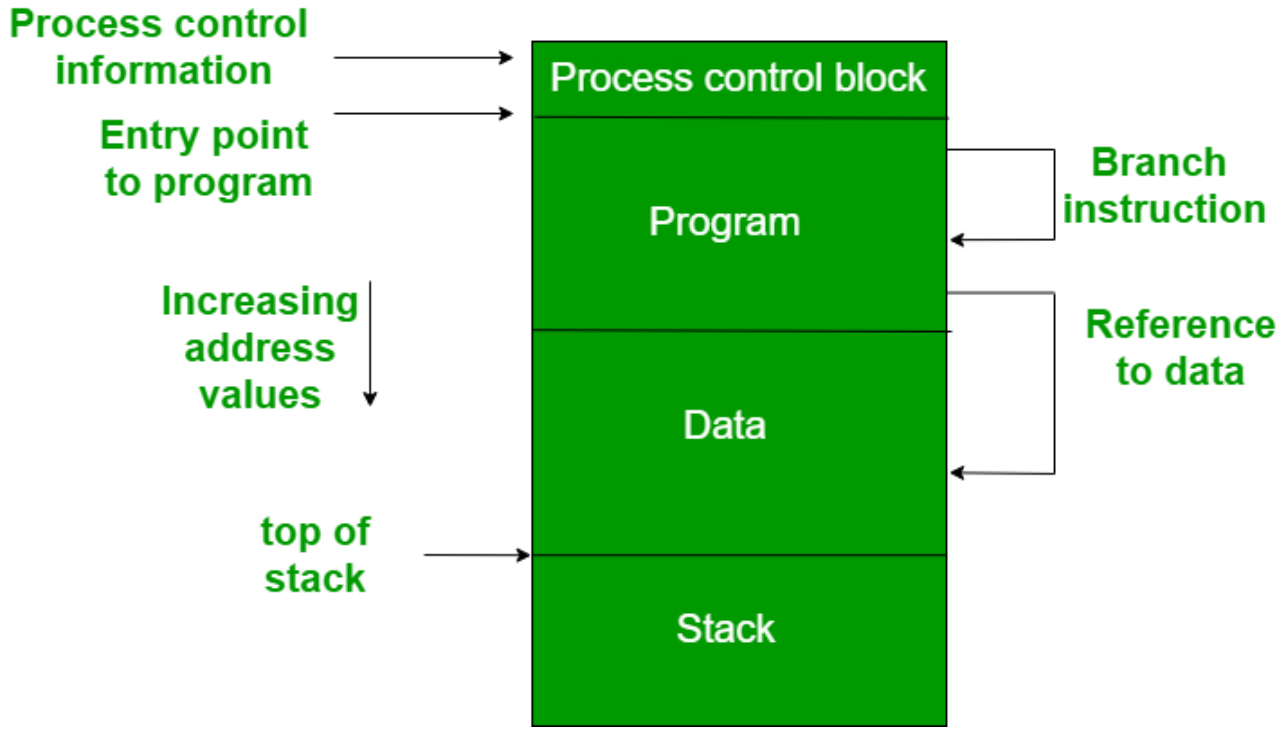
Requirements of Memory Management System

Memory management keeps track of the status of each memory location, whether it is allocated or free. It allocates the memory dynamically to the programs at their request and frees it for reuse when it is no longer needed. Memory management meant to satisfy some requirements that we should keep in mind.

These Requirements of memory management are:

1. **Relocation –** The available memory is generally shared among a 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 this program. Swapping the active processes in and out of the main memory enables the operating system to 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.



1. The figure depicts a process image. The process image is occupying a continuous region of main memory. 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. Data reference instructions contain the address of byte or word of data referenced.

2. **Protection –** There is always a danger when we have multiple programs at the same time as one program may write to the address space of another program. So every process must be protected against unwanted interference when other process tries to write in a process whether accidental or incidental. Between relocation and protection requirement a trade-off occurs as the satisfaction of relocation requirement increases the difficulty of satisfying the protection requirement.

   Prediction of the location of a program in main memory is not possible, that's why it is impossible to check the absolute address at compile time to assure protection. Most of the programming language allows the dynamic calculation of address at run time. The memory protection requirement must be satisfied by the processor rather than the operating system because the operating system can hardly control a process when it occupies the processor. Thus it is possible to check the validity of memory references.
3. **Sharing –** A protection mechanism must have to allow several processes to access the same portion of main memory. Allowing each processes access to the same copy of the program rather than have their own separate copy has an advantage.

   For example, multiple processes may use the same system file and it is natural to load one copy of the file in main memory and let it shared by those processes. It is the task of Memory management to allow controlled access to the shared areas of memory without compromising the protection. Mechanisms are used to support relocation supported sharing capabilities.
4. **Logical organization –** Main memory is organized as linear or it can be a one-dimensional address space which consists of a sequence of bytes or words. Most of the programs can be organized into modules, some of those are unmodifiable (read-only, execute only) and some of those contain data that can be modified. To effectively deal with a user program, the operating system and computer hardware must support a basic module to provide the required protection and sharing. It has the following advantages:
   - Modules are written and compiled independently and all the references from one module to another module are resolved by `the system at run time.
   - Different modules are provided with different degrees of protection.
   - There are mechanisms by which modules can be shared among processes. Sharing can be provided on a module level that lets the user specify the sharing that is desired.

5. **Physical organization –** The structure of computer memory has two levels referred to as main memory and secondary memory. Main memory is relatively very fast and costly as compared to the secondary memory. Main memory is volatile. Thus secondary memory is provided for storage of data on a long-term basis while the main memory holds currently used programs. The major system concern between main memory and secondary memory is the flow of information and it is impractical for programmers to understand this for two reasons:
    - The programmer may engage in a practice known as overlaying when the main memory available for a program and its data may be insufficient. It allows different modules to be assigned to the same region of memory. One disadvantage is that it is time-consuming for the programmer.
    - In a multiprogramming environment, the programmer does not know how much space will be available at the time of coding and where that space will be located inside the memory.

Fixed (or static) Partitioning in Operating System

Fixed partitioning, also known as static partitioning, is one of the earliest memory management techniques used in operating systems. In this method, the main memory is divided into a fixed number of partitions at system startup, and each partition is allocated to a process. These partitions remain unchanged throughout system operation, ensuring a simple, predictable memory allocation process. Despite its simplicity, fixed partitioning has several limitations, such as internal fragmentation and inflexible handling of varying process sizes. This article delves into the advantages, disadvantages, and applications of fixed partitioning in modern operating systems.

What is Fixed (or static) Partitioning in the Operating System?

Fixed (or static) partitioning is one of the earliest and simplest memory management techniques used in operating systems. It involves dividing the main memory into a fixed number of partitions at system startup, with each partition being assigned to a process. These partitions remain unchanged throughout the system's operation, providing each process with a designated memory space. This method was widely used in early operating systems and remains relevant in specific contexts like embedded systems and real-time applications. However, while fixed partitioning is simple to implement, it has significant limitations, including inefficiencies caused by internal fragmentation.

1. In fixed partitioning, the memory is divided into fixed-size chunks, with each chunk being reserved for a specific process. When a process requests memory, the operating system assigns it to the appropriate partition. Each partition is of the same size, and the memory allocation is done at system boot time.

2. Fixed partitioning has several advantages over other memory allocation techniques. First, it is simple and easy to implement. Second, it is predictable, meaning the operating system can ensure a minimum amount of memory for each process. Third, it can prevent processes from interfering with each other's memory space, improving the security and stability of the system.
3. However, fixed partitioning also has some disadvantages. It can lead to internal fragmentation, where memory in a partition remains unused. This can happen when the process's memory requirements are smaller than the partition size, leaving some memory unused. Additionally, fixed partitioning limits the number of processes that can run concurrently, as each process requires a dedicated partition.

Overall, fixed partitioning is a useful memory allocation technique in situations where the number of processes is fixed, and the memory requirements for each process are known in advance. It is commonly used in embedded systems, real-time systems, and systems with limited memory resources.

In operating systems, Memory Management is the function responsible for allocating and managing a computer's main memory. Memory Management function keeps track of the status of each memory location, either allocated or free to ensure effective and efficient use of Primary Memory.

There are two Memory Management Techniques:

1. **Contiguous**
2. **Non-Contiguous**

Contiguous Memory Allocation:

In contiguous memory allocation, each process is assigned a single continuous block of memory in the main memory. The entire process is loaded into one contiguous memory region.

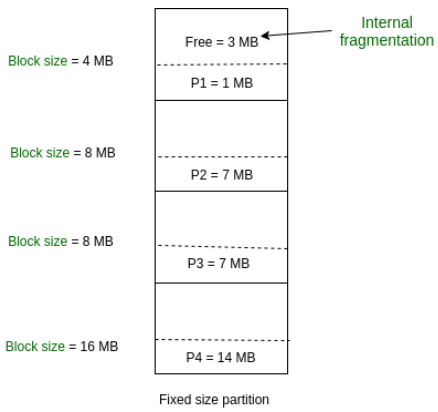
In Contiguous Technique, executing process must be loaded entirely in the main memory.

Contiguous Technique can be divided into:

- Fixed (or static) partitioning
- Variable (or dynamic) partitioning

  Fixed Partitioning:

  This is the oldest and simplest technique used to put more than one process in the main memory. In this partitioning, the number of partitions (non-overlapping) in RAM is **fixed but the size** of each partition may or **may not be the same**. As it is a **contiguous** allocation,



hence no spanning is allowed. Here partitions are made before execution or during system configure.

As illustrated in above figure, first process is only consuming 1MB out of 4MB in the main memory.
Hence,Internal Fragmentation in first block is (4-1) = 3MB.
Sum of Internal Fragmentation in every block = (4-1)+(8-7)+(8-7)+(16-14)= 3+1+1+2 = 7MB.

Suppose process P5 of size 7MB comes. But this process cannot be accommodated in spite of available free space because of contiguous allocation (as spanning is not allowed). Hence, 7MB becomes part of External Fragmentation.

Advantages of Fixed Partitioning

- **Easy to implement:** The algorithms required are simple and straightforward.
- **Low overhead:** Requires minimal system resources to manage, ideal for resource-constrained systems.
- **Predictable**: Memory allocation is predictable, with each process receiving a fixed partition.
- **No external fragmentation:** Since the memory is divided into fixed partitions and no spanning is allowed, external fragmentation is avoided.
- **Suitable for systems with a fixed number of processes:** Ideal for systems where the number of processes and their memory requirements are known in advance.
- **Prevents process interference:** Ensures that processes do not interfere with each other's memory, improving system stability.
- **Efficient memory use:** Particularly in systems with fixed, known processes and batch processing scenarios.
- **Good for batch processing:** Works well in environments where the number of processes remains constant over time.
- **Better control over memory allocation:** The operating system has clear control over how memory is allocated and managed.
- **Easy to debug:** Fixed Partitioning is easy to debug since the size and location of each process are predetermined.

Disadvantages of Fixed Partitioning

1. **Internal Fragmentation:**Main memory use is inefficient. Any program, no matter how small, occupies an entire partition. This can cause internal fragmentation.

2. **Limit process size:**Process of size greater than the size of the partition in Main Memory cannot be accommodated. The partition size cannot be varied according to the size of the incoming process size. Hence, the process size of 32MB in the above-stated example is invalid.

3. **Limitation on Degree of Multiprogramming:** Partitions in Main Memory are made before execution or during system configure. Main Memory is divided into a fixed number of partitions. Suppose if there arepartitions in RAM andare the number of processes, thenn2<=n1condition must be fulfilled. Number of processes greater than the number of partitions in RAM is invalid in Fixed Partitioning.

Difference Between Fragmentation and Segmentation in OS

Memory management is an essential part of the operating system that concerns itself with the allocation, control, and release of memory resources. This is because every application that is executed will require some space for its data, code, and execution stacks, which also creates the need for appropriate strategies to address. Two main aspects of the concepts that have a primary influence on this process are considered to be fragmentation and segmentation. Fragmentation and segmentation are key terms that one should learn the meaning of so as to comprehend how operating systems strive to make the best use of memory while ensuring systems are still functional. This article provides a deeper understanding of both concepts.

What is Fragmentation?

Fragmentation, as the name suggests, is a process in which free memory space is broken into little pieces. In this, memory blocks cannot be allocated to processes due to their small size and such blocks remain unused. It usually occurs in dynamic memory allocation systems when many of the free blocks are too small to satisfy any request.

Example

- For instance, if a program requests 10 MB, the OS will accept such a request and allocate 12 MB because of rounding off to the nearest allocation unit. When the program terminates, this block becomes a “hole” if not all 12 MB are reused efficiently.

Types of Fragmentation

- **External Fragmentation:** occurs every time a part of free memory is broken into tiny regions and is distributed around the system which makes it difficult to locate a consecutive region for new allocations.
- **Internal Fragmentation:** occurs when a fixed block memory is allocated to a process yet a fraction of it is required leading to space wastage in the blocks.

Advantages

- Does not require a lot of work and there is easy implementation of dynamic memory allocation.
- Variable-size allocation is possible hence the size of blocks does not have to be very rigid.

Disadvantages

- If left unmanaged for long periods, this process can cause excessive memory wastage.
- Sometimes, the insertion of compaction has to be employed in order to have the most efficient use of space, and this can be quite expensive in terms of the CPU.

What is Segmentation?

Segmentation, as name suggests, is basically a memory management technique that supports user's view of memory and is also known as non-contiguous memory allocation technique. In this, each process is divided into number of segments and detail about each segment can be stored in table that is known as segment table. It is basically a process that creates variable-sized address spaces in computer storage for related data. Segments are not fixed in size.

Example

- A program can be split into three segments: code, data, and stack. Each segment can be independently allocated in memory, and the operating system keeps track of these segments using a segment table.

Advantages

- This provides both a more rational and more practical way of handling memory.
- Can also lead to external fragmentation principally if segments are allocated and de-allocated at almost the same time.

Disadvantages

- Hard to implement as it requires the use of segment tables, and a segment management algorithm.
- May also lead to external fragmentation if segments are allocated and deallocated frequently.

Difference Between Fragmentation and Segmentation in OS

Fragmentation	Segmentation
In this, storage space is used inefficiently that in turn reduce capacity and performance.	In this, memory is divided into variable size parts usually known as segments.
Types of fragmentation includes internal and external fragmentation.	Types of segmentation includes virtual memory and simple segmentation.
Its main purpose is to help operating system use the available space on storage device.	Its main purpose is to give user's view of process.
It reduces efficiency in memory management.	It simply allows for better efficiency in memory management.
In this, memory blocks are not used i.e., it remains unused.	It usually works a memory management technique to execute processes.

Fragmentation	Segmentation
It is generally associated with IP.	It is generally associated with TCP.
It is an unwanted problem that causes wastage of memory and inflexibility.	Its advantages include less overhead, larger segment size than actual page size, no internal fragmentation, etc.