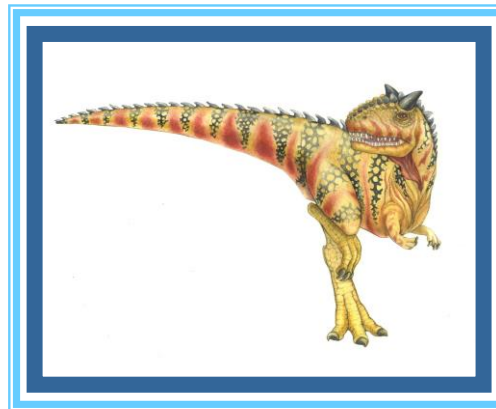


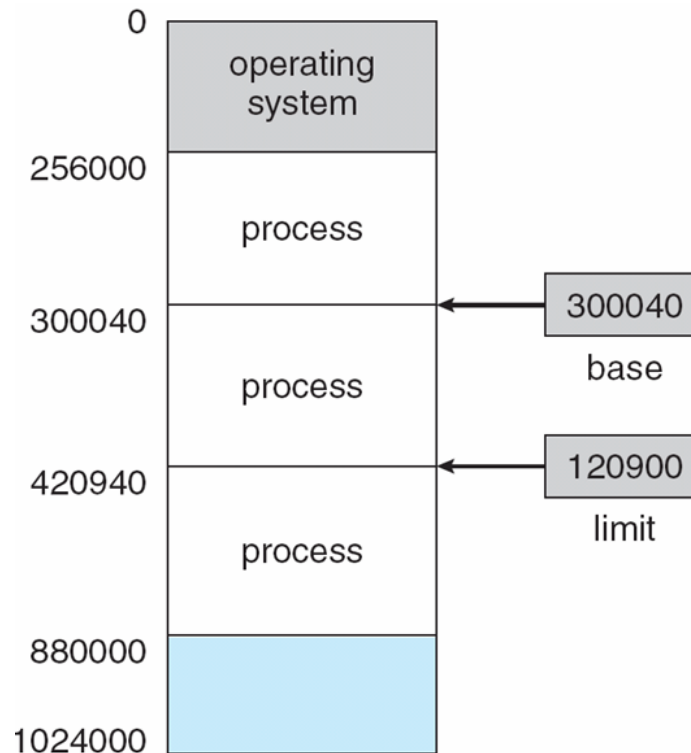
OS-5

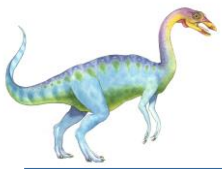




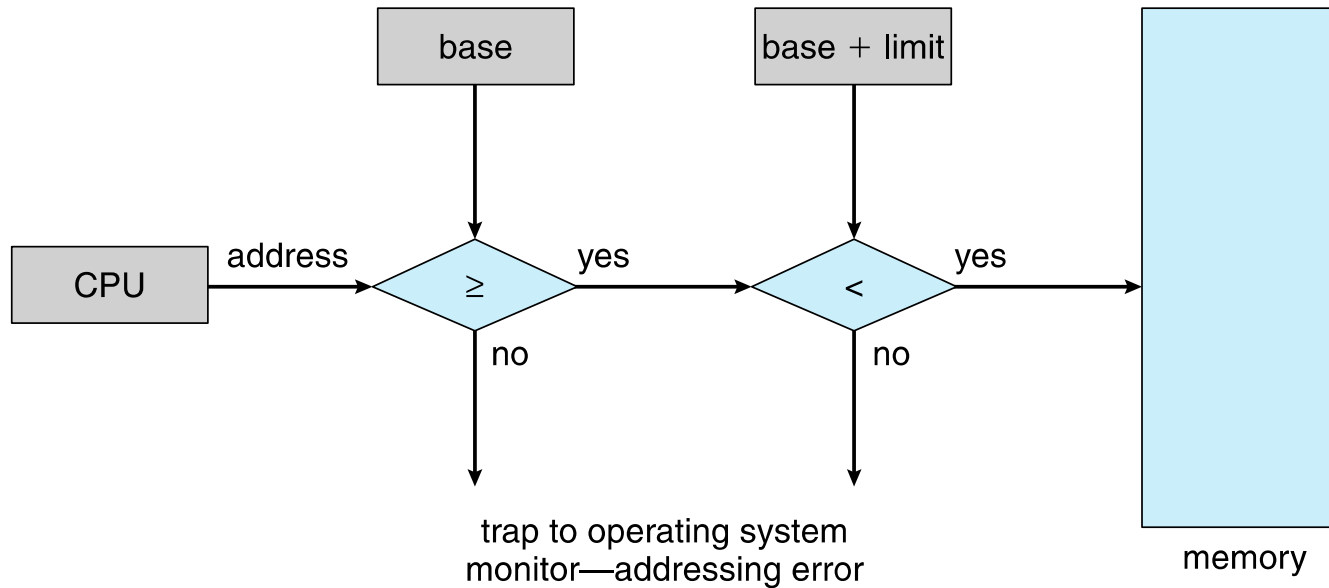
Base and Limit Registers

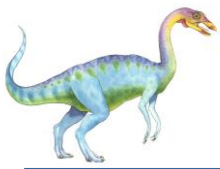
- A pair of **base** and **limit registers** define the logical address space
- CPU must check every memory access generated in user mode to be sure it is between base and limit for that user





Hardware Address Protection

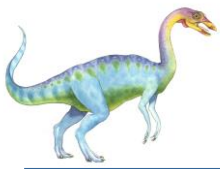




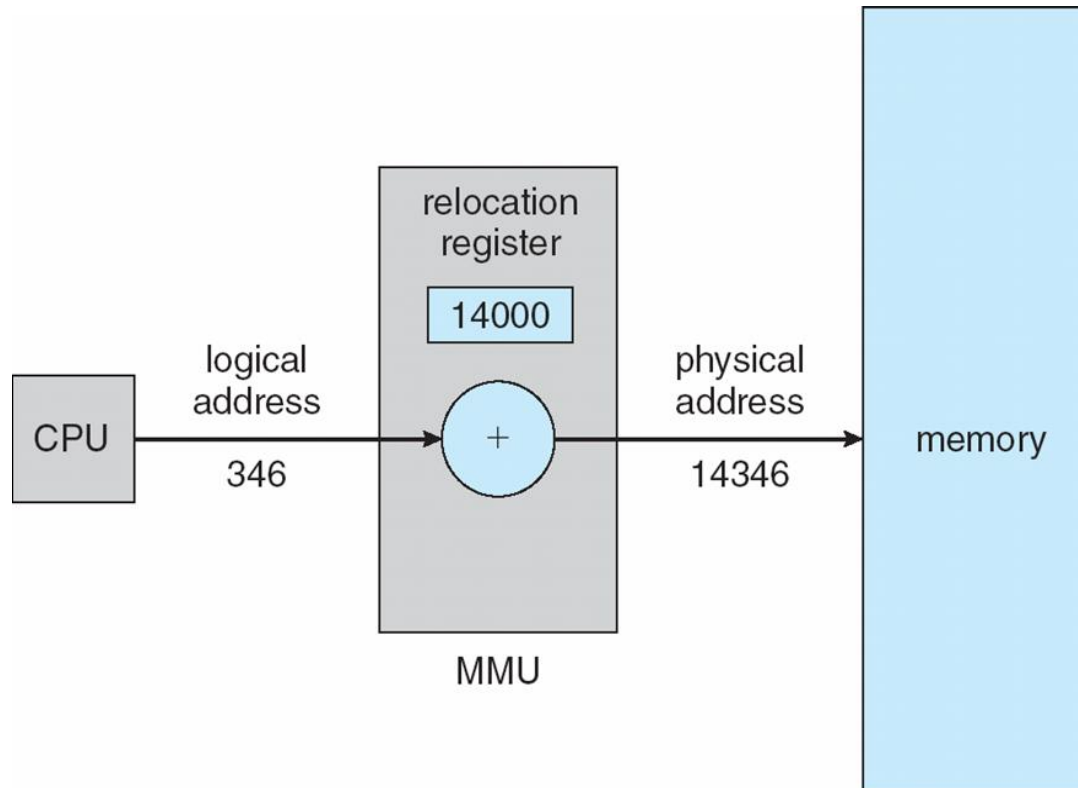
Logical vs. Physical Address Space

- 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
- **Logical address space** is the set of all logical addresses generated by a program
- **Physical address space** is the set of all physical addresses generated by a program





Dynamic relocation using a relocation register



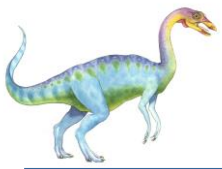


Swapping

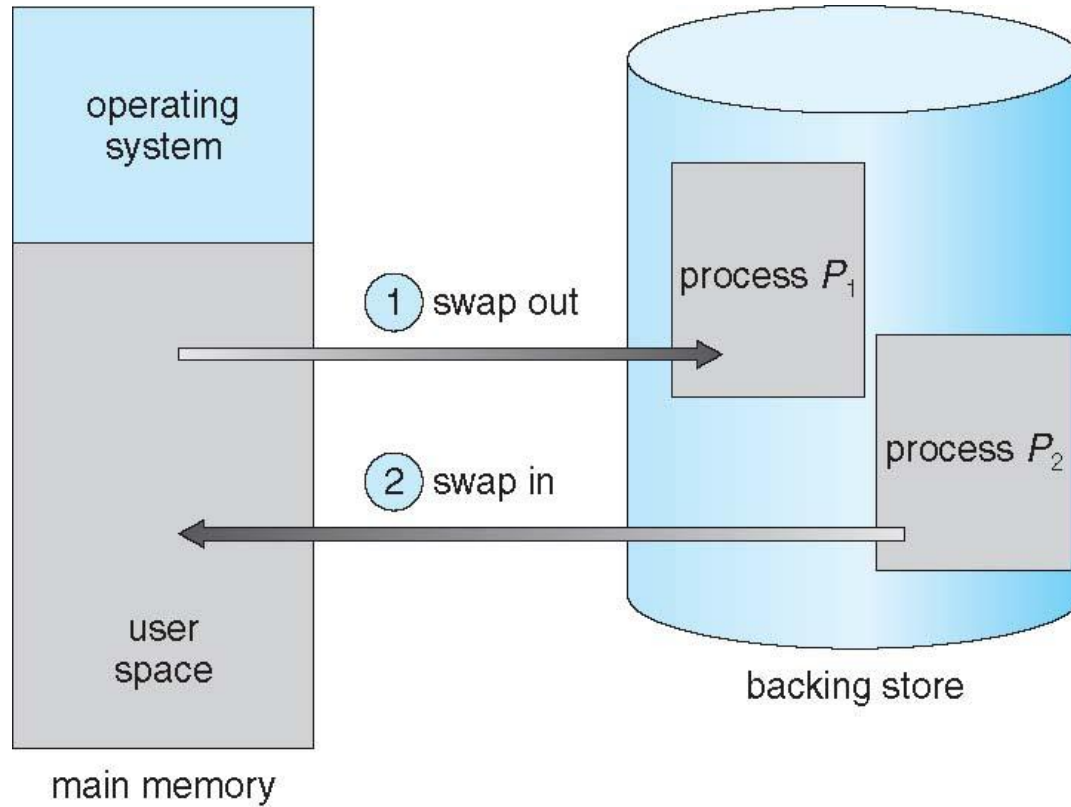
- A process can be **swapped** temporarily out of memory to a backing store, and then brought back into memory for continued execution
- **Roll out, roll in** – swapping variant used for priority-based scheduling algorithms; lower-priority process is swapped out so higher-priority process can be loaded and executed
- Major part of swap time is transfer time; total transfer time is directly proportional to the amount of memory swapped
- System maintains a **ready queue** of ready-to-run processes which have memory images on disk

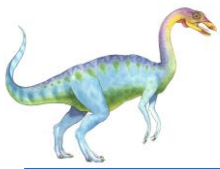
******Dynamic Loading** and **Dynamic Linking** are given as class lecture, follow the class note.





Schematic View of Swapping





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





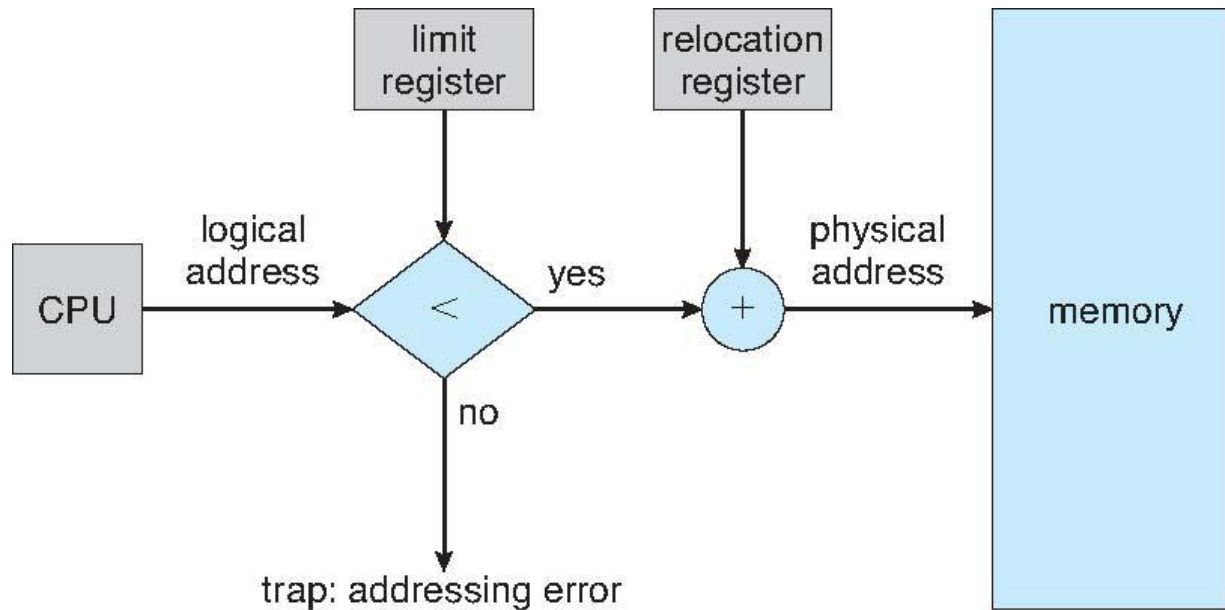
Contiguous Allocation (Cont.)

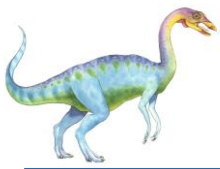
- Relocation registers used to protect user processes from each other, and from changing operating-system code and data
 - **Base register contains value of smallest physical address**
 - **Limit register contains range of logical addresses** – each logical address must be less than the limit register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





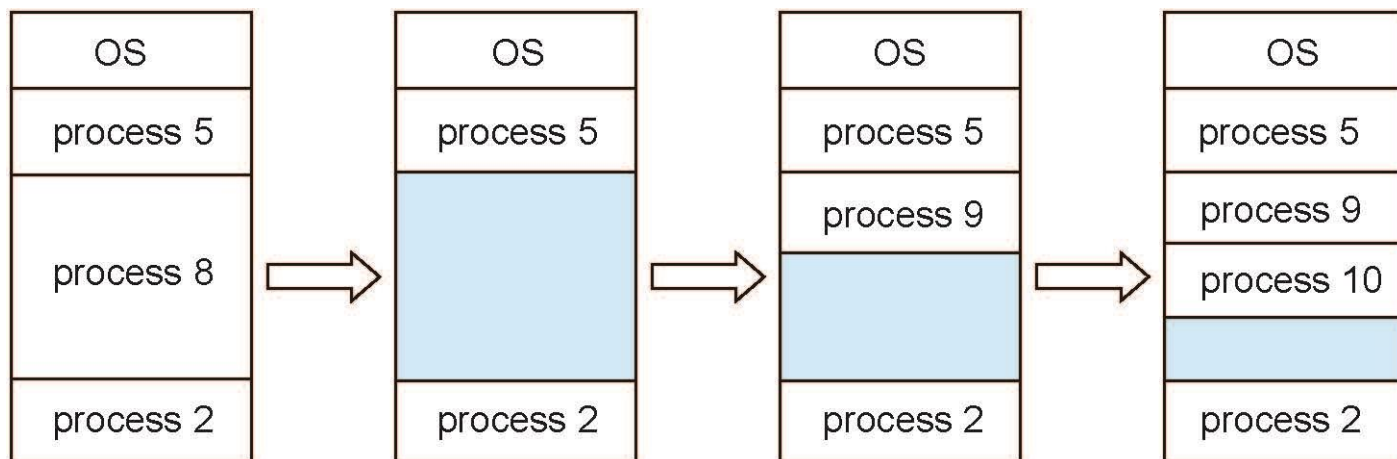
Hardware Support for Relocation and Limit Registers

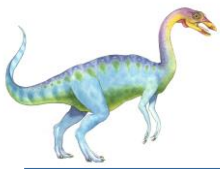




Multiple-partition allocation

- 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?

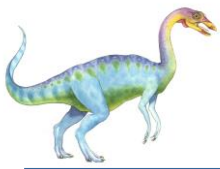
- **First-fit:** Allocate the *first* hole that is big enough

- **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

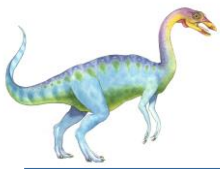




Paging

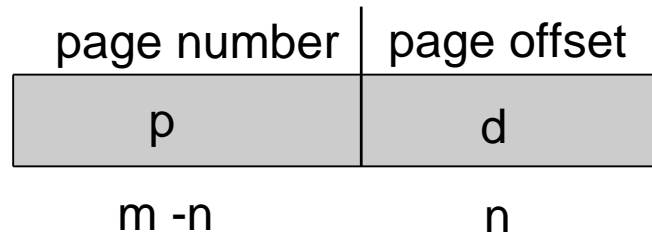
- Physical address space of a process can be noncontiguous; process is allocated physical memory whenever the latter is available
 - Avoids external fragmentation
 - Avoids problem of varying sized memory chunks
- Divide physical memory into fixed-sized blocks called **frames**
 - Size is power of 2, between 512 bytes and 16 Mbytes
- Divide logical memory into blocks of same size called **pages**
- Keep track of all free frames
- To run a program of size **N** pages, need to find **N** free frames and load program
- Set up a **page table** to translate logical to physical addresses
- Backing store likewise split into pages
- Still have Internal fragmentation





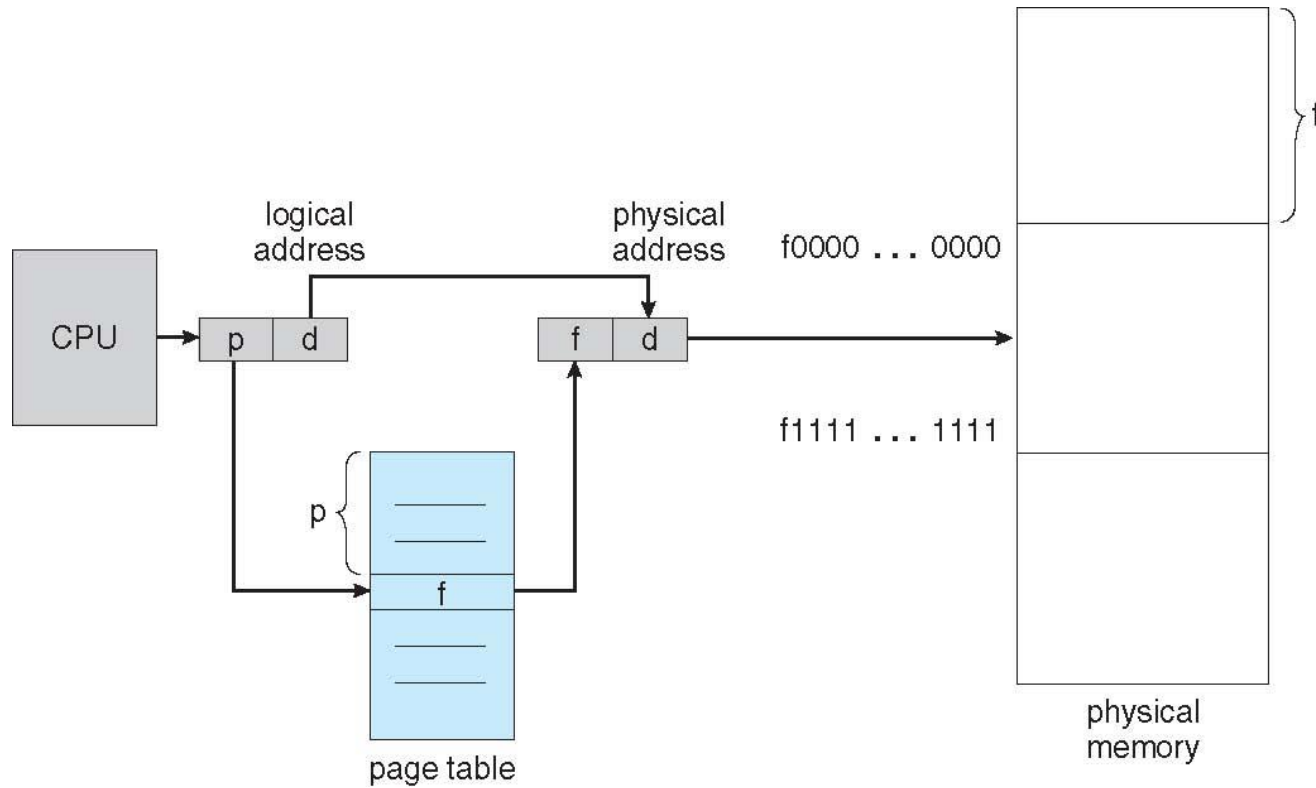
Address Translation Scheme

- Address generated by CPU is divided into:
 - **Page number** (p) – used as an index into a **page table** which contains base address of each page in physical memory
 - **Page offset** (d) – combined with base address to define the physical memory address that is sent to the memory unit



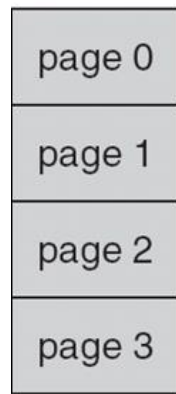


Paging Hardware

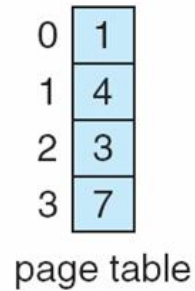




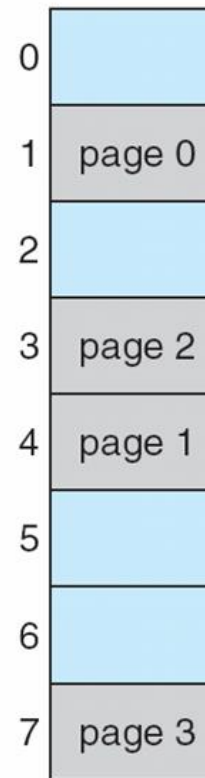
Paging Model of Logical and Physical Memory



logical
memory



frame
number



physical
memory





Paging Example

| | |
|----|---|
| 0 | a |
| 1 | b |
| 2 | c |
| 3 | d |
| 4 | e |
| 5 | f |
| 6 | g |
| 7 | h |
| 8 | i |
| 9 | j |
| 10 | k |
| 11 | l |
| 12 | m |
| 13 | n |
| 14 | o |
| 15 | p |

logical memory

| | |
|---|---|
| 0 | 5 |
| 1 | 6 |
| 2 | 1 |
| 3 | 2 |

page table

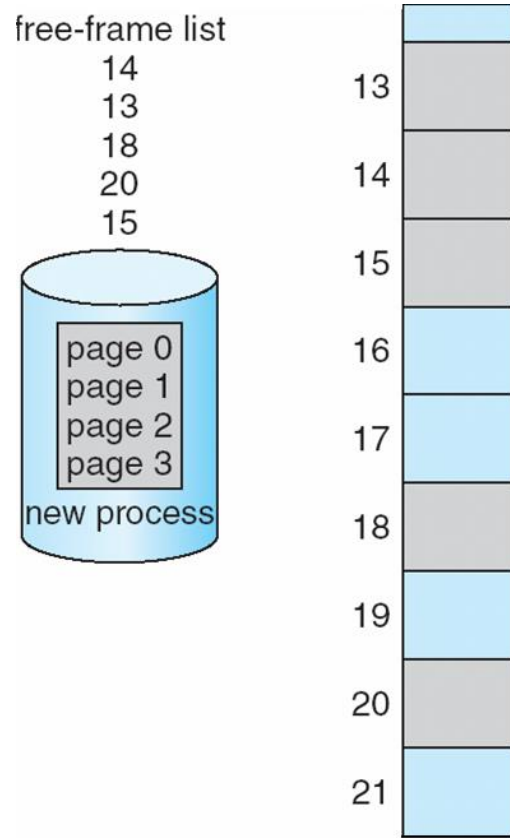
| | |
|----|------------------|
| 0 | |
| 4 | i j k l |
| 8 | m n o p |
| 12 | |
| 16 | |
| 20 | a b c d |
| 24 | e f g h |
| 28 | |

physical memory



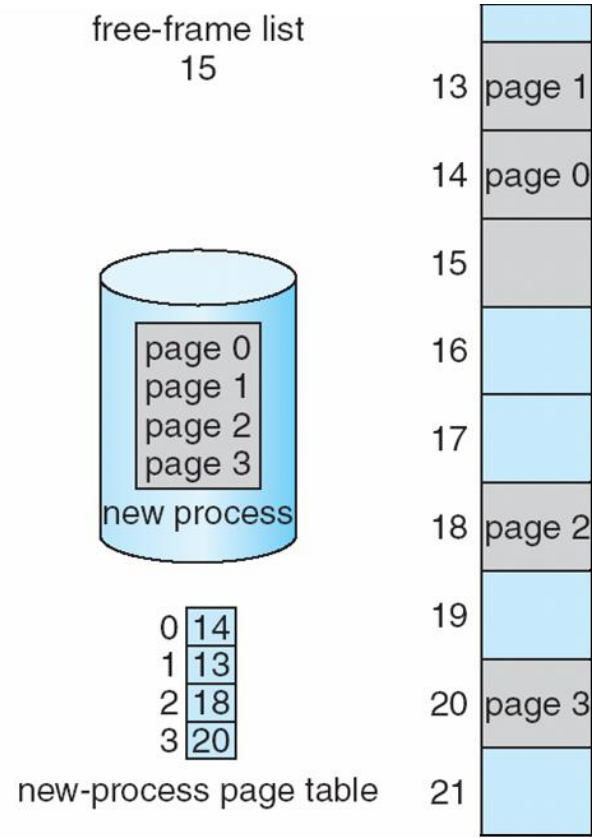


Free Frames



(a)

Before allocation



(b)

After allocation

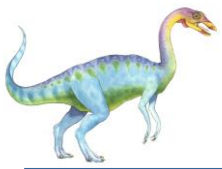




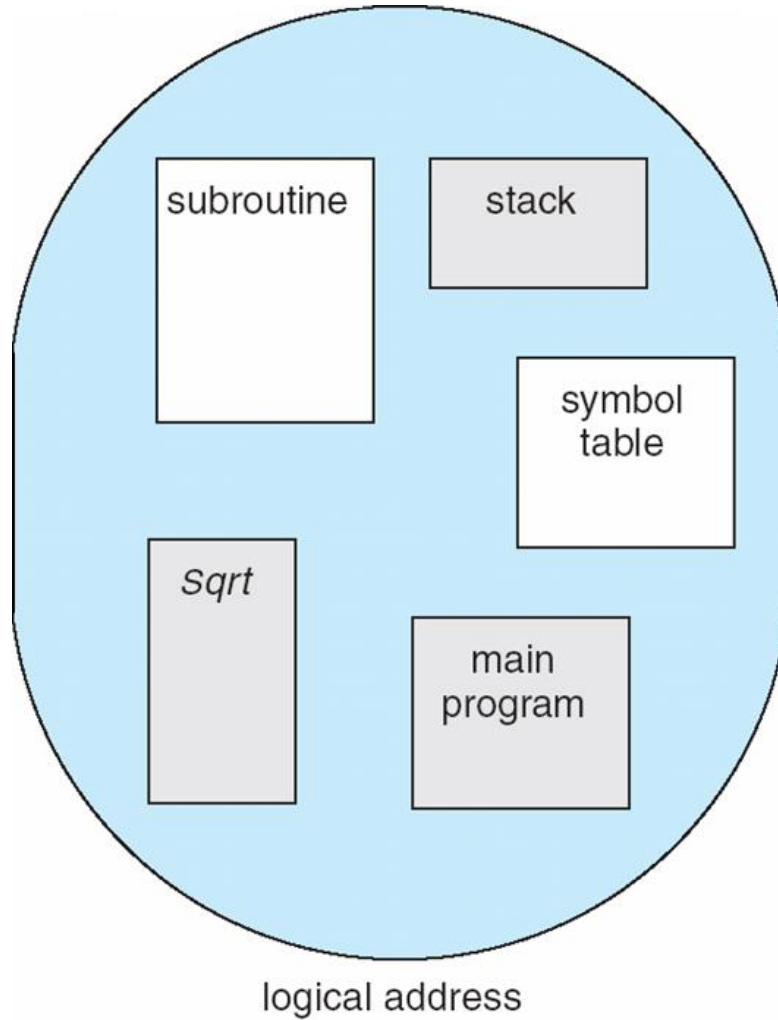
Segmentation

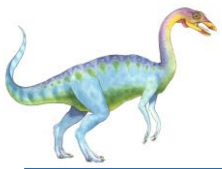
- Memory-management scheme **that supports user view of memory**
- A program is a collection of segments
 - A segment is a logical unit such as:
 - main program
 - procedure
 - function
 - method
 - object
 - local variables, global variables
 - common block
 - stack
 - symbol table
 - arrays



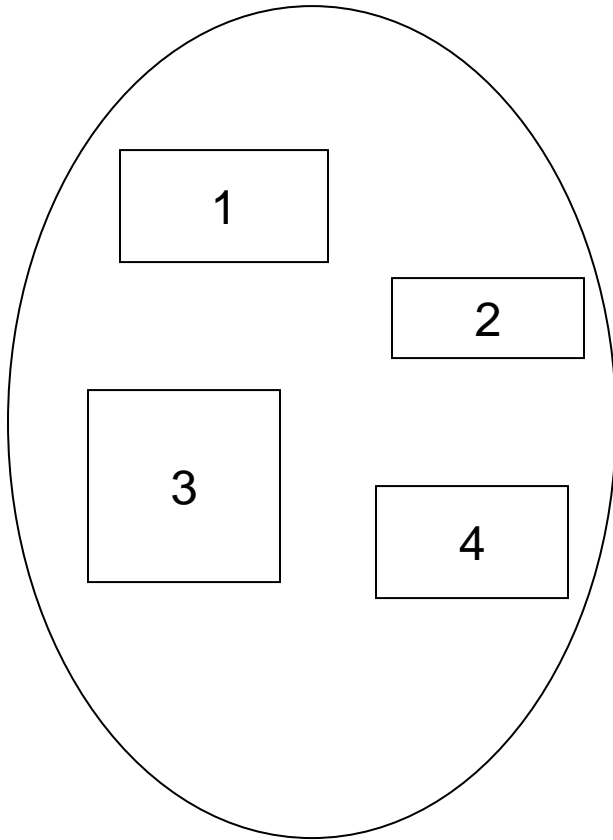


User's View of a Program

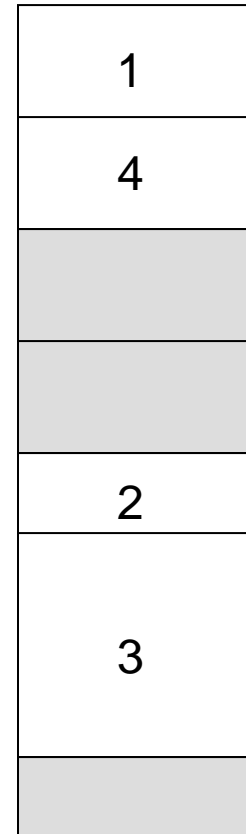




Logical View of Segmentation

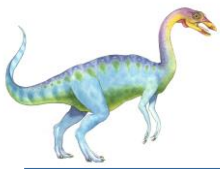


user space



physical memory space

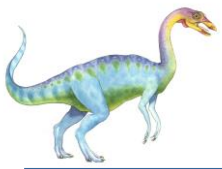




Segmentation Architecture

- Logical address consists of a two tuple:
 <segment-number, offset>,
- **Segment table** – maps two-dimensional physical addresses; each table entry has:
 - **base** – contains the starting physical address where the segments reside in memory
 - **limit** – specifies the length of the segment
- **Segment-table base register (STBR)** points to the segment table's location in memory
- **Segment-table length register (STLR)** indicates number of segments used by a program;
 segment number **s** is legal if **s** < **STLR**

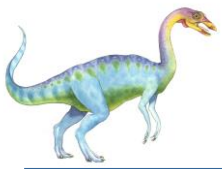




Segmentation Architecture (Cont.)

- Protection
 - With each entry in segment table associate:
 - ▶ validation bit = 0 \Rightarrow illegal segment
 - ▶ read/write/execute privileges
- Protection bits associated with segments; **Since segments vary in length, memory allocation is a dynamic storage-allocation problem**





Segmentation Hardware

