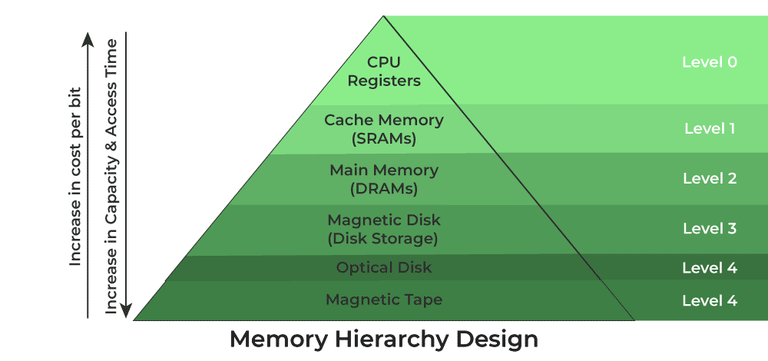
**Why Memory Hierarchy is Required in the System?**

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

**Types of Memory Hierarchy**

This Memory Hierarchy Design is divided into 2 main types:

* **External Memory or Secondary Memory:** Comprising of Magnetic Disk, Optical Disk, and Magnetic Tape i.e. peripheral storage devices which are accessible by the processor via an I/O Module.
* **Internal Memory or Primary Memory:** Comprising of Main Memory, Cache Memory &[CPU registers](https://www.geeksforgeeks.org/different-classes-of-cpu-registers/). This is directly accessible by the processor.



*Memory Hierarchy Design*

**Memory Hierarchy Design**

**1. Registers**

[Registers](https://www.geeksforgeeks.org/different-classes-of-cpu-registers/) are small, high-speed memory units located in the CPU. They are used to store the most frequently used data and instructions. Registers have the fastest access time and the smallest storage capacity, typically ranging from 16 to 64 bits.

**2. Cache Memory**

[Cache memory](https://www.geeksforgeeks.org/cache-memory-in-computer-organization/) is a small, fast memory unit located close to the CPU. It stores frequently used data and instructions that have been recently accessed from the main memory. Cache memory is designed to minimize the time it takes to access data by providing the CPU with quick access to frequently used data.

**3. Main Memory**

[Main memory](https://www.geeksforgeeks.org/memory-management-in-operating-system/), also known as RAM (Random Access Memory), is the primary memory of a computer system. It has a larger storage capacity than cache memory, but it is slower. Main memory is used to store data and instructions that are currently in use by the CPU.

**Types of Main Memory**

* **Static RAM:** [Static RAM](https://www.geeksforgeeks.org/difference-between-sram-and-dram/) stores the binary information in flip flops and information remains valid until power is supplied. It has a faster access time and is used in implementing cache memory.
* **Dynamic RAM:** It stores the binary information as a charge on the capacitor. It requires refreshing circuitry to maintain the charge on the capacitors after a few milliseconds. It contains more memory cells per unit area as compared to SRAM.

**4. Secondary Storage**

Secondary storage, such as [hard disk drives (HDD) and solid-state drives (SSD)](https://www.geeksforgeeks.org/difference-between-hard-disk-drive-hdd-and-solid-state-drive-ssd/), is a non-volatile memory unit that has a larger storage capacity than main memory. It is used to store data and instructions that are not currently in use by the CPU. Secondary storage has the slowest access time and is typically the least expensive type of memory in the memory hierarchy.

**5. Magnetic Disk**

[Magnetic Disks](https://www.geeksforgeeks.org/magnetic-disk-memory/) are simply circular plates that are fabricated with either a metal or a plastic or a magnetized material. The Magnetic disks work at a high speed inside the computer and these are frequently used.

**6. Magnetic Tape**

[Magnetic Tape](https://www.geeksforgeeks.org/magnetic-tape-memory/) is simply a magnetic recording device that is covered with a plastic film. It is generally used for the backup of data. In the case of a magnetic tape, the access time for a computer is a little slower and therefore, it requires some amount of time for accessing the strip.

**Characteristics of Memory Hierarchy**

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

**Advantages of Memory Hierarchy**

* It helps in removing some destruction, and managing the memory in a better way.
* It helps in spreading the data all over the computer system.
* It saves the consumer’s price and time.

**System-Supported Memory Standards**

According to the memory Hierarchy, the system-supported memory standards are defined below:

| **Level** | **1** | **2** | **3** | **4** |
| --- | --- | --- | --- | --- |
| **Name** | Register | Cache | Main Memory | Secondary Memory |
| **Size** | <1 KB | less than 16 MB | <16GB | >100 GB |
| **Implementation** | Multi-ports | On-chip/SRAM | DRAM (capacitor memory) | Magnetic |
| **Access Time** | 0.25ns to 0.5ns | 0.5 to 25ns | 80ns  to 250ns | 50 lakh ns |
| **Bandwidth** | 20000 to 1 lakh MB | 5000 to 15000 | 1000 to 5000 | 20 to 150 |
| **Managed by** | Compiler | Hardware | Operating System | Operating System |
| **Backing Mechanism** | From cache | from Main Memory |  |  |

# **Main Memory**

The main memory acts as the central storage unit in a computer system. It is a relatively large and fast memory which is used to store programs and data during the run time operations.

The primary technology used for the main memory is based on semiconductor integrated circuits. The integrated circuits for the main memory are classified into two major units.

1. RAM (Random Access Memory) integrated circuit chips
2. ROM (Read Only Memory) integrated circuit chips

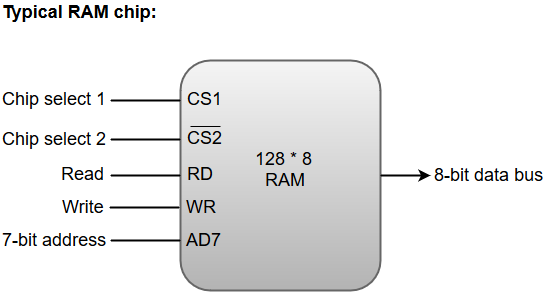
## RAM integrated circuit chips

The RAM integrated circuit chips are further classified into two possible operating modes, **static** and **dynamic**.

The primary compositions of a static RAM are flip-flops that store the binary information. The nature of the stored information is volatile, i.e. it remains valid as long as power is applied to the system. The static RAM is easy to use and takes less time performing read and write operations as compared to dynamic RAM.

The dynamic RAM exhibits the binary information in the form of electric charges that are applied to capacitors. The capacitors are integrated inside the chip by MOS transistors. The dynamic RAM consumes less power and provides large storage capacity in a single memory chip.

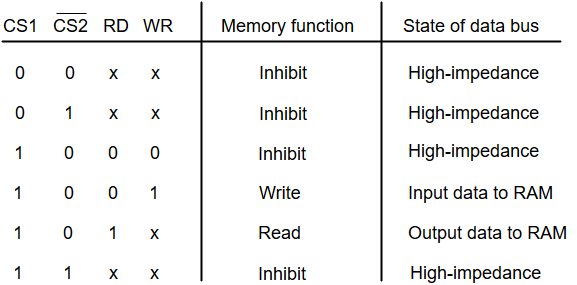
RAM chips are available in a variety of sizes and are used as per the system requirement. The following block diagram demonstrates the chip interconnection in a 128 \* 8 RAM chip.



* A 128 \* 8 RAM chip has a memory capacity of 128 words of eight bits (one byte) per word. This requires a 7-bit address and an 8-bit bidirectional data bus.
* The 8-bit bidirectional data bus allows the transfer of data either from memory to CPU during a **read** operation or from CPU to memory during a **write** operation.
* The **read** and **write** inputs specify the memory operation, and the two chip select (CS) control inputs are for enabling the chip only when the microprocessor selects it.
* The bidirectional data bus is constructed using **three-state buffers**.
* The output generated by three-state buffers can be placed in one of the three possible states which include a signal equivalent to logic 1, a signal equal to logic 0, or a high-impedance state.

#### Note: The logic 1 and 0 are standard digital signals whereas the high-impedance state behaves like an open circuit, which means that the output does not carry a signal and has no logic significance.

The following function table specifies the operations of a 128 \* 8 RAM chip.



From the functional table, we can conclude that the unit is in operation only when CS1 = 1 and CS2 = 0. The bar on top of the second select variable indicates that this input is enabled when it is equal to 0.

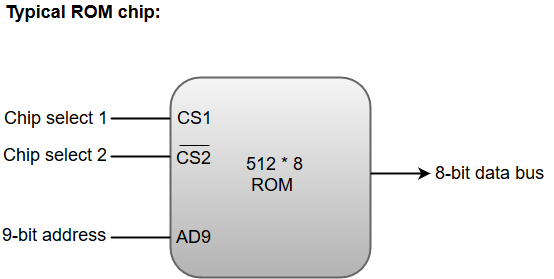
## ROM integrated circuit

The primary component of the main memory is RAM integrated circuit chips, but a portion of memory may be constructed with ROM chips.

A ROM memory is used for keeping programs and data that are permanently resident in the computer.

Apart from the permanent storage of data, the ROM portion of main memory is needed for storing an initial program called a **bootstrap loader**. The primary function of the **bootstrap loader** program is to start the computer software operating when power is turned on.

ROM chips are also available in a variety of sizes and are also used as per the system requirement. The following block diagram demonstrates the chip interconnection in a 512 \* 8 ROM chip.



* A ROM chip has a similar organization as a RAM chip. However, a ROM can only perform read operation; the data bus can only operate in an output mode.
* The 9-bit address lines in the ROM chip specify any one of the 512 bytes stored in it.
* The value for chip select 1 and chip select 2 must be 1 and 0 for the unit to operate. Otherwise, the data bus is said to be in a high-impedance state.

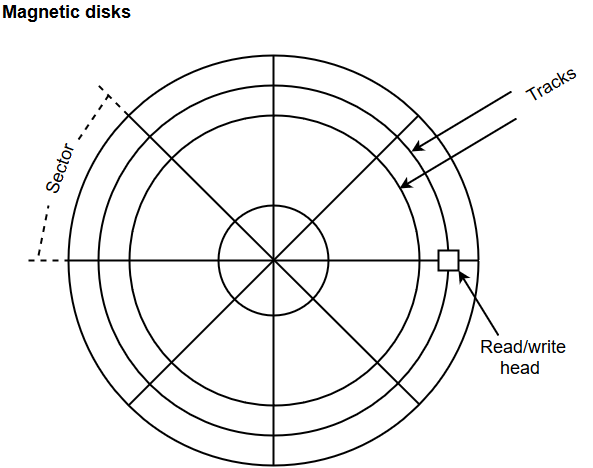
# **Auxiliary Memory**

An Auxiliary memory is known as the lowest-cost, highest-capacity and slowest-access storage in a computer system. It is where programs and data are kept for long-term storage or when not in immediate use. The most common examples of auxiliary memories are magnetic tapes and magnetic disks.

## Magnetic Disks

A magnetic disk is a type of memory constructed using a circular plate of metal or plastic coated with magnetized materials. Usually, both sides of the disks are used to carry out read/write operations. However, several disks may be stacked on one spindle with read/write head available on each surface.

The following image shows the structural representation for a magnetic disk.



* The memory bits are stored in the magnetized surface in spots along the concentric circles called tracks.
* The concentric circles (tracks) are commonly divided into sections called sectors.

## Magnetic Tape

Magnetic tape is a storage medium that allows data archiving, collection, and backup for different kinds of data. The magnetic tape is constructed using a plastic strip coated with a magnetic recording medium.

The bits are recorded as magnetic spots on the tape along several tracks. Usually, seven or nine bits are recorded simultaneously to form a character together with a parity bit.

Magnetic tape units can be halted, started to move forward or in reverse, or can be rewound. However, they cannot be started or stopped fast enough between individual characters. For this reason, information is recorded in blocks referred to as records.

# **Associative Memory**

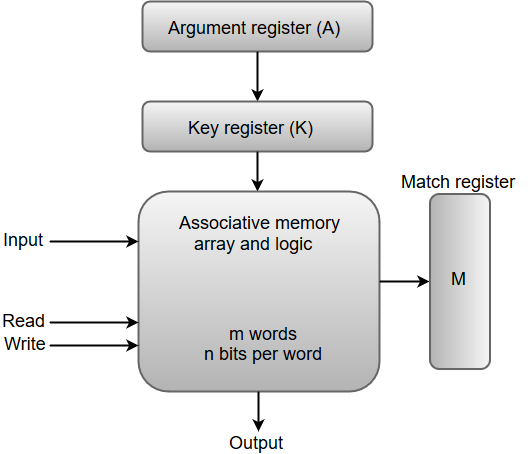
An associative memory can be considered as a memory unit whose stored data can be identified for access by the content of the data itself rather than by an address or memory location.

Associative memory is often referred to as **Content Addressable Memory (CAM)**.

When a write operation is performed on associative memory, no address or memory location is given to the word. The memory itself is capable of finding an empty unused location to store the word.

On the other hand, when the word is to be read from an associative memory, the content of the word, or part of the word, is specified. The words which match the specified content are located by the memory and are marked for reading.

The following diagram shows the block representation of an Associative memory.



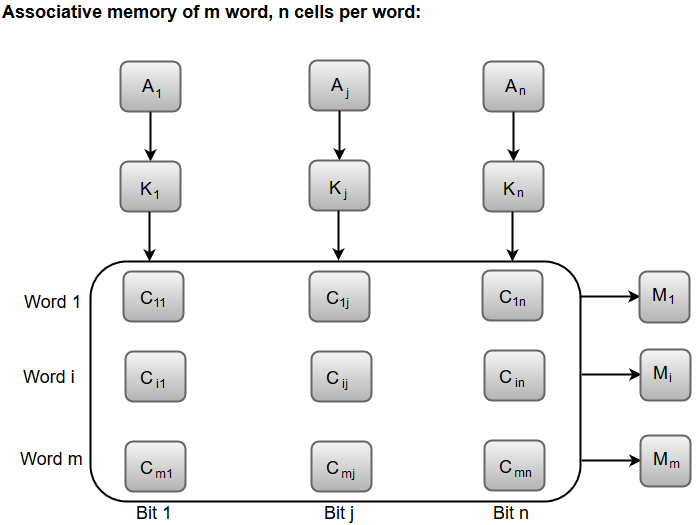
From the block diagram, we can say that an associative memory consists of a memory array and logic for 'm' words with 'n' bits per word.

The functional registers like the argument register **A** and key register **K** each have **n** bits, one for each bit of a word. The match register **M** consists of **m** bits, one for each memory word.

The words which are kept in the memory are compared in parallel with the content of the argument register.

The key register (K) provides a mask for choosing a particular field or key in the argument word. If the key register contains a binary value of all 1's, then the entire argument is compared with each memory word. Otherwise, only those bits in the argument that have 1's in their corresponding position of the key register are compared. Thus, the key provides a mask for identifying a piece of information which specifies how the reference to memory is made.

The following diagram can represent the relation between the memory array and the external registers in an associative memory.



The cells present inside the memory array are marked by the letter C with two subscripts. The first subscript gives the word number and the second specifies the bit position in the word. For instance, the cell Cij is the cell for bit **j** in word **i**.

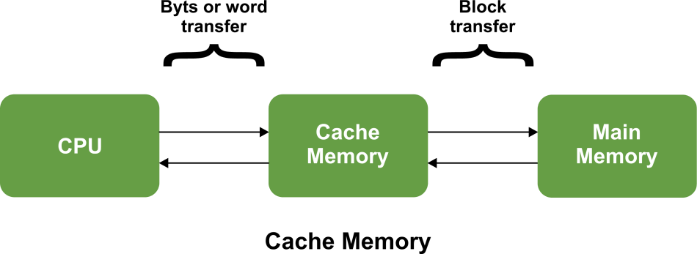
A bit Aj in the argument register is compared with all the bits in column **j** of the array provided that Kj = 1. This process is done for all columns **j** = 1, 2, 3......, n.

If a match occurs between all the unmasked bits of the argument and the bits in word **i**, the corresponding bit Mi in the match register is set to 1. If one or more unmasked bits of the argument and the word do not match, Mi is cleared to 0.

# **Cache Memory**

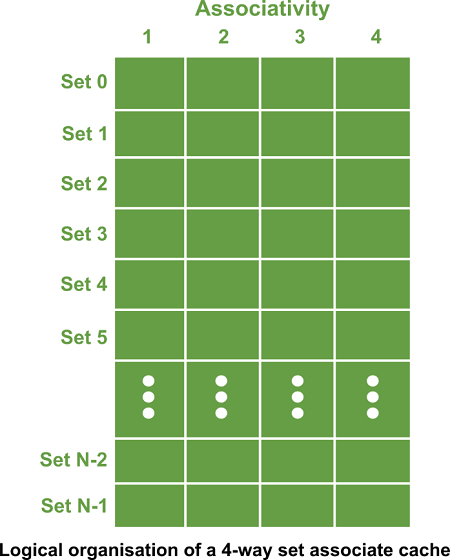
The data or contents of the main memory that are used frequently by CPU are stored in the cache memory so that the processor can easily access that data in a shorter time. Whenever the CPU needs to access memory, it first checks the cache memory. If the data is not found in cache memory, then the CPU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

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



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

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



As shown in the above sets are represented by the rows. The example contains N sets and each set contains four blocks. Whenever an access is made to cache, the cache controller does not search the entire cache in order to look for a match. Rather, the controller maps the address to a particular set of the cache and therefore searches only the set for a match.

If a required block is not found in that set, the block is not present in the cache and cache controller does not search it further. This kind of cache organisation is called set associative because the cache is divided into distinct sets of blocks. As each set contains four blocks the cache is said to be four way set associative.

**The basic operation of a cache memory is as follows:**

* When the CPU needs to access memory, the cache is examined. If the word is found in the cache, it is read from the fast memory.
* If the word addressed by the CPU is not found in the cache, the main memory is accessed to read the word.
* A block of words one just accessed is then transferred from main memory to cache memory. The block size may vary from one word (the one just accessed) to about 16 words adjacent to the one just accessed.
* The performance of the cache memory is frequently measured in terms of a quantity called **hit ratio**.
* When the CPU refers to memory and finds the word in cache, it is said to produce a **hit**.
* If the word is not found in the cache, it is in main memory and it counts as a **miss**.
* The ratio of the number of hits divided by the total CPU references to memory (hits plus misses) is the hit ratio.

## Levels of memory:

**Level 1**

It is a type of memory in which data is stored and accepted that are immediately stored in CPU. Most commonly used register is accumulator, Program counter, address register etc.

**Level 2**

It is the fastest memory which has faster access time where data is temporarily stored for faster access.

**Level 3**

It is memory on which computer works currently. It is small in size and once power is off data no longer stays in this memory.

**Level 4**

It is external memory which is not as fast as main memory but data stays permanently in this memory.

## Cache Mapping:

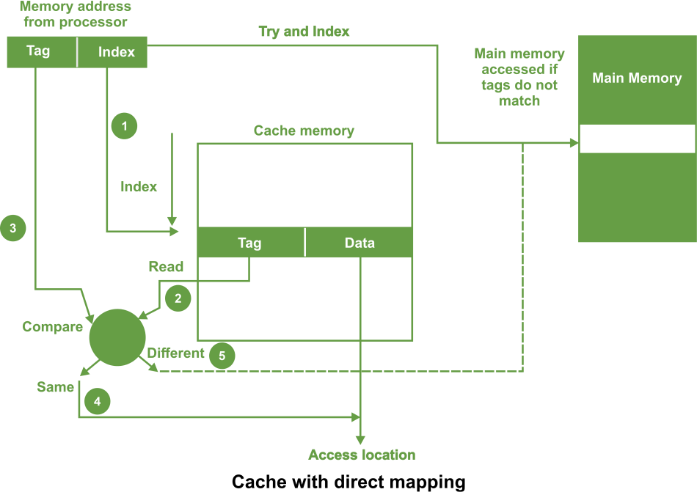
There are three different types of mapping used for the purpose of cache memory which are as follows:

* Direct mapping,
* Associative mapping
* Set-Associative mapping

AD

### Direct Mapping -

In direct mapping, the cache consists of normal high-speed random-access memory. Each location in the cache holds the data, at a specific address in the cache. This address is given by the lower significant bits of the main memory address. This enables the block to be selected directly from the lower significant bit of the memory address. The remaining higher significant bits of the address are stored in the cache with the data to complete the identification of the cached data.



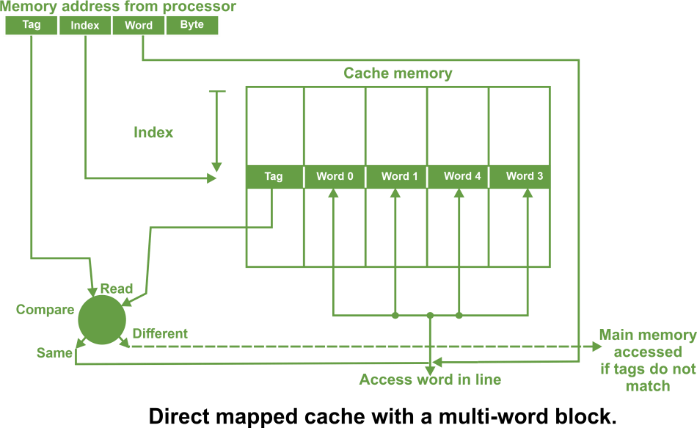
As shown in the above figure, the address from processor is divided into two field a tag and an index.

The tag consists of the higher significant bits of the address and these bits are stored with the data in cache. The index consists of the lower significant b of the address. Whenever the memory is referenced, the following sequence of events occurs

1. The index is first used to access a word in the cache.
2. The tag stored in the accessed word is read.
3. This tag is then compared with the tag in the address.
4. If two tags are same this indicates cache hit and required data is read from the cache word.
5. If the two tags are not same, this indicates a cache miss. Then the reference is made to the main memory to find it.

For a memory read operation, the word is then transferred into the cache. It is possible to pass the information to the cache and the process simultaneously.

In direct mapped cache, there can also be a line consisting of more than one word as shown in the following figure



In such a case, the main memory address consists of a tag, an index and a word within a line. All the words within a line in the cache have the same stored tag

AD

The index part in the address is used to access the cache and the stored tag is compared with required tag address.

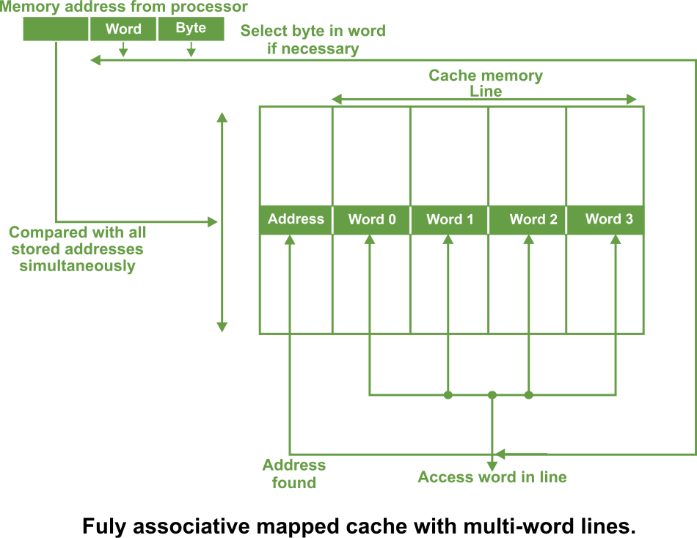
For a read operation, if the tags are same, the word within the block is selected for transfer to the processor. If tags are not same, the block containing the required word is first transferred to the cache. In direct mapping, the corresponding blocks with the same index in the main memory will map into the same block in the cache, and hence only blocks with different indices can be in the cache at the same time. It is important that all words in the cache must have different indices. The tags may be the same or different.

### Set Associative Mapping -

In set associative mapping a cache is divided into a set of blocks. The number of blocks in a set is known as associativity or set size. Each block in each set has a stored tag. This tag together with index completely identify the block.

Thus, set associative mapping allows a limited number of blocks, with the same index and different tags.

An example of four way set associative cache having four blocks in each set is shown in the following figure



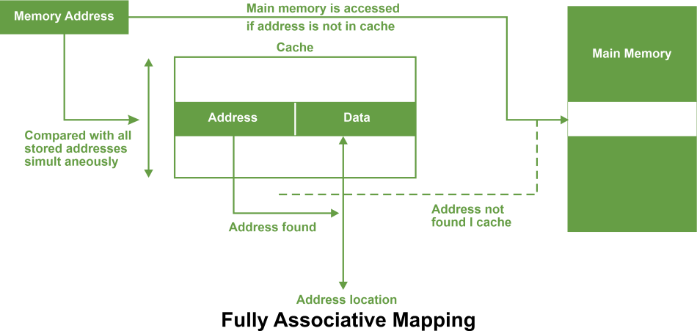
**In this type of cache, the following steps are used to access the data from a cache:**

1. The index of the address from the processor is used to access the set.
2. Then the comparators are used to compare all tags of the selected set with the incoming tag.
3. If a match is found, the corresponding location is accessed.
4. If no match is found, an access is made to the main memory.

The tag address bits are always chosen to be the most significant bits of the full address, the block address bits are the next significant bits and the word/byte address bits are the least significant bits. The number of comparators required in the set associative cache is given by the number of blocks in a set. The set can be selected quickly and all the blocks of the set can be read out simultaneously with the tags before waiting for the tag comparisons to be made. After a tag has been identified, the corresponding block can be selected.

### Fully associative mapping

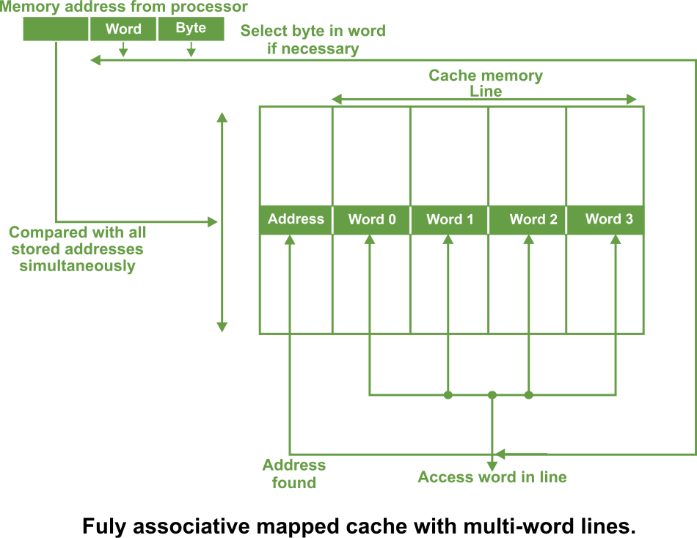
In fully associative type of cache memory, each location in cache stores both memory address as well as data.



Whenever a data is requested, the incoming memory address a simultaneously compared with all stored addresses using the internal logic the associative memory.

If a match is found, the corresponding is read out. Otherwise, the main memory is accessed if address is not found in cache.

This method is known as fully associative mapping approach because cached data is related to the main memory by storing both memory address and data in the cache. In all organisations, data can be more than one word as shown in the following figure.



A line constitutes four words, each word being 4 bytes. In such case, the least significant part of the address selects the particular byte, the next part selects the word, and the remaining bits form the address. These address bits are compared to the address in the cache. The whole line can be transferred to and from the cache in one transaction if there are sufficient data paths between the main memory and the cache. With only one data word path, the words of the line have to be transferred in separate transactions.

The main **advantage** of fully associative mapped cache is that it provides greatest flexibility of holding combinations of blocks in the cache and conflict for a given cache.

**It suffers from certain disadvantages:**

1. It is expensive method because of the high cost of associative memory.
2. It requires a replacement algorithm in order to select a block to be removed whenever cache miss occurs.
3. Such an algorithm must be implemented in hardware to maintain a high speed of operation.

The fully associative mechanism is usually employed by microprocessors with small internal cache.

# **What is Virtual Memory in OS (Operating System)?**

Virtual Memory is a storage scheme that provides user an illusion of having a very big main memory. This is done by treating a part of secondary memory as the main memory.

In this scheme, User can load the bigger size processes than the available main memory by having the illusion that the memory is available to load the process.

Instead of loading one big process in the main memory, the Operating System loads the different parts of more than one process in the main memory.

By doing this, the degree of multiprogramming will be increased and therefore, the CPU utilization will also be increased.

## How Virtual Memory Works?

In modern word, virtual memory has become quite common these days. In this scheme, whenever some pages needs to be loaded in the main memory for the execution and the memory is not available for those many pages, then in that case, instead of stopping the pages from entering in the main memory, the OS search for the RAM area that are least used in the recent times or that are not referenced and copy that into the secondary memory to make the space for the new pages in the main memory.

Since all this procedure happens automatically, therefore it makes the computer feel like it is having the unlimited RAM.

## Demand Paging

Demand Paging is a popular method of virtual memory management. In demand paging, the pages of a process which are least used, get stored in the secondary memory.

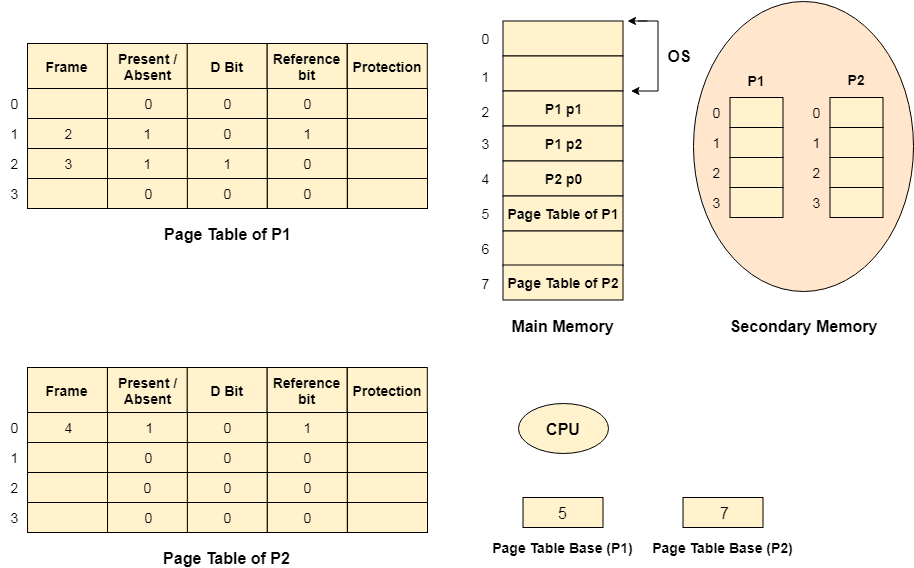
A page is copied to the main memory when its demand is made or page fault occurs. There are various page replacement algorithms which are used to determine the pages which will be replaced. We will discuss each one of them later in detail.

## Snapshot of a virtual memory management system

Let us assume 2 processes, P1 and P2, contains 4 pages each. Each page size is 1 KB. The main memory contains 8 frame of 1 KB each. The OS resides in the first two partitions. In the third partition, 1st page of P1 is stored and the other frames are also shown as filled with the different pages of processes in the main memory.

The page tables of both the pages are 1 KB size each and therefore they can be fit in one frame each. The page tables of both the processes contain various information that is also shown in the image.

The CPU contains a register which contains the base address of page table that is 5 in the case of P1 and 7 in the case of P2. This page table base address will be added to the page number of the Logical address when it comes to accessing the actual corresponding entry.



### Advantages of Virtual Memory

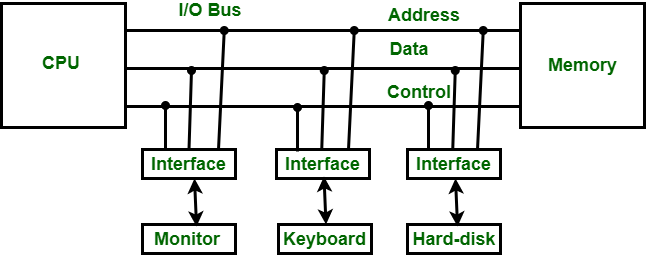
1. The degree of Multiprogramming will be increased.
2. User can run large application with less real RAM.
3. There is no need to buy more memory RAMs.

### Disadvantages of Virtual Memory

1. The system becomes slower since swapping takes time.
2. It takes more time in switching between applications.
3. The user will have the lesser hard disk space for its use.

Input/output interfaces

[Input-Output Interface](https://www.geeksforgeeks.org/structure-of-input-output-interface/) is used as an method which helps in transferring of information between the internal storage devices i.e. memory and the external peripheral device . A peripheral device is that which provide input and output for the computer, it is also called Input-Output devices. For Example: A keyboard and mouse provide Input to the computer are called input devices while a monitor and printer that provide output to the computer are called output devices. Just like the external hard-drives, there is also availability of some peripheral devices which are able to provide both input and output.



*Input-Output Interface*

In micro-computer base system, the only purpose of peripheral devices is just to provide **special communication links** for the interfacing them with the CPU. To resolve the differences between peripheral devices and CPU, there is a special need for communication links.

The major differences are as follows:

1. The nature of peripheral devices is electromagnetic and electro-mechanical. The nature of the CPU is electronic. There is a lot of difference in the mode of operation of both peripheral devices and CPU.
2. There is also a synchronization mechanism because the data transfer rate of peripheral devices are slow than CPU.
3. In peripheral devices, data code and formats are differ from the format in the CPU and memory.
4. The operating mode of peripheral devices are different and each may be controlled so as not to disturb the operation of other peripheral devices connected to CPU.

There is a special need of the additional hardware to resolve the differences between CPU and peripheral devices to supervise and synchronize all input and output devices.

### Functions of Input-Output Interface:

1. It is used to synchronize the operating speed of CPU with respect to input-output devices.
2. It selects the input-output device which is appropriate for the interpretation of the input-output device.
3. It is capable of providing signals like control and timing signals.
4. In this data buffering can be possible through data bus.
5. There are various error detectors.
6. It converts serial data into parallel data and vice-versa.
7. It also convert digital data into analog signal and vice-versa.

# **Asynchronous Data Transfer in Computer Organization**

The internal operations in an individual unit of a digital system are synchronized using clock pulse. It means clock pulse is given to all registers within a unit. And all data transfer among internal registers occurs simultaneously during the occurrence of the clock pulse. Now, suppose any two units of a digital system are designed independently, such as CPU and I/O interface.

If the registers in the I/O interface share a common clock with CPU registers, then transfer between the two units is said to be synchronous. But in most cases, the internal timing in each unit is independent of each other, so each uses its private clock for its internal registers. In this case, the two units are said to be asynchronous to each other, and if data transfer occurs between them, this data transfer is called **Asynchronous Data Transfer**.

But, the Asynchronous Data Transfer between two independent units requires that control signals be transmitted between the communicating units so that the time can be indicated at which they send data. These two methods can achieve this asynchronous way of data transfer:

* **Strobe control:** A strobe pulse is supplied by one unit to indicate to the other unit when the transfer has to occur.
* **Handshaking:** This method is commonly used to accompany each data item being transferred with a control signal that indicates data in the bus. The unit receiving the data item responds with another signal to acknowledge receipt of the data.

The strobe pulse and handshaking method of asynchronous data transfer is not restricted to I/O transfer. They are used extensively on numerous occasions requiring the transfer of data between two independent units. So, here we consider the transmitting unit as a source and receiving unit as a destination.

For example, the CPU is the source during output or write transfer and the destination unit during input or read transfer.

Therefore, the control sequence during an asynchronous transfer depends on whether the transfer is initiated by the source or by the destination.

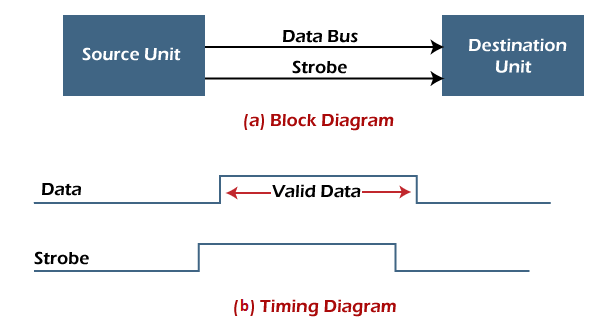
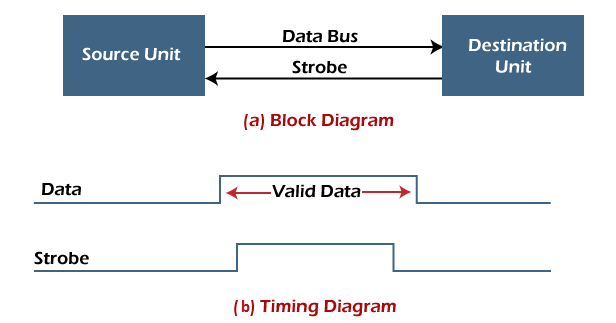
So, while discussing each data transfer method asynchronously, you can see the control sequence in both terms when it is initiated by source or by destination. In this way, each data transfer method can be further divided into parts, source initiated and destination initiated.

### Asynchronous Data Transfer Methods

The asynchronous data transfer between two independent units requires that control signals be transmitted between the communicating units to indicate when they send the data. Thus, the two methods can achieve the asynchronous way of data transfer.

**1. Strobe Control Method**

The Strobe Control method of asynchronous data transfer employs a single control line to time each transfer. This control line is also known as a strobe, and it may be achieved either by source or destination, depending on which initiate the transfer.

1. **Source initiated strobe:** In the below block diagram, you can see that strobe is initiated by source, and as shown in the timing diagram, the source unit first places the data on the data bus.  
     
   After a brief delay to ensure that the data resolve to a stable value, the source activates a strobe pulse. The information on the data bus and strobe control signal remains in the active state for a sufficient time to allow the destination unit to receive the data.  
   The destination unit uses a falling edge of strobe control to transfer the contents of a data bus to one of its internal registers. The source removes the data from the data bus after it disables its strobe pulse. Thus, new valid data will be available only after the strobe is enabled again.  
   In this case, the strobe may be a memory-write control signal from the CPU to a memory unit. The CPU places the word on the data bus and informs the memory unit, which is the destination.
2. **Destination initiated strobe:** In the below block diagram, you see that the strobe initiated by destination, and in the timing diagram, the destination unit first activates the strobe pulse, informing the source to provide the data.  
     
   The source unit responds by placing the requested binary information on the data bus. The data must be valid and remain on the bus long enough for the destination unit to accept it.  
   The falling edge of the strobe pulse can use again to trigger a destination register. The destination unit then disables the strobe. Finally, and source removes the data from the data bus after a determined time interval.  
   In this case, the strobe may be a memory read control from the CPU to a memory unit. The CPU initiates the read operation to inform the memory, which is a source unit, to place the selected word into the data bus.

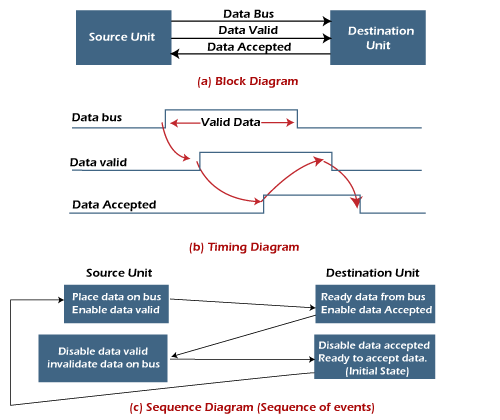
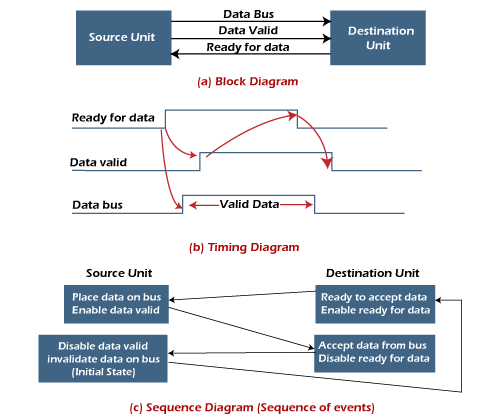
**2. Handshaking Method**

The strobe method has the disadvantage that the source unit that initiates the transfer has no way of knowing whether the destination has received the data that was placed in the bus. Similarly, a destination unit that initiates the transfer has no way of knowing whether the source unit has placed data on the bus.

So this problem is solved by the handshaking method. The handshaking method introduces a second control signal line that replays the unit that initiates the transfer.

In this method, one control line is in the same direction as the data flow in the bus from the source to the destination. The source unit uses it to inform the destination unit whether there are valid data in the bus.

The other control line is in the other direction from the destination to the source. This is because the destination unit uses it to inform the source whether it can accept data. And in it also, the sequence of control depends on the unit that initiates the transfer. So it means the sequence of control depends on whether the transfer is initiated by source and destination.

* **Source initiated handshaking:** In the below block diagram, you can see that two handshaking lines are "**data valid**", which is generated by the source unit, and "**data accepted**", generated by the destination unit.  
    
  The timing diagram shows the timing relationship of the exchange of signals between the two units. The source initiates a transfer by placing data on the bus and enabling its data valid signal. The destination unit then activates the data accepted signal after it accepts the data from the bus.  
  The source unit then disables its valid data signal, which invalidates the data on the bus.  
  After this, the destination unit disables its data accepted signal, and the system goes into its initial state. The source unit does not send the next data item until after the destination unit shows readiness to accept new data by disabling the data accepted signal.  
  This sequence of events described in its sequence diagram, which shows the above sequence in which the system is present at any given time.
* **Destination initiated handshaking:** In the below block diagram, you see that the two handshaking lines are "**data valid**", generated by the source unit, and "**ready for data**" generated by the destination unit.  
  Note that the name of signal data accepted generated by the destination unit has been changed to ready for data to reflect its new meaning.  
    
  The destination transfer is initiated, so the source unit does not place data on the data bus until it receives a ready data signal from the destination unit. After that, the handshaking process is the same as that of the source initiated.  
  The sequence of events is shown in its sequence diagram, and the timing relationship between signals is shown in its timing diagram. Therefore, the sequence of events in both cases would be identical.

AD

### Advantages of Asynchronous Data Transfer

Asynchronous Data Transfer in computer organization has the following advantages, such as:

* It is more flexible, and devices can exchange information at their own pace. In addition, individual data characters can complete themselves so that even if one packet is corrupted, its predecessors and successors will not be affected.
* It does not require complex processes by the receiving device. Furthermore, it means that inconsistency in data transfer does not result in a big crisis since the device can keep up with the data stream. It also makes asynchronous transfers suitable for applications where character data is generated irregularly.

### Disadvantages of Asynchronous Data Transfer

There are also some disadvantages of using asynchronous data for transfer in computer organization, such as:

AD

* The success of these transmissions depends on the start bits and their recognition. Unfortunately, this can be easily susceptible to line interference, causing these bits to be corrupted or distorted.
* A large portion of the transmitted data is used to control and identify header bits and thus carries no helpful information related to the transmitted data. This invariably means that more data packets need to be sent.

Modes of transfer

Certainly! In computer system organization, the modes of data transfer refer to the ways in which data can be transferred between various components of the computer system. Let's explore the two primary modes of data transfer: Parallel and Serial.

### 1. \*\*Parallel Data Transfer:\*\*

\*\*Description:\*\*

- \*\*Multiple Lines:\*\* In parallel data transfer, multiple data lines are used simultaneously to transmit several bits of data at the same time.

- \*\*Wider Paths:\*\* It's like having a wider road where multiple vehicles (bits of data) can travel side by side.

\*\*Key Points:\*\*

- \*\*Faster Transfer:\*\* Because multiple bits are sent simultaneously, parallel transfer is generally faster than serial transfer.

- \*\*Wider Data Paths:\*\* Requires more physical connections (wires or traces) to facilitate the parallel transmission of bits.

\*\*Example Analogy:\*\*

Imagine a conveyor belt with several lanes, and each lane carries a different type of item. All items move together, reaching the end of the belt at the same time.

### 2. \*\*Serial Data Transfer:\*\*

\*\*Description:\*\*

- \*\*Single Line:\*\* In serial data transfer, bits are sent one after the other along a single communication line.

- \*\*Narrow Path:\*\* It's like a single-lane road where data bits travel sequentially.

\*\*Key Points:\*\*

- \*\*Simpler Connections:\*\* Requires fewer physical connections than parallel transfer.

- \*\*Slower than Parallel:\*\* Generally slower than parallel transfer because bits are transmitted one at a time.

\*\*Example Analogy:\*\*

Think of a train where each carriage represents a bit of data. The carriages travel one after the other along a single track.

### \*\*Choosing Between Parallel and Serial:\*\*

- \*\*Factors to Consider:\*\*

- \*\*Speed Requirements:\*\* If high speed is crucial, parallel transfer might be preferred.

- \*\*Physical Constraints:\*\* Serial transfer is often used when there are limitations on physical space or the number of connections.

- \*\*Advantages and Disadvantages:\*\*

- \*\*Parallel Transfer:\*\*

- \*Advantages:\* Faster data transfer.

- \*Disadvantages:\* More complex connections, potential for synchronization issues.

- \*\*Serial Transfer:\*\*

- \*Advantages:\* Simpler connections, better suited for long-distance communication.

- \*Disadvantages:\* Slower than parallel transfer.

In summary, parallel and serial modes of data transfer represent different approaches to moving information within a computer system. The choice between them depends on factors such as speed requirements, physical constraints, and the specific needs of the system or application.