Unit IV **(8 Hrs)**

Memory management : Contiguous and non-contiguous, Swapping, Paging, Segmentation and demand Paging, Virtual Memory, Management of Virtual memory: allocation, fetch and replacement

Memory Management Requirements

1) Relocation :-

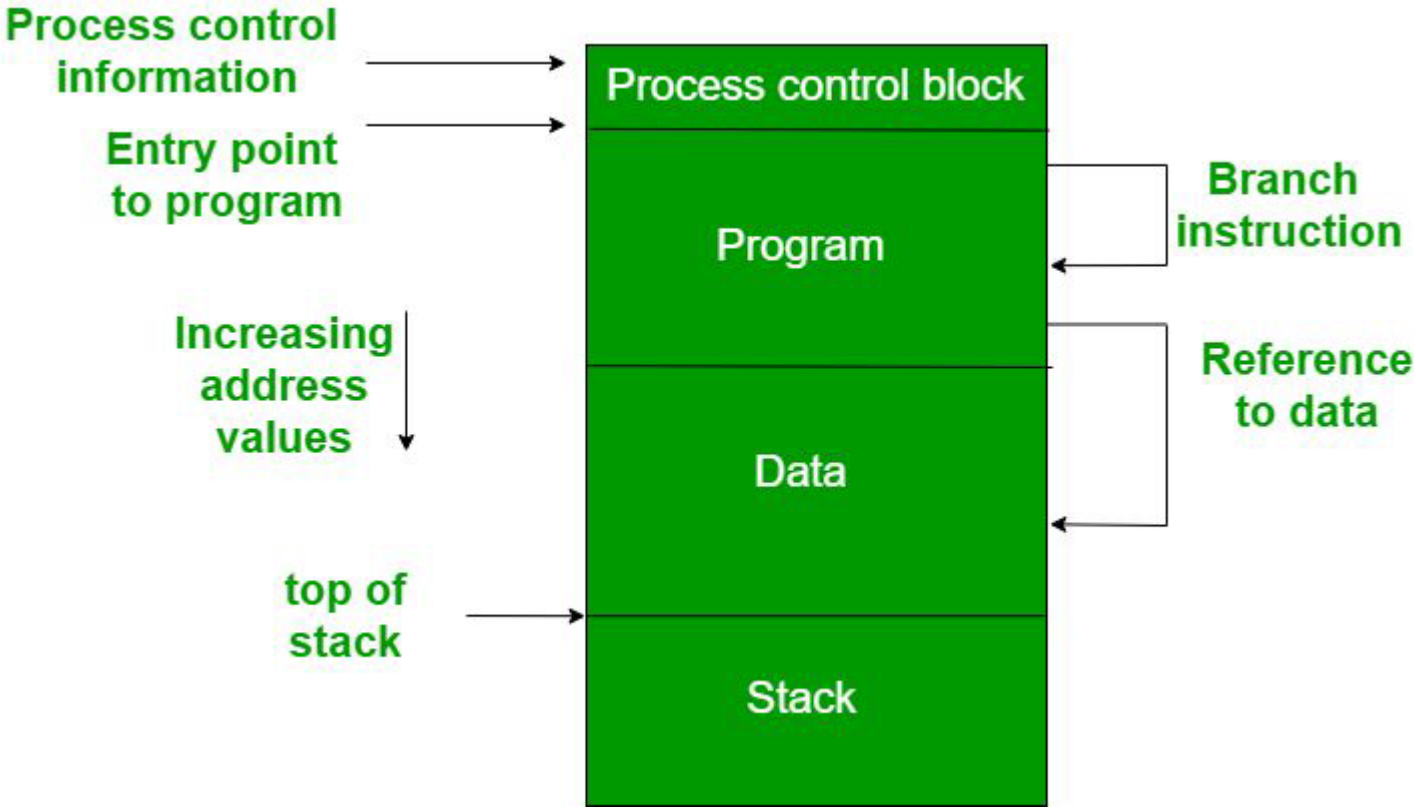
The available memory is generally shared among the number of processes in a multiprogramming system, so it is not possible to know in advance which other programs will be resident in main memory at the time of execution of his program. Due to Swapping the active process may swapped with other. Also due to swapping out , the operating system have a larger pool of ready-to-execute process.

When a program gets swapped out to a disk memory, then it is not always possible that when it is swapped back into main memory then it occupies the previous memory location, since the location may still be occupied by another process. We may need to relocate the process to a different area of memory. Thus there is a possibility that program may be moved in main memory due to swapping.

All the time the operating system will need to know many things including the location of process control information, the execution stack, and the code entry.

Within a program, there are memory references in various instructions and these are called logical addresses.

After loading of the program into main memory, the processor and the operating system must be able to translate logical addresses into physical addresses. Branch instructions contain the address of the next instruction to be executed.



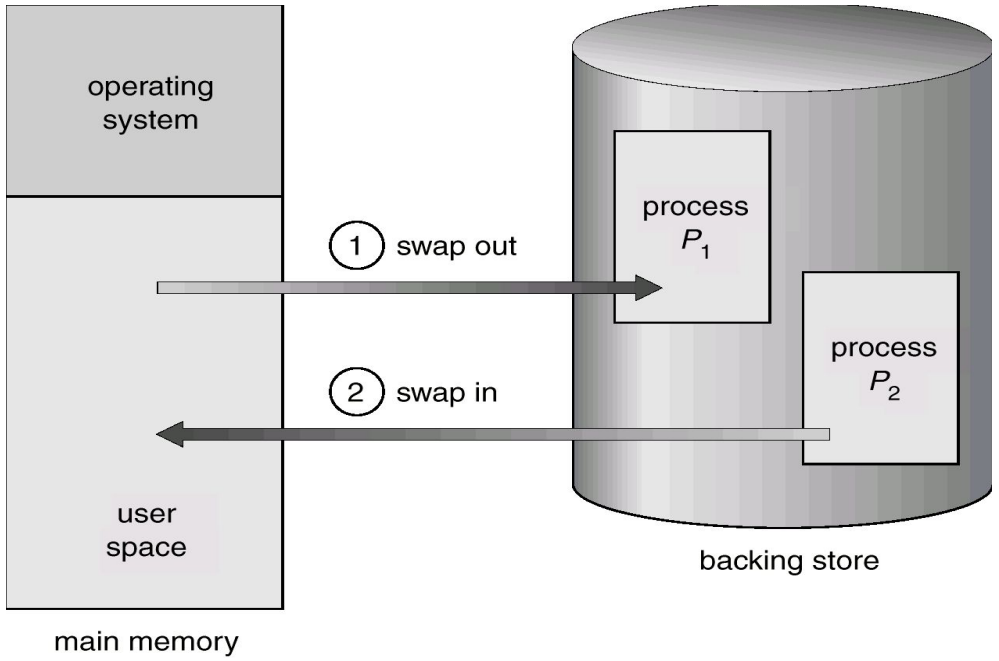
Swapping :-

A process can be *swapped* temporarily out of memory to a *backing store*, and then brought back into memory for continued execution.

Backing store – fast disk large enough to accommodate copies of all memory images for all users; must provide direct access to these memory images.

Roll out, roll in – swapping variant used for priority-based scheduling algorithms; lower-priority process is swapped out so higher-priority process can be loaded and executed.

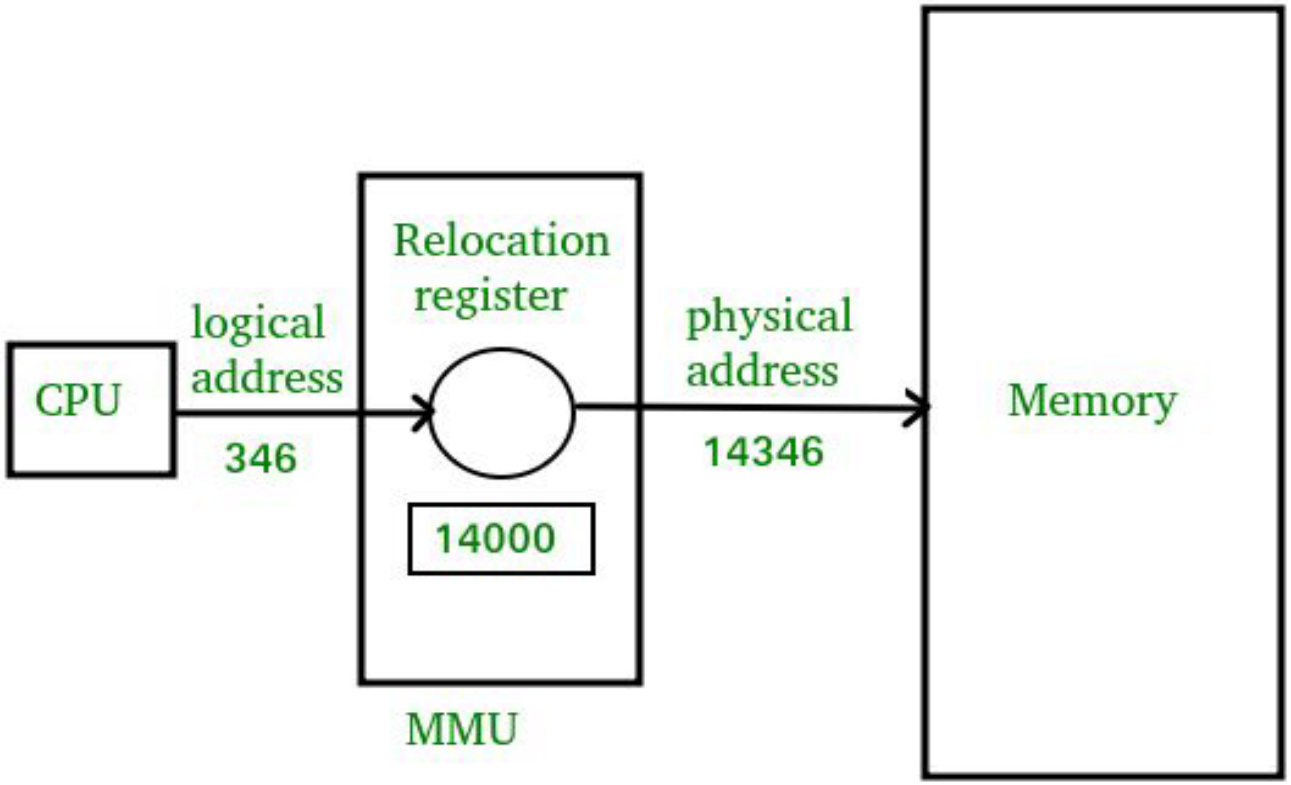
Major part of swap time is transfer time; total transfer time is directly proportional to the *amount* of memory swapped.



Logical Address is generated by CPU while a program is running. The logical address is virtual address as it does not exist physically, therefore, it is also known as Virtual Address. This address is used as a reference to access the physical memory location by CPU.
The hardware device called Memory-Management Unit is used for mapping logical address to its corresponding physical address.

Physical Address identifies a physical location of required data in a memory. The user never directly deals with the physical address but can access by its corresponding logical address. The user program generates the logical address and thinks that the program is running in this logical address but the program needs physical memory for its execution, therefore, the logical address must be mapped to the physical address by MMU before they are used.

Program must be brought into memory and placed within a process for it to be executed.
Input queue – collection of processes on the disk that are waiting to be brought into memory for execution.
User programs go through several steps before being executed.



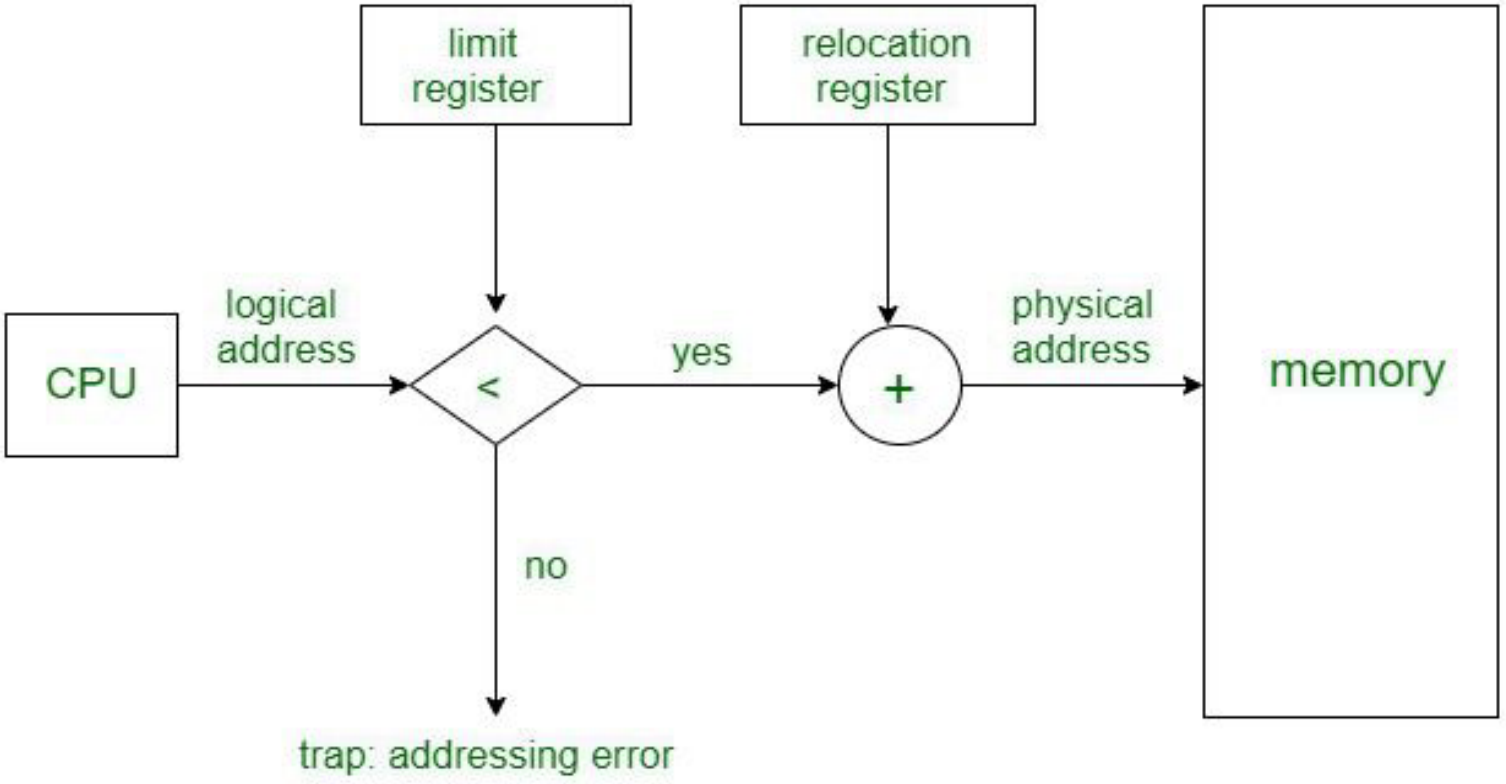
OS
0000
13999
14000

14346

15000

In MMU (Hardware device that maps virtual to physical address.) scheme, the value in the relocation register is added to every address generated by a user process at the time it is sent to memory.

The user program deals with *logical* addresses; it never sees the *real* physical addresses.



The Memory Management Unit is a combination of 2 registers –

1) Base Register (Relocation Register) 2) Limit Register.

Base Register – contains the starting physical address of the process.

Limit Register -mentions the limit relative to the base address on the region occupied by the process.

The logical address generated by the CPU is first checked by the limit register, If the value of the logical address generated is less than the value of the limit register, the base address stored in the relocation register is added to the logical address to get the physical address of the memory location.
If the logical address value is greater than the limit register, then the CPU traps to the OS, and the OS terminates the program by giving fatal error.

In Non Contiguous Memory allocation, processes can be allocated anywhere in available space. The address translation in non-contiguous memory allocation is difficult.

Column 1	Column 2	Column3	Column 4	Column 5
Desk11 01	Desk21 08	Desk31 15	Desk41 22	Desk51 29
Desk12 02	Desk22 09	Desk32 16	Desk42 23	Desk52 30
Desk13 03	Desk23 10	Desk33 17	Desk43 24	Desk53 31
Desk14 04	Desk24 11	Desk34 18	Desk44 25	Desk54 32
Desk15 05	Desk25 12	Desk35 19	Desk45 26	Desk55 33
Desk16 06	Desk26 13	Desk36 20	Desk46 27	Desk56 34
Desk17 07	Desk27 14	Desk37 21	Desk47 28	Desk57 35

Logical Address Desk 36 (It indicates Column 3 and desk No. 6)

But actual desk no is the physical place or Physical address which will be calculated by MMU

Every column contains 7 desk. As given address is from column no 3. So **7+7+6** is the

physical no i.e. 20

2) Sharing :- Protection mechanism must have the flexibility to allow several processes to access the same portion of the main memory. e.g. if the number of process are executing the same program then it is advantageous to allow each process to access the same copy of the program rather than its own separate copy.

3) Protection: When we have two program at the same time there is a danger that one program can write to the address space of another program so in the manner every process should be protected against unwanted interference by other processes. Satisfaction of the relocation requirement increase the difficulty of satisfying the protection requirement. CPU tend to support absolute addressing which means that code runs differently when loaded in different places. This is not possible to check the absolute address at the compile time. Most of the programming languages allow the dynamic calculation of the address at run time.

4) Logical organization :- As user write program in modules with different characteristics then logical organization be like instruction modules are execute-only, data modules are either read-only or read/write and some modules are private other are public. To effective deal with the user program, the operating system and hardware must support a basic form of module to provide the required protection and sharing.

5) Physical organization :- As we know computer memory is organized into two levels main and secondary memory. Main memory is a volatile which provide fast access at relatively high cost and secondary memory is non-volatile which is going to provide slower and cheaper access than main memory. The flow of information is mainly between main memory and secondary memory which is major system concern but sometimes it is impractical for the programmers understand how. As main memory is available for the program and for data also which may be insufficient for that programmer must use mechanism of overlaying that hold the

data and program that are needed at the given time. There is also another factor that going to affect programmer concern that is multiprogramming environment. In such environment, programmer does not know how much space is available.

The main memory can be broadly allocated in two ways –

1) Contiguous Memory Allocation

a) Fixed partition scheme b) Variable partition scheme.

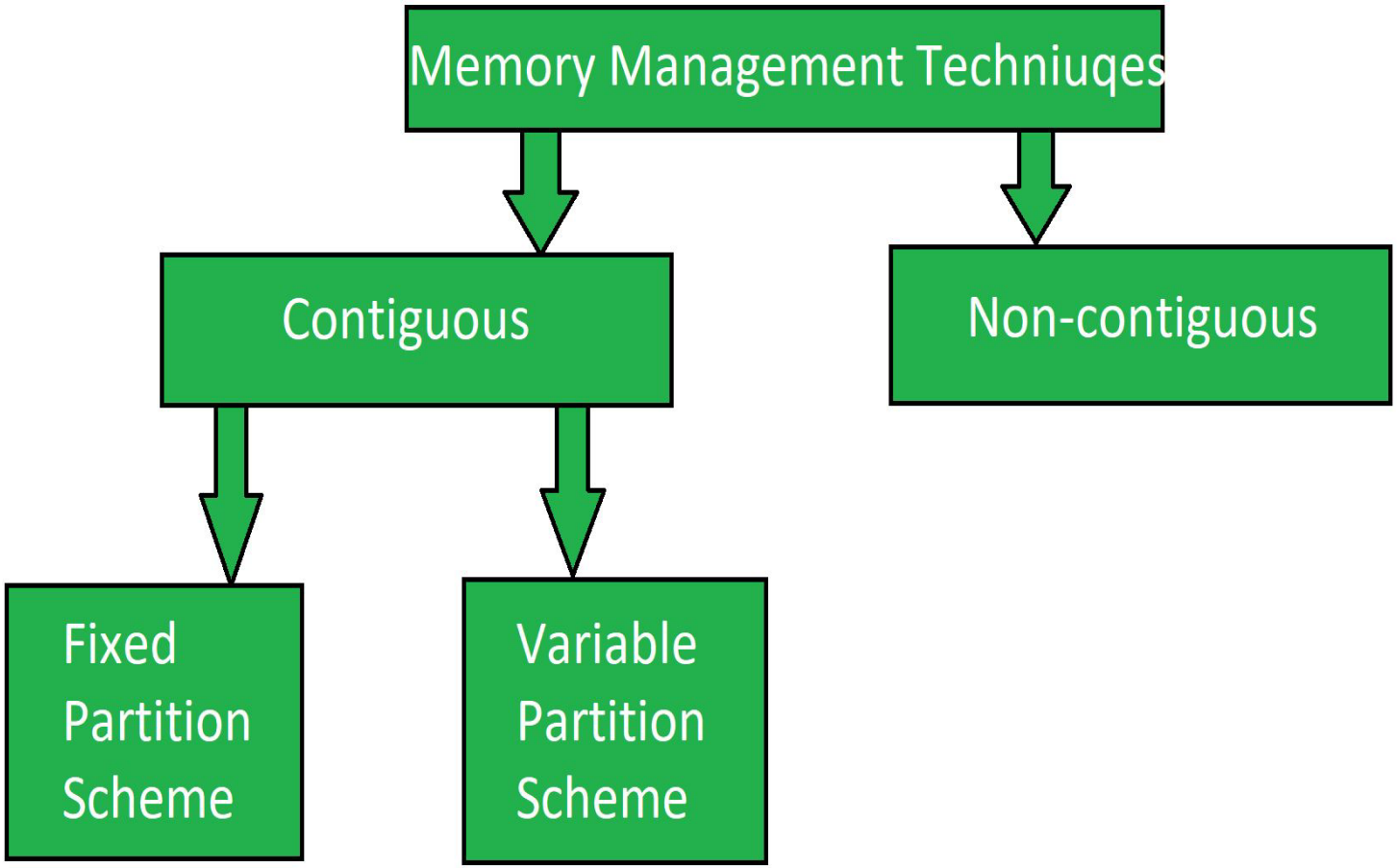
Different Partition Allocation methods are used in Contiguous memory allocations

a) First Fit b) Best Fit c) Worst Fit d) Next Fit

2) Non-Contiguous Memory Allocation

a) Paging b) Multitlevel Paging c) Inverted Paging

d) Segmentation e) Segmented Paging


Memory Management Techniuqes
Contiguous
Non-contiguous
Fixed Partition Scheme
Variable Partition Scheme

In the fixed partition scheme :-

i) memory is divided into fixed number of partitions.

ii) number of partitions are fixed in the memory.

iii) in every partition only one process will be accommodated.

iv) Degree of multi-programming is restricted by number of partitions in the memory.

Maximum size of the process is restricted by maximum size of the partition. Every partition is associated with the *limit registers*.

Limit Registers: It has two limit:

Lower Limit: Starting address of the partition.

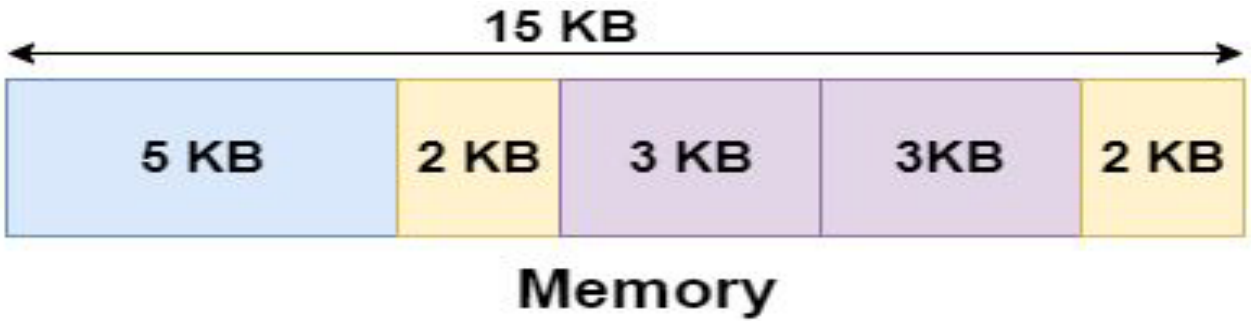
Upper Limit: Ending address of the partition.

Fixed-size Partition Scheme :-

This technique is also known as **Static partitioning**. In this scheme, the system divides the memory into fixed-size partitions. The partitions may or may not be the same size. The size of each partition is fixed as indicated by the name of the technique and it cannot be changed.

In this partition scheme, each partition may contain exactly one process. There is a problem that this technique will limit the degree of multiprogramming because the number of partitions will basically decide the number of processes.

Whenever any process terminates then the partition becomes available for another process.



Advantages of Fixed-size Partition Scheme :-

This scheme is simple and is easy to implement

It supports multiprogramming as multiple processes can be stored inside the main memory.

Management is easy using this scheme

Disadvantages of Fixed-size Partition Scheme :-

1. Internal Fragmentation

Suppose the size of the process is lesser than the size of the partition in that case some size of the partition gets wasted and remains unused. This wastage inside the memory is generally termed as **Internal fragmentation**

2. Limitation on the size of the process

If in a case size of a process is more than that of a maximum-sized partition then that process cannot be loaded into the memory. Due to this, a condition is imposed on the size of the process and it is: the size of the process cannot be larger than the size of the largest partition.

3. Degree of multiprogramming is less

In this partition scheme, as the size of the partition cannot change according to the size of the process. Thus the degree of multiprogramming is very less and is fixed.

Variable-size Partition Scheme :-

This scheme is also known as Dynamic partitioning and is came into existence to overcome the drawback i.e internal fragmentation that is caused by Static partitioning. In this partitioning, scheme allocation is done dynamically.

The size of the partition is not declared initially. Whenever any process arrives, a partition of size equal to the size of the process is created and then allocated to the process. Thus the size of each partition is equal to the size of the process.

As partition size varies according to the need of the process so in this partition scheme there is no internal fragmentation.

Advantages :-

1. **No Internal Fragmentation** :-

As in this partition scheme space in the main memory is allocated strictly according to the requirement of the process thus there is no chance of internal fragmentation. Also, there will be no unused space left in the partition.

2. Degree of Multiprogramming is Dynamic :-

As there is no internal fragmentation in this partition scheme due to which there is no unused space in the memory. Thus more processes can be loaded into the memory at the same time.

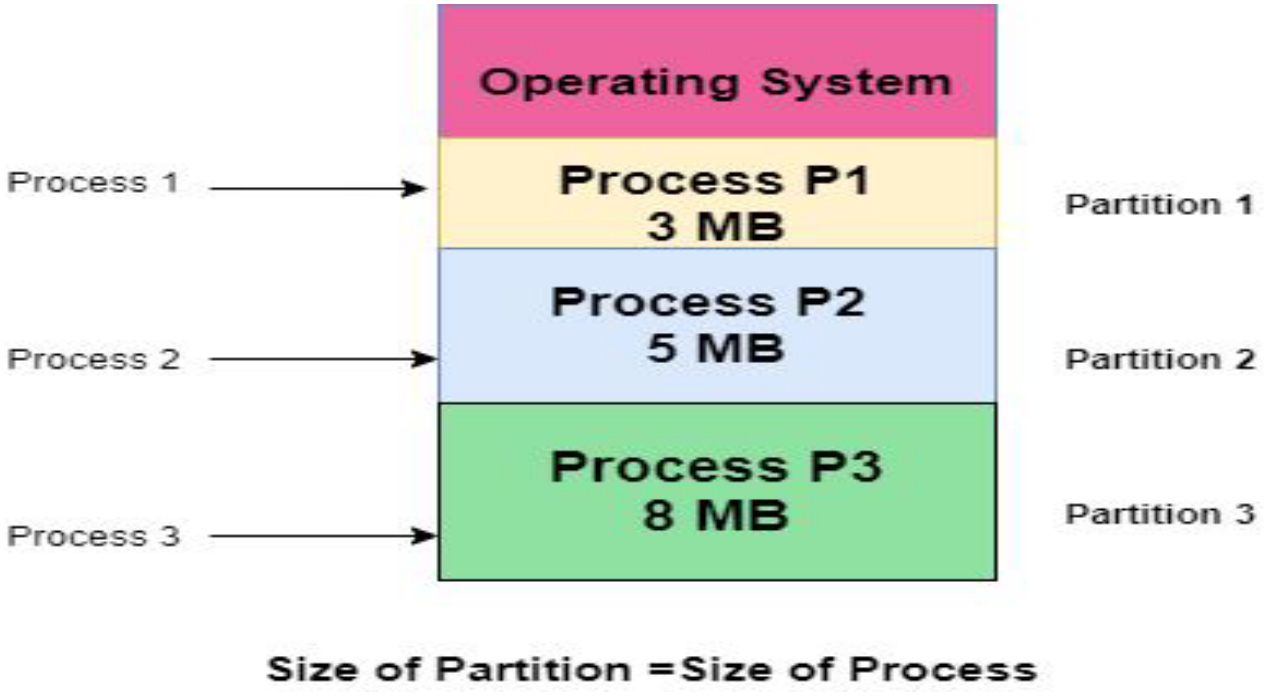
3.No Limitation on the Size of Process :-

In this partition scheme as the partition is allocated to the process dynamically thus the size of the process cannot be restricted because the partition size is decided according to the process size.

Disadvantages :-

1) **External Fragmentation** :- As there is no internal fragmentation which is an advantage of using this partition scheme does not mean there will no external fragmentation.

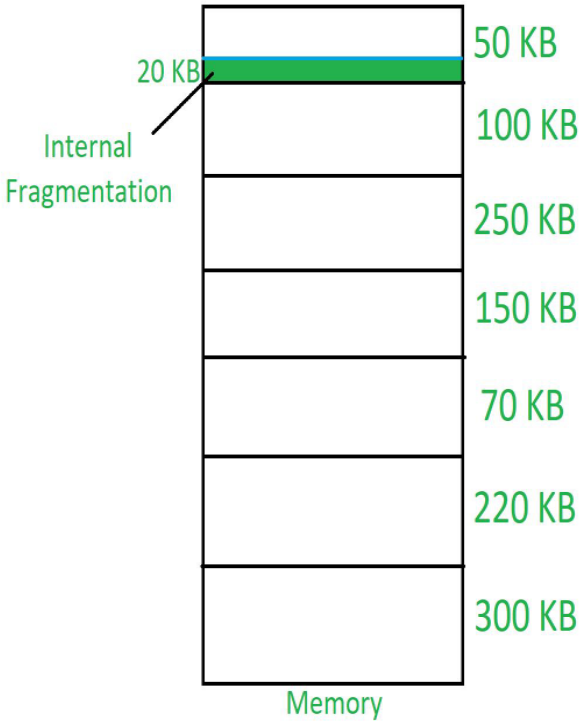
e.g.: In diagram- process P1(3MB) and process P3(8MB) completed their execution. Hence there are two spaces left i.e. 3MB and 8MB. Let's there is a Process P4 of size 15 MB comes. But the empty space in memory cannot be allocated as no spanning is allowed in contiguous allocation.



Because the rule says that process must be continuously present in the main memory in order to get executed. Thus it results in External Fragmentation.

2. Difficult Implementation :-

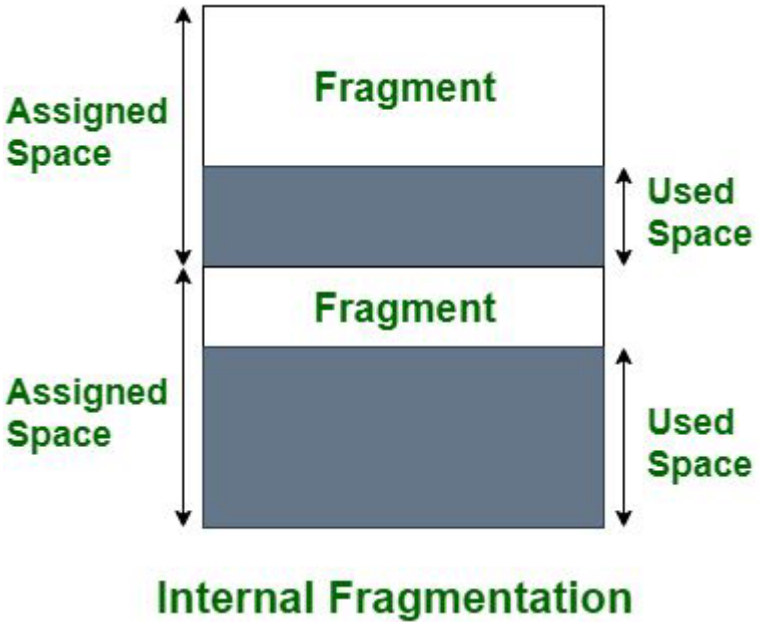
The implementation of this partition scheme is difficult as compared to the Fixed Partitioning scheme as it involves the allocation of memory at run-time rather than during the system configuration. As we know that OS keeps the track of all the partitions but here allocation and deallocation are done very frequently and partition size will be changed at each time so it will be difficult for the operating system to manage everything.



P = 30 KB

In the given figure 50 KB partition is used to load a process of 30 KB. So the remaining 20 KB is wasted.

Internal fragmentation



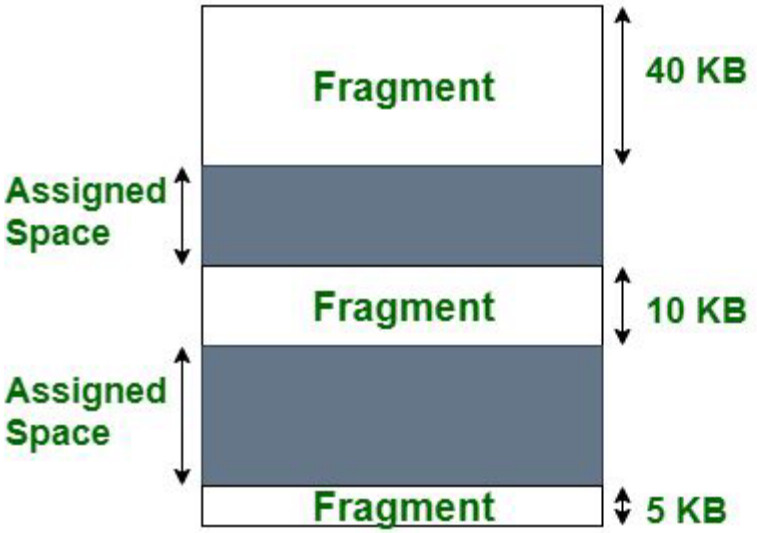
Internal fragmentation found into fixed sized blocks.

Whenever a process request for the memory, the fixed sized block is allotted to the method. The memory allotted to the process is somewhat larger than the memory requested, then the distinction (difference) between allotted and requested memory is that the **Internal fragmentation.**

External Fragmentation is found in variable partition scheme.

Initially the memory is a single continuous free block. Whenever the request by the process arrives, accordingly partition will be made in the memory. If the smaller processes keep on coming then the larger partitions will be made into smaller partitions.
It happens when there's a sufficient quantity of area within the memory to satisfy the memory request of a method. however the process's memory request cannot be fulfilled because the memory offered is during a non-contiguous manner.

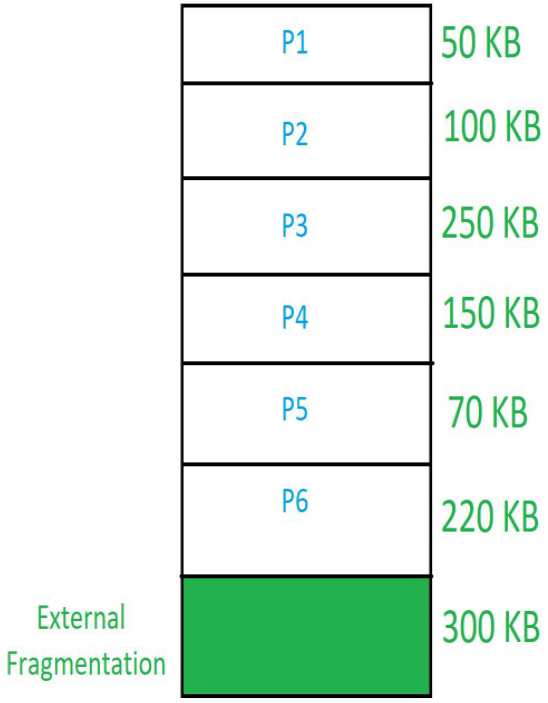
To overcome the problem of external fragmentation, compaction technique is used or non-contiguous memory management techniques are used.



Process 07 needs 50KB memory space

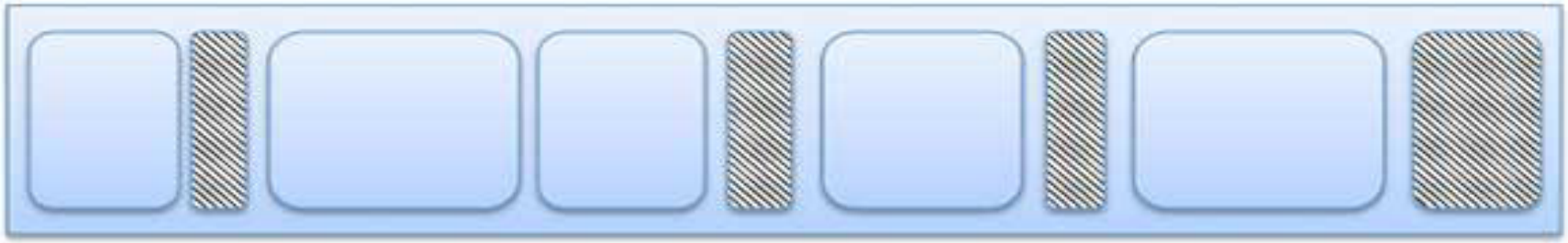
Here is enough space (55 KB) to run a process-07 (required 50 KB) but the memory (fragment) is not contiguous. Here, we use compaction, paging or segmentation to use the free space to run a process.

Moving all the processes toward the top or towards the bottom to make free available memory in a single continuous place is called as compaction. Compaction is undesirable to implement because it interrupts all the running processes in the memory.

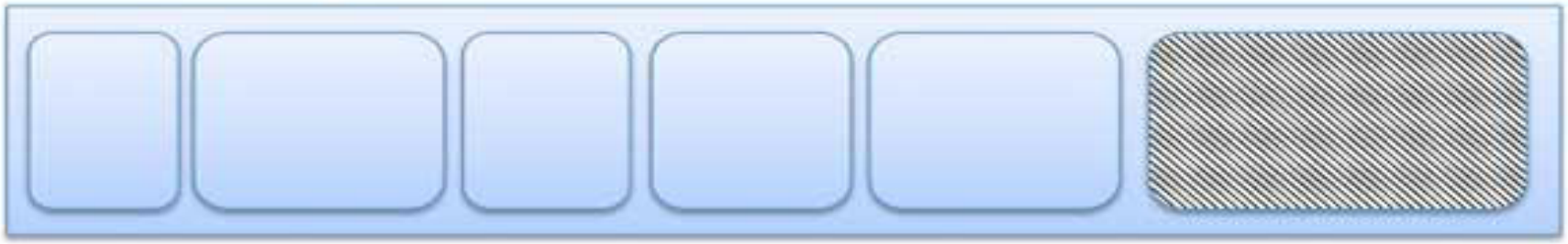


Sr. No.	Internal fragmentation	External fragmentation
1.	In internal fragmentation fixed-sized memory, blocks square measure appointed to process.	In external fragmentation, variable-sized memory blocks square measure appointed to method.
2.	Internal fragmentation happens when the method or process is larger than the memory.	External fragmentation happens when the method or process is removed.
3.	The solution of internal fragmentation is best-fit block.	Solution of external fragmentation is compaction, paging and segmentation.
4.	Internal fragmentation occurs when memory is divided into fixed sized partitions.	External fragmentation occurs when memory is divided into variable size partitions based on the size of processes.
5.	The difference between memory allocated and required space or memory is called Internal fragmentation.	The unused spaces formed between non-contiguous memory fragments are too small to serve a new process, is called External fragmentation .

Fragmented memory before compaction



Memory after compaction



External fragmentation

Total memory space is enough to satisfy a request or to reside a process in it, but it is not contiguous, so it cannot be used.

Internal fragmentation

Memory block assigned to process is bigger. Some portion of memory is left unused, as it cannot be used by another process.

Memory allocation is a process by which computer programs are assigned memory or space.

It is of three types :

First Fit Allocation :- The first hole that is big enough is allocated to the program.

Best Fit Allocation :- The smallest hole that is big enough is allocated to the program.

Worst Fit Allocation :- The largest hole that is big enough is allocated to the program.

Different Partition Allocation methods are used in Contiguous memory allocations

1) First Fit :-

In the first fit approach is to allocate the first free partition or hole large enough which can accommodate the process. It finishes after finding the first suitable free partition.

Advantage

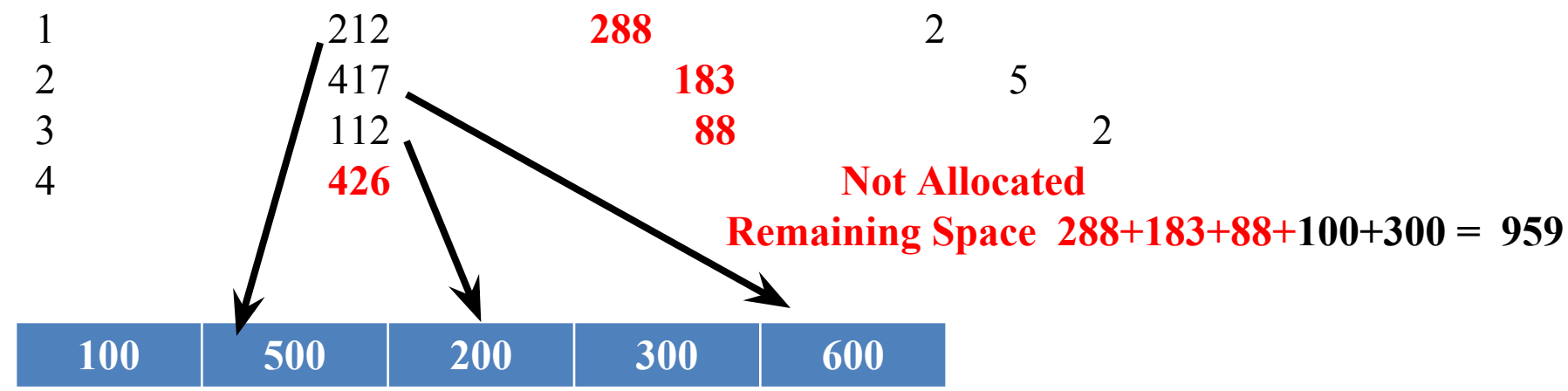
Fastest algorithm because it searches as little as possible.

Disadvantage

The remaining unused memory areas left after allocation become waste if it is too smaller. Thus request for larger memory requirement cannot be accomplished.

Input : Memory Block Size = {100, 500, 200, 300, 600};

Process Size = {212, 417, 112, 426};



2) Best Fit :-

The best fit deals with allocating the smallest free partition which meets the requirement of the requesting process. This algorithm first searches the entire list of free partitions and considers the smallest hole that is adequate. It then tries to find a hole which is close to actual process size needed.

Advantage

Memory utilization is much better than first fit as it searches the smallest free partition first available.

Disadvantage

It is slower and may even tend to fill up memory with tiny useless holes.

Input : Memory Block Size = {100, 500, 200, 300, 600};

Process Size = {212, 417, 112, 426};

Process No.	Process Size	Wasted Size	Block no.
1	212	88	4
2	417	83	2
3	112	88	3
4	426	174	5

Remaining Space **88+83+88+174+100 = 533**

3) Worst fit :-

In worst fit approach is to locate largest available free portion so that the portion left will be big enough to be useful. It is the reverse of best fit.

Advantage

Reduces the rate of production of small gaps.

Disadvantage

If a process requiring larger memory arrives at a later stage then it cannot be accommodated as the largest hole is already split and occupied.

Input : Memory Block Size = {100, 500, 200, 300, 600};

Process Size = {212, 417, 112, 426};

Process No.	Process Size	Wasted Size	Block no.
1	212	383	5
2	417	83	2
3	112	188	4
4	426	**Not Accommodated**	

Remaining Space **383+83+188+100+200 = 766**