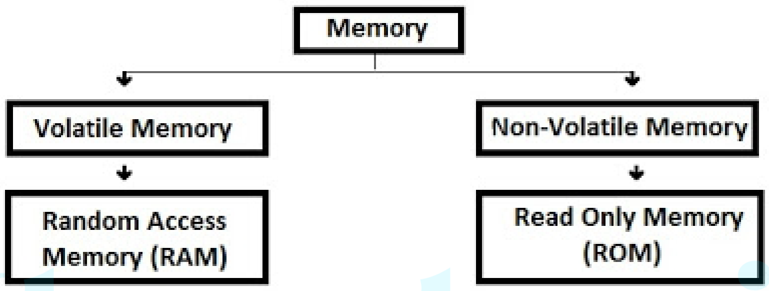
MODULE V
THE MEMORY SYSTEM

Memory is an essential element of a computer. Without its memory, a computer is of hardly any use. Memory plays an important role in saving and retrieving data. The performance of the computer system depends upon the size of the memory. Memory is of following types:

1. **Primary Memory / Volatile Memory.**
2. **Secondary Memory / Non Volatile Memory.**



1. Primary Memory / Volatile Memory: Primary Memory is internal memory of the computer. RAM AND ROM both form part of primary memory. The primary memory provides main working space to the computer.The following terms comes under primary memory of a computer are discussed below:

❖ **Random Access Memory (RAM):** The primary storage is referred to as random access memory (RAM) because it is possible to randomly select and use any location of the memory directly store and retrieve data. It takes same time to any address of the memory as the first address. It is also called read/write memory. The storage of data and instructions inside the primary storage is temporary. It disappears from RAM as soon as the power to the computer is switched off. The memories, which lose their content on failure of power supply, are known as volatile memories .So now we can say that RAM is volatile memory.

RAM is of two types

1. Static RAM (SRAM)
2. Dynamic RAM (DRAM)

- **Static RAM (SRAM)**

  The word **static** indicates that the memory retains its contents as long as power remains applied. However, data is lost when the power gets down due to volatile nature. SRAM chips use a matrix of 6-transistors and no capacitors. Transistors do not require power to prevent leakage, so SRAM need not have to be refreshed on a regular basis. Because of the extra space in the matrix, SRAM uses more chips than DRAM for the same amount of storage space, thus making the manufacturing costs higher. Static RAM is used as cache memory needs to be very fast and small.

- **Dynamic RAM (DRAM)**

  DRAM, unlike SRAM, must be continually **refreshed** in order for it to maintain the data. This is done by placing the memory on a refresh circuit that rewrites the data several hundred times per second. DRAM is used for most system memory because it is cheap and small. All DRAMs are made up of memory cells. These cells are composed of one capacitor and one transistor.

- **Read Only Memory (ROM):** There is another memory in computer, which is called Read Only Memory (ROM). Again it is the ICs inside the PC that form the ROM. The storage of program and data in the ROM is permanent. The ROM stores some standard processing programs supplied by the manufacturers to operate the personal computer. The ROM can only be read by the CPU but it cannot be changed. The basic input/output program is stored in the ROM that examines and initializes various equipment attached to the PC when the power switch is ON. The memories, which do not lose their content on failure of power supply, are known as non-volatile memories. ROM is non-volatile memory.

- **PROM:** There is another type of primary memory in computer, which is called Programmable Read Only Memory (PROM). You know that it is not possible to modify or erase programs stored in ROM, but it is possible for you to store your program in PROM chip. Once the programmers' are written it cannot be changed and remain intact even if power is switched off. Therefore programs or instructions written in PROM or ROM cannot be erased or changed.

- **EPROM:** This stands for Erasable Programmable Read Only Memory, which overcome the problem of PROM & ROM. EPROM chip can be programmed time and again by erasing the information stored earlier in it. Information stored in EPROM exposing the chip for some time ultraviolet light and it erases chip is reprogrammed using a special programming facility. When the EPROM is in use information can only be read.

- **Cache Memory:** The speed of CPU is extremely high compared to the access time of main memory. Therefore the performance of CPU decreases due to the slow speed of main memory. To decrease the mismatch in operating speed, a small memory chip is attached between CPU and Main memory whose access time is very close to the processing speed of CPU. It is called CACHE memory. CACHE memories are accessed much faster than conventional RAM. It is used to store programs or data currently being executed or temporary data frequently used by the CPU. So each memory makes main memory to be faster and larger than it really is. It is also very expensive to have bigger size of cache memory and its size is normally kept small.
- **Registers:** The CPU processes data and instructions with high speed; there is also movement of data between various units of computer. It is necessary to transfer the processed data with high speed. So the computer uses a number of special memory units called registers. They are not part of the main memory but they store data or information temporarily and pass it on as directed by the control unit.

2. Secondary Memory / Non-Volatile Memory:

Secondary memory is external and permanent in nature. The secondary memory is concerned with magnetic memory. Secondary memory can be stored on storage media like floppy disks, magnetic disks, magnetic tapes, This memory can also be stored optically on Optical disks - CD-ROM. The following terms comes under secondary memory of a computer are discussed below:

- **Magnetic Tape:** Magnetic tapes are used for large computers like mainframe computers where large volume of data is stored for a longer time. In PC also you can use tapes in the form of cassettes. The cost of storing data in tapes is inexpensive. Tapes consist of magnetic materials that store data permanently. It can be 12.5 mm to 25 mm wide plastic film-type and 500 meter to 1200 meter long which is coated with magnetic material. The deck is connected to the central processor and information is fed into or read from the tape through the processor. It's similar to cassette tape recorder.

- **Magnetic Disk:** You might have seen the gramophone record, which is circular like a disk and coated with magnetic material. Magnetic disks used in computer are made on the same principle. It rotates with very high speed inside the computer drive. Data is stored on both the surface of the disk. Magnetic disks are most popular for direct access storage device. Each disk consists of a number of invisible concentric circles called tracks. Information is recorded on tracks of a disk surface in the form of tiny magnetic spots. The presence of a magnetic spot represents one bit and its absence represents zero bit. The information stored in a disk can be read many times without affecting the stored data. So the reading operation is non-destructive. But if you want to write a new data, then the existing data is erased from the disk and new data is recorded. For Example-Floppy Disk.
- **Optical Disk:** With every new application and software there is greater demand for memory capacity. It is the necessity to store large volume of data that has led to the development of optical disk storage medium. Optical disks can be divided into the following categories:
  - **a) Compact Disk/ Read Only Memory (CD-ROM**
  - **b) Write Once, Read Many (WORM)**
  - **c) Erasable Optical Disk**

Comparison Table RAM & ROM

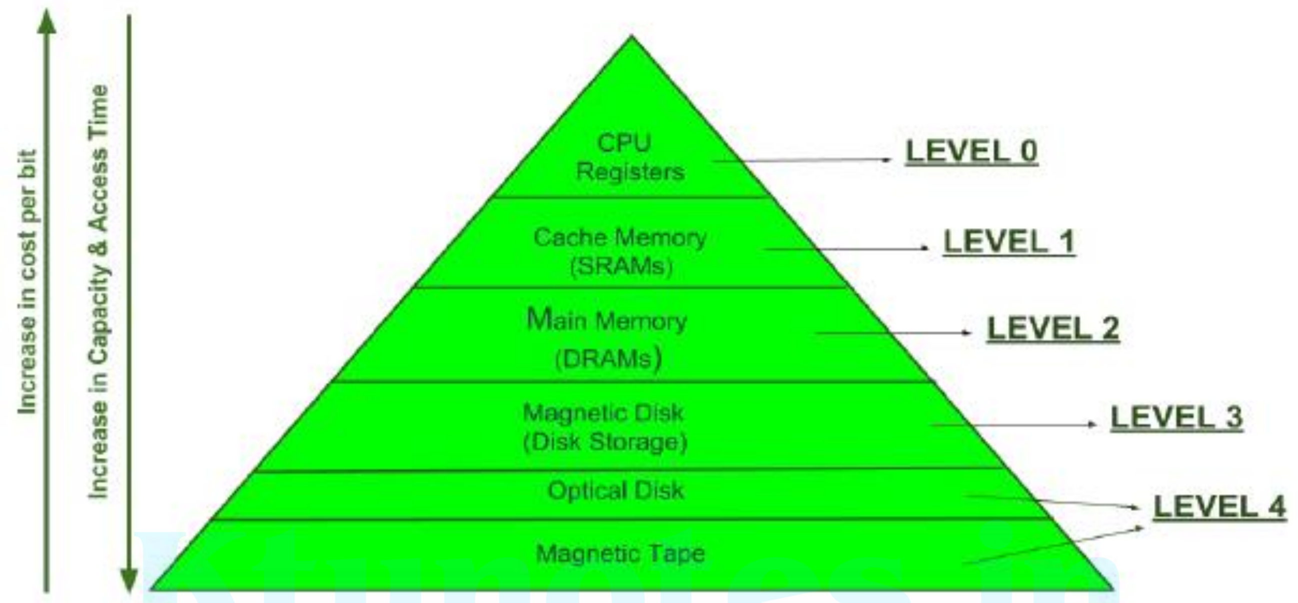
Basis for Comparison	**RAM**	**ROM**
Stands for	Random Access Memory	Read Only Memory
Memory type	Volatile	Non-volatile
Memory capacity	1 to 256 GB per chip	4 to 8 MB per chip
Operation type	Read and Write both.	Only Read.
Speed	Fast	Comparatively slow.
Storage type	Temporary	Permanent
Also referred as	Primary memory	Secondary memory

Basis for Comparison	RAM	ROM
Presence of data according to power source	The stored data in RAM lost in case of power failure.	Data retained in ROM even if the power is turned off.
Accessibility to processor	Processor can directly access the data in RAM.	Processor cannot directly access the data in ROM.
Cost	High	Comparatively low
Types	SRAM and DRAM	PROM, EPROM and EEPROM

Comparison Table SRAM & DRAM

BASIS FOR COMPARISON	SRAM	DRAM
Speed	Faster	Slower
Size	Small	Large
Cost	Expensive	Cheap
Used in	Cache memory	Main memory
Density	Less dense	Highly dense
Construction	Complex and uses transistors and latches.	Simple and uses capacitors and very few transistors.
Single block of memory requires	6 transistors	Only one transistor.
Charge leakage property	Not present	Present hence require power refresh circuitry
Power consumption	Low	High

MEMORY HIERARCHY



Characteristics of Memory Hierarchy are following when we go from top to bottom.

- Capacity in terms of storage increases.
- Cost per bit of storage decreases.
- Frequency of access of the memory by the CPU decreases.
- Access time by the CPU increases

We can infer the following characteristics of Memory Hierarchy Design from above figure:

1. **Capacity:** It is the global volume of information the memory can store. As we move from top to bottom in the Hierarchy, the capacity increases.
2. **Access Time:**
   It is the time interval between the read/write request and the availability of the data. As we move from top to bottom in the Hierarchy, the access time increases.

3. **Performance:**

   Earlier when the computer system was designed without Memory Hierarchy design, the speed gap increases between the CPU registers and Main Memory due to large difference in access time. This results in lower performance of the system and thus, enhancement was required. This enhancement was made in the form of Memory Hierarchy Design because of which the performance of the system increases. One of the most significant ways to increase system performance is minimizing how far down the memory hierarchy one has to go to manipulate data.

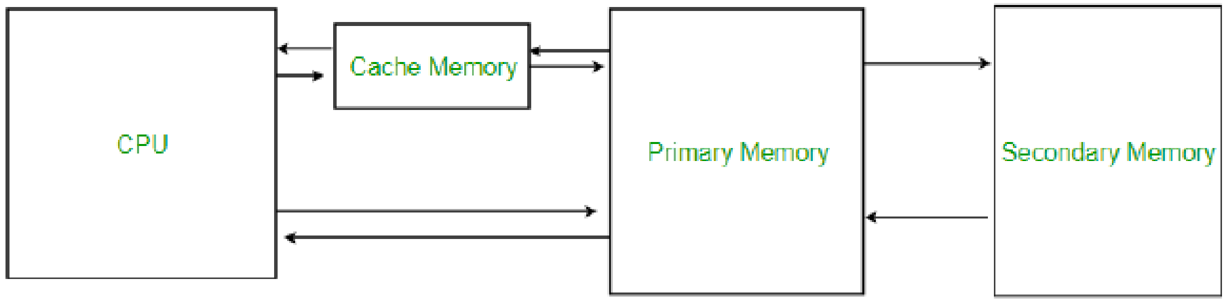
4. **Cost per bit:**

   As we move from bottom to top in the Hierarchy, the cost per bit increases i.e. Internal Memory is costlier than External Memory.

CACHE MEMORY

Cache Memory is a special very high-speed memory. It is used to speed up and synchronizing with high-speed CPU. Cache memory is costlier than main memory or disk memory but economical than CPU registers. Cache memory is an extremely fast memory type that acts as a buffer between RAM and the CPU. It holds frequently requested data and instructions so that they are immediately available to the CPU when needed.

Cache memory is used to reduce the average time to access data from the Main memory. The cache is a smaller and faster memory which stores copies of the data from frequently used main memory locations. There are various different independent caches in a CPU, which store instructions and data.



Levels of memory:

- **Level 1 or Register –** It is a type of memory in which data is stored and accepted that are immediately stored in CPU. Most commonly used register is accumulator, Program counter, address register etc.
- **Level 2 or Cache memory –** It is the fastest memory which has faster access time where data is temporarily stored for faster access.
- **Level 3 or Main Memory –** 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 or Secondary Memory –** It is external memory which is not as fast as main memory but data stays permanently in this memory.

Cache Performance:

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

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

Cache Performance Improvement

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

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

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

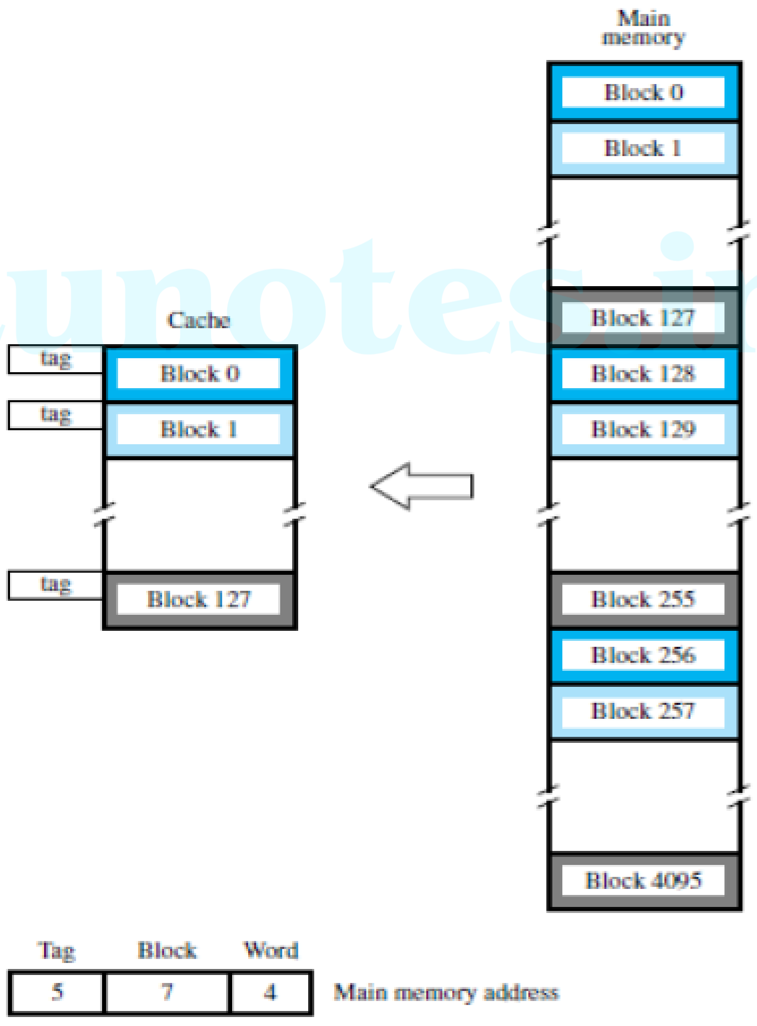
Cache Mapping:

There are three different types of mapping used for the purpose of cache memory which are as follows: Direct mapping, Associative mapping, and Set-Associative mapping. These are explained below.

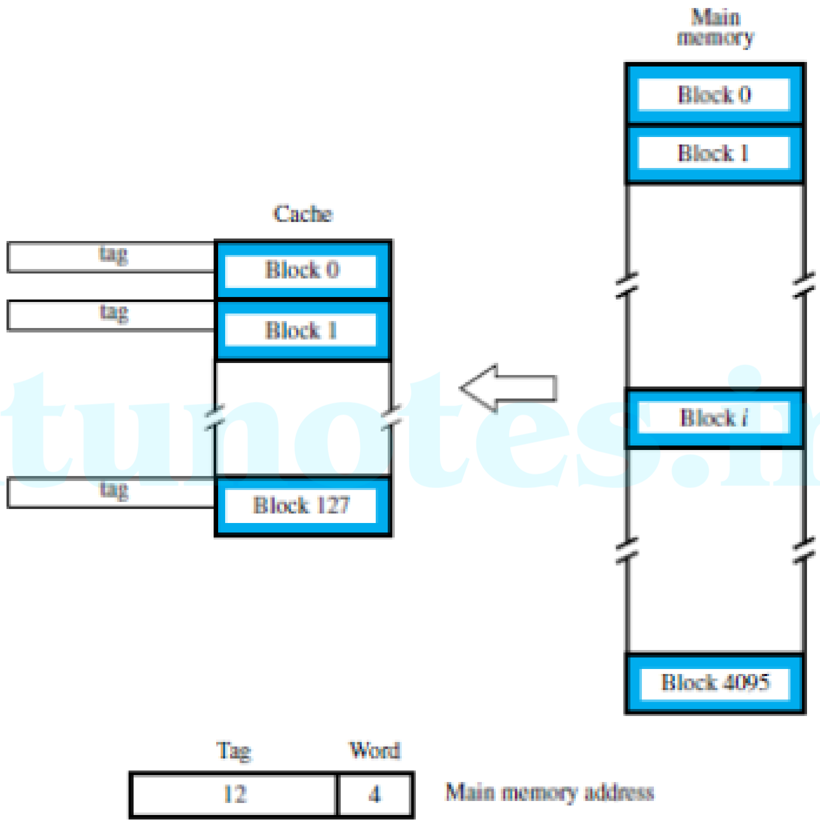
1. **Direct Mapping –** The simplest way to determine cache locations in which to store memory blocks is the *direct-mapping* technique. In this technique, block *j* of the main

memory maps onto block *j* modulo 128 of the cache, as depicted in Figure. Thus, whenever one of the main memory blocks 0, 128, 256, . . . is loaded into the cache, it is stored in cache block 0. Blocks 1, 129, 257, . . . are stored in cache block 1, and so on..

Placement of a block in the cache is determined by its memory address. The memory address can be divided into three fields, as shown in Figure. The low-order 4 bits select one of 16 words in a block. When a new block enters the cache, the 7-bit cache block field determines the cache position in which this block must be stored. If they match, then the desired word is in that block of the cache. If there is no match, then the block containing the required word must first be read from the main memory and loaded into the cache. The direct-mapping technique is easy to implement, but it is not very flexible.

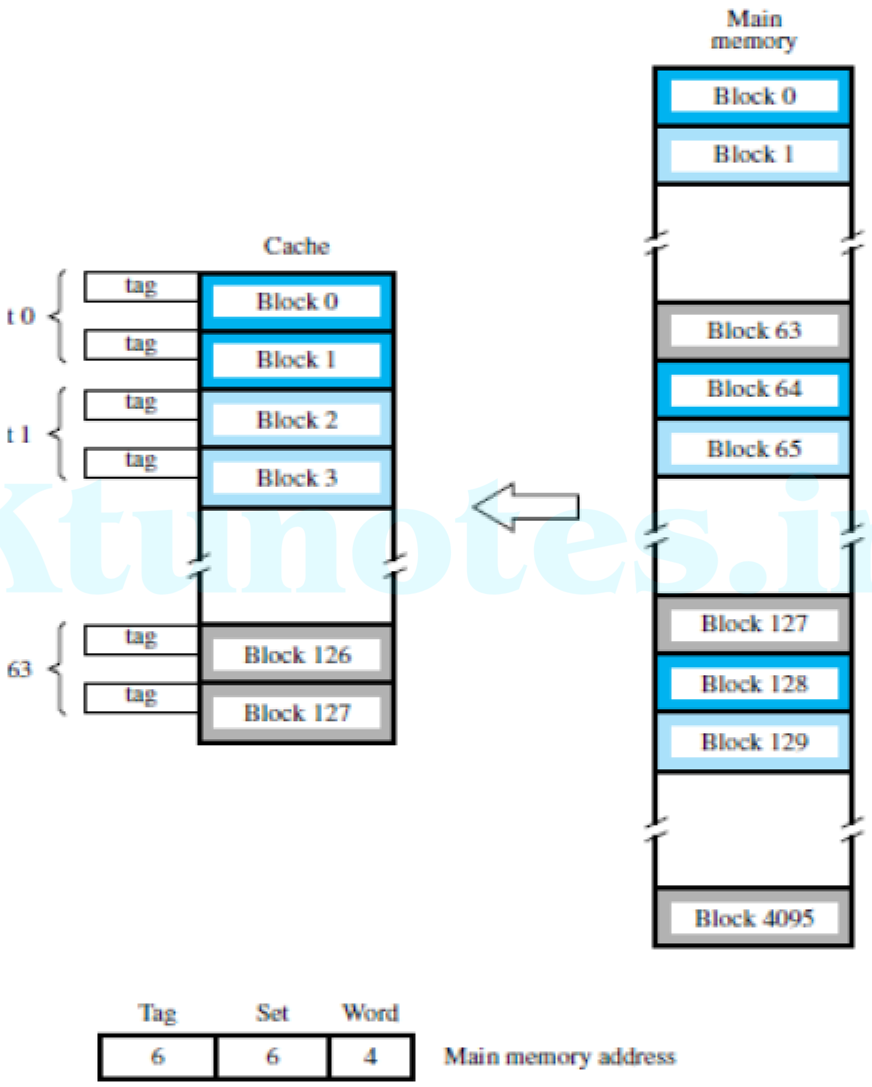


2. **Associative Mapping –** In Associative mapping method, a main memory block can be placed into any cache block position. In this case, 12 tag bits are required to identify a memory block when it is resident in the cache. The tag bits of an address received from the processor are compared to the tag bits of each block of the cache to see if the desired block is present. This is called the *associative-mapping* technique.



It gives complete freedom in choosing the cache location in which to place the memory block, resulting in a more efficient use of the space in the cache. When a new block is brought into the cache, it replaces (ejects) an existing block only if the cache is full. In this case, we need an algorithm to select the block to be replaced. To avoid a long delay, the tags must be searched in parallel. A search of this kind is called an *associative search*.

3. **Set-associative Mapping –** Another approach is to use a combination of the direct- and associative-mapping techniques. The blocks of the cache are grouped into sets, and the mapping allows a block of the main memory to reside in any block of a specific set. Hence, the contention problem of the direct method is eased by having a few choices for block placement.



2-WAY SET ASSOCIATIVE

$$\boldsymbol{Number\ of\ set} = \frac{\boldsymbol{Number\ of\ blocks}}{\mathbf{2}} = \mathbf{128/2=64}$$

Set associative include the combination of Direct & associative Concepts

Direct mapping concept >>>>>> j mod n……….Direct mapping concept

where j= block number in primary memory

n=set number(Block from Primary memory can reside in either block of cache in that set ….associative concept)

Example Block 0 from Primary

0 mod 2 = 0 >>> set 0 ; ie. Block 0 Can reside in either Block in set 0

Example Block 1 from Primary

1 mod 2 =1 >>>> set 1 ; Block 1 Can reside in either Block in set 1

Example Block 2 from Primary

2 mod 2 = 0 >>> set 0 ; ie. Block 2 Can reside in either Block in set 0

At the same time, the hardware cost is reduced by decreasing the size of the associative search. An example of this *set-associative-mapping* technique is shown in Figure for a cache with two blocks per set. In this case, memory blocks 0, 64, 128, . . . , 4032 map into cache set 0, and they can occupy either of the two block positions within this set. Having 64 sets means that the 6-bit set field of the address determines which set of the cache might contain the desired block. The tag field of the address must then be associatively compared to the tags of the two blocks of the set to check if the desired block is present. This two-way associative search is simple to implement. The number of blocks per set is a parameter that can be selected to suit the requirements of a particular computer. For the main memory and cache sizes in Figure, four blocks per set can be accommodated by a 5-bit set field, eight blocks per set by a 4-bit set field, and so on. The extreme condition of 128 blocks per set requires no set bits and corresponds to the fully-associative technique, with 12 tag bits. The other extreme of one block per set is the direct-mapping

Application of Cache Memory

- Usually, the cache memory can store a reasonable number of blocks at any given time, but this number is small compared to the total number of blocks in the main memory.
- The correspondence between the main memory blocks and those in the cache is specified by a mapping function.

Types of Cache

- **Primary Cache –** A primary cache is always located on the processor chip. This cache is small and its access time is comparable to that of processor registers.
- **Secondary Cache –** Secondary cache is placed between the primary cache and the rest of the memory. It is referred to as the level 2 (L2) cache. Often, the Level 2 cache is also housed on the processor chip.

LOCALITY OF REFERENCE

Since size of cache memory is less as compared to main memory. So to check which part of main memory should be given priority and loaded in cache is decided based on locality of reference.

Types of Locality of reference

- **Spatial Locality of reference** This says that there is a chance that element will be present in the close proximity to the reference point and next time if again searched then more close proximity to the point of reference.
- **Temporal Locality of reference** In this Least recently used algorithm will be used. Whenever there is page fault occurs within a word will not only load word in main memory but complete page fault will be loaded because spatial locality of reference rule says that if you are referring any word next word will be referred in its register that's why we load complete page table so the complete block will be loaded.

VIRTUAL MEMORY

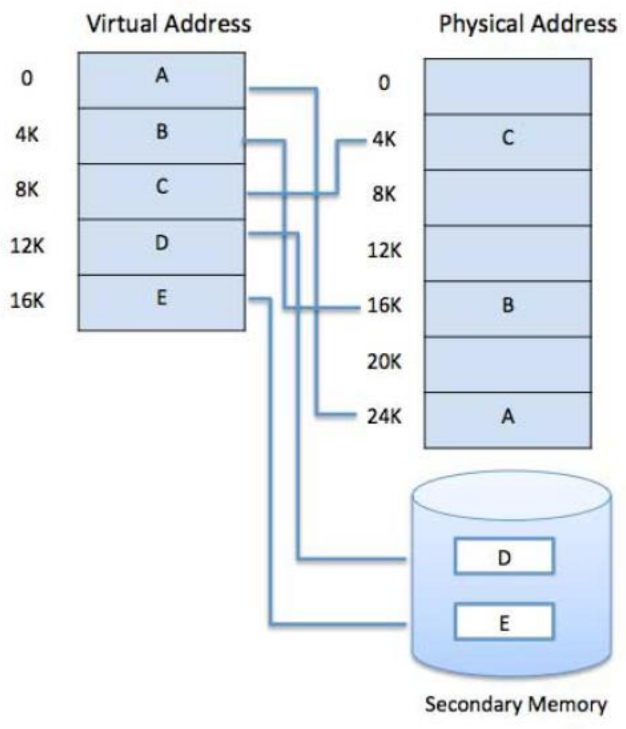
A computer can address more memory than the amount physically installed on the system. This extra memory is actually called virtual memory and it is a section of a hard that's set up to emulate the computer's RAM.

The main visible advantage of this scheme is that programs can be larger than physical memory. Virtual memory serves two purposes. First, it allows us to extend the use of physical memory by using disk. Second, it allows us to have memory protection, because each virtual address is translated to a physical address.

Following are the situations, when entire program is not required to be loaded fully in main memory.

- User written error handling routines are used only when an error occurred in the data or computation.
- Certain options and features of a program may be used rarely.
- Many tables are assigned a fixed amount of address space even though only a small amount of the table is actually used.
- The ability to execute a program that is only partially in memory would counter many benefits.
- Less number of I/O would be needed to load or swap each user program into memory.
- A program would no longer be constrained by the amount of physical memory that is available.
- Each user program could take less physical memory, more programs could be run the same time, with a corresponding increase in CPU utilization and throughput.

Modern microprocessors intended for general-purpose use, a memory management unit, or MMU, is built into the hardware. The MMU's job is to translate virtual addresses into physical addresses. A basic example is given below:

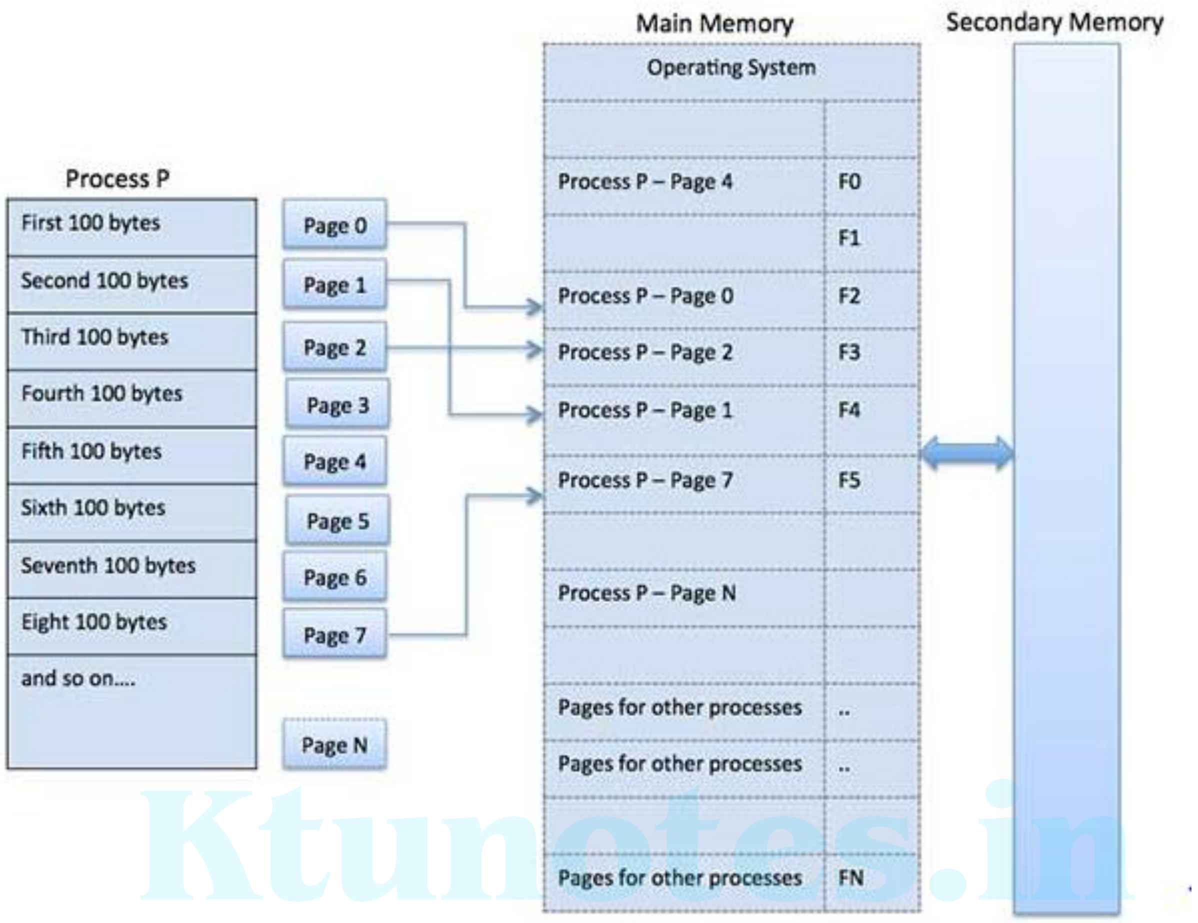


Paging

A computer can address more memory than the amount physically installed on the system. This extra memory is actually called virtual memory and it is a section of a hard that's set up to emulate the computer's RAM. Paging technique plays an important role in implementing virtual memory.

Paging is a memory management technique in which process address space is broken into blocks of the same size called **pages** (size is power of 2, between 512 bytes and 8192 bytes). The size of the process is measured in the number of pages.

Similarly, main memory is divided into small fixed-sized blocks of (physical) memory called **frames** and the size of a frame is kept the same as that of a page to have optimum utilization of the main memory and to avoid external fragmentation.



ADDRESS TRANSLATION

Page address is called **logical address** and represented by **page number** and the **offset**.

Logical Address = Page number + page offset

Frame address is called **physical address** and represented by a **frame number** and the **offset**.

Physical Address = Frame number + page offset

A data structure called **page map table** is used to keep track of the relation between a page of a process to a frame in physical memory.

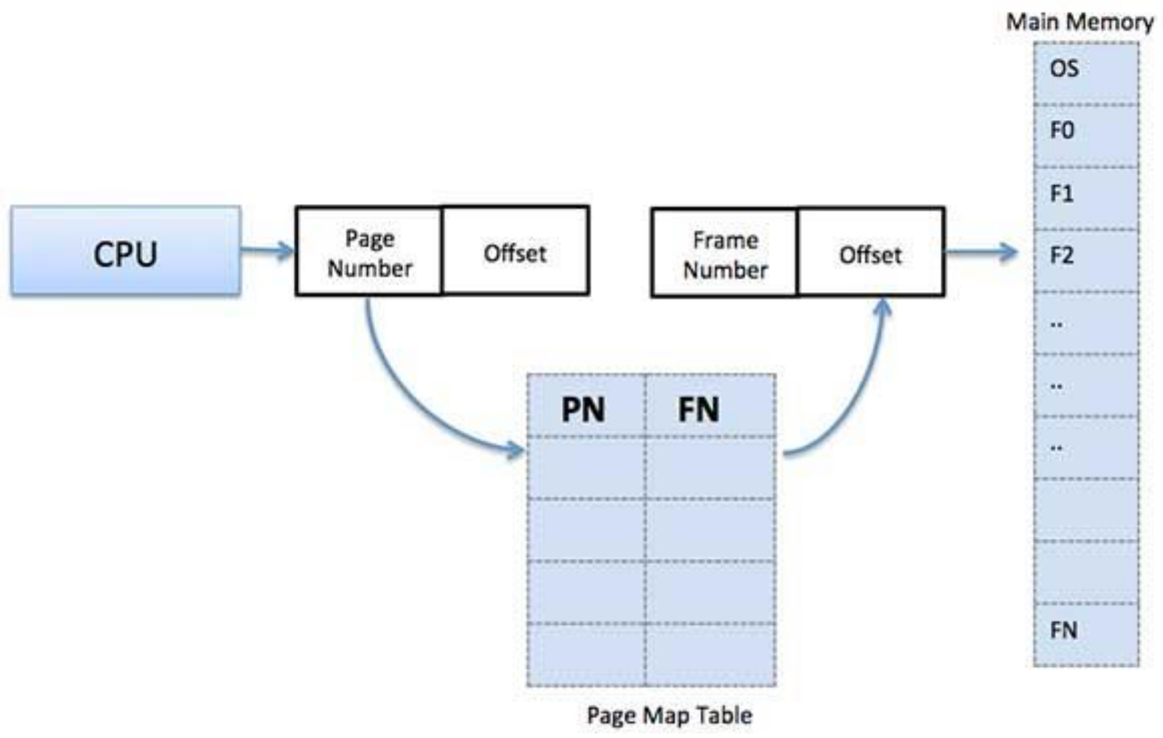


Fig: Paging Hardware

When the system allocates a frame to any page, it translates this logical address into a physical address and create entry into the page table to be used throughout execution of the program.

When a process is to be executed, its corresponding pages are loaded into any available memory frames. Suppose you have a program of 8Kb but your memory can accommodate only 5Kb at a given point in time, then the paging concept will come into picture. When a computer runs out of RAM, the operating system (OS) will move idle or unwanted pages of memory to secondary memory to free up RAM for other processes and brings them back when needed by the program.

This process continues during the whole execution of the program where the OS keeps removing idle pages from the main memory and write them onto the secondary memory and bring them back when required by the program.

Advantages and Disadvantages of Paging

Here is a list of advantages and disadvantages of paging

- Paging reduces external fragmentation, but still suffer from internal fragmentation.
- Paging is simple to implement and assumed as an efficient memory management technique.
- Due to equal size of the pages and frames, swapping becomes very easy.
- Page table requires extra memory space, so may not be good for a system having small RAM.

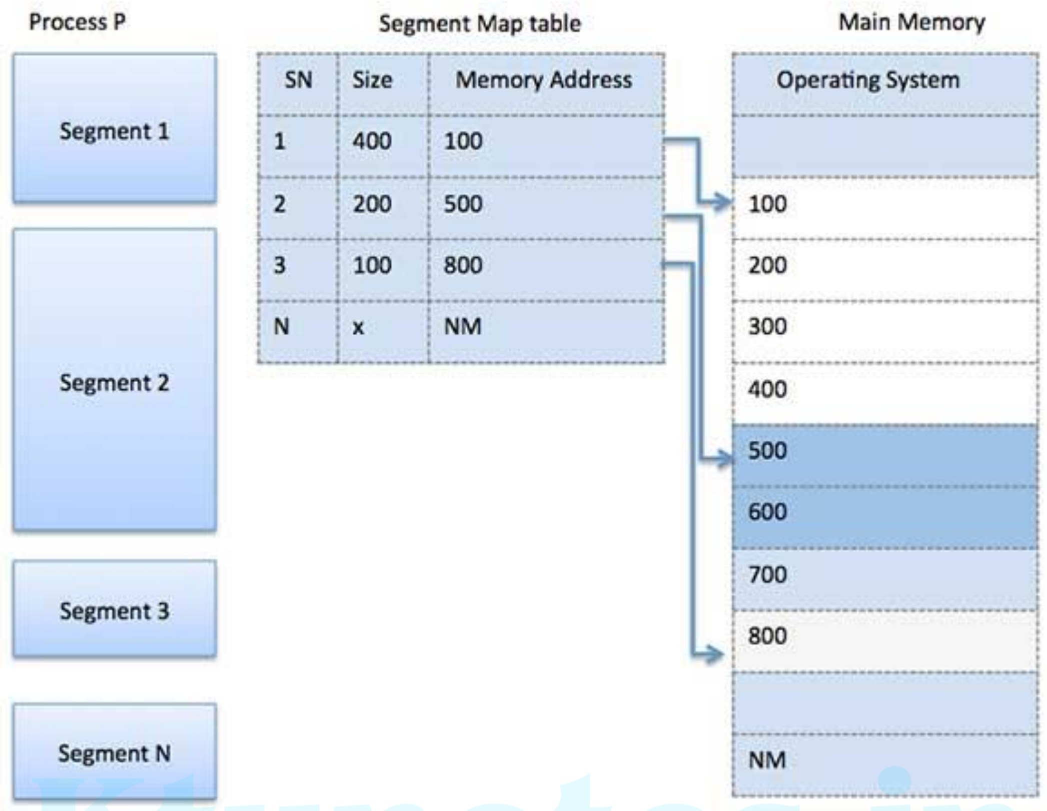
Segmentation

Segmentation is a memory management technique in which each job is divided into several segments of different sizes, one for each module that contains pieces that perform related functions. Each segment is actually a different logical address space of the program.

When a process is to be executed, its corresponding segmentation are loaded into non-contiguous memory though every segment is loaded into a contiguous block of available memory.

Segmentation memory management works very similar to paging but here segments are of variable-length where as in paging pages are of fixed size.

A program segment contains the program's main function, utility functions, data structures, and so on. The operating system maintains a **segment map table** for every process and a list of free memory blocks along with segment numbers, their size and corresponding memory locations in main memory. For each segment, the table stores the starting address of the segment and the length of the segment. A reference to a memory location includes a value that identifies a segment and an offset.



Comparison between Paging & Segmentation

SL NO	PAGING	SEGMENTATION
1	In paging, program is divided into fixed or mounted size pages.	In segmentation, program is divided into variable size sections.
2	For paging operating system is accountable.	For segmentation compiler is accountable.
3	Page size is determined by hardware.	Here, the section size is given by the user.
4	It is faster in the comparison of segmentation.	Segmentation is slow.
5	Paging could result in internal fragmentation.	Segmentation could result in external fragmentation.
6	In paging, logical address is split into page number and page offset.	Here, logical address is split into section number and section offset.
7	Paging comprises a page table which encloses the base address of every page.	While segmentation also comprises the segment table which encloses segment number and segment offset.

8	Page table is employed to keep up the page data.	Section Table maintains the section data.
9	In paging, operating system must maintain a free frame list.	In segmentation, operating system maintain a list of holes in main memory.
10	Paging is invisible to the user	Segmentation is visible to the user.
11	In paging, processor needs page number, offset to calculate absolute address.	In segmentation, processor uses segment number, offset to calculate full address.

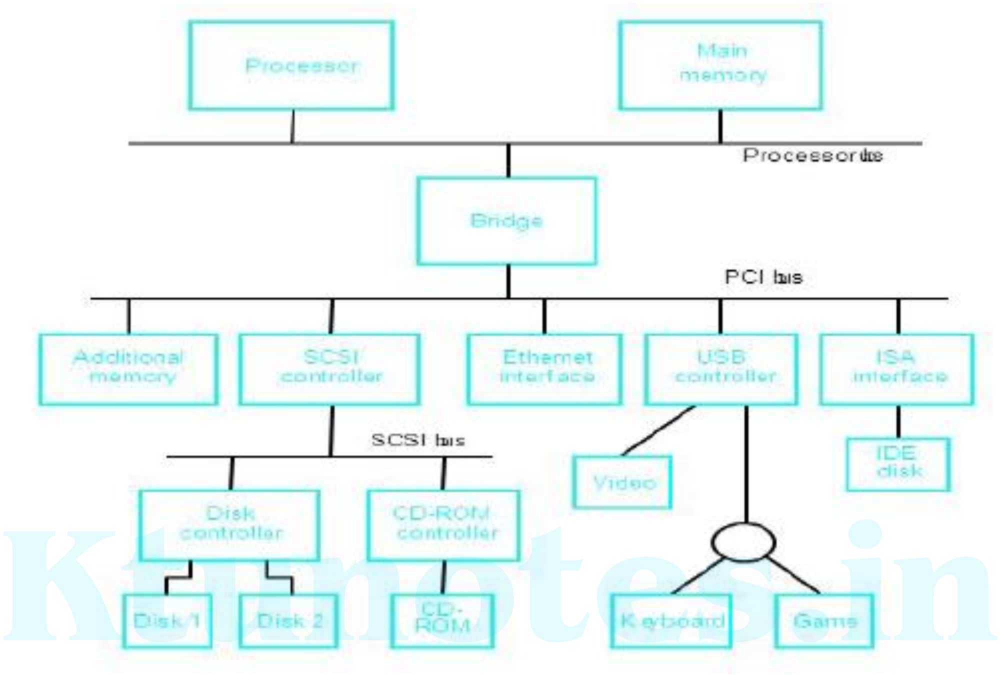
INPUT/OUTPUT ORGANIZATION

The processor bus is the bus defied by the signals on the processor chip itself. Devices that require a very high-speed connection to the processor, such as the main memory, may be connected directly to this bus. For electrical reasons, only a few devices can be connected in this manner. The motherboard usually provides another bus that can support more devices. The two buses are interconnected by a circuit, which we will call a bridge, that translates the signals and protocols of one bus into those of the other. Devices connected to the expansion bus appear to the processor as if they were connected directly to the processor's own bus. The only difference is that the bridge circuit introduces a small delay in data transfers between the processor and those devices.

It is not possible to define a uniform standard for the processor bus. The structure of this bus is closely tied to the architecture of the processor. It is also dependent on the electrical characteristics of the processor chip, such as its clock speed. The expansion bus is not subject to these limitations, and therefore it can use a standardized signaling scheme. A number of standards have been developed. Some have evolved by default, when a particular design became commercially successful. For example, IBM developed a bus they called ISA (Industry Standard Architecture) for their personal computer known at the time as PC AT.

Some standards have been developed through industrial cooperative efforts, even among competing companies driven by their common self-interest in having compatible products. In some cases, organizations such as the IEEE (Institute of Electrical and Electronics Engineers), ANSI (American National Standards Institute), or international bodies such as ISO (International Standards Organization) have blessed these standards and given them an official status.

A given computer may use more than one bus standards. A typical Pentium computer has both a PCI bus and an ISA bus, thus providing the user with a wide range of devices to choose from.



Peripheral Component Interconnect (PCI) Bus:-

The PCI bus is a good example of a system bus that grew out of the need for standardization. It supports the functions found on a processor bus bit in a standardized format that is independent of any particular processor. Devices connected to the PCI bus appear to the processor as if they were connected directly to the processor bus. They are assigned addresses in the memory address space of the processor.

The PCI follows a sequence of bus standards that were used primarily in IBM PCs. Early PCs used the 8-bit XT bus, whose signals closely mimicked those of Intel's 80x86 processors. Later, the 16-bit bus used on the PC At computers became known as the ISA bus. Its extended 32-bit version is known as the EISA bus. Other buses developed in the eighties with similar capabilities are the Microchannel used in IBM PCs and the NuBus used in Macintosh computers.

The PCI was developed as a low-cost bus that is truly processor independent. Its design anticipated a rapidly growing demand for bus bandwidth to support high-speed disks and graphic and video devices, as well as the specialized needs of multiprocessor systems. As a result, the PCI is still popular as an industry standard almost a decade after it was first introduced in 1992.

An important feature that the PCI pioneered is a plug-and-play capability for connecting I/O devices. To connect a new device, the user simply connects the device interface board to the bus. The software takes care of the rest.

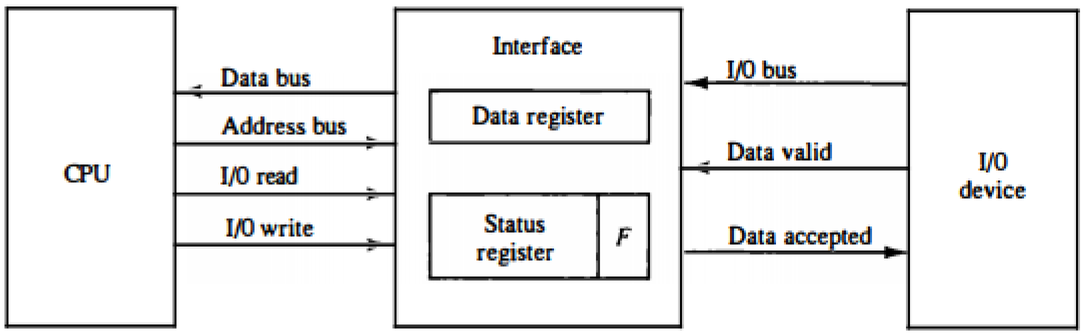
DATA TRANSFER

Programmed I/O

In this mode of data transfer the operations are the results in I/O instructions which is a part of computer program. Each data transfer is initiated by a instruction in the program. Normally the transfer is from a CPU register to peripheral device or vice-versa. Once the data is initiated the CPU starts monitoring the interface to see when next transfer can made. The instructions of the program keep close tabs on everything that takes place in the interface unit and the I/O devices. The transfer of data requires three instructions:

- ❖ Read the status register.
- ❖ Check the status of the flag bit and branch to step 1 if not set or to step 3 if set.
- ❖ Read the data register.

Figure Data transfer from I/O device to CPU.



F = Flag bit

In this technique CPU is responsible for executing data from the memory for output and storing data in memory for executing of Programmed I/O as shown in Fig. Drawback of the Programmed I/O: The main drawback of the Program Initiated I/O was that the CPU has to monitor the units all the times when the program is executing. Thus the CPU stays in a program loop until the I/O unit indicates that it is ready for data transfer. This is a time consuming process and the CPU time is wasted a lot in keeping an eye to the executing of program.

Interrupt Driven I/O

In this method an interrupt facility an interrupt command is used to inform the device about the start and end of transfer. In the meantime the CPU executes other program. When the interface determines that the device is ready for data transfer it generates an Interrupt Request and sends it to the computer. When the CPU receives such an signal, it temporarily stops the execution of the program and branches to a service program to process the I/O transfer and after completing it returns back to task, what it was originally performing. In this type of IO, computer does not check the flag. It continues to perform its task. Whenever any device wants the attention, it sends the interrupt signal to the CPU.CPU then deviates from what it was doing, store the return address from PC and branch to the address of the subroutine. There are two ways of choosing the branch address:

Vectored Interrupt: *Vectored Interrupts* are those which have fixed vector address (starting address of sub-routine) and after executing these, program control is transferred to that address.

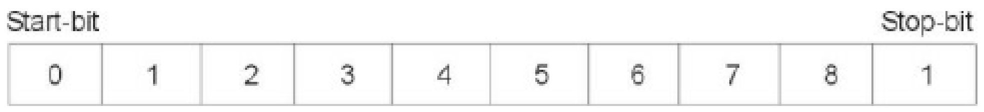
Non-vectored Interrupt: *Non-Vectored Interrupts* are those in which vector address is not predefined. The interrupting device gives the address of sub-routine for these interrupts

ASYNCHRONOUS TRANSMISSION

Asynchronous transmission works in spurts and must insert a start bit before each data character and a stop bit at its termination to inform the receiver where it begins and ends.

The term *asynchronous* is used to describe the process where transmitted data is encoded with start and stop bits, specifying the beginning and end of each character.

An example of asynchronous transmission is shown in the following figure.

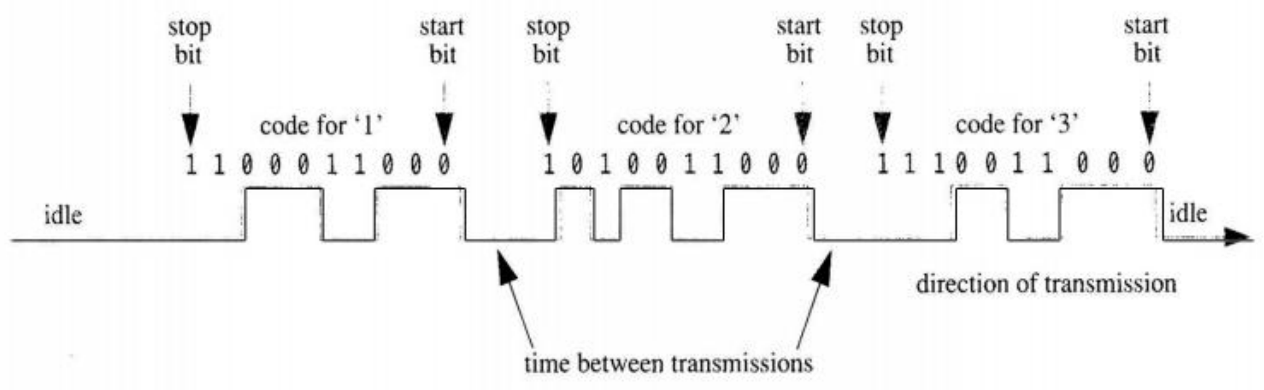


Start-bit									Stop-bit
0	1	2	3	4	5	6	7	8	1

These additional bits provide the timing or synchronization for the connection by indicating when a complete character has been sent or received; thus, timing for each character begins with the start bit and ends with the stop bit.

When gaps appear between character transmissions, the asynchronous line is said to be in a mark state. A mark is a binary 1 (or negative voltage) that is sent during periods of inactivity on the line as shown in the following figure

Asynchronous Transmission of the Digits 1, 2, and 3 Using NRZ Coding



When the mark state is interrupted by a positive voltage (a binary 0), the receiving system knows that data characters are going to follow. It is for this reason that the start bit, which precedes the data character bit, which signals the end of a character, is always a mark bit (binary 1).

The following is a list of characteristics specific to asynchronous communication:

- ❖ Each character is preceded by a start bit 0
- ❖ Gaps or spaces between characters may exist.

With asynchronous transmission, a large text document is organized into long strings of letters (or characters) that make up the words within the sentences and paragraphs. These characters are sent over the communication link one at a time and reassembled at the remote location.

In asynchronous transmission, ASCII character would actually be transmitted using 10 bits. For example, "0100 0001" would become "**1** 0100 0001 **0**". The extra one (or zero, depending on parity bit) at the start and end of the transmission tells the receiver first that a character is coming and secondly that the character has ended. This method of transmission is used when data are sent intermittently as opposed to in a solid stream. In the previous example the start and stop bits are in bold.

The start and stop bits must be of opposite polarity. This allows the receiver to recognize when the second packet of information is being sent.

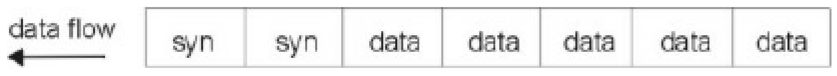
Asynchronous transmission is used commonly for communications over telephone lines.

SYNCHRONOUS TRANSMISSION

The term *synchronous* is used to describe a continuous and timed bound /clock based transfer of data blocks.

- ❖ It is a data transfer method in which a continuous stream of data signals is accompanied by timing signals (generated by an electronic clock) to ensure that the transmitter and the receiver are in step (synchronized) with one another.
- ❖ The data is sent in blocks (called frames or packets) spaced by fixed time intervals
- ❖ Synchronous transmission modes are used when large amounts of data must be transferred very quickly from one location to the other.
- ❖ Synchronous transmission synchronizes transmission speeds at both the receiving and sending end of the transmission by using clock signals.
- ❖ A continual stream of data is then sent between the two nodes.

The data blocks are grouped and spaced in regular intervals and are preceded by special characters called synchronous idle characters. See the following illustration



After the syn characters are received by the remote device, they are decoded and used to synchronize the connection. After the connection is correctly synchronized, data transmission may begin. An analogy of synchronous transmission would be the transmission of a large text document. Before the document is transferred across the synchronous line, it is first broken into blocks of sentences or paragraphs. The blocks are then sent over the communication link to the remote site.

The timing needed for synchronous connections is obtained from the devices located on the communication link. All devices on the synchronous link must be set to the same clocking.

The following is a list of characteristics specific to synchronous communication:

- There are no gaps between characters being transmitted.
- Timing is supplied by modems or other devices at each end of the connection.
- Special syn characters precede the data being transmitted.
- The syn characters are used between blocks of data for timing purposes

Due to there being no start and stop bits the data transfer rate is quicker although more errors will occur, as the clocks will eventually get out of sync, and the receiving device would have the wrong time that had been agreed in protocol for sending /receiving data, so some bytes could become corrupted (by losing bits).

Ways to get around this problem include re-check digits to ensure the bytes is correctly interpreted a protocols (such as Ethernet, SONET, and Token Ring) use synchronous transmission.

Direct Memory Access:

The data transfer between a fast storage media such as magnetic disk and memory unit is limited by the speed of the CPU. Thus we can allow the peripherals directly communicate with each other using the memory buses, removing the intervention of the CPU. This type of data transfer technique is known as DMA or direct memory access. During DMA the CPU is idle and it has no control over the memory buses. The DMA controller takes over the buses to manage the transfer directly between the I/O devices and the memory unit.

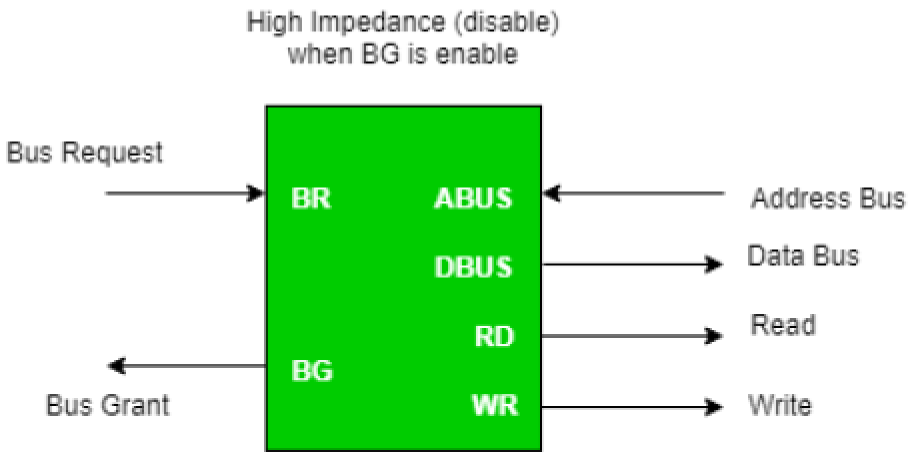


Figure - CPU Bus Signals for DMA Transfer

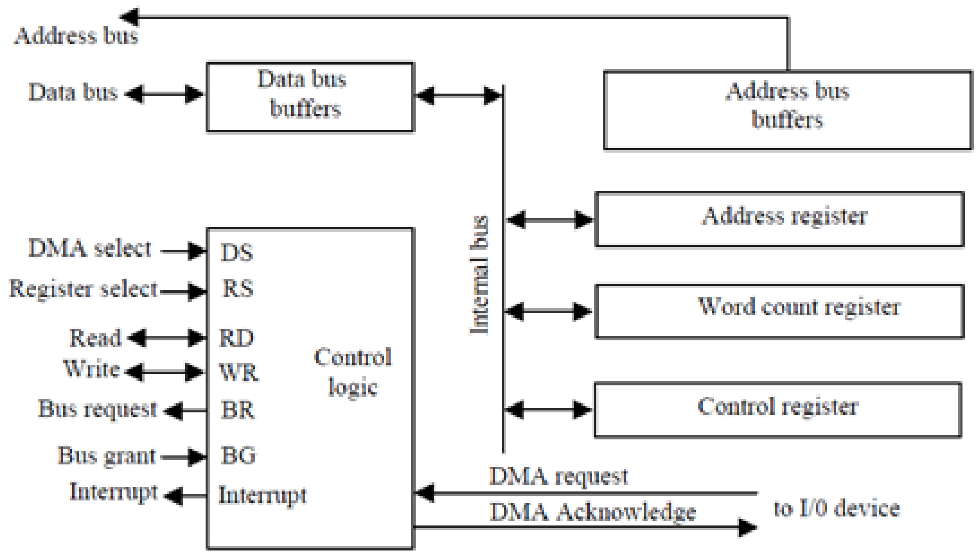
Bus Request : It is used by the DMA controller to request the CPU to relinquish the control of the buses.

Bus Grant : It is activated by the CPU to Inform the external DMA controller that the buses are in high impedance state and the requesting DMA can take control of the buses. Once the DMA has taken the control of the buses it transfers the data. This transfer can take place in many ways.

Types of DMA transfer using DMA controller:

DMA Controller: The DMA controller needs the usual circuits of an interface to communicate with the CPU and I/O device. The DMA controller has three registers:

- Address Register
- Word Count Register
- Control Register



Address Register: Address Register contains an address to specify the desired location in memory. **Word Count Register:** WC holds the number of words to be transferred. The register is incre/decre by one after each word transfer and internally tested for zero. **Control Register:** Control Register specifies the mode of transfer The unit communicates with the CPU via the data bus and control lines. The registers in the DMA are selected by the CPU through the address bus by enabling the DS (DMA select) and RS (Register select) inputs. The RD (read) and WR (write) inputs are bidirectional. When the BG (Bus Grant) input is 0, the CPU can communicate with the DMA registers through the data bus to read from or write to the DMA registers. When BG =1, the DMA can communicate directly with the memory by specifying an address in the address bus and activating the RD or WR control.

Burst Transfer : DMA returns the bus after complete data transfer. A register is used as a byte count, being decremented for each byte transfer, and upon the byte count reaching zero, the DMAC will release the bus. When the DMAC operates in burst mode, the CPU is halted for the duration of the data transfer. Steps involved are:

- Bus grant request time.
- Transfer the entire block of data at transfer rate of device because the device is usually slow than the speed at which the data can be transferred to CPU.
- Release the control of the bus back to CPU So, total time taken to transfer the N bytes = Bus grant request time + (N) * (memory transfer rate) + Bus release control time.

Where,

X µsec =data transfer time or preparation time (words/block)

Y µsec =memory cycle time or cycle time or transfer time (words/block)

% CPU idle (Blocked)=(Y/X+Y)*100

% CPU Busy=(X/X+Y)*100

Cyclic Stealing : An alternative method in which DMA controller transfers one word at a time after which it must return the control of the buses to the CPU. The CPU delays its operation only for one memory cycle to allow the direct memory I/O transfer to "steal" one memory cycle.

Steps Involved are:

1. Buffer the byte into the buffer
2. Inform the CPU that the device has 1 byte to transfer (i.e. bus grant request)
3. Transfer the byte (at system bus speed)
4. Release the control of the bus back to CPU.

Before moving on transfer next byte of data, device performs step 1 again so that bus isn't tied up and the transfer won't depend upon the transfer rate of device. So, for 1 byte of transfer of data, time taken by using cycle stealing mode (T). = time required for bus grant + 1 bus cycle to transfer data + time required to release the bus, it will be N x T

In cycle stealing mode we always follow pipelining concept that when one byte is getting transferred then Device is parallel preparing the next byte. "The fraction of CPU time to the data transfer time" if asked then cycle stealing mode is used.

Where,

X µsec =data transfer time or preparation time

(words/block)

Y µsec =memory cycle time or cycle time or transfer time (words/block)

% CPU idle (Blocked) =(Y/X)*100

% CPU busy=(X/Y)*100

Interleaved mode: In this technique , the DMA controller takes over the system bus when the microprocessor is not using it. An alternate half cycle i.e. half cycle DMA + half cycle processor.