MEMORY MANAGEMENT

SYSTEM DESIGN OS PART 3

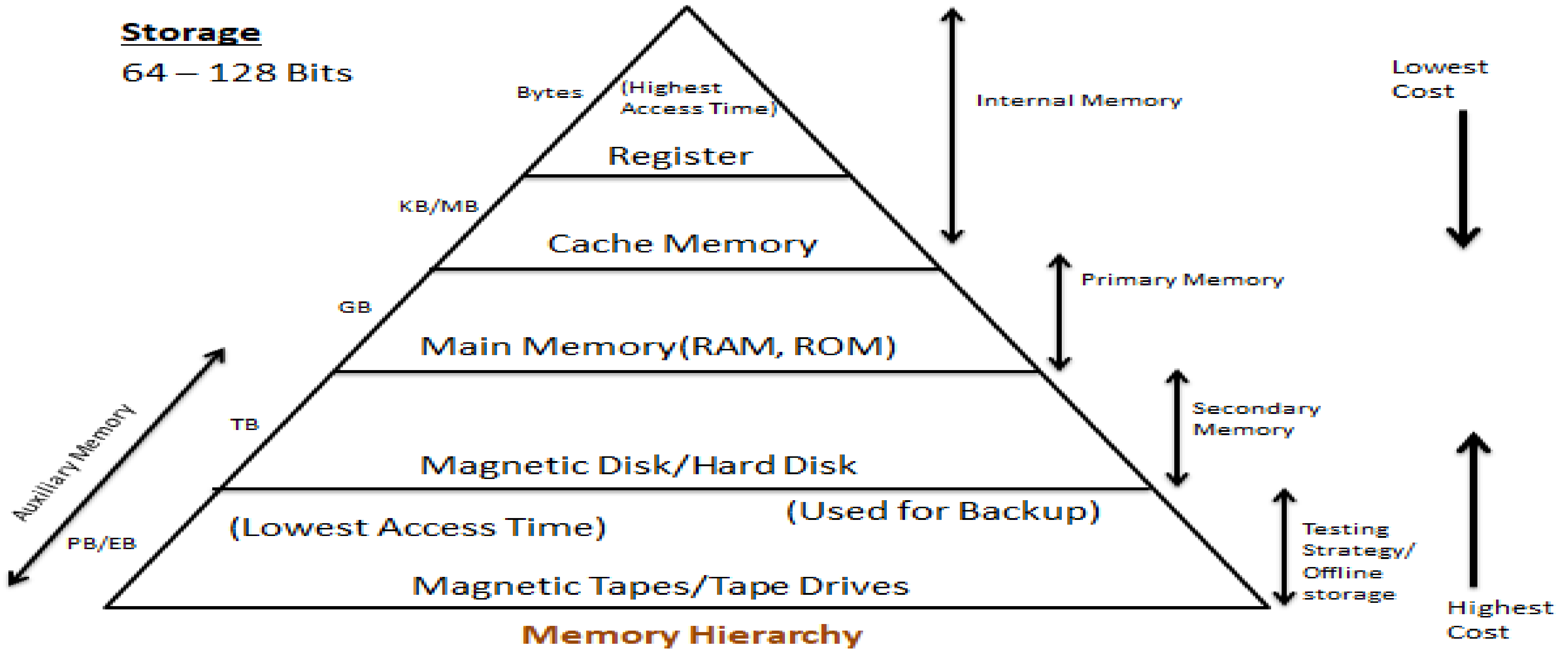
MEMORY MANAGEMENT

Main Memory

Main memory is the place where programs and information are kept when the processor is effectively utilizing them. Main memory is associated with the processor, so moving instructions and information into and out of the processor is extremely fast. Main memory is also known as RAM (Random Access Memory). This memory is volatile. RAM loses its data when a power interruption occurs.

MEMORY MANAGEMENT

Memory Hierarchy



WWW.KEENOTES.COM

MEMORY MANAGEMENT

Memory Management

In a multiprogramming computer, the Operating System resides in a part of memory, and the rest is used by multiple processes. The task of subdividing the memory among different processes is called **Memory Management.**

Need of Memory Management

- ✓Allocate and de-allocate memory before and after process execution.
- ✓To keep track of used memory space by processes.
- ✓To minimize fragmentation issues.
- ✓To proper utilization of main memory.
- ✓To maintain data integrity while executing of process.

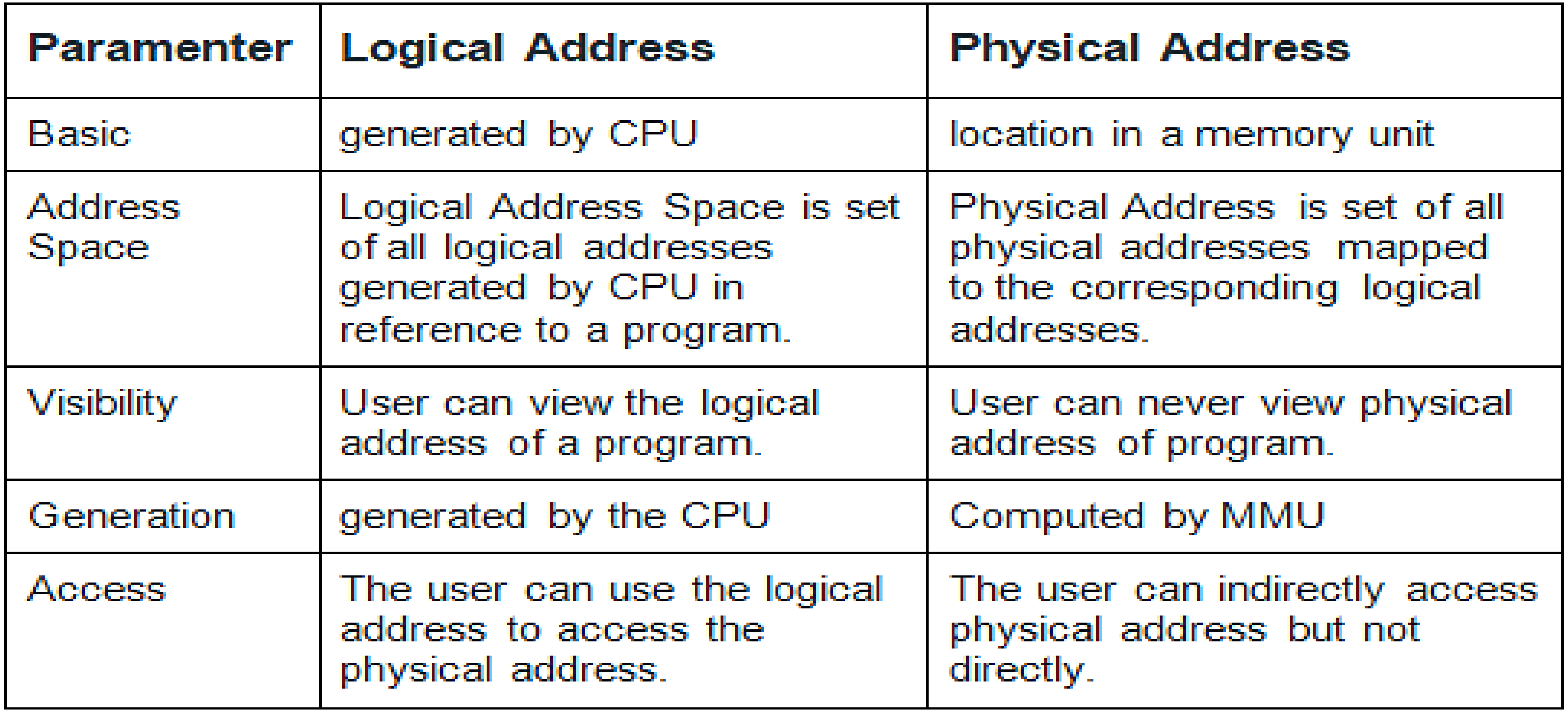
MEMORY MANAGEMENT

Logical and Physical Address Space

- **Logical Address Space**: An address generated by the CPU is known as a “Logical Address”. It is also known as a Virtual address. Logical address space can be defined as the size of the process. A logical address can be changed.

- **Physical Address Space**: An address seen by the memory unit (i.e. the one loaded into the memory address register of the memory) is commonly known as a “Physical Address”. A Physical address is also known as a Real address. The set of all physical addresses corresponding to these logical addresses is known as Physical address space. A physical address is computed by MMU. The run-time mapping from virtual to physical addresses is done by a hardware device Memory Management Unit(MMU). The physical address always remains constant.

MEMORY MANAGEMENT



Paramenter	Logical Address	Physical Address
Basic	generated by CPU	location in a memory unit
Address Space	Logical Address Space is set of all logical addresses generated by CPU in reference to a program.	Physical Address is set of all physical addresses mapped to the corresponding logical addresses.
Visibility	User can view the logical address of a program.	User can never view physical address of program.
Generation	generated by the CPU	Computed by MMU
Access	The user can use the logical address to access the physical address.	The user can indirectly access physical address but not directly.

MEMORY MANAGEMENT

Static and Dynamic Loading

Loading a process into the main memory is done by a loader. There are two different types of loading :

•**Static Loading:** Static Loading is basically loading the entire program into a fixed address. It requires more memory space.

•**Dynamic Loading:** The entire program and all data of a process must be in physical memory for the process to execute. So, the size of a process is limited to the size of physical memory. To gain proper memory utilization, dynamic loading is used. In dynamic loading, a routine is not loaded until it is called. All routines are residing on disk in a relocatable load format. One of the advantages of dynamic loading is that the unused routine is never loaded. This loading is useful when a large amount of code is needed to handle it efficiently.

MEMORY MANAGEMENT

Static and Dynamic Linking

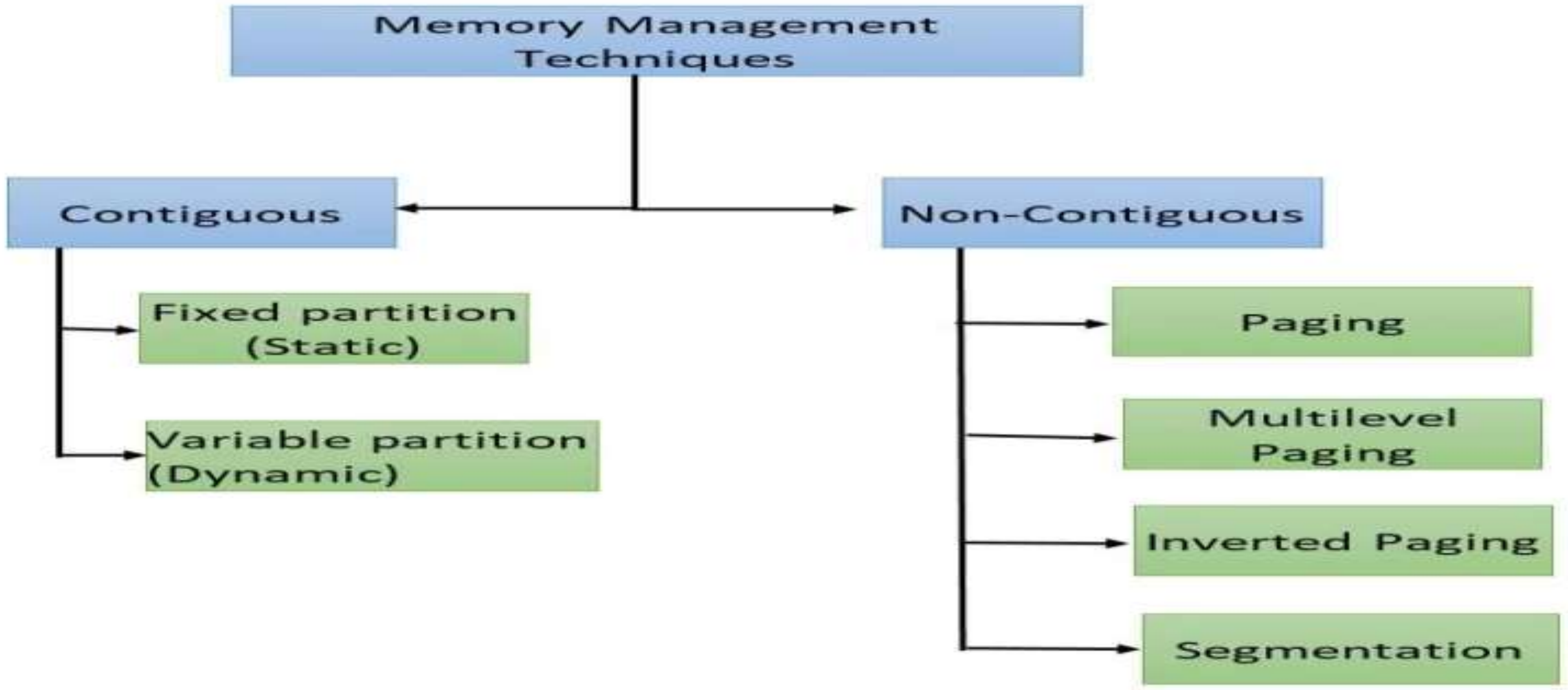
To perform a linking task a linker is used. A linker is a program that takes one or more object files generated by a compiler and combines them into a single executable file.

Static Linking: In static linking, the linker combines all necessary program modules into a single executable program. So there is no runtime dependency. Some operating systems support only static linking, in which system language libraries are treated like any other object module.

Dynamic Linking: The basic concept of dynamic linking is similar to dynamic loading. In dynamic linking, “Stub” is included for each appropriate library routine reference. A stub is a small piece of code. When the stub is executed, it checks whether the needed routine is already in memory or not. If not available then the program loads the routine into memory.

MEMORY MANAGEMENT

Memory Allocation Techniques



MEMORY MANAGEMENT

Contiguous Memory Allocation Technique:

In this technique, contiguous blocks of memory are allocated to each process. So, whenever a process wants to enter the main memory, continuous segment from the totally empty space to the process based on its size are allocated.

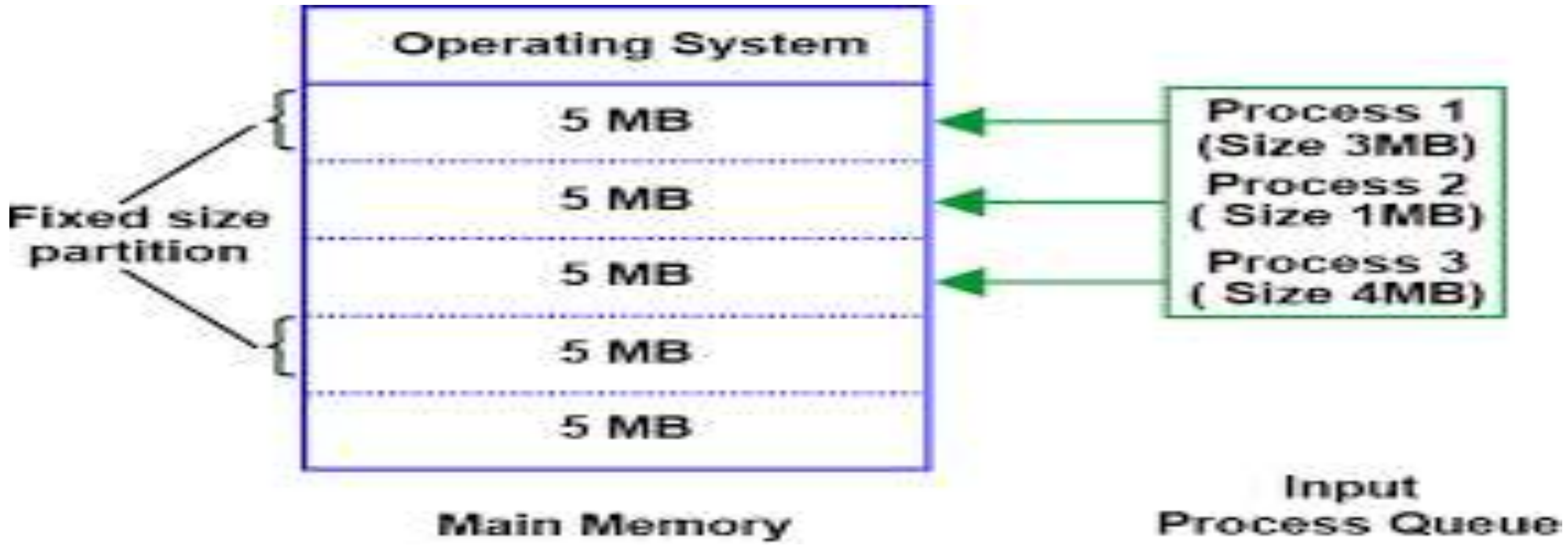
This allocation can be done in two ways:

- ✓ Fixed-size Partition Scheme
- ✓ Variable-size Partition Scheme

MEMORY MANAGEMENT

Fixed-size Partition Scheme:

In this type of contiguous memory allocation technique, each process is allotted a fixed-size continuous block in the main memory. That means there will be continuous blocks of fixed size into which the complete memory will be divided, and each time a process comes in, it will be allotted one of the free blocks. Because irrespective of the size of the process, each is allotted a block of the same size memory space. This technique is also called **static partitioning**.



MEMORY MANAGEMENT

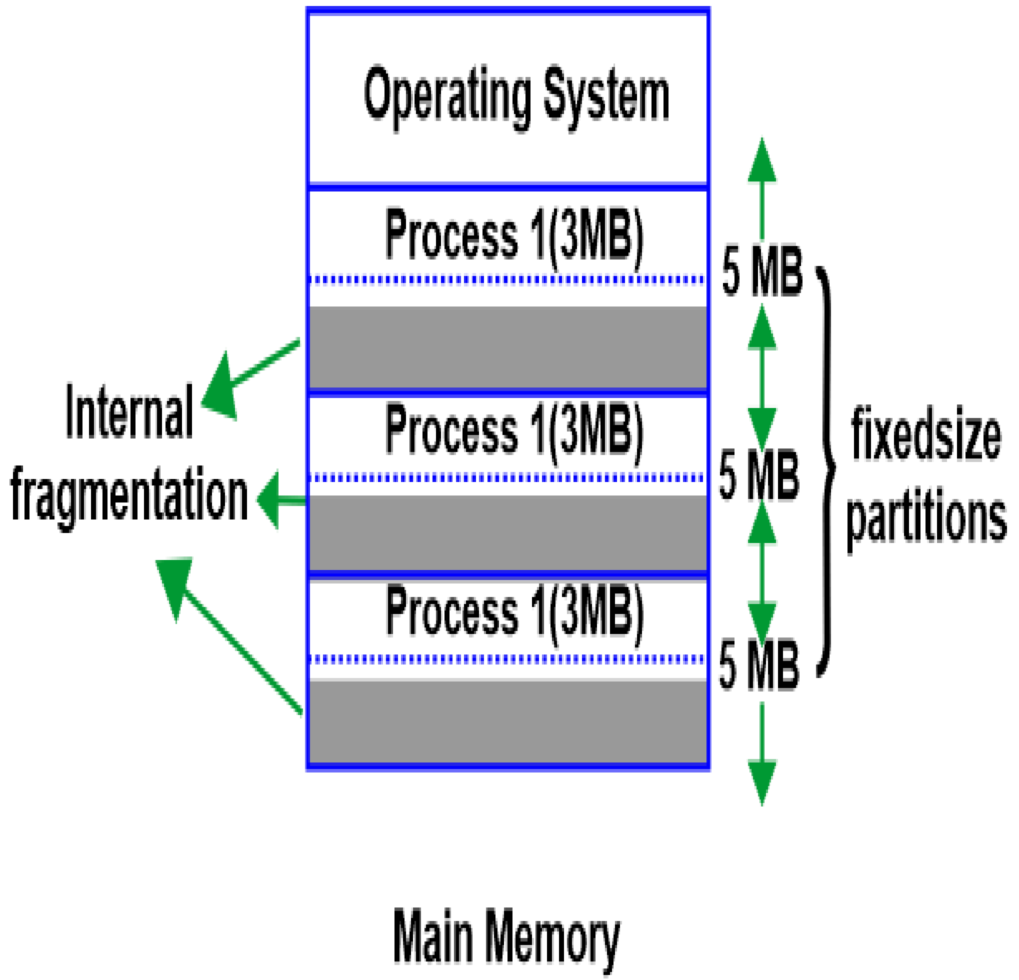
Fixed-size Partition Scheme:

Advantages:

- Supports Multiprogramming
- Simple to implement
- Tracking of blocks is easy

Disadvantages:

- Allocating space to process which have size greater then block size is not possible
- size of the blocks decides the **degree of multiprogramming**
- **Internal Fragmentation**



MEMORY MANAGEMENT

Variable-size Partition Scheme

In this type of contiguous memory allocation technique, no fixed blocks or partitions are made in the memory. Instead, each process is allotted a variable-sized block depending upon its requirements.

As the blocks are variable-sized, which is decided as processes arrive, this scheme is also called **Dynamic Partitioning**.

MEMORY MANAGEMENT

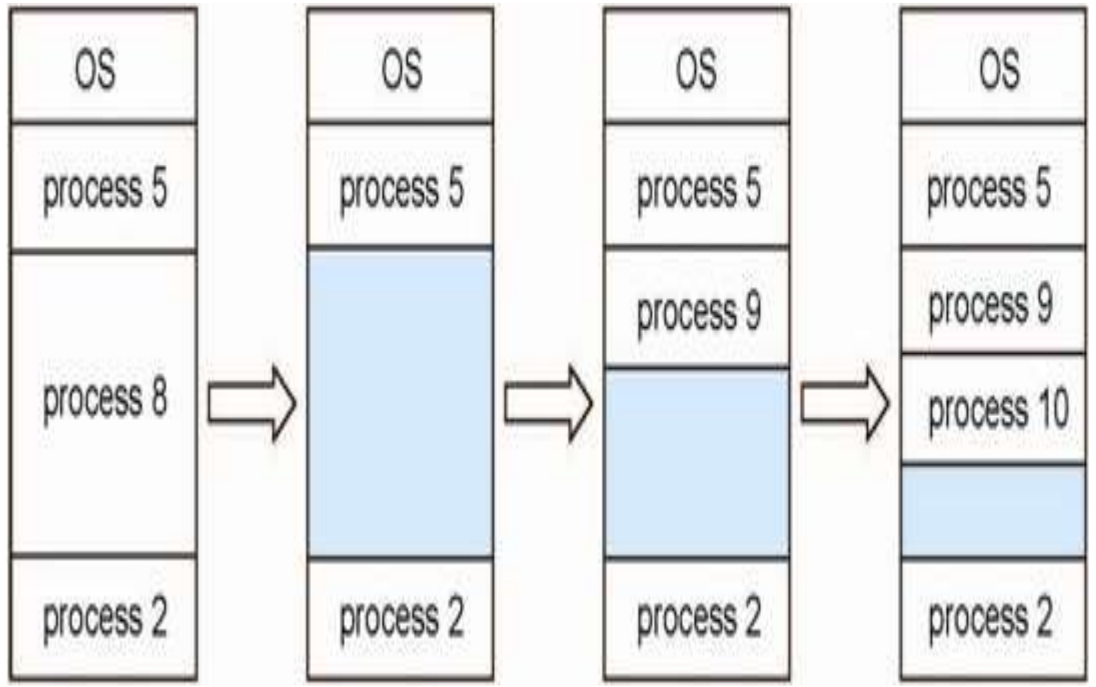
Variable-size Partition Scheme

Advantages:

- No Internal Fragmentation
- Degree of multiprogramming is not dependent on block size
- Large size processes could be allocated space

Disadvantages:

- a variable-size partition scheme is difficult to implement.
- It is difficult to keep track of processes and the remaining space in the memory.
- **External Fragmentation**



SOLUTION OF EXTERNAL FRAGMENTATION IS COMPACTION INCASE DYNAMIC PARTITIONS ARE RELOCATATBLE

MEMORY MANAGEMENT

Strategies Used for Contiguous Memory Allocation Input Queues

- ✓ First Fit : the first available free hole fulfil the requirement of the process allocated
- ✓ Best Fit : In the Best Fit, allocate the smallest hole that is big enough to process requirements. For this, we search the entire list, unless the list is ordered by size.
- ✓ Worst Fit : In the Worst Fit, allocate the largest available hole to process. This method produces the largest leftover hole.

MEMORY MANAGEMENT

Strategies Used for Contiguous Memory Allocation Input Queues

Example: Consider six memory holes of size 200 KB, 400 KB, 600 KB, 500 KB, 300 KB and 250 KB. These partitions need to be allocated to four processes of sizes 357 KB, 210 KB, 468 KB and 491 KB in that order. [Variable size partitioning]

Perform the allocation of processes using-

1.First Fit Algorithm
2.Best Fit Algorithm
3.Worst Fit Algorithm

Find the best allocation strategy

MEMORY MANAGEMENT

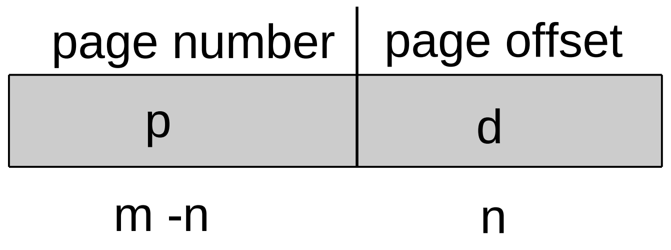
PAGING

- Physical address space of a process can be noncontiguous; process is allocated physical memory whenever the latter is available
  - Avoids external fragmentation
  - Avoids problem of varying sized memory chunks
- Divide physical memory into fixed-sized blocks called **frames**
  - Size is power of 2, between 512 bytes and 16 Mbytes
- Divide logical memory into blocks of same size called **pages**
- Keep track of all free frames
- To run a program of size ***N*** pages, need to find ***N*** free frames and load program
- Set up a **page table** to translate logical to physical addresses
- Backing store likewise split into pages
- Still have Internal fragmentation

MEMORY MANAGEMENT

PAGING

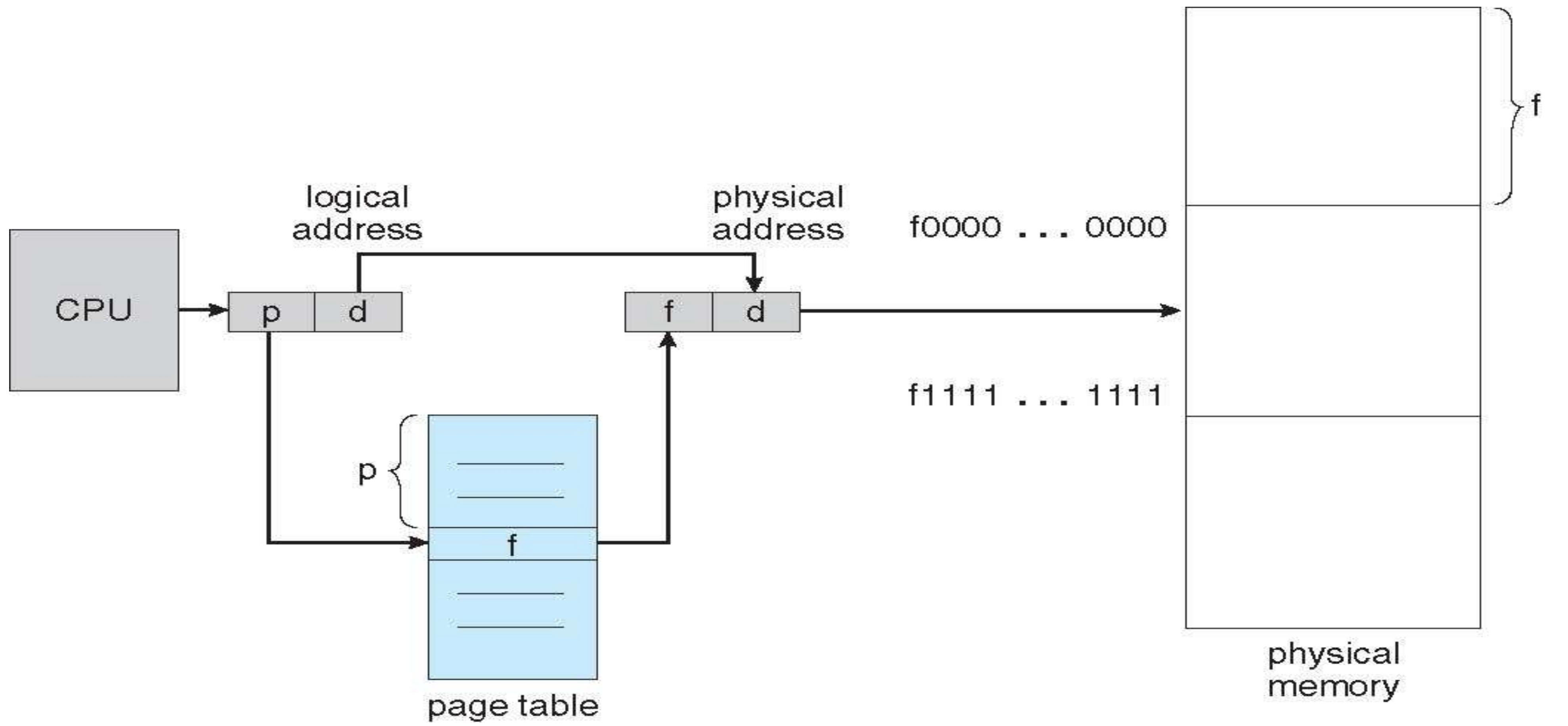
- Address generated by CPU is divided into:
  - **Page number** (***p***) – used as an index into a **page table** which contains base address of each page in physical memory
  - **Page offset** (***d***) – combined with base address to define the physical memory address that is sent to the memory unit



  - For given logical address space 2^m and page size 2^n

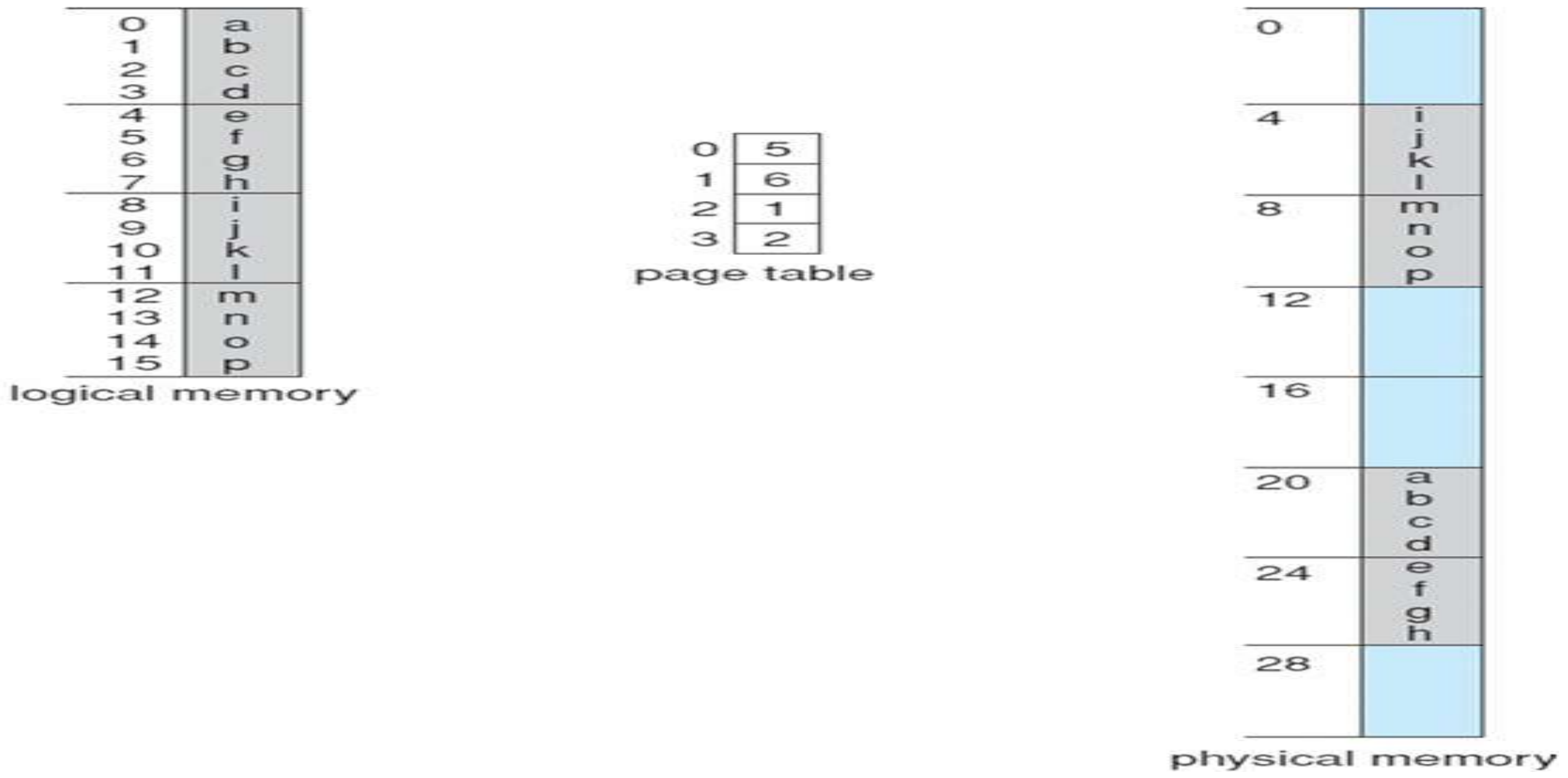
MEMORY MANAGEMENT

PAGING



MEMORY MANAGEMENT

PAGING EXAMPLE



n=2 and *m*=4 32-byte memory and 4-byte pages

MEMORY MANAGEMENT

PAGING EXAMPLE

QUESTION:

LOGICAL ADDRESS SPACE (LAS) is 128KB, PHYSICAL ADDRESS SPACE (PAS) is 512KB and PAGE SIZE is 16KB [Byte addressable]

CALCULATE FOLLOWING:

a) Number of Bits of Logical Address (LA)
b) Number of Bits of Physical Address (PA)
c) Number of pages in process (LAS)
d) Number of Frames in memory (PAS)
e) Page Table Size

MEMORY MANAGEMENT

PAGING EXAMPLE

SOLUTION:

a) Number of Bits of Logical Address (LA)

128KB = 2^{17} Bytes 17 bits will be required for LA

b) Number of Bits of Physical Address (PA)

512KB = 2^{19} Bytes 19 bits will be required for PA

c) Number of pages in process (LAS)

16KB = 2^{14} Bytes formula= LAS/Page Size 8 pages

d) Number of Frames in memory (PAS)

Page size = Frame size formula = PAS/Frame Size 32 Frames

e) Page Table Size

formula = No of Pages * Page entry size

= 8 * 5

= 40 bits

MEMORY MANAGEMENT
PAGING EXAMPLE

Assignment Question I:

GIVEN DATA :

64 MB physical memory
32 bit virtual address space.
page size is 4 KB

Consider the memory as byte addressable.

what is the approximate size of the page table?

MEMORY MANAGEMENT
PAGING EXAMPLE

Assignment Question II:

GIVEN DATA :

Calculating internal fragmentation

Page size = 2,048 bytes
Process size = 72,766 bytes

Consider the memory as byte addressable.

MEMORY MANAGEMENT

PAGING EXAMPLE

Assignment Question III:

GIVEN DATA :

Calculate the size of memory if its address consists of 22 bits and the memory is 2-byte addressable.

MEMORY MANAGEMENT

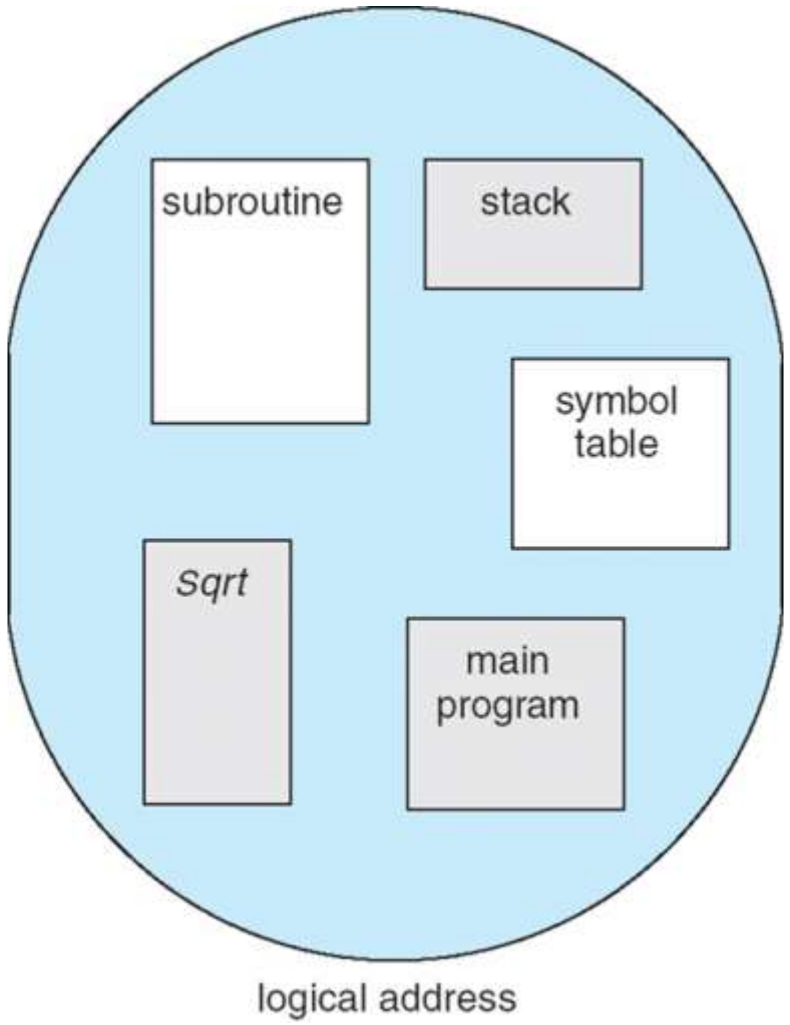
Segmentation

- Memory-management scheme that supports user view of memory
- A program is a collection of segments
  - A segment is a logical unit such as:

    main program
    procedure
    function
    method
    object
    local variables, global variables
    common block
    stack
    symbol table
    arrays

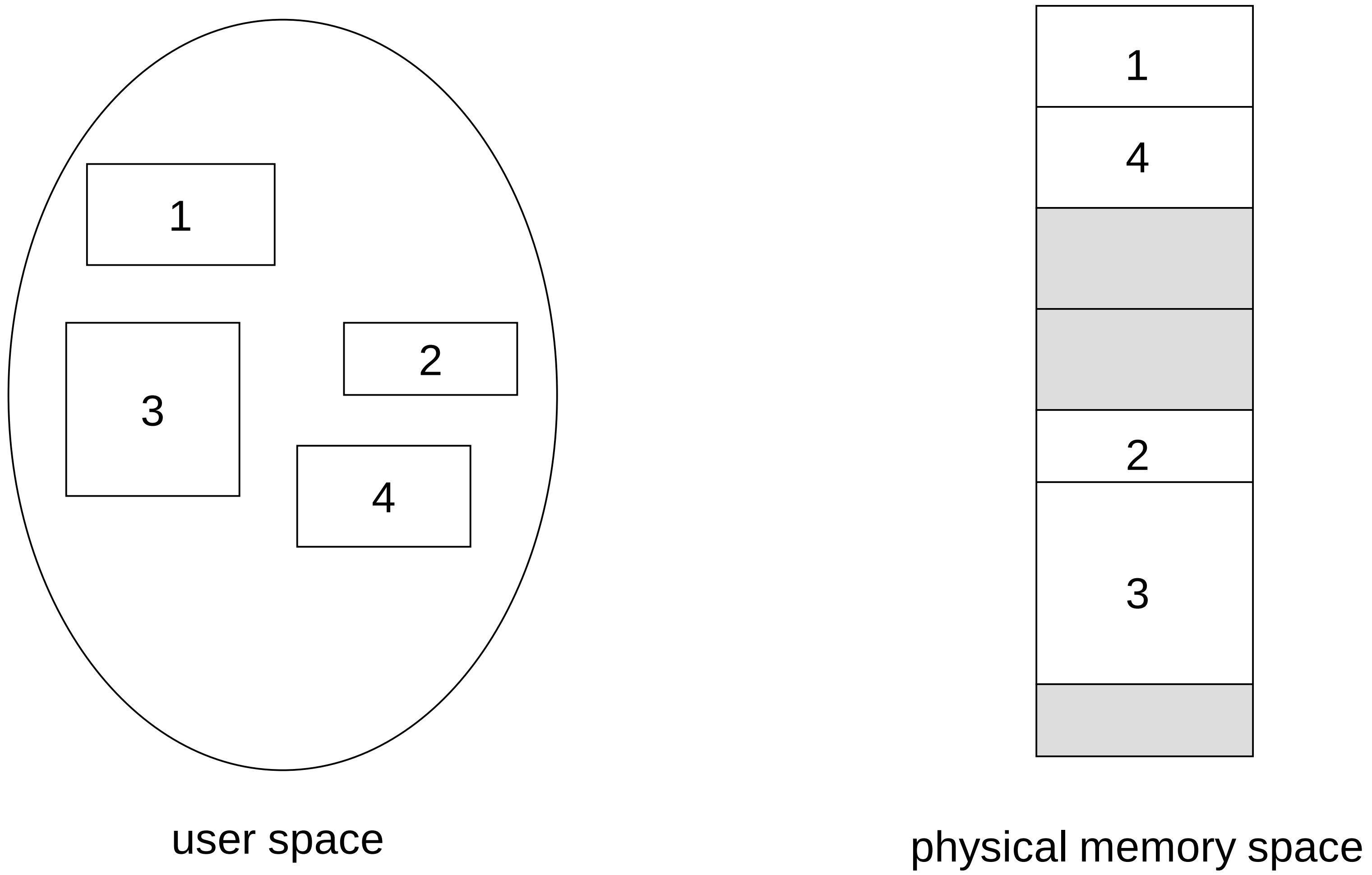
MEMORY MANAGEMENT

Segmentation: User view of Program



MEMORY MANAGEMENT

Segmentation: User view of Program



MEMORY MANAGEMENT

Segmentation:

- Logical address consists of a two tuple:

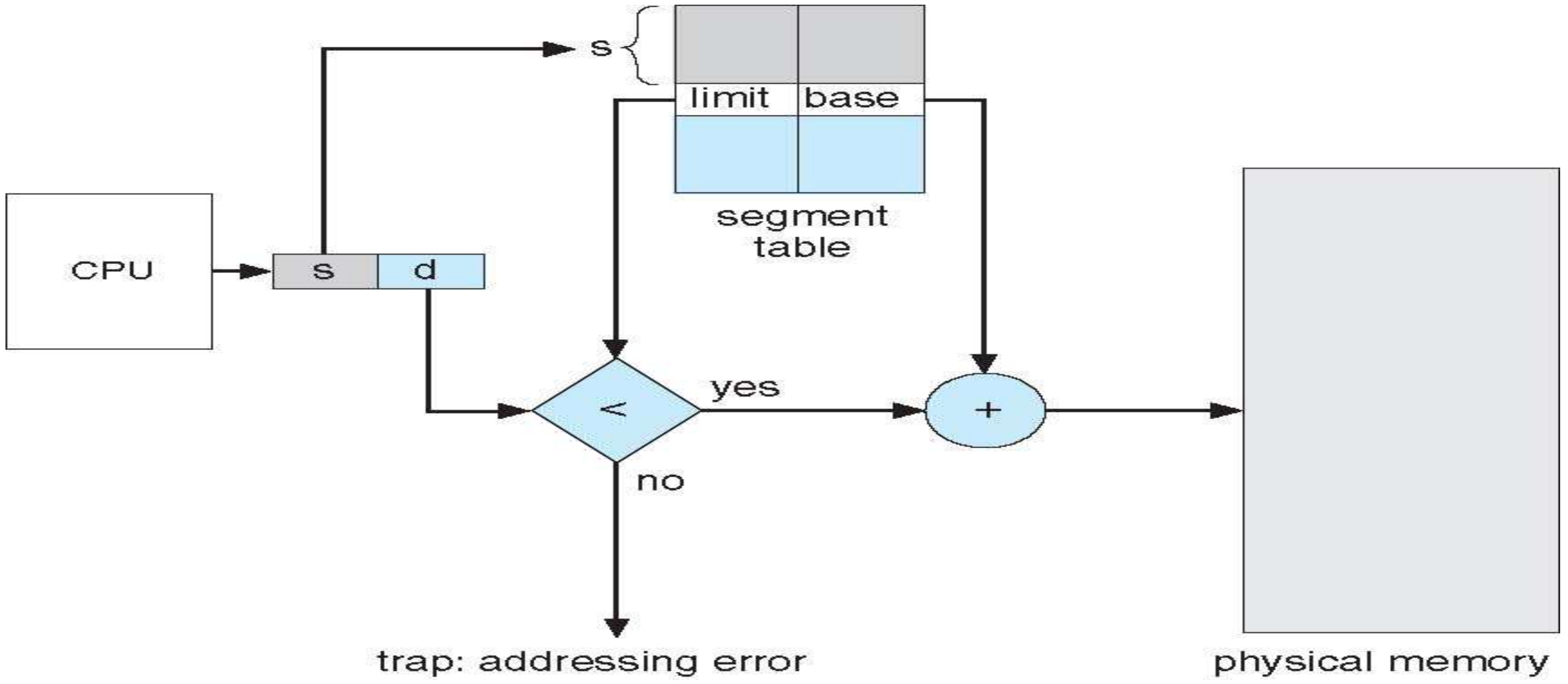
  <segment-number, offset>,

- **Segment table** – maps two-dimensional physical addresses; each table entry has:
  - **base** – contains the starting physical address where the segments reside in memory
  - **limit** – specifies the length of the segment
- **Segment-table base register (STBR)** points to the segment table's location in memory
- **Segment-table length register (STLR)** indicates number of segments used by a program;

  segment number ***s*** is legal if ***s*** < **STLR**

MEMORY MANAGEMENT

Segmentation Hardware:



MEMORY MANAGEMENT

Virtual Memory Concept: (On demand Paging)

- Processes deal with virtual memory – they have the illusion that a very large address space is available to them

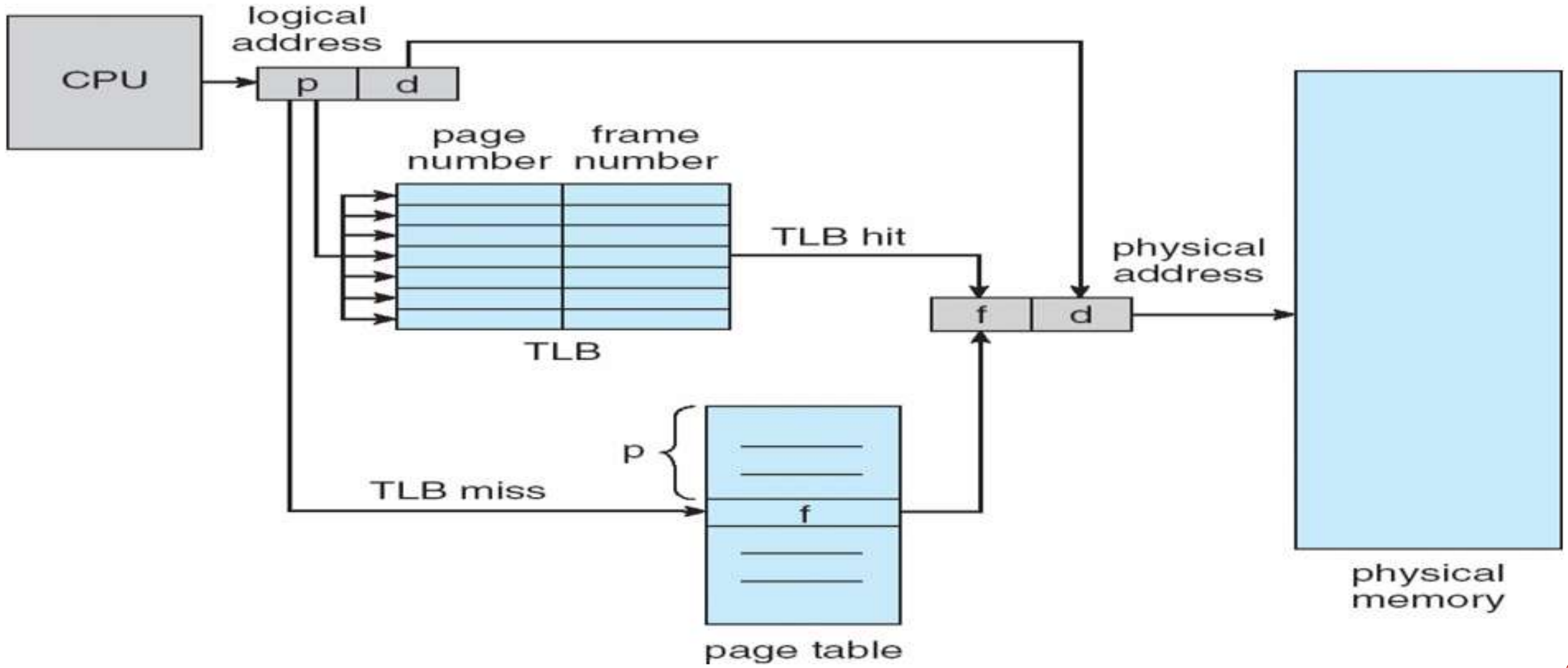
- There is only a limited amount of physical memory that is shared by all processes – a process places part of its virtual memory in this physical memory and the rest is stored on disk (called swap space)

- Thanks to locality, disk access is likely to be uncommon

- The hardware ensures that one process cannot access the memory of a different process

MEMORY MANAGEMENT

Virtual Memory Concept:



MEMORY MANAGEMENT

Virtual Memory Concept:

Effective Memory Access Time = Hit Ratio of TLB * { Access Time of TLB + Access Time of Memory} + Miss Ratio of TLB * { Access Time of TLB + (L+1) * Access Time of Memory}

Where L stands for Number of levels of page table

MEMORY MANAGEMENT

Numerical on EAT

Consider a single level paging scheme with a TLB. Assume no page fault occurs. It takes 20 ns to search the TLB and 100 ns to access the physical memory. If TLB hit ratio is 80%, the effective memory access time is _______ ns.

MEMORY MANAGEMENT

Numerical on EAT

Solution:

Effective Access Time

= 0.8 x { 20 ns + 100 ns } + 0.2 x { 20 ns + (1+1) x 100 ns }

= 0.8 x 120 ns + 0.2 + 220 ns

= 96 ns + 44 ns

= 140 ns

Thus, effective memory access time = 140 ns.

MEMORY MANAGEMENT

PAGE REPLACEMENT ALGORITHMS

Page Fault: A page fault happens when a running program accesses a memory page that is mapped into the virtual address space but not loaded in physical memory. Since actual physical memory is much smaller than virtual memory, page faults happen. In case of a page fault, Operating System might have to replace one of the existing pages with the newly needed page. Different page replacement algorithms suggest different ways to decide which page to replace. The target for all algorithms is to reduce the number of page faults.

MEMORY MANAGEMENT

Page Replacement Algorithms

- ✓ FIFO
- ✓ LRU
- ✓ Optimal Page replacement