Memory allocations techniques

Basic of memory allocation

Address space

- Physical address
- Logical address

Memory Allocation

- Contiguous memory allocation
- Non-contiguous memory allocation

Placement Algorithms

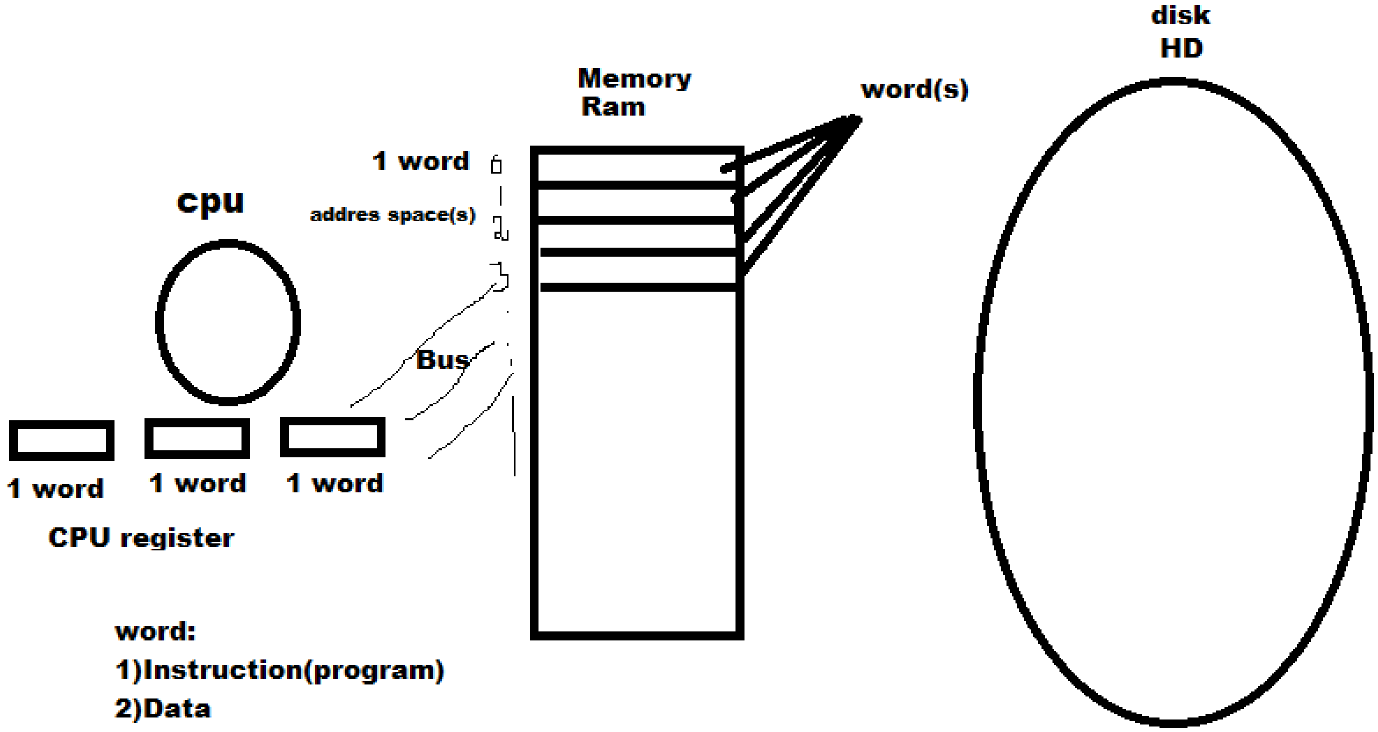
Compaction & paging

Basic of Memory Location

Memory management is the functionality of an operating system which handles or manages primary memory and moves processes back and forth between main memory and disk during execution. Memory management keeps track of each and every memory location, regardless of either it is allocated to some process or it is free. It checks how much memory is to be allocated to processes. It decides which process will get memory at what time. It tracks whenever some memory gets freed or unallocated and correspondingly it updates the status.

To achieve a degree of multiprogramming and proper utilization of memory, memory management is important. Many memory management methods exist, reflecting various approaches, and the effectiveness of each algorithm depends on the situation.

Various method used to load the process into memory

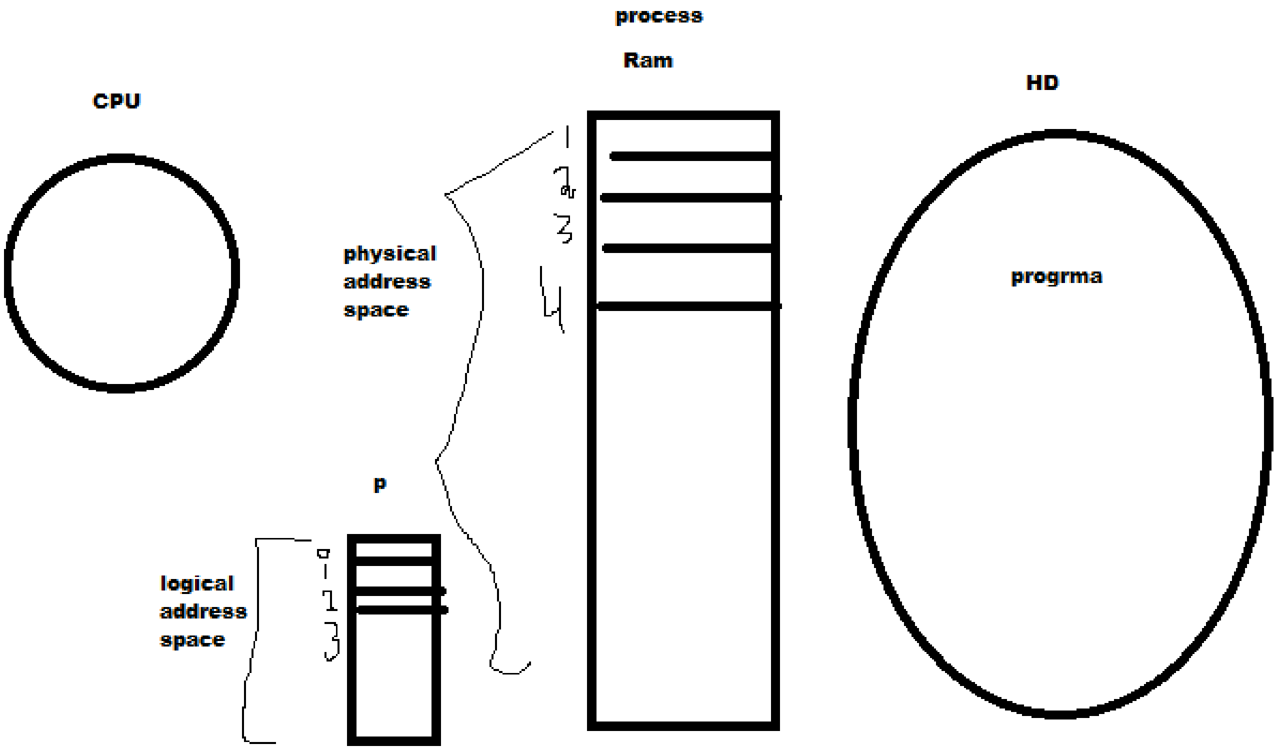


Address space

The range of virtual addresses that the operating system assigns to a separately running program is called an address space. This is the area of contiguous virtual addresses available for executing instructions and storing data.

In operating systems, logical and physical addresses are used to manage and access memory. Here's an overview of each:

Logical address: A logical address, also known as a virtual address, is an address generated by the CPU during program execution. It is the address seen by the process and is relative to the program's address space. The process accesses memory using logical addresses, which are translated by the operating system into physical addresses.
Physical address: A physical address is the actual address in main memory where data is stored.



Memory Allocation

Memory allocation is a process by which computer programs and services are assigned with physical or virtual memory space.

Memory allocation is the process of reserving a partial or complete portion of computer memory for the execution of programs and processes. Memory allocation is achieved through a process known as memory management.

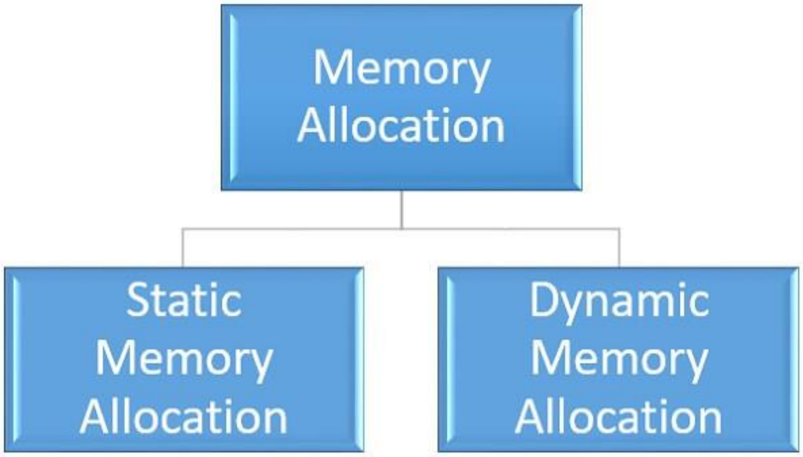
Process placing process in the memory is process allocation

Memory allocation has two core types;

- Static Memory Allocation: The program is allocated memory at compile time.
- Dynamic Memory Allocation: The programs are allocated with memory at run time.

Contiguous and Non-Contiguous Memory Allocation in Operating System

Memory is a huge collection of bytes, and memory allocation refers to allocating space to computer applications. There are mainly two types of memory allocation: *contiguous* and *non-contiguous memory allocation.* Contiguous memory allocation allows a single memory space to complete the tasks. On the other hand, non-contiguous memory allocation assigns the method to distinct memory sections at numerous memory locations.



Contiguous Memory Allocation

It is the type of ***memory allocation method.*** When a process requests the memory, a single contiguous section of memory blocks is allotted depending on its requirements.

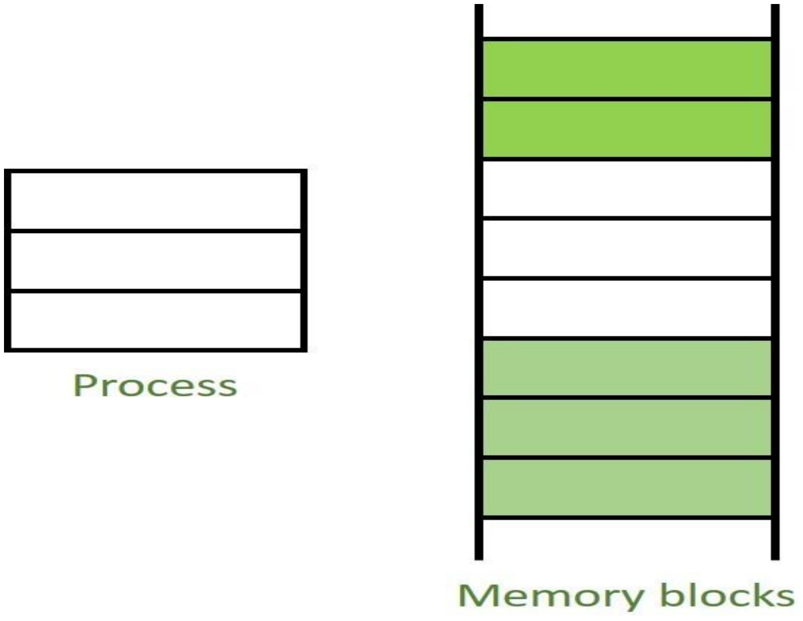
Contiguous memory allocation is basically a method in which a single contiguous section/part of memory is allocated to a process or file needing it. Because of this all the available memory space resides at the same place together, which means that the freely/unused available memory partitions are not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

Advantages

1. It is simple to keep track of how many memory blocks are left, which determines how many more processes can be granted memory space.
2. The read performance of contiguous memory allocation is good because the complete file may be read from the disk in a single task.
3. The contiguous allocation is simple to set up and performs well.

Disadvantages

1. When generating a new file, it must know its eventual size to select the appropriate hole size.
2. When the disk is filled up, it would be necessary to compress or reuse the spare space in the holes.



Contiguous Memory Allocation

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.

It allows a process to obtain multiple memory blocks in various locations in memory based on its requirements. The non-contiguous memory allocation also reduces memory wastage caused by ***internal*** and ***external*** fragmentation because it uses the memory holes created by internal and external fragmentation.

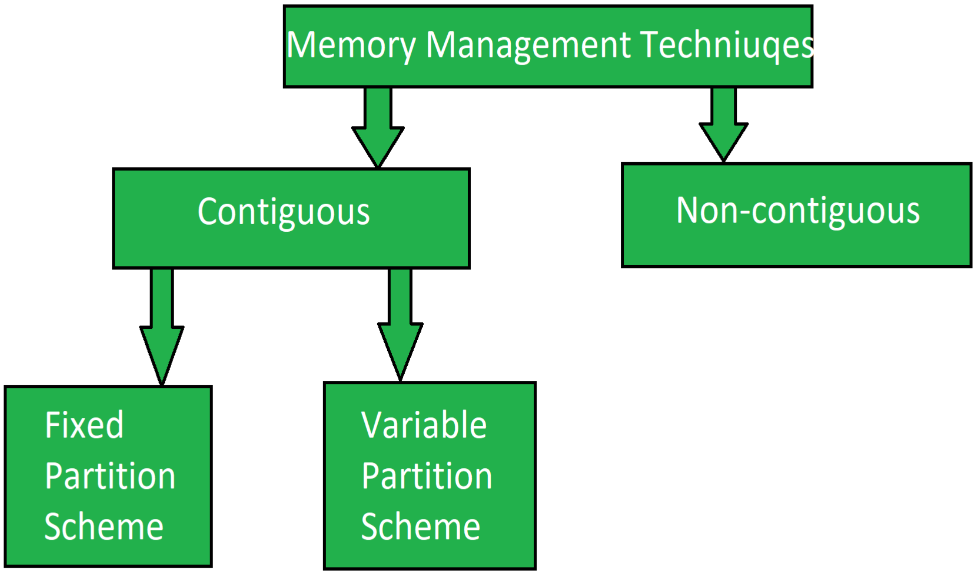
Non-contiguous memory allocation can decrease memory wastage, but it also raises address translation overheads. As the process portions are stored in separate locations in memory, the memory execution is slowed because time is consumed in address translation.

Advantages

1. It has the advantage of reducing memory waste, but it increases overhead because of the address translation.

Disadvantages

1. The downside of this memory allocation is that the access is slow because you must reach the other nodes using pointers and traverse them.



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.

Example

Let's take an example of fixed size partitioning scheme, we will divide a memory size of 15 KB into fixed-size partitions:

It is important to note that these partitions are allocated to the processes as they arrive and the partition that is allocated to the arrived process basically depends on the algorithm followed.

If there is some wastage inside the partition then it is termed Internal Fragmentation**.**

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

Some disadvantages of using this scheme are as follows:

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

As we have shown in the above diagram the 70 KB partition is used to load a process of 50 KB so the remaining 20 KB got wasted.

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. External Fragmentation

It is another drawback of the fixed-size partition scheme as total unused space by various partitions cannot be used in order to load the processes even though there is the availability of space but it is not in the contiguous fashion.

4. 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 of Variable-size Partition Scheme

Some Advantages of using this partition scheme are as follows:

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 of Variable-size Partition Scheme

Some Disadvantages of using this partition scheme are as follows:

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. Let us understand this with the help of an example: In the above 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.

Placement Algorithms

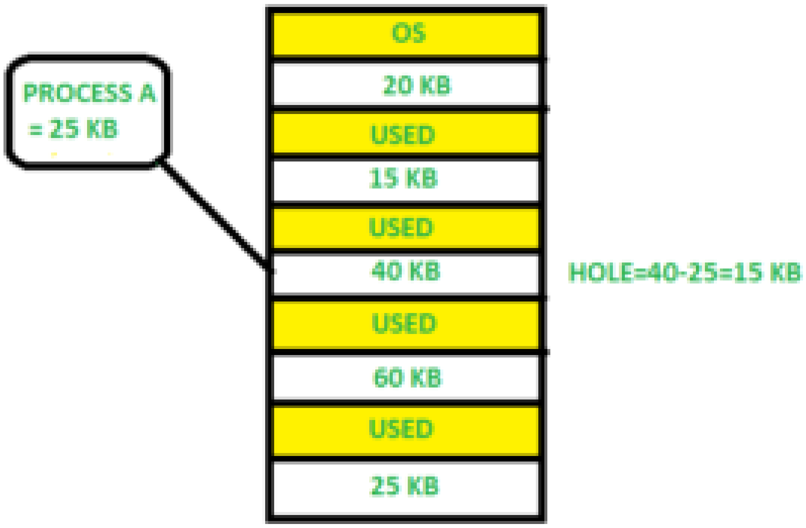
A. First Fit

B. Best Fit

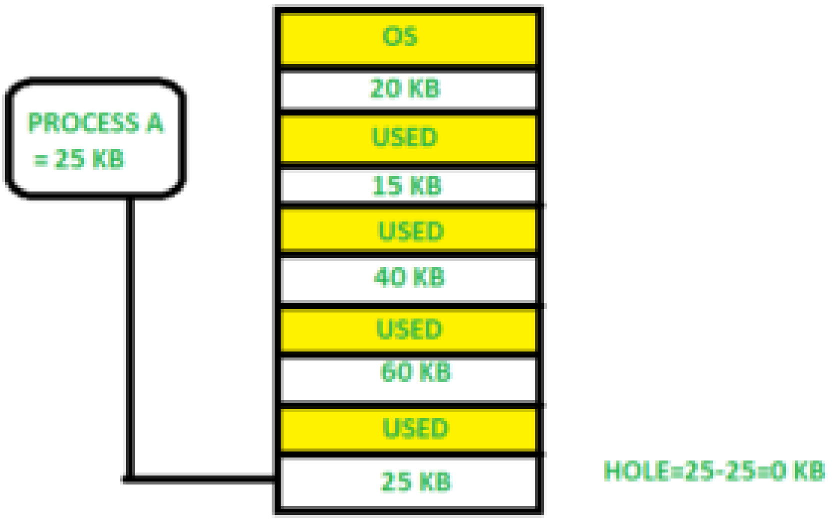
C. Worst Fit

D. Next Fit

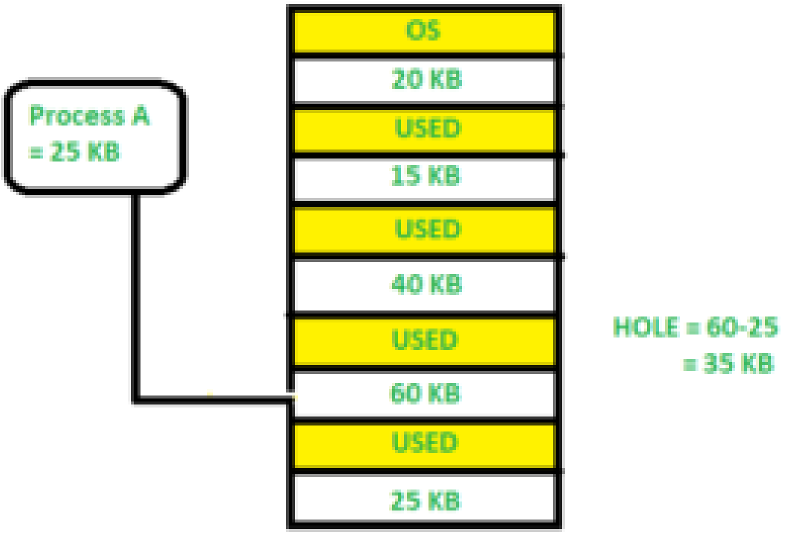
1. First Fit: In the first fit, the partition is allocated which is the first sufficient block from the top of Main Memory. It scans memory from the beginning and chooses the first available block that is large enough. Thus it allocates the first hole that is large enough.



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.



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.



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. For example, see the below exercise.

Exercise: Consider the requests from processes in given order 300K, 25K, 125K, and 50K. Let there be two blocks of memory available of size 300K, 25K, 125K, 50K followed by a block size 350K.

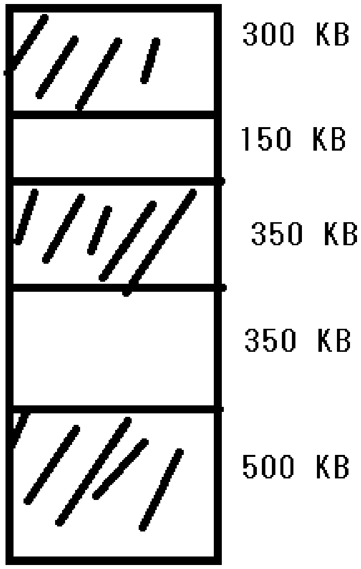
Which of the following partition allocation schemes can satisfy the above requests?

A) Best fit but not first fit.

B) First fit but not best fit.

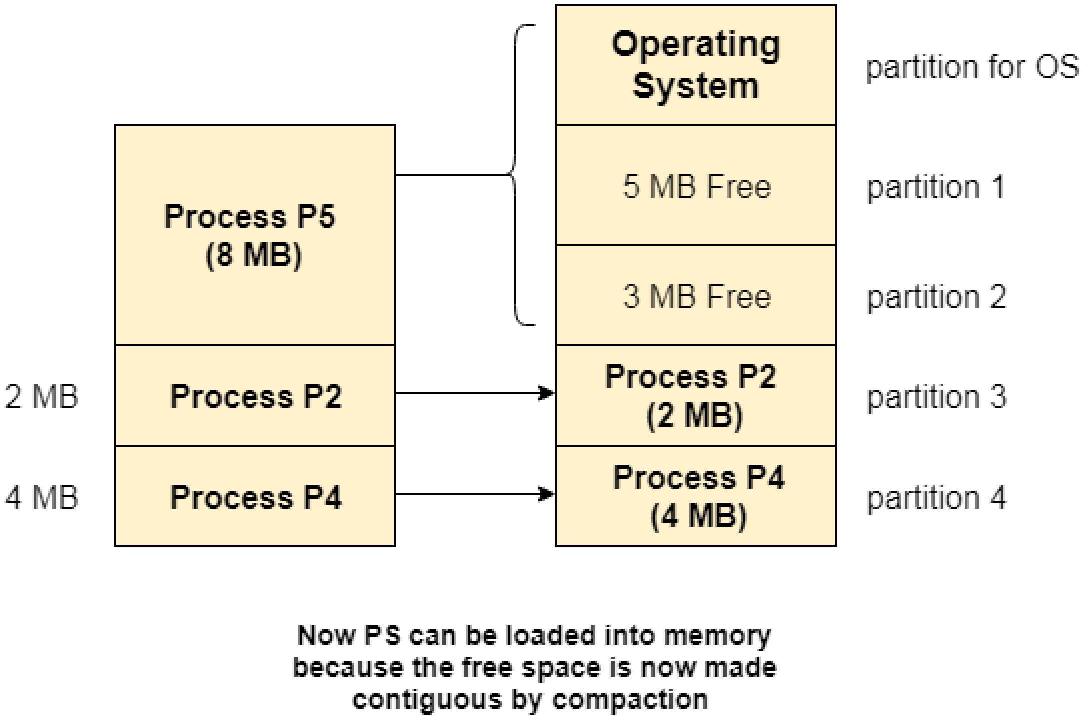
C) Both First fit & Best fit.

D) neither first fit nor best fit.



Compactions.

- Compaction is a process in which the free space is collected in a large memory chunk to make some space available for processes.
- In memory management, swapping creates multiple fragments in the memory because of the processes moving in and out.
- Compaction refers to combining all the empty spaces together and processes.
- Compaction helps to solve the problem of fragmentation, but it requires too much of CPU time.
- It moves all the occupied areas of store to one end and leaves one large free space for incoming jobs, instead of numerous small ones.
- In compaction, the system also maintains relocation information and it must be performed on each new allocation of job to the memory or completion of job from memory.



Compaction

As shown in the image above, the process P5, which could not be loaded into the memory due to the lack of contiguous space, can be loaded now in the memory since the free partitions are made contiguous.

Problem with Compaction

The efficiency of the system is decreased in the case of compaction due to the fact that all the free spaces will be transferred from several places to a single place.

Huge amount of time is invested for this procedure and the CPU will remain idle for all this time. Despite of the fact that the compaction avoids external fragmentation, it makes system inefficient.

Paging

Paging permits a program to allocate noncontiguous blocks of memory. The OS divide programs into pages which are blocks of small and fixed size. Then, it divides the physical memory into frames which are blocks of size equal to page size.