OPERATING SYSTEM

Why We Need Paging

Paging is needed to solve problems of contiguous memory allocation, especially external fragmentation.

Problems Without Paging

- A process needs **one continuous block** of memory
- Free memory may exist but in **small scattered holes**
- Process cannot be loaded even if **total free memory is sufficient**
- Memory wastage occurs

Paging Solves These Problems

- Eliminates **external fragmentation**
- Allows processes to be stored **anywhere in memory**
- Improves **memory utilization**
- Supports **virtual memory**
- Allows execution of programs **larger than physical memory**

What is Paging?

Definition

Paging is a **non-contiguous memory management technique** in which:

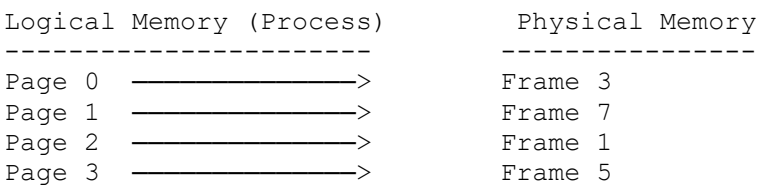
- **Physical memory** is divided into fixed-size blocks called **frames**
- **Logical memory (process)** is divided into same-size blocks called **pages**

A process's pages can be placed **in any available frame** in memory.

How Paging Works

1. Program is divided into **pages**
2. Main memory is divided into **frames**
3. OS maintains a **page table**
4. Page table maps **page number → frame number**
5. Logical address is translated into physical address using the page table

Paging Diagram



Advantages of Paging

- Eliminates **external fragmentation**
- Efficient memory usage
- Simplifies memory allocation
- Supports multiprogramming

Disadvantages of Paging

- **Internal fragmentation** (unused space in last page)
- Extra memory required for page tables
- Address translation overhead

Paging is a memory management technique in which logical memory is divided into pages and physical memory into frames of equal size. Paging allows non-contiguous memory allocation, eliminating external fragmentation and improving memory utilization.