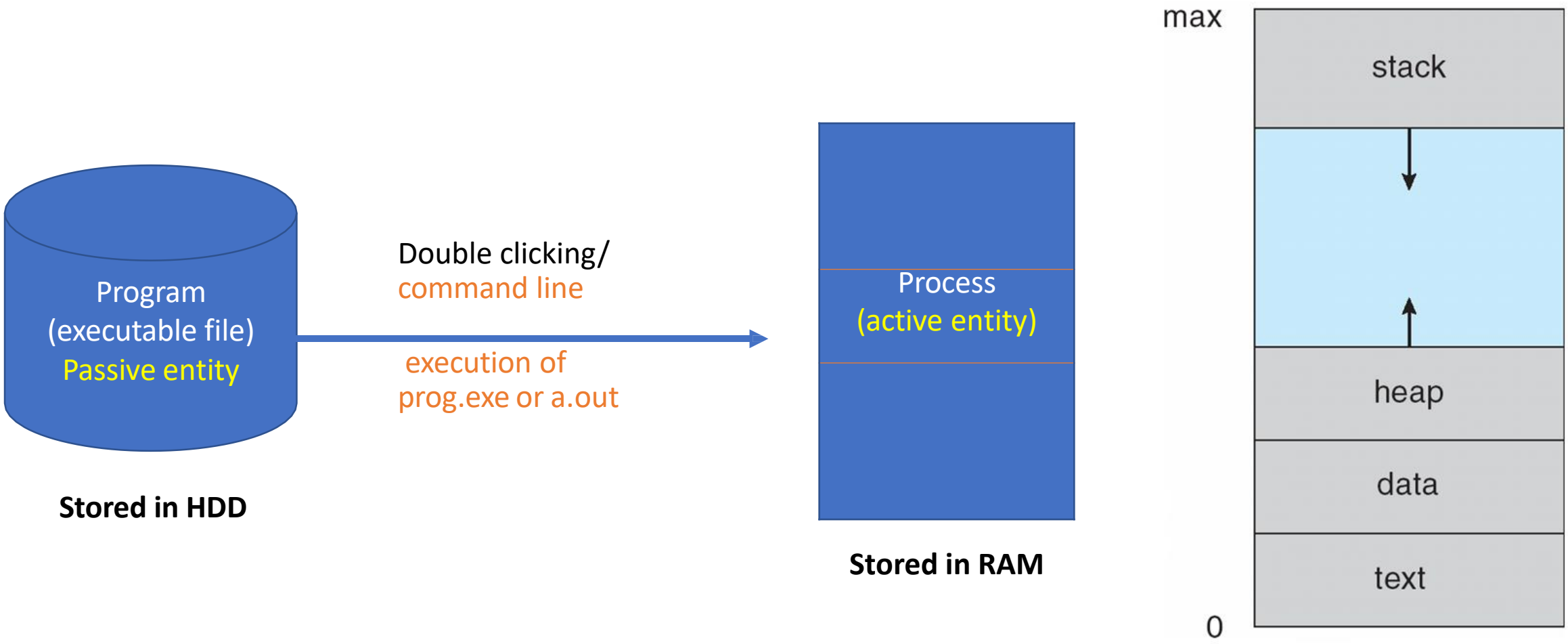
OPERATING SYSTEM: CSET209

PROCESS (PROGRAM IN EXECUTION)

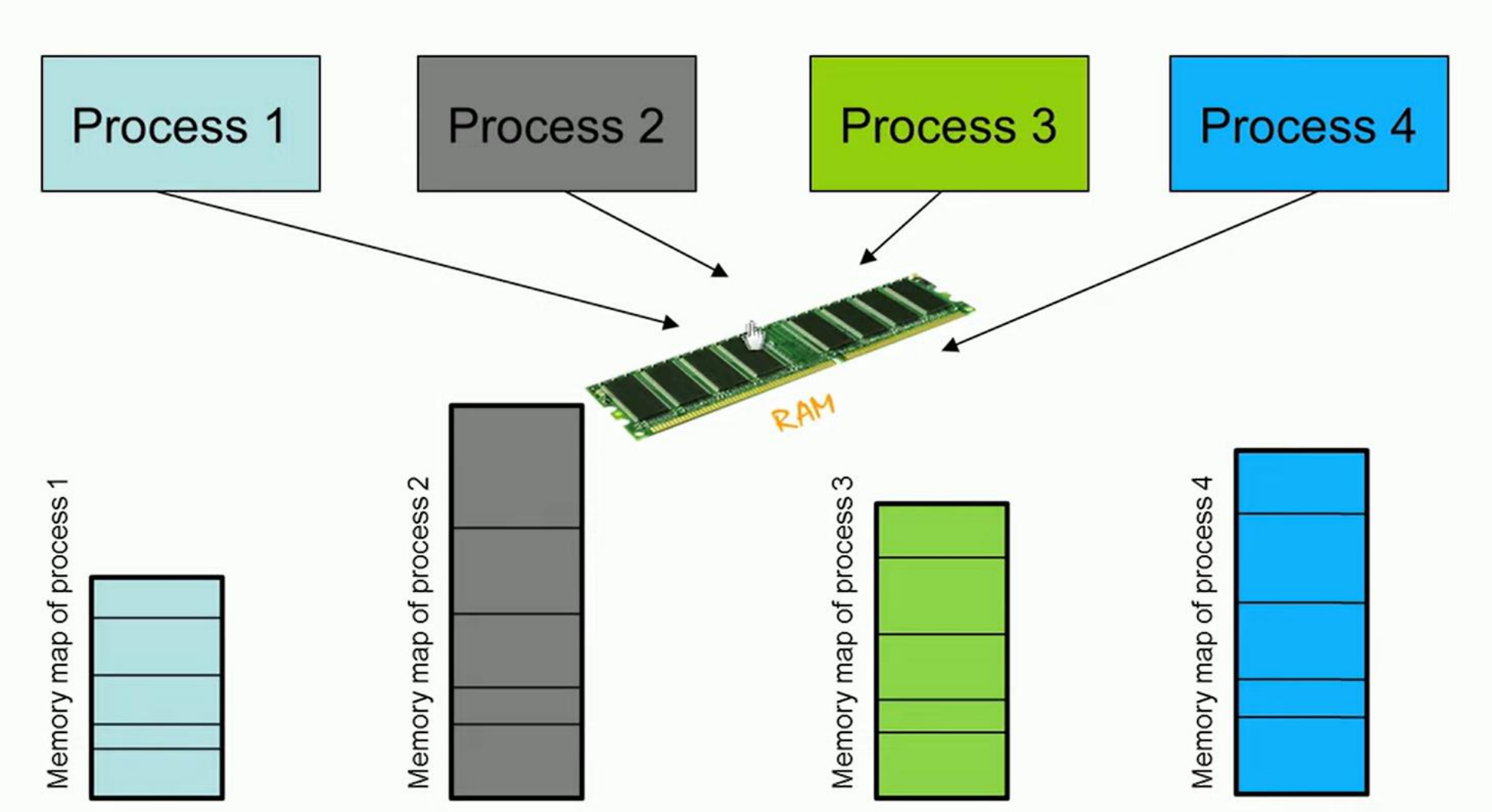


Process: Program under execution called process

Why memory management

- Allocate and de-allocate memory before and after process execution.
- To keep track of used memory space by processes.
- To minimize fragmentation issues.
- To proper utilization of main memory.
- To maintain data integrity while executing of process.

PROCESS SHARES RAM



Static and Dynamic Loading

Loading a process into the main memory is done by a loader. There are two different types of loading :

- **Static Loading:** Static Loading is basically loading the entire program into a fixed address. It requires more memory space.
- **Dynamic Loading:** The entire program and all data of a process must be in physical memory for the process to execute. So, the size of a process is limited to the size of physical memory. To gain proper memory utilization, dynamic loading is used. In dynamic loading, a routine is not loaded until it is called. All routines are residing on disk in a relocatable load format. One of the advantages of dynamic loading is that the unused routine is never loaded. This loading is useful when a large amount of code is needed to handle it efficiently.

Static and Dynamic Linking

To perform a linking task a linker is used. A linker is a program that takes one or more object files generated by a compiler and combines them into a single executable file.

- **Static Linking:** In static linking, the linker combines all necessary program modules into a single executable program. So there is no runtime dependency. Some operating systems support only static linking, in which system language libraries are treated like any other object module.
- **Dynamic Linking:** The basic concept of dynamic linking is similar to dynamic loading. In dynamic linking, "Stub" is included for each appropriate library routine reference. A stub is a small piece of code. When the stub is executed, it checks whether the needed routine is already in memory or not. If not available then the program loads the routine into memory.

Logical and Physical Address Space

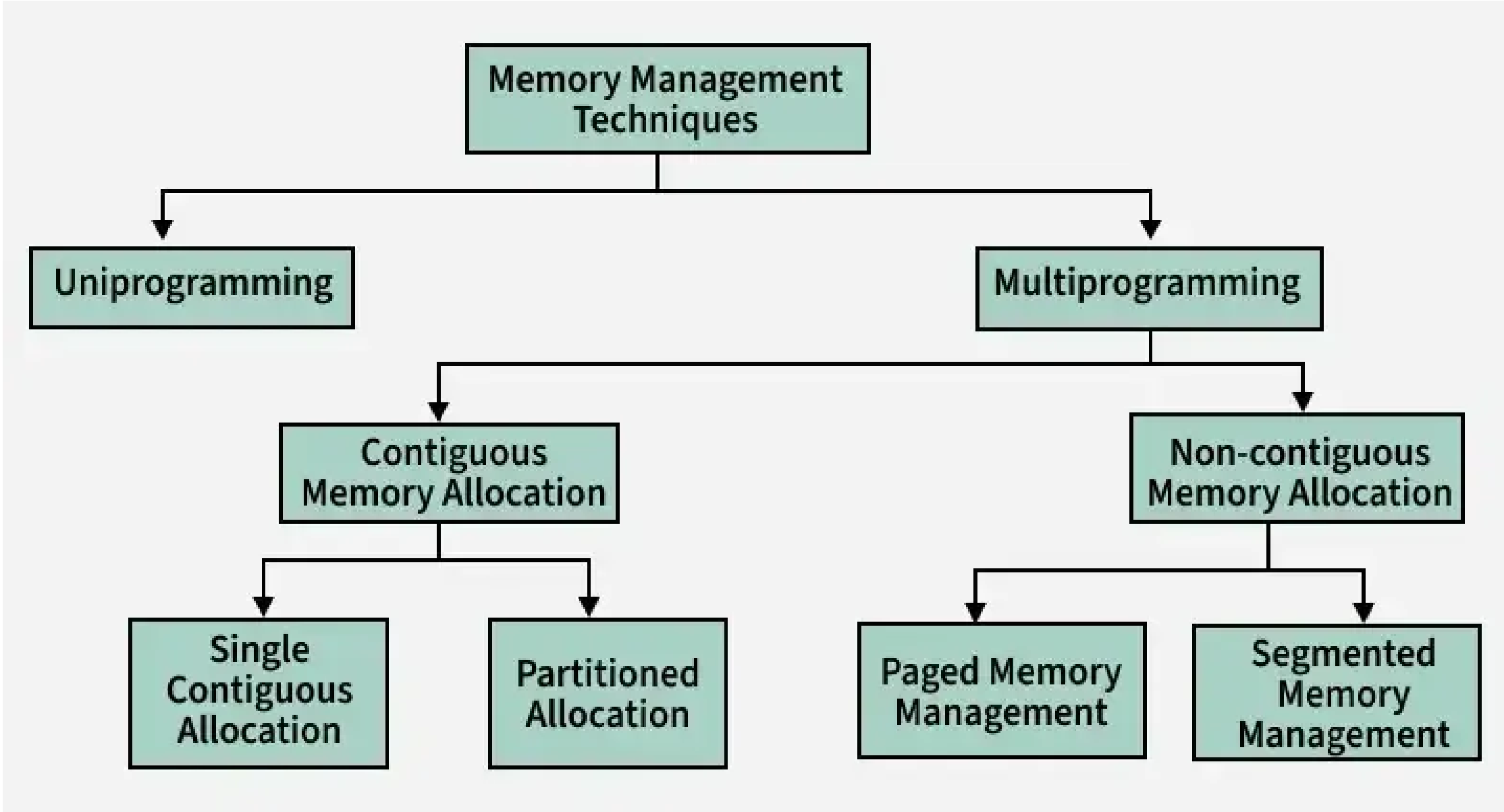
- **Logical Address Space:** An address generated by the CPU is known as a “Logical Address”. It is also known as a Virtual address. Logical address space can be defined as the size of the process. A logical address can be changed.
- **Physical Address Space:** An address seen by the memory unit (i.e. the one loaded into the memory address register of the memory) is commonly known as a “Physical Address”. A Physical address is also known as a Real address. The set of all physical addresses corresponding to these logical addresses is known as Physical address space. A physical address is computed by MMU. The run-time mapping from virtual to physical addresses is done by a hardware device Memory Management Unit(MMU). The physical address always remains constant.

Address binding

The mapping of data and computer instructions to actual memory locations is known as address binding.

Address Binding is divided into three types as follows:

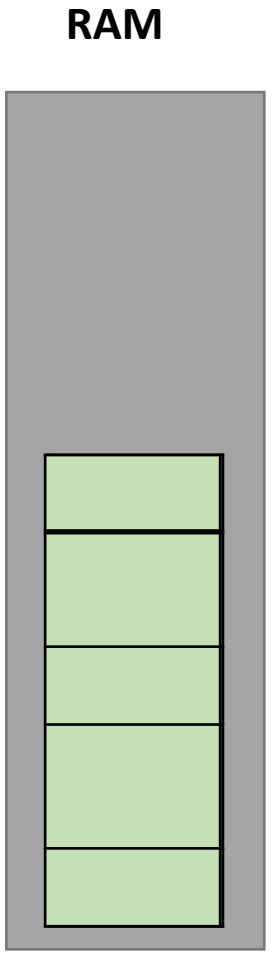
- Compile-time Address Binding
- Load time Address Binding
- Execution time Address Binding


Memory Management Techniques
Uniprogramming
Multiprogramming
Contiguous Memory Allocation
Non-contiguous Memory Allocation
Single Contiguous Allocation
Partitioned Allocation
Paged Memory Management
Segmented Memory Management

Single Contiguous Model

- No Sharing
  - One process occupies RAM at a time i.e. when one process completes, then RAM is allocated to another process.
  - The program occupies a single contiguous block of storage locations, i.e., a set of memory locations with consecutive addresses.

- Limitations
  - Only sequential execution of process will take place. (i.e. Multiprogramming is not supported)
  - Process size is limited by RAM size.



Multiple Partitioning Model

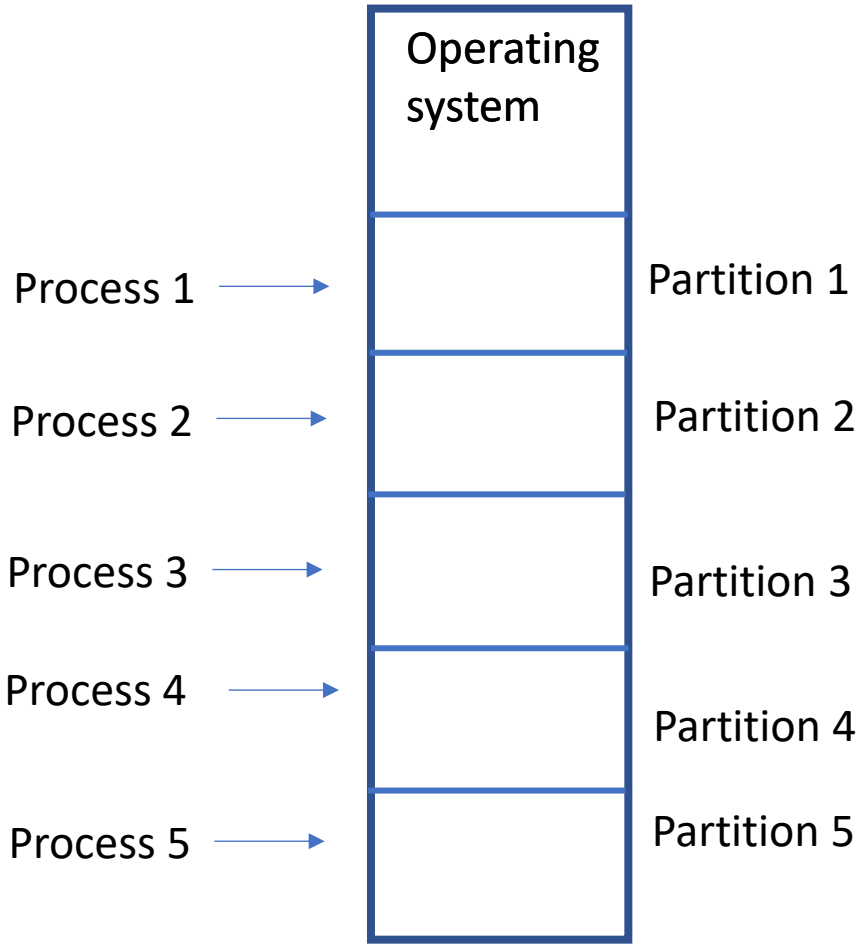
- The single Contiguous memory management scheme is inefficient as it limits computers to execute only one program at a time resulting in wastage in memory space and CPU time.
- In Multiple Partitioning model, the operating system divides the available main memory into multiple parts to load multiple processes into the main memory. Thus multiple processes can reside in the main memory simultaneously.
- The multiple partitioning schemes can be of two types:
  - **Fixed Partitioning**
  - **Dynamic Partitioning**

Fixed Partitioning

Main memory is **divided into partitions of equal or different sizes**.

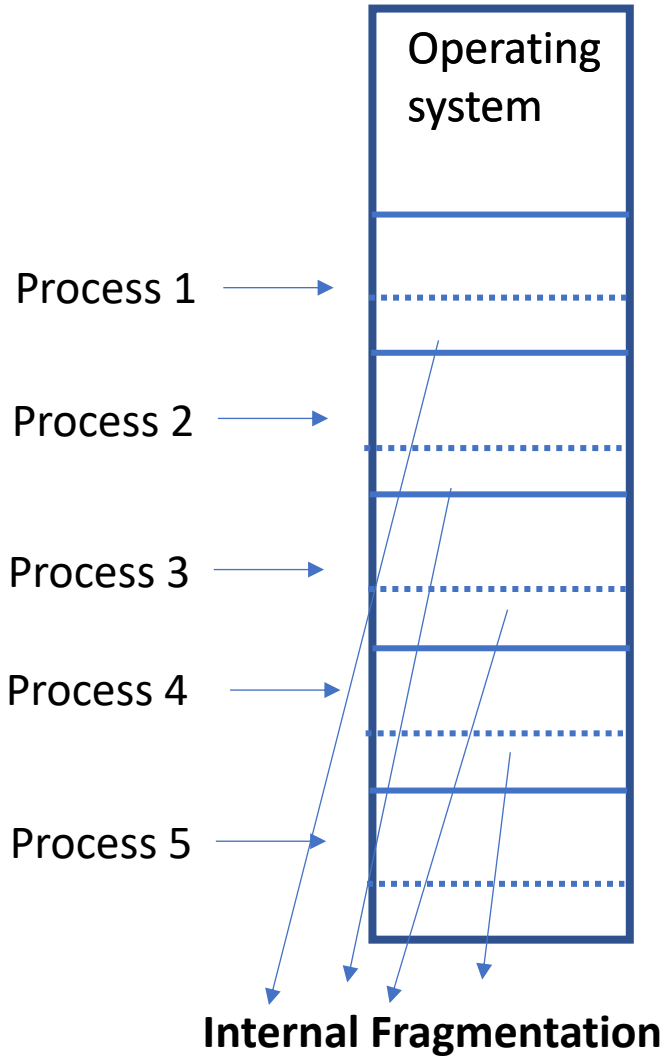
The operating system always resides in the first partition while the other partitions can be used to store user processes.

The memory is assigned to the processes in contiguous way.



Fixed Partitioning: Limitations

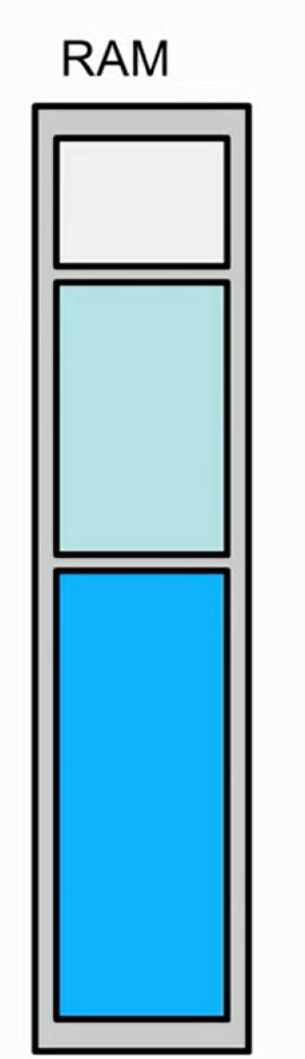
- **Internal Fragmentation**:
  - Occurs when a process occupies less memory than the size of the partition it has been allocated. Unused space within the partition that cannot be used to accommodate other processes. Over time, this can lead to inefficient use of memory, as more and more partitions become partially filled with wasted space.
- **Limitation on the size of the process**
- **Degree of multiprogramming is less**



Dynamic Partitioning

New processes are allocated memory as long as sufficient contiguous space is available.

Hence, it supports multiprogramming.



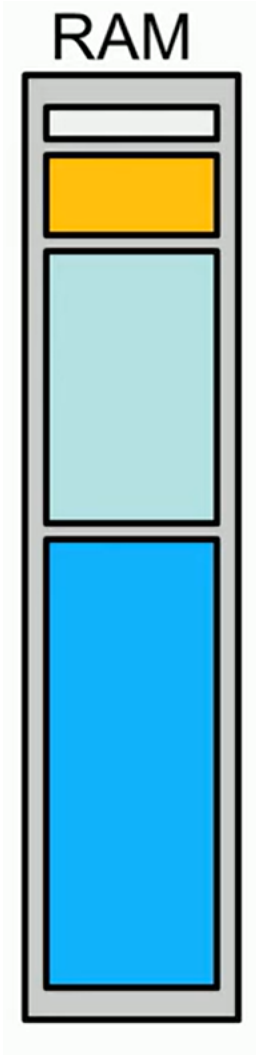
Partition table

Memory Address	Size	Process	Usage
0x0	120k	4	In use
120k	60k	1	In use
180k	30k		Free

Dynamic Partitioning

New processes are allocated memory as long as sufficient contiguous space is available.

Process 5 of 20k has started and is allocated RAM.

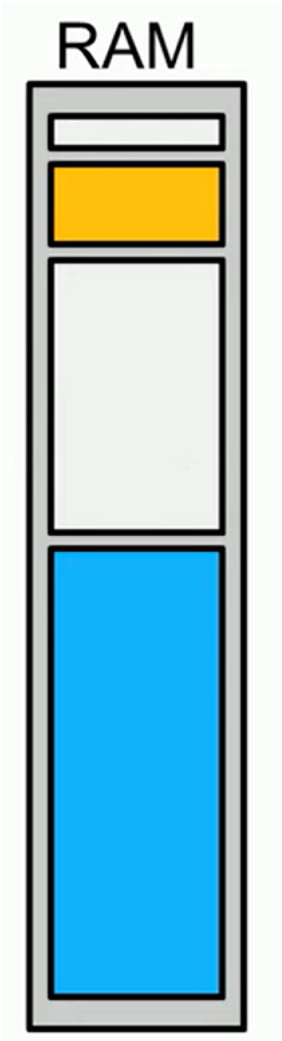


Partition table

Memory Address	Size	Process	Usage
0x0	120k	4	In use
120k	60k	1	In use
180k	20k	5	In Use
200k	10k		Free

Dynamic Partitioning

Process 1 has completed and thus deallocated RAM.



Partition table

Memory Address	Size	Process	Usage
0x0	120k	4	In use
120k	60k		Free
180k	20k	5	In Use
200k	10k		Free

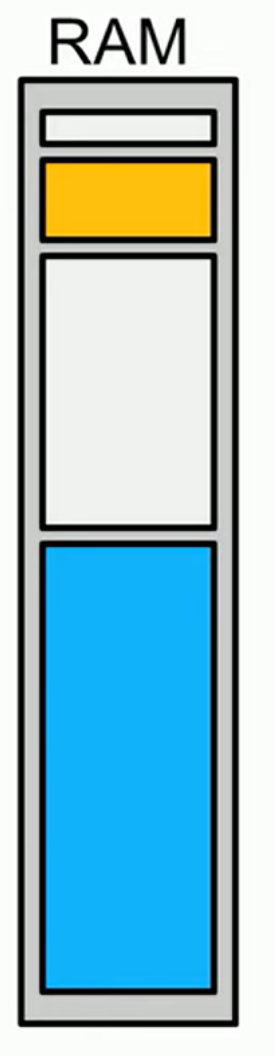
Dynamic Partitioning
(Limitation: External Fragmentation)

External Fragmentation: Total memory space is available to satisfy a request but not contiguous. Thus it results in under utilization of RAM.

New process (6) of size 65k cannot start, even though 70k free memory is available

Partition Table

Memory Address	Size	Process	Usage
0x0	120k	4	In use
120k	60k		Free
180k	20k	5	In Use
200k	10k		Free



Dynamic Partitioning
(Limitation: External Fragmentation)

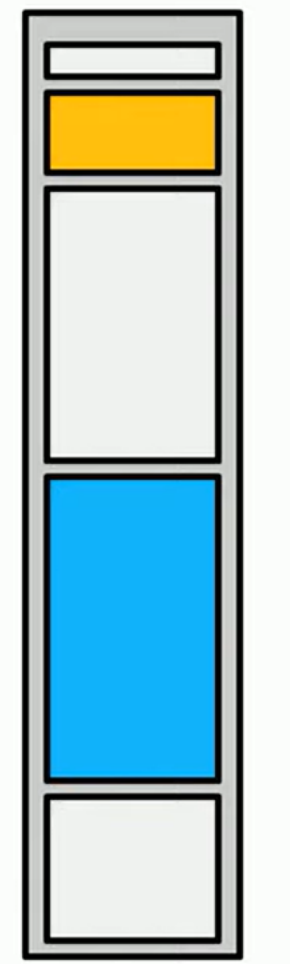
Solution of External Fragmentation is **compaction.**

Compaction: Moving all the processes towards top or bottom to make the available memory to a single continuous place.

Note: it is undesirable to implement. Because it disturb all the running processes in the memory.

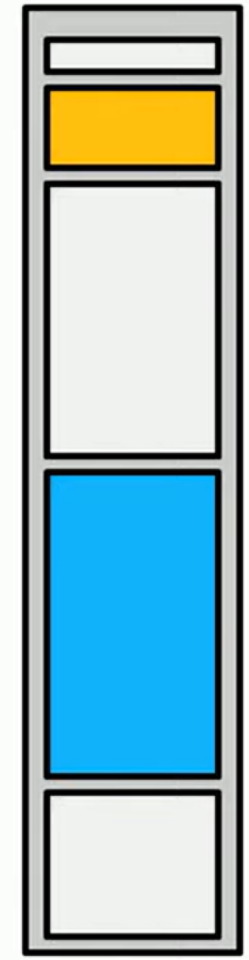
Finding the Right Fit in Dynamic Partitioning

FIRST FIT



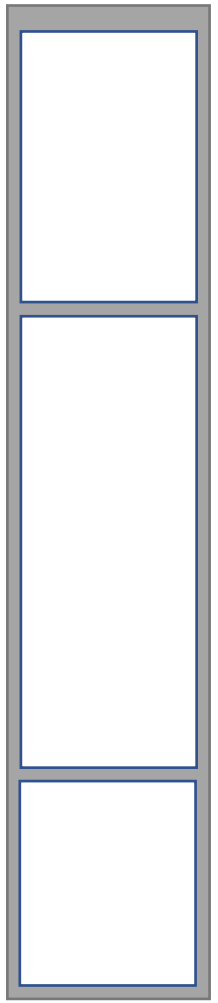
Limitation: Can make Fragmentation worse.

BEST FIT



Limitation: Can degrade the performance

WORST FIT



Limitation: Can degrade the performance

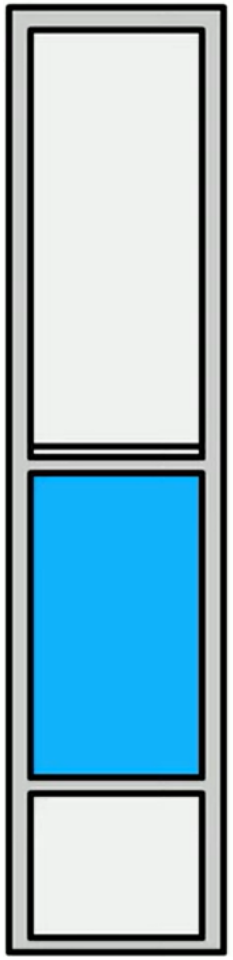
Finding the Right Fit in Dynamic Partitioning

Most commonly used to select a free hole from the set of available holes in dynamic partitioning are as follows.

- **First Fit:** Allocate the first hole that is big enough.
- **Best Fit:** Allocate the smallest hole that is big enough. We must search the entire list, unless the list is ordered by size. This strategy produces the smallest leftover hole.
- **Worst Fit:** Allocate the largest hole. Again, we must search the entire list, unless it is sorted by size. This strategy produces the largest leftover hole, which may be more useful than the smaller leftover hole from a best-fit approach.
- **Next Fit** Next fit is a modified version of ‘first fit’. It begins as first fit to find a free partition but when called next time it starts searching from where it left off, not from the beginning

Merging the Partitions

- When a process completes its execution or is terminated, it needs to removed from the RAM (deallocation).
- Deallocation of memory can have overhead of merging the partitions.



Limitations of Partitioning Model

- Entire memory map of the process needs to be in RAM.
- Fragmentation of memory.
- Performance degradation due to managing partitions and finding a right fit of memory block for a process.

THANK YOU
?