



Module 5

Content

- ? What is Computer Memory?
- ? Why is memory important or needed for a computer?
- ? Types of Computer Memory
 - ? Primary Memory
 - ? Secondary Memory
- ? Primary Vs Secondary Memory
- ? Memory Units
- ? References

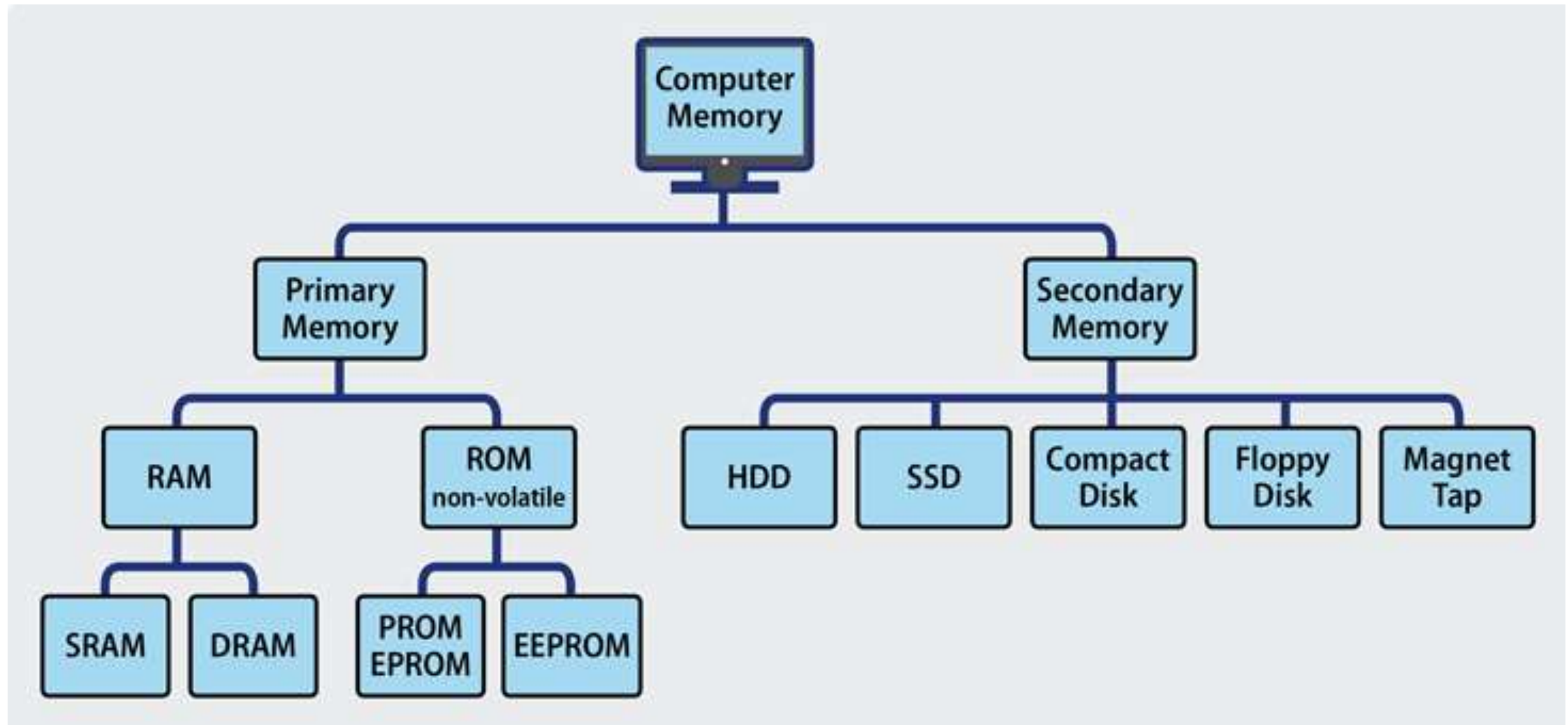
What is Computer Memory

- ? Computer **memory** is any physical device capable of storing information temporarily, like RAM (random access memory), or permanently, like ROM (read-only memory).
- ? Memory devices utilize integrated circuits and are used by operating systems, software, and hardware.

Why is memory important or needed for a computer?

- ? Each device in a computer operates at different speeds and computer memory gives your computer a place to quickly access data.
- ? If the CPU had to wait for a secondary storage device, like a hard disk drive, a computer would be much slower.

Types of Computer Memory



Primary Memory

- ? This is the main memory of the computer. CPU can directly read or write on this memory. It is fixed on the motherboard of the computer.
- ? Primary memory is further divided in two types:
 - 1.RAM(Random Access Memory)
 - 2.ROM(Read Only Memory)

RAM(Random Access Memory)

- ? RAM is a temporary memory. The information stored in this memory is lost as the power supply to the computer is turned off. That's why it is also called **Volatile Memory**.
- ? It stores the data and instruction given by the user and also the results produced by the computer temporarily

ROM(Read only Memory)

- ? Information stored in ROM is permanent in nature,i.e., it holds the data even if the system is switched off.
- ? It holds the starting instructions for the computer. ROM cannot be overwritten by the computer. It is also called Non-Volatile Memory.

Secondary Memory

- ? This memory is permanent in nature. It is used to store the different programs and the information permanently (which were temporarily stored in RAM). It holds the information till we erase it.
- ? **Different types of secondary storage devices are:**
 1. Hard Disc, Compact Disc,
 2. DVD, Pen Drive,
 3. Flash Drive, etc.

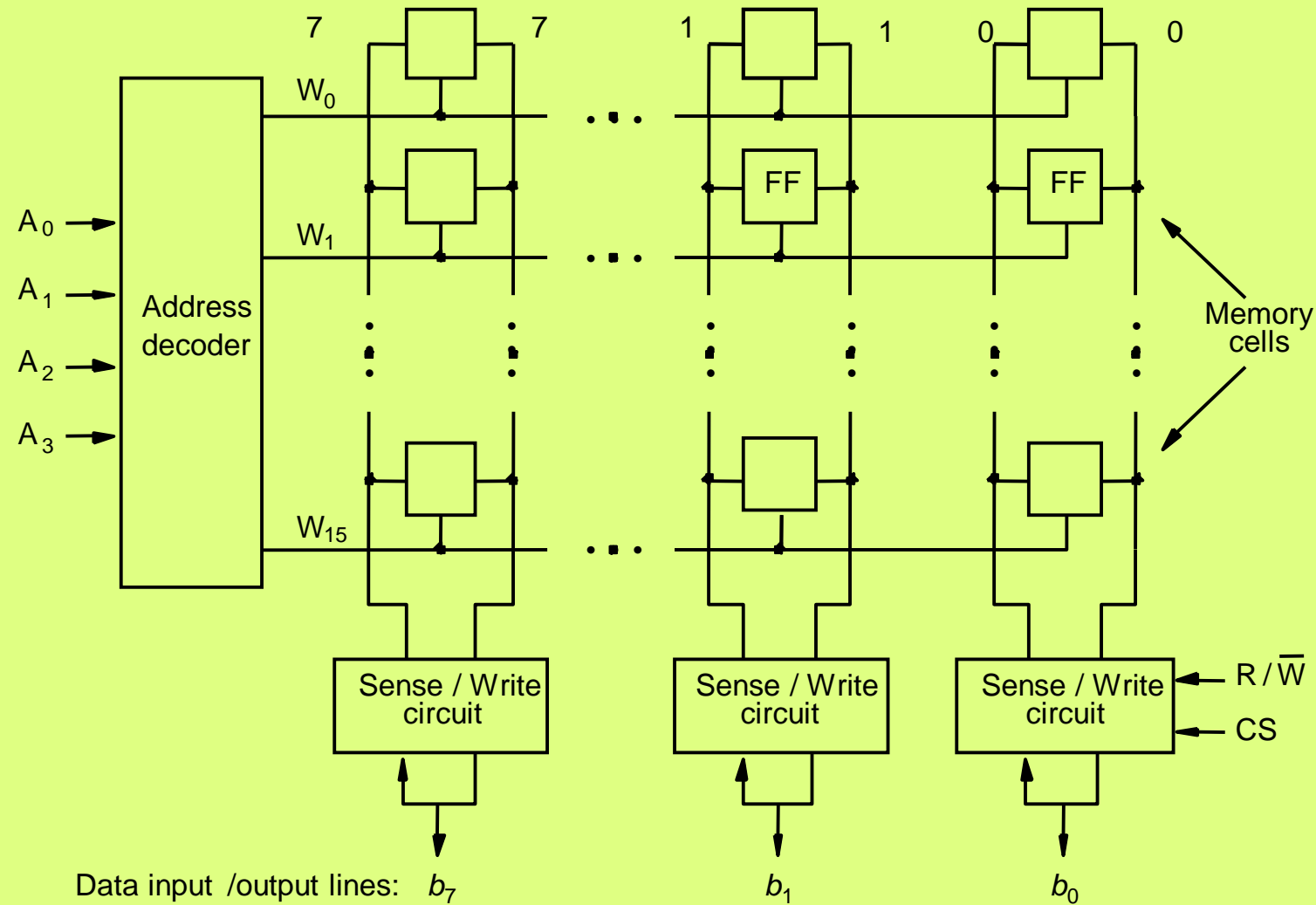
Primary Vs Secondary Memory

Parameter	Primary memory	Secondary memory
Nature	The primary memory is categorized as volatile & nonvolatile memories.	The secondary memory is always a non-volatile memory.
Alias	These memories are also called internal memory.	Secondary memory is known as a Backup memory or Additional memory or Auxiliary memory.
Access	Data is directly accessed by the processing unit.	Data cannot be accessed directly by the processor. It is first copied from secondary memory to primary memory. Only then CPU can access it.
Formation	It's a volatile memory meaning data cannot be retained in case of power failure.	It's a non-volatile memory so that that data can be retained even after power failure.

Internal organization of memory chips

- ? Each memory cell can hold one bit of information.
- ? Memory cells are organized in the form of an array.
- ? One row is one memory word.
- ? All cells of a row are connected to a common line, known as the “word line”.
- ? Word line is connected to the address decoder.
- ? Sense/write circuits are connected to the data input/output lines of the memory chip.

Internal organization of memory chips (Contd.,)



RAM Types

? Static RAMs (SRAMs):

- ? Consist of circuits that are capable of retaining their state as long as the power is applied.
- ? Volatile memories, because their contents are lost when power is interrupted.
- ? Access times of static RAMs are in the range of few nanoseconds.
- ? However, the cost is usually high.

? Dynamic RAMs (DRAMs):

- ? Do not retain their state indefinitely.
- ? Contents must be periodically refreshed.
- ? Contents may be refreshed while accessing them for reading.

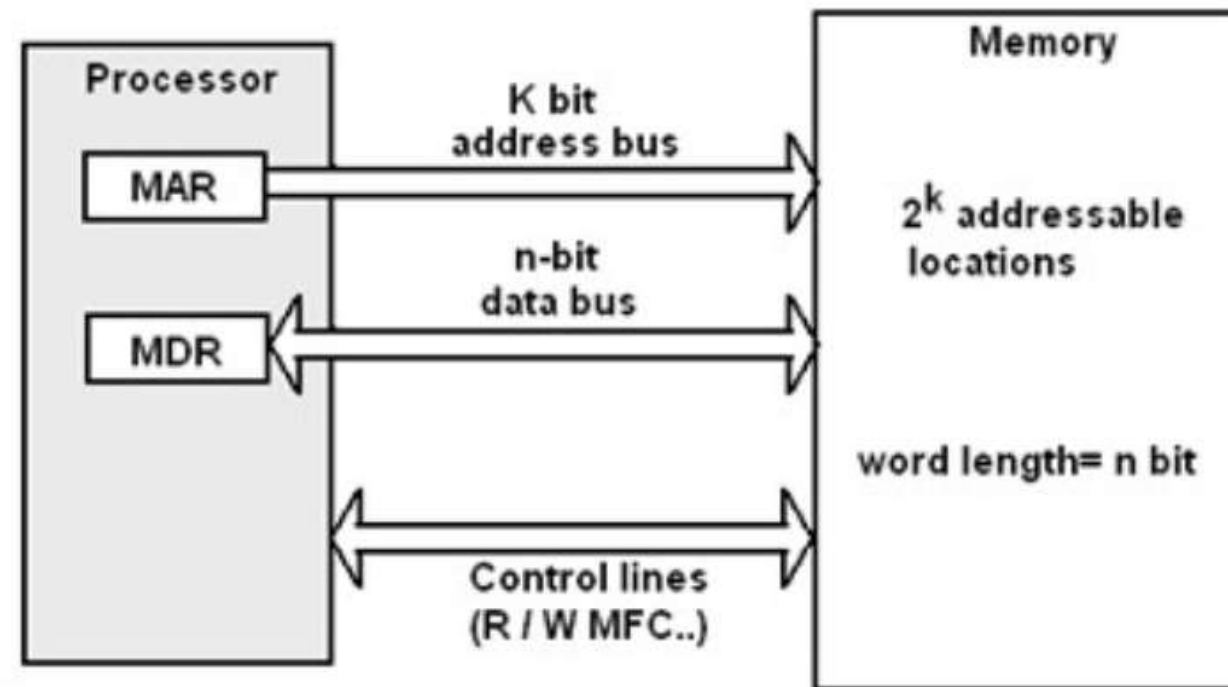
Memory Units

- ? Data in the computer's memory is represented by the two digits 0 and 1.
- ? These two digits are called **Binary Digits** or **Bits**.
- ? A bit is the smallest unit of computer's memory.
- ? Bits=0,1
 - 1 Byte= 8 bits(e.g,11001011)
 - 1 KB(kilobyte) = 1024 Bytes
 - 1 MB(megabyte) = 1024 KB
 - 1 GB(Gigabyte) = 1024 MB
 - 1 TB(Terabyte) = 1024 GB

- The **memory** is divided into a large **number** of **small parts** called **cells**. Each location or cell has a unique address which varies from zero to memory size minus one. For example if computer has 64k words, then this memory unit has $64 * 1024 = 65536$ memory location. The address of these locations varies from 0 to 65535.
- The **maximum size** of the **Main Memory (MM)** that can be used in any computer is **determined by its addressing scheme**. For example: **16-bit computer** that **generates 16-bit addresses** is capable of addressing up to $2^{16} = 64K$ memory locations.
- If a machine **generates 32-bit addresses**, it can access up to $2^{32} = 4GB$ memory locations. This number represents the **size of address space** of the computer.

Data transfer between the main memory & CPU register takes place through **2 registers** namely

- **MAR (Memory Address Register)**
- **MDR (Memory Data Register).**



Memory Hierarchy

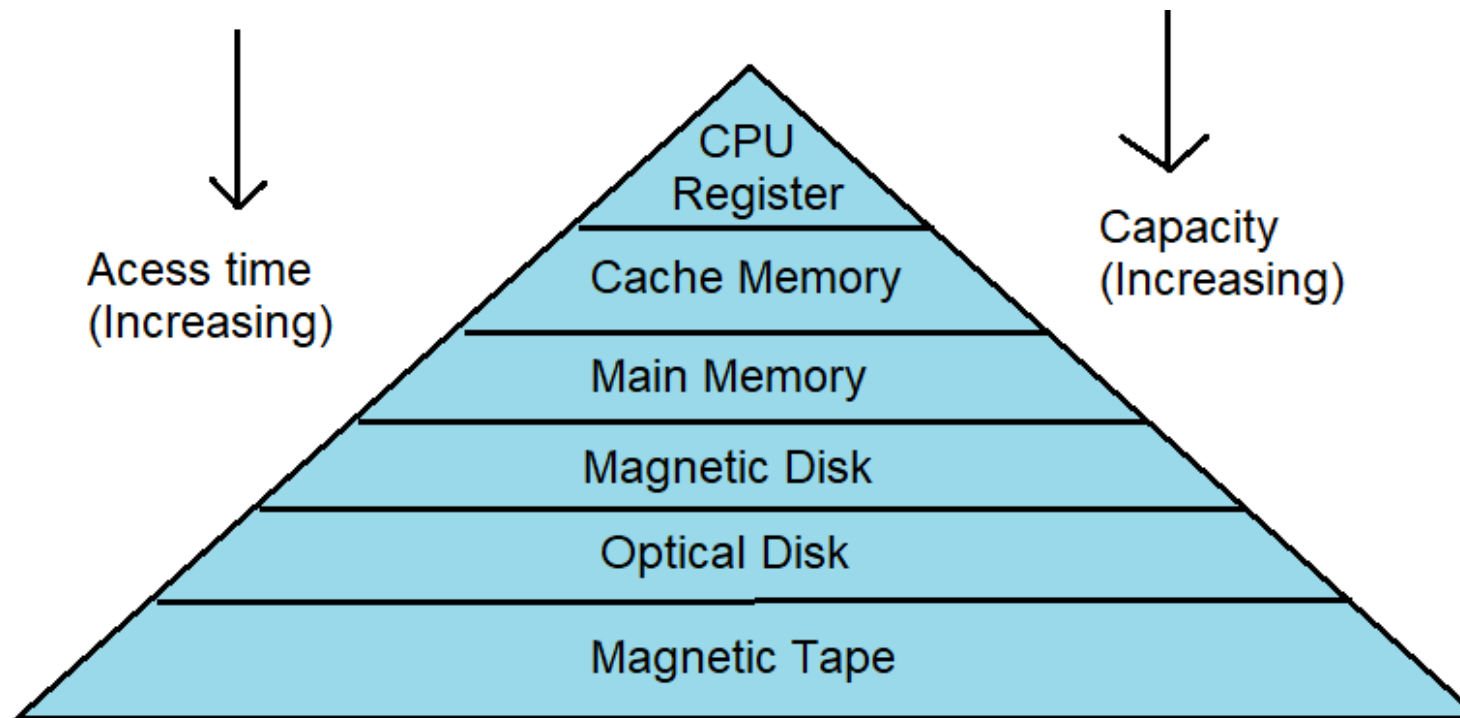
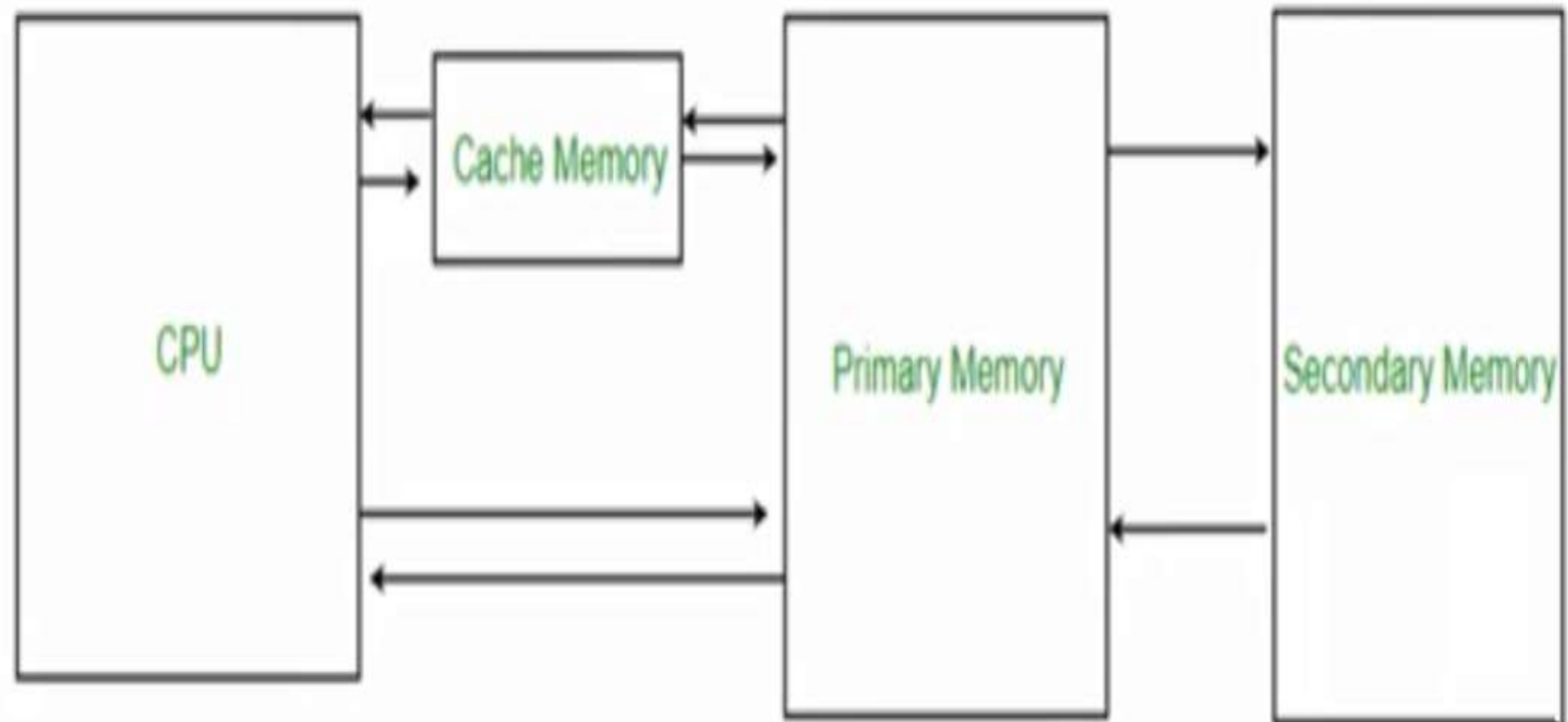


Fig:- Memory Hierarchy

Memory Hierarchy

- At the bottom of the hierarchy are the relatively slow magnetic tapes used to store removable files.
- Next are the magnetic disks used as backup storage.
- The main memory occupies a central position by being able to communicate directly with the CPU and with auxiliary memory devices through an I/O processor.
- When programs not residing in main memory are needed by the CPU, they are brought in from auxiliary memory.



Cache Memories

The Memory
System

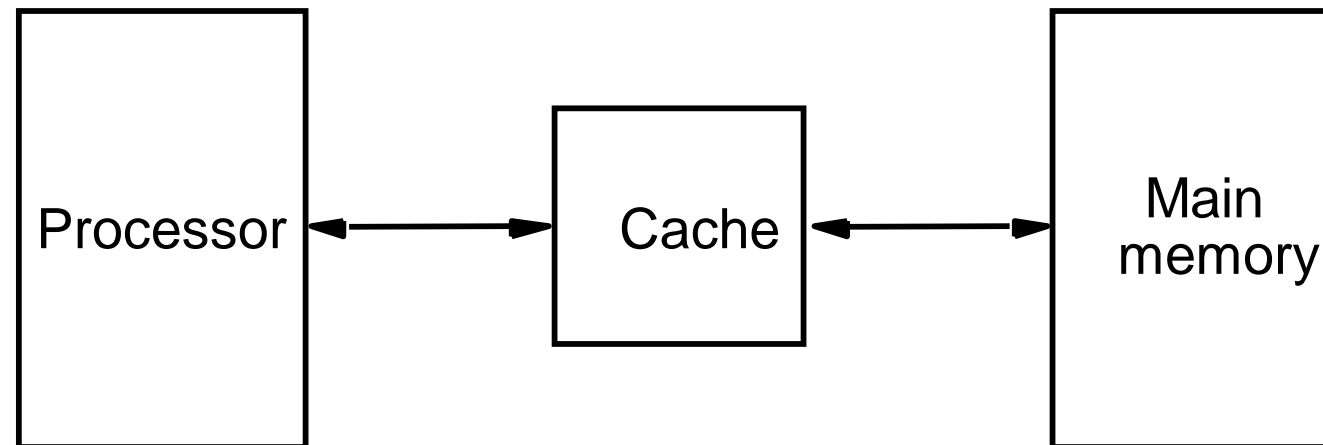
Cache Memories

- Processor is much faster than the main memory.
 - As a result, the processor has to spend much of its time waiting while instructions and data are being fetched from the main memory.
 - Major obstacle towards achieving good performance.
- Speed of the main memory cannot be increased beyond a certain point.
- Cache memory is an architectural arrangement which makes the main memory appear faster to the processor than it really is.
- Cache memory is based on the property of computer programs known as “locality of reference”.

Locality of Reference

- Analysis of programs indicates that many instructions in localized areas of a program are executed repeatedly during some period of time, while the others are accessed relatively less frequently.
 - These instructions may be the ones in a loop, nested loop or few procedures calling each other repeatedly.
 - This is called “locality of reference”.
- **Temporal locality of reference:**
 - Recently executed instruction is likely to be executed again very soon.
- **Spatial locality of reference:**
 - Instructions with addresses close to a recently instruction are likely to be executed soon.

Cache memories



- Processor issues a Read request, a block of words is transferred from the main memory to the cache, one word at a time.
- Subsequent references to the data in this block of words are found in the cache.
- At any given time, only some blocks in the main memory are held in the cache. Which blocks in the main memory are in the cache is determined by a “mapping function”.

- ? When the cache is full, and a block of words needs to be transferred from the main memory, some block of words in the cache must be replaced. This is determined by a “replacement algorithm”.

Cache hit

- Existence of a cache is transparent to the processor. The processor issues Read and Write requests in the same manner.
- If the data is in the cache it is called a Read or Write hit.
- Read hit:
 - The data is obtained from the cache.

- Write hit:

- Cache has a replica of the contents of the main memory.
- Contents of the cache and the main memory may be updated simultaneously. This is the write-through protocol.
- Update the contents of the cache, and mark it as updated by setting a bit known as the dirty bit or modified bit. The contents of the main memory are updated when this block is replaced.
- This is write-back or copy-back protocol.

Cache miss

- If the data is not present in the cache, then a Read miss or Write miss occurs.
- Read miss:
 - Block of words containing this requested word is transferred from the memory.
 - After the block is transferred, the desired word is forwarded to the processor.
 - The desired word may also be forwarded to the processor as soon as it is transferred without waiting for the entire block to be transferred. This is called load-through or early-restart.

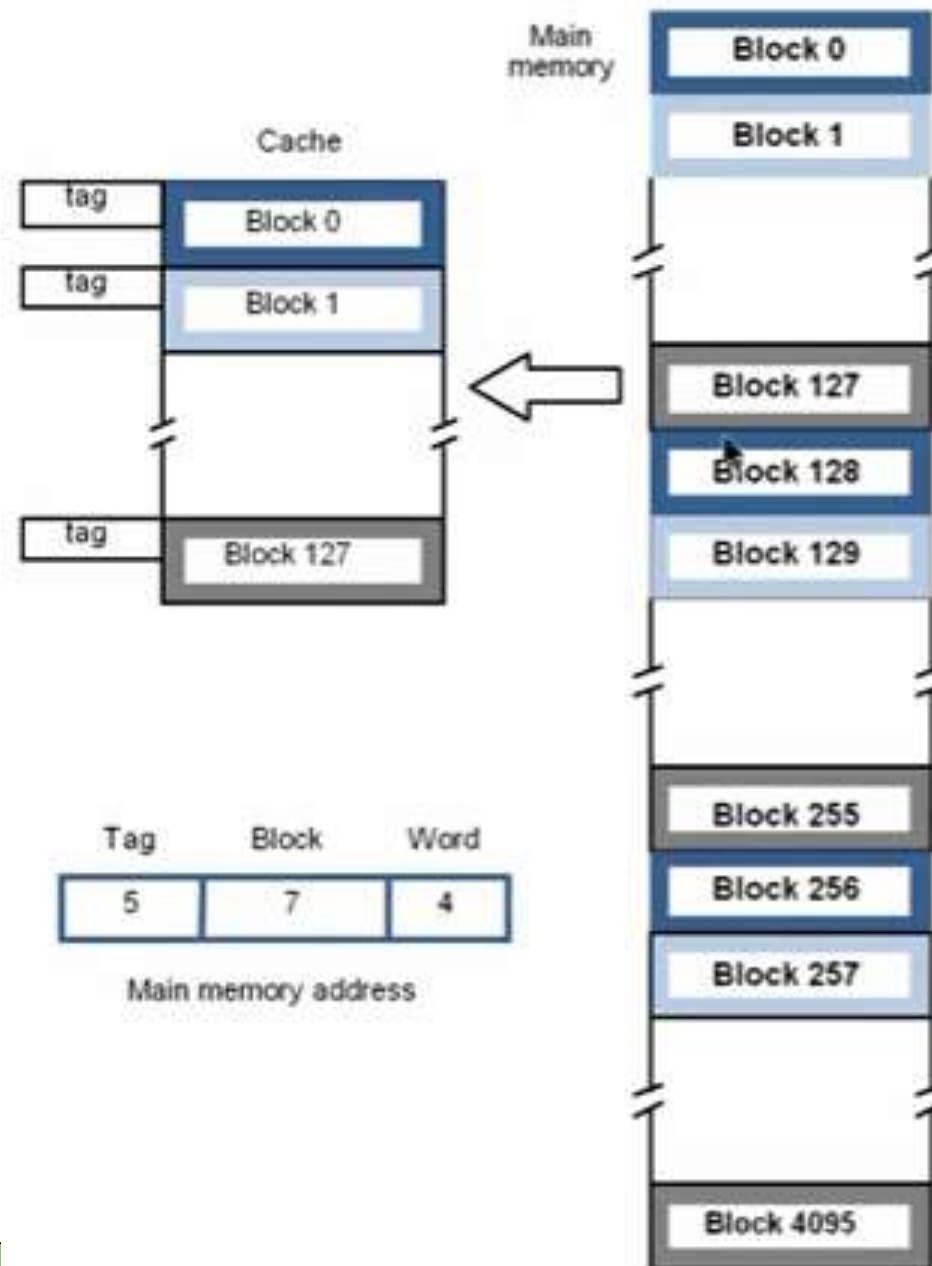
- Write-miss:
 - Write-through protocol is used, then the contents of the main memory are updated directly.
 - If write-back protocol is used, the block containing the addressed word is first brought into the cache. The desired word is overwritten with new information.

Mapping functions

- Mapping functions determine how memory blocks are placed in the cache.
- A simple processor example:
 - Cache consisting of 128 blocks of 16 words each.
 - Total size of cache is 2048 (2K) words.
 - Main memory is addressable by a 16-bit address.
 - Main memory has 64K words.
 - Main memory consisting of 4096 blocks

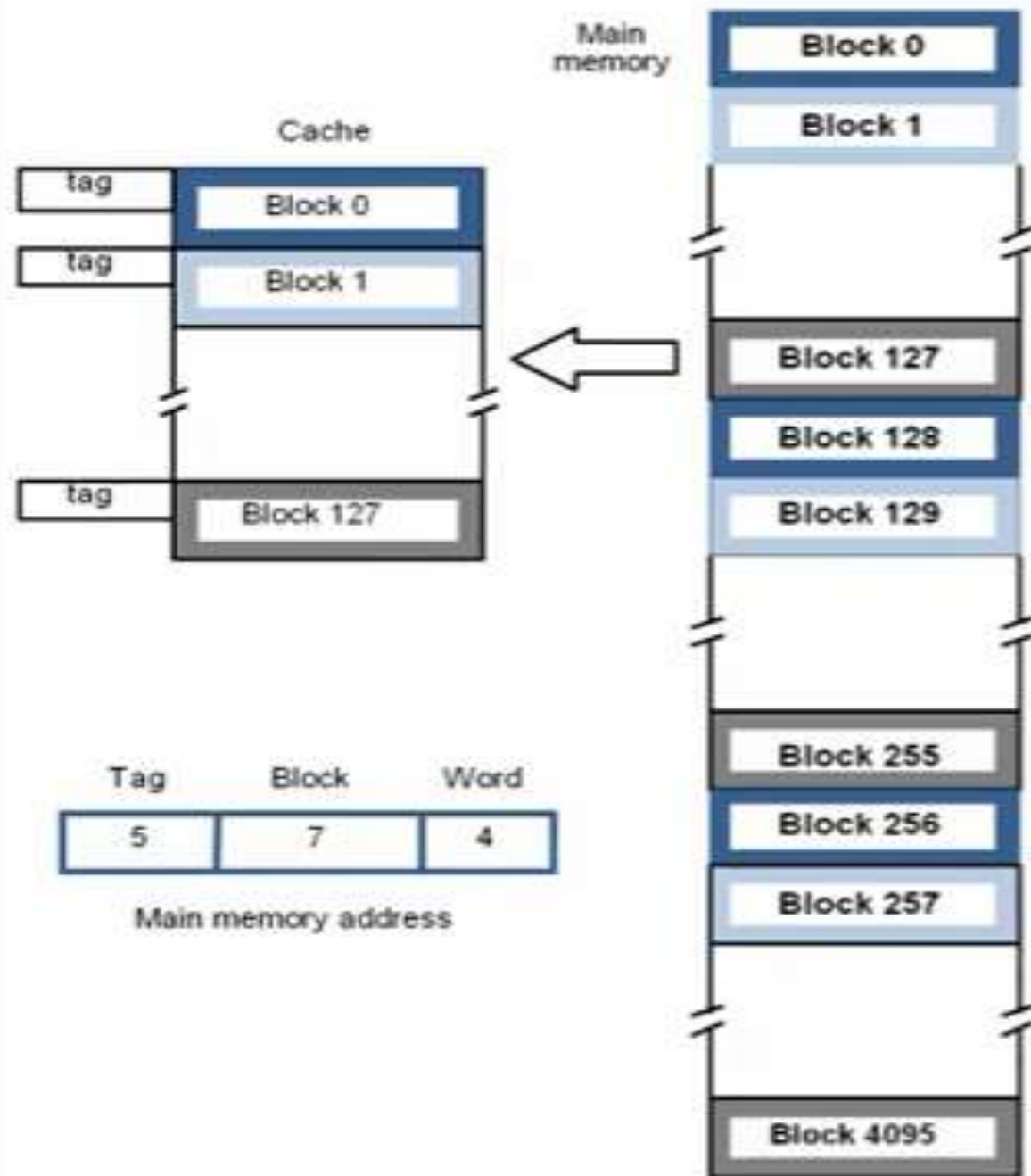
- Three mapping functions:
 - Direct mapping
 - Associative mapping
 - Set-associative mapping.

Direct Mapping



- Block j of the main memory maps to $j \bmod 128$ of the cache.
- Position of a block in cache = $\text{Block number} \% \text{Total no of cache block}$
- Thus one of the main memory blocks **0, 128, 256, ...** is loaded in the cache **block 0**.
- Blocks **1, 129, 257,** are stored in cache **block 1**

Direct Mapping



- **More than one** memory block is mapped onto the same position in the cache.
- May lead to **contention** for cache blocks even if the cache is not full.
- Resolve the contention by allowing new block to **replace the old block**, by a replacement algorithm.



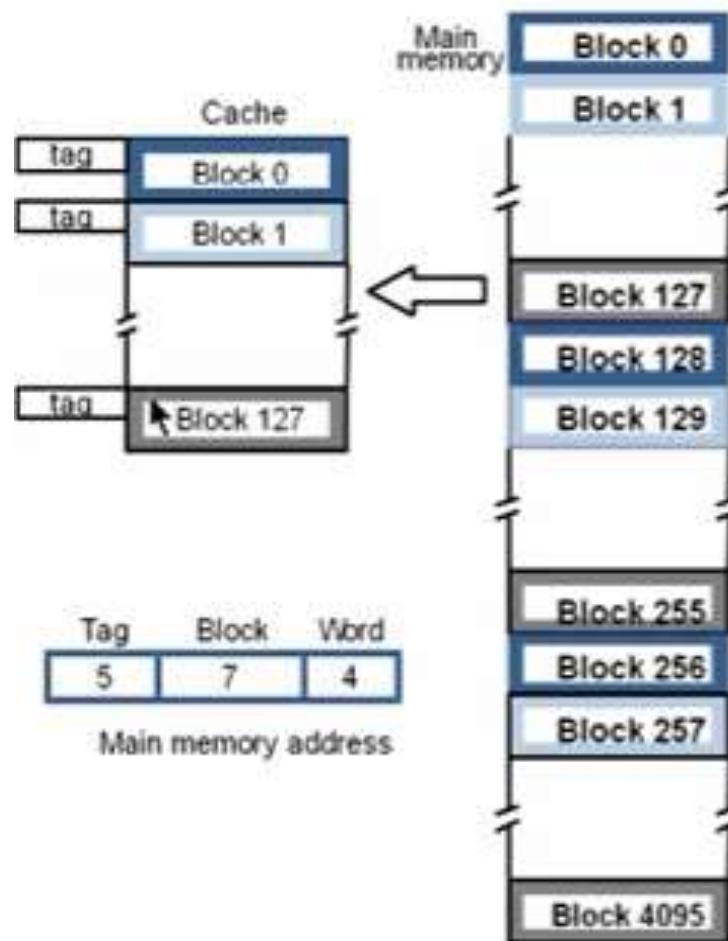
The diagram shows a horizontal rectangle divided into three equal-width sections. Above each section is a label: 'Tag' for the first, 'Block' for the second, and 'Word' for the third. Inside each section is a number: '5' in the first, '7' in the second, and '4' in the third. The entire structure is enclosed in a thin blue border.

Tag	Block	Word
5	7	4

Main memory address

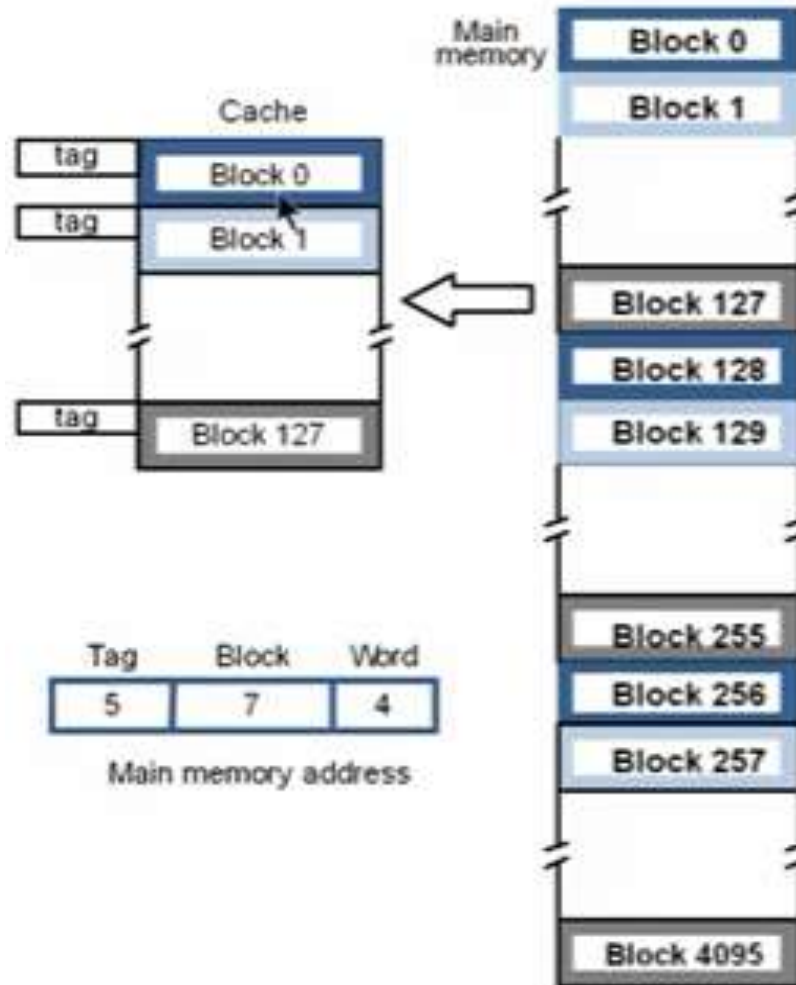
- ? The main memory address is divided into three parts:
- **Tag:** Identifies which block of memory is currently in the cache line.
- **Index (or Block):** Specifies the cache line where the block is stored.
- **Offset (Word):** Specifies the exact word within a block.

Direct Mapping



- Since there are 65536 words
- No of bits in address= $65536 = 2^{16} = 16 \text{ bits}$
- There are 16 words in a **block**, to find a word in it= 16 words= 2^4 Words = **4 Bits**
- There are 127 **Cache blocks**, to address it = 127 blocks = 2^7 Blocks= **7 Bits**
- Remaining bits= $16 - (4+7) = 5 \text{ Bits For Tag.}$
- Tag bit represent which among the 32 blocks reside in cache ($4096 / 128 = 32$, ie, 0,128,256...)

Direct Mapping



- The **high-order 5 bits** of the address are compared with the **tag bits** associated with that cache location.
- If **match occurs** then the desired word is **in that block** of the cache.
- If There is a **no match**, then the **block** containing the required word must first be **read from the main memory** and loaded into the cache.

Merit:

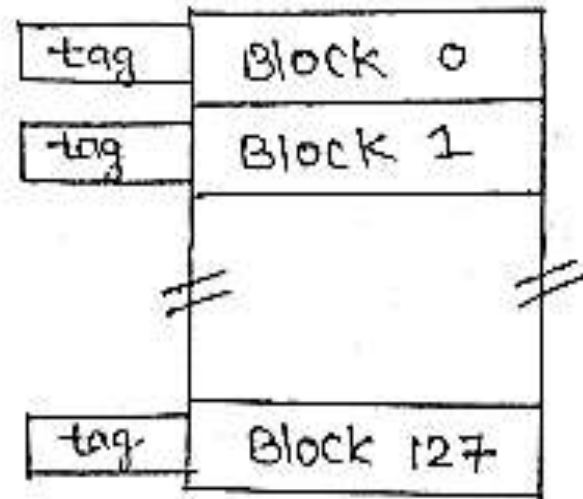
- It is **easy to implement**.

Demerit:

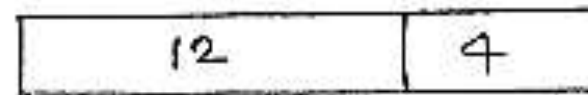
- It is **not very flexible**.

(2) Associative Mapping:-

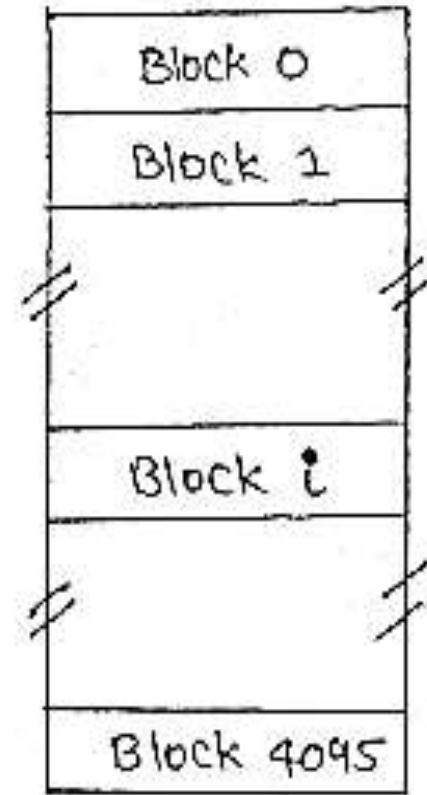
- ? 1) This is more flexible mapping method, in which main memory block can be placed into any cache block position.
- ? 2) In this, 12 tag bits are required to identify a memory block when it is resident in the cache.
- ? 3) The tag bits of an address received from the processor are compared to the tag bits of each block of the cache to see, if the desired block is present. This is known as Associative Mapping technique.
- ? 4) Cost of an associated mapped cache is higher than the cost of direct-mapped because of the need to search all 128 tag patterns to determine whether a block is in cache. This is known as associative search.



Cache




Tag Word
Main Memory Address

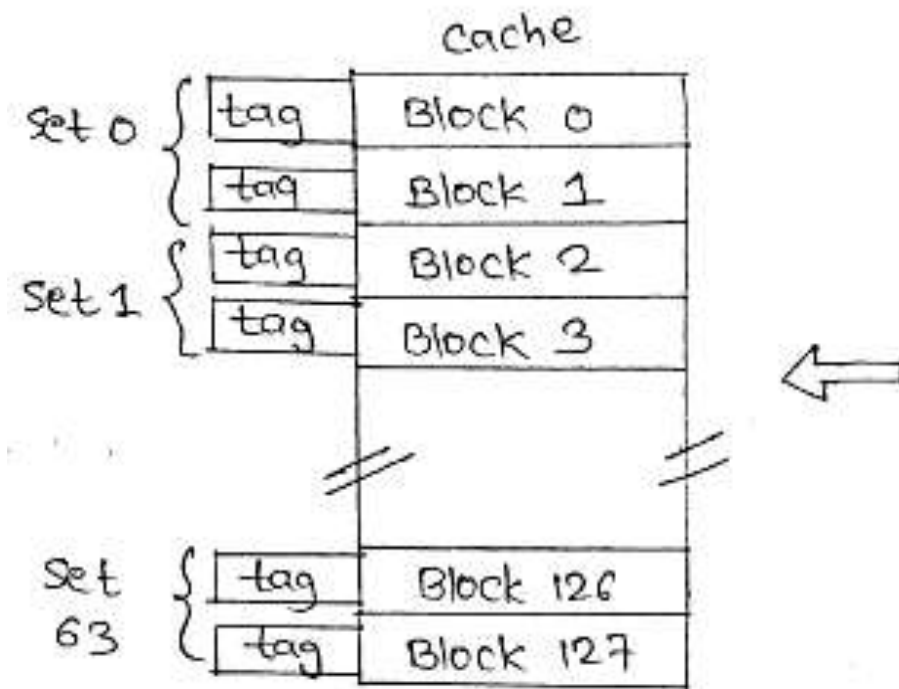


Main Memory

(3) Set-Associated Mapping:-

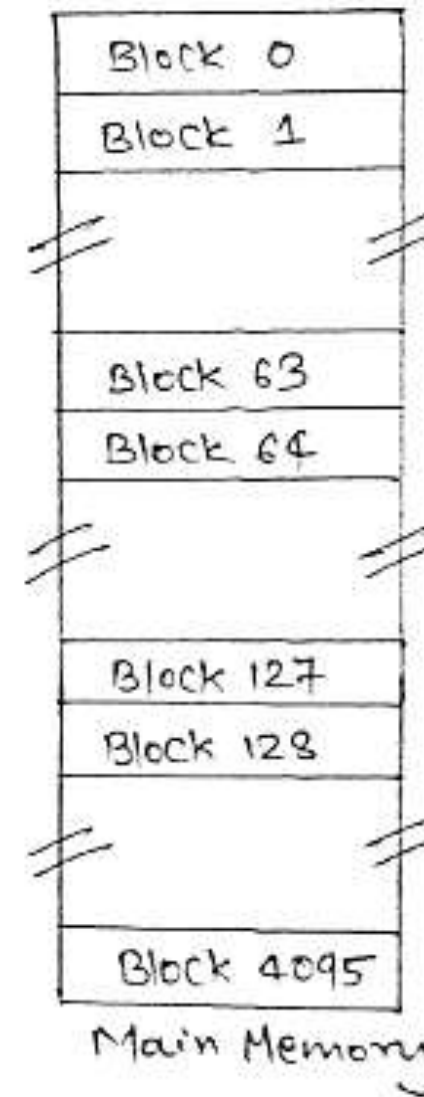
- ? 1) It is the combination of direct and associative mapping technique.
- ? 2) Cache blocks are grouped into sets and mapping allow block of main memory reside into any block of a specific set. Hence contention problem of direct mapping is eased , at the same time , hardware cost is reduced by decreasing the size of associative search.

- 
- ? 3) For a cache with two blocks per set. In this case, memory block 0, 64, 128,.....,4032 map into cache set 0 and they can occupy any two block within this set.
 - ? 4) 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 bits of address must be associatively compared to the tags of the two blocks of the set to check if desired block is present. This is two way associative search.

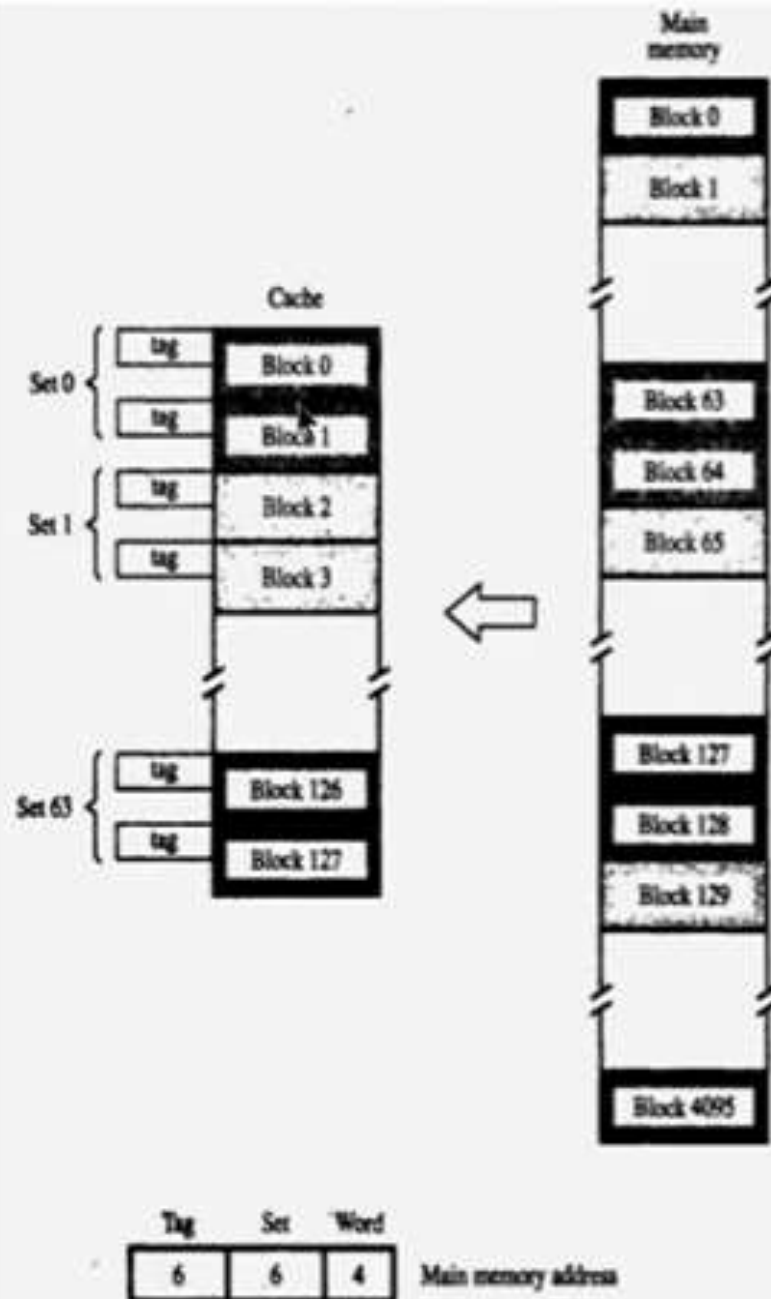


Tag	Set	Word
6	6	4

Main Memory Address

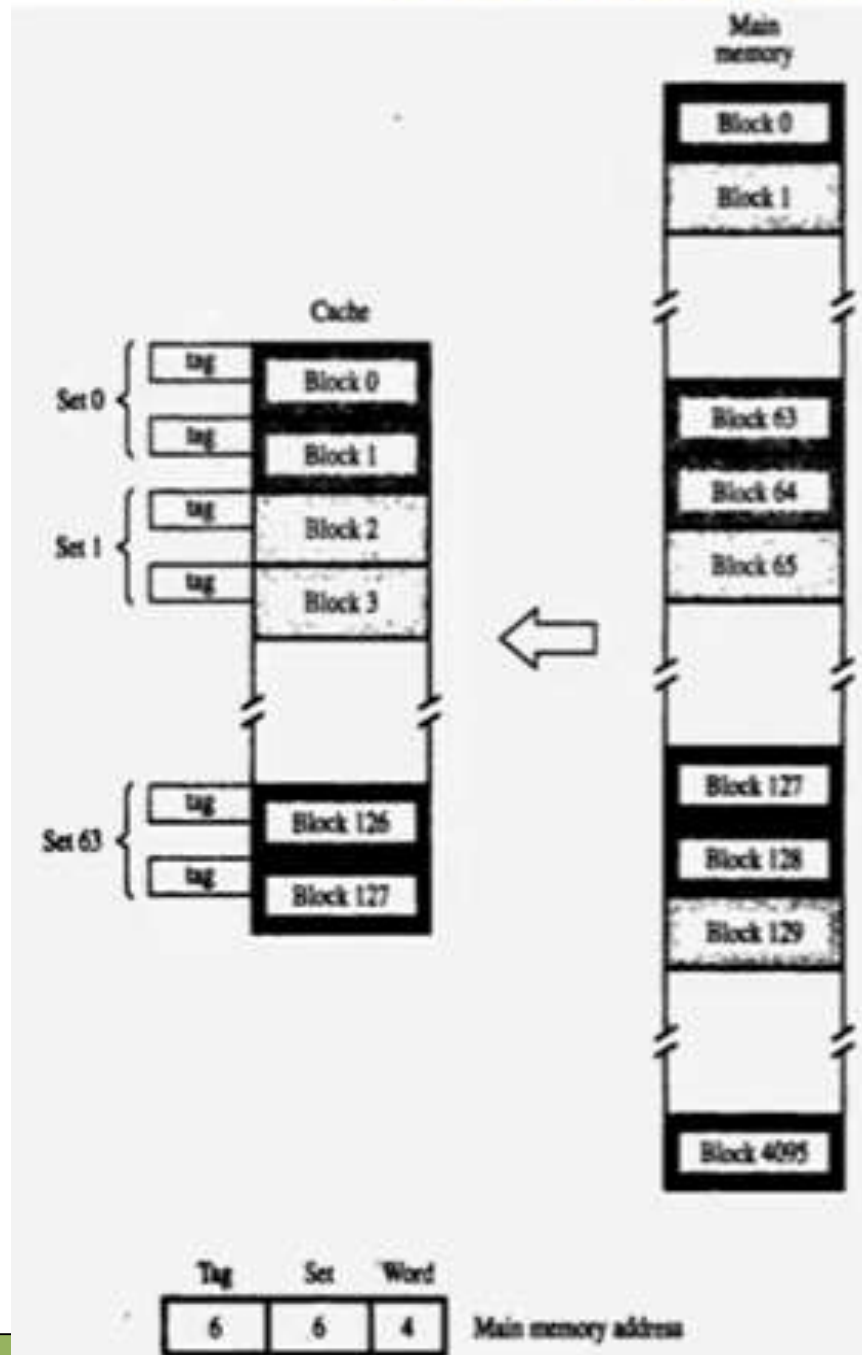


Set Associative Mapping



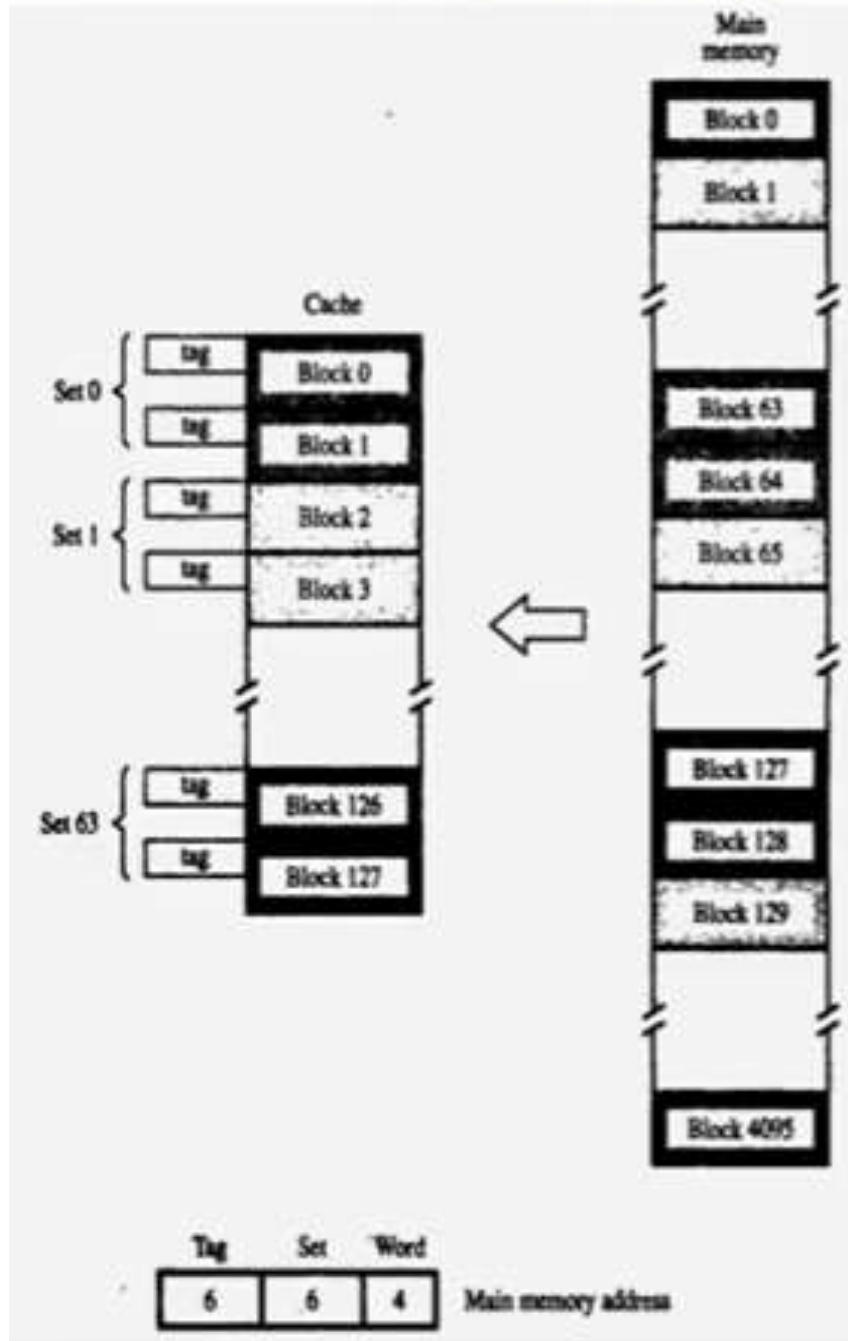
- It is the combination of direct and associative mapping.
- The blocks of the cache are **grouped into sets** and the mapping allows a block of the main memory to **reside in any block of the specified set**.
- In this case, the cache has two blocks per set
 - The memory blocks **0,128.....** maps into cache **set 0** and they can **occupy either of the two block position** within the set.

Set Associative Mapping



- **4 Bits** to address 16 **words** of a block, $16 = 2^4 = 4 \text{ Bits}$ for **word** address
- Here a set has 2 block (2 Way set associative):
- $128/2 = 64 = 2^6 = 6 \text{ Bits}$ for **Set** address
- Remaining = $16 - (4 + 6) = 6 \text{ Bits}$ for **Tag**.
- Number of blocks per set is a design parameter.
 - One extreme is to have **all the blocks in one set** requiring no set bits (**fully associative** mapping).
 - Other extreme is to have **one block per set**, is the same as **direct mapping**.

Set Associative Mapping



- The **contention problem** of the direct method is eased by having a **few choices for block placement**.
- The **hardware cost is reduced by decreasing the size of the associative search**.
- For the main memory and cache sizes in Figure
- four blocks (**4 way set associative**) per set:
- Set field: $128/4 = 32 = 2^5 = 5$ bits set field
- Eight blocks (**8 way set associative**) per set:
- Set field: $128/8 = 16 = 2^4 = 4$ bits set field
- A cache that has k blocks per set is referred to as a k-way set-associative cache

Cache Coherence Problem

- A bit called as “**valid bit**” is provided for each block.
- If the block contains valid data, then the bit is set to “1”, else it is “0”.
- Valid bits are set to 0, when the power is just turned on.
- When a block is loaded into the cache for the first time, the valid bit is set to 1.
- Data transfers between main memory and disk occur directly bypassing the cache.
- Whenever a main memory block is updated by a source that bypasses the cache, a check is made to determine whether the block being loaded is currently in the cache.
- If it is, its valid bit is cleared to 0. This ensures that stale data will not exist in the cache.

- *What happens if the data in the disk and main memory changes and the write-back protocol is being used?*
- In this case, the data in the cache may also have changed and is indicated by the dirty bit.
- The copies of the data in the cache, and the main memory are different. This is called the **cache coherence problem**.
- *One option is to force a write-back before the main memory is updated from the disk.*

Replacement Algorithm:

- ? In direct mapping, the position of each block is pre-determined and there is no need of replacement strategy.
- ? In associative & set associative method, the block position is not pre-determined ;i.e...when the cache is full and if new blocks are brought into the cache, then the cache controller must decide which of the old blocks has to be replaced.

LRU algorithm

- ? when a block is to be over-written, it is sensible to over-write the one that has gone the longest time without being referenced. This block is called **Least recently Used(LRU) block** & the technique is called **LRU algorithm**.
- ? The cache controller track the references to all blocks with the help of block counter.

Eg:

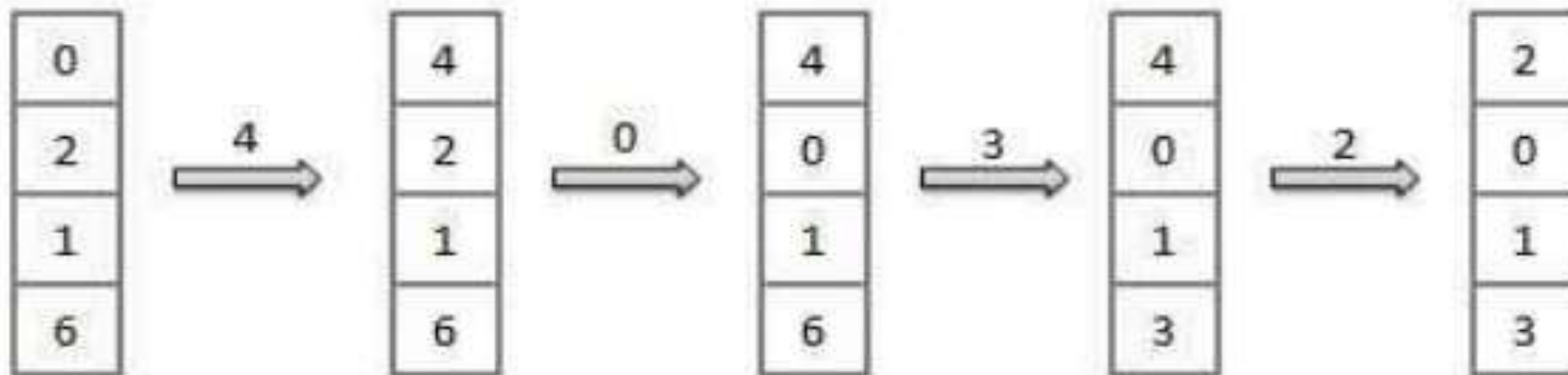
- ? Consider 4 blocks/set in set associative cache,
- ? 2 bit counter can be used for each block.
- ? When a '**hit**' occurs, then block counter=0;The counter with values originally lower than the referenced one are incremented by 1 & all others remain unchanged.
- ? When a '**miss**' occurs & if the set is full, the blocks with the counter value 3 is removed, the new block is put in its place & its counter is set to "0" and other block counters are incremented by "1".

Merit:

- ? The performance of LRU algorithm is improved by randomness in deciding which block is to be over-written.

Reference String : 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1

Misses : x x x x x x x x



$$\text{Fault Rate} = 8 / 12 = 0.67$$

? Optimal Algorithm

? The optimal page replacement method selects that page for a replacement for which the time to the next reference is the longest.

? **Features:**

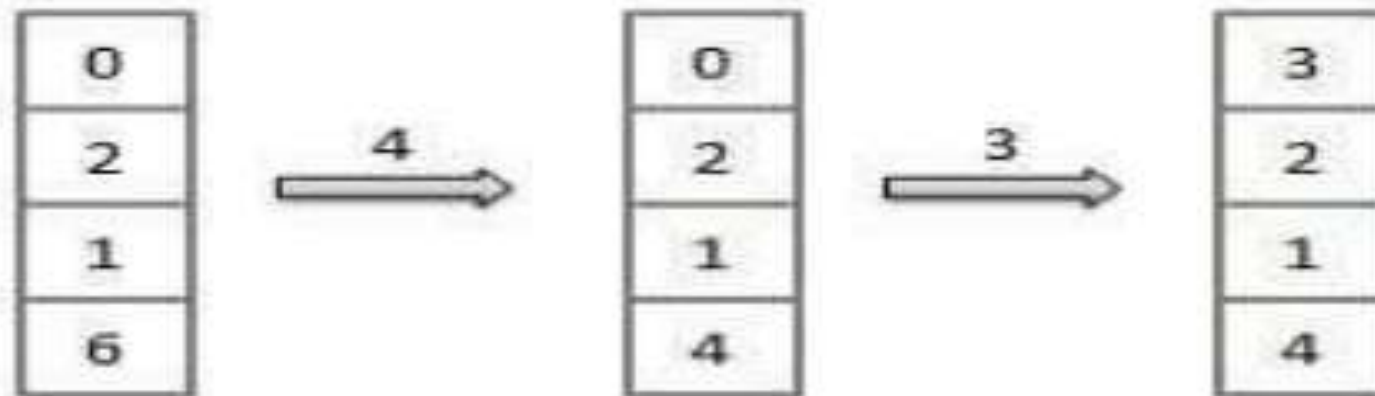
? Optimal algorithm results in the fewest number of page faults. This algorithm is difficult to implement.

? An optimal page-replacement algorithm method has the lowest page-fault rate of all algorithms.

? Replace the page which unlikely to use for a longer period of time. It only uses the time when a page needs to be used.

Reference String : 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1

Misses : x x x x x x



$$\text{Fault Rate} = 6 / 12 = 0.50$$



Memory Management

What is Virtual Memory

- ? 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?

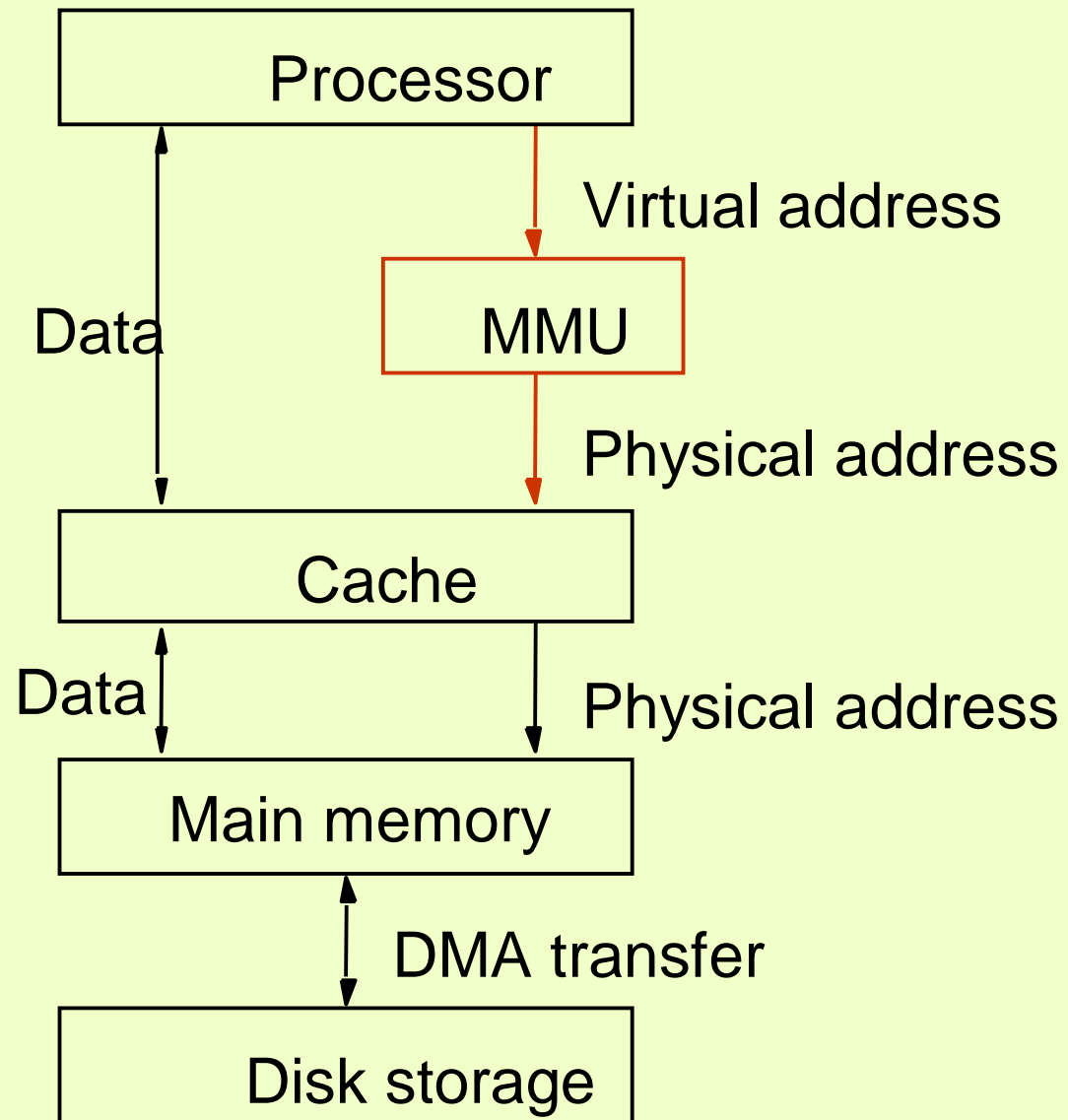
- ? In this scheme, whenever some pages need to be loaded in the main memory for the execution and the memory is not available for those many pages, then in that case, instead of stopping the pages from entering in the main memory, the OS searches for the RAM area that are least used in the recent times 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.

Logical vs. Physical Address Space

- ? The concept of a logical *address space* that is bound to a separate *physical address space* is central to proper memory management
- ? **Logical address** – generated by the CPU; also referred to as *virtual address*
- ? **Physical address** – address seen by the memory unit

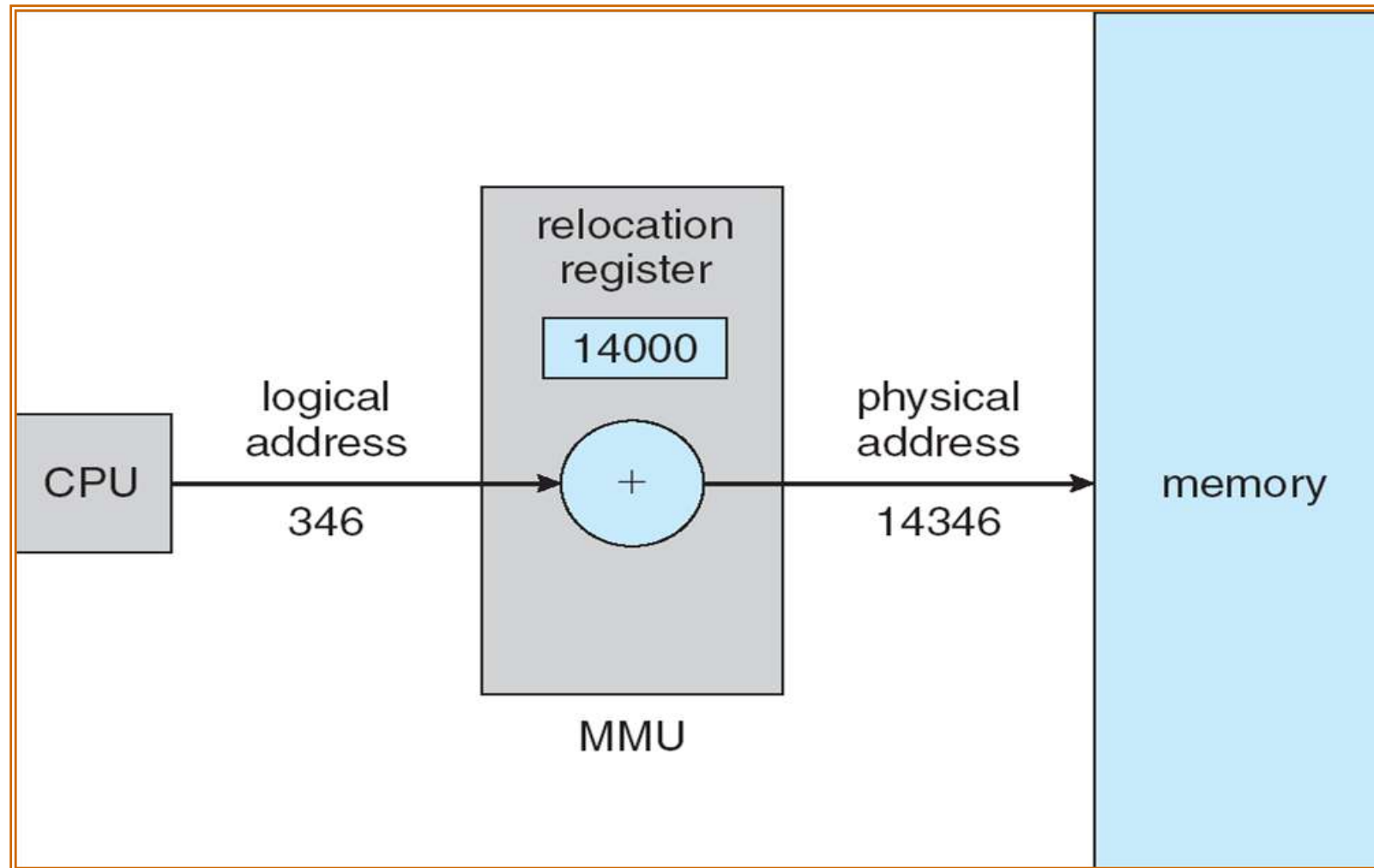
Memory-Management Unit (MMU)

- ? Hardware device that maps virtual to physical address
- ? In MMU scheme, the value in the relocation register is added to every address generated by a user process at the time it is sent to memory
- ? ?
- ? The user program deals with *logical* addresses; it never sees the *real* physical addresses



? Fig: Virtual memory organization

Dynamic relocation using a relocation register



Paging

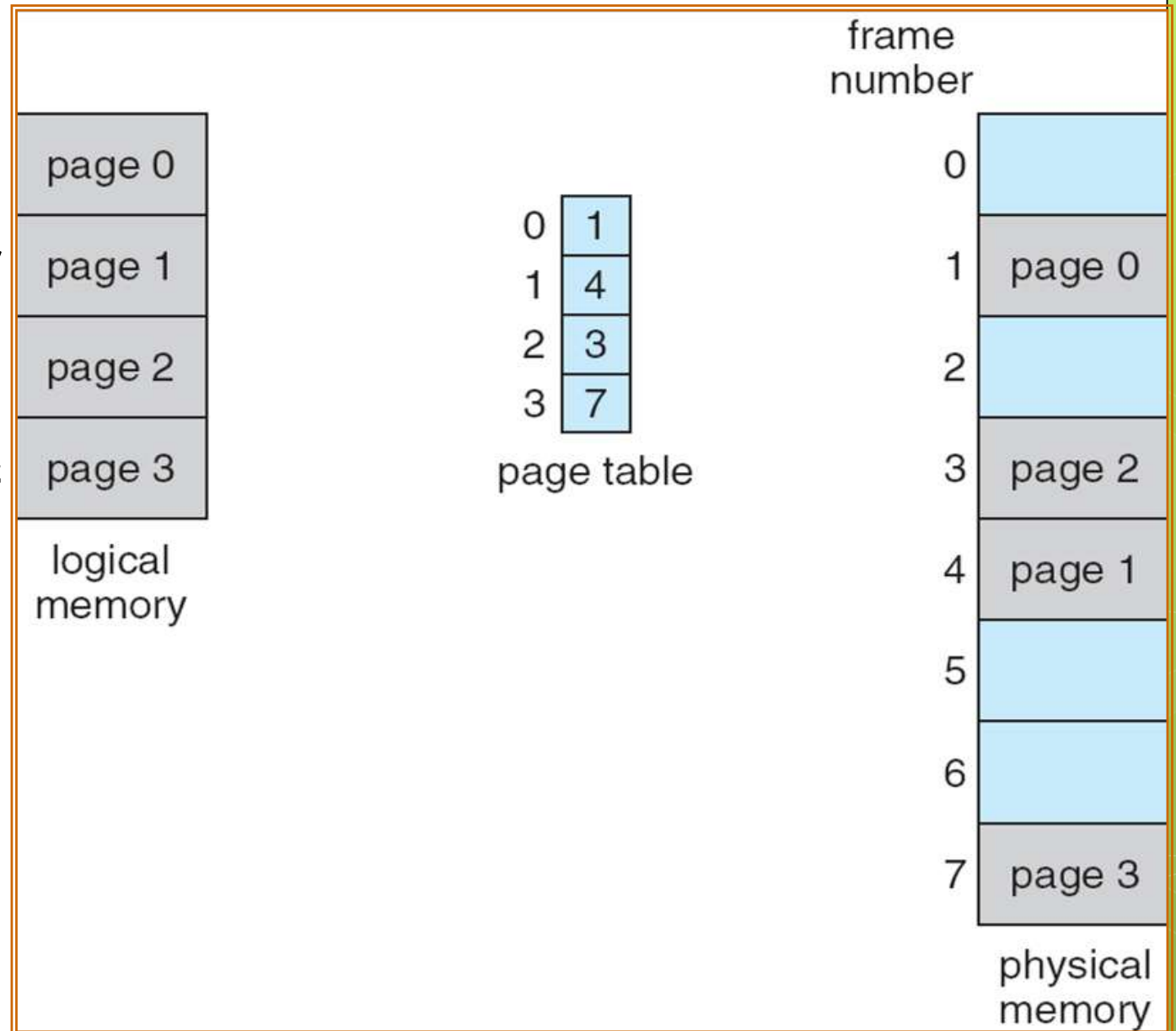
- Paging is a memory management scheme that eliminates the need for a contiguous allocation of physical memory.
- Paging divides virtual memory into small fixed-size blocks called **pages**. Each page is assigned a **virtual page number**
- physical memory is divided into equally sized blocks called **frames**.
- Any page (from any process) can be placed into any available frame

- When the computer runs out of RAM, pages that aren't currently in use are moved to the hard drive, into an area called a swap file. The swap file acts as an extension of RAM.
- When a page is needed again, it is swapped back into RAM, a process known as **page swapping**. This ensures that the operating system (OS) and applications have enough memory to run.

- The operating system maintains a **Page Table** for each process.
- Information about the main memory location of each page is kept in the **page table**.
 - Main memory address where the page is stored.
 - Current status of the page.
- Area of the main memory that can hold a page is called as **page frame**.

Paging Example

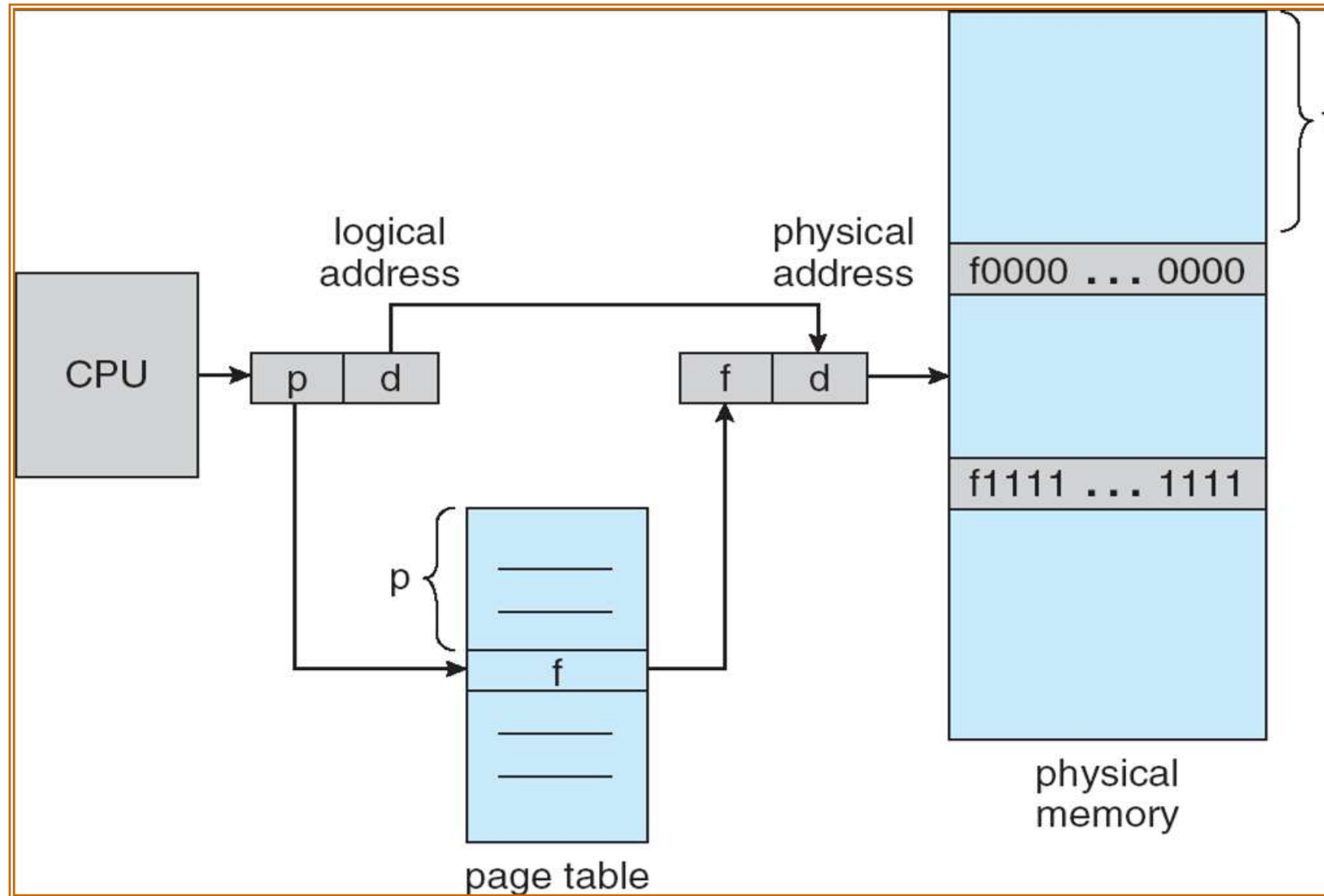
- In the following example, for instance, page 2 of the program's logical memory is currently stored in frame 3 of physical memory:





Address Translation Scheme

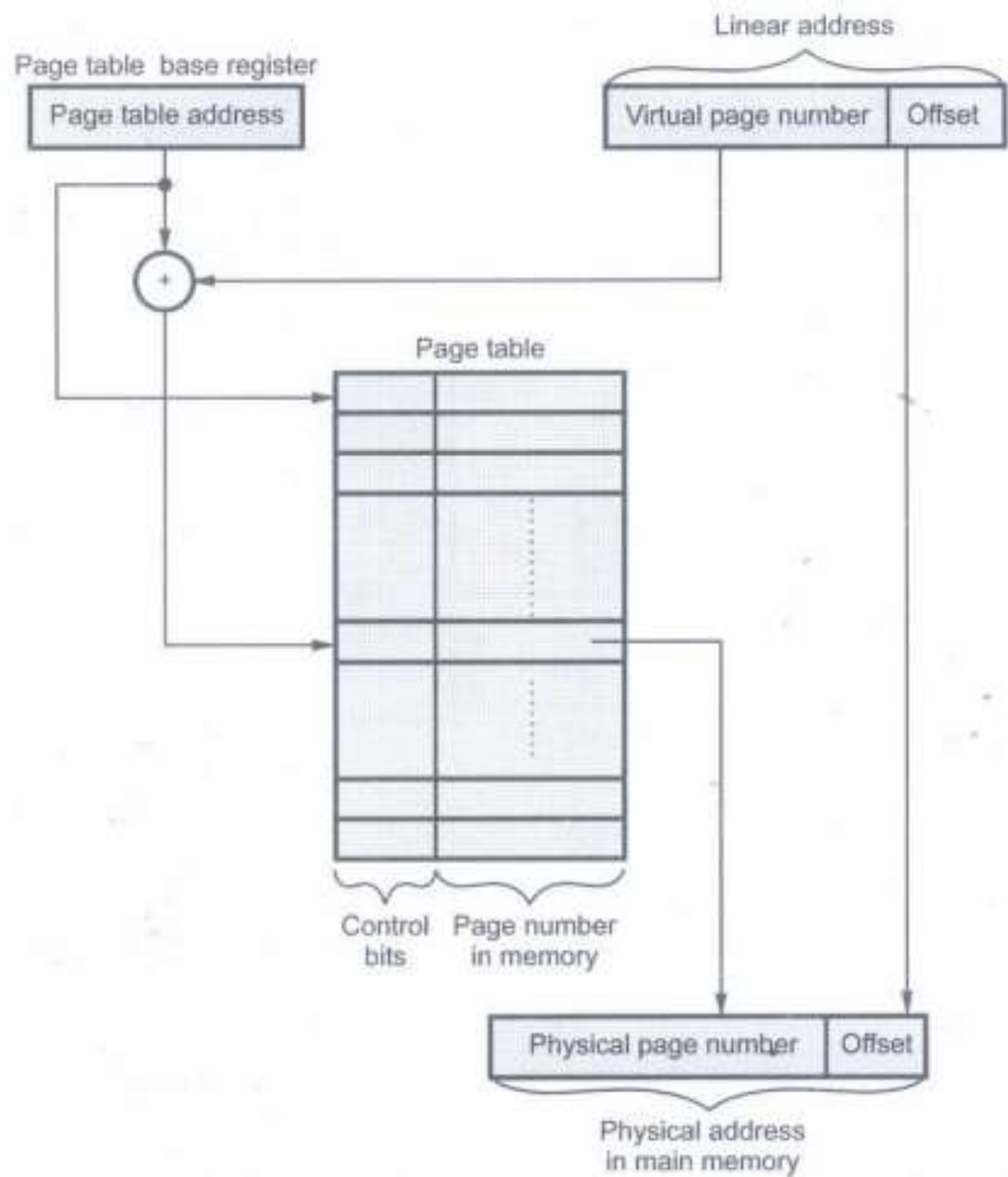
- ? Address generated by CPU is divided into:
 - ? **Page number (p)** – used as an index into a *page table* which contains base address of each page in physical memory.
 - ? **Page offset (d)** – combined with base address to define the physical memory address that is sent to the memory unit.

Address Translation Architecture

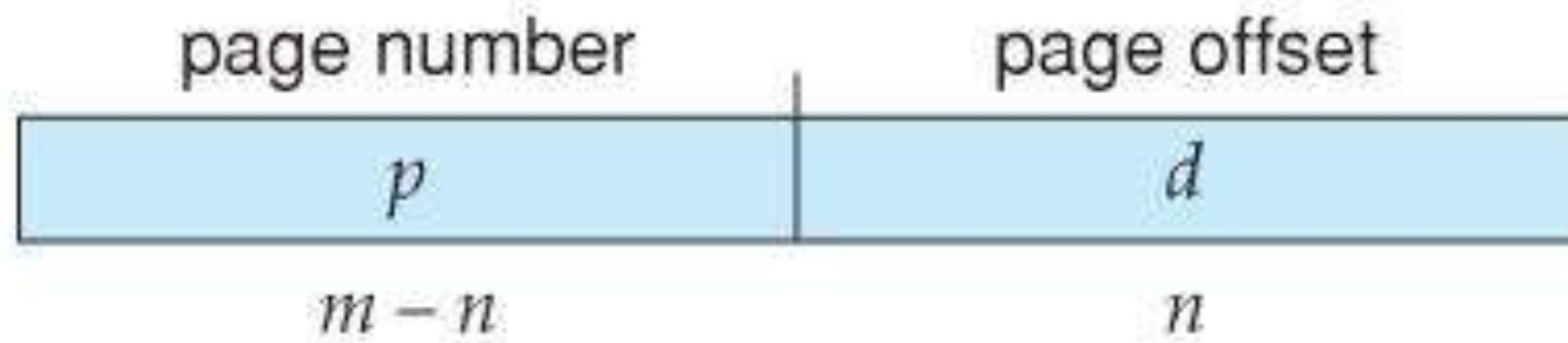


- 
- ? A logical address consists of two parts:
 - ? A page number in which the address resides, and an offset from the beginning of that page. (The number of bits in the page number limits how many pages a single process can address.
 - ? The number of bits in the offset determines the maximum size of each page, and should correspond to the system frame size.)

- 
- ? To obtain the address of the corresponding entry in the page table the virtual page number is added with the contents of page table base register, in which the starting address of the page table is stored.
 - ? The entry in the page table gives the physical page number, in which offset is added to get the physical address of the main memory.



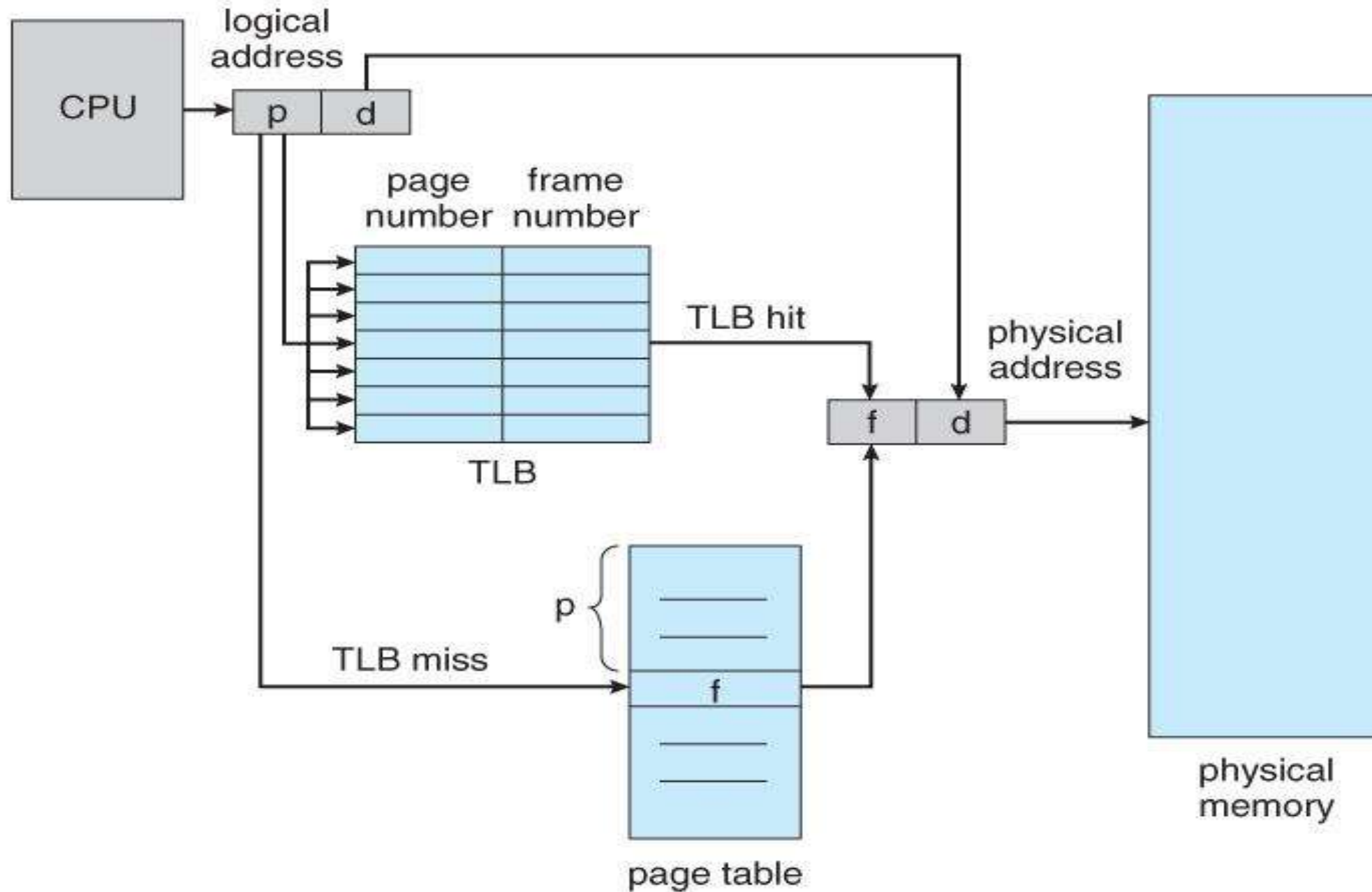
- ? Page numbers, frame numbers, and frame sizes are determined by the architecture, but are typically powers of two, allowing addresses to be split at a certain number of bits.
- ? For example, if the **logical address size** is 2^m and the **page size** is 2^n , then the high-order **m-n bits** of a logical address designate the **page number** and the **remaining n bits** represent the **offset**.



Translation Lookaside Buffer (TLB)

- ? To support demand paging and virtual memory processor has to access page table which is kept in the main memory
- ? To reduce the access time and degradation of performance, a small portion of the page table is accommodated in the memory management unit. This portion is called **Translation Lookaside Buffer (TLB)**.
- ? It is used to hold the page table entries that corresponds to the most recently accessed pages. When processor finds the page table entries in the TLB it does not have to access page table and saves substantial access time.

Hardware Support




- ? Addresses are first checked against the TLB, and if the info is not there (a TLB miss), then the frame is looked up from main memory and the TLB is updated.
- ? If the TLB is full, then replacement strategies range from **least-recently used, LRU** to random.
- ? The percentage of time that the desired information is found in the TLB is termed the **hit ratio**.

Segmentation

- ? Segmentation is another non-contiguous memory allocation scheme like paging.
- ? Like paging, in segmentation, the process isn't divided indiscriminately into mounted(fixed) size pages.
- ? It is a variable-size partitioning theme.
- ? Like paging, in segmentation, secondary and main memory are not divided into partitions of equal size.

- ? The partitions of secondary memory area units are known as **segments**.
- ? The details concerning every segment are hold in a table known as **segmentation table**.
- ? Segment table contains two main data concerning segment, one is **Base**, which is the bottom address of the segment and another is **Limit**, which is the length of the segment.

- 
- ? The segment base contains starting physical address where resides in memory whereas limit specifies length of the segments.
 - ? The segment number is used as index for segment table.

