CHAPTER 3

SYNCHRONIZATION & DEADLOCKS

Chapter Outline

3.0 INTRODUCTION

Process synchronization needs to be implemented to prevent data inconsistency among processes, process deadlocks, and prevent race conditions, which are ***when two or more operations are executed at the same time, not scheduled in the proper sequence and not exited in the critical section correctly.***

3.1 INTER PROCESS COMMUNICATION

Interprocess communication is the mechanism provided by the operating system that allows processes to communicate with each other.

This communication could involve a process letting another process know that some event has occurred or the transferring of data from one process to another.

A diagram that illustrates interprocess communication is as follows :

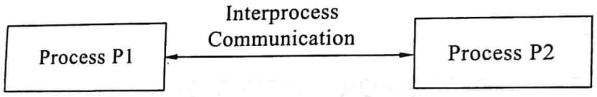

FIG 3.1 :

Synchronization is a necessary part of interprocess communication. It is either provided by the interprocess control mechanism or handled by the communicating processes.

Some of the methods to provide synchronization are as follows :

- **Semaphore :** A semaphore is a variable that controls the access to a common resource by multiple processes. The two types of semaphores are binary semaphores and counting semaphores.
- **Mutual Exclusion :** Mutual exclusion requires that only one process thread can enter the critical section at a time. This is useful for synchronization and also prevents race conditions.
- **Barrier :** A barrier does not allow individual processes to proceed until all the processes reach it. Many parallel languages and collective routines impose barriers.
- **Spinlock :** This is a type of lock. The processes trying to acquire this lock wait in a loop while checking if the lock is available or not. This is known as busy waiting because the process is not doing any useful operation even though it is active.

3.1.1 APPROACHES TO INTERPROCESS COMMUNICATION

The different approaches to implement interprocess communication are given as follows :

- **Pipe :** A pipe is a data channel that is unidirectional. Two pipes can be used to create a two-way data channel between two processes. This uses standard input and output methods. Pipes are used in all POSIX systems as well as Windows operating systems.
- **Socket :** The socket is the endpoint for sending or receiving data in a network. This is true for data sent between processes on the same computer or data sent between different computers on the same network. Most of the operating systems use sockets for interprocess communication.
- **File :** A file is a data record that may be stored on a disk or acquired on demand by a file server. Multiple processes can access a file as required. All operating systems use files for data storage.
- **Signal :** Signals are useful in interprocess communication in a limited way. They are system messages that are sent from one process to another. Normally, signals are not used to transfer data but are used for remote commands between processes.
- **Shared Memory :** Shared memory is the memory that can be simultaneously accessed by multiple processes. This is done so that the processes can communicate with each other. All POSIX systems, as well as Windows operating systems use shared memory.
- **Message Queue :** Multiple processes can read and write data to the message queue without being connected to each other. Messages are stored in the queue until their recipient retrieves them. Message queues are quite useful for interprocess communication and are used by most operating systems.

A diagram that demonstrates message queue and shared memory methods of interprocess communication is as follows :

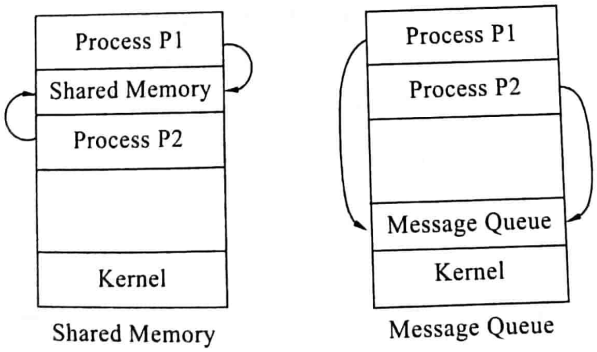

FIG 3.2 : Approaches to Interprocess Communication

3.2 SEMAPHORES

Semaphores are integer variables that are used to solve the critical section problem by using two atomic operations, wait and signal that are used for process synchronization.

The definitions of wait and signal are as follows :

- **Wait :** The wait operation decrements the value of its argument S, if it is positive. If S is negative or zero, then no operation is performed.

```
wait(S)
{
   while (S<=0);
   S++;
}
```

- **Signal :** The signal operation increments the value of its argument S.

```
signal(S)
{
   S++;
}
```

3.2.1 TYPES OF SEMAPHORES

There are two main types of semaphores i.e., counting semaphores and binary semaphores.

Details about these are given as follows :

- **Counting Semaphores :** These are integer value semaphores and have an unrestricted value domain. These semaphores are used to coordinate the resource access, where the semaphore count is the number of available resources. If the resources are added, semaphore count automatically incremented and if the resources are removed, the count is decremented.
- **Binary Semaphores :** The binary semaphores are like counting semaphores but their value is restricted to 0 and 1. The wait operation only works when the semaphore is 1 and the signal operation succeeds when semaphore is 0. It is sometimes easier to implement binary semaphores than counting semaphores.

Advantages of Semaphores :

- Semaphores allow only one process into the critical section. They follow the mutual exclusion principle strictly and are much more efficient than some other methods of synchronization.
- There is no resource wastage because of busy waiting in semaphores as processor time is not wasted unnecessarily to check if a condition is fulfilled to allow a process to access the critical section.
- Semaphores are implemented in the machine independent code of the microkernel. So they are machine independent.

Disadvantages Of Semaphores :

- Semaphores are complicated so the wait and signal operations must be implemented in the correct order to prevent deadlocks.
- Semaphores are impractical for last scale use as their use leads to loss of modularity. This happens because the wait and signal operations prevent the creation of a structured layout for the system.
- Semaphores may lead to a priority inversion where low priority processes may access the critical section first and high priority processes later.

3.3 MONITORS

- Monitors are used for process synchronization. With the help of programming languages, we can use a monitor to achieve mutual exclusion among the processes.
- **Example of monitors: *Java Synchronized methods such as Java offers notify() and wait() constructs.***
- In other words, monitors are defined as the construct of programming language, which helps in controlling shared data access.
- The Monitor is a module or package which encapsulates shared data structure, procedures, and the synchronization between the concurrent procedure invocations.

3.3.1 CHARACTERISTICS OF MONITORS

1. Inside the monitors, we can only execute one process at a time.
2. Monitors are the group of procedures, and condition variables that are merged together in a special type of module.
3. If the process is running outside the monitor, then it cannot access the monitor's internal variable. But a process can call the procedures of the monitor.

4. Monitors offer high-level of synchronization
5. Monitors were derived to simplify the complexity of synchronization problems.
6. There is only one process that can be active at a time inside the monitor.

3.3.2 COMPONENTS OF MONITOR

There are four main components of the monitor are :

1. Initialization.
2. Private Data.
3. Monitor Procedure.
4. Monitor Entry Queue

1. **Initialization :** Initialization comprises the code, and when the monitors are created, we use this code exactly once.
2. **Private Data :** Private data is another component of the monitor. It comprises all the private data, and the private data contains private procedures that can only be used within the monitor. So, outside the monitor, private data is not visible.
3. **Monitor Procedure :** Monitors Procedures are those procedures that can be called from outside the monitor.
4. **Monitor Entry Queue :** Monitor entry queue is another essential component of the monitor that includes all the threads, which are called procedures.

Syntax of Monitor :

```
Monitor Demo // Name of the Monitor
{
variables;
condition variables;
procedurep1 {.....}
procedurep2 {.....}
}
```

3.3.3 CONDITION VARIABLES

There are two types of operations that we can perform on the condition variables of the monitor :

1. Wait 2. Signal

```
Suppose there are two condition variables
condition a, b // Declaring variable
```

1. **Wait Operation :**

wait() : The process that performs wait operation on the condition variables are suspended and locate the suspended process in a block queue of that condition variable.

2. **SIGNAL Operation :**

signal() : If a signal operation is performed by the process on the condition variable, then a chance is provided to one of the blocked processes.

Advantages of Monitor :

- It makes the parallel programming easy, and if monitors are used, then there is less error-prone as compared to the semaphore.

Dis Advantages of Monitor :

- Monitors have to be implemented as part of the programming language.
- The compiler must generate code for them. This gives the compiler the additional burden of having to know what operating system facilities are available to control access to critical sections in concurrent processes.
- Some languages that do support monitors are Java,C#,VisualBasic,Ada and concurrent Euclid.

3.3.3 DIFFERENCE BETWEEN MONITORS AND SEMAPHORE

Monitors	Semaphore
We can use condition variables only in the monitors.	In semaphore, we can use condition variables anywhere in the program, but we cannot use conditions variables in a semaphore.
In monitors, wait always block the caller.	In semaphore, wait does not always block the caller.
The monitors are comprised of the shared variables and the procedures which operate the shared variable.	The semaphore S value means the number of shared resources that are present in the system.
Condition variables are present in the monitor.	Condition variables are not present in the semaphore.

3.4 DEAD LOCKS

A deadlock happens in operating system when two or more processes need some resource to complete their execution that is held by the other process.

In Fig. 3.3, the process 1 has resource 1 and needs to acquire resource 2. Similarly process 2 has resource 2 and needs to acquire resource 1. Process 1 and process 2 are in deadlock as each of them needs the other's resource to complete their execution but neither of them is willing to relinquish their resources.

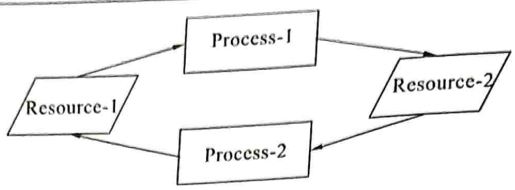

FIG 3.3 : Deadlock in Operating System

3.5 NECESSARY CONDITIONS FOR ARISING DEADLOCKS

- **Mutual Exclusion :** There should be a resource that can only be held by one proce at a time. In the diagram below, there is a single instance of Resource 1 and it is held b Process 1 only.

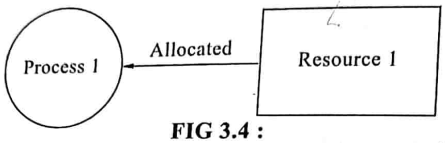

FIG 3.4 :

- **Hold and Wait :** A process can hold multiple resources and still request more resource from other processes which are holding them. In the diagram given below, Process-2 holds Resource-2 and Resource-3 and is requesting the Resource-1 which is held b Process-1.

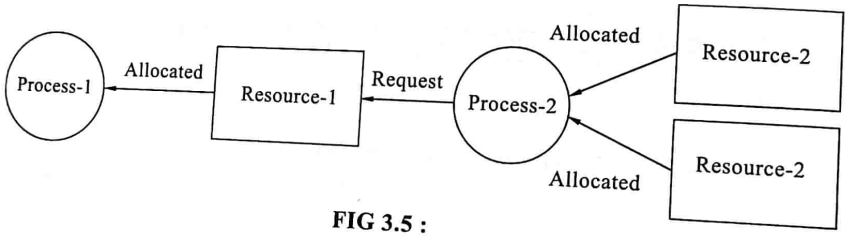

FIG 3.5 :

- **No Preemption :** A resource cannot be preempted from a process by force. A process can only release a resource voluntarily. In the diagram below, Process-2 cannot preempt Resource-1 from Process-1. It will only be released when Process-1 relinquishes it voluntarily after its execution is complete.

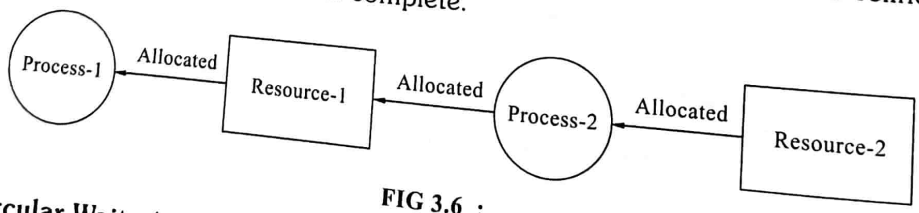

FIG 3.6 :

- **Circular Wait :** A process is waiting for the resource held by the second process, which is waiting for the resource held by the third process and so on, till the last process is waiting for a resource held by the first process. This forms a circular chain.

For example : Process-1 is allocated Resource-2 and it is requesting Resource-1. Similarly, Process 2 is allocated Resource-1 and it is requesting Resource-2. This forms a circular wait loop.

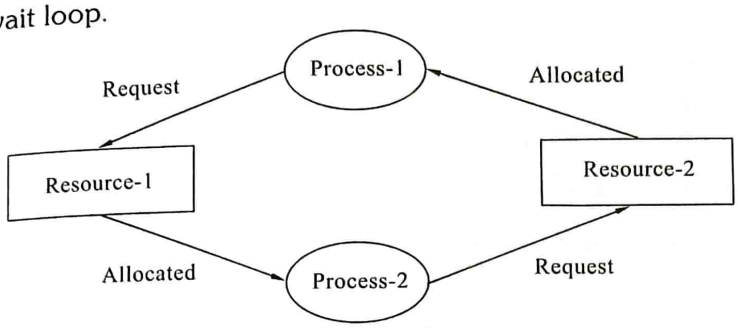

FIG 3.7 :

3.6 METHODS OF HANDLING DEADLOCKS IN OPERATING SYSTEM

The first two methods are used to ensure the system never enters a deadlock.

3.6.1 DEADLOCK PREVENTION

This is done by restraining the ways a request can be made. Since deadlock occurs when all the above four conditions are met, we try to prevent any one of them, thus preventing a deadlock.

Eliminate Mutual Exclusion : It is not possible to dis-satisfy the mutual exclusion because some resources, such as the tape drive and printer, are inherently non-shareable.

Eliminate Hold and Wait :

1. Allocate all required resources to the process before the start of its execution, this way hold and wait condition is eliminated but it will lead to low device utilization. for example, if a process requires printer at a later time and we have allocated printer before the start of its execution printer will remain blocked till it has completed its execution.
2. The process will make a new request for resources after releasing the current set of resources. This solution may lead to starvation.

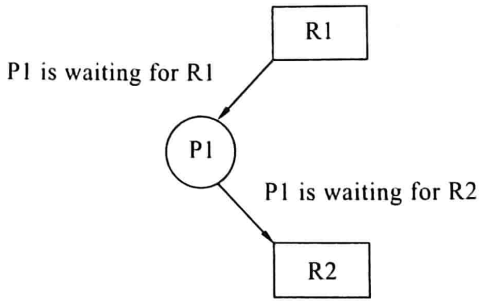

FIG 3.8 : Hold and Wait

Eliminate No Preemption : Preempt resources from the process when resources required by other high priority processes.

Eliminate Circular Wait : Each resource will be assigned with a numerical number. A process can request the resources increasing/decreasing. order of numbering For Example, if P1 process is allocated R5 resources, now next time if P1 ask for R4 R3 lesser than R5 such request will not be granted, only request for resources more than R5 will be granted.

3.7 DEADLOCK AVOIDANCE

Deadlock avoidance can be done with Banker's Algorithm.

Banker's Algoritham : Bankers's Algorithm is resource allocation and deadlock avoidance algorithm which test all the request made by processes for resources, it checks for the safe state, if after granting request system remains in the safe state it allows the request and if there is no safe state it doesn't allow the request made by the process.

Inputs to Banker's Algorithm :

1. Max need of resources by each process.
2. Currently, allocated resources by each process.
3. Max free available resources in the system.

The request will only be granted under the below condition :

1. If the request made by the process is less than equal to max need to that process.
2. If the request made by the process is less than equal to the freely available resource in the system.

Example :

Total resources in system :

A B C D

6 5 7 6

Available system resources are:

A B C D

3 1 1 2

Processes (currently allocated resources) :

A B C D

P1 1 2 2 1

P2 1 0 3 3

P3 1 2 1 0

Processes

A B C

P1 3 3 2

P2 1 2 3

P3 1 3 5

Need = m

Processe

A B C

P1 2 1 0

P2 0 2 0

P3 0 1 4

Note : Deadlo

3.7.1 DEAD

1. **If Resources** an algorithm t a cycle in the

2. In the above cycle R1 →

3. **If there are M** not sufficient be in deadloc

P2 1 0 3 3

P3 1 2 1 0

Processes (maximum resources) :

A B C D

P1 3 3 2 2

P2 1 2 3 4

P3 1 3 5 0

Need = maximum resources - currently allocated resources.

Processes (need resources) :

A B C D

P1 2 1 0 1

P2 0 2 0 1

P3 0 1 4 0

Note : Deadlock prevention is more strict than Deadlock Avoidance.

3.7.1 DEADLOCK DETECTION

1. **If Resources have a Single Instance :**In this case for Deadlock detection, we can run an algorithm to check for the cycle in the Resource Allocation Graph. The presence of a cycle in the graph is a sufficient condition for deadlock.

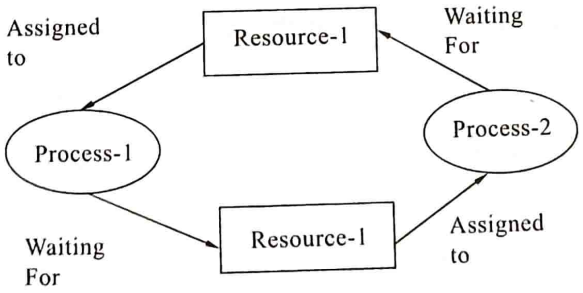

FIG 3.9 :

2. In the above diagram, resource 1 and resource 2 have single instances. There is a cycle R1 → P1 → R2 → P2. So, Deadlock is Confirmed.
3. **If there are Multiple Instances of Resources :** Detection of the cycle is necessary but not sufficient condition for deadlock detection, in this case, the system may or may not be in deadlock varies according to different situations.

3.7.2 DEADLOCK IGNORANCE

In the method, the system assumes that deadlock never occurs. Since the problem of deadlock situation is not frequent, some systems simply ignore it. Operating systems such as UNIX and Windows follow this approach. However, if a deadlock occurs we can reboot our system and the deadlock is resolved automatically.

Note: The above approach is an example of Ostrich Algorithm. It is a strategy of ignoring potential problems on the basis that they are extremely rare.

3.8 DEADLOCK RECOVERY

A traditional operating system such as Windows doesn't deal with deadlock recovery as it is a time and space-consuming process. Real-time operating systems use Deadlock recovery.

1. **Killing the Process :** Killing all the processes involved in the deadlock. Killing process one by one. After killing each process check for deadlock again keep repeating the process till the system recovers from deadlock. Killing all the processes one by one helps a system to break circular wait condition.
2. **Resource Preemption :** Resources are preempted from the processes involved in the deadlock, preempted resources are allocated to other processes so that there is a possibility of recovering the system from deadlock. In this case, the system goes into starvation.

Advantage of Deadlock Method :

- No preemption is needed for deadlocks.
- It is a good method if the state of the resource can be saved and restored easily.
- It is good for activities that perform a single burst of activity.
- It does not need run-time computations because the problem is solved in system design.

Disadvantages of Dadlock Method :

- The processes must know the maximum resource of each type required to execute it.
- Preemptions are frequently encountered.
- It delays the process initiation.
- There are inherent pre-emption losses.
- It does not support incremental request of resources.

1. Which
(a)
(c)
2. When the e calle
(a)
(c)
3. If a exec
(a)
(c)
4. Wha
(a)
(b)
(c)
(d)
5. Wh
(a
(c
6. Se
(a
(
7. W
(
8. A

OBJECTIVE TYPE QUESTIONS

1. **Which process can be affected by other processes executing in the system?**
   (a) cooperating process (b) child process
   (c) parent process (d) init process

2. **When several processes access the same data concurrently and the outcome of the execution depends on the particular order in which the access takes place is called**
   (a) Dynamic condition (b) Race condition
   (c) Essential condition (d) Critical condition

3. **If a process is executing in its critical section, then no other processes can be executing in their critical section. What is this condition called?**
   (a) Mutual exclusion (b) Critical exclusion
   (c) Synchronous exclusion (d) Asynchronous exclusion

4. **What is Interprocess communication?**
   (a) Allows processes to communicate and synchronize their actions when using the same address space
   (b) Allows processes to communicate and synchronize their actions
   (c) Allows the processes to only synchronize their actions without communication
   (d) None of the mentioned

5. **Which of the following two operations are provided by the IPC facility?**
   (a) Write and delete message (b) Delete and receive message
   (c) Send and delete message (d) Receive and send message

6. **Semaphore is a/an _____ to solve the critical section problem.**
   (a) Hardware for a system (b) Special program for a system
   (c) Integer variable (d) None of the mentioned

7. **What are the two atomic operations permissible on semaphores?**
   (a) Wait (b) Stop
   (c) Hold (d) None of the above

8. **A monitor is a type of _____**
   (a) Semaphore (b) Low level synchronization construct
   (c) High level synchronization construct (d) None of the above

9. **A monitor is characterized by _____**
   (a) A set of programmer defined operators
   (b) An identifier
   (c) The number of variables in it
   (d) All of the above

10. **A procedure defined within a _____ can access only those variables declared locally within the _____ and its formal parameters.**
   (a) Process, semaphore
   (b) Process, monitor
   (c) Semaphore, semaphore
   (d) Monitor, monitor

11. **What is a reusable resource?**
   (a) That can be used by one process at a time and is not depleted by that use
   (b) That can be used by more than one process at a time
   (c) That can be shared between various threads
   (d) None of the above

12. **Which of the following condition is required for a deadlock to be possible?**
   (a) Mutual exclusion
   (b) A process may hold allocated resources while awaiting assignment of other resources
   (c) No resource can be forcibly removed from a process holding it
   (d) All of the above

13. **The circular wait condition can be prevented by _____**
   (a) Defining a linear ordering of resource types
   (b) Using thread
   (c) Using pipes
   (d) All of the above

14. **Which one of the following is the deadlock avoidance algorithm?**
   (a) Banker's algorithm
   (b) Round-robin algorithm
   (c) Elevator algorithm
   (d) Karn's algorithm

ANSWERS

1. (a)	2. (b)	3. (a)	4. (b)	5. (d)	6. (c)	7. (a)	8. (c)	9. (a)	10. (d)
11. (a)	12. (d)	13. (a)	14.(a)						

REVIEW QUESTIONS

Part-A

1. Define Deadlock.
2. List three options for breaking an existing deadlock.
3. Define semaphore.
4. Define Monitor.
5. Define the following terms:
   (a) Race Condition
   (b) Through Put
   (c) Critical Section
   (d) Mutual Exclusion

Part-B

1. List three overall strategies for arising deadlocks. **(Mar/Apr. 2013)**
2. List the necessary conditions for arising deadlock. **(Mar/Apr. 2014)**
3. List two ways that we can break the second condition to prevent deadlock. **(Apr/May. 2012)**
4. Explain deadlock recovery process.
5. Explain about semaphore & its operations. **(Mar/Apr. 2013)**
6. Explain about Inter Process Communication. **(Apr. 2007)**
7. Explain how deadlocks can be prevented.
8. What three issues must be considered in case of preemption? **(Mar/Apr. 2014)**
9. Explain Preemtable and non preemtable resource with example.

CHAPTER 4

MEMORY MANAGEMENT

Chapter Outline

4.0 INTRODUCTION

Memory is the important part of the computer that is used to store the data. Its management is critical to the computer system because the amount of main memory available in a computer system is very limited.

At any time, many processes are competing for it. Moreover, to increase performance, several processes are executed simultaneously.

For this, we must keep several processes in the main memory, so it is even more important to manage them effectively.

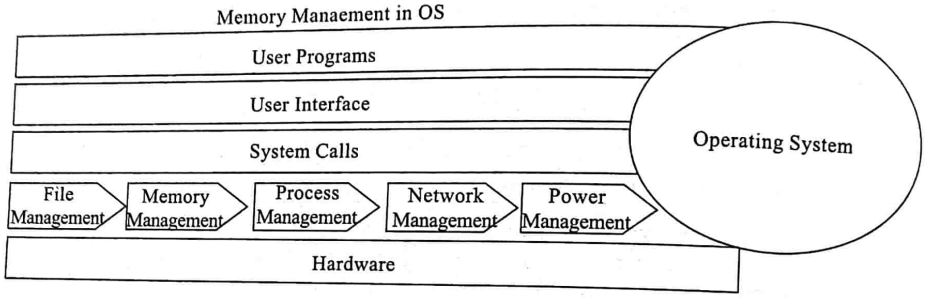

FIG 4.1 :

4.1 ADDRESS BINDING, DYNAMIC LOADING, DYNAMIC LINKING

4.1.1 ADDRESS BINDING

The Association of program instruction and data to the actual physical memory locations is called the ***Address Binding***.

Let's consider the following example given below for better understanding.

Consider a program P1 has the set of instruction such that I1, I2, I3, I4, and program counter value is 10, 20, 30, 40 respectively.

Program P1 :

I1 → 10

I2 → 20

I3 → 30

I4 → 40

Program Counter = 10, 20, 30, 40

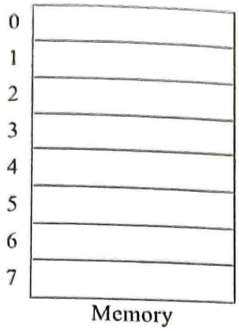

FIG 4.2 :

Types of Address Binding : Address Binding divided into three types as follows.

1. Compile-time Address Binding
2. Load time Address Binding
3. Execution time Address Binding

1. **Compile-time Address Binding :**
   - If the compiler is responsible for performing address binding then it is called compile-time address binding.
   - It will be done before loading the program into memory.
   - The compiler requires interacts with an OS memory manager to perform compile-time address binding.
2. **Load Time Address Binding :**
   - It will be done after loading the program into memory.
   - This type of address binding will be done by the OS memory manager i.e loader.
3. **Execution Time or Dynamic Addressebinding :**
   - It will be postponed even after loading the program into memory.
   - The program will be kept on changing the locations in memory until the time of program execution.
   - The dynamic type of address binding done by the processor at the time of program execution.

Note : The majority of the Operating System practically implement dynamic loading, dynamic linking, dynamic address binding. **For example :** Windows, Linux, Unix all popular OS.

4.1.2 DYNAMIC LOADING

- The complete program and all process data must be in ***physical memory*** to execute a process. As a result, the process size is restricted by the amount of physical memory available.
- Dynamic loading is utilized to ensure ***optimal memory consumption***.
- In dynamic loading, a routine is not loaded until it is invoked.
- All of the routines are stored on disk in a ***reloadable load format***.
- The main advantages of dynamic loading are that new routines are never loaded. This loading is useful when a huge amount of code is required to handle it efficiently.

4.1.3 DYNAMIC LINKING

- Every dynamically linked program contains a small, statically linked function that is called when the program starts.
- This static function only maps the link library into memory and runs the code that the function contains.
- The link library determines what are all the dynamic libraries which the program requires along with the names of the variables and functions needed from those libraries by reading the information contained in sections of the library.
- After which it maps the libraries into the middle of virtual memory and resolves the references to the symbols contained in those libraries.
- We don't know where in the memory these shared libraries are actually mapped: They are compiled into position-independent code (PIC), that can run at any address in memory.

4.1.4 ADVANTAGE

Memory requirements of the program are reduced. A DLL is loaded into memory only once, whereas more than one application may use a single DLL at the moment, thus saving memory space.

Application support and maintenance costs are also lowered.

4.2 OVERLAYS

- The main problem in Fixed partitioning is the size of a process has to be limited by the maximum size of the partition, which means a process can never be span over another.

- In order to solve this as Overlays.
- The concept of **overl** program at the same whatever part you re means just pull it ba **Formally,** "The pro internal memory, re Sometimes it happe program will be eve
- So overlay is a tech memory by keeping Divide the program memory at the sam

Advantages :

- Reduce mem
- Reduce time

Disadvantages :

- Overlap map
- Programmer
- Overlaped m
- Programmm

4.3 SWAPPIN

Swapping is mec memory (or mov other processes. secondary storag

Though perform multiple and big a ***technique for***

The total time ta process to a sec the time the pro

- In order to solve this problem, earlier people have used some solution which is called as Overlays.
- 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 an 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.

  Formally, "The process of **transferring a block** of program code or other data into internal memory, replacing what is already stored".

  Sometimes it happens that compare to the size of the biggest partition, the size of the program will be even more, then, in that case, you should go with overlays.
- So overlay is a technique to run a program that is bigger than the size of the physical memory by keeping only those instructions and data that are needed at any given time. Divide the program into modules in such a way that not all modules need to be in the memory at the same time.

Advantages :

- Reduce memory requirement.
- Reduce time requirement.

Disadvantages :

- Overlap map must be specified by programmer.
- Programmer must know memory requirement.
- Overlaped module must be completely disjoint.
- Programmming design of overlays structure is complex and not possible in all cases.

4.3 SWAPPING

Swapping is mechanisms in which a process can be swapped temporarily out of main memory (or move) to secondary storage (disk) and make that memory available to other processes. At some later time, the system swaps back the process from the secondary storage to main memory.

Though performance is usually affected by swapping process but it helps in running multiple and big processes in parallel and that's the reason swapping is also known as a ***technique for memory compaction***.

The total time taken by swapping process includes the time it takes to move the entire process to a secondary disk and then to copy the process back to memory, as well as the time the process takes to regain main memory.

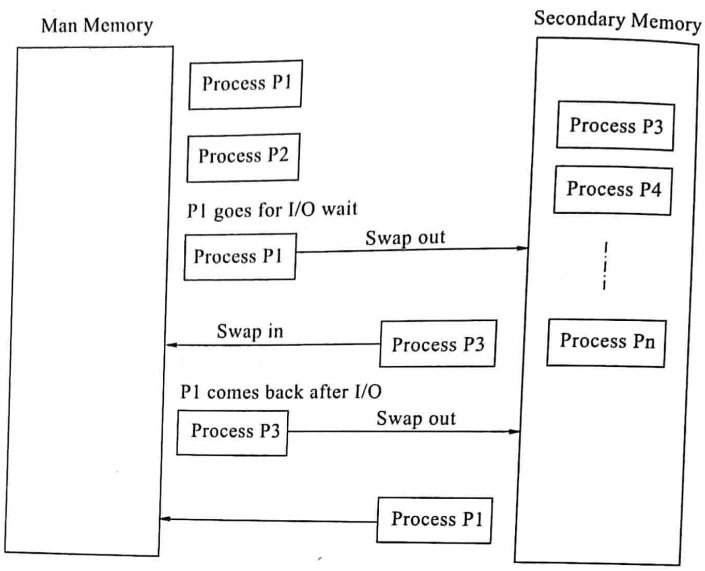

FIG 4.3 :

Let us assume that the user process is of size 2048 KB and on a standard hard disk where swapping will take place has a data transfer rate around 1 MB per second. The actual transfer of the 1000K process to or from memory will take

2048 KB / 1024 KB per second

= 2 seconds

= 2000 milliseconds

Now considering in and out time, it will take complete 4000 milliseconds plus other overhead where the process competes to regain main memory.

Advantages of Swapping :

1. It helps the CPU to manage multiple processes within a single main memory.
2. It helps to create and use virtual memory.
3. Swapping allows the CPU to perform multiple tasks simultaneously. Therefore, processes do not have to wait very long before they are executed.
4. It improves the main memory utilization.

Disadvantages of Swapping :

1. If the computer system loses power, the user may lose all information related to the program in case of substantial swapping activity.

2. If the swapping algorithm is not good, the composite method can increase the number of Page Fault and decrease the overall processing performance.

4.4 SINGLE PARTITION ALLOCATION

- In this type of allocation, relocation-register scheme is used to protect user processes from each other, and from changing operating-system code and data.
- Relocation register contains value of smallest physical address whereas limit register contains range of logical addresses.
- Each logical address must be less than the limit register.

Single Partition Allocaton

- Set aside some memory for the OS
- User Program gets the rest
- Use MMU to translate addresses; simple addition

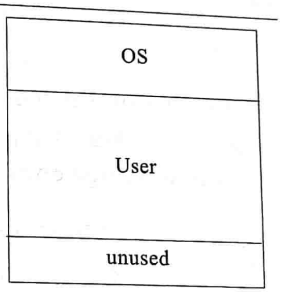

FIG 4.4 :

Advantage : Its simpler.

Disadvantages :

- Poor utilization of memory.
- Multi programming is not possible.

4.5 MULTIPLE-PARTITION ALLOCATION

- In this type of allocation, main memory is divided into a number of fixed-sized partitions where each partition should contain only one process.
- When a partition is free, a process is selected from the input queue and is loaded into the free partition.
- When the process terminates, the partition becomes available for another process.
- 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.

Multiple Partition Allocaton

- OS still gets its own partition
- Place multiple processes in the rest of memory
- Translation is still simple but now processes dependent

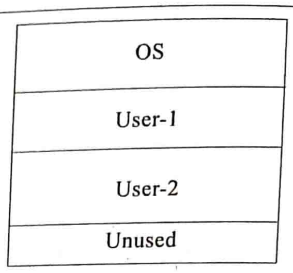

FIG 4.5 :

There are different Placement Algorithm :

1. First Fit
2. Best Fit
3. Worst Fit
4. Next Fit

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.

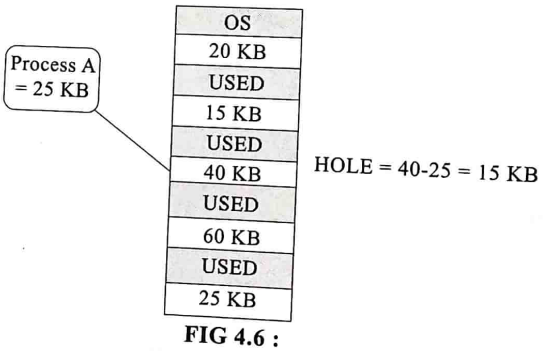

FIG 4.6 :

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.

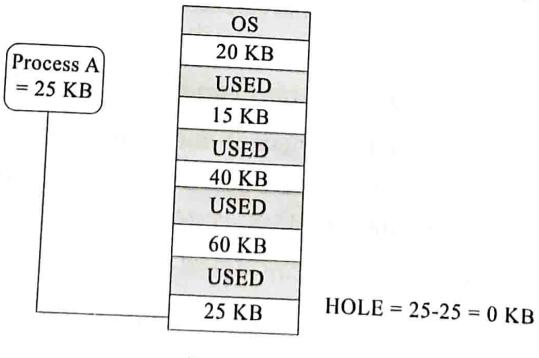

FIG 4.7 :

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.

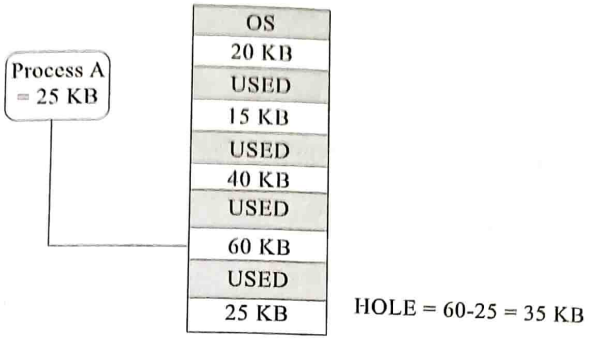

FIG 4.8 :

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.

Is Best-Fit really best ? : Although best fit minimizes the wastage space, it consumes a lot of processor time for searching the block which is close to the required size. Also, Best-fit may perform poorer than other algorithms in some cases.

4.6 FRAGMENTATION

As processes are loaded and removed from memory, the free memory space is broken into little pieces. It happens after sometimes that processes cannot be allocated to memory blocks considering their small size and memory blocks remains unused. This problem is known as ***fragmentation.***

Types of Fragmentation : There are mainly two types of fragmentation in the operating system. These are as follows:

1. Internal Fragmentation.
2. External Fragmentation.

1. **Internal Fragmentation :** When a process is allocated to a memory block, and if the process is smaller than the amount of memory requested, a free space is created in the given memory block. Due to this, the free space of the memory block is unused, which causes **internal** fragmentation.

For Example : Assume that memory allocation in RAM is done using fixed partitioning (i.e., memory blocks of fixed sizes). **2 MB, 4 MB, 4 MB**, and **8 MB** are the available sizes. The Operating System uses a part of this RAM.

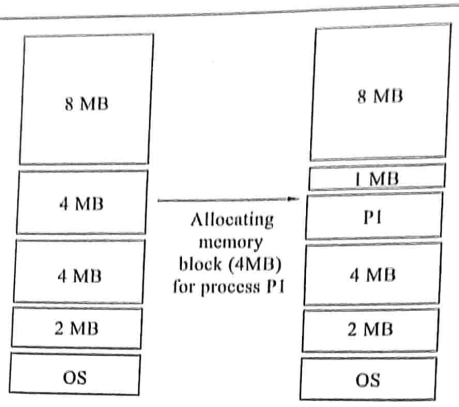

FIG 4.9 :

Let's suppose a process **P1** with a size of **3 MB** arrives and is given a memory block of **4 MB**. As a result, the **1 MB** of free space in this block is unused and cannot be used to allocate memory to another process. It is known as **internal fragmentation**.

How to avoid internal fragmentation ? : The problem of internal fragmentation may arise due to the fixed sizes of the memory blocks. It may be solved by assigning space to the process via dynamic partitioning. Dynamic partitioning allocates only the amount of space requested by the process. As a result, there is no internal fragmentation.

2. **External Fragmentation :** External fragmentation happens when a dynamic memory allocation method allocates some memory but leaves a small amount of memory unusable. The quantity of available memory is substantially reduced if there is too much external fragmentation. There is enough memory space to complete a request, but it is not contiguous. It's known as **external** fragmentation.

For Example :

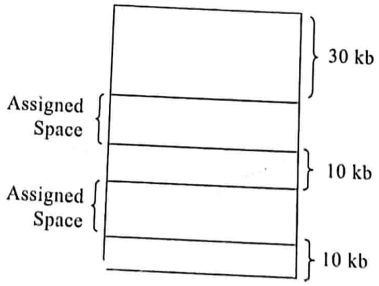

FIG 4.10 : Process 05 needs 45 KB memory space

Let's take the example of external fragmentation. In the above diagram, you can see that there is sufficient space **(50 KB)** to run a process **(05) (need 45 KB)**, but the memory is not contiguous. You can use compaction, paging, and segmentation to use the free space to execute a process.

How to Remove External Fragmentation? : This problem occurs when you allocate RAM to processes continuously. It is done in paging and segmentation, where memory is allocated to processes non-contiguously. As a result, if you remove this condition, external fragmentation may be decreased.

Compaction is another method for removing external fragmentation. External fragmentation may be decreased when dynamic partitioning is used for memory allocation by combining all free memory into a single large block. The larger memory block is used to allocate space based on the requirements of the new processes. This method is also known as defragmentation.

4.6.1 ADVANTAGES AND DISADVANTAGES OF FRAGMENTATION

There are various advantages and disadvantages of fragmentation. Some of them are as follows :

Advantages :

- **Fast Data Writes :** Data write in a system that supports data fragmentation may be faster than reorganizing data storage to enable contiguous data writes.
- **Fewer Failures :** If there is insufficient sequential space in a system that does not support fragmentation, the write will fail.
- **Storage Optimization :** A fragmented system might potentially make better use of a storage device by utilizing every available storage block.

Disadvantages :

- **Need for Regular Defragmentation :** A more fragmented storage device's performance will degrade with time, necessitating the requirement for time-consuming defragmentation operations.
- **Slower Read Times :** The time it takes to read a non-sequential file might increase as a storage device becomes more fragmented.

4.7 PAGING

- **Paging** is a storage mechanism that allows OS to retrieve processes from the secondary storage into the main memory in the form of pages.
- In the Paging method, the main memory is divided into small fixed-size blocks of physical memory, which is called ***frames***.
- The size of a frame should be kept the same as that of a page to have maximum utilization of the main memory and to avoid external fragmentation. Paging is used for faster access to data, and it is a logical concept.

Advantages of Paging :

- Easy to use memory management algorithm.
- No need for external Fragmentation.
- Swapping is easy between equal-sized pages and page frames.

Disadvantages of Paging :

- May cause Internal fragmentation.
- Page tables consume additional memory.
- Multi-level paging may lead to memory reference overhead.

4.8 HOW LOGICAL ADDRESS IS TRANSLATED INTO PHYSICAL ADDRESS

Paging is a memory management scheme that eliminates the need for contiguous allocation of physical memory.

This scheme permits the physical address space of a process to be non-contiguous.

- **Logical Address or Virtual Address (Represented in Bits) :** An address generated by the CPU
- **Logical Address Space or Virtual Address Space (Represented in Words or Bytes) :** The set of all logical addresses generated by a program
- **Physical Address (Represented in Bits) :** An address actually available on memory unit
- **Physical Address Space (Represented in Words or Bytes) :** The set of all physical addresses corresponding to the logical addresses

Example :

- If Logical Address = 31 bit, then Logical Address Space = 2^{31} words = 2 G words (1 G = 2^{30}).

- If Logical Address Space = 128 M words = $2^7 \times 2^{20}$ words, then Logical Address = $\log_2 2^{27}$ = 27 bits.
- If Physical Address = 22 bit, then Physical Address Space = 2^{22} words = 4 M words (1 M = 2^{20})
- If Physical Address Space = 16 M words = $2^4 \times 2^{20}$ words, then Physical Address = $\log_2 2^{24}$ = 24 bits

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 paging technique.

- The Physical Address Space is conceptually divided into a number of fixed-size blocks, called ***frames***.
- The Logical address Space is also splitted into fixed-size blocks, called ***pages***.
- Page Size = Frame Size

Let us consider an example :

- Physical Address = 12 bits, then Physical Address Space = 4 K words
- Logical Address = 13 bits, then Logical Address Space = 8 K words
- Page size = frame size = 1 K words (assumption)

Number of frames = Physical Adress Space/Frame size = 4K/1K = (4) = 2^2
Number of pages = Logical Address Space/Page size = 8 K/1K = (8) = 2^3

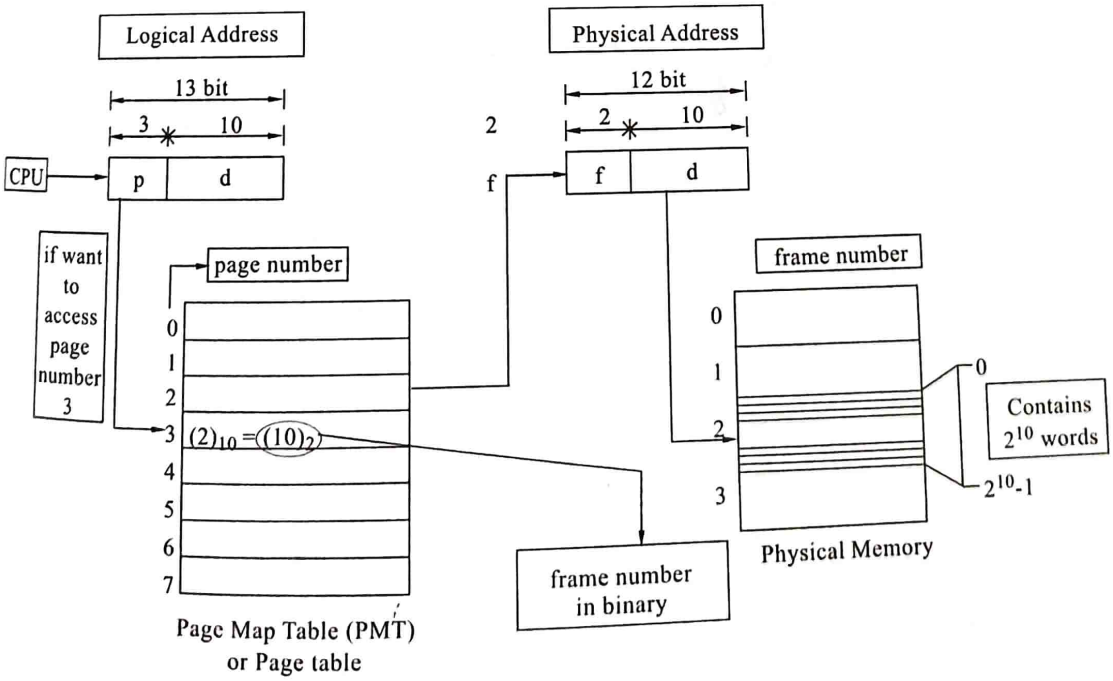

FIG 4.11 :

Address Generated by CPU is Divided into :

- **Page number(p)** : Number of bits required to represent the pages in Logic Address Space or Page number
- **Page offset(d)**: Number of bits required to represent particular word in a page page size of Logical Address Space or word number of a page or page offset

Physical Address is Divided into :

- **Frame Number(f)** : Number of bits required to represent the frame of Physic Address Space or Frame number.
- **Frame Offset(d)** : Number of bits required to represent particular word in a fram or frame size of Physical Address Space or word number of a frame or frame offse

The hardware implementation of page table can be done by using dedicated register But the usage of register for the page table is satisfactory only if page table is small page table contain large number of entries then we can use TLB(translatio Look-aside buffer), a special, small, fast look up hardware cache.

- The TLB is associative, high speed memory.
- Each entry in TLB consists of two parts : a tag and a value.
- When this memory is used, then an item is compared with all tags simultaneously If the item is found, then corresponding value is returned.

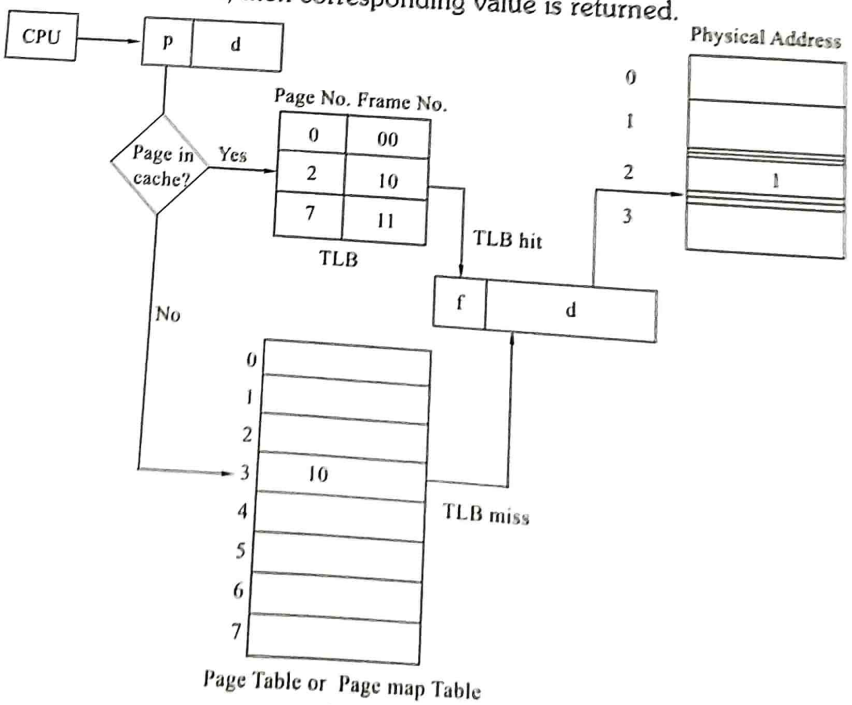

FIG 4.12 :

Main memory access time = m

If page table are kept in main memory,

Effective access time = m(for page table) + m(for particular page in page table)

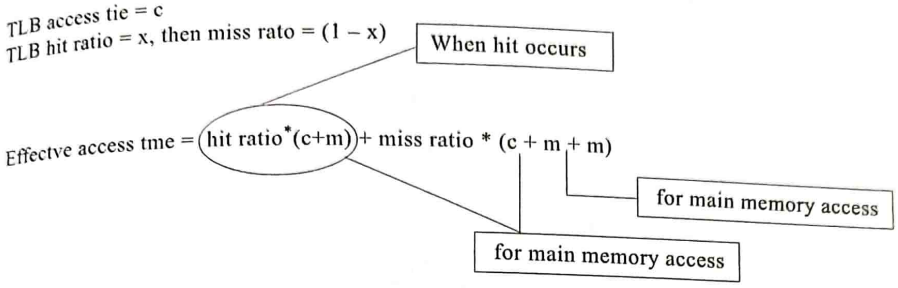

FIG 4.13 :

4.9 SEGMENTATION AND SEGMENTATION WITH PAGING

1. Pure segmentation is not very popular and not being used in many of the operating systems. However, Segmentation can be combined with Paging to get the best features out of both the techniques.

2. In Segmented Paging, the main memory is divided into variable size segments which are further divided into fixed size pages.

   - Pages are smaller than segments.
   - Each Segment has a page table which means every program has multiple page tables.
   - The logical address is represented as Segment Number (base address), Page number and page offset.

Segment Number : It points to the appropriate Segment Number.

Page Number : It Points to the exact page within the segme

Page Offset : Used as an offset within the page frame

Each Page table contains the various information about every page of the segment. The Segment Table contains the information about every segment. Each segment table entry points to a page table entry and every page table entry is mapped to one of the page within a segment.

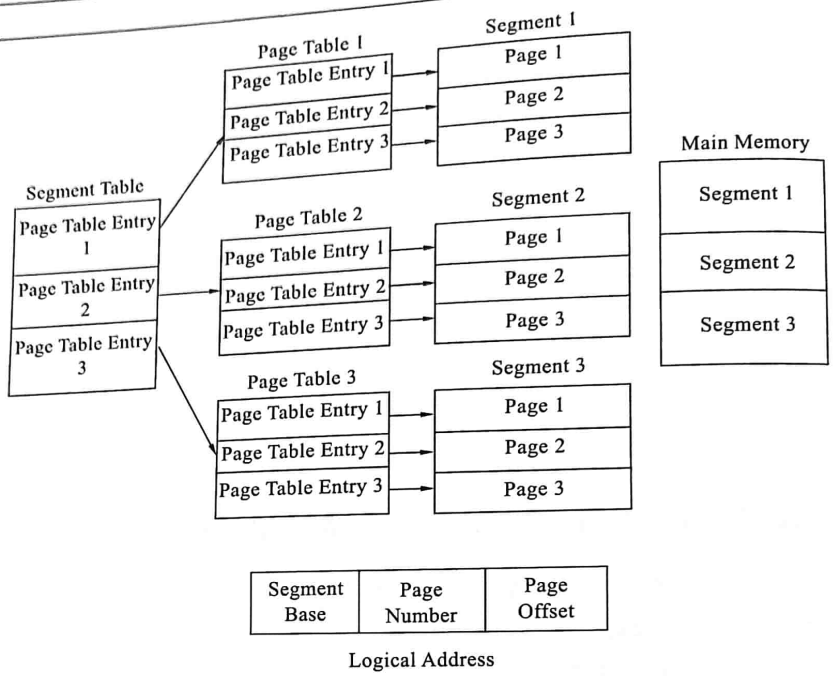

FIG 4.14 :

4.9.1 TRANSLATION OF LOGICAL ADDRESS TO PHYSICAL ADDRESS

- The CPU generates a logical address which is divided into two parts: Segment Number and Segment Offset.
- The Segment Offset must be less than the segment limit. Offset is further divided into Page number and Page Offset. To map the exact page number in the page table, the page number is added into the page table base.
- The actual frame number with the page offset is mapped to the main memory to get the desired word in the page of the certain segment of the process.

Advantages of Segmented Paging :

1. It reduces memory usage.
2. Page table size is limited by the segment size.
3. Segment table has only one entry corresponding to one actual segment.
4. External Fragmentation is not there.
5. It simplifies memory allocation.

Disadvantages of Segmented Paging :

1. Internal Fragmentation will be there.
2. The complexity level will be much higher as compare to paging.
3. Page Tables need to be contiguously stored in the memory.

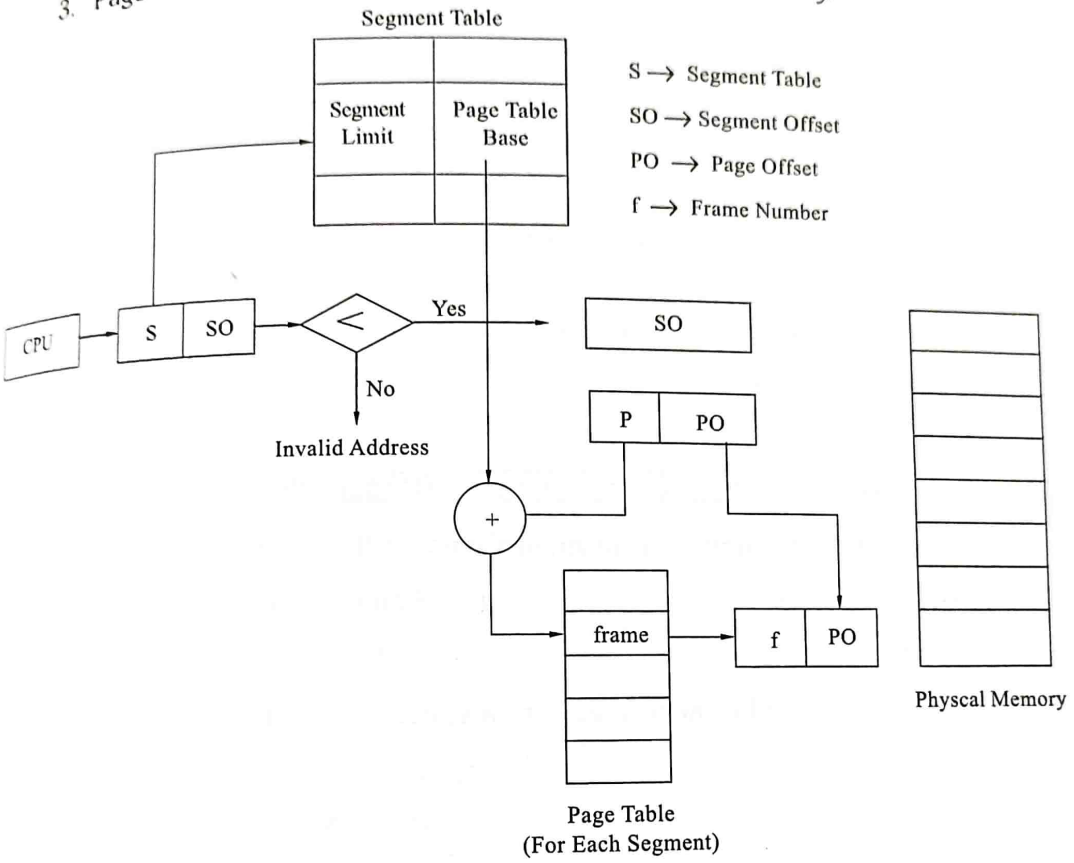

FIG 4.15 :

4.9.2 DIFFERENCE NETWEEN PAGNING & SEGMENTATION

S.No.	Key	Paging	Segmentation
1.	Memory Size	In Paging, a process address space is broken into fixed sized blocks called pages.	In Segmentation, a process address space is broken in varying sized blocks called sections.
2.	Accountability	Operating System divides the memory into pages.	Compiler is responsible to calculate the segment size, the virtual address and actual address.
3.	Size	Page size is determined by available memory.	Section size is determined by the user.

4.	Speed	Paging technique is faster in terms of memory access.	Segmentation is slower than paging
5.	Fragmentation	Paging can cause internal fragmentation as some pages may go underutilized.	Segmentation can cause external fragmentation as some memory blocks may not be used at all.
6.	Logical Address	During paging, a logical address is divided into page number and page offset.	During segmentation, a logical address is divided into section number and section offset.
7.	Table	During paging, a logical address is divided into page number and page offset.	During segmentation, a logical address is divided into section number and section offset.
8.	Data Storage	Page table stores the page data.	Segmentation table stores the segmentation data.

OBJECTIVE TYPE QUESTION

1. **CPU fetches the instruction from memory according to the value of _____** [ ]
   (a) Program counter (b) Status register
   (c) Instruction register (d) Program status word

2. **A memory buffer used to accommodate a speed differential is called _____** [ ]
   (a) Stack pointer (b) Cache
   (c) Accumulator (d) Disk buffer

3. **Which one of the following is the address generated by CPU?** [ ]
   (a) Physical address (b) Absolute address
   (c) Logical address (d) None of the above

4. **Run time mapping from virtual to physical address is done by _____** [ ]
   (a) Memory management unit (b) CPU
   (c) PCI (d) None of the above

5. **Memory management technique in which system stores and retrieves data from secondary storage for use in main memory is called?** [ ]
   (a) Fragmentation (b) Paging
   (c) Mapping (d) None of the above

REVIEW QUESTIONS

Part-A

1. What is overlay?
2. Define Address Binding.
3. What is storage management?
4. What is paging?
5. Define Dynamic Linking.

Part-B

1. Explain briefly about Swapping. **(Mar/Apr. 2014)**
2. Explain about Single Partition allocation. **(Apr/May 2012)**
3. Explain about Multiple Partition allocation. **(Mar/Apr. 2007)**
4. Differentiate between Paging and Segmentation.
5. What is internal fragmentation? Explain in-detail **(Apr. 2008)**
6. What is external fragmentation? Explain in-detail **(Apr/May. 2006)**
7. Explain how logical address is translated into physical address.
8. Describe about Paging Concept. **(Apr/May. 2012)**
9. Describe about segmentation. **(Apr. 2006)**
10. Differentiate between Single and Multiple partition allocation.