Operating System (OS)

Process Synchronization

- When two or more process cooperates with each other, their order of execution must be preserved otherwise there can be conflicts in their execution and inappropriate outputs can be produced.
- A cooperative process is the one which can affect the execution of other process or can be affected by the execution of other process. Such processes need to be synchronized so that their order of execution can be guaranteed.

- The procedure involved in preserving the appropriate order of execution of cooperative processes is known as Process Synchronization.

Race Condition

- A Race Condition typically occurs when two or more threads try to read, write and possibly make the decisions based on the memory that they are accessing concurrently.

Critical Section

- The regions of a program that try to access shared resources and may cause race conditions are called critical section. To avoid race condition among the processes, we need to assure that only one process at a time can execute

Mutual Exclusion

- **Mutual Exclusion** is a property of process synchronization that states that “no two processes can exist in the critical section at any given point of time”.

Mutual exclusion methods are used in concurrent programming to avoid the simultaneous use of a common resource, such as a global variable, by pieces of computer code called critical sections

The requirement of mutual exclusion is that when process P1 is accessing a shared resource R1, another process should not be able to access resource R1 until process P1 has finished its operation with resource R1.

Examples of such resources include files, I/O devices such as printers, and shared data structures.

Conditions Required for Mutual Exclusion

According to the following four criteria, mutual exclusion is applicable:

- When using shared resources, it is important to ensure mutual exclusion between various processes. There cannot be two processes running simultaneously in either of their critical sections.
- It is not advisable to make assumptions about the relative speeds of the unstable processes.

- For access to the critical section, a process that is outside of it must not obstruct another process.
- Its critical section must be accessible by multiple processes in a finite amount of time; multiple processes should never be kept waiting in an infinite loop.

Approaches To Implementing Mutual Exclusion

1. Software method: Leave the responsibility to the processes themselves. These methods are usually highly error-prone and carry high overheads.

2. Hardware method: Special-purpose machine instructions are used for accessing shared resources. This method is faster but cannot provide a complete solution. Hardware solutions cannot give guarantee the absence of deadlock and starvation.

Semaphores

- Semaphores are just normal variables used to coordinate the activities of multiple processes in a computer system. They are used to enforce mutual exclusion, avoid race conditions, and implement synchronization between processes.

- The process of using Semaphores provides two operations: wait (P) and signal (V). The wait operation decrements the value of the semaphore, and the signal operation increments the value of the semaphore. When the value of the semaphore is zero, any process that performs a wait operation will be blocked until another process performs a signal operation.

- Semaphores are used to implement critical sections, which are regions of code that must be executed by only one process at a time. By using semaphores, processes can coordinate access to shared resources, such as shared memory or I/O devices.

- Semaphores are of two types:
- **Binary Semaphore –**
  This is also known as a mutex lock. It can have only two values – 0 and 1. Its value is initialized to 1. It is used to implement the solution of critical section problems with multiple processes.
- **Counting Semaphore –**
  Its value can range over an unrestricted domain. It is used to control access to a resource that has multiple instances.

Critical Section Problem

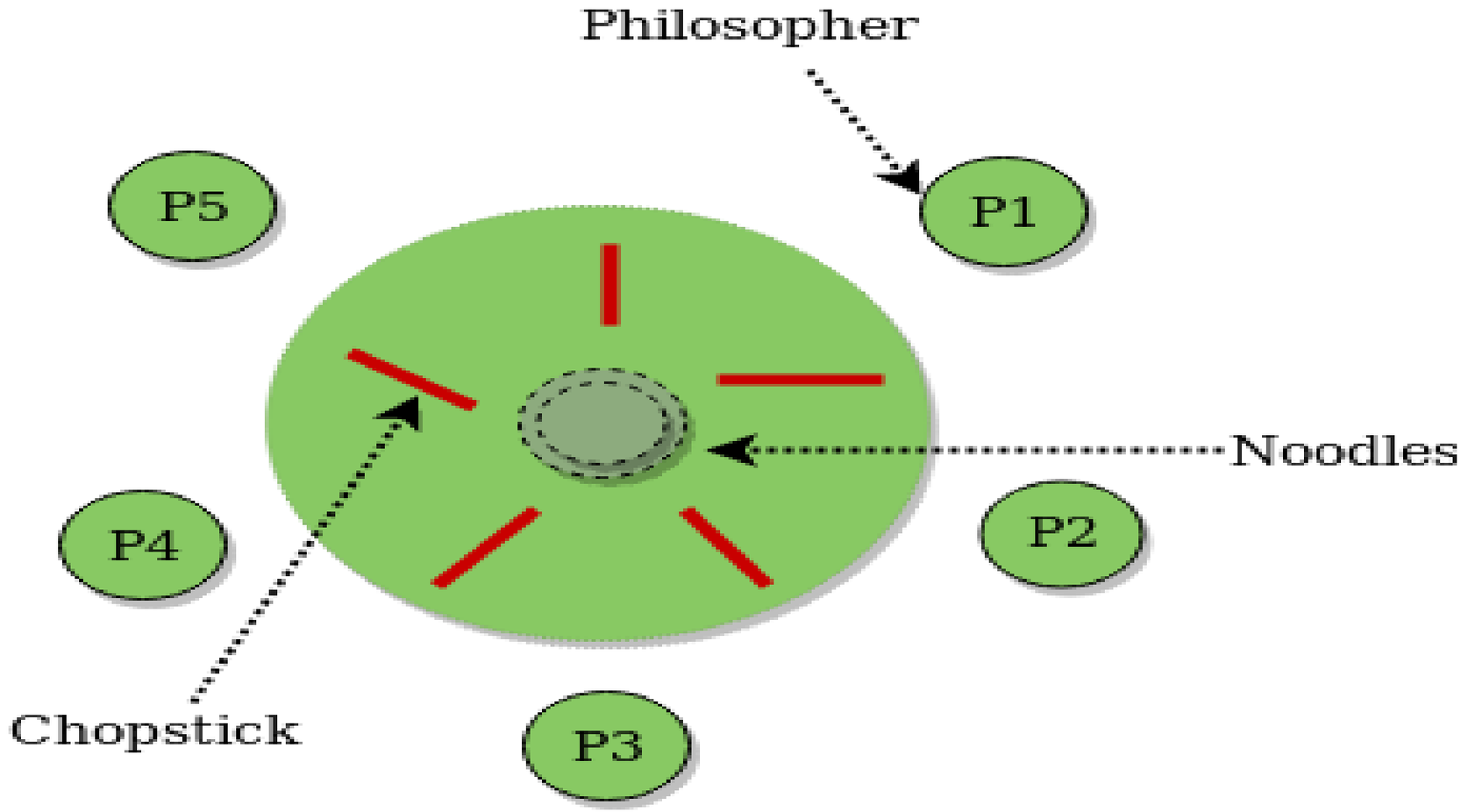
- Critical Section is the part of a program which tries to access shared resources. That resource may be any resource in a computer like a memory location, Data structure, CPU or any IO device.
- The critical section cannot be executed by more than one process at the same time; operating system faces the difficulties in allowing and disallowing the processes from entering the critical section.

- The critical section problem is used to design a set of protocols which can ensure that the Race condition among the processes will never arise.

Dining Philosopher Problem

The dining philosopher's problem is the classical problem of synchronization which says that Five philosophers are sitting around a circular table and their job is to think and eat alternatively. A bowl of noodles is placed at the center of the table along with five chopsticks for each of the philosophers. To eat a philosopher needs both their right and a left chopstick. A philosopher can only eat if both immediate left and right chopsticks of the philosopher is available.

- In case if both immediate left and right chopsticks of the philosopher are not available then the philosopher puts down their (either left or right) chopstick and starts thinking again.
- The dining philosopher demonstrates a large class of concurrency control problems hence it's a classic synchronization problem.


Philosopher
P5
P1
Noodles
P4
P2
Chopstick
P3

Solution of the Dining Philosophers Problem

- We use a semaphore to represent a chopstick and this truly acts as a solution of the Dining Philosophers Problem. Wait and Signal operations will be used for the solution of the Dining Philosophers Problem, for picking a chopstick wait operation can be executed while for releasing a chopstick signal semaphore can be executed.

Algorithm

01. Define the number of philosophers

02. Declare one thread per philosopher

03. Declare one semaphore (represent chopsticks) per philosopher

04. Declare food count.

05. When a philosopher is hungry and if food is less than the available amount

a. See if chopsticks on both sides are free

i. Acquire both chopsticks or eat

ii. restore the chopsticks

b. If chopsticks aren’t free

i. Wait till they are available

06. If food is over, stop else continue

Barber shop problem

- The sleeping barber problem is a classical process synchronization problem. There is a barber-shop having a single barber, one barber chair, and n chairs for the waiting customers. If there is no customer, then the barber sits and sleeps on his own chair. When a customer enters, he wakes up the barber

- Three semaphores are used in the solution to this issue. The first counts the number of customers in the waiting area and is for the customer (customer in the barber chair is not included because he is not waiting). The second mutex is used to give the mutual exclusion necessary for the process to operate, and the barber 0 or 1 is used to determine if the barber is idle or working. The client keeps a record of how many customers are currently waiting in the waiting area, and when that number equals the number of chairs in the area, the next customer exits the barbershop.

- The procedure barber is carried out when the barber arrives in the morning, forcing him to block on the semaphore clients since it is originally 0. The barber then retires to bed till the first client arrives.

Memory Organization

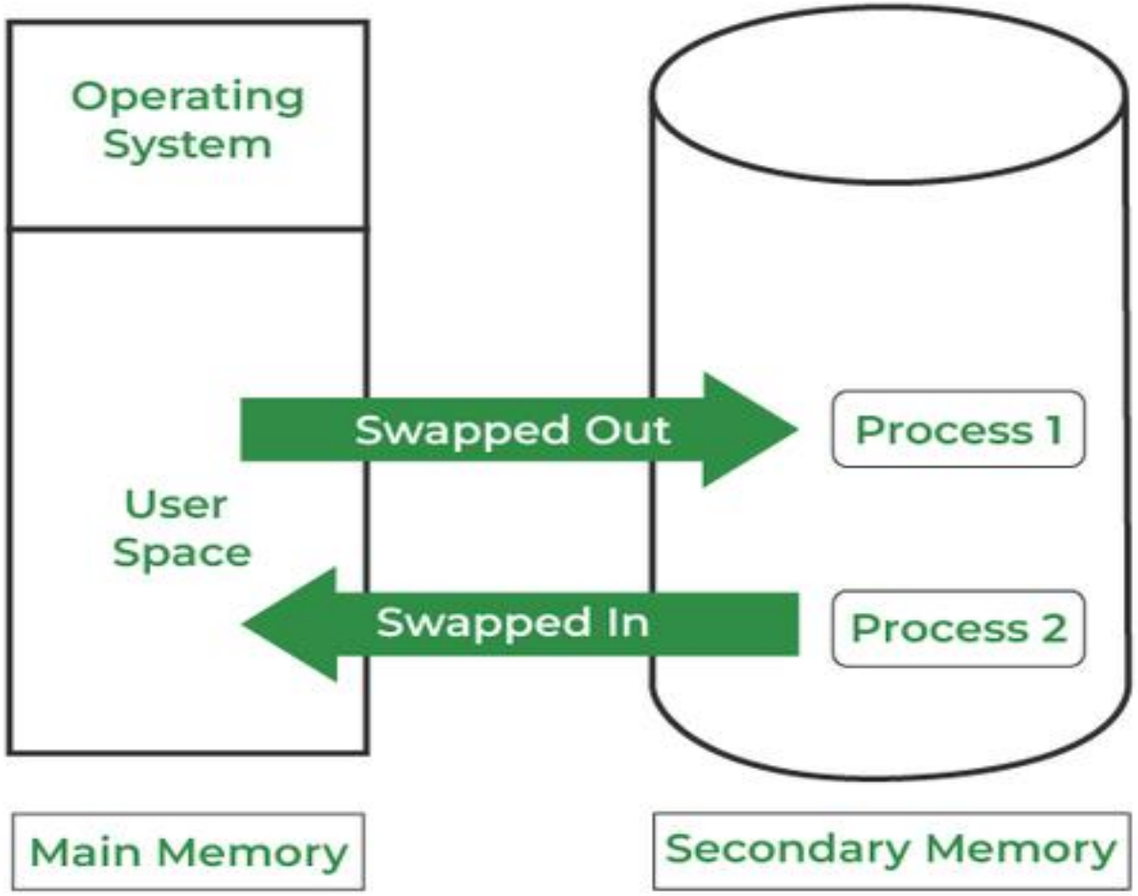
- The term memory can be defined as a collection of data in a specific format. It is used to store instructions and process data. The memory comprises a large array or group of words or bytes, each with its own location. The primary purpose of a computer system is to execute programs. These programs, along with the information they access, should be in the main memory during execution. The CPU fetches instructions from memory according to the value of the program counter.

What is Memory Management

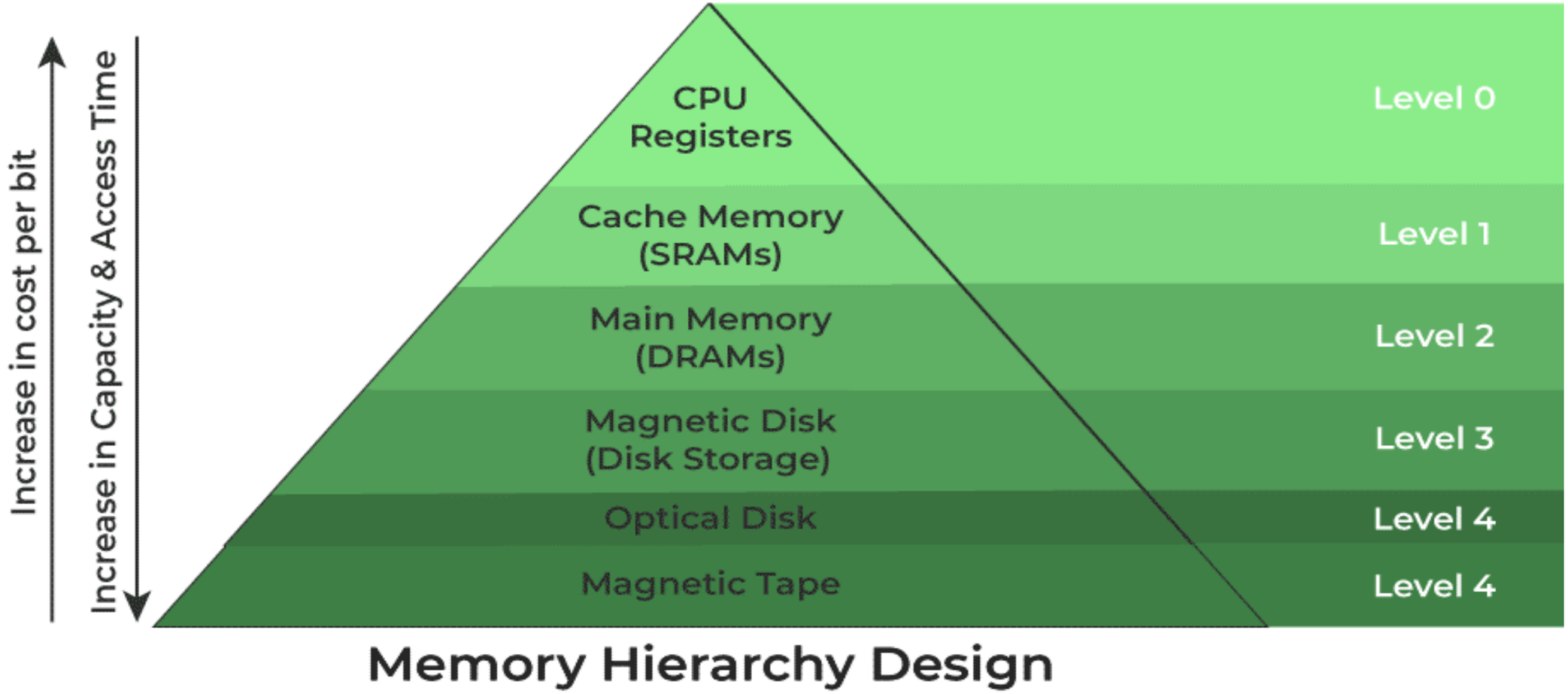
- In a multiprogramming computer, the Operating System resides in a part of memory, and the rest is used by multiple processes. The task of subdividing the memory among different processes is called Memory Management. Memory management is a method in the operating system to manage operations between main memory and disk during process execution. The main aim of memory management is to achieve efficient utilization of memory.

Why Memory Management is Required?

- 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.


Operating System
Swapped Out
Process 1
User Space
Swapped In
Process 2
Main Memory
Secondary Memory

Memory Hierarchy



1. Registers

- 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 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, 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.

4. Secondary Storage

Secondary storage, such as hard disk drives (HDD) and solid-state drives (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 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 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.

Memory Management Strategies

Paging

- In an operating system, paging is a memory management technique that allows the operating system to efficiently manage large amounts of memory by dividing it into fixed-size blocks called pages. Each page is typically 4KB in size, although other sizes can be used.
- When a program needs to access a memory address, the operating system checks to see if the page containing that address is currently in physical memory (i.e., the RAM). If the page is not in memory, the operating system retrieves it from the disk and loads it into an available page frame in physical memory.

- The process of moving pages between physical memory and the disk is called paging, and it is managed by the operating system’s memory manager. Paging allows the operating system to use more memory than is physically available, by swapping pages in and out of physical memory as needed.

Segmentation

- Segmentation is a memory management technique used by operating systems to enable processes to allocate memory in a more flexible manner. In segmentation, the memory is divided into segments, each of which represents a logical unit such as a procedure, stack, or data.
- The operating system maintains a table called a segment table that contains information about each segment, including its base address and length. When a process requests memory, the operating system allocates one or more segments to the process based on its requirements.
- Segmentation provides several benefits over other memory management techniques such as paging.

Swapping

- Operating system (OS) swapping, also known as virtual memory, is a technique used by modern operating systems to manage memory resources. It allows the computer to use more memory than is physically available by moving data between RAM and the hard drive. This technique is essential for running larger applications and multiple programs simultaneously.
- When a program or application runs on a computer, it uses memory space that is allocated to it. If the memory space allocated to a program is not sufficient, the program may crash or fail to run properly. Operating systems use virtual memory to alleviate this issue. Virtual memory swaps data between the RAM and hard drive, allowing the computer to use more memory than is physically available.

- Fragmentation
- Memory management refers to the process of allocating and deallocating memory in a computer system. When a computer program needs memory to store data, it requests it from the operating system, which then allocates a block of memory to the program. When the program is done with the memory, it deallocates it, making it available for other programs to use. However, memory management is not always straightforward, and one common issue that arises is fragmentation.

- Fragmentation occurs when memory is allocated and deallocated in a way that creates small blocks of unused memory scattered throughout the system. These small blocks of unused memory are known as fragments. Over time, the number of fragments can grow, making it more difficult to allocate large contiguous blocks of memory, even though the total amount of free memory may be sufficient.
- There are two types of fragmentation: external fragmentation and internal fragmentation. External fragmentation occurs when there are enough free blocks of memory to satisfy a request, but they are not contiguous, meaning they cannot be combined to form a single larger block.

- Internal fragmentation, on the other hand, occurs when a program requests more memory than it actually needs. This results in a block of memory being allocated that is larger than necessary. The unused portion of the block is wasted and cannot be used by other programs.

- There are several techniques used to manage fragmentation. One approach is to use compaction, which involves moving allocated blocks of memory to create larger contiguous blocks. This can be an expensive operation, especially in systems with large amounts of memory. Another approach is to use dynamic memory allocation techniques that attempt to reduce fragmentation by allocating memory blocks of the appropriate size for a given request, rather than always allocating the exact amount requested.

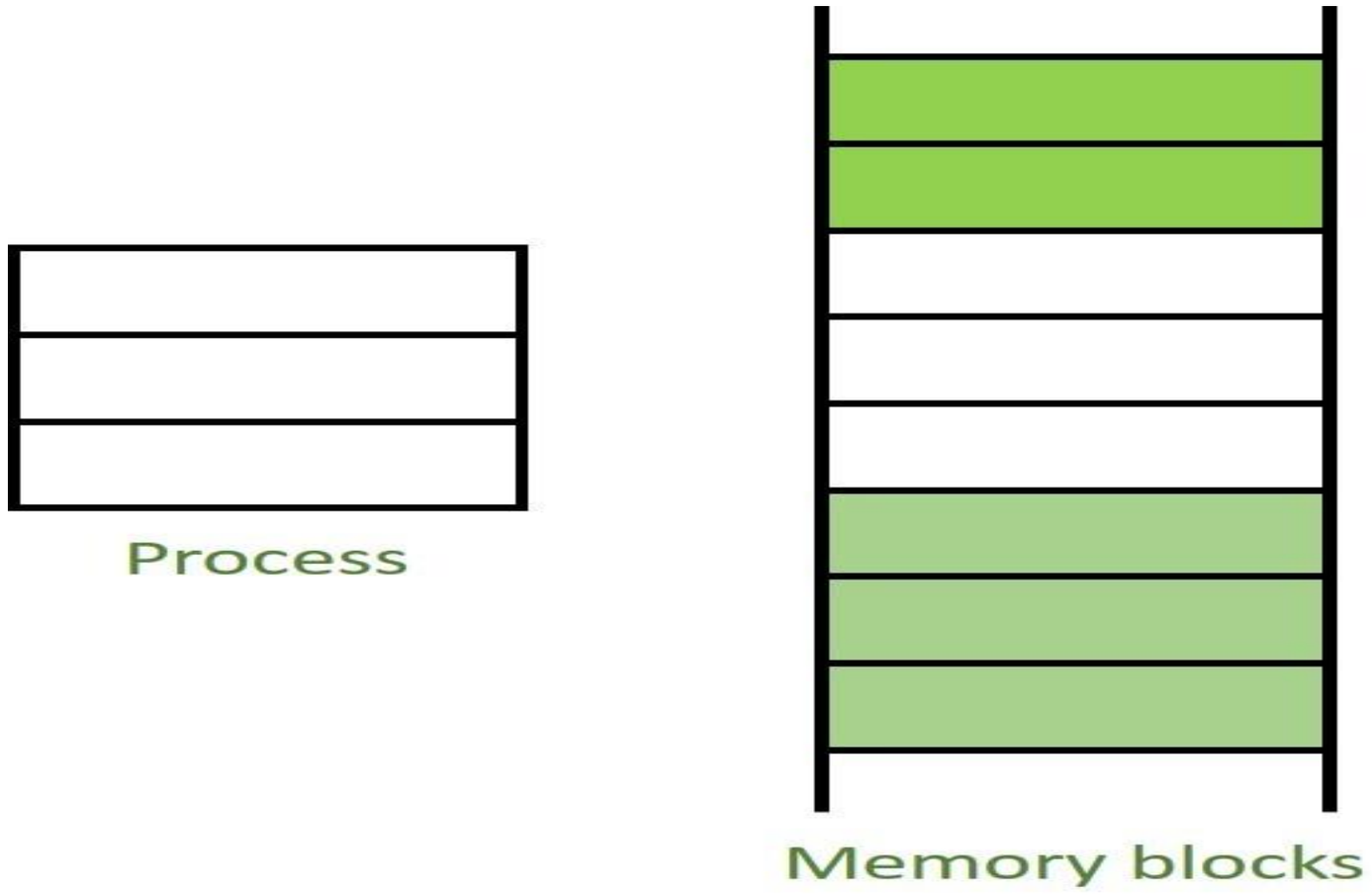
Contiguous Memory Allocation

Allocating space to software applications is referred to as memory allocation. Memory is a sizable collection of bytes. Contiguous and non-contiguous memory allocation are the two basic types of memory allocation.

Contiguous memory allocation enables the tasks to be finished in a single memory region.

Contrarily, non-contiguous memory allocation distributes the procedure throughout many memory locations in various memory sections.

- **1. Contiguous Memory Allocation :** Contiguous memory allocation is basically a method in which a single contiguous section/part of memory is allocated to a process or file needing it. Because of this all the available memory space resides at the same place together, which means that the freely/unused available memory partitions are not distributed in a random fashion here and there across the whole memory space.



Contiguous Memory Allocation

2. Non-Contiguous Memory Allocation : Non-Contiguous memory allocation is basically a method on the contrary to contiguous allocation method, allocates the memory space present in different locations to the process as per it’s requirements. As all the available memory space is in a distributed pattern so the freely available memory space is also scattered here and there. This technique of memory allocation helps to reduce the wastage of memory, which eventually gives rise to Internal and external fragmentation.

•

Difference between Contiguous and Non-contiguous Memory Allocation :

Contiguous Memory Allocation	Non-Contiguous Memory Allocation
Contiguous memory allocation allocates consecutive blocks of memory to a file/process.	Non-Contiguous memory allocation allocates separate blocks of memory to a file/process.
Faster in Execution.	Slower in Execution.
It is easier for the OS to control.	It is difficult for the OS to control.
Overhead is minimum as not much address translations are there while executing a process.	More Overheads are there as there are more address translations.

Both Internal fragmentation and external fragmentation occurs in Contiguous memory allocation method.	Only External fragmentation occurs in Non-Contiguous memory allocation method.
It includes single partition allocation and multi-partition allocation.	It includes paging and segmentation.
Wastage of memory is there.	No memory wastage is there.
In contiguous memory allocation, swapped-in processes are arranged in the originally allocated space.	In non-contiguous memory allocation, swapped-in processes can be arranged in any place in the memory.

Partition Management Techniques

In the operating system, the following are four common memory management techniques.

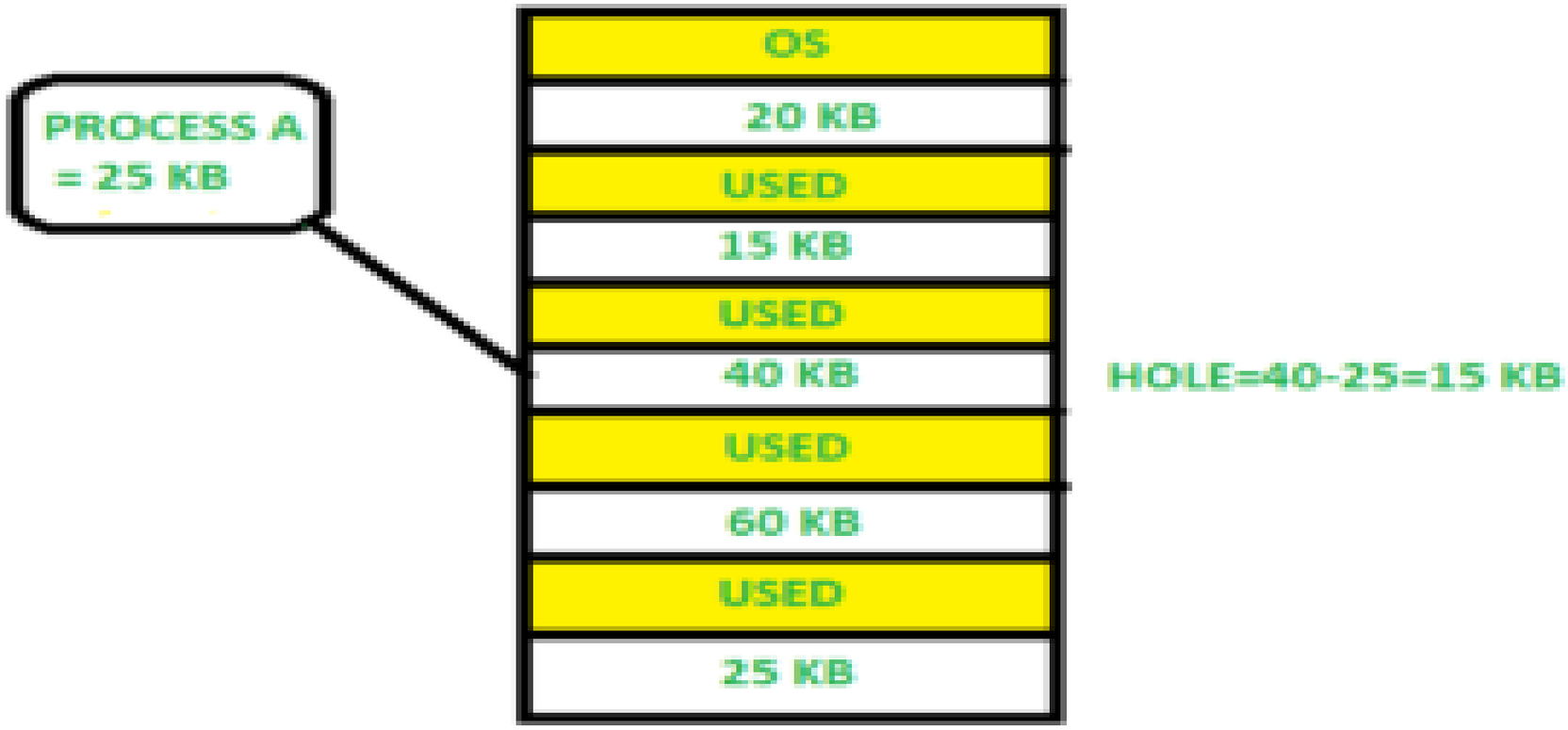
- **Single contiguous allocation:** Simplest allocation method used by MS-DOS. All memory (except some reserved for OS) is available to a process.
- **Partitioned allocation:** Memory is divided into different blocks or partitions. Each process is allocated according to the requirement.
- **Paged memory management:** Memory is divided into fixed-sized units called page frames, used in a virtual memory environment.

- **Segmented memory management:** Memory is divided into different segments (a segment is a logical grouping of the process’ data or code).In this management, allocated memory doesn’t have to be contiguous.
- Most of the operating systems (for example Windows and Linux) use Segmentation with Paging. A process is divided into segments and individual segments have pages.

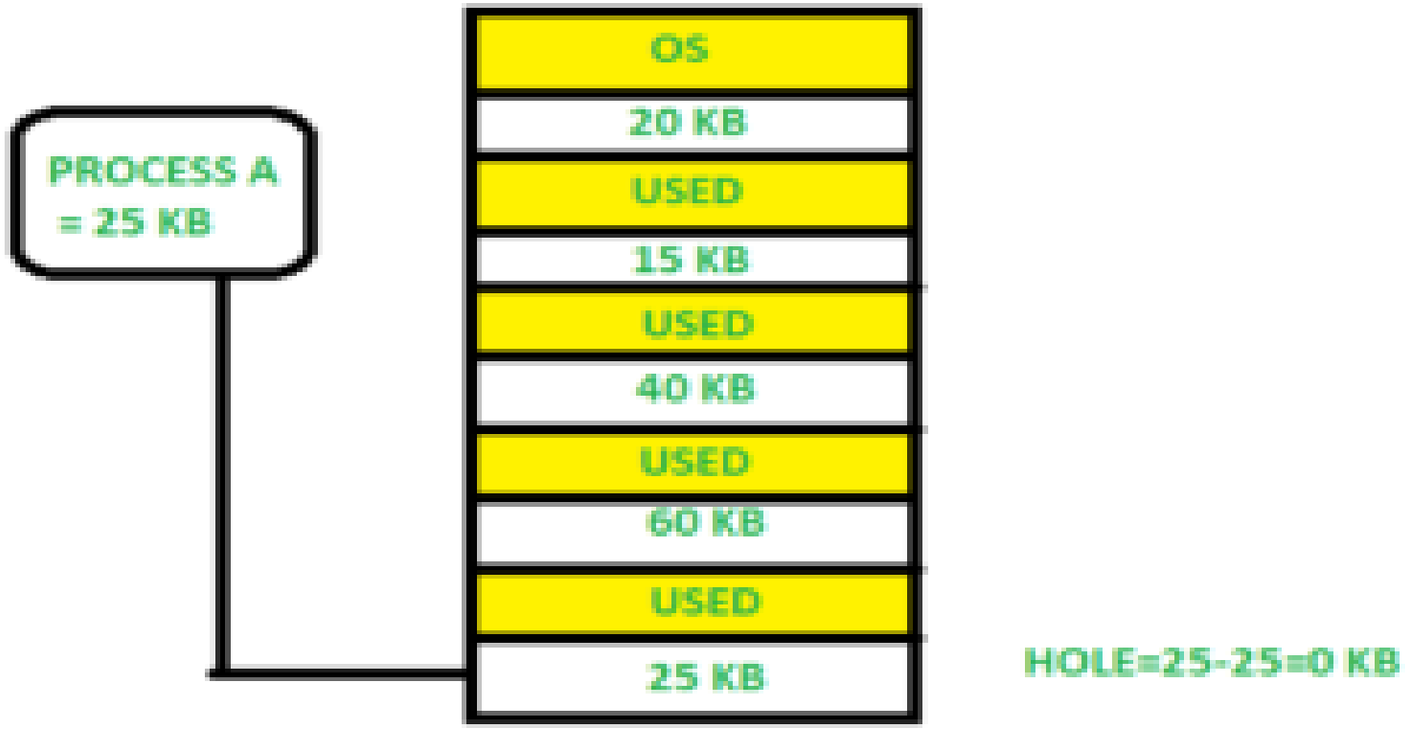
- In **Partition Allocation**, when there is more than one partition freely available to accommodate a process's request, a partition must be selected. To choose a particular partition, a partition allocation method is needed. A partition allocation method is considered better if it avoids internal fragmentation.
- When it is time to load a process into the main memory and if there is more than one free block of memory of sufficient size then the OS decides which free block to allocate.

There are different Placement Algorithm:

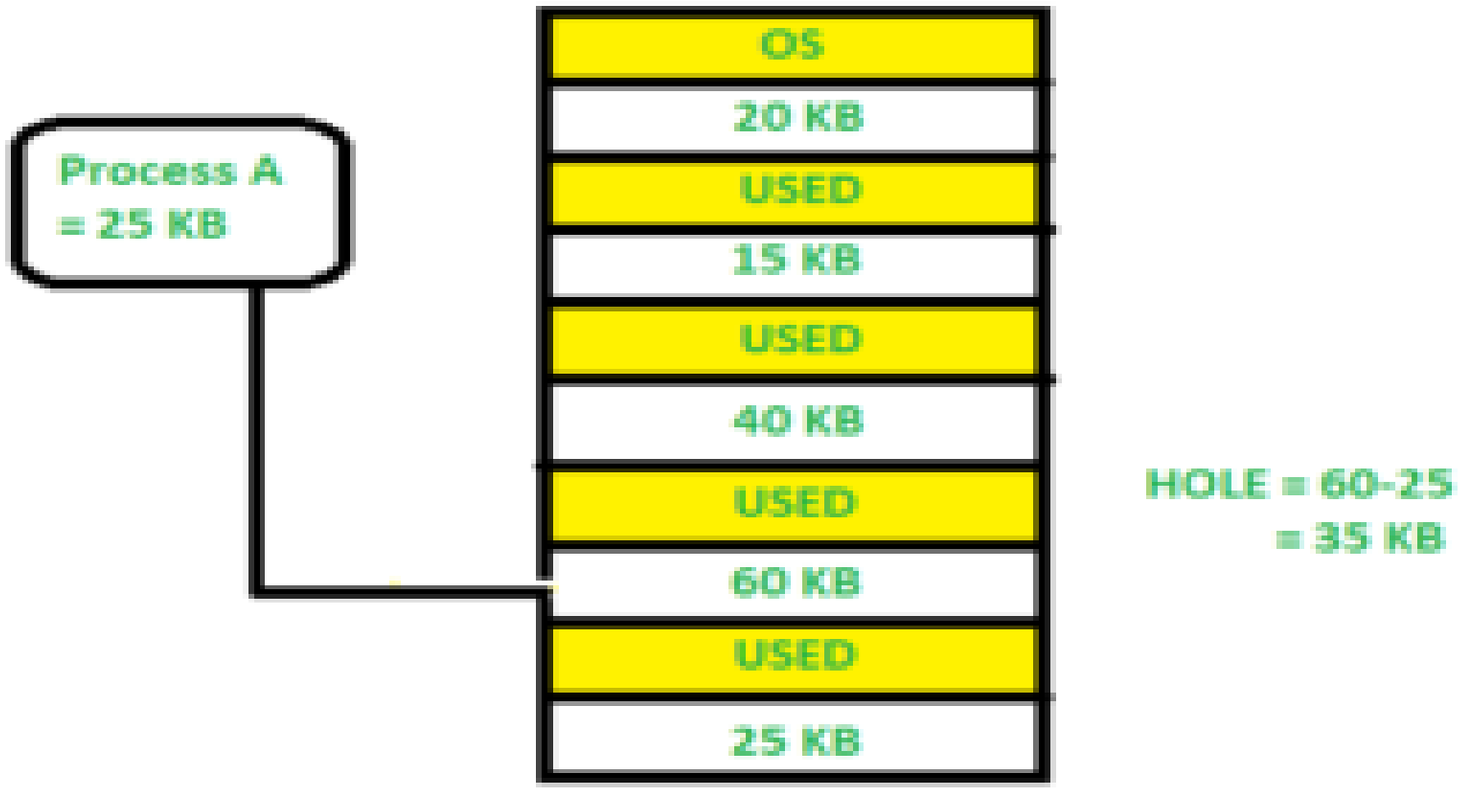
1. First Fit: In the first fit, the partition is allocated which is the first sufficient block from the top of Main Memory. It scans memory from the beginning and chooses the first available block that is large enough. Thus it allocates the first hole that is large enough.


PROCESS A
= 25 KB
OS
20 KB
USED
15 KB
USED
40 KB
USED
60 KB
USED
25 KB
HOLE=40-25=15 KB

2. Best Fit Allocate the process to the partition which is the first smallest sufficient partition among the free available partition. It searches the entire list of holes to find the smallest hole whose size is greater than or equal to the size of the process.


PROCESS A
= 25 KB
OS
20 KB
USED
15 KB
USED
40 KB
USED
60 KB
USED
25 KB
HOLE=25-25=0 KB

- **3. Worst Fit** Allocate the process to the partition which is the largest sufficient among the freely available partitions available in the main memory. It is opposite to the best-fit algorithm. It searches the entire list of holes to find the largest hole and allocate it to process.


Process A
= 25 KB
OS
20 KB
USED
15 KB
USED
40 KB
USED
60 KB
USED
25 KB
HOLE = 60-25
= 35 KB

4. Next Fit: Next fit is similar to the first fit but it will search for the first sufficient partition from the last allocation point.

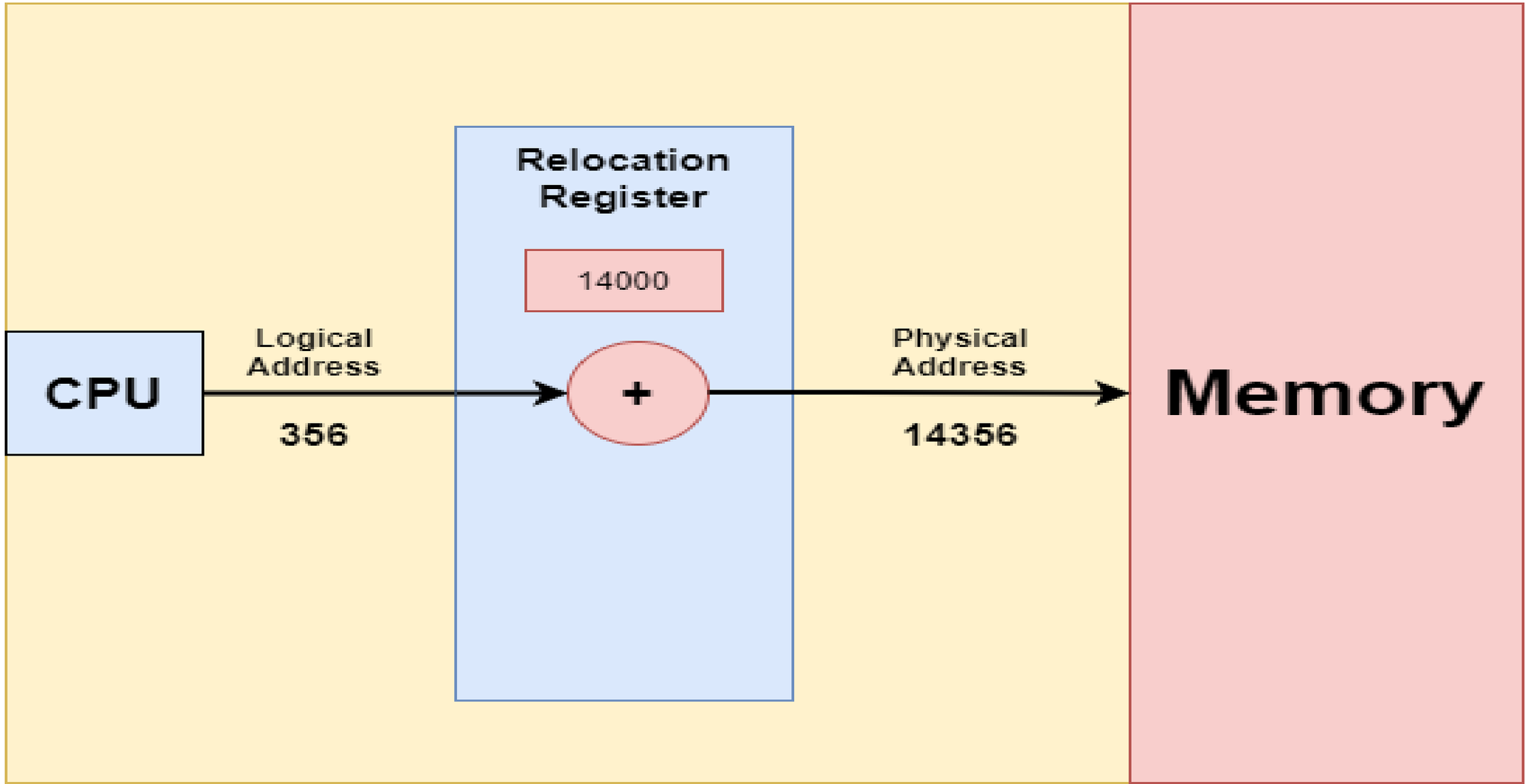
Logical versus Physical Address Space

An address generated by the CPU is commonly referred to as a **logical address** whereas an address seen by the memory unit-that is, the one loaded into the **memory address register** of the memory-is commonly referred to as a **physical address.**

The set of all logical addresses generated by a program is a **logical address space ,**the set of all physical addresses corresponding to these logical addresses is a **physical address space.**

Thus, in_ the execution-time address-binding scheme, the logical and physical address spaces differ.

The run-time mapping from virtual to physical addresses is done by a hardware device called the memory management unit(MMU)


Relocation
Register
14000
CPU
Logical
Address
356
+
Physical
Address
14356
Memory

The base register is now called a relocation register.

The value in the relocation register is added to every address generated by a user process at the time the address is sent to memory (see Figure 8.4). For example, if the base is at 14000, then an attempt by the user to address location 0 is dynamically relocated to location 14000; an access to location 346 is mapped to location 14346.

The user program never sees the real physical addresses. The program can create a pointer to location 346, store it in memory, manipulate it, and compare it with other addresses-all as the number 346. Only when it is used as a memory address (in an indirect load or store, perhaps) is it relocated relative to the base register. The user program deals with logical addresses. The memory-mapping hardware converts logical addresses into physical addresses.

Paging

Paging is a memory management scheme that eliminates the need for a contiguous allocation of physical memory. The process of retrieving processes in the form of pages from the secondary storage into the main memory is known as paging. The basic purpose of paging is to separate each procedure into pages.

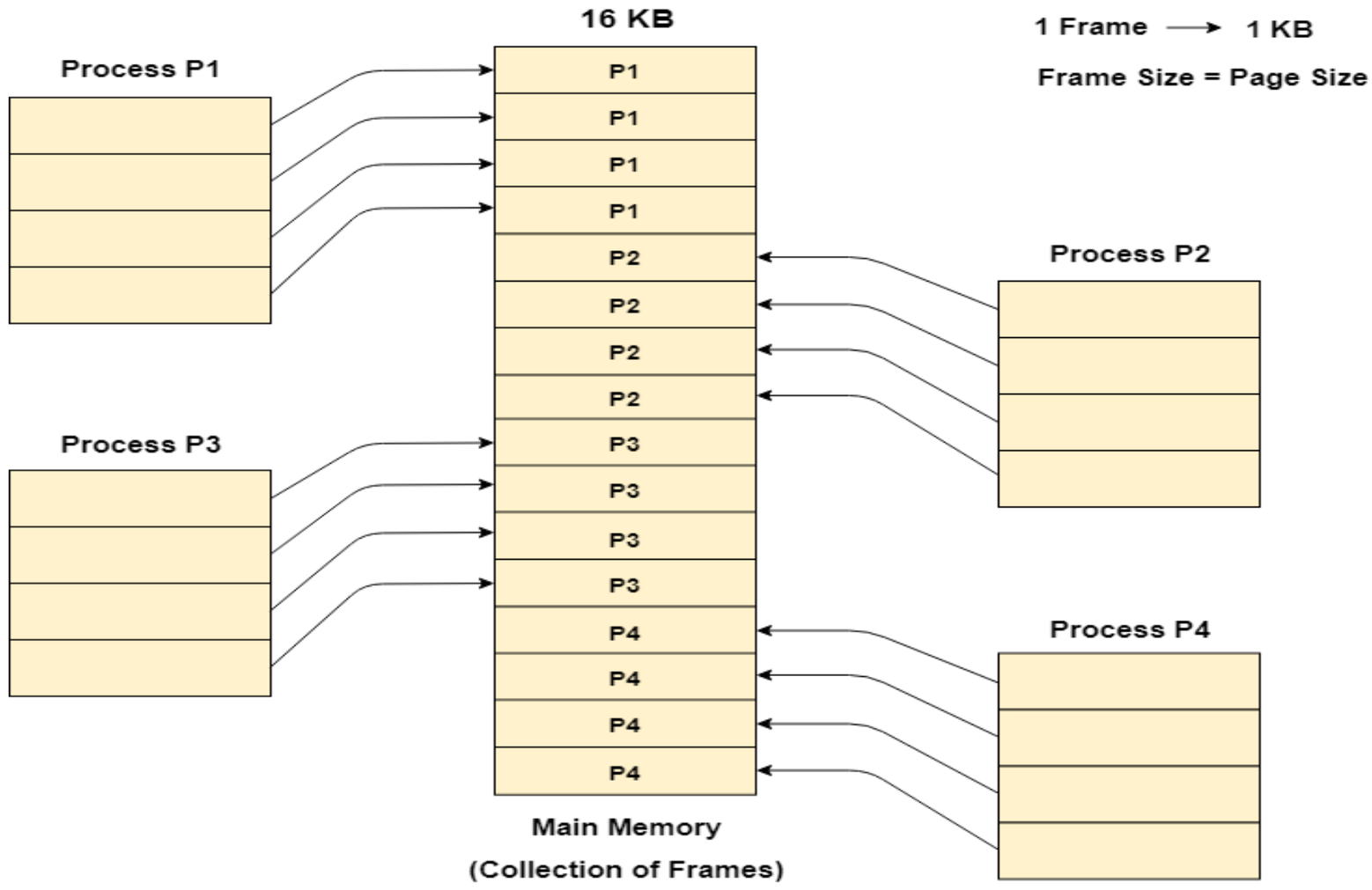
- **Logical Address or Virtual Address:** This is a deal that is generated through the CPU and used by a technique to get the right of entry to reminiscence. It is known as a logical or digital deal because it isn't always a physical vicinity in memory but an opportunity for a connection with a place inside the device's logical address location.
- **Logical Address Space or Virtual Address Space:** This is the set of all logical addresses generated via a software program. It is normally represented in phrases or bytes and is split into regular-duration pages in a paging scheme.
- **Physical Address:** This is a cope that corresponds to a bodily place in reminiscence. It is the actual cope with this that is available on the memory unit and is used by the memory controller to get admission to the reminiscence.
- **Physical Address Space:** This is the set of all bodily addresses that correspond to the logical addresses inside the way's logical deal with place. It is usually represented in words or bytes and is cut up into fixed-size frames in a paging scheme.

The mapping from virtual to physical address is done by the **Memory Management Unit (MMU)** which is a hardware device and this mapping is known as the paging technique.

The Physical Address Space is conceptually divided into a number of fixed-size blocks, called **frames**.

The Logical Address Space is also split into fixed-size blocks, called **pages**.

Page Size = Frame Size


16 KB
1 Frame → 1 KB
Frame Size = Page Size
Process P1
Process P2
Process P3
Process P4
P1
P1
P1
P1
P2
P2
P2
P2
P3
P3
P3
P3
P4
P4
P4
P4
Main Memory
(Collection of Frames)

Paging

Segmentation

A process is divided into Segments. The chunks that a program is divided into which are not necessarily all of the exact sizes are called segments. Segmentation gives the user's view of the process which paging does not provide. Here the user's view is mapped to physical memory.

Types of Segmentation in Operating System

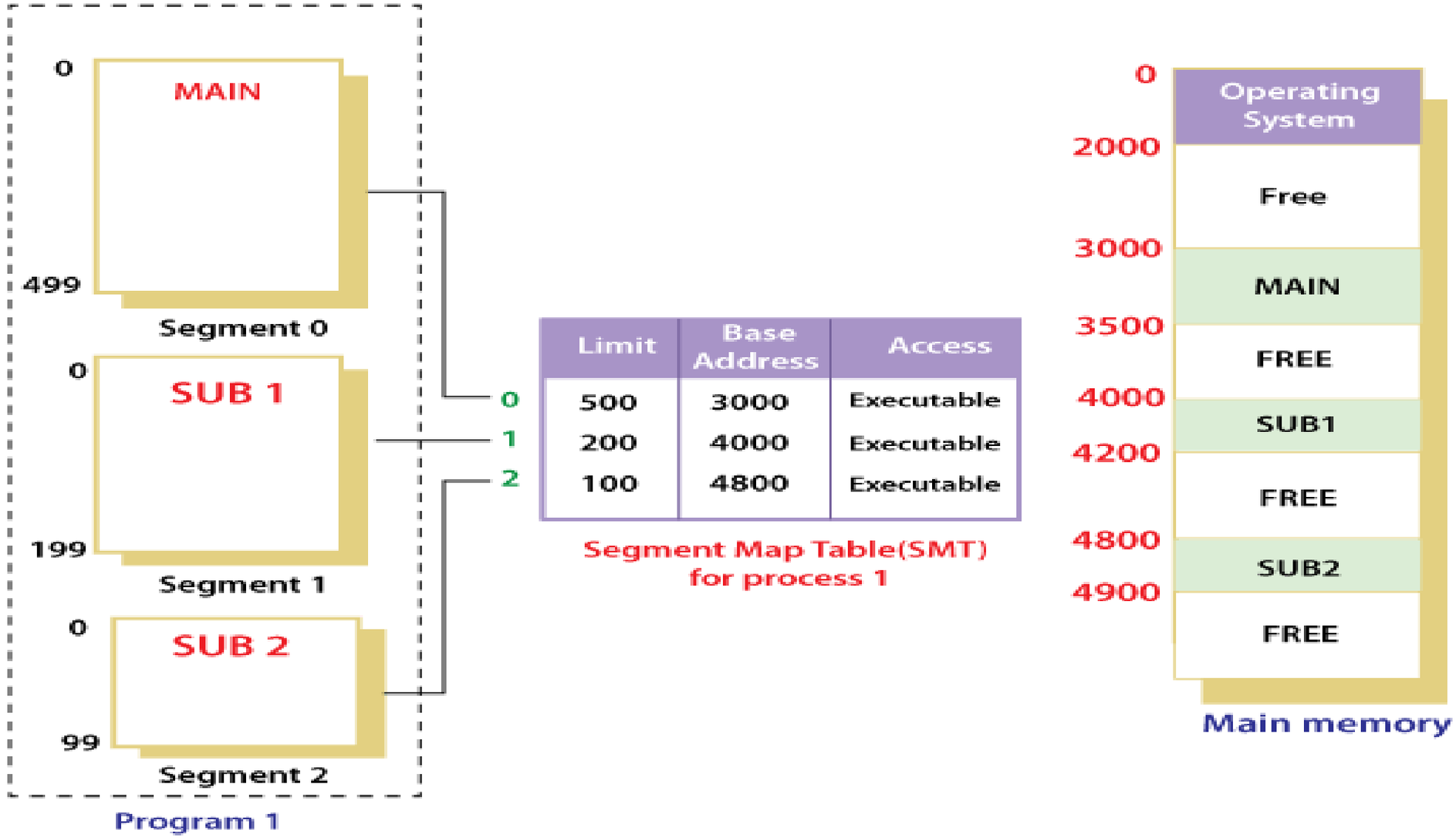
- **Virtual Memory Segmentation:** Each process is divided into a number of segments, but the segmentation is not done all at once. This segmentation may or may not take place at the run time of the program.
- **Simple Segmentation:** Each process is divided into a number of segments, all of which are loaded into memory at run time, though not necessarily contiguously.

There is no simple relationship between logical addresses and physical addresses in segmentation. A table stores the information about all such segments and is called Segment Table.

What is Segment Table?

It maps a two-dimensional Logical address into a one-dimensional Physical address. It’s each table entry has:

- **Base Address:** It contains the starting physical address where the segments reside in memory.
- **Segment Limit:** Also known as segment offset. It specifies the length of the segment.


0
MAIN
499
Segment 0
0
SUB 1
199
Segment 1
0
SUB 2
99
Segment 2
Program 1
0
1
2
Limit
Base Address
Access
500
3000
Executable
200
4000
Executable
100
4800
Executable
Segment Map Table(SMT) for process 1
0
2000
3000
3500
4000
4200
4800
4900
Operating System
Free
MAIN
FREE
SUB1
FREE
SUB2
FREE
Main memory

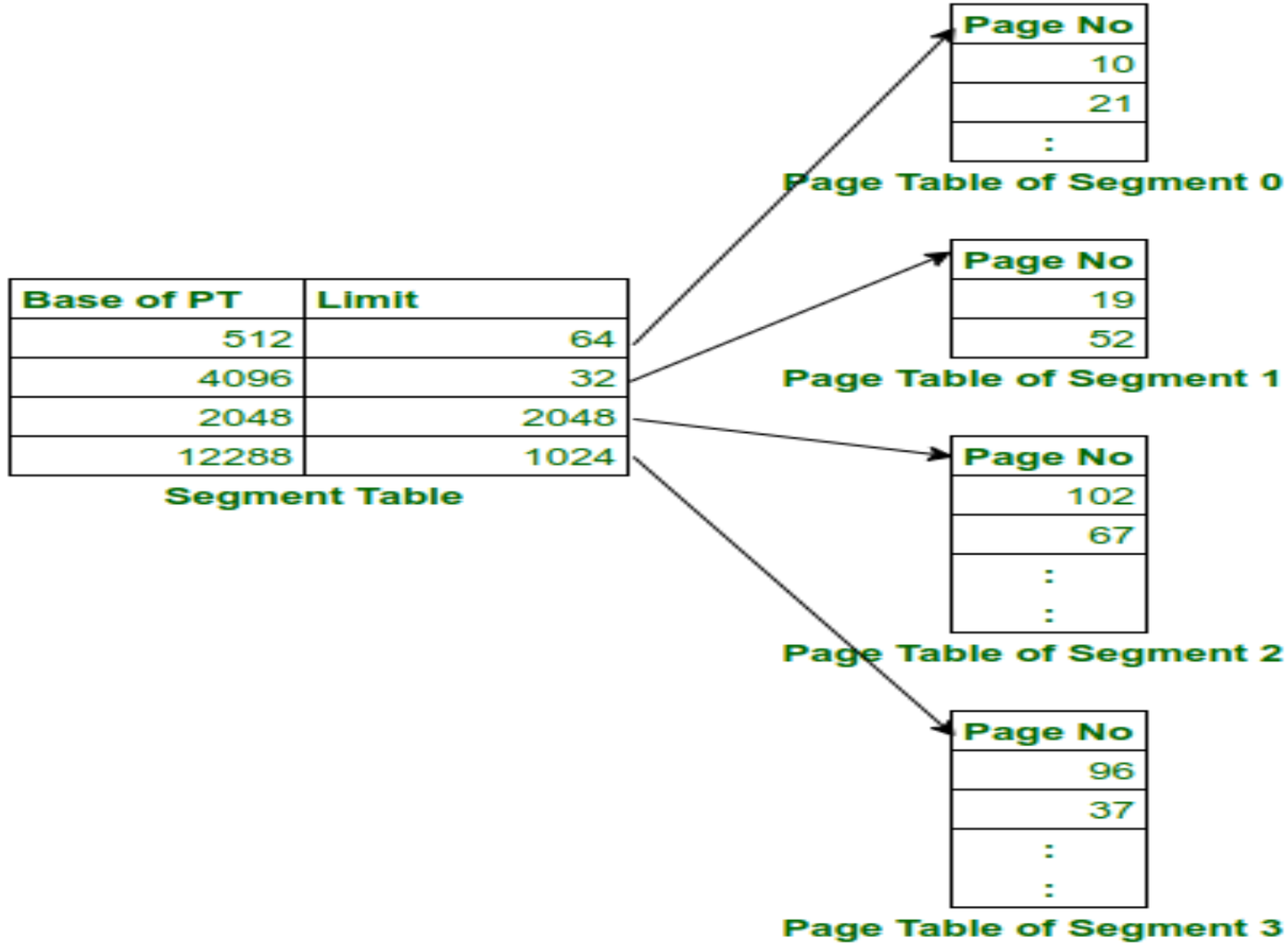
Paged Segmentation and Segmented Paging

Paged Segmentation and Segmented Paging are two different memory management techniques that combine the benefits of paging and segmentation.

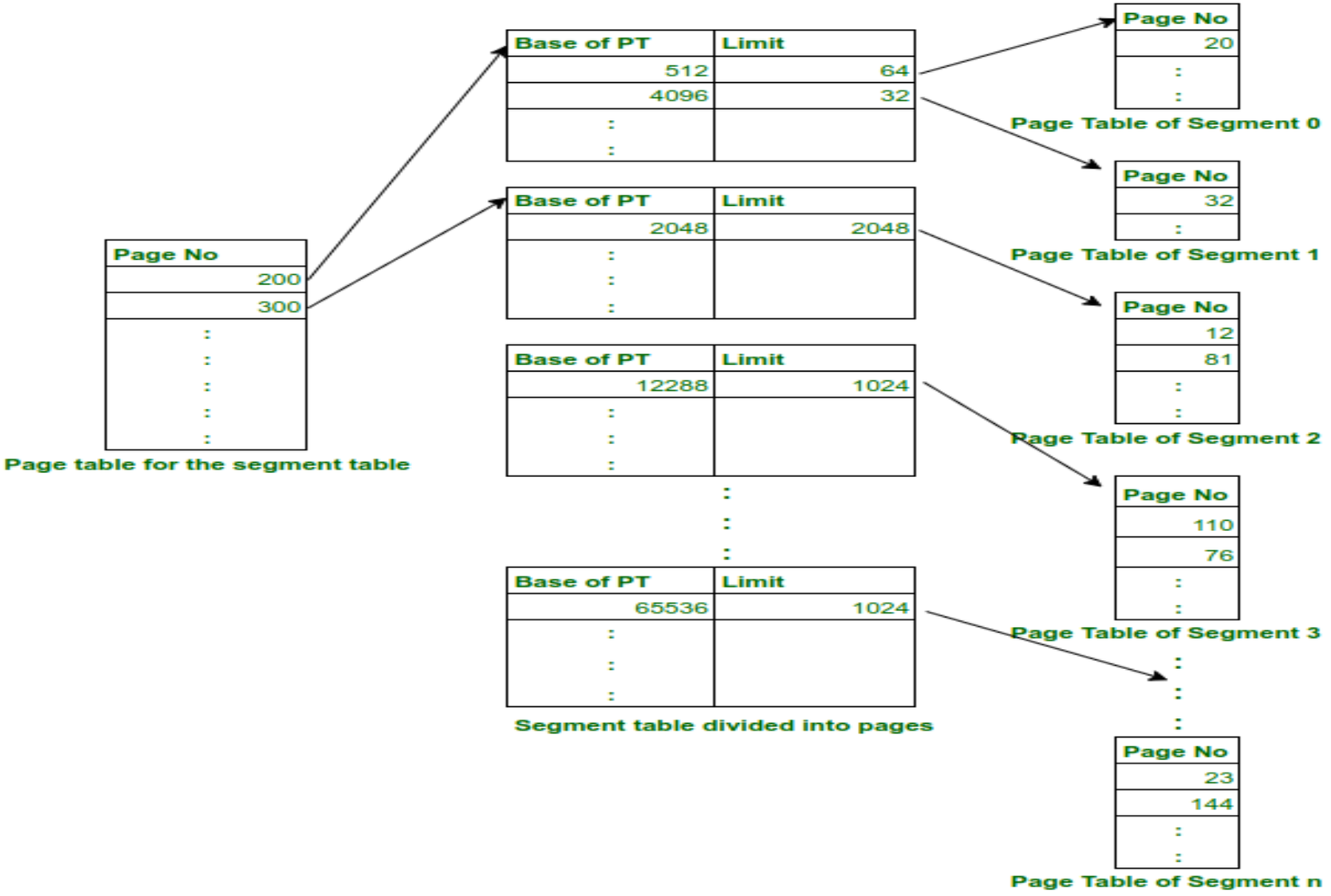
- Paged Segmentation is a memory management technique that divides a process's address space into segments and then divides each segment into pages. This allows for a flexible allocation of memory, where each segment can have a different size, and each page can have a different size within a segment.
- Segmented Paging, on the other hand, is a memory management technique that divides the physical memory into pages, and then maps each logical address used by a process to a physical page. In this approach, segments are used to map virtual memory addresses to physical memory addresses, rather than dividing the virtual memory into pages.

- Both Paged Segmentation and Segmented Paging provide the benefits of paging, such as improved memory utilization, reduced fragmentation, and increased performance. They also provide the benefits of segmentation, such as increased flexibility in memory allocation, improved protection and security, and reduced overhead in memory management.
- However, both techniques can also introduce additional complexity and overhead in the memory management process. The choice between Paged Segmentation and Segmented Paging depends on the specific requirements and constraints of a system, and often requires trade-offs between flexibility, performance, and overhead.

Segmented Paging



Paged Segmentation



Demand paging

Demand paging in os is a technique in which pages are loaded from disk into main memory only when they are needed, i.e., demanded by the program. This technique allows the operating system to save memory space by keeping only those pages in main memory that are currently required by the program.

Demand paging is a technique used in virtual memory systems where pages enter main memory only when requested or needed by the CPU.

In demand paging, the operating system loads only the necessary pages of a program into memory at runtime, instead of loading the entire program into memory at the start.

A page fault occurred when the program needed to access a page that is not currently in memory. The operating system then loads the required pages from the disk into memory and updates the page tables accordingly. This process is transparent to the running program and it continues to run as if the page had always been in memory.

Pure Demand Paging

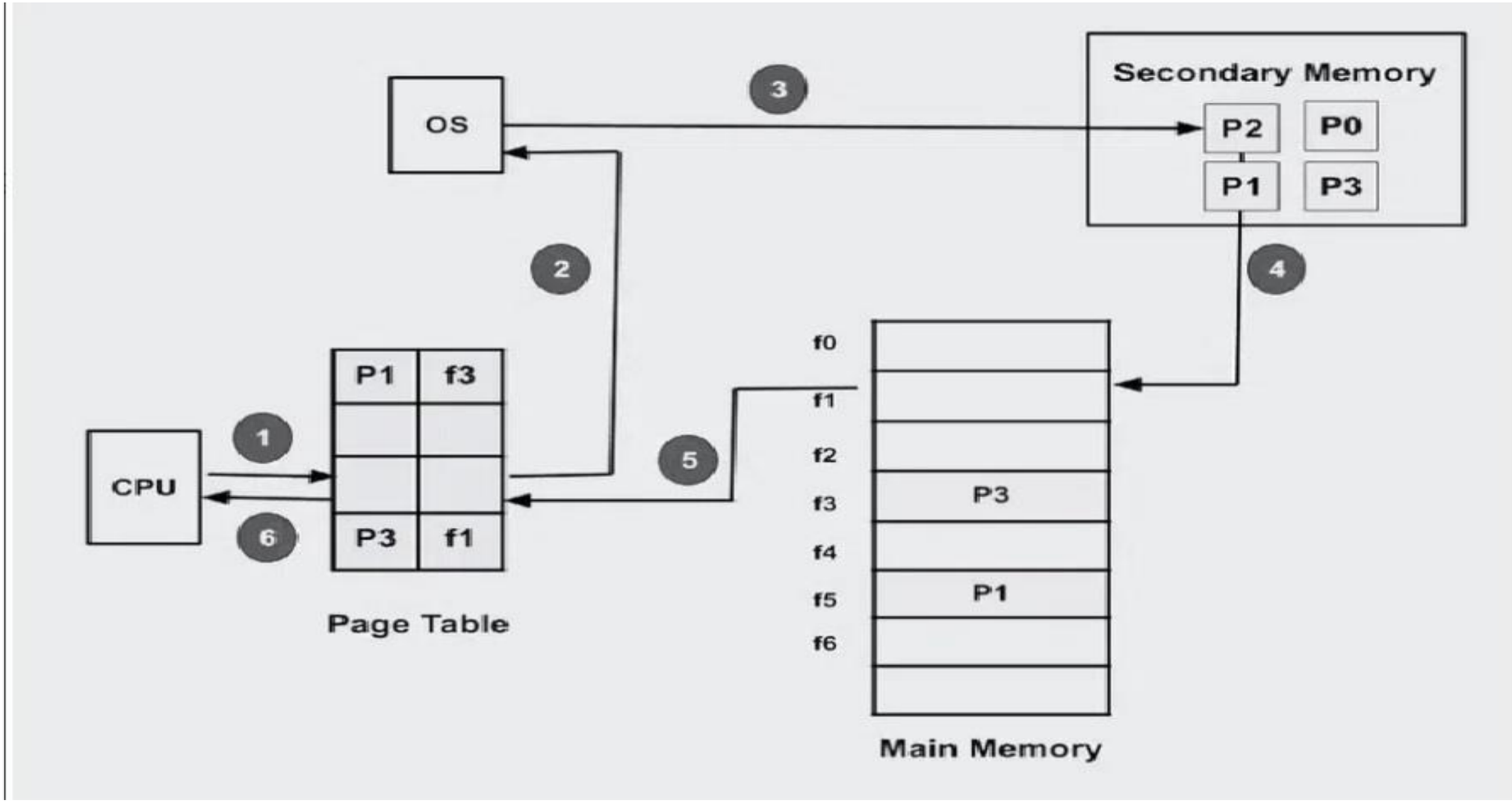
Pure demand paging is a specific implementation of demand paging. The operating system only loads pages into memory when the program needs them. In on-demand paging only, no pages are initially loaded into memory when the program starts, and all pages are initially marked as being on disk.

What is a Page Fault?

- If the referred page is not present in the main memory then there will be a miss and the concept is called Page miss or page fault.
- The CPU has to access the missed page from the secondary memory. If the number of page fault is very high then the effective access time of the system will become very high.

What is Thrashing?

- If the number of page faults is equal to the number of referred pages or the number of page faults are so high so that the CPU remains busy in just reading the pages from the secondary memory then the effective access time will be the time taken by the CPU to read one word from the secondary memory and it will be so high. The concept is called thrashing.


Secondary Memory
P2
P0
P1
P3
OS
3
2
4
f0
f1
f2
f3
f4
f5
f6
P1
f3
P3
f1
CPU
1
5
6
P3
P1
Page Table
Main Memory

Steps that are followed in the working process of the demand paging in the operating system.

- **Program Execution:** When a program starts, the operating system creates a process for the program and allocates a portion of memory to the process.
- **Creating page tables:** The operating system creates page tables for processes, which track which program pages are currently in memory and which are on disk.
- **Page fault handling:** A page fault occurred when the program attempted to access a page that is not currently in memory. The operating system interrupts the program and checks the page tables to see if the required page is on disk.
- **Page Fetch:** If the required page is on disk, the operating system fetches the page from the disk and loads it into memory.
  The page table is then updated to reflect the page's new location in memory.

- **Resuming the program:** Once the required pages have been loaded into memory, the operating system resumes execution of the program where it left off. The program continues to run as if the page had always been in memory.
- **Page replacement:** If there is not enough free memory to hold all the pages a program needs, the operating system may need to replace one or more pages currently in memory with pages currently in memory. on the disk. The page replacement algorithm used by the operating system determines which pages are selected for replacement.
- **Page cleanup:** When a process terminates, the operating system frees the memory allocated to the process and cleans up the corresponding entries in the page tables.

Page Replacement

- Page replacement is a process in an operating system that swaps out a page from the main memory and replaces it with a page from the secondary memory. This happens when a requested page is not in memory and there are no free pages to allocate.
- Page replacement algorithms decide which pages to remove when a new page needs to be loaded into the main memory. The algorithms use limited information about page accesses to guess which pages should be replaced. The goal is to minimize the total number of page misses while balancing the costs of the algorithm.
- The quality of a page replacement algorithm is determined by how much time is spent waiting for page-ins. The less time spent waiting, the better the algorithm.

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.

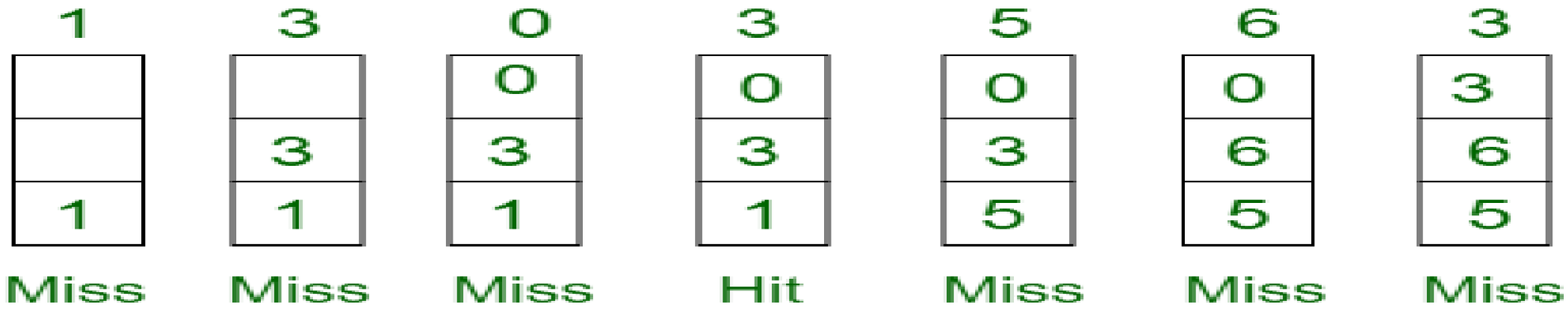
- **1. First In First Out (FIFO):** This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.

- **Example 1:** Consider page reference string 1, 3, 0, 3, 5, 6, 3 with 3 page frames.Find the number of page faults.

Initially, all slots are empty, so when 1, 3, 0 came they are allocated to the empty slots —> **3 Page Faults.**
when 3 comes, it is already in memory so —> **0 Page Faults.** Then 5 comes, it is not available in memory so it replaces the oldest page slot i.e 1. —>**1 Page Fault.** 6 comes, it is also not available in memory so it replaces the oldest page slot i.e 3 —>**1 Page Fault.** Finally, when 3 come it is not available so it replaces 0 **1 page fault.**

Page reference

1, 3, 0, 3, 5, 6, 3



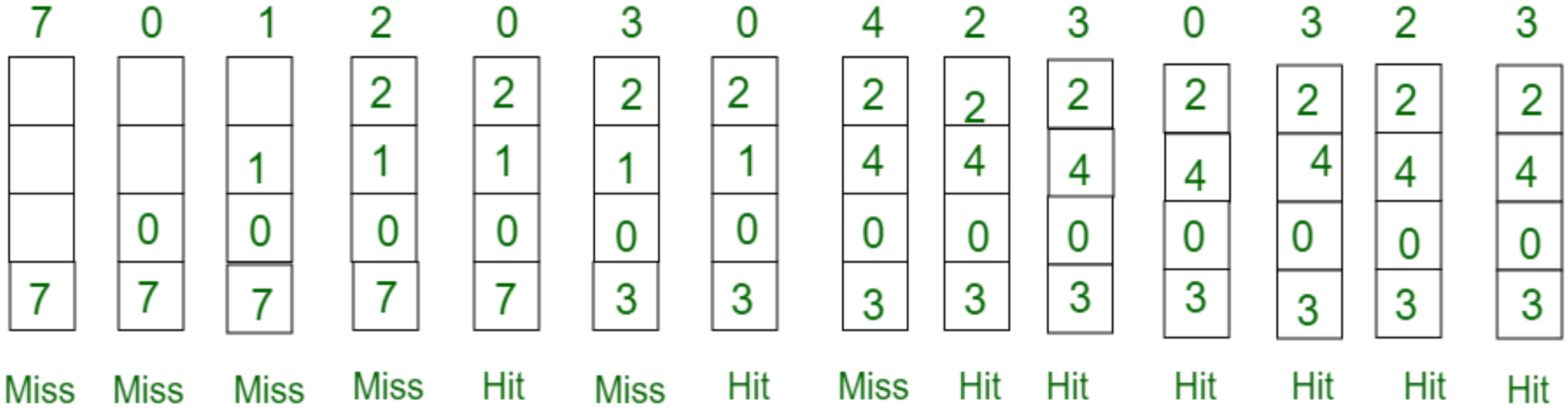
Total Page Fault = 6

- **2. Optimal Page replacement:** In this algorithm, pages are replaced which would not be used for the longest duration of time in the future.
- **Example-2:** Consider the page references 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 3 with 4 page frame. Find number of page fault.

- Initially, all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> **4 Page faults**
  0 is already there so —> **0 Page fault.** when 3 came it will take the place of 7 because it is not used for the longest duration of time in the future.—>**1 Page fault.** 0 is already there so —> **0 Page fault.** 4 will takes place of 1 —> **1 Page Fault.**
- Now for the further page reference string —> **0 Page fault** because they are already available in the memory.
  Optimal page replacement is perfect, but not possible in practice as the operating system cannot know future requests. The use of Optimal Page replacement is to set up a benchmark so that other replacement algorithms can be analyzed against it.

Page reference 7,0,1,2,0,3,0,4,2,3,0,3,2,3

No. of Page frame - 4



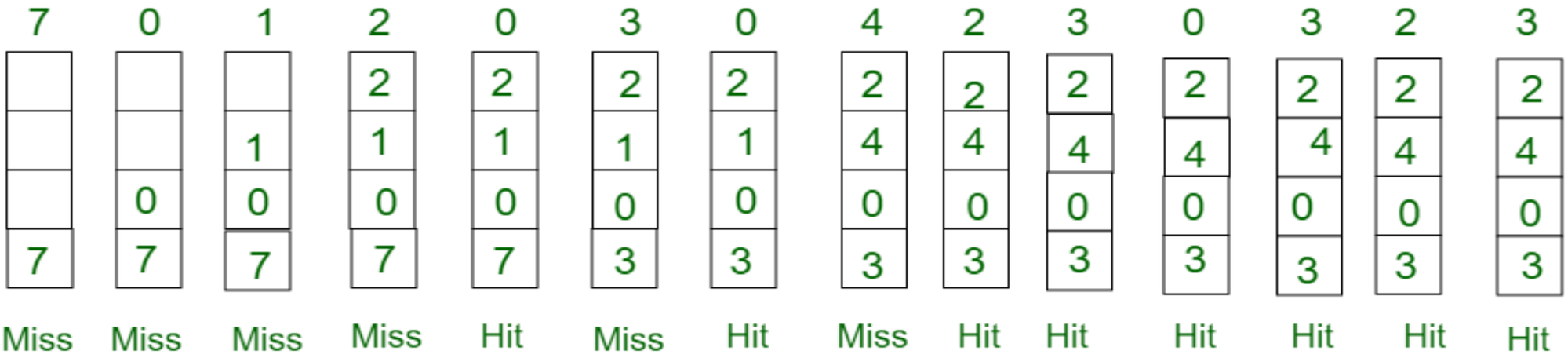
Total Page Fault = 6

- **3. Least Recently Used:** In this algorithm, page will be replaced which is least recently used.
- **Example-3:** Consider the page reference string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 3 with 4 page frames. Find number of page faults.

- Initially, all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> **4 Page faults**
  0 is already their so —> **0 Page fault.** when 3 came it will take the place of 7 because it is least recently used —>**1 Page fault**
  0 is already in memory so —> **0 Page fault**.
  4 will takes place of 1 —> **1 Page Fault**
  Now for the further page reference string —> **0 Page fault** because they are already available in the memory.

Page reference 7,0,1,2,0,3,0,4,2,3,0,3,2,3

No. of Page frame - 4



Total Page Fault = 6

Here LRU has same number of page fault as optimal but it may differ according to question.

Performance of Demand Paging

Paging is a memory management technique used in operating systems to divide a process's virtual memory into fixed-sized pages. The performance of paging depends on various factors, such as:

- Page size: The larger the page size, the less the number of page tables required, which can result in faster memory access times. However, larger page sizes also result in internal fragmentation, where memory is wasted due to the difference between the actual size of a process and the size of a page.
- Page replacement algorithms: The performance of paging depends on the page replacement algorithm used. Common algorithms include FIFO, LRU, ,etc. The choice of algorithm will affect the number of page faults and the time taken to access a page.

- Page table size: The size of the page table used to map virtual addresses to physical addresses affects the speed of memory access. A larger page table results in slower memory access times.
- Page table organization: The organization of the page table can also affect the performance of paging. A hierarchical page table, for example, can reduce the size of the page table and increase the speed of memory access.

Performance of Paging : Evaluating of paging performance is one of the important tasks. Consider the main memory access time is M and the page table is stored in the main memory then the evaluating expression for effective memory access time is as follows.

Effective Memory Access Time (E.M.A.T) = 2M

Features of Performance of Paging :

- Translation lookaside buffer(TLB) is added to improve the performance of paging.
- The TLB is a hardware device implemented using associative registers.
- TLB access time will be very less compared to the main memory access time.
- TLB contains frequently referred page numbers and corresponding frame numbers.

Evaluating Expression for the performance of paging : Consider the TLB access time is ‘c’. And the TLB hit ratio is ‘x’ then the Evaluating Expression for the performance of paging is as follows.

Effective Memory Access Time (E.M.A.T) with TLB

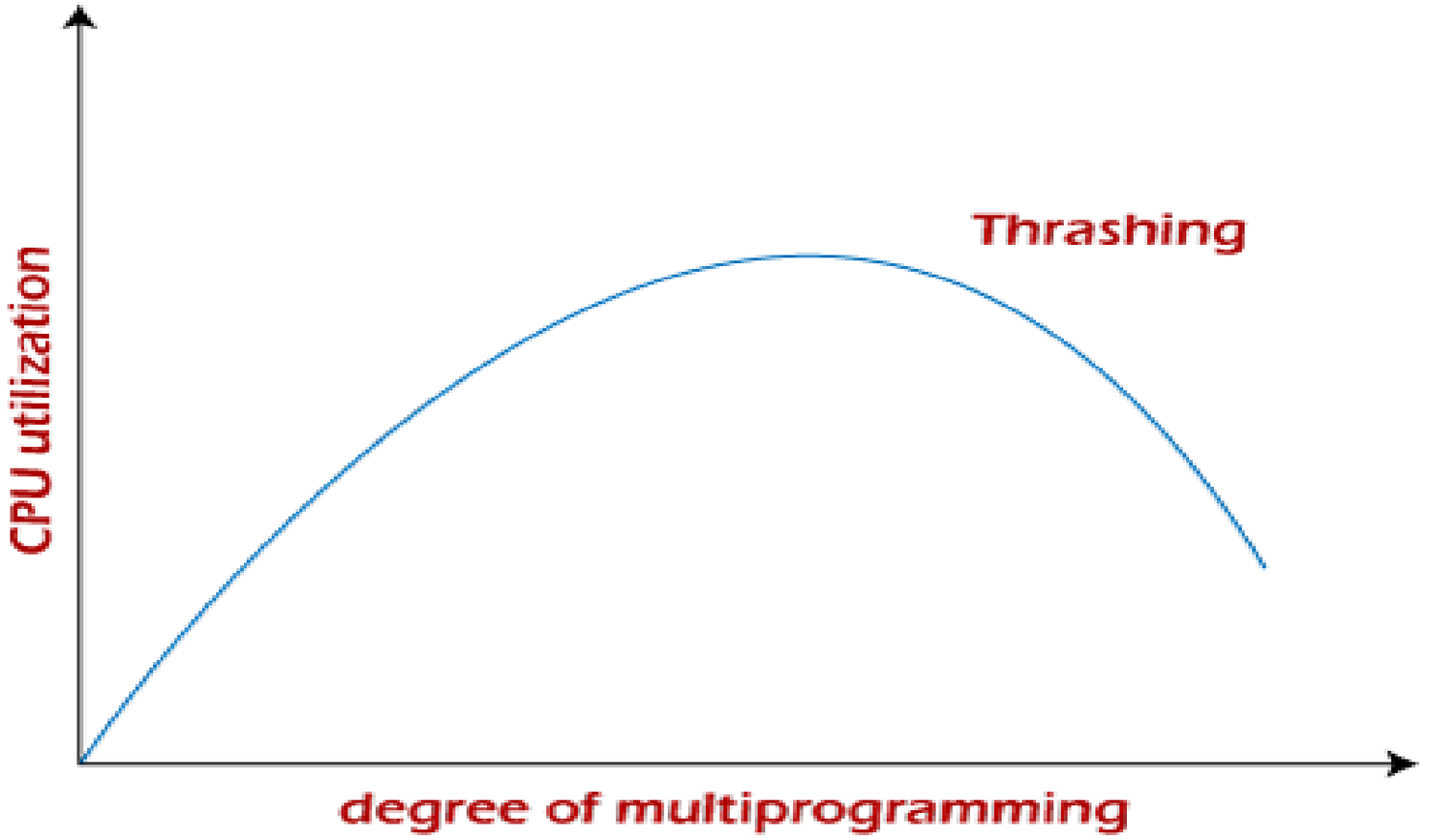
= x(c+m) + (1-x) (c + 2 m)

Thrash

thrash is the poor performance of a virtual memory (or paging) system when the same pages are being loaded repeatedly due to a lack of main memory to keep them in memory.

thrashing occurs when a computer's virtual memory resources are overused, leading to a constant state of paging and page faults, inhibiting most application-level processing. It causes the performance of the computer to degrade or collapse. The situation can continue indefinitely until the user closes some running applications or the active processes free up additional virtual memory resources.

- ***Thrashing*** is when the page fault and swapping happens very frequently at a higher rate, and then the operating system has to spend more time swapping these pages. This state in the operating system is known as thrashing. Because of thrashing, the CPU utilization is going to be reduced or negligible.
- The basic concept involved is that if a process is allocated too few frames, then there will be too many and too frequent page faults. As a result, no valuable work would be done by the CPU, and the CPU utilization would fall drastically.


Thrashing
CPU utilization
degree of multiprogramming

Techniques used to handle the thrashing

1. Working Set

- The set of the pages in the most recent? page reference is known as the working set. If a page is in active use, then it will be in the working set. In case if the page is no longer being used then it will drop from the working set.
- The working set mainly gives the approximation of the locality of the program.
- The accuracy of the working set mainly depends on what is chosen?
- This working set model avoids thrashing while keeping the degree of multiprogramming as high as possible.

2. Page Fault Frequency

- The working-set model is successful and its knowledge can be useful in preparing but it is a very clumpy approach in order to avoid thrashing. There is another technique that is used to avoid thrashing and it is Page Fault Frequency(PFF) and it is a more direct approach.
- The main problem is how to prevent thrashing. As thrashing has a high page fault rate and also we want to control the page fault rate.

- When the Page fault is too high, then we know that the process needs more frames. Conversely, if the page fault-rate is too low then the process may have too many frames.
- We can establish upper and lower bounds on the desired page faults. If the actual page-fault rate exceeds the upper limit then we will allocate the process to another frame. And if the page fault rate falls below the lower limit then we can remove the frame from the process.
- Thus with this, we can directly measure and control the page fault rate in order to prevent thrashing.

Demand Segmentation

The segmentation technology is used in operating systems and the process is divided into many pieces called segments and these segments are of variable size. **In this, segmentation uses a variable partitioning method: one segment is equal to one complete memory block.**

In case of Demand Paging, pages get loaded in the main memory at runtime when the user demands it. In case of Segmentation, **all the sections get loaded at the time of compilation.**

Overlay Concepts

In memory management, overlays refer to a technique used to manage memory efficiently by overlaying a portion of memory with another program or data.

The idea behind overlays is to only load the necessary parts of a program into memory at a given time, freeing up memory for other tasks. The unused portions of the program are kept on disk or other storage, and are loaded into memory as needed. This allows programs to be larger than the available memory, but still run smoothly.

The concept of **overlays** is that whenever a process is running it will not use the complete program at the same time, it will use only some part of it. Then overlays concept says that whatever part you required, you load it and once the part is done, then you just unload it, means just pull it back and get the new part you required and run it.

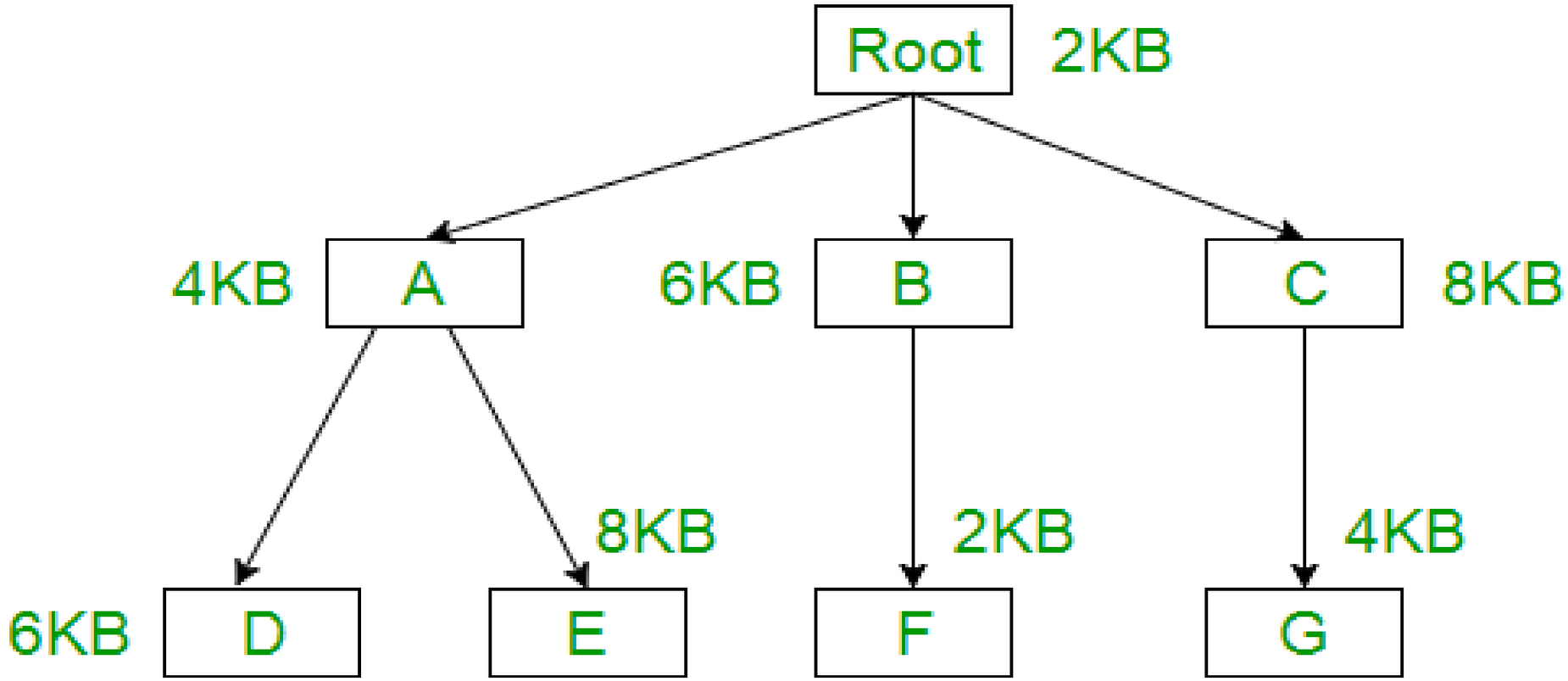
Advantages of using overlays

- Increased memory utilization: Overlays allow multiple programs to share the same physical memory space, increasing memory utilization and reducing the need for additional memory.
- Reduced load time: Only the necessary parts of a program are loaded into memory, reducing load time and increasing performance.
- Improved reliability: Overlays reduce the risk of memory overflow, which can cause crashes or data loss.
- Reduce memory requirement
- Reduce time requirement

Disadvantages of using overlay

- Complexity: Overlays can be complex to implement and manage, especially for large programs.
- Performance overhead: The process of loading and unloading overlays can result in increased CPU and disk usage, which can slow down performance.
- Compatibility issues: Overlays may not work on all hardware and software configurations, making it difficult to ensure compatibility across different systems.
- Overlap map must be specified by programmer
- Programmer must know memory requirement
- Overlapped module must be completely disjoint
- Programming design of overlays structure is complex and not possible in all cases

Example: The overlay tree for a program is as shown below



- What will be the size of the partition (in physical memory) required to load (and
  run) this program?
  (a) 12 KB (b) 14 KB (c) 10 KB (d) 8 KB

Using the overlay concept we need not actually have the entire program inside the main memory.Only we need to have the part which are required at that instance of time, either we need Root-A-D or Root-A-E or Root-B-F or Root-C-G part.

Root+A+D = 2KB + 4KB + 6KB = 12KB

Root+A+E = 2KB + 4KB + 8KB = 14KB

Root+B+F = 2KB + 6KB + 2KB = 10KB

Root+C+G = 2KB + 8KB + 4KB = 14KB

So if we have 14KB size of partition then we can run any of them.

Answer - 14KB

Assignment 2

1. Explain Process control block in brief.
2. Explain Starvation . What are the possible solutions to solve this problem.
3. Explain Memory management strategies.
4. Explain the Process Scheduling. Draw the process state diagram.
5. Assume 3 frames available. The page reference string is

5,1,2,1,3,2,1,4,5,2,3,1,6,5,4,3,2,1.

Calculate no. of page faults in case of FlIFO and LRU page replacement algorithms in case of pure demand paging.


Thank you!