Table of Contents

[MEMORY MANEGEMENT 2](#_Toc134654678)

[NEED FOR MEMORY MANAGEMENT 2](#_Toc134654679)

[MEMORY ALLOCATION 2](#_Toc134654680)

[SWAPING 2](#_Toc134654681)

[LOGICAL VS PHYSICAL ADDRESSES 3](#_Toc134654682)

[MEMORY ADDRESSES 3](#_Toc134654683)

[ADDRESS BINDING 3](#_Toc134654684)

[CONTIGUOUS MEMORY MANAGEMENT 5](#_Toc134654685)

[1. Single contiguous memory management schemes 5](#_Toc134654686)

[2. PARTITIONING or MULTIPLE PARTITIONING 5](#_Toc134654687)

[Fixed Partitioning 5](#_Toc134654688)

[Dynamic Partitioning 6](#_Toc134654689)

[Fragmentation 6](#_Toc134654690)

[METHODS TO ELIMINATE EXTERNAL FRAGMENTATION 7](#_Toc134654691)

[COMPACTION 7](#_Toc134654692)

[PAGING 7](#_Toc134654693)

[SEGMENTATION 7](#_Toc134654694)

[SEGMENTATION TABLE 8](#_Toc134654695)

[Advantages of Segmentation 8](#_Toc134654696)

[Disadvantages of Segmentation 8](#_Toc134654697)

# MEMORY MANEGEMENT

Memory management is the way computers manage their main memory by controlling and coordinating the allocation of memory blocks. It ensures that the operating system, applications, and other running processes have the memory they need to function properly. This technique keeps track of every memory location and whether it is allocated to a process or free. The memory management technique decides which process gets memory and how much memory can be allocated to it. It also keeps track of the amount of memory allocated to each process and updates the status whenever some memory gets freed or unallocated.

## NEED FOR MEMORY MANAGEMENT

Memory management is the process of managing and allocating a computer's main memory to ensure that the operating system, applications, and running processes have the memory they need to operate. It keeps track of which memory locations are in use and which ones are free, decides which processes get memory and how much, and protects processes from interfering with each other. This helps to utilize memory to its fullest extent, allocate space to different application routines, and keep track of memory status.

## MEMORY ALLOCATION

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

Here, main memory is divided into two types of partitions

* Low Memory – Operating system resides in this type of memory.
* High Memory– User processes are held in high memory.

## SWAPING

Swapping is a technique used when there is not enough memory to hold all the active processes in a computer's main memory. In this technique, a process that is not currently being used is temporarily moved from the main memory to the hard disk, where it is stored until it is needed again. This frees up memory for other active processes to run. When the swapped-out process is needed again, it is swapped back into the main memory for execution. The hard disk acts as a backing store for the swapped-out processes.

## LOGICAL VS PHYSICAL ADDRESSES

Logical address and physical address are two types of addresses used in computer memory. Logical address is generated by the CPU when a program is executed, while physical address refers to the actual location of data in memory. The collection of all logical addresses generated by the CPU is called the logical address space.

## MEMORY ADDRESSES

Memory addresses are used to refer to locations in computer memory. Symbolic addresses are used in the source code and include variable names, constants, and instruction labels. Relative addresses, or relocatable addresses, are generated by the compiler by converting symbolic addresses into relative addresses during compilation. Physical addresses are generated by the loader when a program is loaded into main memory.

## ADDRESS BINDING

Address binding is the process of translating virtual or logical memory addresses into actual physical memory addresses in order to execute a program. Logical addresses do not physically exist in memory and are sometimes referred to as virtual addresses, while physical addresses represent actual memory locations. Logical addresses are used as a reference to access physical addresses, which cannot be accessed directly. Address binding can occur in different modes and sequences depending on the system and program being executed.

Address binding in OS can be categorized into three types: compile time, load time, and execution time.

Compile time binding occurs when the memory location for the program is known during compilation. The compiler interacts with the operating system to perform the binding, and the program is allocated memory at execution time.

Load time binding, on the other hand, is used when the memory location for the program is not known at the time of compilation. In this case, the compiler generates relative or relocatable addresses that the loader translates into absolute or physical addresses.

Dynamic binding or execution time binding happens when a process moves from one memory location to another during execution. To handle this situation, special hardware called the memory management unit (MMU) is used. The MMU maintains a relocation register that stores the base address of memory. When the CPU generates the logical address, the MMU combines it with the base address to create a physical address.

# CONTIGUOUS MEMORY MANAGEMENT

Contiguous memory management is a memory allocation scheme in which a program requires a contiguous block of memory to be stored. In this scheme, memory is divided into partitions of fixed sizes, and each partition can contain one program. There are two types of contiguous memory management schemes.

## Single contiguous memory management schemes

The Single contiguous memory management scheme is a basic memory management approach used in early computers. The main memory is divided into two partitions, one for the operating system and the other for user processes. The operating system is usually located in the lower partition. The user process is loaded into the other partition.

## PARTITIONING or MULTIPLE PARTITIONING

In Partitioning or Multiple Partitioning, the available memory is divided into several partitions, and each partition is assigned to a different process.

Each partition can hold one process, and the size of each partition can be fixed or variable.

This scheme allows for efficient use of the main memory as multiple processes can be loaded and executed simultaneously, reducing the waiting time for processes in the ready queue.

### Fixed Partitioning

One method of partitioning is Fixed Partitioning, which divides the main memory into fixed-sized partitions that can hold a single process each. These partitions can be of the same size or different sizes. The degree of multiprogramming is determined by the number of partitions, which is specified at the time of system generation and remains fixed afterward. This scheme is simple to implement, easy to manage and design, but suffers from internal fragmentation and the inflexibility of the number of partitions.

### Dynamic Partitioning

Dynamic Partitioning is a memory management scheme designed to overcome the limitations of the fixed partitioning scheme. In this scheme, each process is allocated only as much memory as required when it is loaded for processing. The partitions used in dynamic partitioning are of variable size, and the number of partitions is not defined at the time of system generation. Although the dynamic partitioning scheme is simple to implement, manage and design, it suffers from external fragmentation. The number of partitions in this scheme is specified at the time of system segmentation, which can cause performance issues.

# Fragmentation

Fragmentation is a problem that can happen in an operating system when a process is loaded and unloaded from memory. This can cause the memory space to become fragmented, meaning that there are small, unused memory blocks that are too small to be assigned to new processes. There are two types of fragmentation: internal and external.

Internal fragmentation happens when the memory allocated to a process is larger than what it actually needs, leaving unused space within the process.

External fragmentation, on the other hand, occurs when there are large enough free memory blocks, but they are not contiguous, making it difficult to allocate them to new processes.

Both types of fragmentation can cause performance issues and inefficiencies in the use of system resources.

# METHODS TO ELIMINATE EXTERNAL FRAGMENTATION

## COMPACTION

Compaction is a technique used to reduce external fragmentation in an OS. It involves shuffling the contents of the memory to place all free memory together in one large block. To make compaction possible, relocation must be dynamic. All free partitions are made contiguous and loaded partitions are brought together. This allows bigger processes to be stored in memory, as free partitions are merged and allocated as per the needs of new processes.

## PAGING

Paging is a memory management technique that allows for the allocation of non-contiguous memory space. It divides main memory into fixed-size blocks called frames, and processes are divided into blocks of the same size called pages. A page table is used to map the virtual addresses used by the CPU to the corresponding physical addresses.

Advantages of paging include an easy-to-use memory management algorithm, no external fragmentation, and easier swapping between equal-sized pages and page frames. However, paging also has disadvantages such as internal fragmentation, additional memory consumption by page tables, and memory reference overhead due to multi-level paging.

## SEGMENTATION

Segmentation is a memory management technique that is similar to paging, but with variable-length segments rather than fixed-size pages. A program's segments include its main function, data structures, and utility functions. The OS maintains a segment map table that contains information about the free memory blocks, their sizes, and their memory locations in the main or virtual memory. Unlike paging, which may divide a single function into different pages that may or may not be loaded into memory at the same time, segmentation divides a process into segments for better efficiency.

### SEGMENTATION TABLE

In segmentation, there is no straightforward connection between logical and physical addresses. To keep track of this information, a table is used which is called a Segment Table. This table maps the two-dimensional logical address to a one-dimensional physical address. Each entry in the table contains two pieces of information: the starting physical address of the segment (called the Base Address) and the length of the segment (called the Limit).

Advantages of Segmentation:

- No internal fragmentation.

- Segment Table consumes less space in comparison to Page table in paging.

- Complete module is loaded all at once, improving CPU utilization.

- Users can divide user programs into modules via segmentation, similar to separate processes' codes.

- Segments can be of variable size, allowing for more fine-grained memory allocation.

- Sharing of memory segments between processes is possible for inter-process communication or for sharing code libraries.

- Protection between segments prevents one process from accessing or modifying another process’s memory segment, increasing the security and stability of the system.

Disadvantages of Segmentation:

- External fragmentation may occur as free memory space is broken into little pieces.

- Overhead is associated with keeping a segment table for each activity.

- Access time to retrieve the instruction increases due to the need for two memory accesses, one for the segment table and the other for main memory.

- Segmentation can lead to external fragmentation as memory becomes divided into smaller segments, resulting in wasted memory and decreased performance.

- The use of a segment table can increase overhead and reduce performance.

- Segmentation can be more complex to implement and manage than paging, particularly when managing multiple segments per process.

- The potential for segmentation faults can increase as a result.

#### SUMMARY OF SEGMENTATION

Segmentation is a memory management technique where the process is divided into segments instead of pages. Each segment of a program comprises the program’s main function, data structures, utility functions, etc. The information about processes is maintained in the form of a segment map table by the OS, which maps logical addresses to physical addresses. Segmentation provides advantages such as no internal fragmentation, efficient CPU utilization, flexibility in memory allocation, sharing of memory segments, and protection between segments. However, segmentation also has its drawbacks, including external fragmentation, overhead in maintaining a segment table, increased access time, and complexity in managing multiple segments per process, potentially leading to segmentation faults.