Vikash Polytechnic, Bargarh

Vikash Polytechnic

Campus: Vikash Knowledge Hub, Barahaguda Canal Chowk, NH6 PO/DIST: Bargarh-768028, Odisha

Lecture Note on Computer System Architecture

Diploma 3rd Semester

Submitted By:- Pratyush Ku. Sarangi

Chapter-1

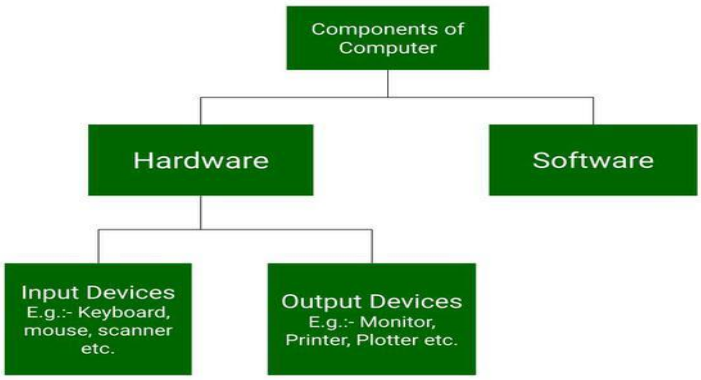
Basic structure of computer hardware

Computer hardware includes the physical parts of a computer, such as a case, central processing unit (CPU), random access memory (RAM), monitor, and mouse which processes the input according to the set of instructions provided to it by the user and gives the desired output.

The computer has mainly has two major components:

1. Hardware
2. Software

In this article, we only discuss computer hardware.



What is Computer Hardware?

Computer hardware is a physical device of computers that we can see and touch. For e.g. Monitor, Central Processing Unit, Mouse, Joystick, etc. Using these devices, we can control computer operations like input and output.

Computer Hardware Parts

These hardware components are further divided into the following categories, which are:

1. Input Devices
2. Output Devices
3. Storage Devices
4. Internal Components

1. Input Devices

Input devices are those devices with the help of which the user interacts with the computer. Or, In other words, with the help of input devices, the user enters the data or information into the computer. This information or data is accepted by the input devices and converted into a computer-acceptable format, which is further sent to the computer system for processing.

- **Keyboard:** It is the most common and main input device for computers. The data is inputted by typing on the keyboard. It consists of 104 keys in total. It contains numeric keys, alphabet keys, and different function keys as well.
- **Mouse:** A mouse is a kind of pointing device which is rolled over to control the cursor on the screen and it has functional keys like left, middle, and right buttons. Using these functional keys, on by the click of which an object is selected or to open a file by just a click of a mouse. It also consists of a sensor inside which

notifies its speed to the computer and according to which the cursor is moved on the screen.

- **Scanner:** As the name suggests, it scans images, documents, etc., and converts them into digital form and that can be further edited and used. It works just like a Xerox machine.
- **Track Ball:** It is a device much like an upside-down mouse. It does not use much space for movement like a mouse. As the trackball remains stationary and the user moves the ball in various directions, it affects the screen movements directly.
- **Light Pen:** It is a light-sensitive device and it is touched to the CRT screen where it can detect, a raster on the screen as it passes by and, with the help of this user can draw anything like lines, figures, or any objects.
- **Microphone:** It is a kind of voice input system that can be attached to a computer system to record sounds. It converts human speech or voice into electrical signals. This electrical signal is processed by the computer and the word is recognized.
- **Optical Character Reader:** It is used to detect alphanumeric characters that are written or printed on paper using a low-frequency light source. This light is absorbed by the dark areas and reflected by the light areas, now this reflected light is received by the photocells. It is like a scanner.
- **Bar Code Reader:** It is used to read bar codes and convert them into electric pulse which will further processed by the computer. Here, the barcode is data that is coded into white and black lines(or light and dark lines).

2. Output Devices

These are the devices that are used to display the output of any task given to the computer in human-readable form.

- **Monitor:** The monitor is the main output device. It is also called VDU(visual display unit) and it looks like a TV screen. The Monitor displays the information from the computer. It is used to display text, video, images, etc.
- **Printer:** A printer is an output device that transfers data from the computer in a printed format by using text or images on paper. There are both colored and black & white printers. Further, there are also different types of printers, like Laser Printer, Dot-matrix printers, and Inkjet printers.
- **Plotter:** It is similar to a printer but potters are large in size. A plotter is used to generate large drawings, architectural blueprints, etc. on paper and these are high-quality images and drawings and large in size.
- **Speakers:** It is a very common output device and it gives sound as an output. Speaker is generally used to play music or anything having sound.

3. Storage Devices

There are some devices that are used for storage purposes and are known as secondary storage devices. Some of them were discussed below:

1. CD (Compact disc): A CD is circular in shape and made up of thin platted glass and plastic polycarbonate material. It has a storage capacity of 600 MB to 700 MB of data. It has a standard size of 12 cm with a hole in the center of about 1.5 cm and 1.2 mm in thickness. There are basically 3 types of CDs, which are:

- **CD-ROM (CD – Read Only Memory):** Contents of this type of CD cannot be erased by the user. Only the publisher is allowed to access the data imprinted on this

CD. CD-ROM is basically used for commercial purposes like for a music album or any application package by a software company.

- **CD-R (CD-Recordable):** In this, content or data can be stored once. After that, they can be read many times but the data or content cannot be rewritten or erased. (Kind of one-time use)
- **CD-RW(CD-Rewritable):** As the name suggests, this type of CD is used to rewrite the content or erase previous content and again write new content many times.

2. DVD (Digital Video/Versatile Disc): A DVD is the same as a CD but with some more features. A DVD comes in single and dual-layer formats. It has much greater storage capacity in comparison to CD. The storage capacity of a DVD with one-sided single layer is – 4.7 GB, one-sided double layer – 8.5 GB, double-sided single layer – 9.4 GB, and double-sided double layer – 17 GB. There are also some types in DVDs, which are :

- **DVD-ROM:** In this type, the contents of the DVD cannot be written on or erased by the user. DVD ROM is used for applications and database for distributing them in large amounts.
- **DVD-R / DVD+R:** DVD-R (DVD minus R) and DVD+R (DVD plus R) are two different kinds of discs and they are once recordable format. Also, they have no difference virtually.
- **DVD-RW / DVD+RW:** This is a kind of rewritable disc and it allows up to 1,000 rewrites.
- **DVD-RAM:** DVD RAM is accessed like a hard disk. It provides high data security and storage capacity. This is a kind of rewritable disc and it allows up to 1,00,000 rewrites.

3. Hard Disk: An hard disk is a non-volatile storage device that uses its read/write heads to store digital data on a magnetic surface of a rigid plate. It is generally 3.5 inches in size for desktops and 2.5 inches in size for laptops. A hard disk can be classified further into 3 types, which are:

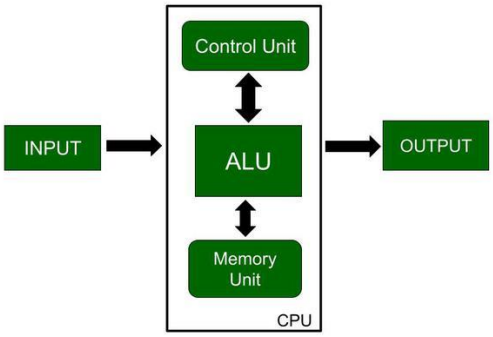
- **Internal Hard Disk:** It has a common storage capacity stated as GB or TB. A system case or cabinet is the place where it is located. It can perform faster operations and its storage is fixed. It is mainly used to store large data files and programs.
- **Internal Cartridges:** The Internal hard disk can't be removed from the system cabinet easily. To resolve this problem Internal Cartridges are introduced. So, Internal cartridges are easy to remove CDs. It has a storage capacity of 2 GB to 160 GB. It is used as an alternative to an internal hard disk.
- **Hard Disk Packs:** It is used by organizations such as banks, and government sector organizations to store large amounts of data. It has a storage capacity of a range of PB(Peta Bytes).

Hardware Components

Some important hardware devices known as the internal components are discussed below:

1. CPU (Central Processing Unit)

The CPU is also known as the heart of the computer. It consists of three units, generally known as the control unit, Arithmetic Logical Unit (ALU), and the memory unit. Below is the block diagram of the CPU is given:



As shown in the diagram input is given to the CPU through input devices. This input goes to memory and the control unit gets instructions from memory. The control unit now decides what to do with the input or instructions and transfers it to ALU. Now, ALU performs various operations like addition, subtraction, multiplication, division, logical operations, etc. After that, the final result gets stored in memory and finally passed to output devices to give the output. So, this is how the CPU works.

2. Motherboard

It is the main circuit board inside a computer and it contains most of the electronic components together. All the components of the computer are directly or indirectly connected to the motherboard. It includes RAM slots, controllers, system chipsets, etc.

3. RAM (Random Access Memory)

It is also known as temporary or volatile memory. It holds the program and data, which are currently in process or processing. All the data is erased as soon as the computer is turned off or in case of a power failure. Data stored in this memory can be changed. There are two types of RAM:-SRAM (Static RAM):DRAM (Dynamic RAM)

4. Video Graphics Array Port

A video input commonly used on computer monitors is called a video graphics array (VGA) port. Verifying that there isn't a loose connection, a damaged cable, or a broken display is one step in troubleshooting a VGA port. Compressed air can also be sprayed inside the VGA port by a computer expert to make sure it's dust-free.

5. Power Supply

All of a computer system's parts are powered by a power source. Typically, a power cord is used to connect a computer tower to an electrical outlet. By turning off the computer, unplugging and separating the power supply cord, or trying a different cord or socket, a technician can diagnose the power supply.

6. Cooling Fan

A computer's system to prevent overheating uses cooling fans. To aid customers who use their computers intensively, such as when streaming video or playing games, many computers contain more than one cooling fan. If a user detects their computer overheating, a computer expert might need to repair the cooling fan. The blades may be examined for any damage and cleared of any foreign objects. A technician's standard method of troubleshooting may involve replacing computer fans.

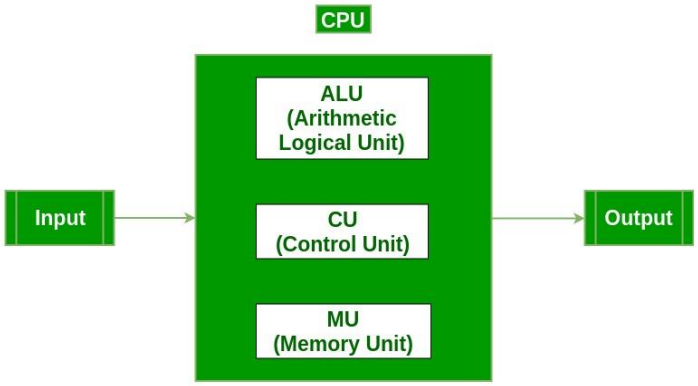
7. Hard Drive

On a computer system, files, programs, and other types of information are stored on hard drives, which are data storage devices. They utilise hard drives, which are magnetically

coated discs used to store digital versions of information. A computer technician can suspect a corrupt hard disk when a hard drive dies.

Functional units of computer

A computer consists of five functionally independent main parts input, memory, arithmetic logic unit (ALU), output and control unit



Input unit: -

The source program/high level language program/coded information/simply data is fed to a computer through input devices keyboard is a most common type. Whenever a key is pressed, one corresponding word or number is translated into its equivalent binary code over a cable & fed either to memory or processor.

Eg: Joysticks, trackballs, mouse, scanners etc are other input devices.

Memory unit: - Its function into store programs and data. It is basically to two types

1. Primary memory

2. Secondary memory

1. Primary memory: -

Is the one exclusively associated with the processor and operates at the electronics speeds programs must be stored in this memory while they are being executed. The memory contains a large number of semiconductors storage cells. Each capable of storing one bit of information. These are processed in a group of fixed site called word. To provide easy access to a word in memory, a distinct address is associated with each word location. Addresses are numbers that identify memory location. Number of bits in each word is called word length of the computer. Programs must reside in the memory during execution. Instructions and data can be written into the memory or read out under the control of processor. Memory in which any location can be reached in a short and fixed amount of time after specifying its address is called random-access memory (RAM). Memory which is only readable by the user and contents of which can't be altered is called read only memory (ROM) it contains operating system. Caches are the small fast RAM units, which are coupled with the processor and are often contained

on the same IC chip to achieve high performance. Although primary storage is essential it tends to be expensive.

2.Secondary memory: -

Is used where large amounts of data & programs have to be stored, particularly information that is accessed infrequently. Examples: - Magnetic disks & tapes, optical disks (ie CD-ROM's), floppies etc.,

Arithmetic logic unit (ALU):- Most of the computer operators are executed in ALU of the processor like addition, subtraction, division, multiplication, etc. the operands are brought into the ALU from memory and stored in high speed storage elements called register. Then according to the instructions the operation is performed in the required sequence. The control and the ALU are may times faster than other devices connected to a computer system. This enables a single processor to control a number of external devices such as key boards, displays, magnetic and optical disks, sensors and other mechanical controllers.

Output unit:- These actually are the counterparts of input unit. Its basic function is to send the processed results to the outside world. Examples:- Printer, speakers, monitor etc.

Control unit:-

It effectively is the nerve center that sends signals to other units and senses their states. The actual timing signals that govern the transfer of data between input unit, processor, memory and output unit are generated by the control unit. Basic operational concepts: - To perform a given task an appropriate program consisting of a list of instructions is stored in the memory. Individual instructions are brought from the memory into the processor, which executes the specified operations. Data to be stored are also stored in the memory.

Examples: - Add LOCA, R0 This instruction adds the operand at memory location LOCA, to operand in register R0 & places the sum into register.

Computer components

Computer components are the essential building parts of developing a functional computer system. The components that make up a computer are called computer components. The processor (CPU), memory, and input/output devices are every computer's three main building blocks. Initially computers were primarily used for numerical computations because any information can be numerically encoded. The ability of computers to interpret information for several purposes was quickly recognized.

There are 5 main computer components that are given below:

- Input Devices
- CPU
- Output Devices
- Primary Memory

- Secondary Memory

Let's look a bit further at each of them.

Input devices

A computer system's input devices are important because they allow users to enter commands and data. Keyboards, mice, scanners, and microphones are numerous examples of input devices.

- The keyboard is the most commonly utilized input device for inserting text and queries into a computer system.
- Mice are another common input tool used to move the cursor on a computer screen.
- Scanners are used for inputting physical documents or images into a computer system.
- Microphones are used to input audio data into a system for computing. They can be used for various tasks, including recording audio for podcasts, participating in video conferences, and creating voice memos for later use.

CPU

A computer machine's "brain" is its central processing unit (CPU). It executes the calculations and commands required for functioning of the computer device. The CPU comprises some components: the control unit, the arithmetic logic unit (ALU), and registers.

- The CPU's control unit is a crucial component. It is in charge of reading and decoding instructions from memory. The right part of the CPU receives and executes these instructions from the control unit.
- The ALU, often known as the arithmetic logic unit, is another crucial CPU part. The ALU operates addition, subtraction, comparison calculations, and other logical and mathematical processes. These operations are carried out using binary logic, which limits operations to the 0 and 1 digits.
- Registers are compact, high-speed data and instruction storage spaces within the CPU. They are utilized to store data that is being processed by the CPU momentarily. Registers are used to accelerate data processing because they are much faster than other forms of memory, such as RAM.
- The CPU's clock speed is yet another crucial aspect that affects overall performance. The clock speed, measured in GHz (gigahertz), controls what number of commands the integral processing unit can process in a second.

Modern CPUs also have additional features like cache memory, virtualization capability, and a couple of cores in addition to the abovementioned components. A small, quick memory called a cache is used to store data and instructions that are utilized frequently. A single CPU may run numerous operating systems thanks to virtualization capability. The CPU can execute numerous tasks simultaneously thanks to multiple cores, enhancing its performance and multitasking capabilities.

Primary Memory

The CPU has direct access to primary memory, sometimes referred to as random access memory (RAM). The data and instructions that are currently being processed are kept in primary memory. The data and instructions are accessed by the CPU from primary memory when a computer programme is running. The information is removed from primary memory once the programme is completed.

- RAM is the most common form of primary memory and is used to store data and instructions that the CPU wishes to access frequently. RAM is volatile, which means that its contents are lost when the computer is turned off. But RAM can be effortlessly and quickly written to and read from, making it a really perfect storage medium for temporary data and instructions.

Other primary memory types, including cache memory, are sometimes used in computer systems. High-speed memory called cache saves information and instructions, which might be utilized often. By lowering the time, the CPU has to wait for data to be received from RAM or secondary storage devices, it is used to speed up the processing of records.

Secondary Memory:

Secondary memory, also called auxiliary storage, is a type of computer memory that is used to store data and programs that aren't currently being utilized by the CPU. In contrast to primary memory, secondary memory is non-volatile, which means that its contents are not lost when the computer is turned off.

There are several types of secondary memory devices, such as hard disk drives (HDD), solid-state drives (SSD), optical disks (including CDs and DVDs), and USB flash drives. These devices have varying storage capacities, read and write speeds, and different capabilities that make them appropriate for different types of applications.

Output Devices:

Output devices are hardware components of a computer system that are used to show or send data from the pc to the user or any other device. They enable customers to view and engage with the information and applications the computer is processing. Speakers, projectors, printers, and monitors are a few examples of output devices.

- Monitors are the most frequently used output devices used to show data on a computer machine. They may be used to show photos, videos, and different forms of data and exist in various sizes and resolutions.
- Printers are another form of output device this is used to print hard copies of papers and other styles of data. They include inkjet and laser printers and are available in various sizes and brands. While laser printers utilize toner to make speedy, high-volume prints, inkjet printers employ liquid ink to produce high-quality prints.
- Speakers are used to output sound from a computer system. They can be connected externally or incorporated into the computer system. They enable users to interact with other forms of multimedia material, view videos, and listen to music.
- Projectors are output device that displays huge images and videos on a screen or wall. They are frequently utilized in presentations and other occasions that call for a sizable display.

Performance Measures

In computer organization, performance refers to the speed and efficiency at which a computer system can execute tasks and process data. A high-performing computer system is one that can perform tasks quickly and efficiently while minimizing the amount of time and resources required to complete these tasks.

There are several factors that can impact the performance of a computer system, including:

- **Processor speed:** The speed of the processor, measured in GHz (gigahertz), determines how quickly the computer can execute instructions and process data.
- **Memory:** The amount and speed of the memory, including RAM (random access memory) and cache memory, can impact how quickly data can be accessed and processed by the computer.
- **Storage:** The speed and capacity of the storage devices, including hard drives and solid-state drives (SSDs), can impact the speed at which data can be stored and retrieved.
- **I/O devices:** The speed and efficiency of input/output devices, such as keyboards, mice, and displays, can impact the overall performance of the system.
- **Software optimization:** The efficiency of the software running on the system, including operating systems and applications, can impact how quickly tasks can be completed.

Improving the performance of a computer system typically involves optimizing one or more of these factors to reduce the time and resources required to complete tasks. This can involve upgrading hardware components, optimizing software, and using specialized performance-tuning tools to identify and address bottlenecks in the system.

Computer performance is the amount of work accomplished by a computer system. The word performance in computer performance means “How well is the computer doing the work it is supposed to do?”. It basically depends on the response time, throughput, and

execution time of a computer system. **Response time** is the time from the start to completion of a task. This also includes:

- Operating system overhead.
- Waiting for I/O and other processes
- Accessing disk and memory
- Time spent executing on the CPU or execution time.

Throughput is the total amount of work done in a given time. **CPU execution time** is the total time a CPU spends computing on a given task. It also excludes time for I/O or running other programs. This is also referred to as simply CPU time. Performance is determined by execution time as performance is inversely proportional to execution time.

```
Performance = (1 / Execution time)
```

And,

```
(Performance of A /  Performance of B)
= (Execution Time of B / Execution Time of A)
```

If given that Processor A is faster than processor B, that means execution time of A is less than that of execution time of B. Therefore, performance of A is greater than that of performance of B. **Example** – Machine A runs a program in 100 seconds, Machine B runs the same program in 125 seconds

```
(Performance of A /  Performance of B)
= (Execution Time of B / Execution Time of A)
= 125 / 100 = 1.25
```

That means machine A is 1.25 times faster than Machine B. And, the time to execute a given program can be computed as:

```
Execution time  = CPU clock cycles x clock cycle time
```

Since clock cycle time and clock rate are reciprocals, so,

```
Execution time  = CPU clock cycles / clock rate
```

The number of CPU clock cycles can be determined by,

```
CPU clock cycles
= (No. of instructions / Program ) x (Clock cycles / Instruction)
= Instruction Count x CPI
```

Which gives,

```
Execution time
= Instruction Count x CPI x clock cycle time
= Instruction Count x CPI / clock rate
```

Units for CPU Execution Time

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

How to Improve Performance?

To improve performance you can either:

- Decrease the CPI (clock cycles per instruction) by using new Hardware.
- Decrease the clock time or Increase clock rate by reducing propagation delays or by use pipelining.
- Decrease the number of required cycles or improve ISA or Compiler.

Execution Upgrade Procedures

The Exhibition upgrade on computer chip execution time is worked with by the accompanying variables in a significant manner.
1.Internal Engineering of the computer chip
2.Instruction Arrangement of the central processor
3.Memory Speed and transmission capacity
4.Percentage utilization of the registers in execution (note: Registers are something like multiple times quicker than memory).

Uses and Benefits of Performance of Computer

Some of the key uses and benefits of a high-performing computer system include:

- **Increased productivity:** A high-performing computer can help increase productivity by reducing the time required to complete tasks, allowing users to complete more work in less time.
- **Improved user experience:** A fast and efficient computer system can provide a better user experience, with smoother operation and fewer delays or interruptions.
- **Faster data processing:** A high-performing computer can process data more quickly, enabling faster access to critical information and insights.
- **Enhanced gaming and multimedia performance:** High-performance computers are better suited for gaming and multimedia applications, providing smoother and more immersive experiences.
- **Better efficiency and cost savings:** By optimizing the performance of a computer system, it is possible to reduce the time and resources required to complete tasks, leading to better efficiency and cost savings.

Chapter-2

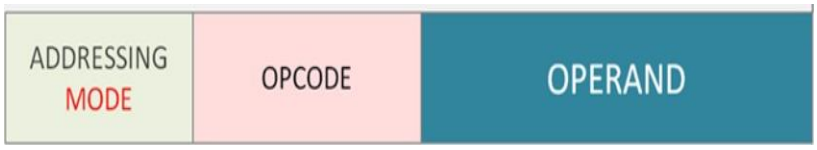
2.1 Fundamentals to instruction

Instruction

Set of commands that we give to processor to which task to do and how to perform that task.

So whatever be the processor (like X86,AMD,intel Pentium etc.) it will perform the task according to the instruction on the basis of provided commands.

Instruction has 3 parts



Types of instruction

1. Data transfer instruction
2. Data manipulation instruction
3. Program control instruction

1. Data transfer instruction

We can assume that according to the name, if we want to transfer data then these instruction we can use

MOV,LD,STA,XCHG, PUSH, POP etc.

If we want to transfer the data between the register to register, register to memory & vice-versa, memory to memory, from input device to O/P device & vice-versa then we need the help of these instruction.

2.Data manipulation instruction

Whenever we want to change to change the existing data we can take the help of the operation in order to perform addition, subtraction, multiplication, AND, OR, Left shift etc.

- Arithmetic (+,-,*,/)
- Logical(AND, OR ,NOT, NOR, NAND, XOR)
- Shift/conversion(left shift, right shift, binary to decimal, octal to hexadecimal etc.)

3.Program control instruction

Whenever we want to use the decision making or control or repetitive statement based instruction we can go for these instruction

- Decision making(if, if-else, nested-if)
- Control statement(do, do-while, for)

2.2 Opcode

- Opcode stands for operation code

- It is the first part of the instruction that tells the computer what function/task is to be perform

E.g. MOV, ADD, SUB etc.

2.3 Operand

- it is the second part of the instruction which indicates the computer system with whom the task is going to perform/where to store the data/where to manipulate the data

- operand may be register, memory or constant

E.g.: MOV R1, R2

Here R1 & R2 are the operand

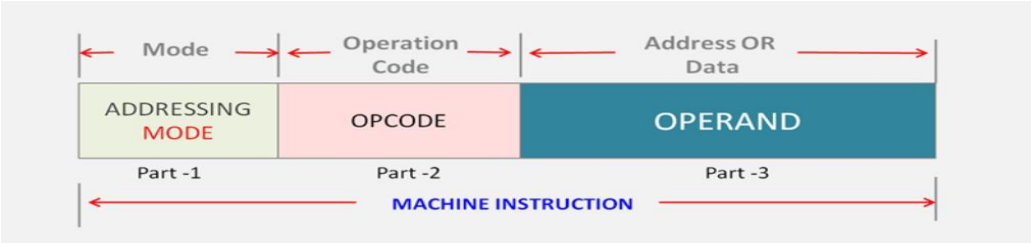
2.4 Instruction format

These consists of 3 parts

1.addressing mode – in which instruction mode we are going to perform the instruction

2.operation code – what is the operation code

3.operand – what are the operand , to whom with perform the operation



Instruction Formats (Zero, One, Two and Three Address Instruction)

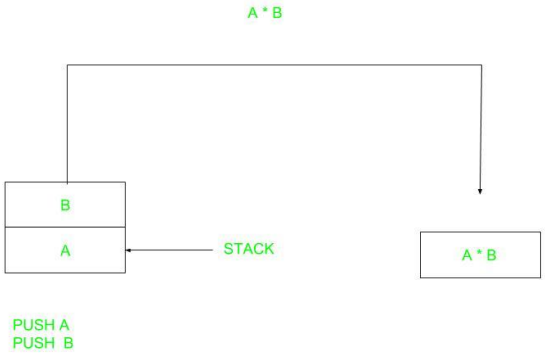
In computer organization, instruction formats refer to the way instructions are encoded and represented in machine language. There are several types of instruction formats, including zero, one, two, and three-address instructions.
Each type of instruction format has its own advantages and disadvantages in terms of code size, execution time, and flexibility. Modern computer architectures typically use a combination of these formats to provide a balance between simplicity and power.

Types of Instructions

Based on the number of addresses, instructions are classified as:
NOTE: We will use the X = (A+B)*(C+D) expression to showcase the procedure.

Zero Address Instructions

These instructions do not specify any operands or addresses. Instead, they operate on data stored in registers or memory locations implicitly defined by the instruction. For example, a zero-address instruction might simply add the contents of two registers together without specifying the register names.



A stack-based computer does not use the address field in the instruction. To evaluate an expression first it is converted to reverse Polish Notation i.e. Postfix Notation.
Expression: X = (A+B)*(C+D)
Postfixed : X = AB+CD+*
TOP means top of stack
M[X] is any memory location

PUSH	A	TOP = A
PUSH	B	TOP = B
ADD		TOP = A+B
PUSH	C	TOP = C
PUSH	D	TOP = D
ADD		TOP = C+D
MUL		TOP = (C+D)*(A+B)

PUSH	A	TOP = A
POP	X	M[X] = TOP

One Address Instructions

These instructions specify one operand or address, which typically refers to a memory location or register. The instruction operates on the contents of that operand, and the result may be stored in the same or a different location. For example, a one-address instruction might load the contents of a memory location into a register.

This uses an implied ACCUMULATOR register for data manipulation. One operand is in the accumulator and the other is in the register or memory location. Implied means that the CPU already knows that one operand is in the accumulator so there is no need to specify it.


```
Expression: X = (A+B)*(C+D)
AC is accumulator
M[] is any memory location
M[T] is temporary location
```

LOAD	A	AC = M[A]
ADD	B	AC = AC + M[B]
STORE	T	M[T] = AC
LOAD	C	AC = M[C]
ADD	D	AC = AC + M[D]
MUL	T	AC = AC * M[T]
STORE	X	M[X] = AC

Two Address Instructions

These instructions specify two operands or addresses, which may be memory locations or registers. The instruction operates on the contents of both operands, and the result may be stored in the same or a different location. For example, a two-address instruction might add the contents of two registers together and store the result in one of the registers.
This is common in commercial computers. Here two addresses can be specified in the instruction. Unlike earlier in one address instruction, the result was stored in the accumulator, here the result can be stored at different locations rather than just accumulators, but require more number of bit to represent the address.

Here destination address can also contain an operand.

```
Expression: X = (A+B)*(C+D)
R1, R2 are registers
M[] is any memory location
```

MOV	R1, A	R1 = M[A]
ADD	R1, B	R1 = R1 + M[B]
MOV	R2, C	R2 = M[C]
ADD	R2, D	R2 = R2 + M[D]
MUL	R1, R2	R1 = R1 * R2
MOV	X, R1	M[X] = R1

Three Address Instructions

These instructions specify three operands or addresses, which may be memory locations or registers. The instruction operates on the contents of all three operands, and the result may be stored in the same or a different location. For example, a three-address instruction might multiply the contents of two registers together and add the contents of a third register, storing the result in a fourth register.

This has three address fields to specify a register or a memory location. Programs created are much short in size but number of bits per instruction increases. These instructions make the creation of the program much easier but it does not mean that program will run much faster because now instructions only contain more information but each micro-operation (changing the content of the register, loading address in the address bus etc.) will be performed in one cycle only.


```
Expression: X = (A+B)*(C+D)
R1, R2 are registers
M[] is any memory location
```

ADD	R1, A, B	R1 = M[A] + M[B]
ADD	R2, C, D	R2 = M[C] + M[D]
MUL	X, R1, R2	M[X] = R1 * R2

Advantages of Zero-Address, One-Address, Two-Address and Three-Address Instructions

Zero-address instructions

- They are simple and can be executed quickly since they do not require any operand fetching or addressing. They also take up less memory space.

One-address instructions

- They allow for a wide range of addressing modes, making them more flexible than zero-address instructions. They also require less memory space than two or three-address instructions.

Two-address instructions

- They allow for more complex operations and can be more efficient than one-address instructions since they allow for two operands to be processed in a single instruction. They also allow for a wide range of addressing modes.

Three-address instructions

- They allow for even more complex operations and can be more efficient than two-address instructions since they allow for three operands to be processed in a single instruction. They also allow for a wide range of addressing modes.

Disadvantages of Zero-Address, One-Address, Two-Address and Three-Address Instructions

Zero-address instructions

- They can be limited in their functionality and do not allow for much flexibility in terms of addressing modes or operand types.

One-address instructions

- They can be slower to execute since they require operand fetching and addressing.

Two-address instructions

- They require more memory space than one-address instructions and can be slower to execute since they require operand fetching and addressing.

Three-address instructions

- They require even more memory space than two-address instructions and can be slower to execute since they require operand fetching and addressing.

2.5 Addressing Modes

The term addressing modes refers to the way in which the operand of an instruction is specified. The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually executed.

Types of addressing modes

1. Implied addressing mode
2. Immediate addressing mode
3. Register addressing mode
4. Register indirect addressing mode
5. Auto increment decrement addressing mode
6. Direct addressing mode
7. Indirect addressing mode
8. Relative addressing mode
9. Base register addressing mode
10. Indexed addressing mode

1. Implied addressing mode

- Operand is specified implicitly in the definition of the instruction
- Used for zero address & one-address instruction
- Means the value of operand is implicitly defined in the instruction itself

E.g. INC A

CLC (used to reset Carry flag to 0)

2. Immediate addressing mode

Operand is directly provided as constant in this addressing mode.

No computation required to calculate effective address because the value of constant is same throughout the program.

Here we don't go to any address location or any register, we are dealing with only constants

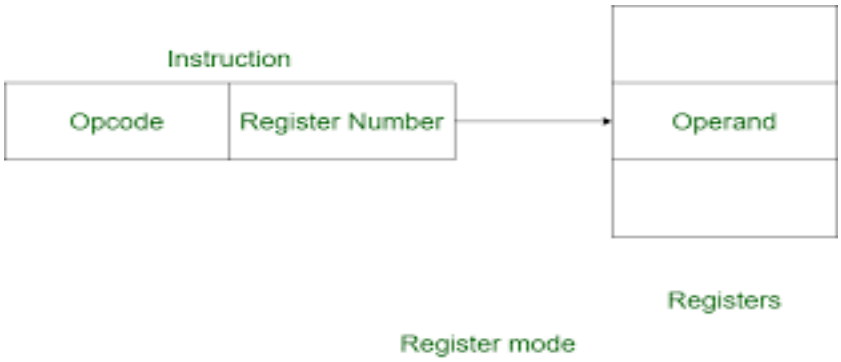
E.g.: ADD R1, #3

3.Register addressing mode

Operand is present in the register

Register number is present in the instruction.

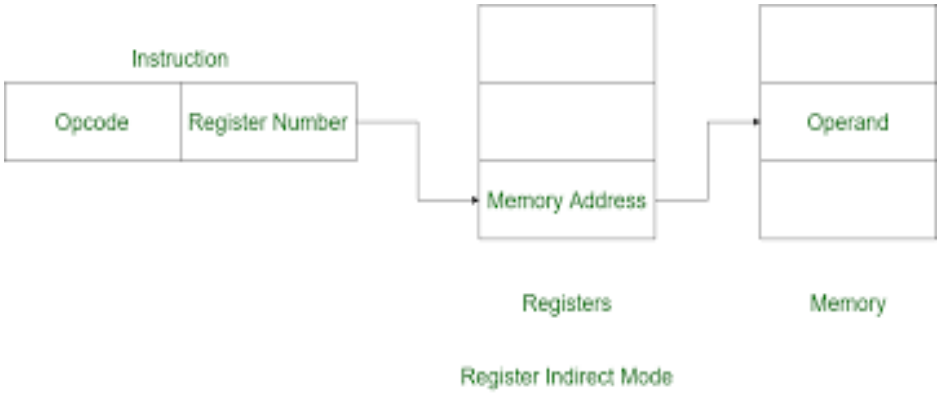
We have to go to the register no. as specified by the instruction ,pick the value & execute the instruction as specified by the opcode



Register mode

4.Register indirect addressing mode

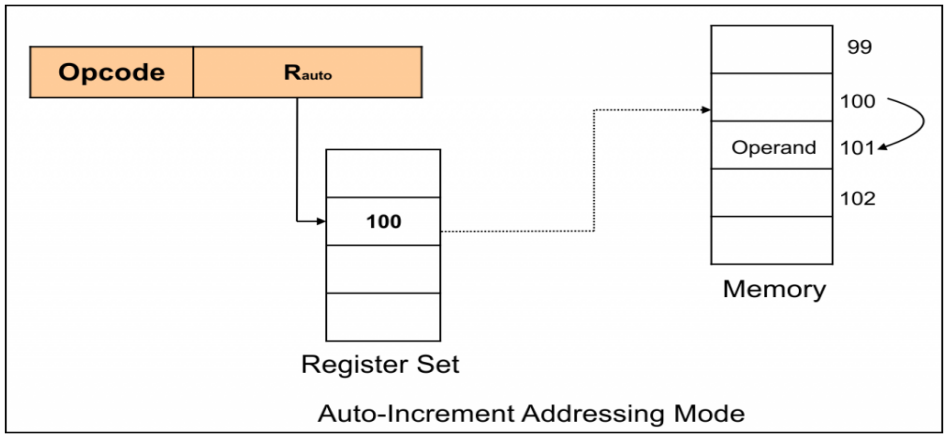
Register contains the address of operand rather than operand itself



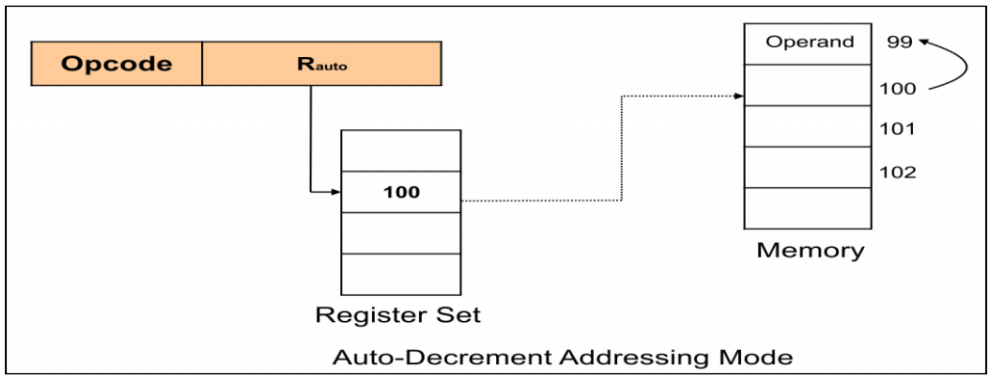
Register Indirect Mode

5.Auto increment & auto decrement addressing mode

In the case of auto increment mode, the content present in the register is initially incremented, and then the content that is incremented in the register is used in the form of an effective address. Once the content present in the register in the auto-increment addressing mode is accessed by its instruction, the content of the register is incremented to refer to the next operand.

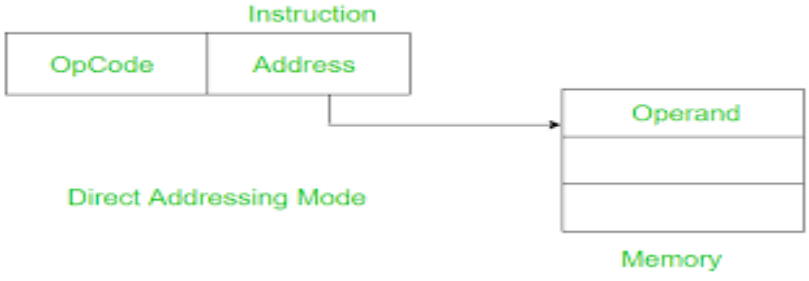


In the case of auto decremented mode, the content present in the register is initially decremented, and then the content that is decremented in the register is used in the form of an effective address. Once the content present in the register in the auto- decrement addressing mode is accessed by its instruction, the content of the register is decremented to refer to the next operand.



6.Direct addressing mode

In this addressing mode actual address given in the instruction. CPU reads the address, goes to the address & fetches the data



Eg: Add R1, (1001)

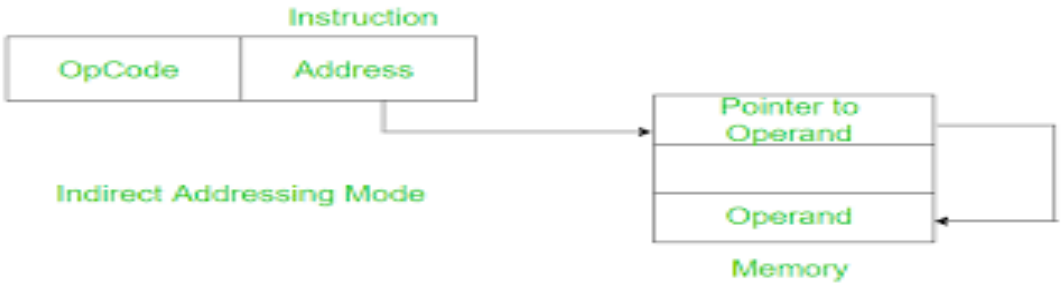
Here 1001 is the address where the operand is stored.

7.Indirect addressing mode

Used to implement pointers & implement parameters

Here, the address field in the instruction contains the memory location or register where the effective address of the operand is present

Two memory access is required



8.Relative addressing mode

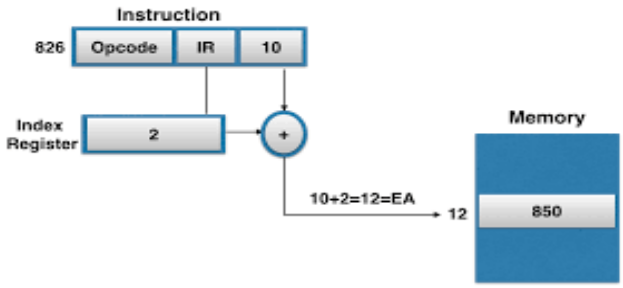
This mode is another version of the displacement address mode. The program counter's content is added to the instruction's address part to obtain the effective address.

EA = A + (PC)
Here, EA: Effective address, PC: program counter.

9.Base register addressing mode

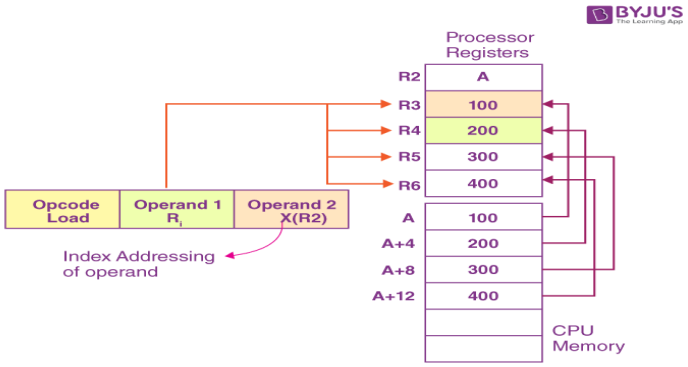
This addressing mode is used to relocate the program.

In our PC/laptop we have limited RAM & it can contain/handle limited number of programs. So to avoid this limitation we have switch in and switch out in between programs. So in this switch in/switch out process programs get different address



10.Indexed addressing mode

Indexed addressing means that the final address for the data is determined by adding an offset to a base address.
This memory address mode is ideal to store and access values stored in arrays. Arrays are often stored as a complete block in memory (A block of consecutive memory locations). The array has a base address which is the location of the first element, then an index is used that adds an offset to the base address in order to fetch the specified element within the array.



Chapter-3

Register File

A register file is an array of processor registers in a central processing unit (CPU). The instruction set architecture of a CPU will almost always define a set of registers which are used to stage data between memory and the functional units on the chip.

Program counter

Program counter stores the address of the next instruction to be executed which is going to perform.

So program counter contains the address, as it is containing the address the size of the Program counter depends upon size of address bits

Instruction Cycle

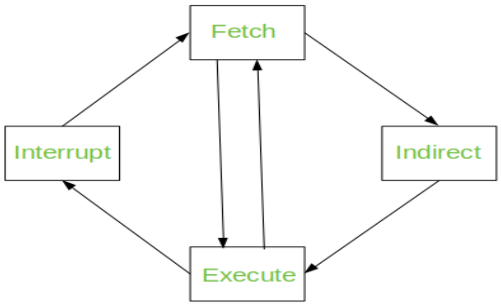
Memory address registers(MAR) : It is connected to the address lines of the system bus. It specifies the address in memory for a read or write operation.

Memory Buffer Register(MBR) : It is connected to the data lines of the system bus. It contains the value to be stored in memory or the last value read from the memory.
Program Counter(PC) : Holds the address of the next instruction to be fetched.
Instruction Register(IR) : Holds the last instruction fetched.
In computer organization, an instruction cycle, also known as a fetch-decode-execute cycle, is the basic operation performed by a central processing unit (CPU) to execute an instruction. The instruction cycle consists of several steps, each of which performs a specific function in the execution of the instruction. The major steps in the instruction cycle are:

1. Fetch: In the fetch cycle, the CPU retrieves the instruction from memory. The instruction is typically stored at the address specified by the program counter (PC). The PC is then incremented to point to the next instruction in memory.
2. Decode: In the decode cycle, the CPU interprets the instruction and determines what operation needs to be performed. This involves identifying the opcode and any operands that are needed to execute the instruction.
3. Execute: In the execute cycle, the CPU performs the operation specified by the instruction. This may involve reading or writing data from or to memory, performing arithmetic or logic operations on data, or manipulating the control flow of the program.
4. There are also some additional steps that may be performed during the instruction cycle, depending on the CPU architecture and instruction set:
5. Fetch operands: In some CPUs, the operands needed for an instruction are fetched during a separate cycle before the execute cycle. This is called the fetch operands cycle.
6. Store results: In some CPUs, the results of an instruction are stored during a separate cycle after the execute cycle. This is called the store results cycle.
7. Interrupt handling: In some CPUs, interrupt handling may occur during any cycle of the instruction cycle. An interrupt is a signal that the CPU receives from an external device or software that requires immediate attention. When an interrupt occurs, the CPU suspends the current instruction and executes an interrupt handler to service the interrupt.

These cycles are the basic building blocks of the CPU's operation and are performed for every instruction executed by the CPU. By optimizing these cycles, CPU designers can improve the performance and efficiency of the CPU, allowing it to execute instructions faster and more efficiently.

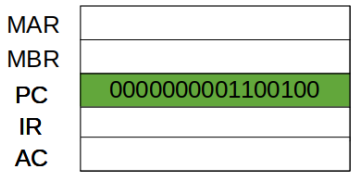
The Instruction Cycle –
Each phase of Instruction Cycle can be decomposed into a sequence of elementary micro-operations. In the above examples, there is one sequence each for the *Fetch, Indirect, Execute and Interrupt Cycles*.



The Instruction Cycle

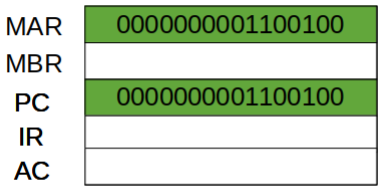
The *Indirect Cycle* is always followed by the *Execute Cycle*. The *Interrupt Cycle* is always followed by the *Fetch Cycle*. For both fetch and execute cycles, the next cycle depends on the state of the system.

- **The Fetch Cycle –**
  At the beginning of the fetch cycle, the address of the next instruction to be executed is in the *Program Counter*(PC).



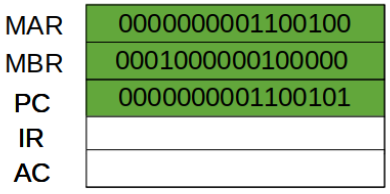
BEGINNING

- Step 1: The address in the program counter is moved to the memory address register(MAR), as this is the only register which is connected to address lines of the system bus.



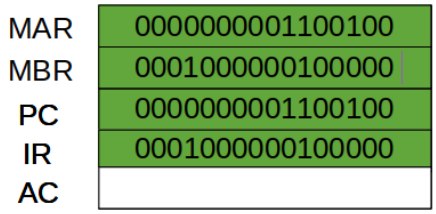
FIRST STEP

- Step 2: The address in MAR is placed on the address bus, now the control unit issues a READ command on the control bus, and the result appears on the data bus and is then copied into the memory buffer register(MBR). Program counter is incremented by one, to get ready for the next instruction. (These two action can be performed simultaneously to save time)



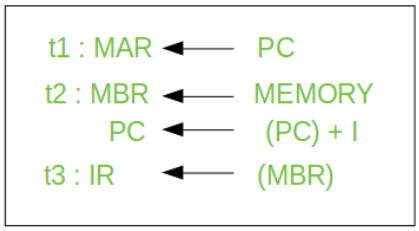
SECOND STEP

- Step 3: The content of the MBR is moved to the instruction register(IR).



FIRST STEP

- Thus, a simple *Fetch Cycle* consist of three steps and four micro-operation. Symbolically, we can write these sequence of events as follows:-



Hardware control

It is a control unit that uses a fixed set of logic gates & circuits to execute the instruction

The control signals for each instruction are hardwared into control unit, so the control unit has a dedicated circuit for each possible instructions.

Hardwared control units are complex & fast but they can be inflexible & difficult to modify

The control hardware can be viewed as a state machine that changes from one state to another in every clock cycle, depending on the contents of the instruction register, condition codes & external I/P.

The o/p of the state machine are the control signals.

RISC architecture based on hardwared control unit

Microprogram control

It is a control unit that uses a microcode to execute instruction. The microcode is a set of instructions that can be modified/updated allowing for greater flexibility & ease of modification

The control signals for each instruction are generated by a microprogram that is stored in memory rather than being hardwared into the control unit

Microprogram control unit are slower than the hardwared control unit because they require an extra step of decoding the microcode to generate computer signals but they are more flexible &,easier to modify.

They are commonly used in modern CPUs because they allow for easier implementation of complex instruction sets & better support for instruction set extension

Hardware control vs microprogram control

Parameter	Hardware control	microprogram control
implementation	Fixed set of logic gates & circuits	Micro code stored in memory
Flexibility	Less flexible, difficult to modify	More flexible, easy to modify
Instruction set	Supports limited instruction	Supports complex instruction
Complexity of design	Complex design, hard to implement	Simple design & easy to implement
Speed	Faster operation	Slower execution due to microcode
Debugging & testing	Difficult to debug & test	Easy to debug & test
Size & cost	Large size, high cost	Small size, low cost
Maintenance & upgradability	Difficult to upgrade & maintain	Easy to upgrade & maintain

Chapter-4

Memory characteristics

1. **Capacity:** It is the global volume of information the memory can store. As we move from top to bottom in the Hierarchy, the capacity increases.
2. **Access Time:** It is the time interval between the read/write request and the availability of the data. As we move from top to bottom in the Hierarchy, the access time increases.
3. **Performance:** Earlier when the computer system was designed without a Memory Hierarchy design, the speed gap increased between the CPU registers and Main Memory due to a large difference in access time. This results in lower performance of the system and thus, enhancement was required. This enhancement was made in the form of Memory Hierarchy Design because of which the performance of the system increases. One of the most significant ways to increase system performance is minimizing how far down the memory hierarchy one has to go to manipulate data.
4. **Cost Per Bit:** As we move from bottom to top in the Hierarchy, the cost per bit increases i.e. Internal Memory is costlier than External Memory.

Reliability – It is related to the lifetime of the device. Measured as Mean Time Between Failure (MTBF), in the units of days/years. Ex: Think of how frequently you replace your Hard disk while the CPU is still usable.

Memory hierarchy

In the Computer System Design, Memory Hierarchy is an enhancement to organize the memory such that it can minimize the access time. The Memory Hierarchy was developed based on a program behaviour known as locality of references. The figure below clearly demonstrates the different levels of the memory hierarchy.

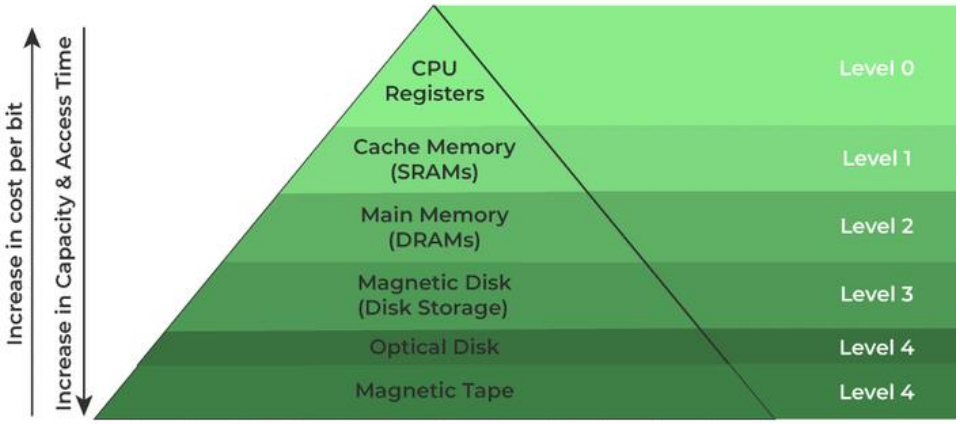
Why Memory Hierarchy is Required in the System?

Memory Hierarchy is one of the most required things in Computer Memory as it helps in optimizing the memory available in the computer. There are multiple levels present in the memory, each one having a different size, different cost, etc. Some types of memory like cache, and main memory are faster as compared to other types of memory but they are having a little less size and are also costly whereas some memory has a little higher storage value, but they are a little slower. Accessing of data is not similar in all types of memory, some have faster access whereas some have slower access.

Types of Memory Hierarchy

This Memory Hierarchy Design is divided into 2 main types:

- **External Memory or Secondary Memory:** Comprising of Magnetic Disk, Optical Disk, and Magnetic Tape i.e. peripheral storage devices which are accessible by the processor via an I/O Module.
- **Internal Memory or Primary Memory:** Comprising of Main Memory, Cache Memory & CPU registers. This is directly accessible by the processor.



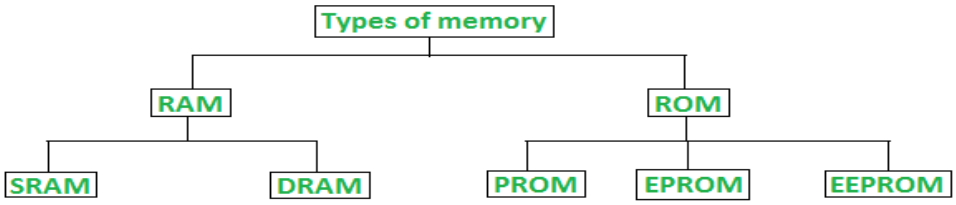
Memory Hierarchy Design

RAM & ROM ORGANISATION

Memory is the most essential element of a computing system because without it computer can't perform simple tasks. Both types of memory (RAM and ROM) are important for the computer, but they serve different purposes. RAM is used to store data that the computer is currently using, while ROM is used to store data that the computer needs to boot and operate. RAM is faster than ROM, as the data stored in it can be accessed and modified in any order, while data stored in ROM can only be read.

Computer memory is of two basic types:

1. Primary memory (RAM and ROM)
2. Secondary memory (Hard Drive, CD, etc).



Classification of computer memory

Random Access Memory (RAM)

- It is also called read-write *memory* or the *main memory* or the *primary memory*.
- The programs and data that the CPU requires during the execution of a program are stored in this memory.
- It is a volatile memory as the data is lost when the power is turned off.

Types of Random Access Memory (RAM)

- Static RAM (SRAM)
- Dynamic RAM (DRAM)

1. Static RAM: SRAM stands for Static Random Access Memory. It is a type of semiconductor which is widely used in computing devices and microprocessors.
2. Dynamic RAM: DRAM stands for Dynamic Random Access Memory. It is made of Capacitors and has smaller data life span than Static RAM.

DRAM	SRAM
1. Constructed of tiny capacitors that leak electricity.	1.Constructed of circuits similar to D flip-flops.
2.Requires a recharge every few milliseconds to maintain its data.	2.Holds its contents as long as power is available.
3.Inexpensive.	3.Expensive.
4. Slower than SRAM.	4. Faster than DRAM.
5. Can store many bits per chip.	5. Can not store many bits per chip.
6. Uses less power.	6.Uses more power.
7.Generates less heat.	7.Generates more heat.
8. Used for main memory.	8. Used for cache.

Difference between SRAM and DRAM

Advantages of Random Access Memory (RAM)

- **Speed:** RAM is much faster than other types of storage, such as a hard drive or solid-state drive, which means that the computer can access the data stored in RAM more quickly.
- **Flexibility:** RAM is volatile memory, which means that the data stored in it can be easily modified or deleted. This makes it ideal for storing data that the computer is currently using or processing.
- **Capacity:** The capacity of RAM can be easily upgraded, which allows the computer to store more data in memory and thus improve performance.
- **Power Management:** RAM consumes less power compared to hard drives, and solid-state drives, which makes it an ideal memory for portable devices.

Disadvantages of Random Access Memory (RAM)

- **Volatility:** RAM is volatile memory, which means that the data stored in it is lost when the power is turned off. This can be a problem for important data that needs to be preserved, such as unsaved work or files that have not been backed up.
- **Capacity:** The capacity of RAM is limited, and although it can be upgraded, it may still not be sufficient for certain applications or tasks that require a lot of memory.

- **Cost:** RAM can be relatively expensive compared to other types of memory, such as hard drives or solid-state drives, which can make upgrading the memory of a computer or device more costly.

Read-Only Memory (ROM)

- Stores crucial information essential to operate the system, like the program essential to boot the computer.
- It is non-volatile.
- Always retains its data.
- Used in embedded systems or where the programming needs no change.
- Used in calculators and peripheral devices.
- ROM is further classified into four types- M*ROM*, *PROM*, *EPROM*, and *EEPROM*.

Types of Read-Only Memory (ROM)

1. PROM (Programmable Read-Only Memory)
2. EPROM (Erasable Programmable Read Only Memory)
3. EEPROM (Electrically Erasable Programmable Read Only Memory)
4. MROM (Mask Read Only Memory)

1. PROM (Programmable read-only memory): It can be programmed by the user. Once programmed, the data and instructions in it cannot be changed.
2. EPROM (Erasable Programmable read-only memory): It can be reprogrammed. To erase data from it, expose it to ultraviolet light. To reprogram it, erase all the previous data.
3. EEPROM (Electrically erasable programmable read-only memory): The data can be erased by applying an electric field, with no need for ultraviolet light. We can erase only portions of the chip.
4. MROM(Mask ROM): Mask ROM is a kind of read-only memory, that is masked off at the time of production. Like other types of ROM, mask ROM cannot enable the user to change the data stored in it. If it can, the process would be difficult or slow.

Advantages of Read Only Memory (ROM)

- **Non-volatility:** ROM is non-volatile memory, which means that the data stored in it is retained even when the power is turned off. This makes it ideal for storing data that does not need to be modified, such as the BIOS or firmware for other hardware devices.
- **Reliability:** Because the data stored in ROM is not easily modified, it is less prone to corruption or errors than other types of memory.
- **Power Management:** ROM consumes less power compared to other types of memory, which makes it an ideal memory for portable devices.

Disadvantages of Read Only Memory (ROM)

- **Limited Flexibility:** ROM is read-only memory, which means that the data stored in it cannot be modified. This can be a problem for applications or firmware that need to be updated or modified.
- **Limited Capacity:** The capacity of ROM is typically limited, and upgrading it can be difficult or expensive.

- **Cost:** ROM can be relatively expensive compared to other types of memory, such as hard drives or solid-state drives, which can make upgrading the memory of a computer or device more costly.

Difference between RAM and ROM

RAM	ROM
1. Temporary Storage.	1. Permanent storage.
2. Store data in MBs.	2. Store data in GBs.
3. Volatile.	3. Non-volatile.
4.Used in normal operations.	4. Used for startup process of computer.
5. Writing data is faster.	5. Writing data is slower.

Difference between RAM and ROM

Virtual Memory

Virtual memory is a memory management technique where secondary memory can be used as if it were a part of the main memory. Virtual memory is a common technique used in a computer's operating system (OS).

Virtual memory uses both hardware and software to enable a computer to compensate for physical memory shortages, temporarily transferring data from random access memory (RAM) to disk storage. Mapping chunks of memory to disk files enables a computer to treat secondary memory as though it were main memory.

Today, most personal computers (PCs) come with at least 8 GB (gigabytes) of RAM. But, sometimes, this is not enough to run several programs at one time. This is where virtual memory comes in. Virtual memory frees up RAM by swapping data that has not been used recently over to a storage device, such as a hard drive or solid-state drive (SSD).

Virtual memory is important for improving system performance, multitasking and using large programs. However, users should not overly rely on virtual memory, since it is considerably slower than RAM. If the OS has to swap data between virtual memory and RAM too often, the computer will begin to slow down -- this is called thrashing.

Virtual memory was developed at a time when physical memory -- also referenced as RAM -- was expensive. Computers have a finite amount of RAM, so memory will eventually run out when multiple programs run at the same time. A system using virtual memory uses a section

of the hard drive to emulate RAM. With virtual memory, a system can load larger or multiple programs running at the same time, enabling each one to operate as if it has more space, without having to purchase more RAM.

How virtual memory works

Virtual memory uses both hardware and software to operate. When an application is in use, data from that program is stored in a physical address using RAM. A memory management unit (MMU) maps the address to RAM and automatically translates addresses. The MMU can, for example, map a logical address space to a corresponding physical address.

If, at any point, the RAM space is needed for something more urgent, data can be swapped out of RAM and into virtual memory. The computer's memory manager is in charge of keeping track of the shifts between physical and virtual memory. If that data is needed again, the computer's MMU will use a context switch to resume execution.

While copying virtual memory into physical memory, the OS divides memory with a fixed number of addresses into either pagefiles or swap files. Each page is stored on a disk, and when the page is needed, the OS copies it from the disk to main memory and translates the virtual addresses into real addresses.

However, the process of swapping virtual memory to physical is rather slow. This means using virtual memory generally causes a noticeable reduction in performance. Because of swapping, computers with more RAM are considered to have better performance.

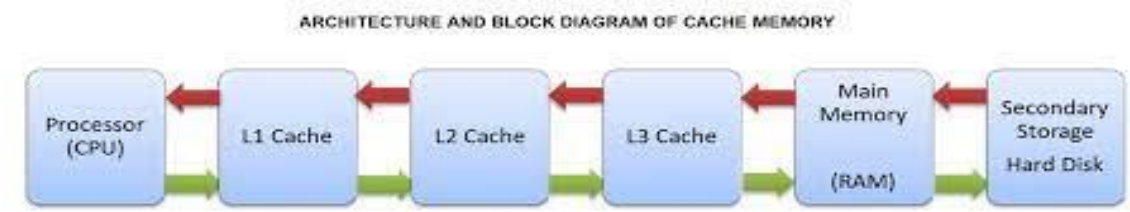
What are the benefits of using virtual memory?

The advantages to using virtual memory include:

- It can handle twice as many addresses as main memory.
- It enables more applications to be used at once.
- It frees applications from managing shared memory and saves users from having to add memory modules when RAM space runs out.
- It has increased speed when only a segment of a program is needed for execution.
- It has increased security because of memory isolation.
- It enables multiple larger applications to run simultaneously.
- Allocating memory is relatively inexpensive.
- It does not need external fragmentation.
- CPU use is effective for managing logical partition workloads.
- Data can be moved automatically.
- Pages in the original process can be shared during a fork system call operation that creates a copy of itself.

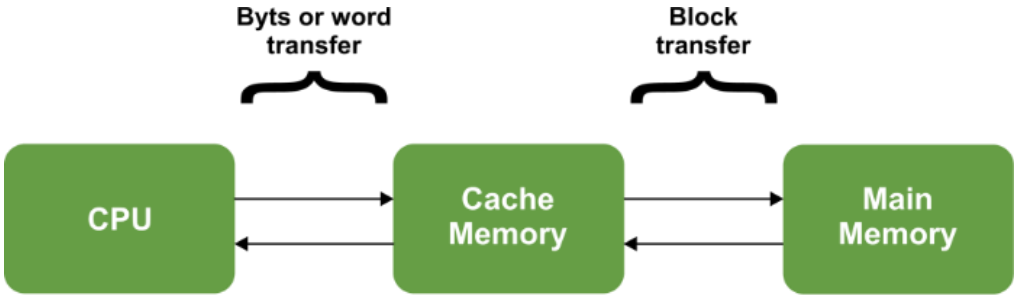
In addition to these benefits, in a virtualized computing environment, administrators can use virtual memory management techniques to allocate additional memory to a virtual machine (VM) that has run out of resources. Such virtualization management tactics can improve VM performance and management flexibility.

Cache memory



Cache Memory is a special very high-speed memory. The data or contents of the main memory that are used frequently by CPU are stored in the cache memory so that the processor can easily access that data in a shorter time. Whenever the CPU needs to access memory, it first checks the cache memory. If the data is not found in cache memory, then the CPU moves into the main memory.

Cache memory is placed between the CPU and the main memory. The block diagram for a cache memory can be represented as:



Cache Memory

The cache is the fastest component in the memory hierarchy and approaches the speed of CPU components.

Cache memory is organised as distinct set of blocks where each set contains a small fixed number of blocks.

Levels of Memory

Level 1 or Register: It is a type of memory in which data is stored and accepted that are immediately stored in the CPU. The most commonly used register is Accumulator, Program counter, Address Register, etc.

Level 2 or Cache memory: It is the fastest memory that has faster access time where data is temporarily stored for faster access.

Level 3 or Main Memory: It is the memory on which the computer works currently. It is small in size and once power is off data no longer stays in this memory.

Level 4 or Secondary Memory: It is external memory that is not as fast as the main memory but data stays permanently in this memory.

Cache Performance

When the processor needs to read or write a location in the main memory, it first checks for a corresponding entry in the cache.

If the processor finds that the memory location is in the cache, a Cache Hit has occurred and data is read from the cache.

If the processor does not find the memory location in the cache, a **cache miss** has occurred. For a cache miss, the cache allocates a new entry and copies in data from the main memory, then the request is fulfilled from the contents of the cache.

The performance of cache memory is frequently measured in terms of a quantity called **Hit ratio.**

Hit Ratio(H) = hit / (hit + miss) = no. of hits/total accesses

Miss Ratio = miss / (hit + miss) = no. of miss/total accesses = 1 - hit ratio(H)

We can improve Cache performance using higher cache block size, and higher associativity, reduce miss rate, reduce miss penalty, and reduce the time to hit in the cache.

Application of Cache Memory

Here are some of the applications of Cache Memory.

1. **Primary Cache:** A primary cache is always located on the processor chip. This cache is small and its access time is comparable to that of processor registers.
2. **Secondary Cache:** Secondary cache is placed between the primary cache and the rest of the memory. It is referred to as the level 2 (L2) cache. Often, the Level 2 cache is also housed on the processor chip.
3. **Spatial Locality of Reference:** Spatial Locality of Reference says that there is a chance that the element will be present in close proximity to the reference point and next time if again searched then more close proximity to the point of reference.
4. **Temporal Locality of Reference:** Temporal Locality of Reference uses the Least recently used algorithm will be used. Whenever there is page fault occurs within a word will not only load the word in the main memory but the complete page fault will be loaded because the spatial locality of reference rule says that if you are referring to any word next word will be referred to in its register that's why we load complete page table so the complete block will be loaded.

Advantages of Cache Memory

- Cache Memory is faster in comparison to main memory and secondary memory.
- Programs stored by Cache Memory can be executed in less time.
- The data access time of Cache Memory is less than that of the main memory.
- Cache Memory stored data and instructions that are regularly used by the CPU, therefore it increases the performance of the CPU.

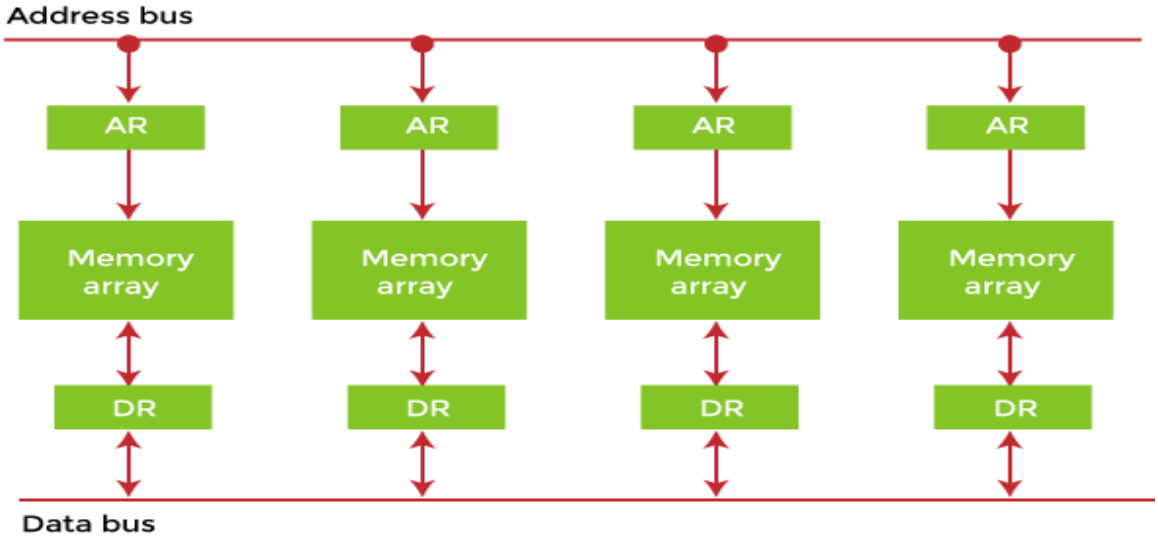
Disadvantages of Cache Memory

- Cache Memory is costlier than primary memory and secondary memory.
- Data is stored on a temporary basis in Cache Memory.

- Whenever the system is turned off, data and instructions stored in cache memory get destroyed.
- The high cost of cache memory increases the price of the Computer System.

Interleaved Memory

Interleaved memory is designed to compensate for the relatively slow speed of dynamic random-access memory (DRAM) or core memory by spreading memory addresses evenly across memory banks. In this way, contiguous memory reads and writes use each memory bank, resulting in higher memory throughput due to reduced waiting for memory banks to become ready for the operations.
It is different from multi-channel memory architectures, primarily as interleaved memory does not add more channels between the main memory and the memory controller. However, channel interleaving is also possible, for example, in ***Freescale*** i.MX6 processors, which allow interleaving to be done between two channels. With interleaved memory, memory addresses are allocated to each memory bank.



Example of Interleaved Memory

It is an abstraction technique that divides memory into many modules such that successive words in the address space are placed in different modules.

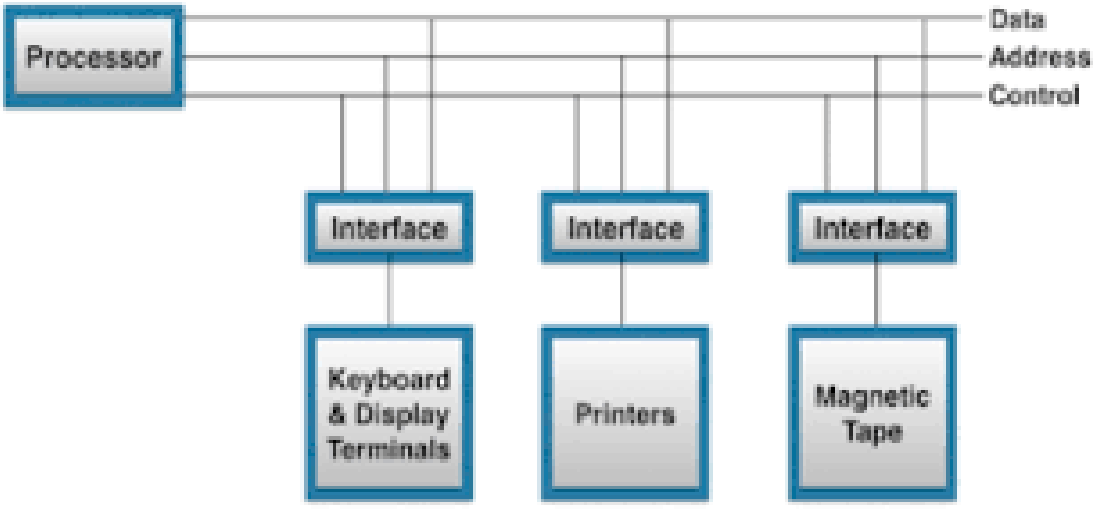
Suppose we have 4 memory banks, each containing 256 bytes, and then the Block Oriented scheme (no interleaving) will assign virtual addresses 0 to 255 to the first bank and 256 to 511 to the second bank. But in Interleaved memory, virtual address 0 will be with the first bank, 1 with the second memory bank, 2 with the third bank and 3 with the fourth, and then 4 with the first memory bank again.

Chapter – 5

Input-Output Interface

When we type something from our keyboard, the input data is transferred to the computer's CPU, and the screen displays the output data. But how does our computer's CPU or processors share information with the external Input-Output devices? Well, we can achieve this with the input-output Interface.
The Input-output interface allows transferring information between external input/output devices (i.e., peripherals) and processors. This article will discuss the Input-Output Interface, its structure, and function.

The input-output interface allows transferring information between external input/output devices (i.e., peripherals) and processors. A peripheral device provides input/output for the computer, and it is also known as an Input-Output device.
This input-output interface in a computer system exists in a particular hardware component between the system's bus and peripherals. This component is known as the "interface unit".
The below figure shows a typical input-output interface between the processor and different peripherals:



1. In the above figure, we see that every peripheral device has an interface unit associated with it.
2. For example, A mouse or keyboard that provides input to the computer is called an input device while a printer or monitor that provides output to the computer is called an output device.

Why Input-Output Interface

We require the input-output Interface because many differences exist between each peripheral and the central computer while transferring data. Some significant differences between the peripheral and CPU are:

- The nature of the CPU is electronic, and that of the peripheral device is electro-mechanical and electromagnetic. So, we can see many differences in the mode of operation of both CPU and peripheral devices.

- We have a synchronization mechanism because the data transfer rate is slower in peripheral devices than CPU.
- In peripheral devices, data code and formats differ from the format in the CPU.
- The operating modes of peripheral devices are different, and each can be controlled not to disturb the operation of any other peripheral devices connected to the CPU.

Functions of Input-Output Interface

The primary functions of the input-output Interface are listed below:

- It can synchronize the operating speed of the CPU to peripherals.
- It selects the peripheral appropriate for the interpretation of the input-output devices.
- It provides signals like timing and control signals.
- In this, data buffering may be possible through the data bus.
- It has various error detectors.
- It can convert serial data into parallel and vice versa.
- It can convert digital data into analogue signals and vice versa.

I/O Bus and Interface Modules

I/O buses are the routes used for peripheral devices to interact with the processor. A typical connection of the I/O buses to I/O devices is displayed in the figure.

The I/O buses include control lines, address lines, and data lines. In any general computer, the printer, keyboard, magnetic disk, and display terminal are commonly connected. Every peripheral unit has an interface associated with it. Every interface decodes the address and control received from the I/O bus.

It can describe the control and address received from the computer peripheral and supports signals for the computer peripheral controller. It can also conduct the transfer of information or data between peripheral and processor and can also integrate the data flow.

The I/O buses are linked to all the peripheral interfaces from the computer processor. The processor locates a device address on the address line for interaction with a specific device. Every interface contains an address decoder that monitors the address lines, attached to the I/O bus.

When the interface recognizes the address, it activates the direction between the bus and the device that it controls. The interface will disable the peripherals whose address is not equivalent to the address in the bus.

Modes of Transfer in COA

In computer organization and architecture, ***the "modes of transfer" refer to the different methods used for transferring data between components of a computer system***

In computer organization and architecture (COA), there are several modes of transfer, but three of the most common are:

1. Programmed I/O transfer

2. Interrupt-driven transfer

3. Direct Memory Access (DMA)

1. Programmed I/O transfer

Programmed I/O transfer is a data transfer mode in computer organization and architecture. In this mode of transfer***, data is transferred between an I/O device and the computer's memory under the control of a program***. The central processing unit (CPU) actively manages the transfer, sending commands to the I/O device to initiate the transfer and monitor its progress.

In a programmed I/O transfer, the CPU sends a command to the I/O device to initiate the transfer and then waits for the transfer to complete. During the transfer, the CPU cannot perform other tasks as it is busy managing the transfer.

Programmed I/O transfer provides low-level control over the transfer, as the CPU can issue commands to the I/O device to control the transfer. This can be useful for certain applications where fine-grained control over the transfer is required. However, it also means that the CPU is tied up during the transfer, potentially slowing down other tasks that the computer performs.

Programmed I/O transfer is typically slower than other transfer modes, such as direct memory access (DMA). However, it can still be useful in certain situations, such as when low-level control over the transfer is required or when the amount of data is small.

Programmed I/O transfer works as follows:

1. The CPU sends a command to the I/O device to initiate the transfer. This command may specify the transfer direction (i.e., from memory to the I/O device or from the I/O device to memory), the starting address in memory where the data is to be stored or retrieved, and the number of bytes to be transferred.
2. The I/O device receives the command and begins the transfer. It transfers the data to or from the specified memory location, one byte at a time.
3. The CPU repeatedly polls the status register of the I/O device to monitor the progress of the transfer. The status register contains information about the current state of the I/O device, including whether the transfer is complete.
4. When the transfer is complete, the I/O device sets a flag in the status register to indicate that the transfer is done. The CPU reads the status register, and when it sees the flag, it knows that the transfer is complete.
5. The CPU can resume other tasks after the transfer is finished. In programmed I/O transfer, the CPU manages the transfer and monitors its progress. This allows for low-level control over the transfer but also means that the CPU is tied up during the transfer and cannot perform other tasks. Programmed I/O transfer is typically slower than other transfer modes, such as direct memory access (DMA), as the CPU must repeatedly poll the status register to monitor the transfer.

Here are a few examples of how programmed I/O transfer might be used in a computer system:

1. Keyboard input: When a user types on the keyboard, the keyboard sends a signal to the CPU indicating that data is ready for transfer. The CPU uses programmed I/O transfer to receive the data from the keyboard, one character at a time.

2. Serial communication: When a computer communicates with another device over a serial connection, the CPU uses programmed I/O transfer to send and receive data. The CPU sends a command to the serial communication device to initiate the transfer and then waits for the transfer to complete.

3. Disk I/O: When a computer reads data from a disk or writes data to a disk, the CPU uses programmed I/O transfer to transfer the data. The CPU sends a command to the disk controller to initiate the transfer and then waits for the transfer to complete.
4. Display output: When a computer displays data on a screen, the CPU uses programmed I/O transfer to send the data to the display adapter. The CPU sends a command to the display adapter to initiate the transfer and then waits for the transfer to complete.
These are just a few examples of how programmed I/O transfer might be used in a computer system. The specific details of the transfer will vary depending on the I/O device and the architecture of the computer system.

Advantages of Programmed I/O Transfer

1. Flexibility: Programmed I/O transfer allows for fine-grained control over the transfer. The CPU can issue commands to the I/O device to control the transfer, which can be useful for certain applications requiring low-level control.
2. Compatibility: Programmed I/O transfer works with a wide range of I/O devices, making it a versatile transfer mode.
3. Debugging: As the CPU actively manages the transfer, diagnosing and fixing problems that may occur during the transfer is easier.
4. Simple Implementation: Programmed I/O transfer can be implemented using simple programming constructs, such as loops and status register polling.
5. Cost Effective: Programmed I/O transfer does not require specialized hardware, making it a cost-effective solution for transferring data in certain situations.
These are just a few advantages of programmed I/O transfer. The advantages will depend on the particular system used and the application's requirements.

Disadvantages of Programmed I/O transfer:

1. Slow performance: As the CPU must manage the transfer and poll the status register, programmed I/O transfer is slower than other transfer modes, such as **direct memory access (DMA).**
2. CPU utilization: During the transfer, the CPU is tied up, which can negatively impact the system's overall performance.
3. Complexity: Implementing programmed I/O transfer can be more complex than other modes of transfer, such as DMA, which can make it more difficult to diagnose and fix problems that may occur during the transfer.
4. Lack of scalability: Programmed I/O transfer could be better suited for high-speed transfers, as the CPU cannot manage multiple transfers simultaneously.
5. Limited to low-level control: While the low-level control offered by programmed I/O transfer can be an advantage in some cases, it can also be a disadvantage, as it does not allow for higher-level abstractions that may be desirable for certain applications.

2. Interrupt-driven I/O Transfer

The interrupt-driven transfer is a data transfer mode in computer organization and architecture. In this transfer mode, ***the CPU is notified of incoming data through an interrupt request.*** An interrupt is a signal sent to the CPU indicating that an I/O device has data ready for transfer.

When the CPU receives an interrupt, it suspends its current task and handles it. The CPU communicates with the I/O device to initiate the data transfer and then waits for the transfer to complete. Once the transfer is complete, the CPU returns to its previous task.
The interrupt-driven transfer balances the control offered by programmed I/O transfer and the speed offered by direct memory access (DMA). The CPU controls the transfer, but the transfer occurs in the background, freeing up the CPU to perform other tasks. This transfer mode is commonly used in systems requiring high-speed data transfer and low-level control over the transfer. Examples include keyboard input, serial communication, disk I/O, and display output.

Interrupt-driven transfer works as follows:

1. The CPU sends a command to the I/O device to initiate the transfer. Depending on the specific device, this command could be as simple as a write to a control register or more complex.
2. The I/O device responds to the command by making the data available for transfer.
3. The CPU polls the status register of the I/O device to determine when the data is ready for transfer.
4. When the data is ready, the CPU transfers the data from the I/O device to memory or from memory to the I/O device, one word at a time.
5. The CPU repeats the process until all data has been transferred.

In programmed I/O transfer, the CPU actively manages the transfer, providing fine-grained control. However, this also means that the CPU is tied up during the transfer, which can negatively impact performance.

Here are a few examples of Interrupt-driven transfer

The interrupt-driven transfer is a common data transfer mode in computer organization and architecture used in various applications. Here are a few examples:

1. Keyboard Input: When a user presses a key on the keyboard, an interrupt is generated to notify the CPU that data is ready for transfer. The CPU handles the interrupt, reads the data from the keyboard, and stores it in memory.

2. Serial Communication: When a serial device, such as a modem or a serial-to-USB converter, has data ready to transfer, it generates an interrupt to notify the CPU. The CPU handles the interrupt, reads the data from the device, and stores it in memory.

3. Disk I/O: When a disk drive has data to transfer, it generates an interrupt to notify the CPU. The CPU handles the interrupt, reads the data from the disk drive, and stores it in memory.

4. Display Output: When the graphics card has data to transfer to the display, it generates an interrupt to notify the CPU. The CPU handles the interrupt, reads the data from the graphics card, and transfers it to the display.

These are just a few examples of the many applications that use interrupt-driven transfer. The specific applications will depend on the particular system being used and the application's requirements.

Here are a few advantages of Interrupt-driven transfer

The interrupt-driven transfer has several advantages over other modes of data transfer, including:

1. Improved System Performance: Interrupt-driven transfer allows the CPU to perform other tasks while waiting for the I/O transfer to complete, which can result in improved system performance.
2. Real-time Processing: Interrupt-driven transfer can be used for real-time processing, as the CPU is notified immediately when data is ready for transfer. This can be important for applications that require the timely processing of data.
3. Lower CPU Overhead: Unlike programmed I/O transfer, interrupt-driven transfer requires less CPU overhead, as the CPU is not constantly polling the status register of the I/O device.
4. Dynamic Load Balancing: Interrupt-driven transfer allows the CPU to dynamically balance the load between processing tasks and I/O tasks, as it can handle interrupts as they occur.
5. Improved Responsiveness: Interrupt-driven transfer improves responsiveness, as the CPU can quickly respond to incoming data. This is especially important in interactive applications, such as games or multimedia applications.
Overall, the interrupt-driven transfer provides a balance between the control offered by programmed I/O transfer and the speed offered by direct memory access (DMA), making it a versatile mode of data transfer in computer organization and
architecture.

Here are a few disadvantages of Interrupt-driven transfer

Like any mode of data transfer, the interrupt-driven transfer has its disadvantages as well, including:
1. Complexity: Interrupt-driven transfer can be more complex to implement than other transfer modes, as it requires interrupting handlers and managing interrupt priority levels.
2. Latency: Interrupt-driven transfer can introduce latency into the system, as the CPU must handle the interrupt and perform the data transfer. This can be especially problematic in real-time applications that require fast response times.
3. Overhead: Interrupt-driven transfer introduces additional overhead into the system, as the CPU must handle the interrupt and perform the data transfer. This can be especially problematic in systems with limited processing power.
4. Unpredictable Interrupt Latency: The latency associated with interrupt-driven transfer can be unpredictable, as it can be affected by the number of other interrupts being processed by the CPU at the time.
5. Error Handling: Error handling can be more complex in interrupt-driven transfer, as errors can occur at any point in the transfer.
Overall, interrupt-driven transfer is a powerful mode of data transfer in computer organization and architecture, but it requires careful consideration of the
trade-offs between performance, complexity, and overhead.

Direct Memory Access (DMA)

Direct Memory Access (DMA) is a data transfer method in computer organization and architecture that allows an I/O device to transfer data directly to or from memory without the involvement of the CPU. DMA allows I/O devices to transfer data to or from memory at high speeds without putting additional strain on the CPU.
In DMA, a **DMA controller** is used to manage data transfer between the I/O device and memory. The DMA controller is responsible for setting up the transfer, starting it, and

monitoring it to ensure it is completed successfully. The CPU is not involved in the actual data transfer but is notified when it is complete.
DMA is often used for high-bandwidth I/O devices, such as disk drives, network interfaces, and graphics cards, where the CPU is not fast enough to manage the data transfer in real time. DMA allows these devices to transfer data to or from memory at high speeds without putting additional strain on the CPU.

Direct Memory Access (DMA) works as follows:

1. **Initialization**: The CPU initializes the DMA transfer by setting up the source and destination addresses, the size of the transfer, and any other relevant parameters.
2. **Request**: The I/O device requests a DMA transfer by asserting a request signal to the DMA controller.
3. **Grant**: The DMA controller grants the request and starts the transfer by asserting a grant signal to the I/O device.
4. **Data Transfer**: The I/O device transfers the data directly to or from memory without involving the CPU. The DMA controller manages the transfer and provides necessary addresses and control signals.
5. **Completion**: When the transfer is complete, the I/O device asserts a completion signal to the DMA controller. The DMA controller then informs the CPU that the transfer is complete.

The CPU can perform other tasks during the transfer as the DMA controller handles the data transfer. This allows the CPU to improve its overall performance and responsiveness.
Overall, DMA provides a high-speed data transfer method that allows I/O devices to transfer data to or from memory without putting additional strain on the CPU. This improves system performance and responsiveness, especially for high-bandwidth I/O devices.

Some examples of Direct Memory Access (DMA) in action:

1. Disk Drives: DMA is often used for disk drives, where high-speed data transfer is critical for good performance. DMA allows disk drives to transfer data to or from memory at high speeds without putting additional strain on the CPU.
2. Network Interfaces: Network interfaces also use DMA to transfer data to or from memory at high speeds. This allows the CPU to handle other tasks while the network interface performs the data transfer.
3. Graphics Cards: Graphics cards use DMA to transfer large amounts of video data to or from memory. This allows the GPU to render images quickly without putting additional strain on the CPU.
4. Sound Cards: Sound cards use DMA to transfer audio data to or from memory, which allows the CPU to handle other tasks while the sound card is playing or recording audio.
5. Peripheral Devices: Many peripheral devices, such as printers, scanners, and USB devices, also use DMA to transfer data to or from memory. This allows the CPU to handle other tasks while the peripheral device performs the data transfer.
These are just a few examples of Direct Memory Access (DMA). DMA is widely used in computer systems to provide high-speed data transfer for I/O devices, which helps to improve system performance and responsiveness.

Advantages of Direct Memory Access (DMA):

1. Improved Performance: DMA allows I/O devices to transfer data directly to or from memory at high speeds without involving the CPU. This improves system performance and responsiveness, especially for high-bandwidth I/O devices.
2. CPU Offload: By handling the data transfer, DMA reduces the load on the CPU. This allows the CPU to perform other tasks more efficiently and improves system performance.
3. High Bandwidth: DMA provides a high-speed method of data transfer that is well suited to high-bandwidth I/O devices. This results in improved system performance and responsiveness for these devices.
4. Increased Efficiency: DMA allows I/O devices to transfer data directly to or from memory without CPU intervention. This results in increased efficiency and improved system performance.
5. Real-time Processing: DMA is often used for real-time processing applications, such as audio and video playback, where high-speed data transfer is critical for good performance.
Overall, DMA provides a high-speed data transfer method that allows I/O devices to transfer data to or from memory without putting additional strain on the CPU.
This results in improved system performance and responsiveness and makes DMA a key component of many computer systems.

Disadvantages of Direct Memory Access (DMA):

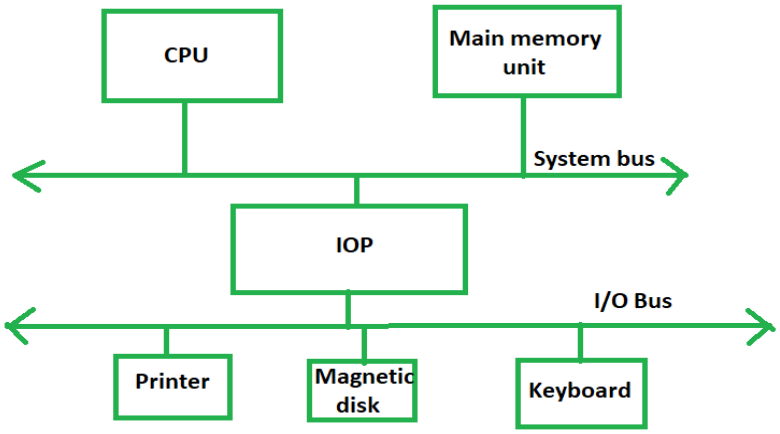
1. Complexity: DMA requires a significant amount of hardware and software support, which makes the overall system more complex and difficult to manage.
2. Resource Management: DMA requires careful management of system resources, such as memory and I/O device access, to ensure that data transfers do not interfere with other system operations.
3. Latency: In some cases, DMA may introduce latency into the system, as the CPU may have to wait for the DMA transfer to complete before it can perform other tasks.
4. Security Risks: DMA provides direct access to memory, which can introduce security risks if the system is not properly secured. For example, malware or other malicious software could use DMA to bypass security measures and access sensitive data.
5. Interrupt Latency: DMA can introduce additional interrupt latency into the system, as the CPU may have to handle additional interrupt requests from the DMA controller.
These are a few disadvantages of Direct Memory Access (DMA). While DMA provides many benefits for computer systems, it is important to consider these limitations and design systems accordingly to ensure that the advantages of DMA outweigh the disadvantages.

Input-Output Processor

The **DMA mode** of data transfer reduces the CPU's overhead in handling I/O operations. It also allows parallelism in CPU and I/O operations. Such parallelism is necessary to avoid the wastage of valuable CPU time while handling I/O devices whose speeds are much slower as compared to CPU. The concept of DMA operation can be extended to relieve the CPU further from getting involved with the execution of I/O operations. This gives rise to the development of special purpose processors called **Input-Output Processor (IOP) or IO channels**.

The Input-Output Processor (IOP) is just like a CPU that handles the details of I/O operations. It is more equipped with facilities than those available in a typical DMA controller. The IOP can fetch and execute its own instructions that are specifically designed to characterize I/O transfers. In addition to the I/O-related tasks, it can perform other processing tasks like arithmetic, logic, branching, and code translation. The main memory unit takes a pivotal role. It communicates with the processor by means of DMA.

The Input-Output Processor is a specialized processor which loads and stores data in memory along with the execution of I/O instructions. It acts as an interface between the system and devices. It involves a sequence of events to execute I/O operations and then store the results in memory.



Input-Output Processor

Features of an Input-Output Processor

- **Specialized Hardware:** An IOP is equipped with specialized hardware that is optimized for handling input/output operations. This hardware includes input/output ports, DMA controllers, and interrupt controllers.
- **DMA Capability:** An IOP has the capability to perform Direct Memory Access (DMA) operations. DMA allows data to be transferred directly between peripheral devices and memory without going through the CPU, thereby freeing up the CPU for other tasks.
- **Interrupt Handling:** An IOP can handle interrupts from peripheral devices and manage them independently of the CPU. This allows the CPU to focus on executing application programs while the IOP handles interrupts from peripheral devices.
- **Protocol Handling:** An IOP can handle communication protocols for different types of devices such as Ethernet, USB, and SCSI. This allows the IOP to interface with a wide range of devices without requiring additional software support from the CPU.
- **Buffering:** An IOP can buffer data between the CPU and peripheral devices. This allows the IOP to handle large amounts of data without overloading the CPU or the peripheral devices.
- **Command Processing:** An IOP can process commands from peripheral devices independently of the CPU. This allows the CPU to focus on executing application programs while the IOP handles peripheral device commands.

- **Parallel Processing:** An IOP can perform input/output operations in parallel with the CPU. This allows the system to handle multiple tasks simultaneously and improve overall system performance.

Applications of I/O Processors

- **Data Acquisition Systems:** I/O processors can be used in data acquisition systems to acquire and process data from various sensors and input devices. The I/O processor can handle high-speed data transfer and perform real-time processing of the acquired data.
- **Industrial Control Systems:** I/O processors can be used in industrial control systems to interface with various control devices and sensors. The I/O processor can provide precise timing and control signals, and can also perform local processing of the input data.
- **Multimedia Applications:** I/O processors can be used in multimedia applications to handle the input and output of multimedia data, such as audio and video. The I/O processor can perform real-time processing of multimedia data, including decoding, encoding, and compression.
- **Network Communication Systems:** I/O processors can be used in network communication systems to handle the input and output of data packets. The I/O processor can perform packet routing, filtering, and processing, and can also perform encryption and decryption of the data.
- **Storage Systems:** I/O processors can be used in storage systems to handle the input and output of data to and from storage devices. The I/O processor can handle high-speed data transfer and perform data caching and prefetching operations.

Advantages of Input-Output Processor

- The I/O devices can directly access the main memory without the intervention of the processor in I/O processor-based systems.
- It is used to address the problems that arise in the Direct memory access method.
- **Reduced Processor Workload:** With an I/O processor, the main processor doesn't have to deal with I/O operations, allowing it to focus on other tasks. This results in more efficient use of the processor's resources and can lead to faster overall system performance.
- **Improved Data Transfer Rates:** Since the I/O processor can access memory directly, data transfers between I/O devices and memory can be faster and more efficient than with other methods.
- **Increased System Reliability:** By offloading I/O tasks to a dedicated processor, the system can be made more fault-tolerant. For example, if an I/O operation fails, it won't affect other system processes.
- **Scalability:** I/O processor-based systems can be designed to scale easily, allowing for additional I/O processors to be added as needed. This can be particularly useful in large-scale data centres or other environments where the number of I/O devices is constantly changing.

- **Flexibility:** I/O processor-based systems can be designed to handle a wide range of I/O devices and interfaces, providing more flexibility in system design and allowing for better customization to meet specific requirements.

Disadvantages of Input-Output Processor

- **Cost:** I/O processors can add significant costs to a system due to the additional hardware and complexity required. This can be a barrier to adoption, especially for smaller systems.
- **Increased Complexity:** The addition of an I/O processor can increase the overall complexity of a system, making it more difficult to design, build, and maintain. This can also make it harder to diagnose and troubleshoot issues.
- **Limited Performance Gains:** While I/O processors can improve system performance by offloading I/O tasks from the main processor, the gains may not be significant in all cases. In some cases, the additional overhead of the I/O processor may actually slow down the system.
- **Synchronization Issues:** With multiple processors accessing the same memory, synchronization issues can arise, leading to potential data corruption or other errors.
- **Lack of Standardization:** There are many different I/O processor architectures and interfaces available, which can make it difficult to develop standardized software and hardware solutions. This can limit interoperability and make it harder for vendors to develop compatible products.

Chapter – 6

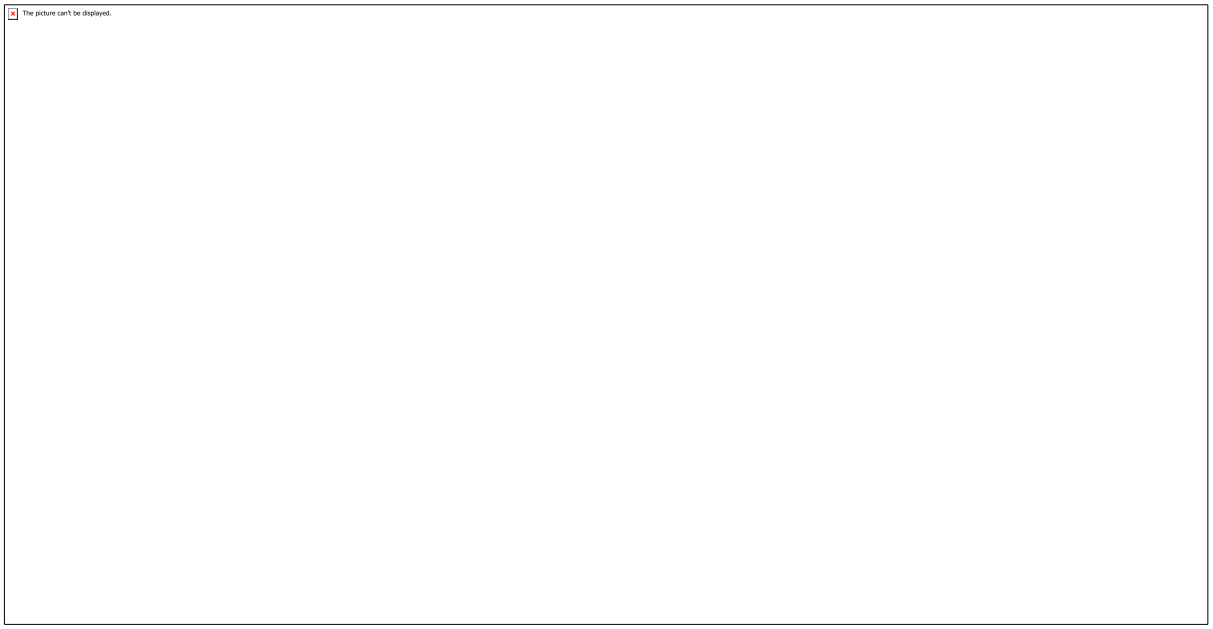
System Bus

- A bus is a set of electrical wires (lines) that connects the various hardware components of a computer system.
- It works as a communication pathway through which information flows from one hardware component to the other hardware component.
- Bus structures in computer plays important role in connecting the internal components of the computer. The bus in the computer is the *shared transmission medium*. This means multiple components or devices use the same bus structure to transmit the information signals to each other.

Components Of A System Bus-

The system bus consists of three major components-

- Data Bus
- Address Bus
- Control Bus



Address Lines:

- Used to carry the address to memory and IO.
- Unidirectional.
- Based on the width of an address bus we can determine the capacity of a main memory
- The width of the address bus determines the memory capacity of the system. The content of address lines is also used for addressing I/O ports. The higher-order bits determine the bus module, and the lower-ordered bits determine the address of memory locations or I/O ports.

- The content of the address lines of the bus determines the source or destination of the data present on the data bus. The number of address lines together is referred to as the address bus. The number of address lines in the address bus determines its *width*.
- Whenever the processor has to read a word from memory, it simply places the address of the corresponding word on the address line.

Data Lines

- Data lines coordinate in transferring the data among the system components. The data lines are collectively called data bus. A data bus may have 32 lines, 64 lines, 128 lines, or even more lines. The number of lines present in the data bus defines the *width* of the data bus.
- Each data line is able to transfer only one bit at a time. So the number of data lines in a data bus determines how many bits it can transfer at a time. The performance of the system also depends on the width of the data bus.
- Used to carry the binary data between the CPU, memory and IO.
- Bidirectional.
- Based on the width of a data bus we can determine the word length of a CPU.
- Based on the word length we can determine the performance of a CPU.

Control Lines:

- Used to carry the control signals and timing signals
- Control signals indicate the type of operation. The control signal consists of the *command* and *timing information*
- Timing Signals are used to synchronize the memory and IO operations with a CPU clock.
- Typical Control Lines may include Memory Read/Write, IO Read/Write, Bus Request/Grant, etc.

Basic parameters of Bus Design

1) Bus Types
2) Method of Arbitration
3) Timing
4) Bus Width
5) Data Transfer Type
6) Block Data Transfer

1) Bus Types

A) **Dedicated**

- A line is permanently assigned either to one function.
- An example of functional dedication is the use of separate dedicated address and data line.

B) **Multiplexed**

- Using the same lines for multiple purpose.

- Eg:- Address and data information may be transmitted over the same set of lines.
- At the beginning of the data transfer the address is placed on the bus and the address valid line is activated.
- The address is then remove from the same bus line is used for data transfer.

C) **Physical Dedication**

- The use of multiple buses, each of which connects to only a subset of modules.

2) Method of Arbitration

- Determining who can use the bus at a particular time.

A) **Centralized**

- A single hardware device called the bus controller or arbiter allocate time on the bus.
- The device may be a separate or a part of a processor.

B) **Distributed**

- There is no centralized controllers.
- Each module contains assess control logic and the modules act together.

3) Timing

A) **Synchronous Timing**

- Bus includes a clock line upon which a clock transmits a regular sequence of alternating 1's and 0's
- A single 1-0 transition is referred to as a clock cycle or bus cycle.
- All other devices on the bus can read the clock line.
- All events start at the beginning of a clock cycle

B) **Asynchronous Timing**

- The occurrence of one event on a bus follows and depends on the occurrence of a previous event.
- Harder to implement and text than synchronous timing.

4) Bus Width

- The width of data bus has an impact on the databus has an impact on the system performance.
- The wider data bus, the greater number of bit transferred at one time.
- The wider address bus, the greater range of location that can be referenced.

5) Data Transfer Type

- **Read-Modify-Write :** A read followed immediately by a write to the same address.
- **Read-After-Write :** Consisting of a write followed immediately by a read from the same address (for error checking purposes).

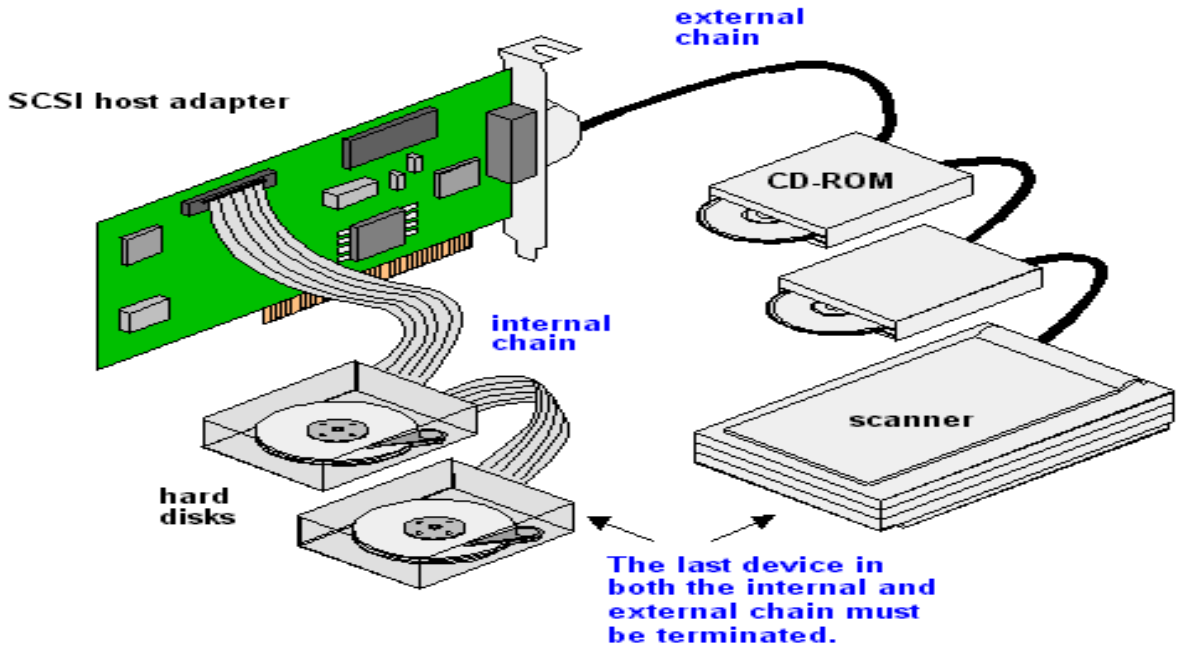
6) Block Data Transfer

- One address cycle followed by n data cycles.
- First data item to or from specified address.

- Remaining data items to or from subsequent addresses.

Small Computer systems interface (SCSI)

A small computer systems interface (SCSI) is a standard interface for connecting peripheral devices to a PC. Depending on the standard, generally it can connect up to 16 peripheral devices using a single bus including one host adapter. SCSI is used to increase performance, deliver faster data transfer transmission and provide larger expansion for devices such as CD-ROM drives, scanners, DVD drives and CD writers. SCSI is also frequently used with RAID, servers, high-performance PCs and storage area networks SCSI has a controller in charge of transferring data between the devices and the SCSI bus. It is either embedded on the motherboard or a host adapter is inserted into an expansion slot on the motherboard. The controller also contains SCSI basic input/output system, which is a small chip providing the required software to access and control devices. Each device on a parallel SCSI bus must be assigned a number between 0 and 7 on a narrow bus or 0 and 15 on a wider bus. This number is called an SCSI ID. Newer serial SCSI IDs such as serial attached SCSI (SAS) use an automatic process assigning a 7-bit number with the use of serial storage architecture initiators.



USB was designed to standardize the connection of peripherals like pointing devices, keyboards, digital still, and video cameras. But soon devices such as printers, portable media players, disk drives, and network adaptors to personal computers used USB to communicate and to supply electric power. It is commonplace to many devices and has largely replaced interfaces such as serial ports and parallel ports. USB connectors have replaced other types of battery chargers for portable devices with themselves.

USB(Universal Serial Bus)

Universal Serial Bus (USB) is an industry standard that establishes specifications for connectors, cables, and protocols for communication, connection, and power supply between personal computers and their peripheral devices. There have been 3 generations of USB specifications:

- USB 1.x
- USB 2.0
- USB 3.x

The first USB was formulated in the mid-1990s. USB 1.1 was announced in 1995 and released in 1996. It was too popular and grab the market till about the year 2000. In the duration of USB 1.1 Intel announced a USB host controller and Philips announced USB audio for isochronous communication with consumer electronics devices.

In April of 2000, USB 2.0 was announced. USB 2.0 has multiple updates and additions. The USB Implementer Forum (USB IF) currently maintains the USB standard and it was released in 1996.

Where are the USB ports?

In modern times, all computers contain at least one USB port in different locations. Below, a list is given that contains USB port locations on the devices that may help you out to find them.

Laptop computer: A laptop computer may contain one to four ports on the left or right side, and some laptops have on the behind of the laptop computer.

Desktop computer: Usually, a desktop computer has 2 to 4 USB ports in the front and 2 to 8 ports on the backside.

Tablet computer: On the tablet, a USB connection is situated in the charging port and is sometimes USB-C and usually micro USB.

Smartphone: In the form of micro USB or USB-C, a USB port is used for both data transfer and charging, similar to tablets on smartphones.

USB connector types

There are different shapes and sizes available for the USB connector. Also, there are numerous versions of USB connectors, such as Mini USB, Micro USB, etc.

1. **Mini -USB:** Mini USB is used with digital cameras and computer peripherals and divided into A-type, B-type and AB-type. It is also known as mini-B and is the most common type of interface. On the latest devices, Micro-USB and USB-C cables have largely replaced the mini-USB. It transfers data and power between two devices as it is made of coaxial cable. Also, it is applied to MP3 players, digital cameras, and mobile hard drives. In a mini USB cable, one-end is a much smaller quadrilateral hub, and the other end is a standard flat-head USB hub. Thus, it is easily plugged into mobile devices. The mini USB can also be used to transfer data between computers with at least one USB port but is mainly used for charging devices. It includes two advantages: Waterproofness and Portability.
2. **Micro-USB:** It is a reduced version of the USB (Universal Serial Bus). It was announced in 2007 and designed to replace mini-USB and developed for connecting compact and mobile devices such as digital cameras, smartphones, GPS devices, Mp3 players and photo printers.

   Micro A, micro B and **micro USB 3 are** the three varieties of Micro-USB. The type Micro-A and Micro-B have a connector size of 6.85 x 1.8 mm, although the Micro-A connector has a greater maximum overmild size. USB 3 micro is more similar to micro B, but it has better speed as compared to micro B because it includes an additional collection of pins on the side for twice the wires. Micro versions are hot-swappable, and plug-and-play like standard USB and micro-USB is still widely used with electronic devices.
3. **USB Type-C:** On most modern newer Android smartphones and other USB-connected devices, a USB Type-C cable is a relatively new type of connector. It is used for delivering data and power to computing devices. As compared to other forms of USB connections, USB-C cables are reversible; they can be plugged either way in the devices, whether they are upside down.

Parallel processing

Parallel processing is a computing technique when multiple streams of calculations or data processing tasks co-occur through numerous central processing units (CPUs) working concurrently. This article explains how parallel processing works and examples of its application in real-world use cases.

What Is Parallel Processing?

Parallel processing is a computing technique when multiple streams of calculations or data processing tasks co-occur through numerous central processing units (CPUs) working concurrently.

How Does Parallel Processing Work?

In general, parallel processing refers to dividing a task between at least two microprocessors. When processing is done in parallel, a big job is broken down into several smaller jobs better suited to the number, size, and type of available processing units. After the task is divided, each processor starts working on its part without talking to the others. Instead, they use software to stay in touch with each other and find out how their tasks are going.

After all the program parts have been processed, the result is a fully processed program segment. This is true whether the number of processors and tasks and processors were equal and they all finished simultaneously or one after the other.

There are two types of parallel processes: fine-grained and coarse-grained. Tasks communicate with one another numerous times per second in fine-grained parallelism to deliver results in real-time or very close to real-time. The slower speed of coarse-grained parallel processes results from their infrequent communication.

The main goal of parallel processing is to boost a computer's processing power and increase throughput, or the volume of work one can do in a given time. One can use many functional units to create a parallel processing system by carrying out similar or dissimilar activities concurrently.

This is a more sophisticated way of saying that dividing the work makes things easier. You could still split the load between different processors in the same computer, or it could be

split between different computers connected by a computer network. Users can accomplish the same objective in several ways.

Techniques in Parallel Processing

1. Pipelining:
    - Divides an operation into stages, with each stage executed in parallel.
2. Multithreading:
    - Allows multiple threads to run simultaneously, sharing resources within a processor.
3. Vectorization:
    - Utilizes SIMD units to perform the same operation on multiple data points simultaneously.
4. Load Balancing:
    - Distributes workload evenly among processors to avoid bottlenecks.
5. Synchronization:
    - Coordinates the execution of parallel tasks, ensuring correct ordering and data consistency.

Parallel Processing Examples

Parallel processing or parallel computing, has many important uses today. This includes:
1. Supercomputers for use in astronomy
2. Making predictions in agriculture
3. Risk calculations and cryptocurrencies in banking
4. Video post-production effects
5. The American Summit computer
6. Accurate medical imaging
7. Desktops and laptops

Linear Pipelining

A linear pipeline is a design strategy where a sequence of processing stages (or "pipeline stages") is arranged linearly, so each stage completes a part of the task and passes the result to the next stage. It allows multiple instructions or operations to be processed simultaneously, with each one being at a different stage, increasing the throughput of the system.

Key Concepts of Linear Pipelines:

1. Stages: The pipeline is divided into distinct stages, each handling a specific part of instruction processing (like fetch, decode, execute, and write-back). Each stage operates independently and is synchronized with a clock cycle.

2. Pipeline Phases:

- Instruction Fetch (IF): Retrieves the instruction from memory.

- Instruction Decode (ID): Decodes the fetched instruction to determine the operation and the operands.

- Execute (EX): Performs the required operations (arithmetic, logic, memory access).

- Memory Access (MEM): Accesses the memory if needed (e.g., for load/store instructions).

- Write-back (WB): Writes the result of the instruction back to a register.

For example, a linear pipeline in a simple CPU might be a 5-stage pipeline following these phases: IF, ID, EX, MEM, and WB.

3. Parallel Processing: Linear pipelines enable parallel processing by allowing multiple instructions to be in the pipeline simultaneously, with each instruction occupying a different stage. For instance, while one instruction is in the execute stage, the next instruction can be in the decode stage, and the following one in the fetch stage.

4. Pipeline Hazards:

- Data Hazards: Occur when an instruction depends on the result of a previous instruction that hasn't completed its write-back.

- Control Hazards: Arise due to branch instructions that affect the flow of instructions in the pipeline.

- Structural Hazards: Occur when hardware resources are insufficient for handling the parallelism in the pipeline.

5. Pipeline Performance:

- Throughput: The number of instructions that complete per unit time, which improves with pipelining since multiple instructions are processed at once.

- Latency: The time taken for a single instruction to go through the entire pipeline, which generally stays the same as a non-pipelined design.

Example of Linear Pipeline Execution

Assume a 5-stage linear pipeline, and let's look at how it would process instructions:

Clock Cycle	IF (Fetch)	ID (Decode)	EX (Execute)	MEM (Memory)	WB (Write-back)
1	Inst 1				

| 2 | Inst 2 | Inst 1 | | | |

| 3 | Inst 3 | Inst 2 | Inst 1 | | |

| 4 | Inst 4 | Inst 3 | Inst 2 | Inst 1 | |

| 5 | Inst 5 | Inst 4 | Inst 3 | Inst 2 | Inst 1 |

| 6 | Inst 6 | Inst 5 | Inst 4 | Inst 3 | Inst 2 |

| ... | ... | ... | ... | ... | ... |

In this example:

- Instruction 1 completes its write-back stage in clock cycle 5.

- Instruction 2 completes in clock cycle 6, and so on.

- Each instruction is processed in a sequence through the pipeline stages, thus enabling multiple instructions to be completed in a shorter amount of time.

Advantages of Linear Pipelines:

- Increased Throughput: By overlapping the execution of multiple instructions, the pipeline increases the instruction throughput.

- Better Resource Utilization: Each stage of the pipeline is designed to perform a specific task, optimizing resource usage.

- Reduced Idle Time: The CPU components stay active for longer, improving efficiency.

Disadvantages of Linear Pipelines

- Pipeline Hazards: Hazards may require inserting *stall* cycles (delays) or additional control logic to handle dependencies and branch instructions.

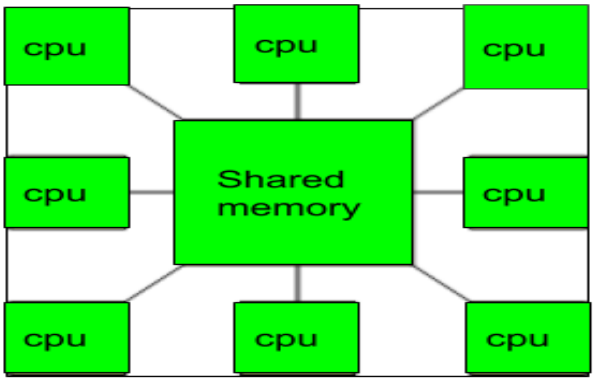
- Complexity in Control: Managing hazards, dependencies, and ensuring data integrity can complicate control unit design.

- Diminishing Returns: Increasing pipeline stages beyond a point yields limited performance benefits due to the increase in complexity and delays.

Multiprocessor

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.



Applications of Multiprocessor –

1. As a uniprocessor, such as single instruction, single data stream (SISD).
2. As a multiprocessor, such as single instruction, multiple data stream (SIMD), which is usually used for vector processing.
3. Multiple series of instructions in a single perspective, such as multiple instruction, single data stream (MISD), which is used for describing hyper-threading or pipelined processors.
4. Inside a single system for executing multiple, individual series of instructions in multiple perspectives, such as multiple instruction, multiple data stream (MIMD).

Benefits of using a Multiprocessor –

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

Advantages:

Improved performance: Multiprocessor systems can execute tasks faster than single-processor systems, as the workload can be distributed across multiple processors.

Better scalability: Multiprocessor systems can be scaled more easily than single-processor systems, as additional processors can be added to the system to handle increased workloads.

Increased reliability: Multiprocessor systems can continue to operate even if one processor fails, as the remaining processors can continue to execute tasks.

Reduced cost: Multiprocessor systems can be more cost-effective than building multiple single-processor systems to handle the same workload.

Enhanced parallelism: Multiprocessor systems allow for greater parallelism, as different processors can execute different tasks simultaneously.

Disadvantages:

Increased complexity: Multiprocessor systems are more complex than single-processor systems, and they require additional hardware, software, and management resources.

Higher power consumption: Multiprocessor systems require more power to operate than single-processor systems, which can increase the cost of operating and maintaining the system.
Difficult programming: Developing software that can effectively utilize multiple processors can be challenging, and it requires specialized programming skills.
Synchronization issues: Multiprocessor systems require synchronization between processors to ensure that tasks are executed correctly and efficiently, which can add complexity and overhead to the system.
Limited performance gains: Not all applications can benefit from multiprocessor systems, and some applications may only see limited performance gains when running on a multiprocessor system.

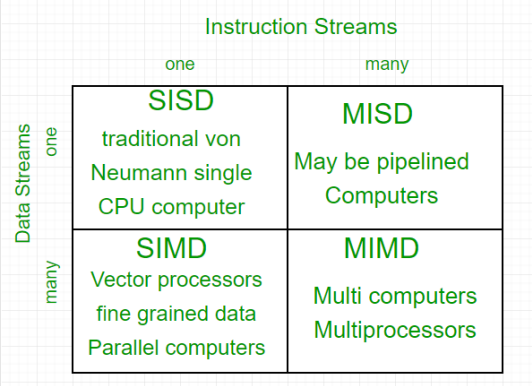
Flynn''s Classification

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.

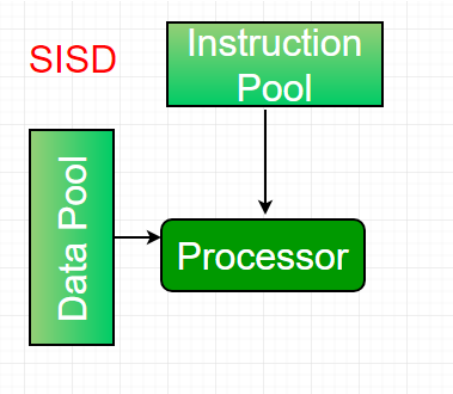
Flynn's taxonomy is a useful tool for understanding different types of computer architectures and their strengths and weaknesses. The taxonomy highlights the importance of parallelism in modern computing and shows how different types of parallelism can be exploited to improve performance.

systems are classified into four major categories:



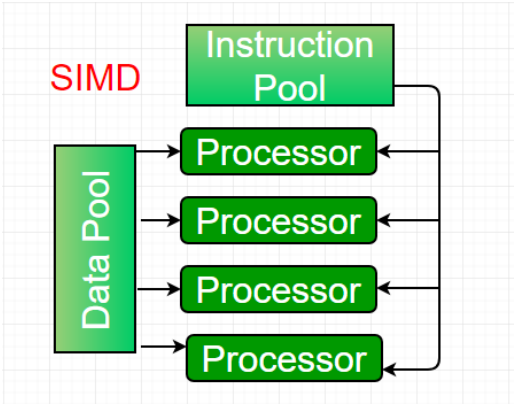
Flynn's classification –

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.



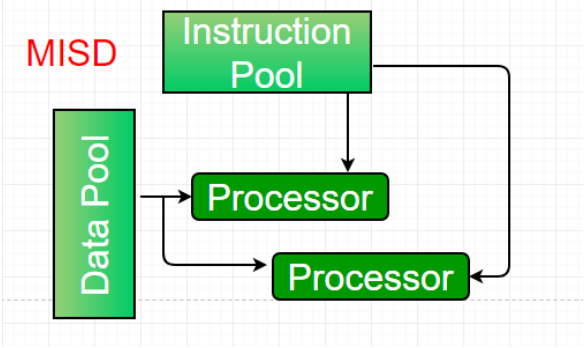
The speed of the processing element in the SISD model is limited(dependent) by the rate at which the computer can transfer information internally. Dominant representative SISD systems are IBM PC, and workstations.

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.



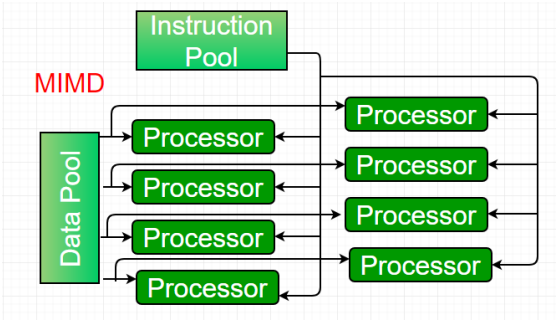
Dominant representative SIMD systems are Cray's vector processing machines.

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.



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.



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.

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.