# 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**

## COURSE NAME : 23CST202 – OPERATING SYSTEMS

## II YEAR/ IV SEMESTER

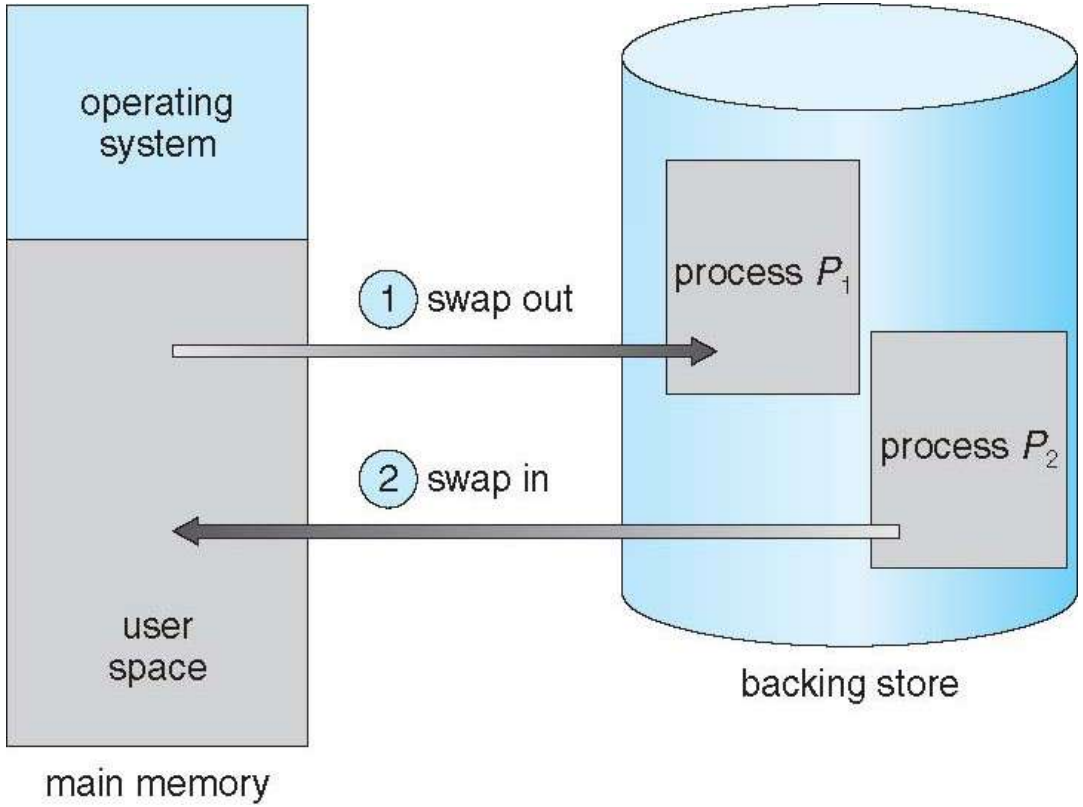
## UNIT – III STORAGE MANAGEMENT

## Topic: Contiguous memory allocation and Swapping

Dr.V.Savitha
Associate Professor
Department of Computer Science and Engineering

# Swapping

- A process can be **swapped** temporarily out of memory to a backing store, and then brought back into memory for continued execution
  - Total physical memory space of processes can exceed physical memory



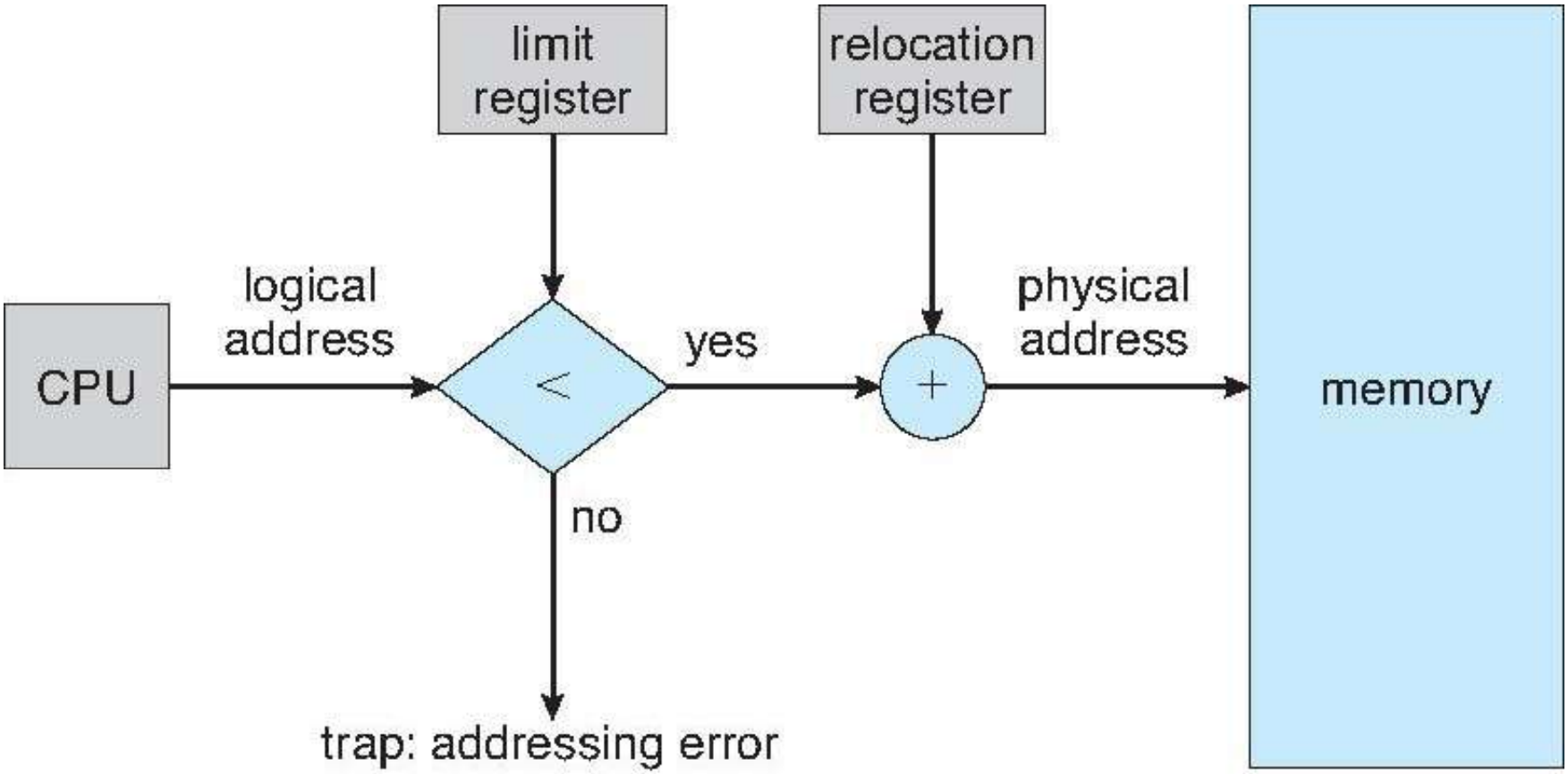
# Contiguous Allocation

- Main memory must support both OS and user processes
- Limited resource, must allocate efficiently
- Contiguous allocation is one early method
- Main memory usually into two **partitions**:
  - Resident operating system, usually held in low memory with interrupt vector
  - User processes then held in high memory
  - Each process contained in single contiguous section of memory

# Contiguous Allocation

- Relocation registers used to protect user processes from each other, and from changing operating-system code and data
  - Base register contains value of smallest physical address
  - Limit register contains range of logical addresses – each logical address must be less than the limit register
  - MMU maps logical address *dynamically*
  - Can then allow actions such as kernel code being **transient** and kernel changing size

# Hardware Support for Relocation and Limit Registers

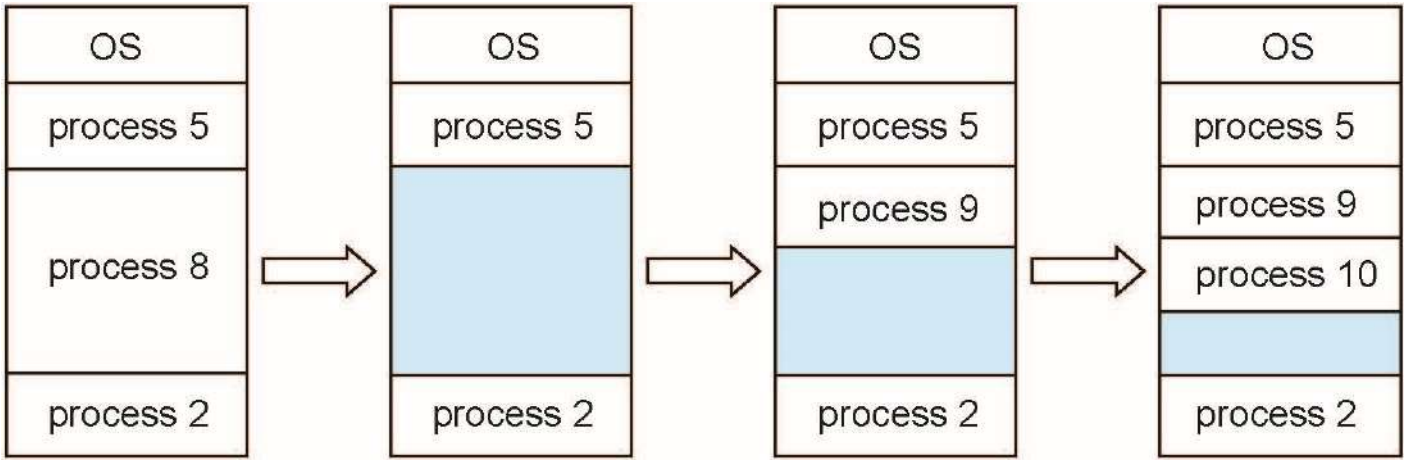


# Multiple-partition allocation

Multiple-partition allocation

- Degree of multiprogramming limited by number of partitions
- **Variable-partition** sizes for efficiency (sized to a given process' needs)
- **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
- Process exiting frees its partition,adjacent free partitions combined
- 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

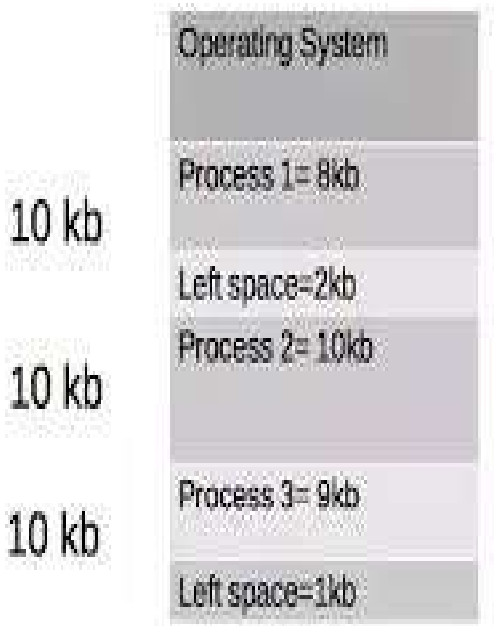
First-fit and best-fit better than worst-fit in terms of speed and storage utilization

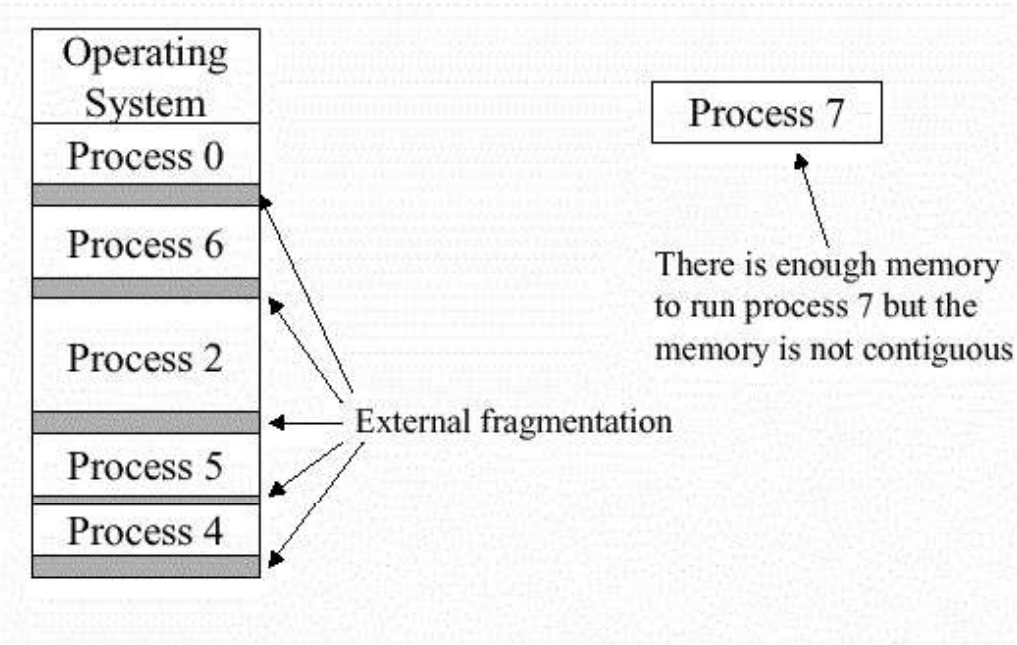
# Fragmentation

- **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

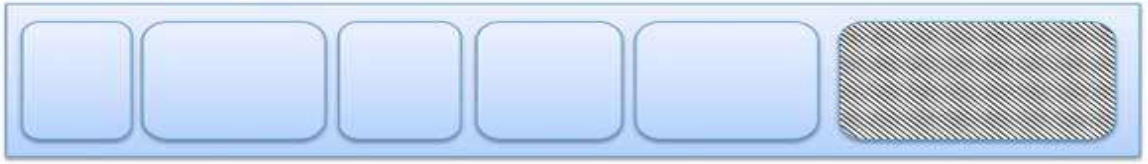
# Fragmentation




Fragmented memory before compaction

Memory after compaction



# Fragmentation

- 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
- Now consider that backing store has same fragmentation problems