UNIT 3 MEMORY AND I/O SYSTEMS

MEMORY HIERARCHY:

- **Memory Hierarchy** refers to the arrangement of different types of computer memory in a system, organized in a hierarchical manner based on their access times, capacities and costs.
- The primary goal of memory hierarchy is to bridge the speed gap between the fast but expensive, and the slow but inexpensive storage technologies.
- This arrangement ensures that data can be efficiently accessed by the CPU and other processing units based on their immediate needs.
- The memory hierarchy levels are CPU registers, cache memory, main memory or primary memory, magnetic disks or secondary memory, Optical disks and magnetic tapes.

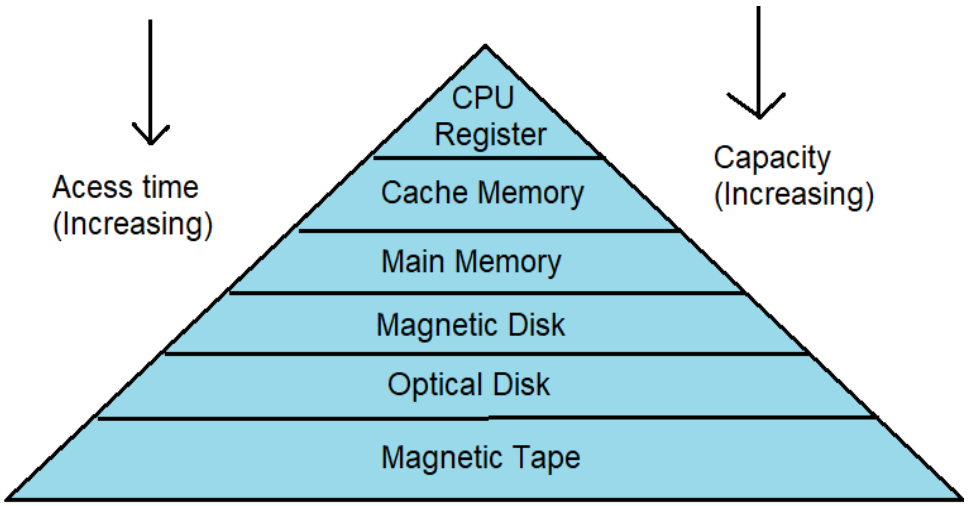


Fig:- Memory Hierarchy

This Memory Hierarchy Design is divided into 2 types:

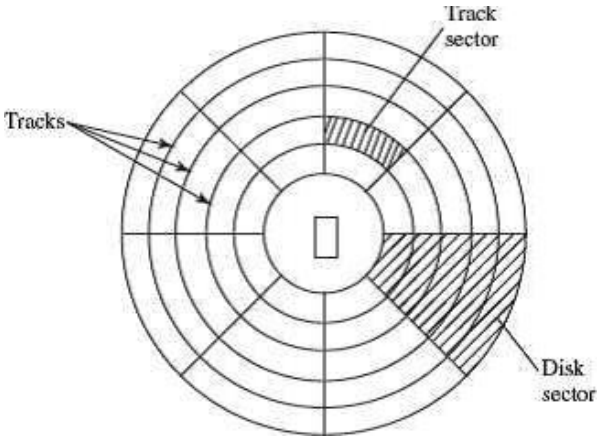
Primary or internal memory: It consists of CPU registers, Cache Memory, Main Memory, and these are directly accessible by the processor.

Secondary or external memory: It consists of a Magnetic Disk, Optical Disk, Magnetic Tape, which are accessible by processor via I/O Module.

1. **Registers** are the fastest type of memory and are located inside the CPU. They are used to store the most frequently used data and instructions.
2. **Cache memory** is located close to the CPU and is much faster than main memory. It is used to store copies of data that have been recently accessed from main memory.
3. **Main memory** is the largest type of memory in a computer system. It is slower than cache memory, but it is much larger.

4. Magnetic Disk:

- Magnetic Disk is a type of storage device that uses magnetic recording to store and retrieve digital data. It is also commonly known as a hard disk drive (HDD) or simply a hard drive.
- These are made up of one or more platters that are coated with a magnetic material. The platters are divided into tracks and sectors, and each sector can store a certain amount of data. They are used as a secondary storage device in computers.



5. Optical Disk:

- The optical disk storage system includes a rotating disk coated with a diminished layer of metal that facilitates a reflective surface and a laser beam, which is used as a read/write head for recording information onto the disk.
- Optical disks are used as a secondary storage device in computers, and they are also used for storing music, movies, and other multimedia data.

Optical Disk

Magnetic Tape

6. Magnetic Tape:

- It is a type of storage device that uses a magnetic field to store data. Magnetic tapes are made up of a thin plastic ribbon that is coated with a magnetic material.
- The data is stored in linear fashion on tape, and each location on tape can store a certain amount of data. Data read/write speed is slower because of sequential access.
- They are used as secondary storage device in computers & also used for data backup.

Advantages of Memory Hierarchy:

1. **Access Time:** The access time in the memory hierarchy of a computer system is the interval of the time among the data availability and request to read or write.
2. **Faster access:** Due to multiple levels in memory hierarchy, users get faster access to frequently used data. Cache memory is fastest memory, stores frequently used data.
3. **Capacity:** It is the amount of information that can be stored. The capacity increases as we move from top to bottom in the hierarchy.
4. **Increased processing speed:** By memory hierarchy, operations can be performed much faster as the fastest memory can be used for the most frequently used data. The CPU can access the data in a faster manner, which in turn, increases the processing speed.
5. **Performance:** In old times, designing a computer system was done without memory hierarchy. The gap of speed between the main memory and the CPU registers is enhanced because of the huge inconsistency in access time, which will cause the lower performance of the system. So, the enhancement in the memory was mandatory. This was designed in the memory hierarchy model due to the system's performance increase.

Disadvantages of Memory Hierarchy:

1. **Increased cost:** The memory hierarchy can increase the cost of the computer system by requiring multiple types of memory.
2. **Increased complexity:** The memory hierarchy can increase the complexity of the computer system by requiring more complex hardware and software.
3. **Reduced flexibility:** The memory hierarchy can reduce the flexibility of the computer system by making it more difficult to add or remove memory.

MEMORY TECHNOLOGIES:

It refers to the different ways that data can be stored and accessed in a computer system. There are two main types: **volatile** and **non-volatile**.

Volatile memory is memory that loses its contents when the power is turned off. This type of memory is typically used for storing data that is actively being used by the computer, such as the operating system, programs, and data files. **Volatile memory technologies include:**

1. **Dynamic Random-Access Memory (DRAM)** is the most common type of volatile memory. DRAM is relatively inexpensive and has a high access speed. However, DRAM requires constant refreshing to maintain its contents.
2. **Static Random-Access Memory (SRAM)** is a type of volatile memory that is faster than DRAM, but it is also more expensive. SRAM does not require refreshing, which makes it more reliable than DRAM.

Non-volatile memory is memory that retains its contents when power is turned off. This type of memory is typically used for storing data that needs to be preserved even when the computer is turned off, such as configuration settings, user data, and program files. **Non-volatile memory technologies include:**

1. **Read-only memory (ROM)** is a type of non-volatile memory that can only be read. ROM is typically used to store the computer's BIOS, which is the software that initializes the computer when it is turned on.
2. **Programmable read-only memory (PROM)** is a type of non-volatile memory that can be programmed once. It is typically used to store configuration settings for computer.
3. **Erasable programmable read-only memory (EPROM)** is a type of non-volatile memory that can be erased and programmed multiple times. EPROM is typically used to store firmware updates for the computer.
4. **Electrically erasable programmable read-only memory (EEPROM)** is a type of non-volatile memory that can be erased and programmed electrically. It is typically used to store configuration settings for devices such as printers and network cards.

CACHE BASICS: MEASURING AND IMPROVING CACHE PERFORMANCE

Cache is a small, fast memory component located closer to the processor in a computer system. Its purpose is to store frequently accessed data and instructions, allowing for faster access compared to retrieving the information from the main memory.

CACHE BASICS:

1. **Cache Hierarchy:** Modern computer systems typically have multiple levels of cache, such as L1, L2, and sometimes L3 caches.
    - **Level 1 (L1) cache:** This is the fastest and smallest level of cache. It is typically located on the CPU chip itself.
    - **Level 2 (L2) cache:** This is the slower than L1 cache and larger than L1 cache. It is typically located on the CPU chip or on a separate chip close to the CPU.
    - **Level 3 (L3) cache:** This is the slowest and largest level of cache. It is typically located on a separate chip close to the CPU.
2. **Cache Lines:**
    - Cache memory is divided into fixed-size blocks called cache lines. Each cache line stores a small portion of the main memory's data.
    - When the processor requests data, the cache controller checks if the data is present in the cache. If it is present, cache hit occurs, and the data is fetched from the cache. Otherwise, cache miss occurs, and the data must be retrieved from the main memory.

MEASURING CACHE PERFORMANCE:

1. **Cache Hit Rate:**

   The **cache hit rate** indicates the percentage of memory accesses that result in cache hits.

   It is calculated by dividing number of cache hits by the total number of memory accesses.

   A higher cache hit rate suggests better cache performance.

   The formula for cache miss rate is:

   Cache hit rate = (Number of cache hits) / (No. of cache hits + No. of cache misses)

   The cache hit rate can be expressed as a percentage by multiplying the result by 100.

   Here is an example of how to calculate the cache hit rate:

   If a cache has 100 cache hits and 20 cache misses, then cache hit rate is 83%.

   Number of cache hits = 100

   Number of cache misses = 20

   Cache hit rate = (100) / (100 + 20) = 0.83 = 83%

2. **Cache Miss Rate:**

   The cache miss rate is the opposite of the cache hit rate and represents the percentage of memory accesses that result in cache misses.

   It is calculated by dividing number of cache misses by total number of memory accesses.

   A lower cache miss rate indicates better cache performance.

   The formula for cache miss rate is:

   Cache miss rate = (No. of cache misses) / (No. of cache hits + No. of cache misses)

   The cache miss rate can be expressed as a percentage by multiplying the result by 100.

   Here is an example of how to calculate the cache miss rate:

   If cache has 100 cache hits and 20 cache misses, then cache miss rate is 16%.

   Number of cache hits = 100

   Number of cache misses = 20

   Cache miss rate = (20) / (100 + 20) = 0.16 = 16%

3. **Cache Hit Time and Cache Miss Penalty:**

   Cache Hit Time refers to the time required to access data from the cache.

   Cache Miss Penalty is the time required to retrieve data from the main memory in case of a cache miss.

   Lower hit time and miss penalty contribute to better cache performance.

IMPROVING CACHE PERFORMANCE:

1. **Increasing the cache size:** This will allow the cache to store more data, which can lead to a higher hit ratio and a lower miss rate.
2. **Using a higher associativity:** This will allow the cache to store multiple copies of the same data, which can also lead to a higher hit ratio.
3. **Using a good cache replacement policy:** This will help to minimize the number of misses when the cache is full.
4. **Using prefetching:** This technique can be used to bring data into the cache before it is actually needed. This can help to reduce the number of misses.
5. **Using cache partitioning:** This technique can be used to divide the cache into multiple partitions, each of which can be used for a different type of data. This can help to improve the hit ratio for each type of data.
6. **Using cache hierarchies:** It uses multiple levels of cache, with each level being smaller and faster than the level above it. It helps to improve the overall performance of the cache.

VIRTUAL MEMORY:

- **Virtual memory** is the feature of an operating system (OS). It is responsible for memory management. In the Virtual Memory, the Physical Memory (Hard Disk) will be treated as the Logical Memory (random access memory (RAM)).
- With the help of virtual memory, we can temporarily increase the size of Logical Memory as from the Physical Memory. In this, all the space of Hard Disk can be used as the Logical Memory So that a user can execute any Number of programs.
- **Virtual memory** is memory management technique that allows computer to have much larger address space than the amount of physical memory that is actually available.
- Techniques to manage virtual memory: Paging and Segmentation.
- **Paging** is a memory management technique in virtual memory where the memory address space of a process is divided into fixed-size blocks called "**pages**." The physical memory (RAM) is also divided into blocks of the same size called "**frames**." Paging allows for efficient memory management by mapping virtual pages to physical frames, enabling the system to load only the required pages into RAM and retrieve others from disk when needed.
- **Segmentation** is another memory management technique in virtual memory where the memory address space of a process is divided into variable-sized logical units called "**segments**." Each segment represents a distinct part of the program, such as code, data, or stack.

Advantages of Virtual Memory:

1. **Increased Memory Capacity**: Virtual memory allows a computer to use more memory than it physically has available. This means that programs can run even if there is not enough physical memory available.
2. **Improved Performance**: Virtual memory can help to improve performance by reducing the amount of time it takes to access data. This is achieved by using a combination of physical and virtual memory to store and access data.
3. **Simplified Memory Management**: Virtual memory simplifies memory management by allowing the computer to manage memory automatically. This eliminates the need for users to manage memory manually, which can be complex and time-consuming.
4. **Increased Compatibility**: Virtual memory makes it easier for software developers to create programs that are compatible with a wide range of hardware configurations. This is because virtual memory provides a standardized way of managing memory.
5. **Enhanced Multitasking**: Virtual memory allows a computer to run multiple programs at the same time without running out of memory. This is because virtual memory provides each program with its own virtual memory space, allowing them to run independently of each other.

Disadvantages of Virtual Memory:

1. **Slower Performance**: While virtual memory can improve performance in some cases, it can also slow down a computer if it is overused. This can happen if the computer does not have enough physical memory available, and the system is constantly swapping data between physical and virtual memory.
2. **Increased Complexity**: Virtual memory can add complexity to a computer system, making it more difficult to manage.
3. **Risk of Data Loss**: Virtual memory can increase the risk of data loss if the system crashes or loses power. This is because data that is stored in virtual memory may not be saved to disk until later, and can be lost if the system fails before it is saved.
4. **Dependence on Hard Disk**: Virtual memory relies heavily on the hard disk to function properly. This means that any issues with the hard disk can have a significant impact on system performance.
5. **Fragmentation**: Virtual memory can lead to fragmentation, which is when data is split up into small pieces and scattered across the hard disk. This can slow down the computer, as the system has to spend more time searching for data that is stored in multiple locations.

TLB (TRANSLATION LOOKASIDE BUFFER):

- **Translation Lookaside Buffer (TLB)** is a small, fast memory that stores the recent translations of virtual memory to physical memory.
- It is used to reduce the time taken to access a user memory location.
- **TLB** is a cache of the page table, which is a data structure that maps virtual addresses to physical addresses.
- The **Page Table** is typically stored in main memory, but the **TLB** is stored in the CPU's memory. This means that the TLB can be accessed much faster than the page table.
- When a program references a virtual address, the TLB is checked first to see if the translation is already stored in the TLB.
  - ✓ If the translation is found in the TLB, the physical address can be quickly retrieved.
  - ✓ If the translation is not found in the TLB, a page fault occurs and the operating system must retrieve the translation from the page table.

TLB includes following terminologies:

1. **TLB size:** The size of the TLB is typically a few hundred entries. This means that the TLB can only store a small subset of the page table entries.
2. **TLB replacement policy:** The TLB replacement policy is the algorithm that is used to decide which entries should be removed from the TLB when it is full. The most common TLB replacement policy is the least recently used (LRU) policy.
3. **TLB hit:** A TLB hit occurs when the translation for a virtual address is found in the TLB.
4. **TLB miss:** A TLB miss occurs when a translation for virtual address is not found in TLB.
5. **TLB miss penalty:** The TLB miss penalty is the time it takes to access the page table and retrieve the translation for a virtual address.

Benefits of using TLBs:

1. **Reduced memory access time:** TLBs can significantly reduce the time it takes to access memory by providing a fast path to the page table.
2. **Improved performance:** TLBs can improve the performance of virtual memory systems by reducing the number of page faults.
3. **Reduced CPU overhead:** TLBs can reduce the CPU overhead associated with memory management by reducing the number of times the CPU has to access the page table.

I/O SYSTEMS:

- Input/output (I/O) systems are responsible for transferring data between the computer's memory and peripheral devices, such as disk drives, printers, and network adapters. I/O systems are typically implemented in hardware, but they can also be implemented in software.
- I/O systems are an essential part of any computer system, as they allow the computer to interact with the outside world. I/O systems can have a significant impact on the performance of a computer system, so it is important to choose an I/O system that is well-designed and efficient.

The I/O system typically consists of three main components:

- **I/O Controller:** The I/O controller is a hardware device that is responsible for communicating with the peripheral device. The I/O controller typically has a number of registers that are used to control the peripheral device.
- **I/O Driver:** The I/O driver is a software module that is responsible for managing the I/O controller and transferring data between the memory and the peripheral device. The I/O driver typically has a number of functions that are used to read and write data to the peripheral device.
- **I/O Scheduler:** The I/O scheduler is a software module that is responsible for scheduling I/O requests and ensuring that the I/O system is efficient. The I/O scheduler typically uses a queue to store I/O requests. The I/O scheduler then decides which I/O request should be processed next.

Benefits of I/O systems:

1. **Improved performance:** I/O systems can improve the performance of a computer system by offloading the I/O operations from the CPU.
2. **Reduced CPU overhead:** I/O systems can reduce the CPU overhead associated with I/O operations by allowing the CPU to focus on other tasks.
3. **Improved reliability:** I/O systems can improve the reliability of a computer system by providing a way to handle errors that occur during I/O operations.

Challenges of I/O systems:

- **I/O Latency:** I/O operations can be slow, which can impact the overall performance of a computer system.
- **I/O Contention:** When multiple I/O devices are competing for access to the same I/O bus, it can lead to performance degradation.
- **I/O Errors:** I/O errors can occur, which can corrupt data or cause the system to crash.

PROGRAMMED INPUT OUTPUT:

- **Programmed input/output (PIO)** is a technique for transferring data between the computer's memory and peripheral devices. In this technique, the CPU is responsible for initiating and controlling all I/O operations.
- **The CPU typically uses a series of instructions to perform an I/O operation.**
  - ✓ The **first instruction** typically specifies the I/O device that will be accessed.
  - ✓ The **next instruction** typically specifies the operation that will be performed, such as reading or writing data.
  - ✓ The **final instruction** typically specifies the address of the memory location where the data will be stored or retrieved.
- The CPU then polls the I/O device to see if the operation has completed.
- When the operation has completed, the I/O device will typically signal the CPU by setting a status bit. The CPU can then check the status bit to determine if the operation has completed.
- Programmed I/O is a simple and straightforward way to perform I/O operations. However, it can be inefficient, as the CPU is often idle while the I/O device is performing the operation.

Advantages of Programmed I/O:

1. **Simple:** PIO is a simple and straightforward technique for performing I/O operations.
2. **Direct:** The CPU has direct control over the I/O devices, which can be helpful for debugging and troubleshooting.
3. **Flexible:** PIO can be used to transfer data to and from any I/O device.

Disadvantages of Programmed I/O:

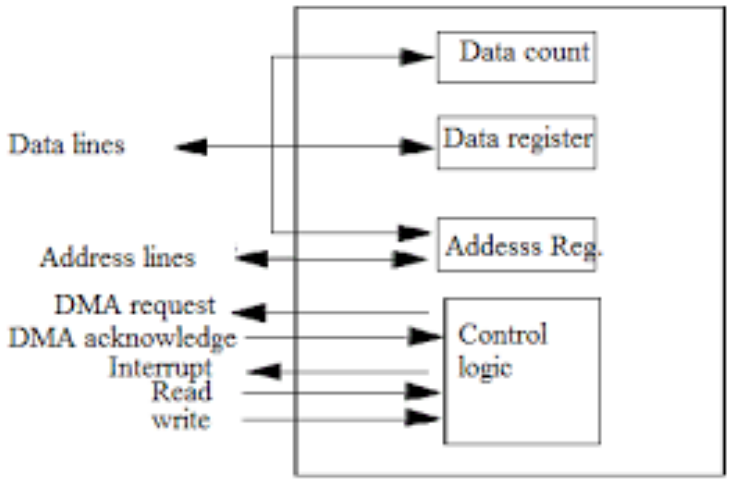
1. **Inefficient:** PIO can be inefficient, as the CPU is often idle while the I/O device is performing the operation.
2. **CPU-bound:** PIO can be CPU-bound, which means that the CPU can be a bottleneck for I/O operations.
3. **Complex:** PIO can be complex to implement, especially for devices with complex addressing modes.

DMA and INTRRUPTS:

DMA (Direct Memory Access):

- In a typical computer system, when data needs to be transferred between devices, the CPU is responsible for managing the entire process. The CPU initiates the transfer, retrieves data from one device, stores it temporarily in its own memory (RAM), and then transfers it to the destination device. This process requires the CPU's direct involvement, and it can be time-consuming and inefficient.
- **Direct Memory Access**, is a technique used in computer systems to transfer data between devices without involving the central processing unit (CPU) directly.
- DMA is especially useful when large amounts of data need to be moved between devices, such as in data storage, networking, or multimedia applications.
- The **DMA Controller** is responsible for control the data transfer process. It takes control of the system bus, accesses the memory, and moves the data between the source and destination devices.
- **DMA Controller** is a type of control unit that works as an interface for the data bus and the I/O Devices. DMA Controller also contains an address unit, which generates the address and selects an I/O device for the transfer of data.

Here we are showing the block diagram of the DMA Controller.



The DMA controller registers have three registers as follows.

- **Address register –** It contains the address to specify the desired location in memory.
- **Word count register –** It contains the number of words to be transferred.
- **Control register –** It specifies the transfer mode (read/write).

Advantages of DMA:

1. **Increased Data Transfer Speed:** DMA allows data to be transferred between peripheral devices and memory directly, without involving the CPU. This reduces overhead on the CPU and increases data transfer speeds, making it more efficient for bulk data transfers.
2. **Off-loading CPU:** DMA offloads data transfer tasks from the CPU, freeing it up to perform other computational tasks. This improves overall system performance and multitasking capabilities.
3. **Reduced Latency:** Since DMA transfers data directly to and from memory, it reduces the latency associated with CPU involvement. This is particularly important in real-time applications, such as audio and video streaming, where delays could cause disruptions.
4. **Streaming and Burst Transfers:** DMA is well-suited for streaming data and burst transfers. It can move data continuously without CPU intervention, making it ideal for high-bandwidth applications.
5. **Efficient I/O Handling:** DMA significantly handles I/O operations, as it allows for more efficient data movement between peripherals and memory.
6. **Improved Energy Efficiency:** By offloading data transfer tasks from the CPU, it can lead to lower power consumption since, CPU can enter low-power states more frequently.

Disadvantages of DMA:

1. **Complexity:** DMA controllers and their associated setup and configuration can be complex, especially when dealing with multiple DMA channels and coordination with the CPU and peripherals.
2. **Limited Control:** While DMA can improve data transfer efficiency, it also means that the CPU has less control over the data transfer process. This lack of control can sometimes lead to conflicts or issues if not properly managed.
3. **Potential for Data Corruption:** If not implemented correctly, DMA transfers can lead to data corruption. DMA transfers bypass the CPU's memory protection mechanisms, making it crucial to ensure proper memory access control and error handling.
4. **Synchronization and Coordination:** In systems with multiple DMA channels and concurrent data transfers, proper synchronization and coordination become important to avoid conflicts and ensure data integrity.
5. **Limited Support for All Devices:** Not all peripheral devices are DMA-capable. Some older or simpler devices may not support DMA, which could limit the advantages of using DMA in certain scenarios.
6. **Complex Debugging:** Debugging issues related to DMA can be more challenging than traditional CPU-driven data transfers, as they involve hardware-level interactions that might be harder to diagnose and resolve.

INTERRUPTS:

- **Interrupts** are a fundamental mechanism in computer architecture that allows devices or external events to asynchronously pause the normal execution of a program and transfer control to a specific routine called **interrupt handler** or **interrupt service routine (ISR)**.
- **Interrupts** are crucial for handling time-critical events, input/output (I/O) operations, and facilitating communication between different parts of a computer system.
- When an interrupt occurs, the current execution of the CPU is suspended, and the processor immediately switches its attention to the interrupting device or event.
- The CPU saves the current execution context, which includes the program counter (the memory address of the next instruction to be executed), processor status, and other relevant registers, onto the stack.
- The **Interrupt Controller** is responsible for identifying and prioritizing interrupts from various devices and delivering them to the CPU.
- When an interrupt is raised, the interrupt controller interrupts the CPU by declaring a specific interrupt line, triggering the interrupt signal. The CPU detects this interrupt signal and responds by pausing its current execution.

DMA stands for direct memory access. It is a technique for transferring data between the computer's memory and peripheral devices without involving the CPU. This can free up the CPU to perform other tasks, such as processing instructions or handling other I/O requests.

Interrupts are a way for I/O devices to signal the CPU that they need attention. When an I/O device interrupts the CPU, the CPU will stop what it is doing and handle the interrupt.

Here is a table that summarizes the key differences between DMA and interrupts:

Aspect	**DMA (Direct Memory Access)**	**Interrupt**
Function	Data transfer mechanism	Event handling mechanism
CPU Involvement	Data transfer occurs without CPU	Requires CPU to handle interrupt
Usage	Bulk data transfers	Handling time-sensitive events
Transfer Speed	Faster data transfer	Depends on CPU's responsiveness
Data Transfer Size	Handles larger data blocks	Handles small units of data or events
Purpose	Efficient data movement between devices	Responding to external/internal events
Latency	Lower latency as CPU is not involved	Can introduce latency depending on ISR (interrupt service routine)
Examples	Transfer data between disk and RAM	Handling keyboard input, I/O completion

I/O PROCESSORS:

- In computer architecture, I/O (Input/Output) processors, also known as I/O controllers or I/O coprocessors, are specialized hardware components that handle input and output operations between the CPU and external devices.
- They offload the CPU from the burden of managing I/O operations, improving overall system performance and efficiency.
- I/O processors serve as intermediaries between CPU and external devices, such as storage devices (hard drives, SSDs), network interfaces, display devices, keyboards, and mouse.
- They provide the necessary interfaces and protocols for communication between the CPU and these devices.

Features of I/O Processors:

1. **I/O Interface:** Input/ Output processors provide the interface between CPU and external devices. They enable communication and data exchange between the CPU and peripherals. The I/O interface may include physical connectors, electrical signaling, and communication protocols that allow devices and the I/O processor to exchange data and control signals.
2. **Data Transfer:** I/O processors facilitate the transfer of data between CPU and external devices. They handle low-level details of data transmission, including data formatting, serialization, and deserialization, as well as error detection and correction. They ensures that data is properly packaged, transmitted, and received by appropriate devices.
3. **Device Control:** I/O processors control the operation of external devices. They issue commands to devices, configure their settings, and manage their status. This includes initializing devices, setting up data transfer modes, and handling device-specific protocols. The I/O processor translates high-level commands from the CPU into device-specific commands that the devices understand.
4. **Buffering and Caching:** I/O processors often include buffers and caches to optimize data transfer between the CPU and devices. **Buffers** act as temporary storage areas that hold data during I/O operations, allowing for smoother data flow between devices and the CPU. Whereas, **Caches** store frequently accessed data from devices, reducing the need for repeated access to slower external storage.
5. **Interrupt Handling:** Input/ Output processors play a crucial role in interrupt handling. They can generate interrupts to inform the CPU about events or status changes occurring in external devices. **For example**, when data is ready to be read from a device or when a device encounters an error. The Input/ Output processor coordinates with the interrupt controller to prioritize and manage these interrupts, ensuring timely and appropriate response by the CPU.

6. **DMA (Direct Memory Access) Control:** Some I/O processors include DMA capabilities. DMA allows devices to transfer data directly to and from system memory without CPU involvement. I/O processors with DMA controllers manages the data transfer process, taking control of the system bus, accessing memory, and moving data between devices and memory. This offloads (unburden) the CPU from managing the data transfer, improving overall system efficiency.
7. **Protocol Conversion:** I/O processors may handle protocol conversion between different devices and the CPU. **For example**, they can convert data between serial and parallel formats, or between different networking protocols. This enables devices with different communication standards to interact faultlessly with the CPU.