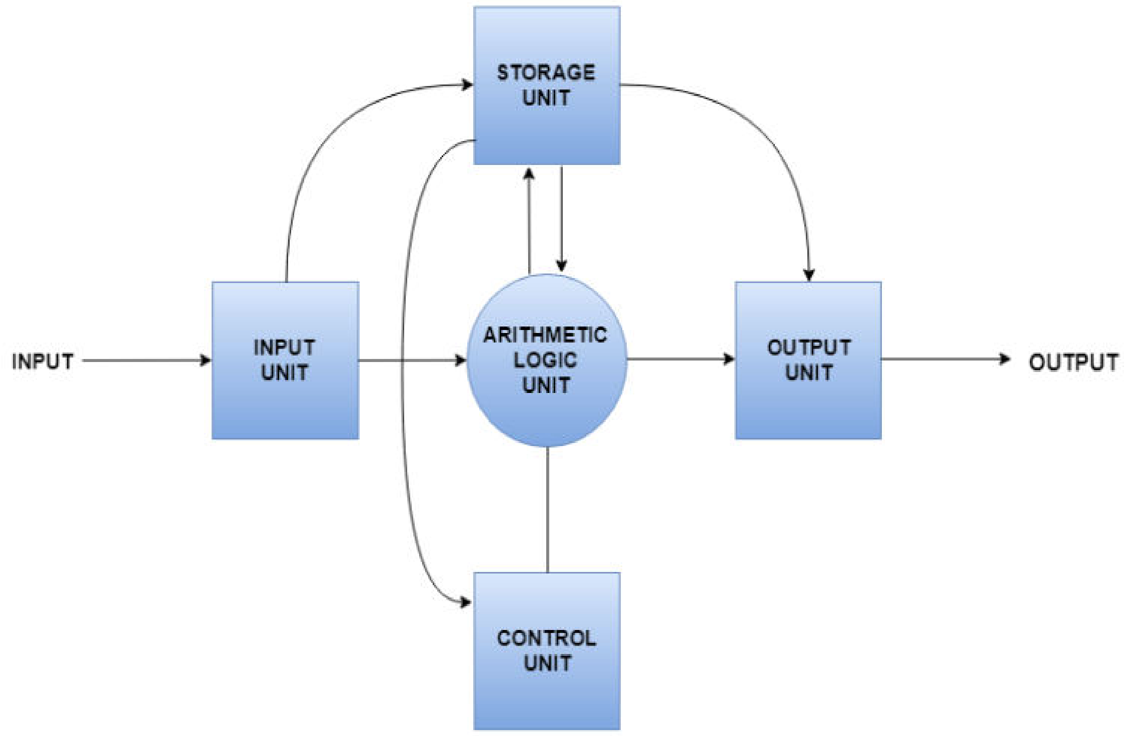
task 1 : the basic way of Computer architecture and what is the best?

Computer architecture

comprises rules, methods, and procedures that describe the execution and functionality of the entire computer system. In general terms, computer architecture refers to how a computer system is designed using compatible technologies.

Computer System Architecture

A computer system is basically a machine that simplifies complicated tasks. It should maximize performance and reduce costs as well as power consumption.The different components in the Computer System Architecture are Input Unit, Output Unit, Storage Unit, Arithmetic Logic Unit, Control Unit etc.



<u>A diagram that shows the flow of data between these units is as follows</u>

The input data travels from input unit to ALU. Similarly, the computed data travels from ALU to output unit. The data constantly moves from storage unit to ALU and back again. This is because stored data is computed on before being stored again. The control unit controls all the other units as well as their data.

- ## <u>Input Unit</u>

The input unit provides data to the computer system from the outside. So, basically it links the external environment with the computer. It takes data from the input devices, converts it into machine language and then loads it into the computer system. Keyboard, mouse etc. are the most commonly used input devices.

- **Output Unit**

The output unit provides the results of computer process to the users i.e it links the computer with the external environment. Most of the output data is the form of audio or video. The different output devices are monitors, printers, speakers, headphones etc.

- **Storage Unit**

Storage unit contains many computer components that are used to store data. It is traditionally divided into primary storage and secondary storage.Primary storage is also known as the main memory and is the memory directly accessible by the CPU. Secondary or external storage is not directly accessible by the CPU. The data from secondary storage needs to be brought into the primary storage before the CPU can use it. Secondary storage contains a large amount of data permanently.

- **Arithmetic Logic Unit**

All the calculations related to the computer system are performed by the arithmetic logic unit. It can perform operations like addition, subtraction, multiplication, division etc. The control unit transfers data from storage unit to arithmetic logic unit when calculations need to be performed. The arithmetic logic unit and the control unit together form the central processing unit.

- **Control Unit**

This unit controls all the other units of the computer system and so is known as its central nervous system. It transfers data throughout the computer as required including from storage unit to central processing unit and vice versa. The control unit also dictates how the memory, input output devices, arithmetic logic unit etc. should behave.

Types of Computer Architecture

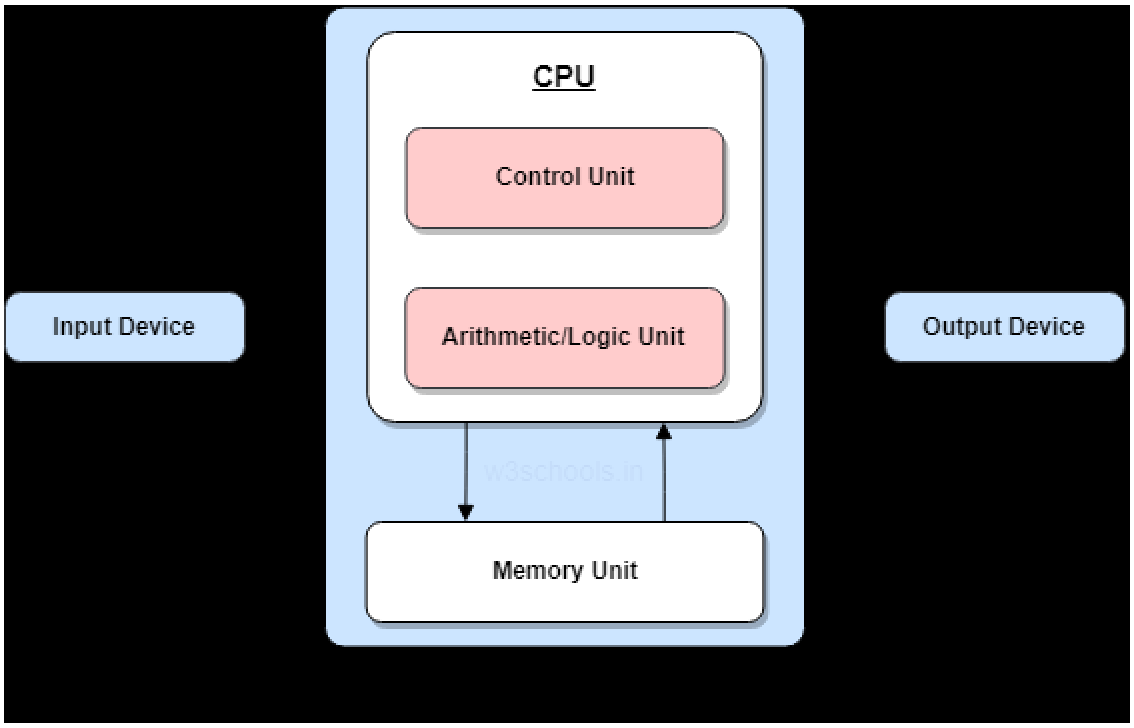
- Von-Neumann Architecture

- Harvard Architecture
- Instruction Set Architecture
- Micro-architecture
- System Design

Von-Neumann Architecture

John von Neumann coined and developed this architecture. The computer we are using nowadays is based on the von Neumann architecture. It has some concepts. It is also known as Princeton architecture. It renders a unique design for the electronic digital systems having the following components:

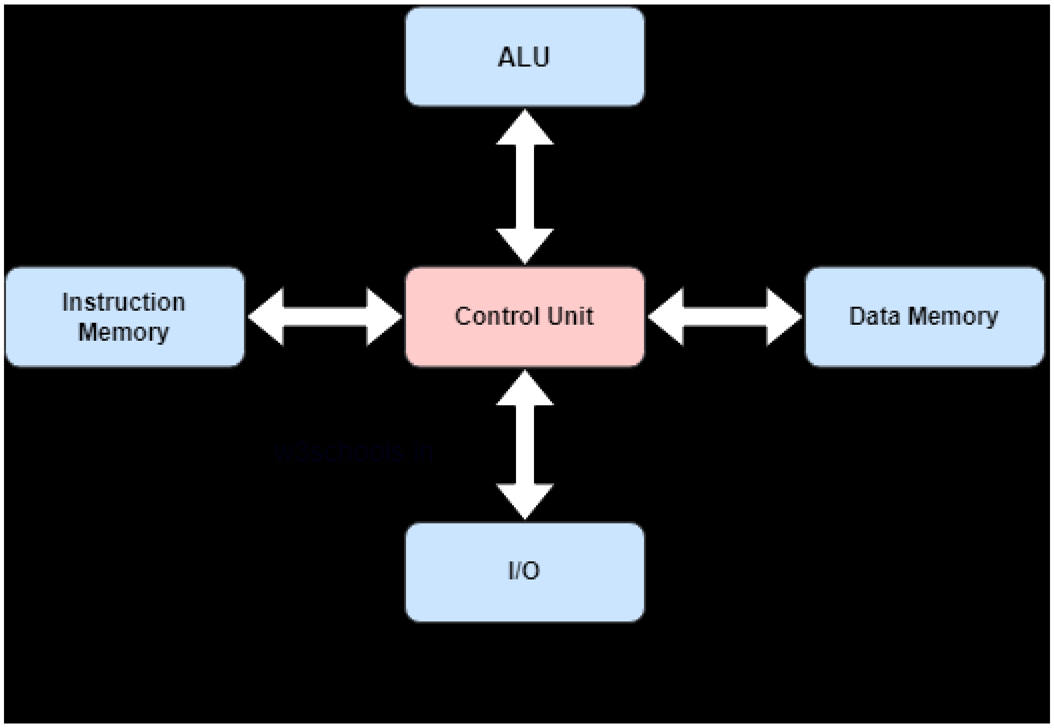
- **A Central Processing Unit (CPU) with arithmetic and logic unit (ALU) and processors with attached registers.**
- **A memory that can store data and instructions.**
- **External mass storage or secondary storage.**
- **A Control Unit (CU) with the ability to hold instructions in the program counter (PC) or instruction register (IR).**
- **Input and output mechanisms and peripherals.**



Harvard Architecture

Harvard Architecture consists of code and data laid in distinct memory sections. It requires a separate memory block for data and instruction. It has solely contained data storage within the Central Processing Unit (CPU). A single collection of clock cycles is needed. Data accessibility in one memory is done by a single memory location in the case of Harvard architecture.

One typical example is the Punch card. Moreover, modern computers may have the latest CPU processes for both methods but disparate them in a hardware design.



Instruction Set Architecture

Another notable digital computer architecture is the Instruction Set Architecture. The architecture holds a collection of instructions that the processor renders and surmises. It consists of two instruction sets: RISC (Reduced Instruction Set Computer) and CISC (Complex Instruction Set Computer).

It enables versatile implementations of an ISA; commonly differ in features such as performance, physical size, and monetary price. It empowers the evolution of the micro-architectures, implementing ISA as an exclusive, higher-performance system that can run software on preceding generations of execution.

Micro-architecture

Micro-architecture is the structural design of a microprocessor. This computer organization leverages a method where the instruction set architecture holds a built-in processor. Engineers and hardware scientists implement instruction set architecture (ISA) with various micro-architectures that vary because of changing technology. It includes the technologies used, resources, and methods. Using this, the processors physically devised to administer a particular instruction set.

Simply, it is a logical form of all electronic elements and data pathways present in the microprocessor, designed in a specific way. It allows for the optimal completion of instructions. In academe, it is called computer organization.

System Design

System design itself defines a design that can serve user requirements like system architecture, computer modules having various interfaces, and data management within a system. The term product development is connective to the system design. It is the process by which we can take marketing information to create a product design.

Von-Neumann Architecture

is the best because The von Neumann design thus constitutes the foundation of modern computing. The Harvard architecture, a similar model, had committed data addresses and buses for reading and writing to memory. It wins because von Neumann's architecture was easier to execute in real hardware.

task 2: programming languages support garbage collection

Garbage Collection:

is a process for automatic memory management. The collector manages to collect garbage code (data, objects, functions, variables) or memory occupied by objects that are no longer in use.

Garbage collection plays an important part in some modern languages. It is implemented differently in different languages, though. Higher-level languages are more likely to have garbage collection as a standard feature.

Garbage collection frees developers from manual memory management. For example, when you're working with low-level languages such as C language, most of the time you are going to

manually allocate and deallocate the memory.

programming languages support garbage collection:

- Lisp has used garbage collection since John McCarthy invented it in 1958. Java, Scala, Python, and .NET/C# are all popular GC languages. Additional garbage collection languages include the relatively young Go, Ruby, D, OCaml, and Swift, as well the older languages Eiffel, Haskell, ML, Modula-3, Perl, Prolog, Scheme, and Smalltalk.

- Java, Python, and .NET/C# are some of the more popular programming languages that implement garbage collection. The Java virtual machine (JVM) actually provides four different garbage collectors: serial, parallel, concurrent mark and sweep, and G1GC, the garbage first garbage collector. G1GC is now the default in Java; it is a regionalized and generational parallel compacting collector that achieves soft real-time goals.

- Python, specifically the standard CPython implementation, combines reference counting with three-level generational collection that only focuses on cleaning container objects. The .NET CLR (common language runtime) uses a three-level generational mark and compact collection algorithm. The CLR also segregates memory objects into two heaps, one for large objects (85,000 bytes or higher) and one for small objects; the large object heap usually isn't compacted, just marked and swept, but can be compacted if necessary.

-The C language provides several functions for handling memory allocation and management.

- **malloc — This function allocates memory and leaves the memory uninitialized.**
- **free — This function releases a block of memory specified by address.**
- **calloc— This function allocates memory and initializes all bits to zero.**
- **dealloc — This function re-allocates memory extending it up to new size.**

Working with C sometimes can be very painful. You need to care about memory management, making a safe copy, deallocating at the right time and correctly, etc.

There's a library for garbage collection in C and C++ called Boehm GC. It is a conservative garbage collector for replacing the for C malloc or C++ new.

- Java is another language that makes use of garbage collection. It is automatically done by the JVM (Java Virtual Machine), a virtual machine that enables a computer to run Java programs.

Unlike in C, you don't need to handle the memory management using functions. In Java, there are four types of garbage collection:

- **Serial Garbage Collector — All events of the collector are conducted in a single-thread environment.**
- **Parallel Garbage Collector — Uses multiple threads to speed up garbage collection. It is the default collector used by the JVM.**
- **Concurrent Mark Sweep (CMS) Garbage Collector — Uses multiple threads that scan the heap memory.**
- **Garbage First (G1) Garbage Collector — Used when there's a large data (more than 4GB) memory (heap space).**

Choosing the right garbage collection for your Java application really depends on a lot of factors—the way your application handles memory, the lifetime of your objects, how your machine works, etc.

-JavaScript works with garbage collection. When you create something in JavaScript—for example, a function—the value is allocated. When this function or other creation is no longer used, it is deallocated from memory automatically.

You might have heard of memory leaks before. Memory leaks usually happen when the memory is incorrectly allocated or when data is allocated in memory but cannot be accessed by the running code.

task 3: what is Bios? and why we use it?

- Short for Basic Input/Output System, the BIOS (pronounced bye-oss) is a ROM chip found on motherboards that lets you access and set up your computer system at the most basic level.

- A BIOS contains the instructions your computer needs to load its basic hardware, including the POST mentioned above. If your system fails the POST, you will hear a series of beeps; different beep sequences indicate various issues.

- BIOS firmware is non-volatile, meaning that the settings are saved and can be recovered even if the machine no longer has power.

The Functions of a BIOS:

- **POST: The POST function tests the hardware before loading the operating system; we already discussed this function earlier.**
- **Bootstrap loader: This function locates a capable operating system. If the loader finds that system, the BIOS passes control over to it.**
- **BIOS drivers: These are low-level drivers that give your system basic control over its hardware.**
- **BIOS setup: This function is a configuration program that lets you configure your system's hardware settings. This configuration includes system settings like time, date, and passwords.**

And here's a list of functions you can do with most BIOS systems:

- **Change the boot order**
- **Load BIOS setup defaults**

- **Flash (Update) BIOS**
- **Create/Delete a BIOS password**
- **Change the date and time**
- **Change floppy drive settings***
- **Change hard drive settings**
- **Change CD/DVD/BD drive settings**
- **View the amount of memory Installed**
- **Change the boot up num lock status**
- **Enable or disable the computer logo**
- **Enable or disable the quick Power On Self Test (POST)**
- **Enable or disable the CPU internal cache**
- **Enable or disable the BIOS caching**
- **Change the CPU settings**
- **Change the memory settings**
- **Change system voltages**
- **Enable/disable RAID**
- **Enable/disable the onboard USB**
- **Enable/disable the onboard IEEE1394**
- **Enable/disable the onboard audio**
- **Enable/disable the onboard floppy controller***
- **Enable/disable the onboard serial or parallel ports**
- **Enable/disable ACPI**
- **Change the ACPI suspend type**

- **Change the power button function**
- **Change the power-on settings**
- **Change which display gets initialized first on the multi-display setups**
- **Reset the Extended System Configuration Data (ESCD)**
- **Enable or disable the BIOS control of system resources**
- **Alter the fan's speed settings**
- **View CPU and system temperatures**
- **View the fan speeds**
- **View system voltages**

task 4: linx vs unix? example?

Linux is an operating system that was developed by Linus Torvalds in 1991. The name "Linux" originates from the Linux kernel. It is an open-source software that is completely free to use. It is used for computer hardware and software, game development, mainframes, etc. It can run various client programs.

Unix is a portable, multi-tasking, bug-fixing, multi-user operating system developed by AT&T. It started as a one-man venture under the initiative of Ken Thompson of Bell Labs. It proceeded to turn out to become the most widely used operating system. It is used in web servers, workstations, and PCs. Many business applications are accessible on it.

Linux and Unix are both operating systems that are commonly used in enterprise and server environments. While there are some similarities between them, there are also some key differences.

Differences	Linux	Unix
Origins	Linux was developed in the 1990s by Linus Torvalds as a free and open-source alternative to Unix.	Unix was developed in the 1970s at Bell Labs
Introduction	Linux is Open Source, and a large number of programmers work together online and contribute to its development.	Unix was developed by AT&T Labs, different commercial vendors, and non-profit organizations.

Licensing	Linux, on the other hand, is open-source software and can be used freely without any licensing fees.	Unix is a proprietary ary operating system, meaning that it requires a license to use.
Kernels	both have a similar design but are less complex than the Unixhold-upthat kernel.	both have a similar design but larger and more complex than the Linux kernel.
Availability	On the other hand, Linux is widely used on both enterprise and personal computers.	Unix is typically found on enterprise-level servers and workstations and is less commonly used on personal computers.
Community Support:	Linux has a large and active community of developers and users who contribute to its development and provide support.	While Unix also has a community, it is generally smaller and more focused on enterprise-level users.
Accessibility	It is an open-source operating system which is freely accessible to everyone.	It is an operating system which can only be utilized by its copywriters.

bug fixing time	Threat recognition and solution is very fast because Linux is mainly community driven. So, if any Linux client poses any sort of threat, a team of qualified developers starts working to resolve this threat.	Unix clients require longer hold up time, to get the best possible bug-fixing,and a patch
File system supports	File system supports – Ext2, Ext3, Ext4, Jfs, ReiserFS, Xfs, Btrfs, FAT, FAT32, NTFS	File system supports – jfs, gpfs, hfs, hfs+, ufs, xfs, zfs
Graphical User Interface	Linux provides two GUIs, KDE and Gnome. But there are many other options. For example, LXDE, Xfce, Unity, Mate, and so on.	Initially, Unix was a command-based OS, however later a GUI was created called Common Desktop Environment. Most distributions now ship with Gnome.
Use Cases	It is used everywhere from servers, PCs, smartphones, tablets to mainframes.	It is used on servers, workstations, and PCs.

Shell Compatibility	The default interface is BASH (Bourne Again Shell). Anybody can use Linux whether a home client, developer or a student.	It initially used Bourne shell. But it is also compatible with other GUIs. Developed mainly for servers, workstations, and mainframes.
Source Code Availability	The source is accessible to the general public.	The source is not accessible to the general public.
Hardware Compatibility	Originally developed for Intel's x86 hardware processors. It is available for more than twenty different types of CPU which also includes an ARM.	It is available on PA-RISC and Itanium machines.
Virus Threats	It has about 60-100 viruses listed to date.	It has about 85-120 viruses listed to date (rough estimate).
Operating System Versions	Some Linux versions are Ubuntu, Debian GNU, Arch Linux, etc.	Some Unix versions are SunOS, Solaris, SCO UNIX, AIX, HP/UX, ULTRIX, etc.

In summary, while Unix and Linux share some similarities in terms of their design and functionality, they also have some key differences in terms of licensing, kernel design, command line interface, availability, and community support. Ultimately, the choice between Unix and Linux will depend on the specific needs of the user and the intended use case.

task 5 : what is Fragmentation

Fragmentation is an unwanted problem in the operating system in which the processes are loaded and unloaded from memory, and free memory space is fragmented. Processes can't be assigned to memory blocks due to their small size, and the memory blocks stay unused. It is also necessary to understand that as programs are loaded and deleted from memory, they generate free space or a hole in the memory. These small blocks cannot be allotted to new arriving processes, resulting in inefficient memory use.

The conditions of fragmentation depend on the memory allocation system. As the process is loaded and unloaded from memory, these areas are fragmented into small pieces of memory that cannot be allocated to incoming processes. It is called fragmentation.

Causes of Fragmentation

User processes are loaded and unloaded from the main memory, and processes are kept in memory blocks in the main memory. Many spaces remain after process loading and swapping that another process cannot load due to their size. Main memory is available, but its space is insufficient to load another process because of the dynamical allocation of main memory processes.

Types of Fragmentation

There are mainly two types of fragmentation in the operating system. These are as follows:

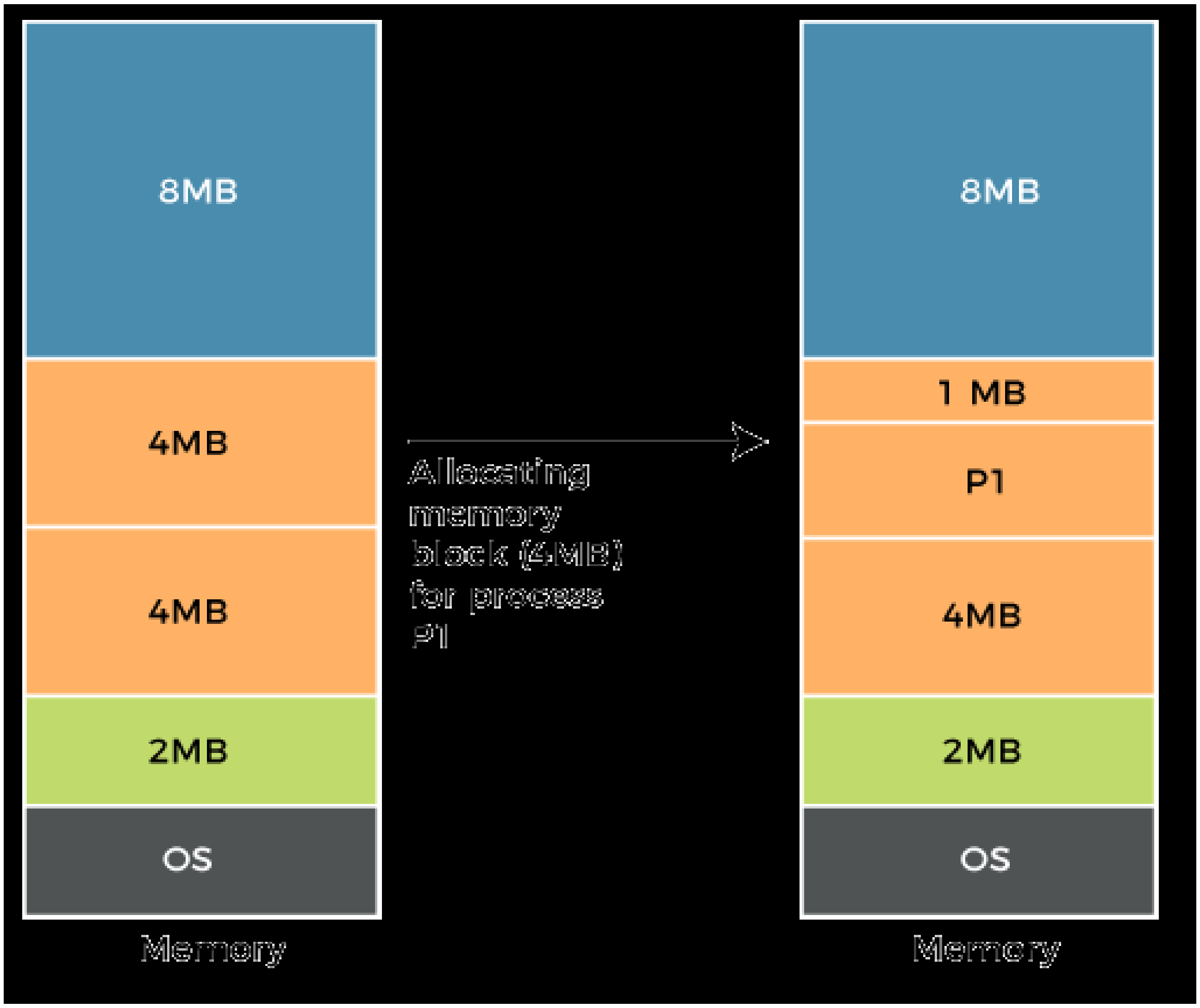
- Internal Fragmentation
- External Fragmentation

Internal Fragmentation

When a process is allocated to a memory block, and if the process is smaller than the amount of memory requested, a free space is created in the given memory block. Due to this, the free space of the memory block is unused, which causes internal fragmentation.

For Example:

Assume that memory allocation in RAM is done using fixed partitioning (i.e., memory blocks of fixed sizes). 2MB, 4MB, 4MB, and 8MB are the available sizes. The Operating System uses a part of this RAM.



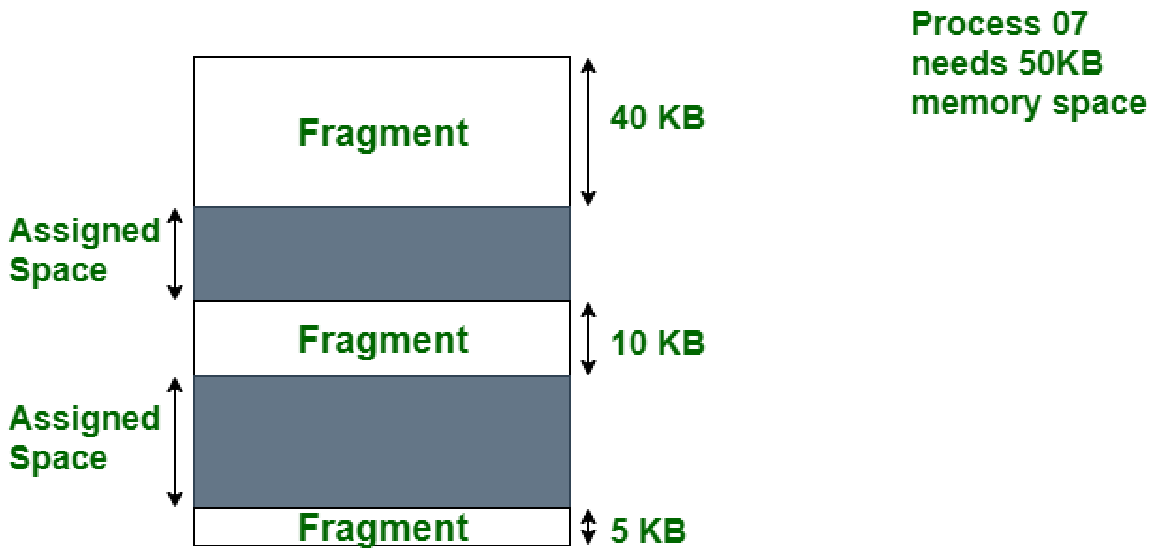
-a process P1 with a size of 3MB arrives and is given a memory block of 4MB. As a result, the 1MB of free space in this block is unused and cannot be used to allocate memory to another process. It is known as internal fragmentation.

How to avoid internal fragmentation?

The problem of internal fragmentation may arise due to the fixed sizes of the memory blocks. It may be solved by assigning space to the process via dynamic partitioning. Dynamic partitioning allocates only the amount of space requested by the process. As a result, there is no internal fragmentation.

External Fragmentation

External fragmentation happens when a dynamic memory allocation method allocates some memory but leaves a small amount of memory unusable. The quantity of available memory is substantially reduced if there is too much external fragmentation. There is enough memory space to complete a request, but it is not contiguous. It's known as external fragmentation.



How to remove external fragmentation?

This problem occurs when you allocate RAM to processes continuously. It is done in paging and segmentation, where memory is allocated to processes non-contiguously. As a result, if you remove this condition, external fragmentation may be decreased.

Compaction is another method for removing external fragmentation. External fragmentation may be decreased when dynamic partitioning is used for memory allocation by combining all free memory into a single large block. The larger memory block is used to allocate space based on the requirements of the new processes. This method is also known as defragmentation.

Advantages:

- **Fast Data Writes**

Data write in a system that supports data fragmentation may be faster than reorganizing data storage to enable contiguous data writes.

- **Fewer Failures**

If there is insufficient sequential space in a system that does not support fragmentation, the write will fail.

- **Storage Optimization**

A fragmented system might potentially make better use of a storage device by utilizing every available storage block.

Disadvantages

- **Need for regular defragmentation**

A more fragmented storage device's performance will degrade with time, necessitating the requirement for time-consuming defragmentation operations.

- **Slower Read Times**

The time it takes to read a non-sequential file might increase as a storage device becomes more fragmented.

task 6: compare between 3 scheduling algorithms

- Round robin
- Priority Scheduling
- First Come First Serve

Round robin

is a CPU scheduling algorithm where each process is cyclically assigned a fixed time slot. It is the preemptive version of First come First Serve CPU Scheduling algorithm. Round Robin CPU Algorithm generally focuses on Time Sharing technique.

Characteristics of Round robin:

- It's simple, easy to use, and starvation-free as all processes get the balanced CPU allocation.

- One of the most widely used methods in CPU scheduling as a core.
- it is considered preemptive as the processes are given to the CPU for a very limited time.

Advantages of Round robin:

- Round robin seems to be fair as every process gets an equal share of CPU.
- The newly created process is added to the end of the ready queue.
- No starvation as every process gets a chance for its execution.
- Used in time-sharing systems.

Disadvantages:

- The CPU is left idle due to a lot of context switching.

Priority Scheduling

Preemptive Priority CPU Scheduling Algorithm is a pre-emptive method of CPU scheduling algorithm that works based on the priority of a process. In this algorithm, the editor sets the functions to be as important, meaning that the most important process must be done first. In the case of any conflict, that is, where there are more than one processor with equal value, then the most important CPU planning algorithm works on the basis of the FCFS (First Come First Serve) algorithm.

Characteristics of Priority Scheduling:

- Schedules tasks based on priority.
- When the higher priority work arrives while a task with less priority is executed, the higher priority work takes the place of the less priority one and
- The latter is suspended until the execution is complete.
- Lower is the number assigned, higher is the priority level of a process.

Disadvantages of Priority Scheduling:

- One of the most common demerits of the Preemptive priority CPU scheduling algorithm is the Starvation Problem. This is the problem in which a process has to wait for a longer amount of time to get scheduled into the CPU. This condition is called the starvation problem.

First Come First Serve:

FCFS considered to be the simplest of all operating system scheduling algorithms. First come first serve scheduling algorithm states that the process that requests the CPU first is allocated the CPU first and is implemented by using FIFO queue.

Characteristics of FCFS:

- FCFS supports non-preemptive and preemptive CPU scheduling algorithms.
- Tasks are always executed on a First-come, First-serve concept.
- FCFS is easy to implement and use.
- This algorithm is not much efficient in performance, and the wait time is quite high.

Advantages of FCFS:

- Easy to implement
- First come, first serve method

Disadvantages of FCFS:

- FCFS suffers from Convoy effect.
- The average waiting time is much higher than the other algorithms.
- FCFS is very simple and easy to implement and hence not

much efficient.

task 7: parallel processing vs threads

Parallel processing

A processing technique in which multiple processors or multiple processing cores in a single computer work together to complete one job more quickly.

Multithreaded programming

is programming multiple, concurrent execution threads. These threads could run on a single processor. Or there could be multiple threads running on multiple processor cores.

- **Multithreading on a Single Processor**

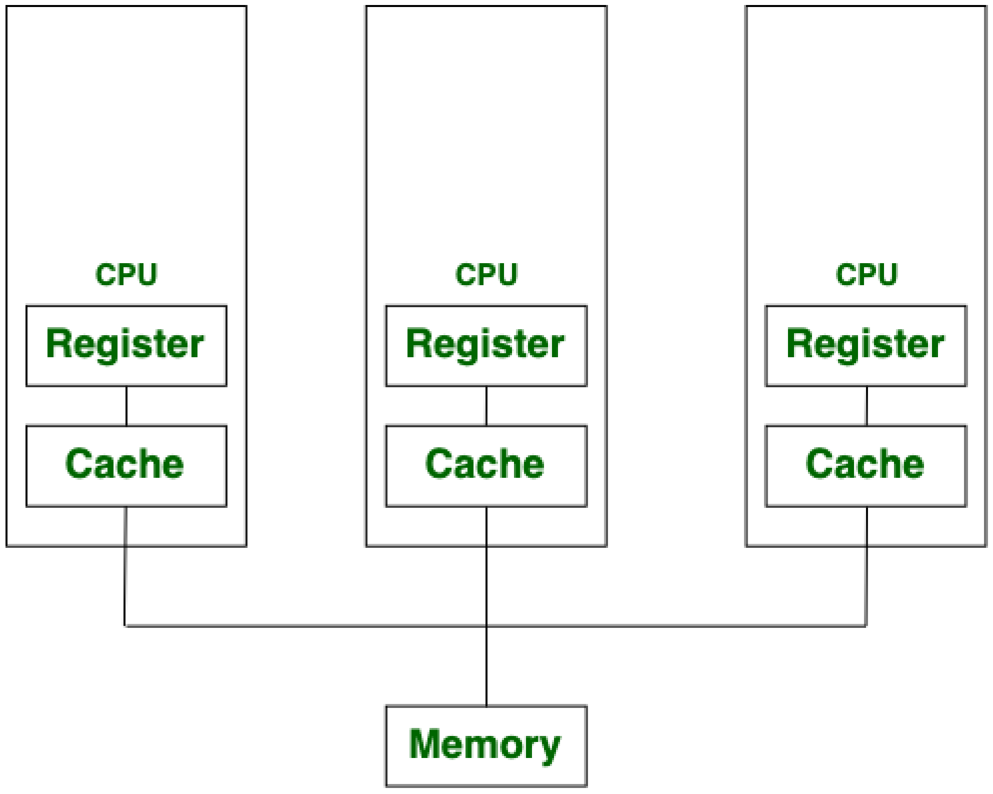
Multithreading on a single processor gives the illusion of running in parallel. In reality, the processor is switching by using a scheduling algorithm. Or, it's switching based on a combination of external inputs (interrupts) and how the threads have been prioritized.

- **Multithreading on Multiple Processors**

Multithreading on multiple processor cores is truly parallel. Individual microprocessors work together to achieve the result more efficiently. There are multiple parallel, concurrent tasks happening at once.

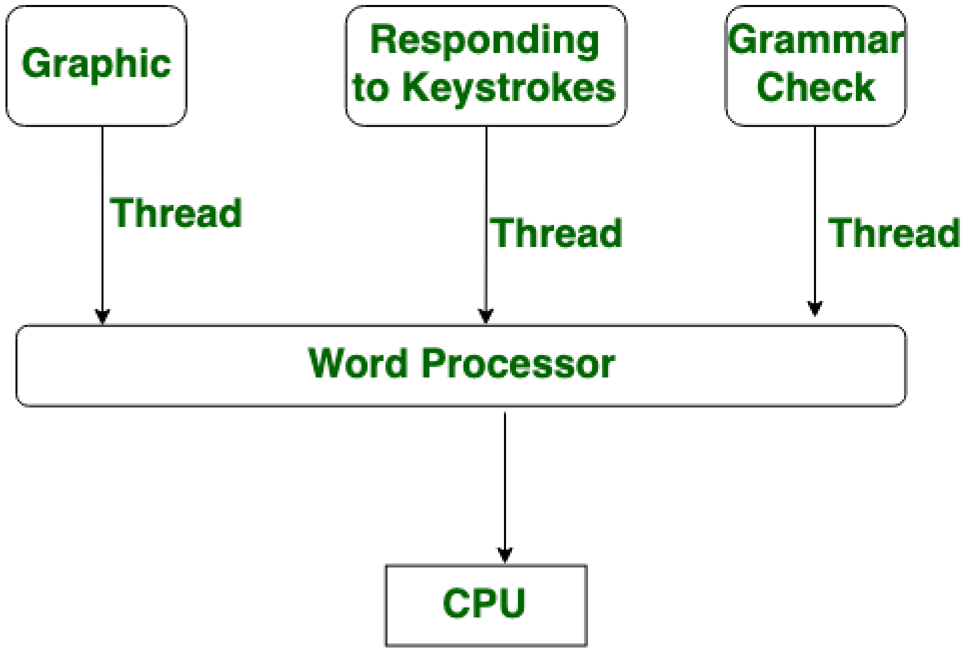
Both **Multiprocessing** and **Multithreading** are used to increase the computing power of a system. **Multiprocessing:** Multiprocessing is a system that has more than one or two processors. In Multiprocessing, CPUs are added for increasing computing speed of the system. Because of Multiprocessing, There are many processes are executed simultaneously. Multiprocessing are classified into two categories:

```
1. Symmetric Multiprocessing
2. Asymmetric Multiprocessing
```



Multiprocessing

Multithreading: Multithreading is a system in which multiple threads are created of a process for increasing the computing speed of the system. In multithreading, many threads of a process are executed simultaneously and process creation in multithreading is done according to economical.



Multithreading

Difference between Multiprocessing and Multithreading:

S.NO	Multiprocessing	Multithreading
1	In Multiprocessing, CPUs are added for increasing computing power .	While In Multithreading, many threads are created of a single process for increasing computing power.

2	In Multiprocessing, Many processes are executed simultaneously.	While in multithreading, many threads of a process are executed simultaneously.
3	Multiprocessing are classified into **Symmetric and Asymmetric.**	While Multithreading is not classified in any categories.
4	In Multiprocessing, Process creation is a time-consuming process.	While in Multithreading, process creation is according to economical.
5	In Multiprocessing, every process owned a separate address space.	While in Multithreading, a common address space is shared by all the threads.

task 8: programming languges support multithreading

- . Programming languages, including C and C++, have been developed to allow the use and management of several threads. Both C and C++ now have libraries for the thread.

task 9: clean code

Clean code is clear, understandable, and maintainable. When you write clean code, you're keeping in mind the other people who may read and interpret your code at a later time. You're helping others understand the purpose of your code so that they can make changes to it eventually.

task 10: The SOLID Principles of clean code

- S – Single Responsibility principle
- O – Open-Closed principle
- L – Liskov Substitution principle
- I – Interface Segregation principle
- D – Dependency Inversion principle

Single Responsibility principle

According to Rober C. Martin, the Single Responsibility principle means "a class or modules should have one, and only one, reason to change," or we can say, "There should be no more than one reason to modify a class or a module."

Example: An EmployeeReviewer class is only responsible for counting the appraisal score of an employee according to his/her performance criteria fixed by the company. This class should only change if the criteria fixed by the company will change.

Open-Closed Principle (OCP)

The OCP states that, "A software artifact should be open for extension, but closed for modification." It means that a software structure should be designed so that, with every new release we be adding code for new functionality instead of modifying the previous one. In other words, the behavior of a software artifact ought to be extendable, without having to modify that artifact.

Example: Software development is similar to building a multi-story building. So the design should be like that if we are going to add a new feature then this can be easily achieved by adding some new functions/methods or classes instead of modifying the previous one. If we are planning to make the house that already has a room in it, then to make a kitchen, if we have to modify the already created rooms for it, then obviously the design for building the house is not correct. The same rules apply for designing the software system.

Liskov Substitution Principle (LSP)

LSP tells us that "Derived classes must be substitutable for their base classes." What exactly Barbara Liskov wrote "What is wanted here is something like the following substitution property: If for each object O1 of type S there is an object O2 of type T such that for all programs P defined in terms of T, the behavior of P is unchanged when O1 is substituted for O2 then S is a subtype of T." In very easy words, we can also say that "objects in a program should be replaceable with instances of their subtypes without altering the correctness of the program."

Example: You have an instance of the class Bike which our program uses to perform a drive action. This instance could be replaced by an instance of the class Honda, if Honda is a subclass of Bike.

Interface Segregation Principle (ISP)

ISP states, "Many client-specific interfaces are better than one general purpose interface." The heart of the principle is quite simple. If you have a class that has several clients, rather than loading the class with all the methods that the clients need, create specific interfaces for each client and multiply them into the class.

Example: Suppose we have a company management system with different type of roles like admin, HR, employee, etc. Instead of a class which can perform all the actions for all these types, we would create separate classes for each type which implement their specific actions according to our use case.

Dependency Inversion Principle (DIP)

DIP states, "Depend upon abstractions. Do not depend upon concretions." The DIP states the primary mechanism. Dependency Inversion is the strategy of depending upon interfaces or abstract functions and classes, rather than upon concrete functions and classes. This structure starts at the top and points down towards details. High-level modules depend upon lower level modules, which depend upon yet lower level modules.

Example: Imagine we have a class distributor which is able to share a blog post on different platforms like Twitter and Facebook. According to the Interface Segregation Principle, the distributor uses a composition of several instances, like a TwitterShareAction and a FacebookShareAction. Now the goal of the Dependency Inversion Principle is to not depend on concrete methods of the share action classes, like sharePostOnTwitter and sharePostOnFacebook. Instead, the Distributor class would define an interface called Sharing which is implemented by TwitterShareAction and FacebookShareAction. It declares the abstract method sharePost. The Distributor class doesn’t have to know the details of the concrete share actions. It just calls the sharePost method.

By using the SOLID Principles, code complexity is reduced by being more explicit and straightforward; readability is greatly improved; coupling is generally reduced; and your code has a better chance of cleanly evolving.