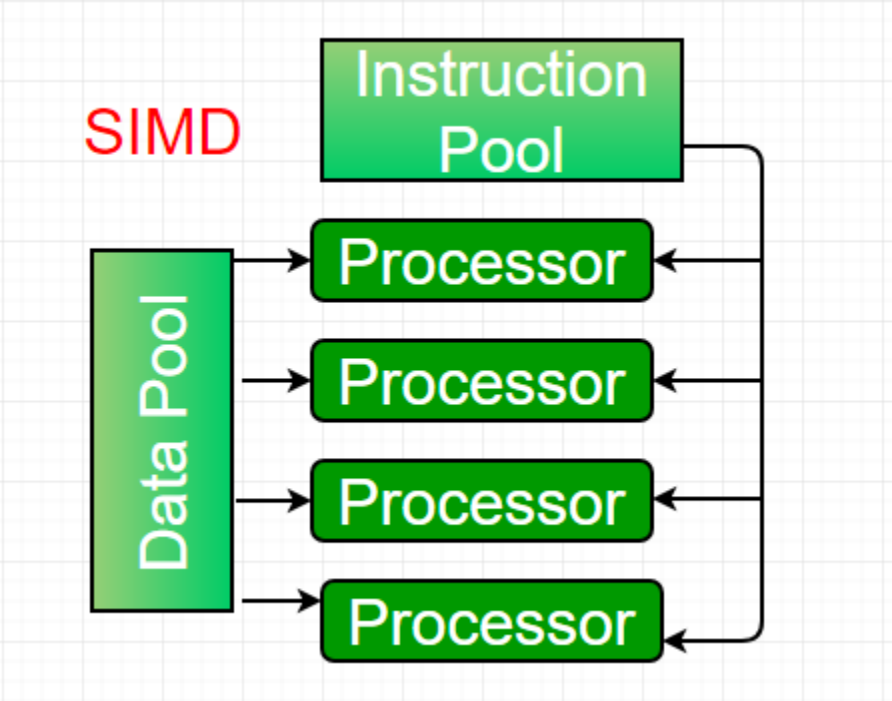


Flynn's classification –

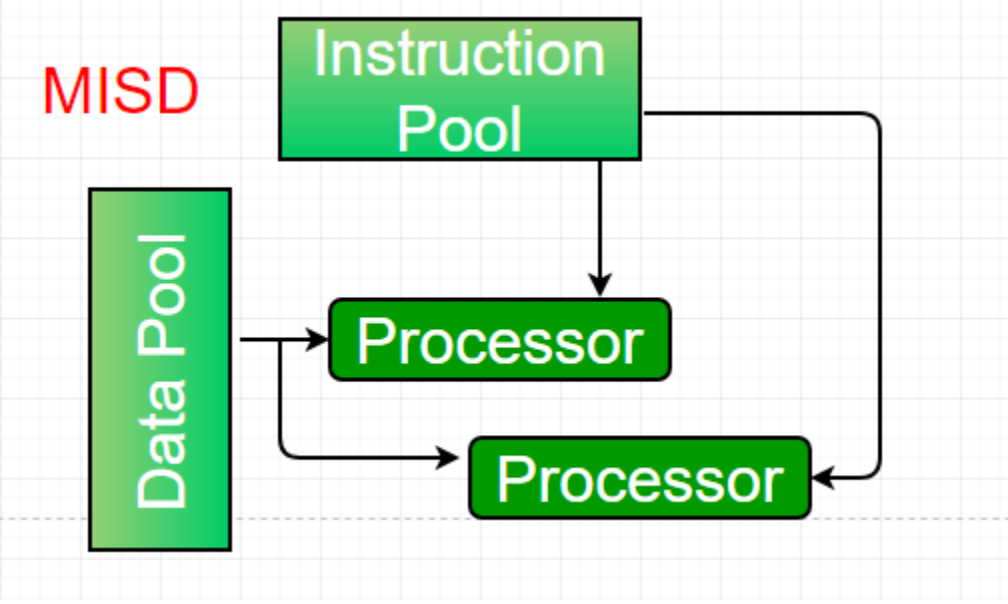
1. **Single-instruction, single-data (SISD) systems –** An SISD computing system is a uniprocessor machine that is capable of executing a single instruction, operating on a single data stream. In SISD, machine instructions are processed in a sequential manner and computers adopting this model are popularly called sequential computers. Most conventional computers have SISD architecture. All the instructions and data to be processed have to be stored in primary memory.



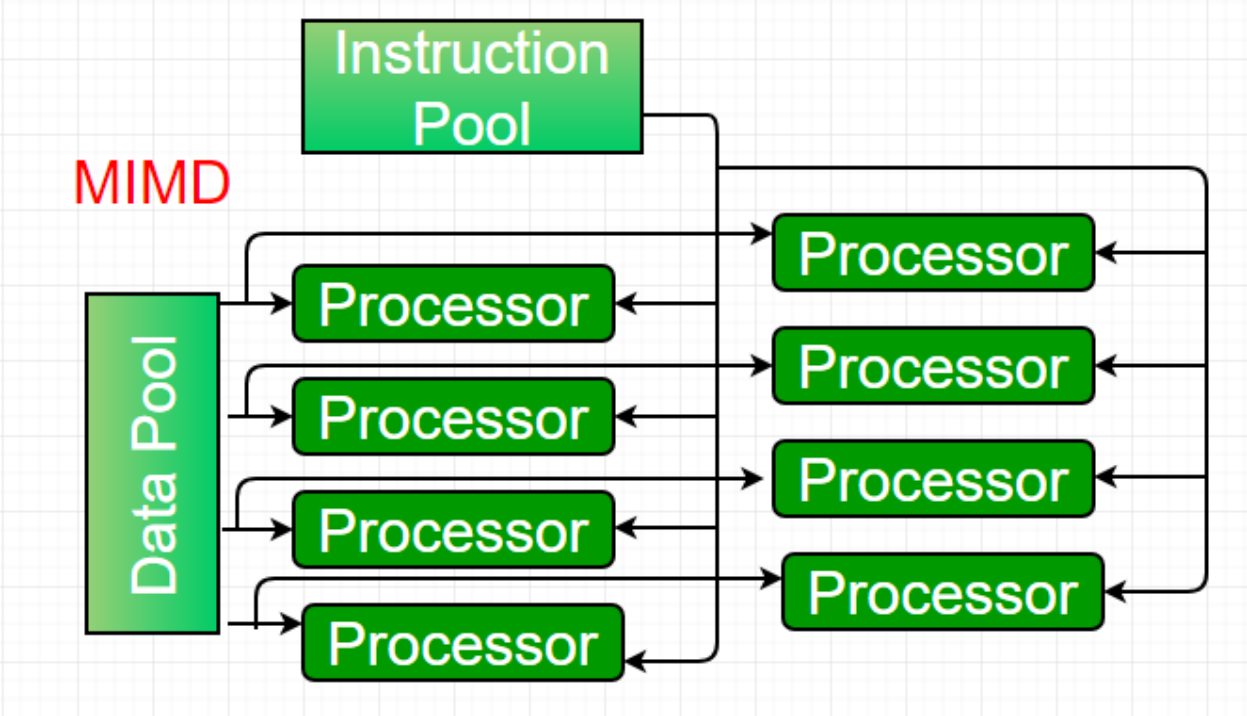
1. The speed of the processing element in the SISD model is limited(dependent) by the rate at which the computer can transfer information internally. Dominant representative SISD systems are IBM PC, and workstations.
2. **Single-instruction, multiple-data (SIMD) systems –** An SIMD system is a multiprocessor machine capable of executing the same instruction on all the CPUs but operating on different data streams. Machines based on a SIMD model are well suited to scientific computing since they involve lots of vector and matrix operations. So that the information can be passed to all the processing elements (PEs) organized data elements of vectors can be divided into multiple sets(N-sets for N PE systems) and each PE can process one data set.



2. Dominant representative SIMD systems are Cray's vector processing machines.
3. **Multiple-instruction, single-data (MISD) systems –** An MISD computing system is a multiprocessor machine capable of executing different instructions on different PEs but all of them operate on the same dataset.



3. Example Z = sin(x)+cos(x)+tan(x) The system performs different operations on the same data set. Machines built using the MISD model are not useful in most applications, a few machines are built, but none of them are available commercially.
4. **Multiple-instruction, multiple-data (MIMD) systems –** An MIMD system is a multiprocessor machine that is capable of executing multiple instructions on multiple data sets. Each PE in the MIMD model has separate instruction and data streams; therefore machines built using this model are capable of any application. Unlike SIMD and MISD machines, PEs in MIMD machines work asynchronously.



4. MIMD machines are broadly categorized into **shared-memory MIMD** and **distributed-memory MIMD** based on the way PEs are coupled to the main memory. In the **shared memory MIMD** model (tightly coupled multiprocessor systems), all the PEs are connected to a single global memory and they all have access to it. The communication between PEs in this model takes place through the shared memory, modification of the data stored in the global memory by one PE is visible to all other PEs. The dominant representative shared memory MIMD systems are Silicon Graphics machines and Sun/IBM's SMP (Symmetric Multi-Processing). In **Distributed memory MIMD** machines (loosely coupled multiprocessor systems) all PEs have a local memory. The communication between PEs in this model takes place through the interconnection network (the inter-process communication channel, or IPC). The network connecting PEs can be configured to tree, mesh, or in accordance with the requirement. The shared-memory MIMD architecture is easier to program but is less tolerant to failures and harder to extend with respect to the distributed memory MIMD model. Failures in a shared-memory MIMD affect the entire system, whereas this is not the case in the distributed model, in which each of the PEs can be easily isolated. Moreover, shared memory MIMD architectures are less likely to scale because the addition of more PEs leads to memory

contention. This is a situation that does not happen in the case of distributed memory, in which each PE has its own memory. As a result of practical outcomes and user requirements, distributed memory MIMD architecture is superior to the other existing models.

Flynn's taxonomy itself does not have any inherent advantages or disadvantages. It is simply a classification scheme for computer architectures based on the number of instruction streams and data streams that can be processed simultaneously. However, the different types of computer architectures that fall under Flynn's taxonomy have their own advantages and disadvantages. Here are some examples:

1. SISD architecture: This is the simplest and most common type of computer architecture. It is easy to program and debug and can handle a wide range of applications. However, it does not offer significant performance gains over traditional computing systems.
2. SIMD architecture: This type of architecture is highly parallel and can offer significant performance gains for applications that can be parallelized. However, it requires specialized hardware and software and is not well-suited for applications that cannot be parallelized.
3. MISD architecture: This type of architecture is not commonly used in practice, as it is difficult to find applications that can be decomposed into independent instruction streams.
4. MIMD architecture: This type of architecture is highly parallel and can offer significant performance gains for applications that can be parallelized. It is well-suited for distributed computing, parallel processing, and other high-performance computing applications. However, it requires specialized hardware and software and can be challenging to program and debug.

Overall, the advantages and disadvantages of different types of computer architectures depend on the specific application and the level of parallelism that can be exploited. Flynn's taxonomy is a useful tool for understanding the different types of computer architectures and their potential uses, but ultimately the choice of architecture depends on the specific needs of the application.

Some additional features of Flynn's taxonomy include:

Concurrency: Flynn's taxonomy provides a way to classify computer architectures based on their concurrency, which refers to the number of tasks that can be executed simultaneously.

Performance: Different types of architectures have different performance characteristics, and Flynn's taxonomy provides a way to compare their performance based on the number of concurrent instructions and data streams.

Parallelism: Flynn's taxonomy highlights the importance of parallelism in computer architecture and provides a framework for designing and analyzing parallel processing systems.

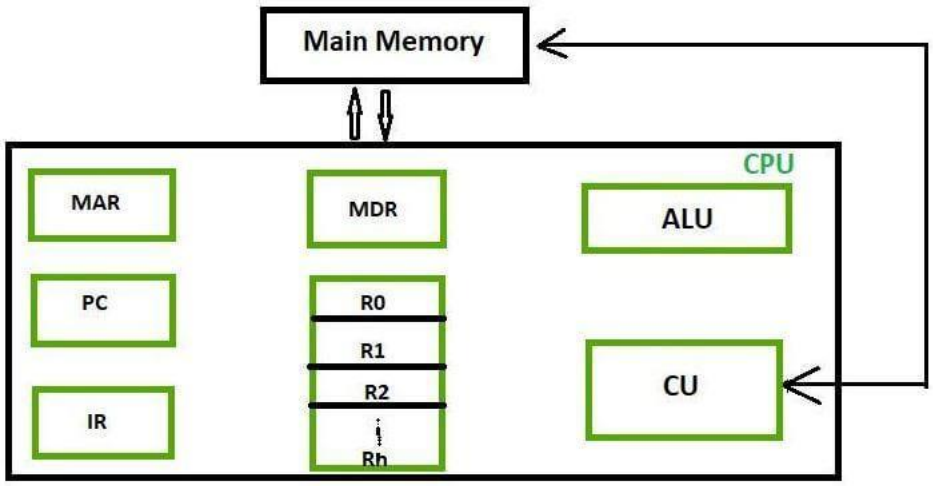
Types of Register in Computer Organization

In Computer Organization, the register is utilized to acknowledge, store, move information and directions that are being utilized quickly by the CPU. There are different kinds of registers utilized for different reasons. Some of the commonly used registers are:

- AC (accumulator)
- DR (Data registers)
- AR (Address registers)
- PC (Program counter)
- MDR (Memory data registers)
- IR (index registers)
- MBR (Memory buffer registers)
- These registers are utilized for playing out the different operations. When we perform some operations, the CPU utilizes these registers to perform the operations. When we provide input to the system for a certain operation, the provided information or the input gets stored in the registers. Once the ALU arithmetic and logical unit process the output, the processed data is again provided to us by the registers.
- The sole reason for having a register is the quick recovery of information that the CPU will later process. The CPU can use RAM over the hard disk to retrieve the memory, which is comparatively a much faster option, but the speed retrieved from RAM is still not enough. Therefore, we have catch memory, which is faster than registers. These registers work with CPU memory like catch and RAM to complete the task quickly.

Operation Performed by Registers

Following major operations performed by registers, such as:



- **Fetch:** The fetch operation is utilized for taking the directions by the client. The instructions that are stored away into the main memory for later processing are fetched by registers.

- **Decode:** This operation is utilized for deciphering the instructions implies the instructions are decoded the CPU will discover which operation is to be performed on the instructions.
- **Execute:** The CPU performs this operation. Also, results delivered by the CPU are then stored in the memory, and after that, they are shown on the client Screen.

Here are the following types of registers in computer organization, such as:

S.NO	NAME	SYMBOL	FUNCTIONING
1	Accumulator	AC	An accumulator is the most often utilized register, and it is used to store information taken from memory.
2	Memory address registers	MAR	Address location of memory is stored in this register to be accessed later. It is called by both MAR and MDR together
3	Memory data registers	MDR	All the information that is supposed to be written or the information that is supposed to be read from a certain memory address is stored here
4	General-purpose register	GPR	Consist of a series of registers generally starting from R0 and running till Rn - 1. These registers tend to store any form of temporary data that is sent to a register during any undertaking process. More GPR enables the register to register addressing, which increases processing speed.
5	Program counter	PC	These registers are utilized in keeping the record of a program that is being executed or under execution. These registers consist of the memory address of the next instruction to be fetched. PC points to the address of the next instruction to be fetched from the main memory when the previous instruction has been completed successfully. Program Counter (PC) also functions to count the number of instructions. The incrementation of PC depends on the type of architecture being used. If we use a 32-bit architecture, the PC gets incremented by 4 every time to fetch the next instruction.

6	Instructions registers	IR	Instruction registers hold the information about to be executed. The immediate instructions received from the system are fetched and stored in these registers. Once the instructions are stored in registers, the processor starts executing the set instructions, and the PC will point to the next instructions to be executed
7	Condition code registers		These have different flags that depict the status of operations. These registers set the flags accordingly if the result of operation caused zero or negative
8	Temporary registers	TR	Holds temporary data
9	Input registers	INPR	Carries input character
10	Output registers	OUTR	Carries output character
11	Index registers	BX	We use this register to store values and numbers included in the address information and transform them into effective addresses. These are also called base registers. These are used to change operand address at the time of execution, also stated as BX
12	Memory buffer register	MBR	MBR - Memory buffer registers are used to store data content or memory commands used to write on the disk. The basic functionality of these is to save called data from memory. MBR is very similar to MDR
13	Stack control registers	SCR	Stack is a set of location memory where data is stored and retrieved in a certain order. Also called last in first out (LIFO), we can only retrieve a stack at the second position only after retrieving out the first one, and stack control registers are mainly used to manage the stacks in the computer. SP - BP is stack control registers. Also, we can use DI, SI, SP, and BP as 2 byte or 4-byte registers. EDI, ESI, ESP, and EBP are 4 - byte registers

14	Flag register	FR	Flag registers are used to indicate a particular condition. The size of the registered flag is 1 - 2 bytes, and each registered flag is furthermore compounded into 8 bits. Each registered flag defines a condition or a flag. The data that is stored is split into 8 separate bits. Basic flag registers - Zero flags Carry flag Parity flag Sign flag Overflow flag.
15	Segment register	SR	Hold address for memory
16	Data register	DX	Hold memory operand