SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A+' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF INFORMATION TECHNOLOGY

23CST202 – Operating Systems

II YEAR - IV SEM

UNIT 3 – MEMORY MANAGEMENT

Syllabus

- UNIT I OVERVIEW AND PROCESS MANAGEMENT 9
- Introduction - Computer System Organization, Architecture, Operation, Process Management – Memory Management – Storage Management – Operating System – Process concept – Process scheduling – Operations on processes – Cooperating processes – Inter process communication. Threads - Multi-threading Models – Threading issues.
- UNIT II PROCESS SCHEDULING AND SYNCHRONIZATION 10
- CPU Scheduling - Scheduling criteria – Scheduling algorithms – Multiple-processor scheduling – Real time scheduling – Algorithm Evaluation. Process Synchronization - The critical-section problem – Synchronization hardware – Semaphores – Classical problems of synchronization. Deadlock - System model – Deadlock characterization – Methods for handling deadlocks – Deadlock prevention – Deadlock avoidance – Deadlock detection – Recovery from deadlock.
- UNIT III MEMORY MANAGEMENT 9
- Memory Management - Background – Swapping – Contiguous memory allocation – Paging – Segmentation – Segmentation with paging. Virtual Memory - Background – Demand paging – Process creation – Page replacement – Allocation of frames – Thrashing.
- UNIT IV FILE SYSTEMS 8
- File concept - Access methods – Directory structure – Files System Mounting – File Sharing – Protection. File System Implementation - Directory implementation – Allocation methods – Free-space management.
- UNIT V I/O SYSTEMS 9
- I/O Systems - I/O Hardware – Application I/O interface – Kernel I/O subsystem – Streams – Performance. Mass-Storage Structure: Disk scheduling – Disk management – Swap-space management – RAID – Disk attachment – Stable storage – Tertiary storage. Case study: Implementation of Distributed File system in Cloud OS / Mobile OS.

- L :45 P:0 T: 45 PERIODS

Memory Management

- Background
- Swapping
- Contiguous Allocation
- Paging
- Segmentation
- Segmentation with Paging

Background

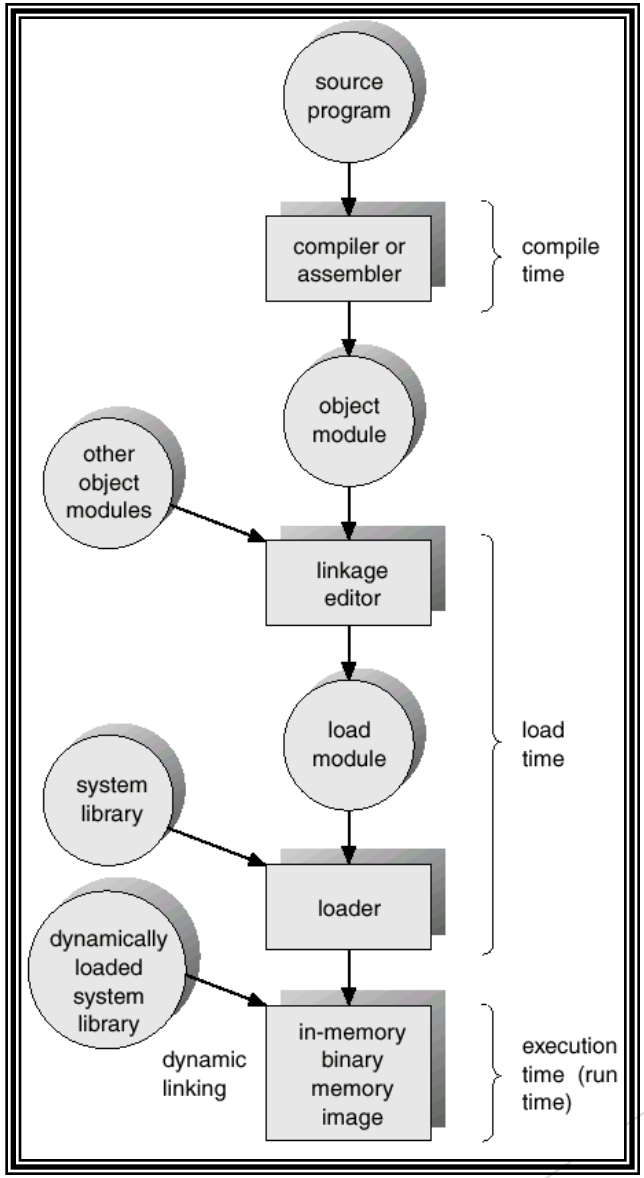
- Program must be brought into memory and placed within a process for it to be run.
- *Input queue* – collection of processes on the disk that are waiting to be brought into memory to run the program.
- User programs go through several steps before being run.

Binding of Instructions and Data to Memory

Address binding of instructions and data to memory addresses can happen at three different stages.

- **Compile time**: If memory location known a priori, absolute code can be generated; must recompile code if starting location changes.
- **Load time**: Must generate *relocatable* code if memory location is not known at compile time.
- **Execution time**: Binding delayed until run time if the process can be moved during its execution from one memory segment to another. Need hardware support for address maps (e.g., *base* and *limit registers*).

Multistep Processing of a User Program



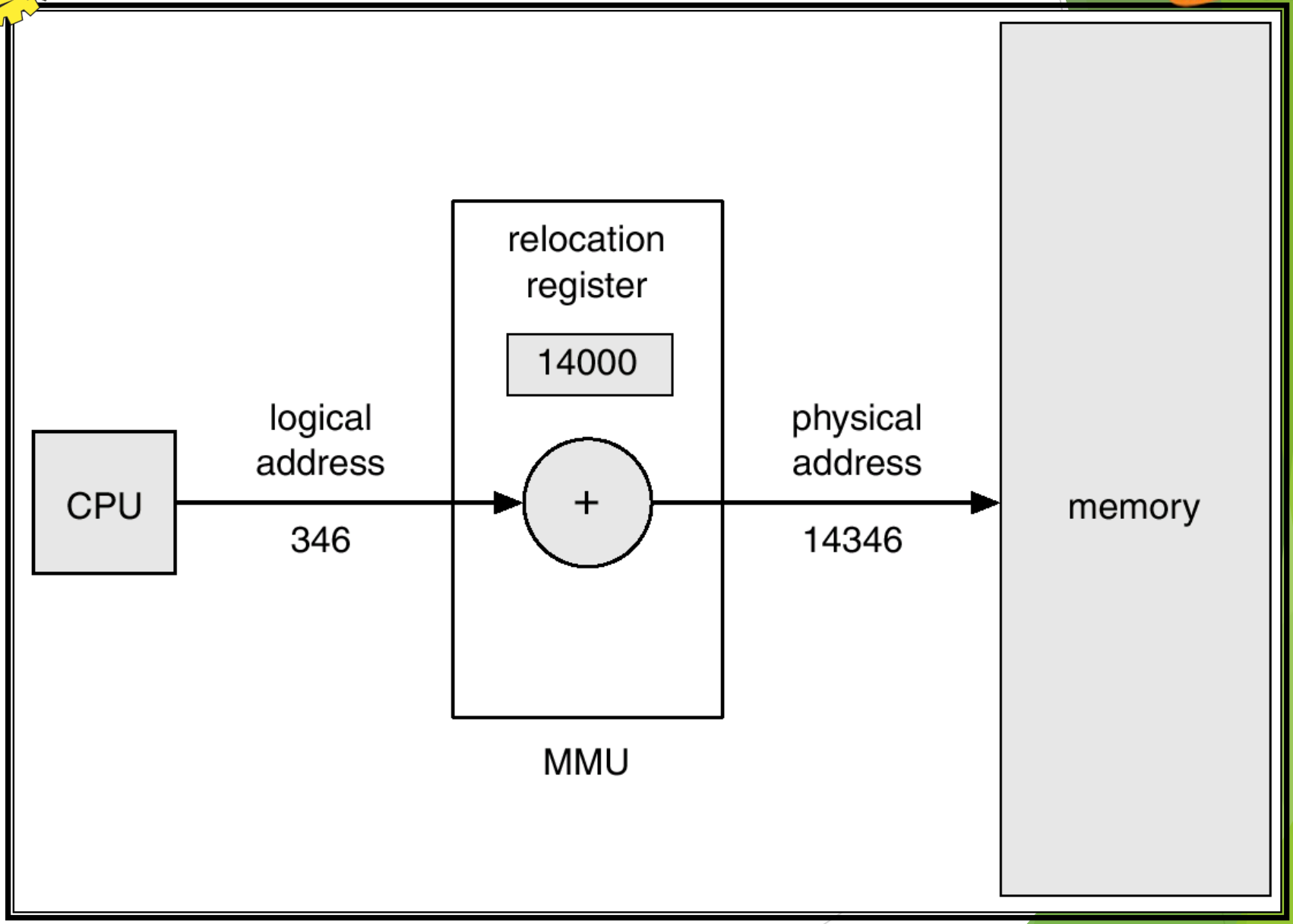
Logical vs. Physical Address Space

- The concept of a logical *address space* that is bound to a separate *physical address space* is central to proper memory management.
  - *Logical address* – generated by the CPU; also referred to as *virtual address*.
  - *Physical address* – address seen by the memory unit.
- Logical and physical addresses are the same in compile-time and load-time address-binding schemes; logical (virtual) and physical addresses differ in execution-time address-binding scheme.

Memory-Management Unit (MMU)

- Hardware device that maps virtual to physical address.
- In MMU 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.

Dynamic relocation using a relocation regis



Dynamic Loading

- Routine is not loaded until it is called
- Better memory-space utilization; unused routine is never loaded.
- Useful when large amounts of code are needed to handle infrequently occurring cases.
- No special support from the operating system is required implemented through program design.

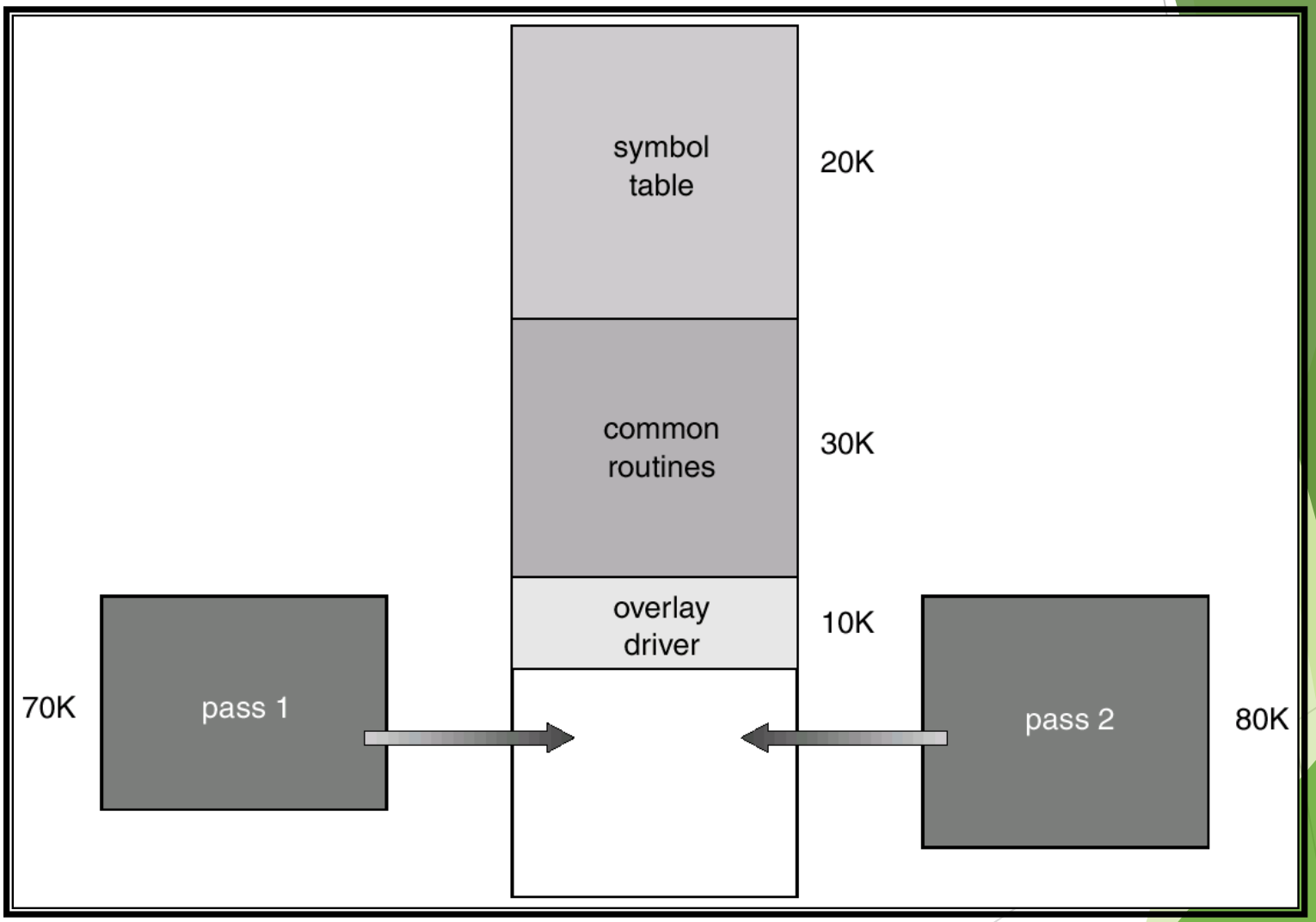
Dynamic Linking

- Linking postponed until execution time.
- Small piece of code, *stub*, used to locate the appropriate memory-resident library routine.
- Stub replaces itself with the address of the routine, and executes the routine.
- Operating system needed to check if routine is in processes' memory address.
- Dynamic linking is particularly useful for libraries.

Overlays

- Keep in memory only those instructions and data that are needed at any given time.
- Needed when process is larger than amount of memory allocated to it.
- Implemented by user, no special support needed from operating system, programming design of overlay structure is complex

Overlays for a Two-Pass Assembler



Memory Management in Operating System

- The term memory can be defined as a collection of data in a specific format.
- It is used to store instructions and process data.
- The memory comprises a large array or group of words or bytes, each with its own location.
- The primary purpose of a computer system is to execute programs.
- These programs, along with the information they access, should be in the main memory during execution.
- The CPU fetches instructions from memory according to the value of the program counter.

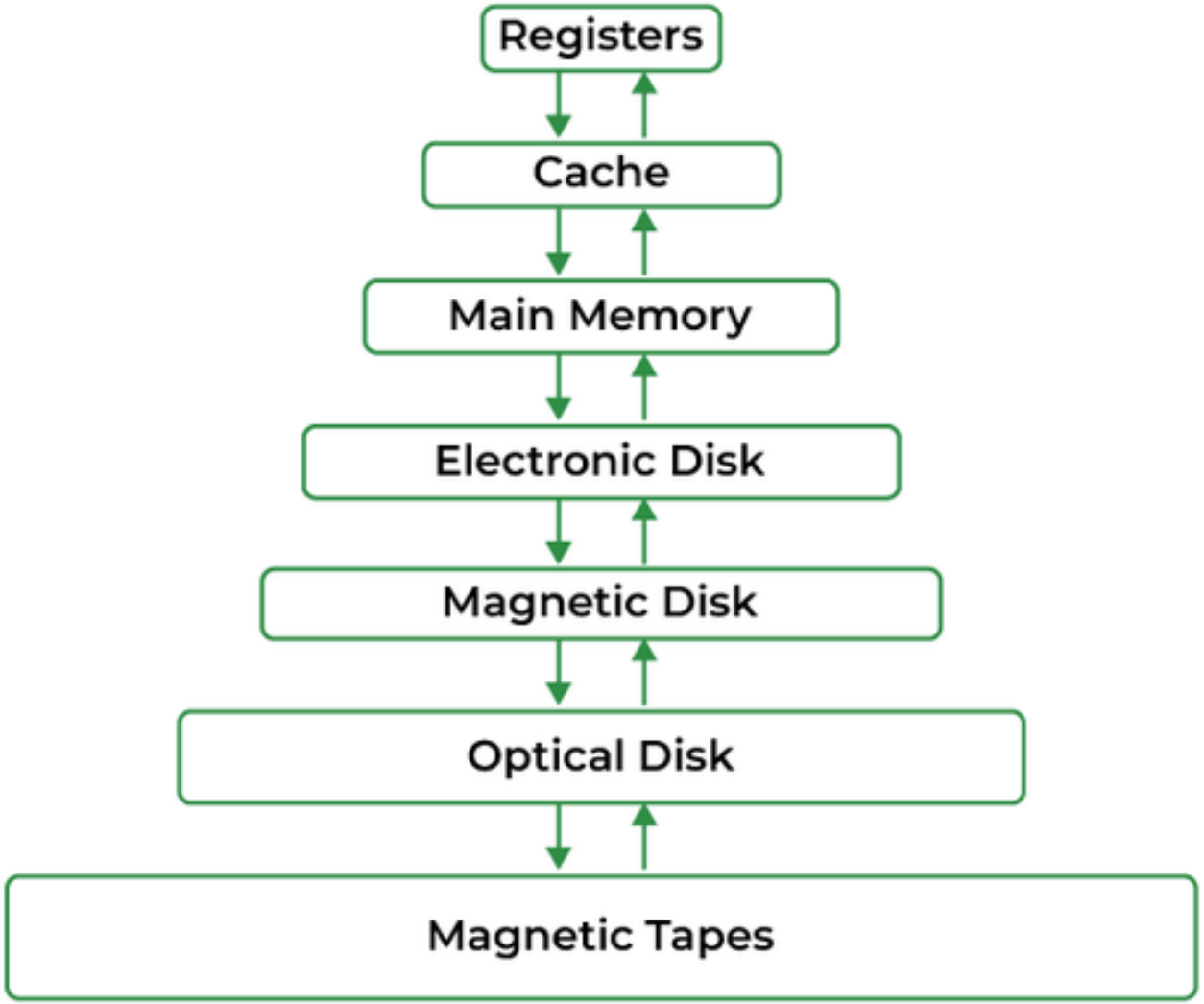
- To achieve a degree of multiprogramming and proper utilization of memory, memory management is important.
- Many memory management methods exist, reflecting various approaches, and the effectiveness of each algorithm depends on the situation.

- Before we start Memory management, let us know what is main memory is.

What is Main Memory?

- The main memory is central to the operation of a Modern Computer.
- Main Memory is a large array of words or bytes, ranging in size from hundreds of thousands to billions.
- Main memory is a repository of rapidly available information shared by the CPU and I/O devices.
- Main memory is the place where programs and information are kept when the processor is effectively utilizing them.
- Main memory is associated with the processor, so moving instructions and information into and out of the processor is extremely fast.
- Main memory is also known as RAM (Random Access Memory).
- This memory is volatile.
- RAM loses its data when a power interruption occurs.

What is Main Memory?



What is Memory Management?

- Memory management mostly involves management of main memory.
- In a multiprogramming computer, the Operating System resides in a part of the main memory, and the rest is used by multiple processes.
- The task of subdividing the memory among different processes is called Memory Management.
- Memory management is a method in the operating system to manage operations between main memory and disk during process execution.
- The main aim of memory management is to achieve efficient utilization of memory.

Why Memory Management is Required?

- 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.

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.

Logical and Physical Address Space

- **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.

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**
- **Dynamic Loading**

Static and Dynamic Loading

- **Static Loading:**
- Static Loading is basically loading the entire program into a fixed address.
- It requires more memory space.

Static and Dynamic Loading

- **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.

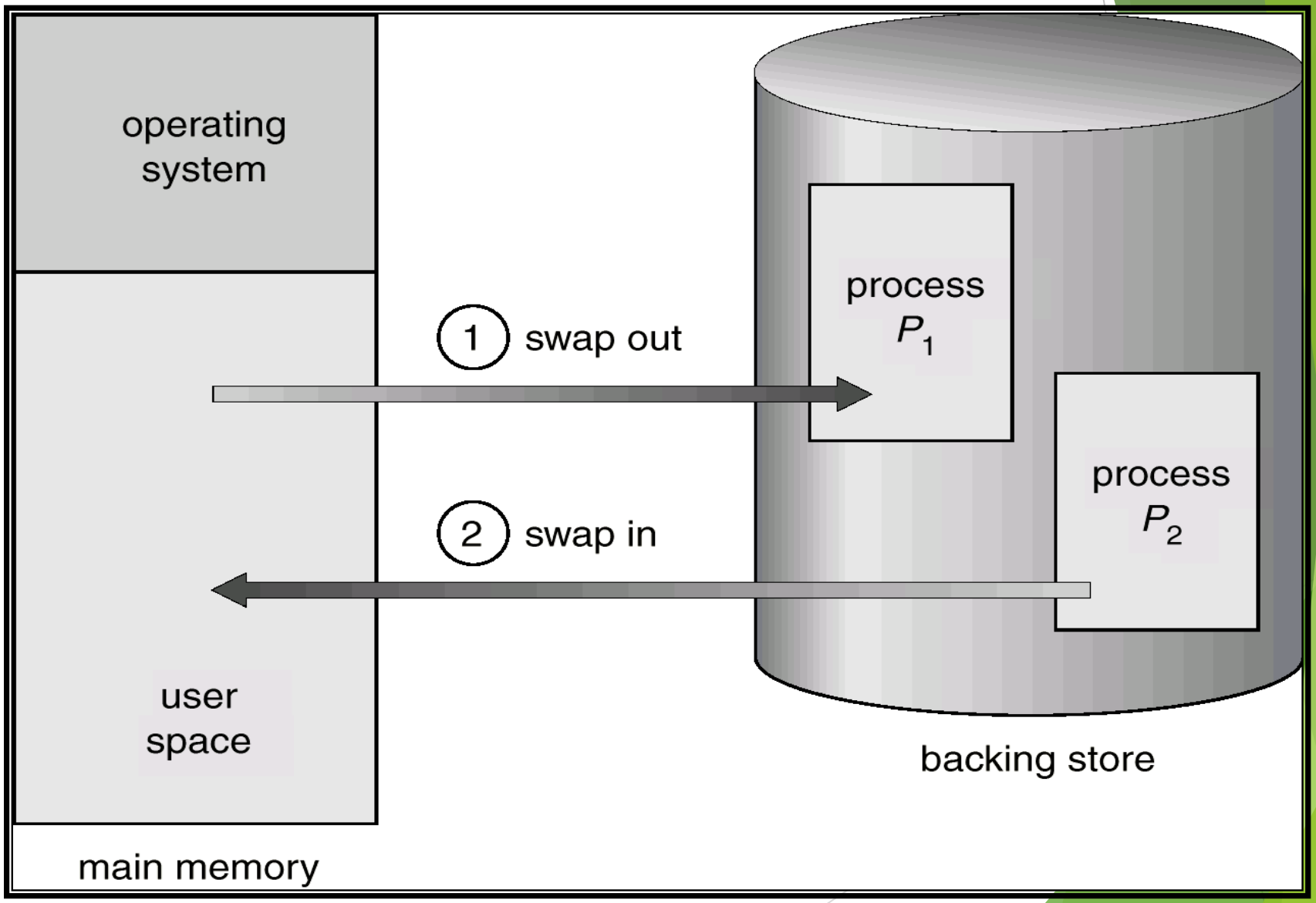
Static and Dynamic Linking

- **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.

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.
- Modified versions of swapping are found on many systems, i.e., UNIX, Linux, and Windows.

Schematic View of Swapping

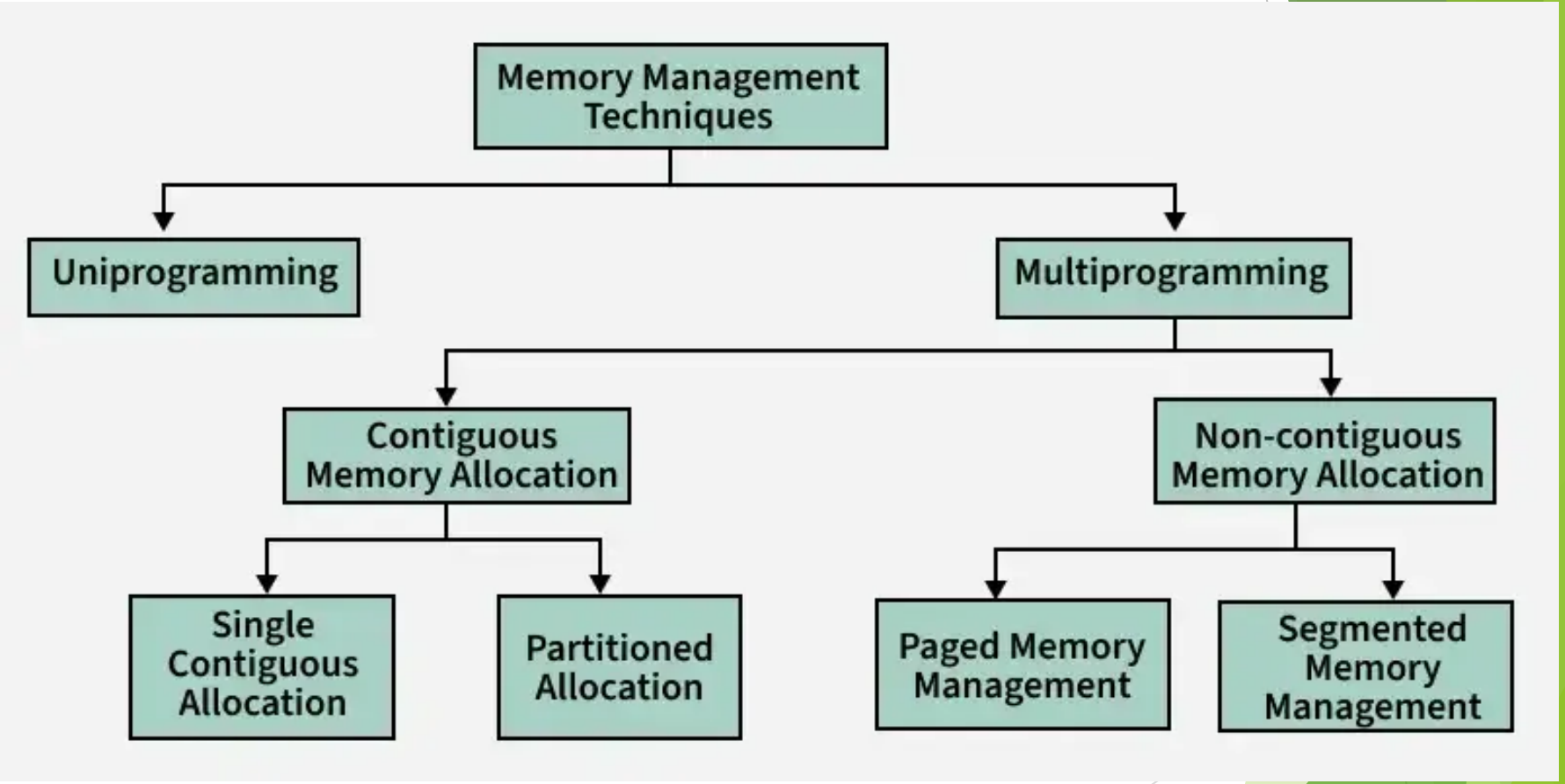


Memory Management Techniques

- Memory management techniques are methods used by an operating system to efficiently allocate, utilize, and manage memory resources for processes.
- These techniques ensure smooth execution of programs and optimal use of system memory

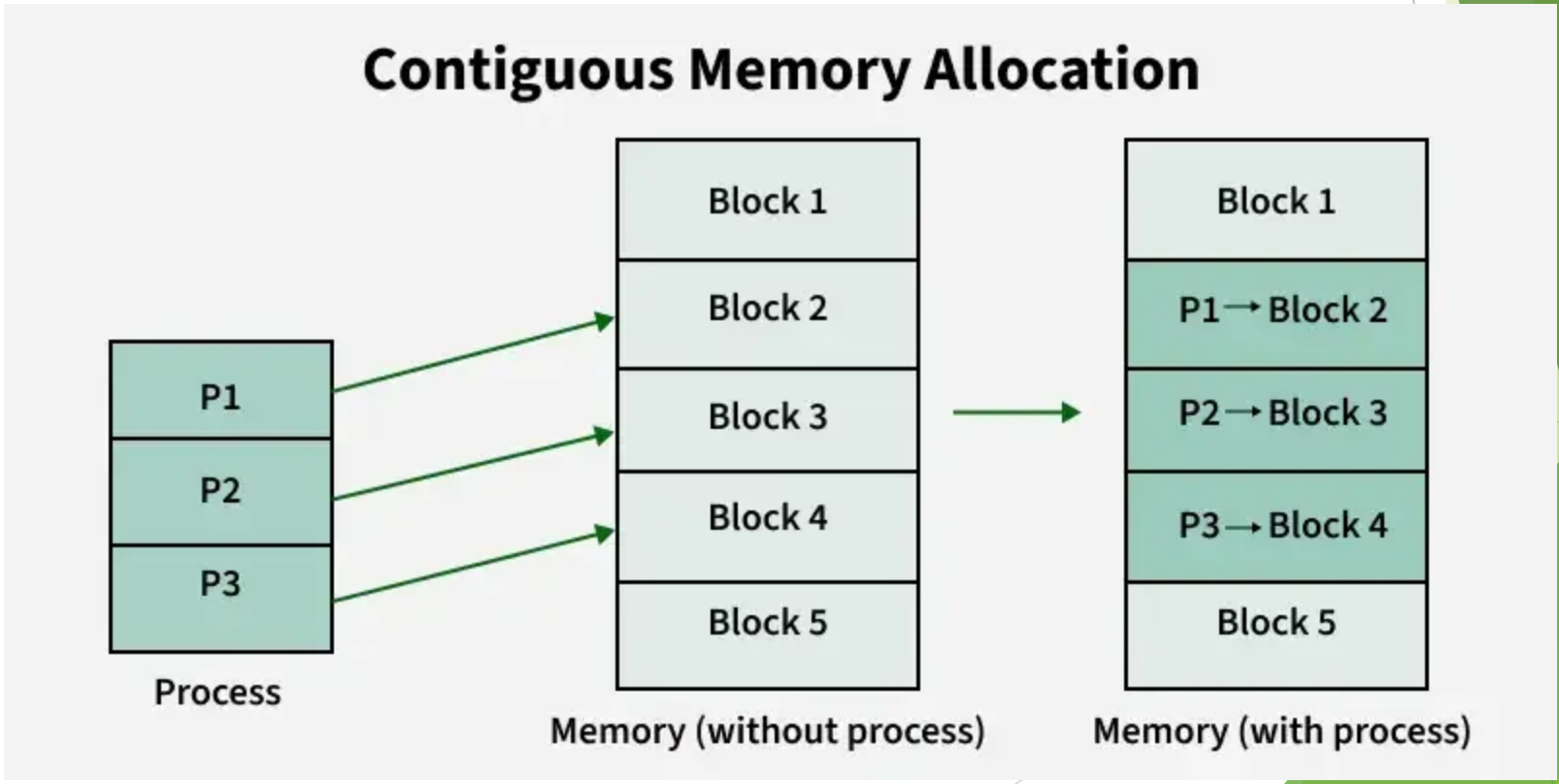
- Different Memory Management techniques are:

Memory Management Techniques



Contiguous Allocation

- Contiguous memory allocation is a memory management method where each process is given a single, continuous block of memory.
- This means all the data for a process is stored in adjacent memory locations.



Partition Allocation Methods

To gain proper memory utilization, memory allocation must be allocated efficient manner.

One of the simplest methods for allocating memory is to divide memory into several fixed-sized partitions and each partition contains exactly one process.

Thus, the degree of multiprogramming is obtained by the number of partitions.

- **Fixed partition allocation:** Memory is divided into fixed-sized partitions during system initialization. Each partition can hold only one process.

- **Dynamic Partition Allocation:** In this allocation strategy, Memory is divided into variable-sized partitions based on the size of the processes.

Partition Allocation Methods

When it is time to load a process into the main memory and if there is more than one free block of memory of sufficient size then the OS decides which free block to allocate.

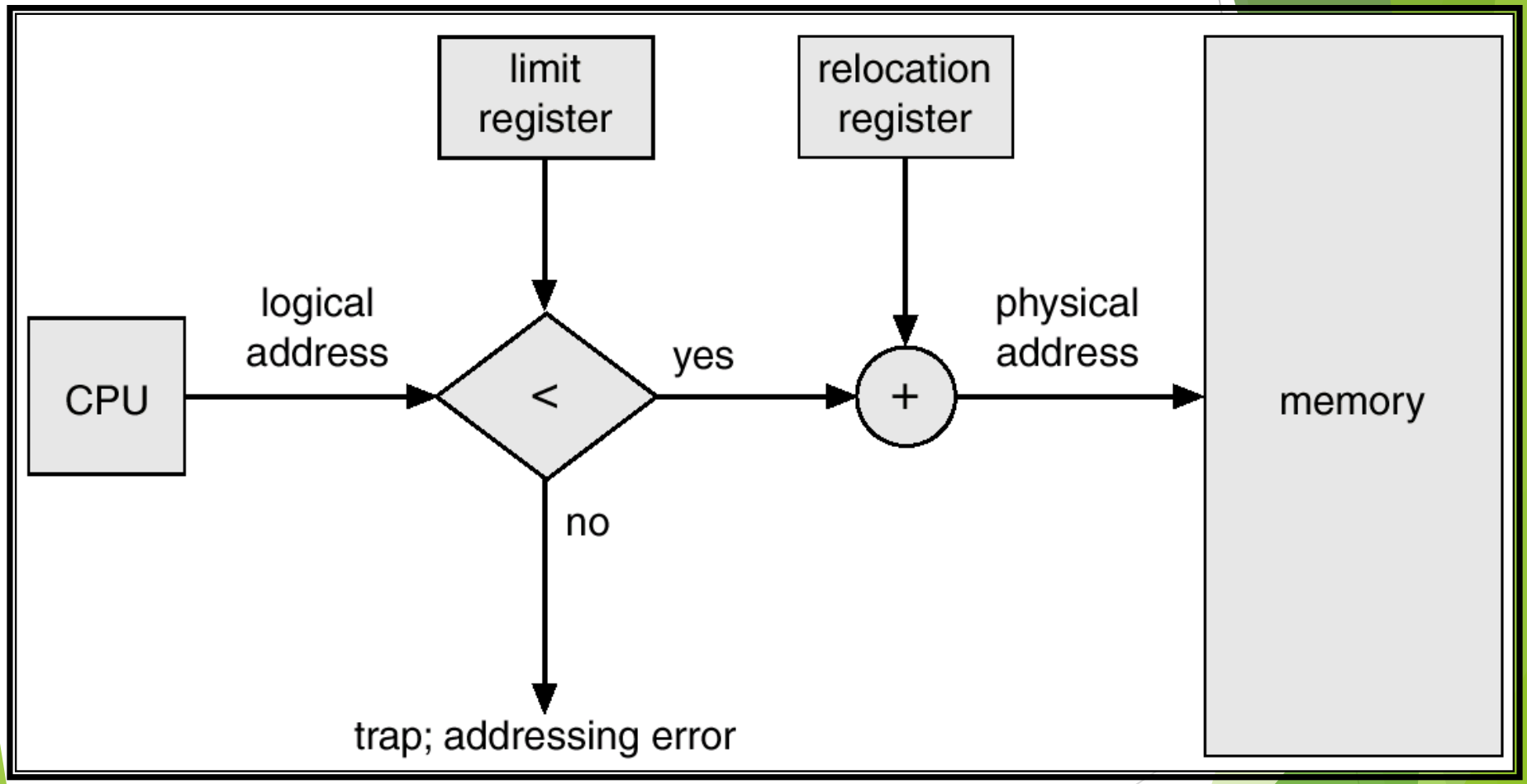
There are different Placement Algorithm:

1. First Fit
2. Best Fit
3. Worst Fit
4. Next Fit

Contiguous Allocation

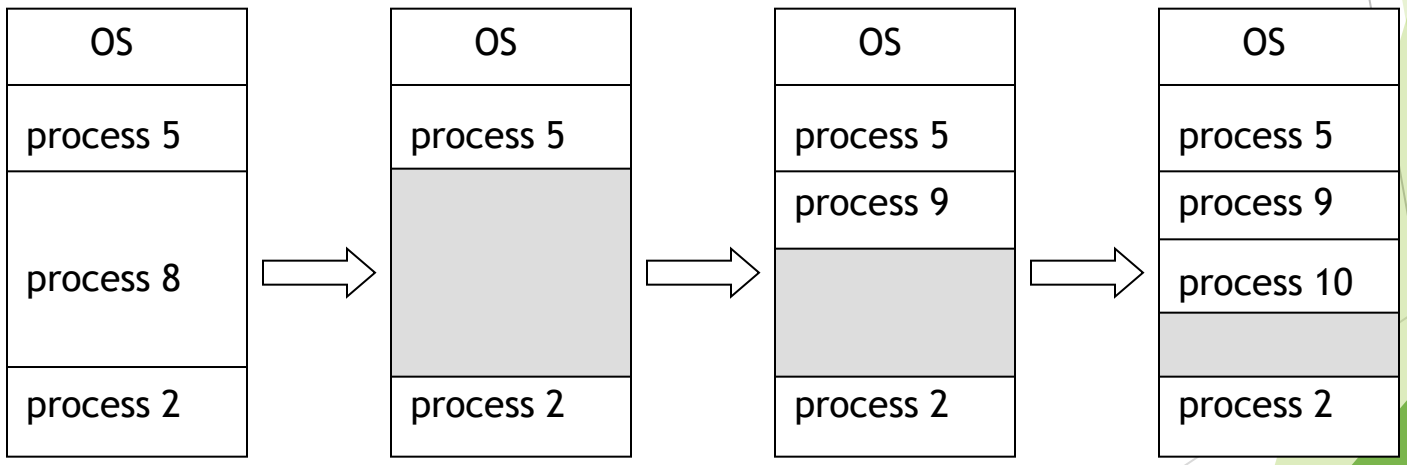
- Main memory usually into two partitions:
  - Resident operating system, usually held in low memory with interrupt vector.
  - User processes then held in high memory.
- Single-partition allocation
  - Relocation-register scheme used to protect user processes from each other, and from changing operating-system code and data.
  - Relocation register contains value of smallest physical address; limit register contains range of logical addresses – each logical address must be less than the limit register.

Hardware Support for Relocation and Limit Registers



Contiguous Allocation (Cont.)

- Multiple-partition allocation
  - *Hole* – block of available memory; holes of various size are scattered throughout memory.
  - When a process arrives, it is allocated memory from a hole large enough to accommodate it.
  - Operating system maintains information about:
    a) allocated partitions b) free partitions (hole)



Dynamic Storage-Allocation Problem

How to satisfy a request of size *n* from a list of free holes.

- **First-fit**: Allocate the *first* hole that is big enough.
- **Best-fit**: Allocate the *smallest* hole that is big enough; must search entire list, unless ordered by size. Produces the smallest leftover hole.
- **Worst-fit**: Allocate the *largest* hole; must also search entire list. Produces the largest leftover hole.

First-fit and best-fit better than worst-fit in terms of speed and storage utilization.

Fragmentation

- **External Fragmentation** – total memory space exists to satisfy a request, but it is not contiguous.
- **Internal Fragmentation** – allocated memory may be slightly larger than requested memory; this size difference is memory internal to a partition, but not being used.
- Reduce external fragmentation by compaction
  - Shuffle memory contents to place all free memory together in one large block.
  - Compaction is possible *only* if relocation is dynamic, and is done at execution time.
  - I/O problem
    - Latch job in memory while it is involved in I/O.
    - Do I/O only into OS buffers.