**Introduction to Memory Management & Contiguous Memory Allocation**

**1. What is Memory Management?**

- Definition: Memory management is the process of coordinating and managing computer memory, which includes the allocation and deallocation of memory spaces as required by programs.

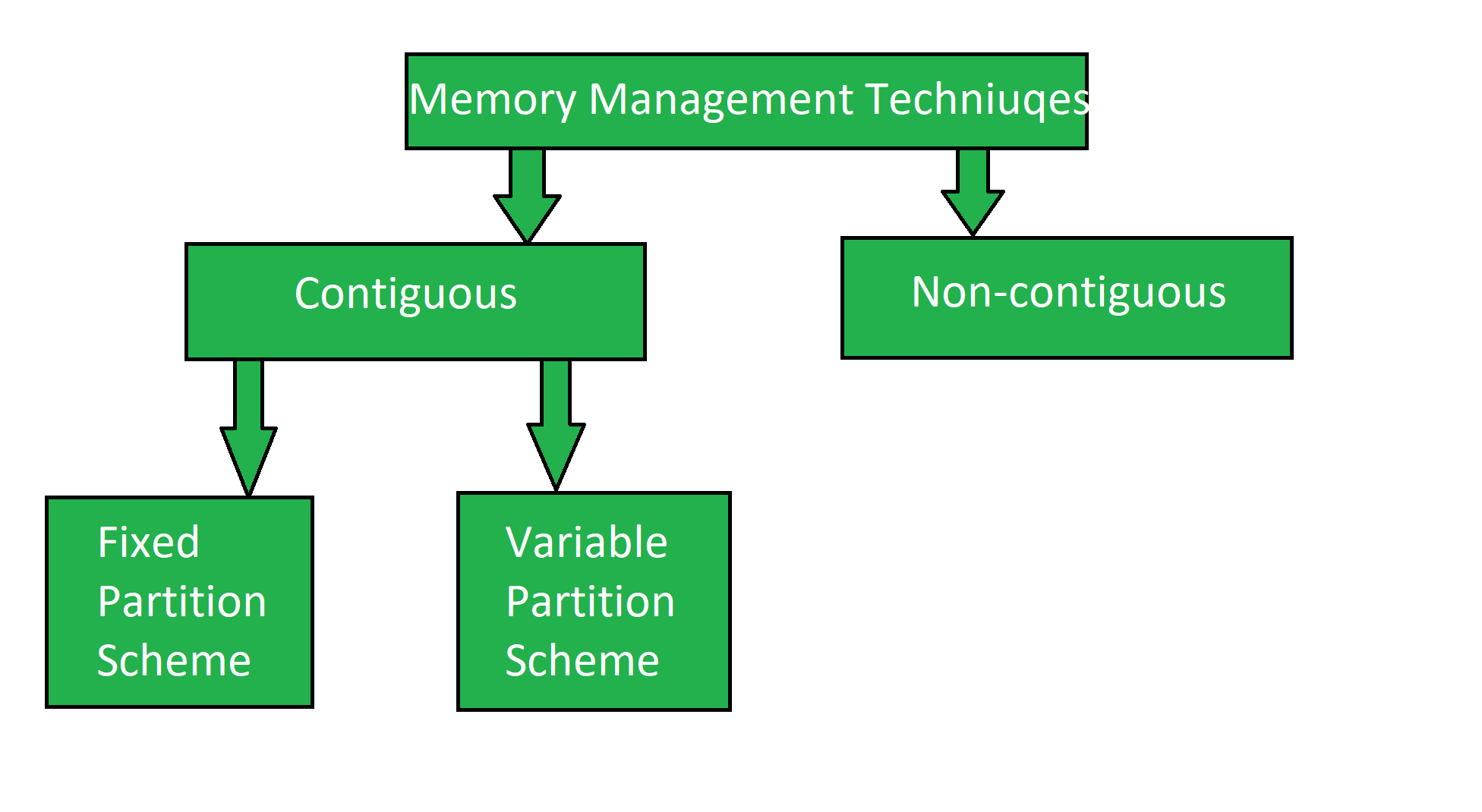
- Goals of Memory Management:

- Efficient memory utilization.

- Protection and isolation between processes.

- Fast allocation and deallocation.

- Memory Hierarchy Overview: Registers, Cache, RAM, Hard Disk.

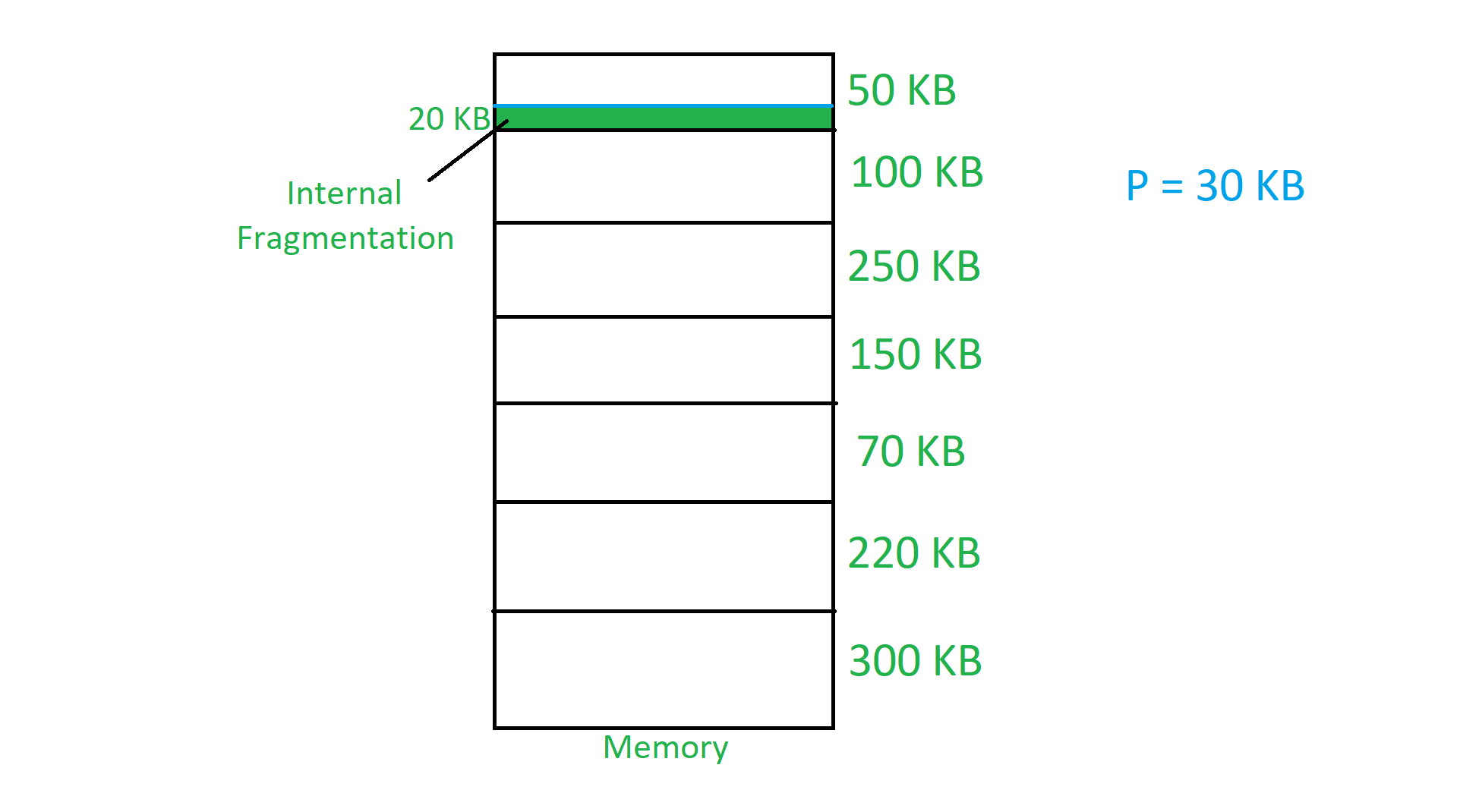


**2. Contiguous Memory Allocation**

- Definition: Memory is allocated in a single continuous block. Each process gets a single block of contiguous memory.

- Types:

**- Fixed Partitioning**: Dividing memory into fixed-size blocks. Fixed means number of partitions are fixed in the memory. In the fixed partition, in every partition only one process will be accommodated.



Every partition is associated with the limit registers.

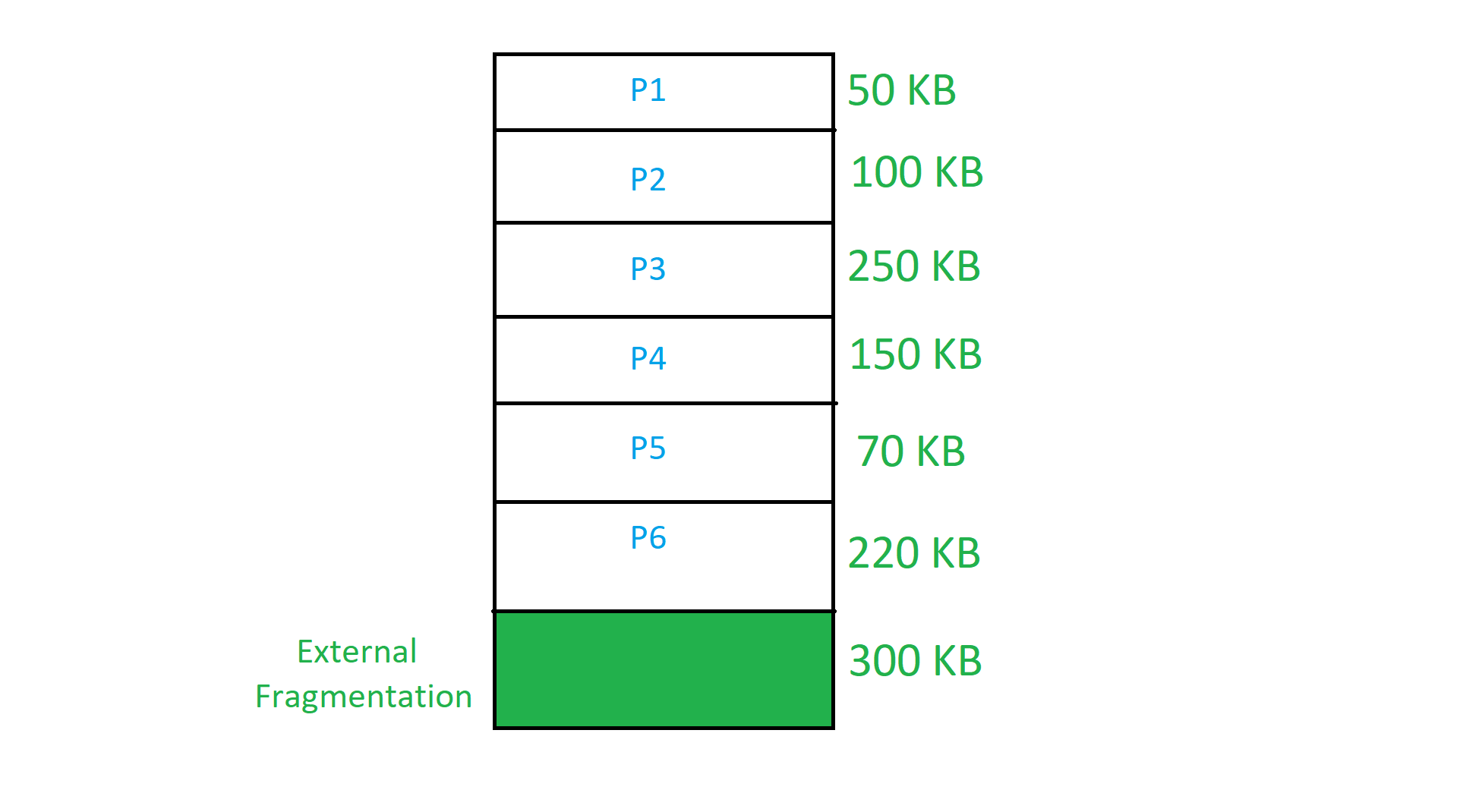
* **Limit Registers:** It has two limit:
* **Lower Limit:** Starting address of the partition.
* **Upper Limit:** Ending address of the partition.

**Disadvantages Fix partition scheme**

* Maximum process size <= Maximum partition size.
* The degree of multiprogramming is directly proportional to the number of partitions.
* Internal fragmentation which is discussed above is present.
* If a process of 19kb wants to allocate and we have free space which is not continuous we are not able to allocate the space.

**Dynamic Partitioning**: Dividing memory into variable-sized blocks based on process needs. If the smaller processes keep on coming then the larger partitions will be made into smaller partitions

* In variable partition schema initially, the memory will be full contiguous free block
* Memory divided into partitions according to the process size where process size will vary.
* One partition is allocated to each active partition.



- Advantages:

- Simple to implement.

- Easy to track which blocks are free or used.

- Disadvantages:

- External Fragmentation: Unused memory blocks spread across different areas.

- Internal Fragmentation: Wastage inside allocated blocks if the block size is larger than needed.

- Example:

- Imagine a memory space of 100 MB with three processes requiring 25 MB, 40 MB, and 20 MB, allocated consecutively.

- Diagram:

- Show a contiguous block of memory with three processes and some free space left between them.

**3. Memory Allocation Techniques**

- First Fit, Best Fit, Worst Fit: Techniques for finding the free memory block that fits the process's needs.

- Example:

- Process A (25MB), B (40MB), and C (15MB) request memory. Demonstrate how each allocation method would allocate these processes differently.

- Diagram:

- Visualize memory blocks showing how each method works.

**Memory Allocation Table**

| **Memory Block Size** | **Initial State** | **First Fit** | **Best Fit** | **Worst Fit** |
| --- | --- | --- | --- | --- |
| **10MB** | Free | 10MB free | 10MB free | 10MB free |
| **20MB** | Free | 5MB free (after Process C) | 5MB free (after Process C) | 5MB free (after Process C) |
| **30MB** | Free | 5MB free (after Process A) | 5MB free (after Process A) | 0MB free (after Process B) |
| **50MB** | Free | 10MB free (after Process B) | 10MB free (after Process B) | 25MB free (after Process A) |

**Process Allocation Breakdown**

| **Process** | **Memory Needed** | **First Fit** | **Best Fit** | **Worst Fit** |
| --- | --- | --- | --- | --- |
| **Process A** | 25MB | Allocated to 30MB block (5MB free) | Allocated to 30MB block (5MB free) | Allocated to 50MB block (25MB free) |
| **Process B** | 40MB | Allocated to 50MB block (10MB free) | Allocated to 50MB block (10MB free) | Allocated to 30MB block (0MB free) |
| **Process C** | 15MB | Allocated to 20MB block (5MB free) | Allocated to 20MB block (5MB free) | Allocated to 20MB block (5MB free) |

**Paging**

**1. What is Paging?**

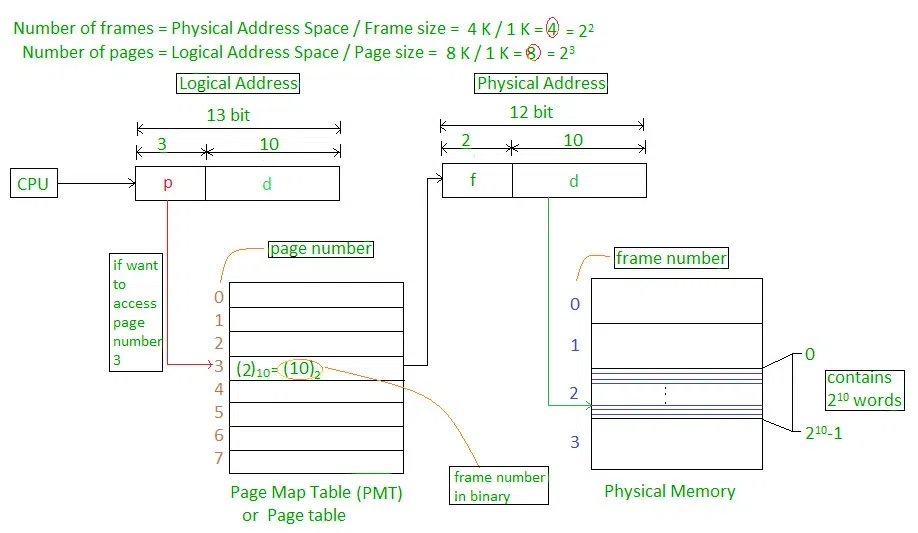
- Definition: Paging is a memory management scheme that eliminates the need for contiguous memory allocation by dividing memory into fixed-size pages.

- Key Concepts:

- Pages: Fixed-size blocks of memory in the process.

- Frames: Corresponding blocks of physical memory.

- Page Table: Maps virtual pages to physical frames.



- Advantages:

- No external fragmentation.

- Efficient memory use by fitting pages into available frames.

- Disadvantages:

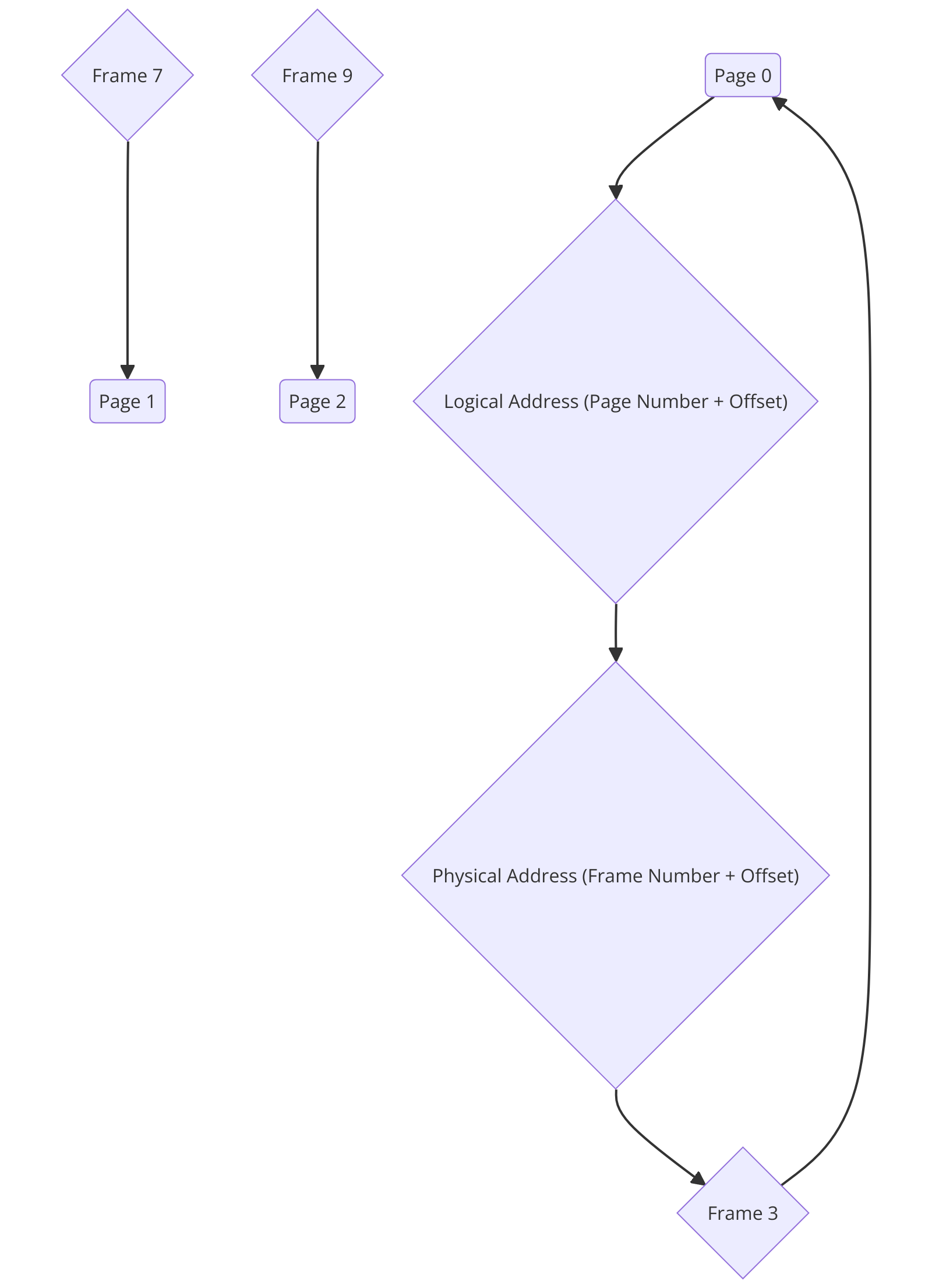
- Page Table Overhead: Each process has its page table, which consumes memory.

- Internal Fragmentation: If a process doesn't fully use a page, the rest is wasted.

**2. Working of Paging**

- Example:

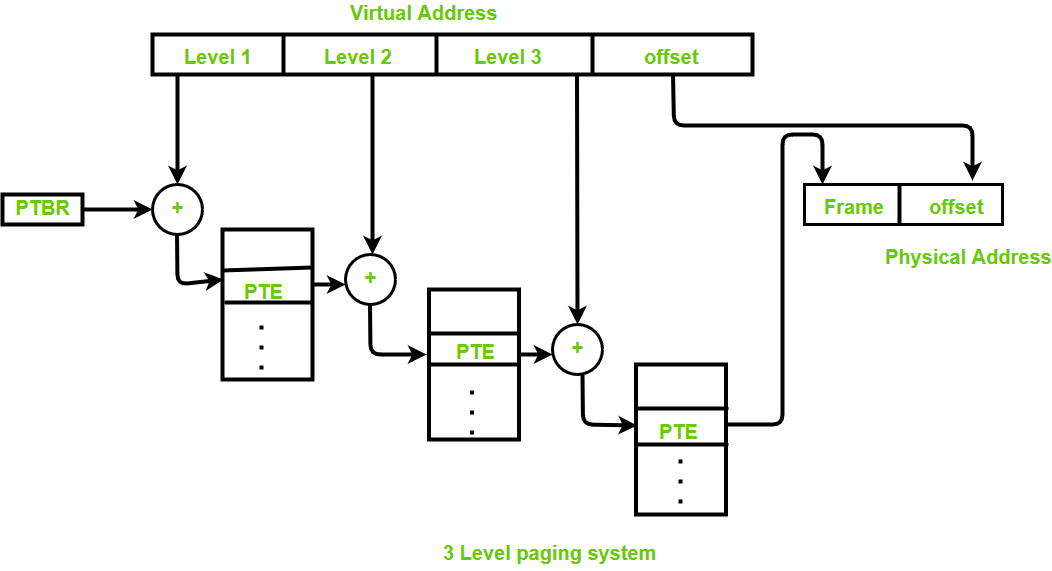
- Assume a process has a 12 KB address space and the system uses 4 KB pages. The process is split into three pages and loaded into non-contiguous frames in memory.



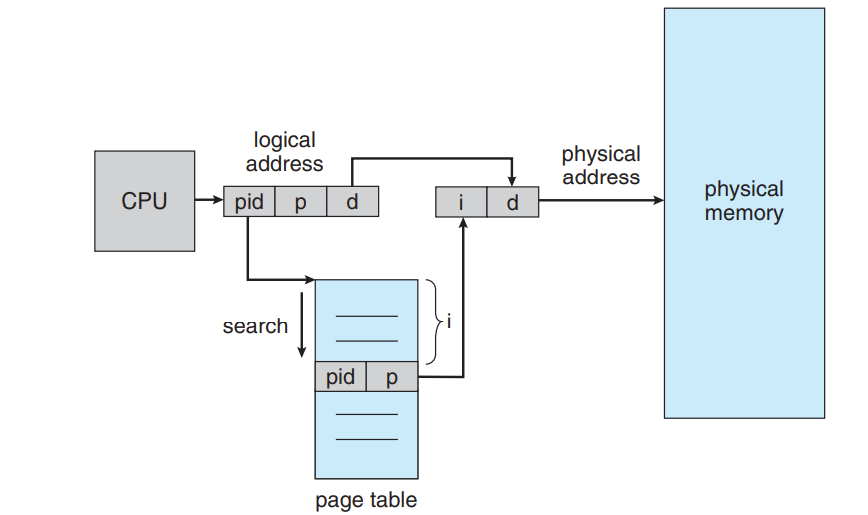
**3. Types of Paging**

- Simple Paging: Basic page-to-frame mapping.

- Multi-Level Paging: Multiple levels of page tables, used in large address spaces.



- Inverted Paging: A single page table for all processes, reducing memory overhead.



**Segmentation**

**1. What is Segmentation?**

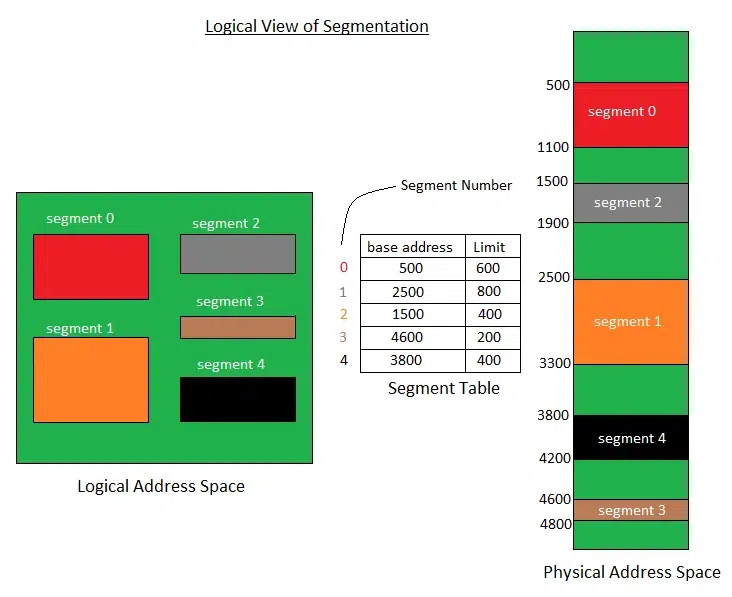
- Definition: Segmentation is a memory management scheme where memory is divided into variable-sized segments based on logical divisions like code, data, and stack.

- Key Concepts:

- Each segment represents a logical section (e.g., code, data, stack).

- Segments can be of different sizes, unlike paging where all pages are of fixed size.

- Segment Table: Maps segment numbers to physical addresses.



- Advantages:

- Segments correspond to the programmer's view of memory, making it easier to handle.

- More flexible than paging as segments can vary in size.

- Disadvantages:

- External Fragmentation: Over time, free memory becomes fragmented as segments are allocated and deallocated.

**2. Segmentation vs. Paging**

- Paging: Divides memory into fixed-size blocks. No logical division.

- Segmentation: Divides memory based on logical components (code, data, stack).

| **Paging** | **Segmentation** |
| --- | --- |
| For the paging operating system is accountable. | For segmentation compiler is accountable. |
| Page size is determined by hardware. | Here, the segment size is given by the user. |
| It is faster in comparison to segmentation. | Segmentation is slow. |
| Paging could result in internal fragmentation. | Segmentation could result in external fragmentation. |
| In paging, the logical address is split into a page number and page offset. | Here, the logical address is split into segment number and segment offset. |
| Paging comprises a page table that encloses the base address of every page. | While segmentation also comprises the segment table which encloses segment number and segment offset. |
| The page table is employed to keep up the page data. |  |
| In paging, the operating system must maintain a free frame list. | In segmentation, the operating system maintains a list of holes in the main memory. |
| Paging is invisible to the user. | Segmentation is visible to the user. |
| In paging, the processor needs the page number, and offset to calculate the absolute address. | In segmentation, the processor uses segment number, and offset to calculate the full address. |
| It is hard to allow sharing of procedures between processes. | Facilitates sharing of procedures between the processes. |
| In paging, a programmer cannot efficiently handle data structure. | It can efficiently handle data structures. |
| This protection is hard to apply. | Easy to apply for protection in segmentation. |
| The size of the page needs always be equal to the size of frames. | There is no constraint on the size of segments. |
| A page is referred to as a physical unit of information. | A segment is referred to as a logical unit of information. |
| Paging results in a less efficient system. | Segmentation results in a more efficient system. |