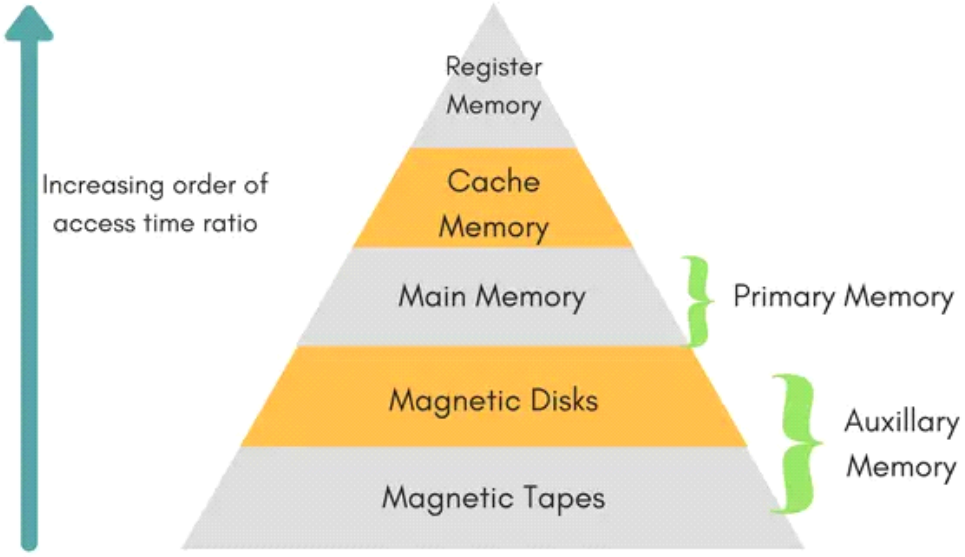
Memory Organization



Memory hierarchy refers to the organization and arrangement of different types of memory in a computer system, based on their speed, capacity, and cost. It involves multiple levels of memory, each with different characteristics, designed to optimize the overall performance of the system.

Whenever CPU wants to access any data it will first search it in the cache memory and if it finds the requires data in the cache then it will access it otherwise it will continue its search and will move on the main memory and if there also it doesn't get the desired data then it will move to the auxillary memory.

Main memory

Main memory, also known as primary memory or RAM (Random Access Memory). It is the primary storage area where the computer stores data and instructions that are actively being used by the processor. Main memory is a volatile type of memory, meaning its contents are temporary and are lost when the computer is powered off. It is different from secondary storage devices like hard disk drives or solid-state drives, which provide long-term storage but are slower.

RAM (Random Access Memory)

- RAM is a volatile memory.
- RAM allows for fast read and write operations, making it ideal for tasks that require quick access to data.
- Common types of RAM include DDR4, DDR3, and DDR2, with different generations offering varying speeds.

ROM (Read Only Memory)

- ROM is a non volatile memory.
- The contents of ROM are pre-programmed during manufacturing and cannot be modified by normal computer operations.
- Common types of ROM include PROM (Programmable Read-Only Memory), EPROM (Erasable Programmable Read-Only Memory), and EEPROM (Electrically Erasable Programmable Read-Only Memory).

There are two types of RAM : SRAM & DRAM

SRAM	***DRAM***
SRAM stands for Static Random-Access Memory. It is a type of computer memory that is used to store and retrieve data quickly. SRAM is a volatile memory, meaning it can retain data even when the power is turned off.	DRAM stands for Dynamic Random Access Memory. It is a type of computer memory that is commonly used in electronic devices such as computers, smartphones, and tablets. DRAM is a volatile memory, which means that it requires power to retain data. When the power is turned off, the data stored in DRAM is lost.
It is made up of flip flops.	It is made up of capacitors.
Examples are : cache memory.	Examples are : main memory.
It is of high speed as compared to DRAM.	It is usually slower than SRAM.
It is of high cost.	It is of low cost.

There are 3 types of ROM : PROM, EPROM, EEPROM

PROM (Programmable ROM)

It can be programmed only once by the user and once it was programmed it will never be changed.

EPROM (Erasable Programmable ROM)

It is an advanced version of PROM and as PROM can be programmed only once compared to that EPROM can be programmed multiple times. If we want to erase the program then we can do it with the help of UV-rays. In EPROM if we want to delete a program then we have to delete the entire program.

EEPROM (Electronically Erasable Programmable ROM)

It is also an advanced form of EPROM and can be programmed multiple times and if we want to erase it then we can do it electrically. In EEPROM if we want to delete a program then we can delete any specific part of the program instead of the entire program.

Difference between SRAM & DRAM

SRAM (Static RAM)	***DRAM (Dynamic RAM)***
Static RAM is built using flip-flops, which are composed of multiple transistors.	Dynamic RAM uses capacitors to store data.
SRAM does not require refreshing as it uses flip-flops to store data. It retains the data until the power supply is turned off.	DRAM needs to be refreshed periodically because the charge stored in the capacitors leaks away over time.
SRAM provides faster access times compared to DRAM.	DRAM has slower access times due to the additional overhead of refreshing cycles.
SRAM consumes more power than DRAM. It requires a continuous power supply to maintain the stored data.	DRAM consumes less power compared to SRAM. It only requires power during read and write operations, and the refreshing process consumes additional power.
SRAM is more expensive to manufacture than DRAM due to its higher complexity.	DRAM is more cost-effective to produce due to its higher density.

Associative Memory

Associative memory is a type of computer memory that allows the retrieval of information based on its content rather than its specific address. In simple terms, it's like having a memory that can recall things based on their characteristics rather than their exact location.

Imagine you have a drawer full of socks. Each sock has a different color and pattern. Now, let's say you want to find a specific pair of socks with a specific pattern, but you don't remember where you put them in the drawer. If you had an associative memory, you could simply think about the pattern you are looking for, and the memory would automatically retrieve the socks that match that pattern. It would search through the entire drawer and pull out the socks that meet your criteria.

In computer systems, associative memory works in a similar way. It allows the computer to search for data based on its content rather than its physical address. This can be useful in tasks such as searching through a database for specific information or recognizing patterns in data.

Locality of reference

Locality of reference in cache memory refers to the observation that programs tend to access a relatively small portion of the available memory at any given time. It suggests that if a specific memory location is accessed once, it is likely to be accessed again in the near future.

To understand locality of reference, let's consider an analogy of a library. Imagine you are studying for an exam and you have a small table with limited space to keep books. Instead of going to the library every time you need a book, you decide to keep a few frequently used books on your table.

In this analogy, the books on your table represent the cache memory, and the library represents the main memory. The limited space on your table corresponds to the limited capacity of the cache memory.

Now, as you study, you tend to refer to the same books repeatedly because they contain the most relevant information for your exam. You rarely need to go to the library to fetch new books because the ones on your table are sufficient for your immediate needs. This behavior of frequently accessing a small set of books represents the principle of locality of reference.

Temporal Locality

Temporal locality refers to the tendency of a program to access the same data or instructions repeatedly over a short period of time. In simpler terms, if a program accesses a particular piece of data, it is likely to access that same data again in the near future. This can be due to loops or repeated instructions in the program. Temporal locality is based on the idea that recently accessed data is likely to be accessed again soon. To improve performance, computer systems can take advantage of temporal locality by keeping recently accessed data in a cache, a faster and smaller memory, so that it can be accessed quickly when needed again.

Spatial Locality

Spatial locality, on the other hand, refers to the tendency of a program to access data or instructions that are located close to each other in memory. In other words, if a program accesses a specific memory location, it is likely to access nearby memory locations as well. This can be due to sequential or contiguous memory accesses in the program. Spatial locality is based on the idea that nearby data is likely to be accessed in the near future. Computer systems can exploit spatial locality by using techniques like prefetching, where they fetch data from memory that is likely to be needed soon and store it in cache, thereby reducing the time taken to access that data.

Cache Memory

Whenever CPU needs to access or use some particular data for a continuous period of time at that time the data is placed in cache memory as if cache memory is comparatively faster than the main memory and also to overcome the speed mismatch of data transfer between main memory to CPU.

It is a small size memory compared to the main memory and also very much faster than main memory. It is located between the CPU and the main memory and is a volatile memory that stores data temporarily until the power is on.

Data transfer between main memory to cache memory happens in the form of blocks.

Data transfer between CPU to cache memory happens in the form of words.

Cache Hit

If the required word or data that CPU needs to access finds in the cache memory then it is called cache hit..

Cache Miss

If the required data that the CPU needs to access doesn't finds in the cache memory then it is called cache miss.

There are levels of cache memory which are as follows :

L1 Cache

It is the fastest level cache as if it comes attached with the processor and its size ranges from 1KB to 64KB. It is a very high speed memory becuase it's made up of SRAM. It is also known as internal cache or primary cache.

L2 Cache

Level 2 cache is larger then L1 cache and also slower than L1 cache. It stores the most recently used or more frequently used data and is also known as secondary cache. It is designed to reduce time in where data is already has accessed previously. It's size ranges from 64KB to 4MB. It is located between L1 cache and RAM.

L3 Cache

It is an enhanced form of L2 cache and is located in the motherboard and is comparatively slower than the L1 and L2 cache as if L1 and L2 cache is within the CPU. It is faster then the main memory and is slower than the L2 cache. it's size ranges from 3MB or more than that.

L1 is the fastest among all the cache levels after that comes L2 and L3 cache.

L1 and L2 is located in the processor while L3 is located in the motherboard.

Cache Mapping

Cache mapping is a technique in which the contents of the main memory are brought to the cache memory. The transfer of data from main memory to cache memory is a mapping process.

And when we brought the data of the secondary memory to main memory then the process is called paging.

Associative mapping

It is the fastest and most flexible mapping technique. It refers to a technique that allows the computer to quickly locate data stored in memory by using a key or identifier.

To explain it simply, think of a large library with thousands of books. If all the books were randomly placed on the shelves, it would take a lot of time and effort to find a specific book. However, if the library uses an associative mapping system, each book is assigned a unique identifier or key, such as a call number or barcode.

When you want to find a particular book, you don't need to search through every shelf. Instead, you use the key, which is associated with a specific location in the library. This allows you to directly access the book you're looking for without wasting time searching through all the shelves.

Similarly, in computer memory management, associative mapping uses keys or addresses to quickly locate data stored in memory. Instead of searching through the entire memory

Direct mapping

In direct mapping, the cache memory is divided into equal-sized sections called cache lines. Each cache line has a specific address range where data from the main memory can be stored. When the CPU needs to access data, it checks if the data is present in

the cache memory by comparing the memory address with the address ranges of the cache lines.

Here's how direct mapping works: When a CPU wants to read or write data from/to the main memory, it first checks if the data is already present in the cache. It does this by comparing the memory address of the data with the address range of the cache lines. If the data is found in the cache, it's called a cache hit, and the CPU can quickly retrieve or update the data.

However, if the data is not present in the cache, it's called a cache miss. In this case, the CPU needs to fetch the data from the main memory and store it in the cache for future use.

Direct mapping determines the location in the cache where the data should be stored based on its memory address. It uses a mathematical function, typically a modulo operation, to calculate the cache line where the data should be placed.

Set associative mapping

Set associative mapping is a technique used in computer systems to map memory addresses to specific locations in the cache, which is a small, fast memory used to store frequently accessed data. In simple terms, it's like finding a parking spot for your car in a multi-level parking garage.

Imagine you have a large parking garage with many floors and each floor has several parking spots. Each parking spot can be considered as a location in the cache memory. Now, when you want to park your car, you first need to find an available spot. Set associative mapping helps with this process.

Virtual Memory

Virtual memory is a computer's way of pretending that it has more memory than it actually does. It's like having a magical extra storage space that the computer can use when it runs out of physical memory (RAM).

When you open programs or run tasks on your computer, they need memory to store and process data. However, if your computer's physical memory is limited and gets filled up, virtual memory comes to the rescue.

Virtual memory uses a portion of your computer's hard drive as a temporary storage area. When the physical memory is full, the computer moves some of the less-used data from RAM to the virtual memory space on the hard drive. This frees up space in the RAM for new data and programs.

Need of Virtual Memory

The need for virtual memory arises from the fact that computers have limited physical memory (RAM) available, but they often need to run multiple programs or tasks at the same time.

Imagine you have a small desk with limited space to work on. You can only have a few things on your desk at once. Now, what if you have more work to do than can fit on your desk? You need a way to temporarily store some of the things you're not using at the moment, but still want to keep them close by.

Virtual memory serves a similar purpose for computers. It allows them to handle more tasks and programs than they could with just the physical memory. When the computer runs out of space in the RAM, virtual memory provides additional storage by using a portion of the hard drive.

Importance of Virtual Memory

The importance of virtual memory can be understood in a simple way. Here are a few key reasons why virtual memory is important:

- Virtual memory allows your computer to run multiple programs and tasks simultaneously, even if the physical memory is limited. It provides a way to efficiently manage and juggle the memory requirements of different programs. Without virtual memory, you would be restricted to running only a few programs at a time.

- Virtual memory expands the effective memory capacity of a computer. It uses a portion of the hard drive as temporary storage, effectively creating an illusion of having more memory than the physical RAM. This enables you to work with larger files and run memory-intensive programs that would otherwise exceed the available physical memory.

Swapping Process in Virtual Memory
In virtual memory, the swapping process refers to the movement of data between the physical memory (RAM) and the virtual memory space on the hard drive. It is a mechanism used by the computer to manage the limited physical memory efficiently.

Imagine you have a desk with limited space, and you need to work on multiple projects. However, your desk can only accommodate a few projects at a time. So, you decide to keep some projects in a storage box under the desk and swap them as needed.

Similarly, in the swapping process, when your computer's physical memory gets filled up with data from running programs, the operating system selects some less-used data and transfers it from the RAM to the virtual memory space on the hard drive. This frees up space in the RAM for new data.

Virtual Address

Virtual addresses are a way for programs to access and manage memory without worrying about the physical location of data. They provide an abstraction layer and enable features like virtual memory, allowing computers to efficiently use their memory resources.

The advantage of virtual addresses is that they provide a layer of abstraction. Programs running on your computer can use virtual addresses to access data without worrying about the underlying physical memory layout. This abstraction makes it easier for multiple programs to run simultaneously and share memory resources without interfering with each other.

Address Mapping

Address mapping is a fundamental process that ensures the correct translation and retrieval of data between virtual addresses used by programs and the physical addresses where the data resides in memory.

Address Mapping using Pages
Paging is one common technique used for address mapping. Address mapping using paging involves dividing memory into fixed-size pages and translating logical addresses to physical addresses using a page table. This enables efficient memory management and allows programs to use more memory than is physically available.

By using paging, the operating system can efficiently manage memory by dividing it into smaller, manageable units. It allows programs to use a logical address space that is larger than the physical memory available, as only the necessary pages are loaded into memory at any given time.

Page Fault

When the CPU needs to access a particular page in the main memory but the page is not present there then it is called page fault.

Page Replacement Algorithm

Imagine you have a bookshelf (representing the memory) with limited space to store books (representing pages). Now, let's say you want to add a new book to the shelf, but there is no space available. What do you do? You have to remove one of the existing books from the shelf to make room for the new book. This process of selecting which book to remove is similar to how a page replacement algorithm works.

The goal of a page replacement algorithm is to minimize the number of times data needs to be retrieved from the slower secondary storage (like the hard disk) and brought into the faster main memory. When a program requests a page that is not currently in memory, the algorithm determines which page to remove from the memory to make space for the new page.

First-In-First-Out (FIFO)
This algorithm treats the memory as a queue. When a page needs to be replaced, it evicts the page that was brought into memory first. It's like the first book you put on your bookshelf is the first one you take out if you need to make space.

Least Recently Used (LRU)
This algorithm assumes that pages that have not been used for a long time are less likely to be needed in the near future. It

keeps track of the order in which pages are accessed and selects the least recently used page for replacement. It mimics the idea that if you haven't looked at a book on your bookshelf for a while, you're more likely to remove it to make space for a new book.

Optimal
The optimal algorithm is theoretical and not practical in most cases. It assumes perfect knowledge of future page requests. It replaces the page that will not be needed for the longest duration in the future.

Memory Interleaving
Memory interleaving is a technique used in computer systems to enhance the overall performance of memory access. Imagine you have a computer with multiple memory modules (chips) installed, and each module can store a certain amount of data. Instead of treating each memory module independently, memory interleaving allows the computer to combine and treat them as a single, larger memory space.

memory interleaving is a technique that combines and organizes data from multiple memory modules in a specific pattern to improve memory access speed and system performance.

Memory Modules

Memory modules, in simple terms, are small electronic components that store data in a computer. They are like the "storage units" of a computer's memory system. Think of them as small boxes where the computer keeps information for quick access.

Imagine you have a desk with drawers. Each drawer can hold different items like pens, papers, and sticky notes. Similarly, a computer has memory modules that can hold and retrieve information.

Memory modules are typically rectangular in shape and fit into slots on the computer's motherboard. They come in different sizes and capacities, such as 4 GB, 8 GB, or even 16 GB. The size determines how much data can be stored in the module.

Low Order Memory Interleaving
In low order memory interleaving, the memory is divided into smaller sections called banks. Each bank contains a portion of the memory addresses. When a memory access occurs, it is distributed across these banks in a round-robin fashion. This means that consecutive memory accesses are directed to different banks, reducing the chances of accessing the same bank repeatedly. It helps to distribute the memory traffic and reduces conflicts, improving overall memory performance.

High Order Memory Interleaving
High order memory interleaving takes the concept of interleaving a step further. Instead of dividing memory into banks, it divides it into larger sections called memory modules. Each memory module contains a range of memory addresses. When a memory access occurs, it is distributed across these modules in a round-robin manner. This technique is beneficial in systems where multiple memory modules are used. High order memory interleaving helps balance the load and maximize the utilization of memory modules.

Cache Coherence

Cache coherence refers to the consistency of data stored in different caches that are part of a computer system. In simpler terms, it ensures that multiple copies of the same data in different caches are kept up to date and synchronized.

Imagine you have a big family, and everyone has their own personal notebooks to write down important information. Now, let's say everyone in the family has a copy of a grocery list, and they can add or remove items from it whenever they want. However, there's a catch: everyone's notebook is independent, and they don't communicate directly with each other.

Now, let's suppose that one family member wants to add a new item to the grocery list. They write it down in their notebook, but the other family members still have the old version of the list in their notebooks. This creates a problem because different family members have different versions of the grocery list, and it becomes challenging to keep track of what items need to be bought.

Cache coherence is crucial because it prevents data inconsistencies and ensures that all parts of the computer system operate on the most recent and accurate data. Without cache coherence, different components of a system might have different versions of the same data, leading to incorrect results and unexpected behavior.

I/O Organization

I/O Interface

In simple terms, an I/O (Input/Output) interface is a connection or set of connections that allows different devices or systems to communicate with each other and exchange information.

Think of it as a bridge between two entities. One entity could be a computer or a device that needs to send or receive data, and the other entity could be another device, such as a printer, a keyboard, a monitor, or even another computer.

The I/O interface provides a standardized way for these devices to interact, regardless of their specific designs or technologies. It defines the rules and protocols for how data is transmitted, received, and interpreted.

For example, let's consider a USB (Universal Serial Bus) interface. It's a common I/O interface that you may be familiar with. It allows you to connect various devices like keyboards, mice, printers, and external storage drives to your computer.

When you plug in a USB device, the I/O interface enables communication between the computer and the device. It ensures that the computer can recognize the device, send and receive data to/from it, and perform actions based on the received data.

Peripheral Devices

Peripheral devices are like the helpers of a computer. They are external devices that connect to your computer and provide additional functionality or input/output options. Imagine your computer as the main brain, and these peripherals as its hands, eyes, and ears.

Here are some common examples of peripheral devices:

- **Keyboard:** It allows you to input text, numbers, and commands by pressing the keys.
- **Mouse:** It helps you move the cursor on the screen and select items by clicking buttons.
- **Monitor:** It displays the visual output of your computer, like text, images, and videos.
- **Printer:** It lets you transfer digital documents or images onto paper.
- **Scanner:** It helps convert physical documents or images into digital files that can be stored on your computer.
- **Speakers:** They produce sound and allow you to listen to music, videos, or any audio output from your computer.
- **Webcam:** It captures video and enables video conferencing, live streaming, or taking pictures.
- **External Hard Drive:** It provides extra storage space for saving files, documents, photos, and videos.
- **USB Flash Drive:** It is a small portable storage device that you can use to transfer or store files.
- **Headphones:** They let you listen to audio privately without disturbing others.

Difference Between Peripheral and CPU & Memory

Peripheral	***CPU & Memory***
Peripherals refer to external devices or components connected to a computer system that provide input, output, or storage capabilities. They expand the functionality of the computer and enable users to	The CPU is the primary component responsible for executing instructions and performing calculations in a computer system. It carries out the actual processing tasks, such as running software programs and performing

interact with the system. Examples of peripherals include keyboards, mice, monitors, printers, scanners, and external storage devices.	mathematical operations. Memory, on the other hand, refers to the storage component of a computer that holds data and instructions required by the CPU to perform tasks efficiently.
Peripherals are external devices physically connected to the computer system. They are typically located outside the main computer case and connected via ports or cables.	The CPU and memory are internal components of the computer system. The CPU is usually a chip (microprocessor) mounted on the motherboard, while memory modules are also attached to the motherboard or installed in slots.
Peripherals provide input or output capabilities to the computer system. For example, a keyboard and mouse provide input to control the computer, while a monitor or printer displays or produces output. Peripherals do not directly participate in the actual processing of data and instructions.	The CPU is responsible for executing instructions and performing calculations. It fetches data and instructions from memory, performs operations on the data, and stores results back in memory. The memory holds the data and instructions required by the CPU during processing. Both the CPU and memory actively participate in data processing.

In summary, peripherals are external devices that expand the capabilities of a computer system by providing input, output, or storage functions. On the other hand, the CPU and memory are internal components responsible for executing instructions, performing calculations, and storing data required for processing tasks.

Interface Unit & I/O Bus

In simple terms, the interface unit and I/O (Input/Output) bus are components of a computer system that help in the communication between the CPU and peripherals or external devices.

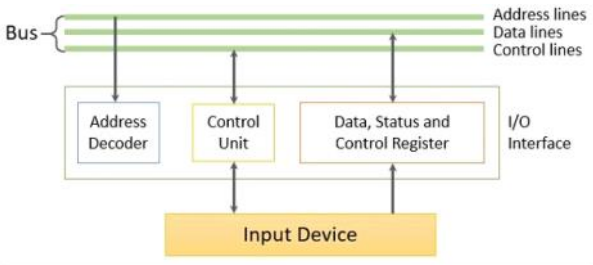
The interface unit, also known as the I/O interface or I/O controller, acts as a bridge between the CPU and peripherals. Its main purpose is to facilitate communication and data transfer between the CPU and external devices. The interface unit translates the data and control signals from the CPU into a format that can be understood by the peripherals, and vice versa.

Think of the interface unit as a translator or mediator. When you type on a keyboard or click a mouse, the interface unit takes the signals from these peripherals and converts them into a format that the CPU can understand. Similarly, when the CPU needs to send data to a printer or display information on a monitor, the interface unit takes the data from the CPU and converts it into a format that the peripheral can process.

The I/O bus, also referred to as the I/O channel or expansion bus, is a communication pathway or set of wires that connects the CPU to the peripherals. It provides a means for the CPU to exchange data and control signals with the external devices.

Imagine the I/O bus as a highway that connects different parts of a city. In this case, the city represents the computer system, and the different parts represent the CPU and peripherals. The I/O bus acts as a pathway that allows data and control signals to travel between the CPU and the peripherals efficiently.

In summary, the interface unit acts as a translator between the CPU and peripherals, converting signals and data into formats that each can understand. The I/O bus provides a communication pathway, like a highway, connecting the CPU and peripherals, allowing for efficient data transfer between them.



Bus Structure for I/O Interface of an Input Device

I/O Commands

I/O commands, or Input/Output commands, refer to instructions used in programming languages to interact with devices or handle input and output operations. These commands allow programs to communicate with users, read data from files, write data to files, and perform other input/output tasks.

Here are some common types of I/O commands:

1. **Read:** This command is used to retrieve data or input from a user or a file. It allows programs to capture information from various sources. For example, if you want to read a user's name from the keyboard or read a line of text from a file, you would use a read command.

2. **Write/Print:** This command is used to display or output data to the user or to a file. It allows programs to present information or results. For instance, if you want to display a message on the screen or write a line of text to a file, you would use a write or print command.

3. **Open:** This command is used to establish a connection or access to a file. It allows programs to prepare a file for reading or writing operations. When you want to work with a file, you would typically start by opening it using an open command.

4. **Close:** This command is used to terminate the connection or access to a file. It allows programs to release the resources associated with a file after they are done with it. When you finish working with a file, you would close it using a close command.

I/O Mapping

I/O mapping, also known as Input/Output mapping, refers to the process of connecting and managing input and output devices in a computer system. It involves assigning specific addresses or memory locations to these devices so that the computer can communicate with them effectively.

In simple terms, think of your computer as a big office with different departments and employees. Each employee has a unique address or desk where they work. Similarly, devices connected to a computer, such as keyboards, mice, printers, and disk drives, need specific addresses so that the computer knows where to send and receive data from them.

Uses of I/O Mapping

I/O mapping is used in various scenarios where a computer system needs to communicate with external devices. Here are a few examples to help explain its usage in a simple way :

- **Keyboard and Mouse:** When you type on a keyboard or move a mouse, your actions need to be translated into digital signals that the computer can understand. I/O mapping is used to connect these input devices to the computer so that it can receive the keystrokes or mouse movements and react accordingly.

- **Display and Printer:** When you want to see something on your computer screen or print a document, the computer needs to send the appropriate data to the display or printer. I/O mapping helps establish a connection between the computer and these output devices so that it can send the correct information to be displayed or printed.

- **External Storage:** Devices like hard drives, USB flash drives, and memory cards are used to store data externally. I/O mapping allows the computer to access and transfer data to and from these storage devices. This enables you to save files, retrieve information, or install software on your computer.

- **Networking:** When you connect your computer to a network, I/O mapping helps facilitate communication between your computer and other devices on the network. It enables data transfer over network cables or wireless connections, allowing you to browse the internet, send emails, or share files with other computers.

In summary, I/O mapping is used whenever a computer system needs to interact with external devices, such as input devices

(keyboard, mouse), output devices (display, printer), storage devices (hard drives, USBs), or network devices (routers, modems). It ensures that data can be properly sent to and received from these devices, enabling seamless communication and functionality between the computer and the external world.