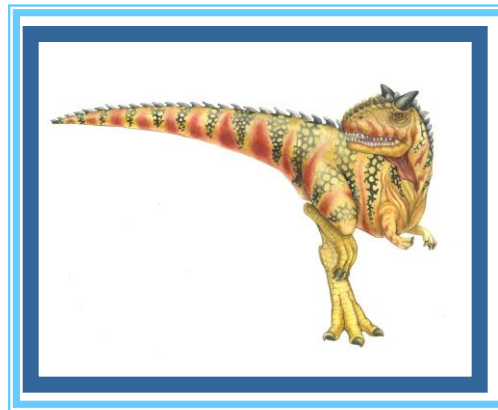


# Chapter 8: Main Memory

---





# Background

---

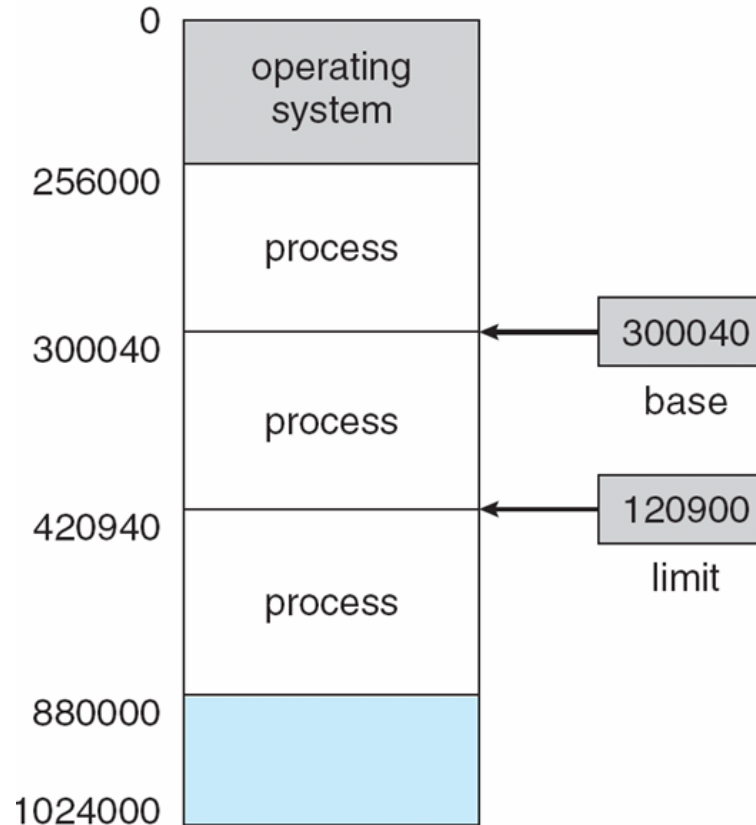
- Program must be brought (from disk) into memory and placed within a process for it to be run
  - Main memory and registers are only storage CPU can access directly
- Register access in one CPU clock (or less), Main memory can take many cycles
  - **Cache** sits between main memory and CPU registers
- Protection of memory required to ensure correct operation





# Base and Limit Registers

- A pair of **base** and **limit** registers define the logical address space





# Binding of Instructions and Data to Memory

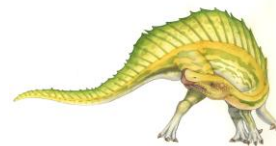
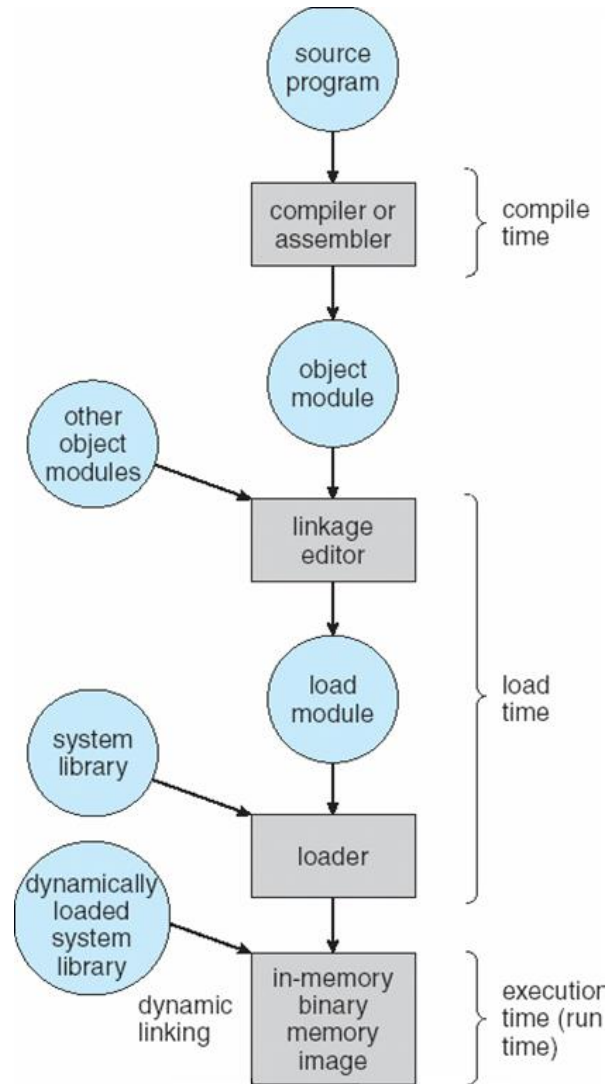
---

- 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

---

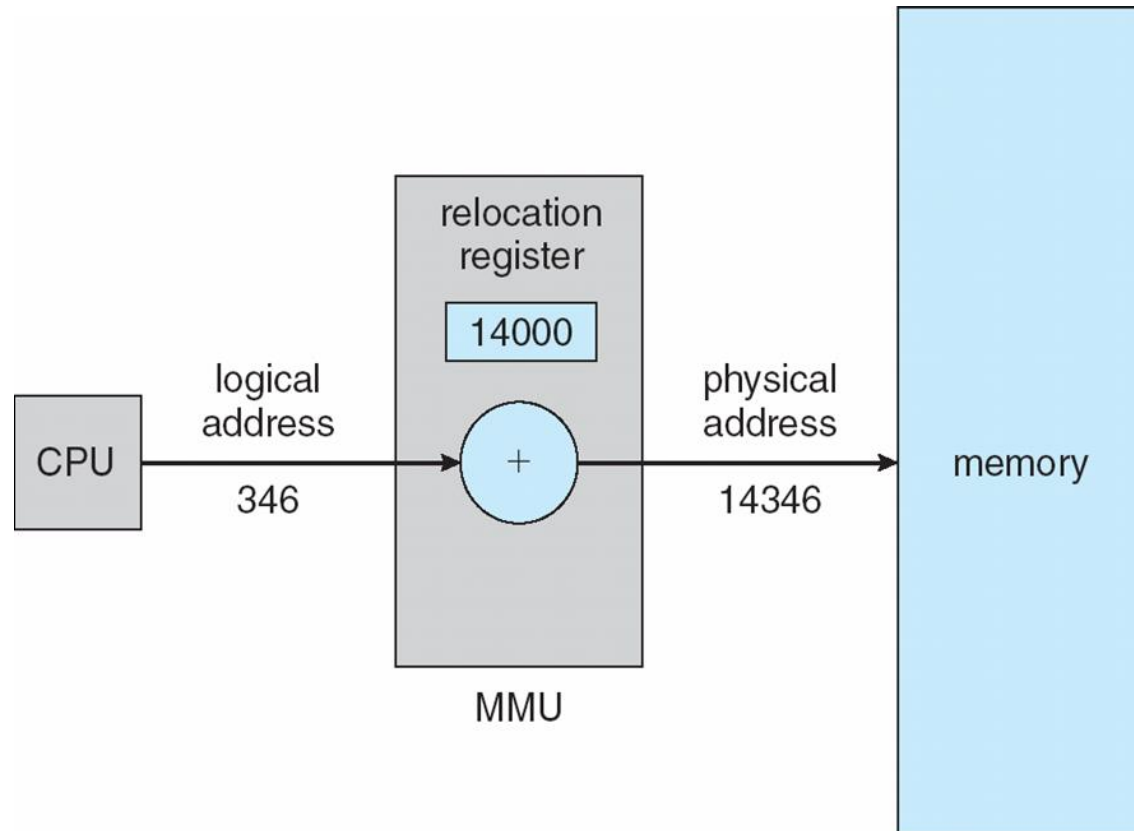
- The concept of a logical address space that is bound to a separate **physical address space** is central to proper memory management
  - **Logical address** – generated by the CPU; also referred to as **virtual address**
  - **Physical address** – address seen by the memory unit
- Logical and physical 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





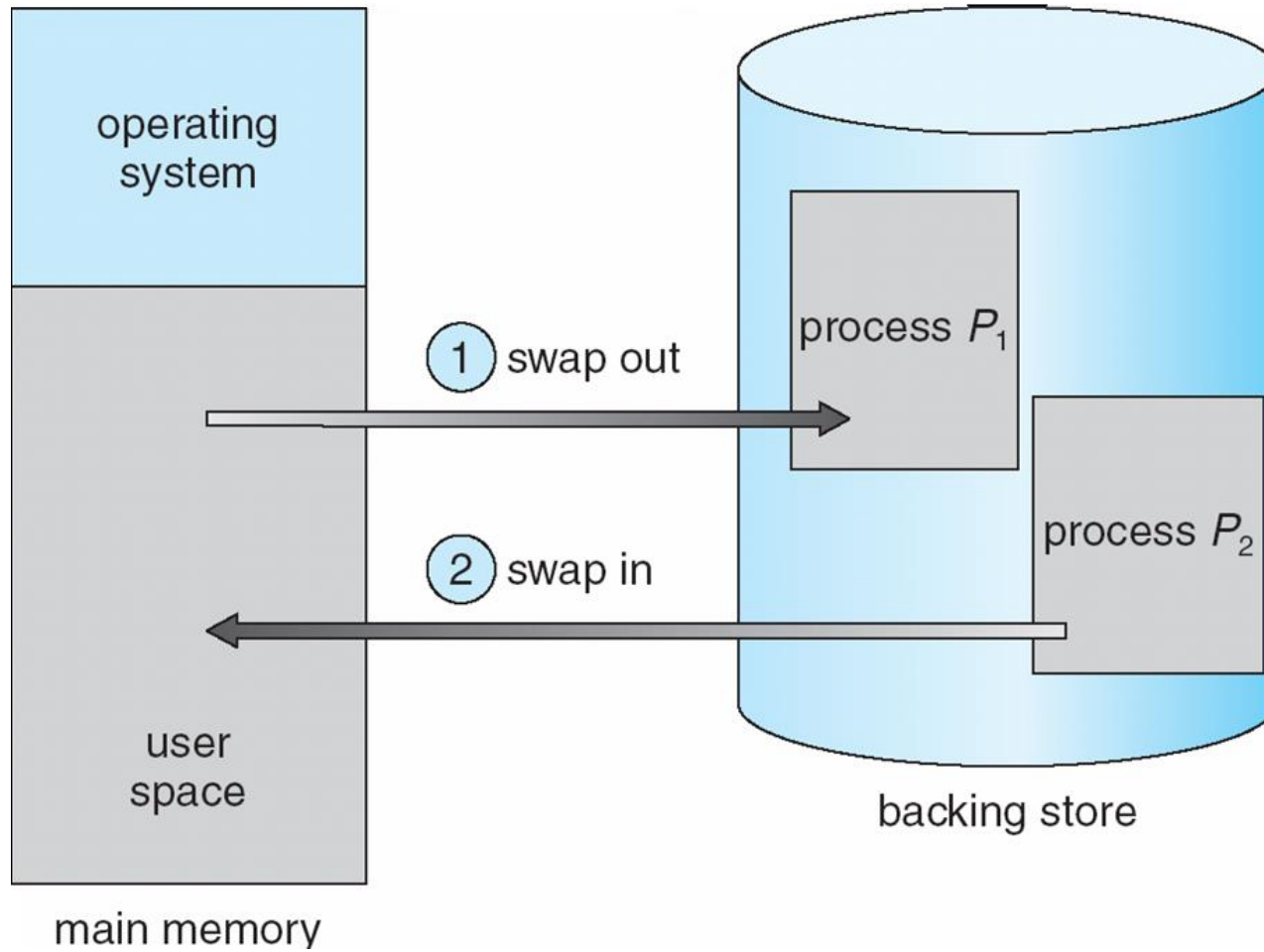
# Memory-Management Unit (MMU)

- 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





# Schematic View of Swapping







# Contiguous Allocation

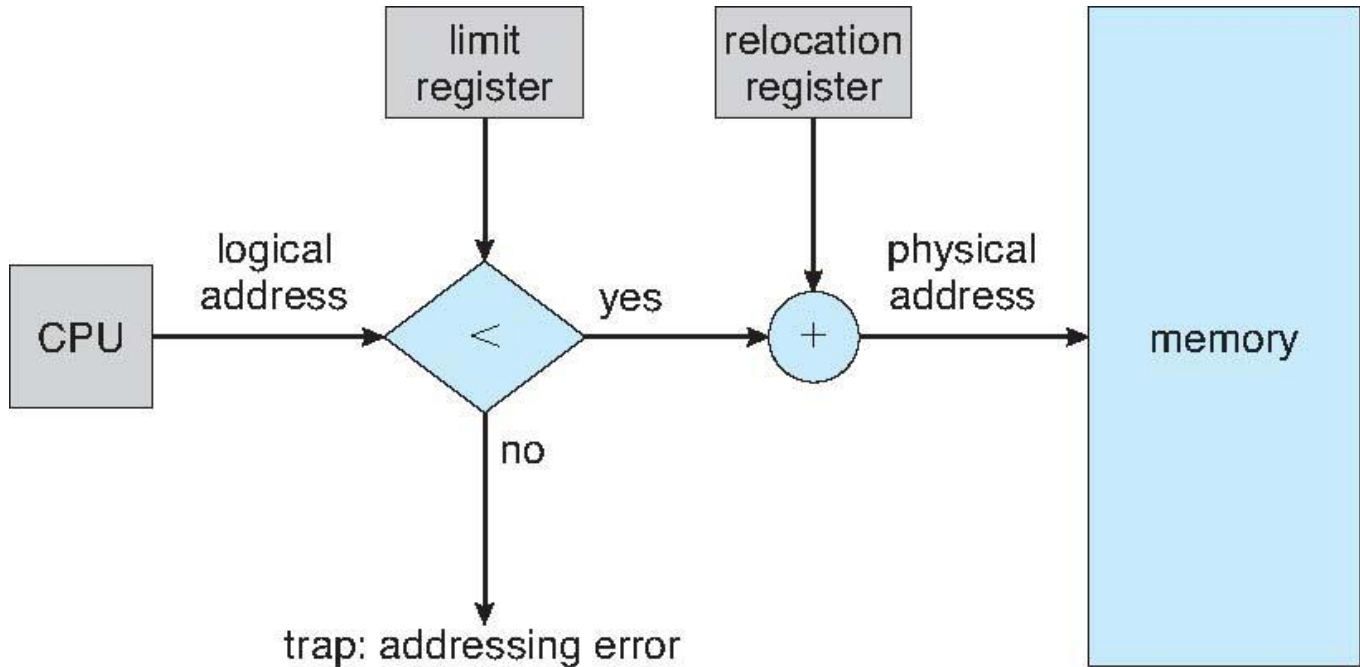
---

- Main memory usually into two partitions:
  - Resident operating system, usually held in low memory with interrupt vector
  - User processes then held in high memory
  
- 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*





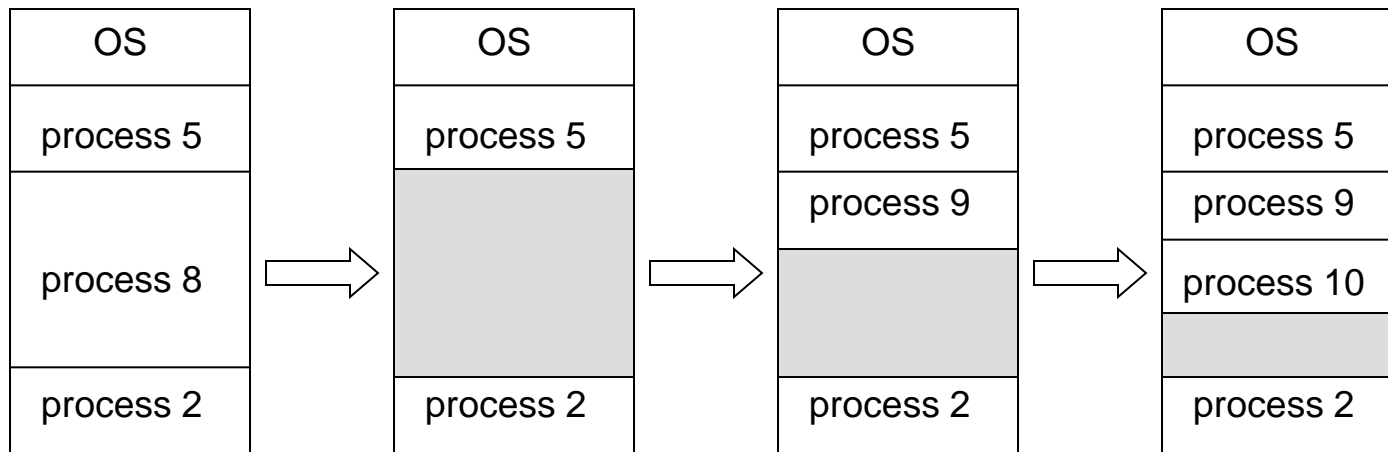
# Hardware Support for Relocation and Limit Registers





# 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





# 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





# Paging

---

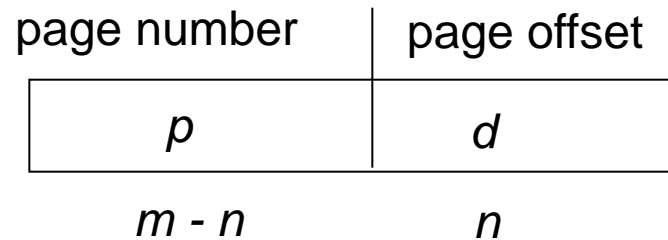
- Logical address space of a process can be noncontiguous; process is allocated physical memory whenever the latter is available
  - Divide physical memory into fixed-sized blocks called **frames** (size is power of 2, between 512 bytes and 8,192 bytes)
  - Divide logical memory into blocks of same size 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
  - 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

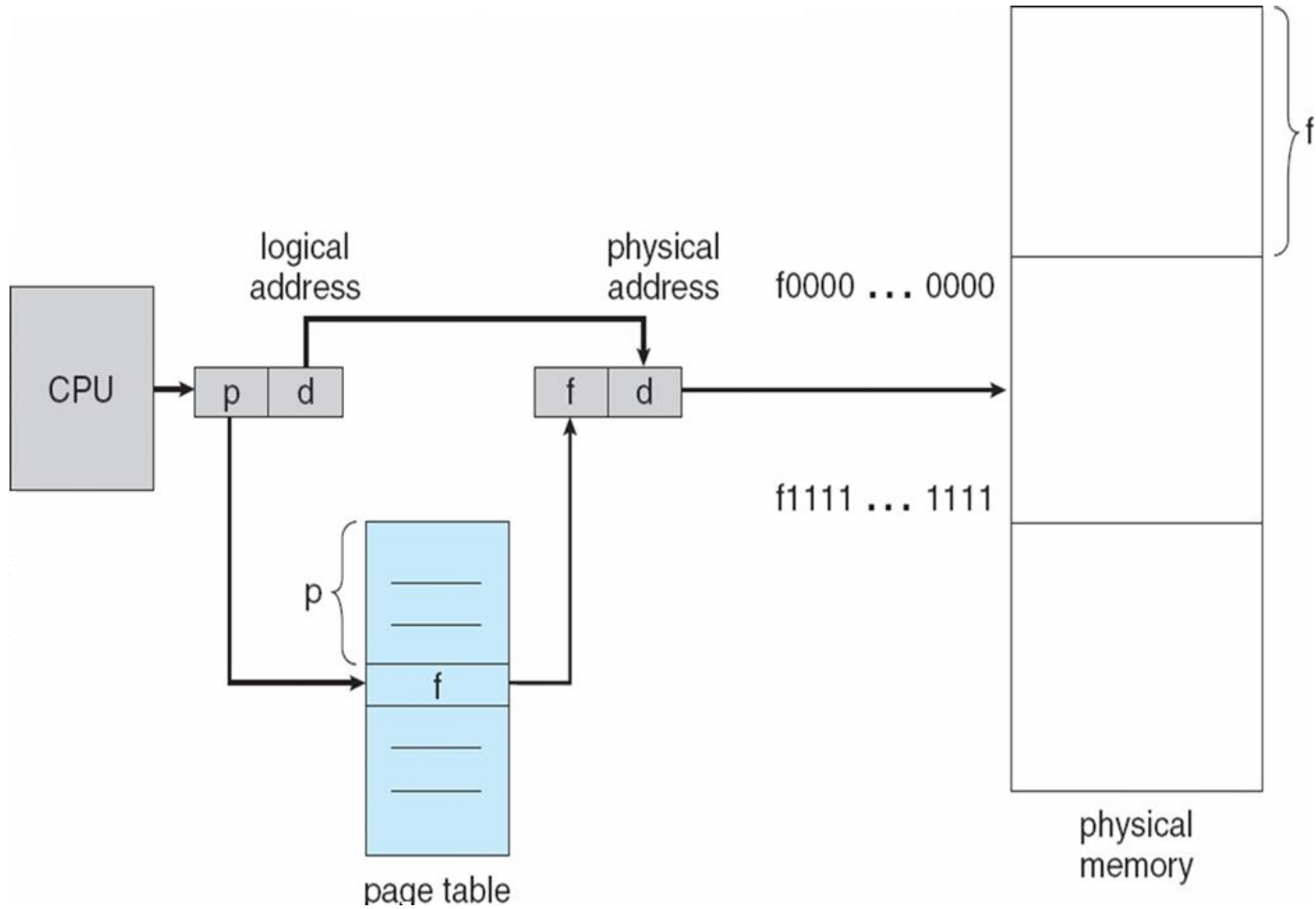


- For given logical address space  $2^m$  and page size  $2^n$





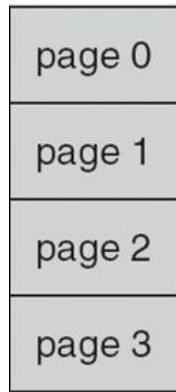
# Paging Hardware



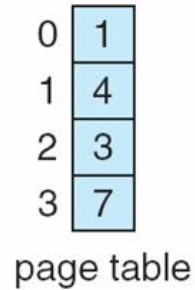




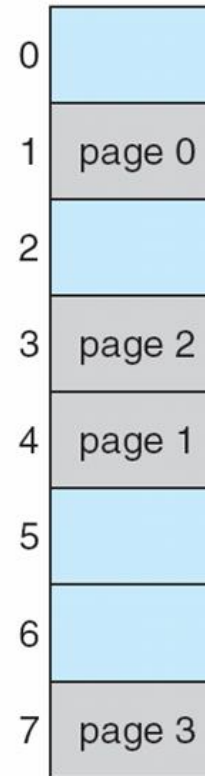
# Paging Model of Logical and Physical Memory



logical  
memory



frame  
number

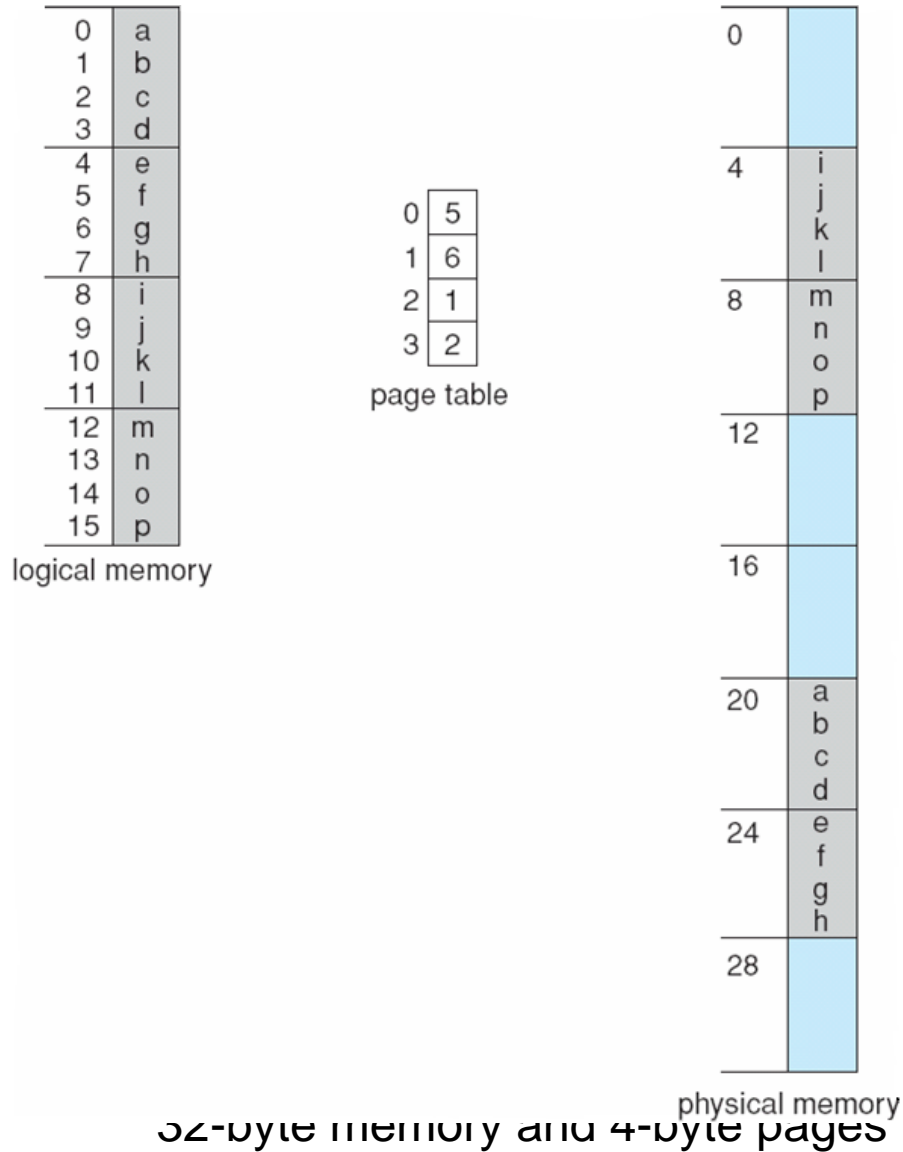


physical  
memory



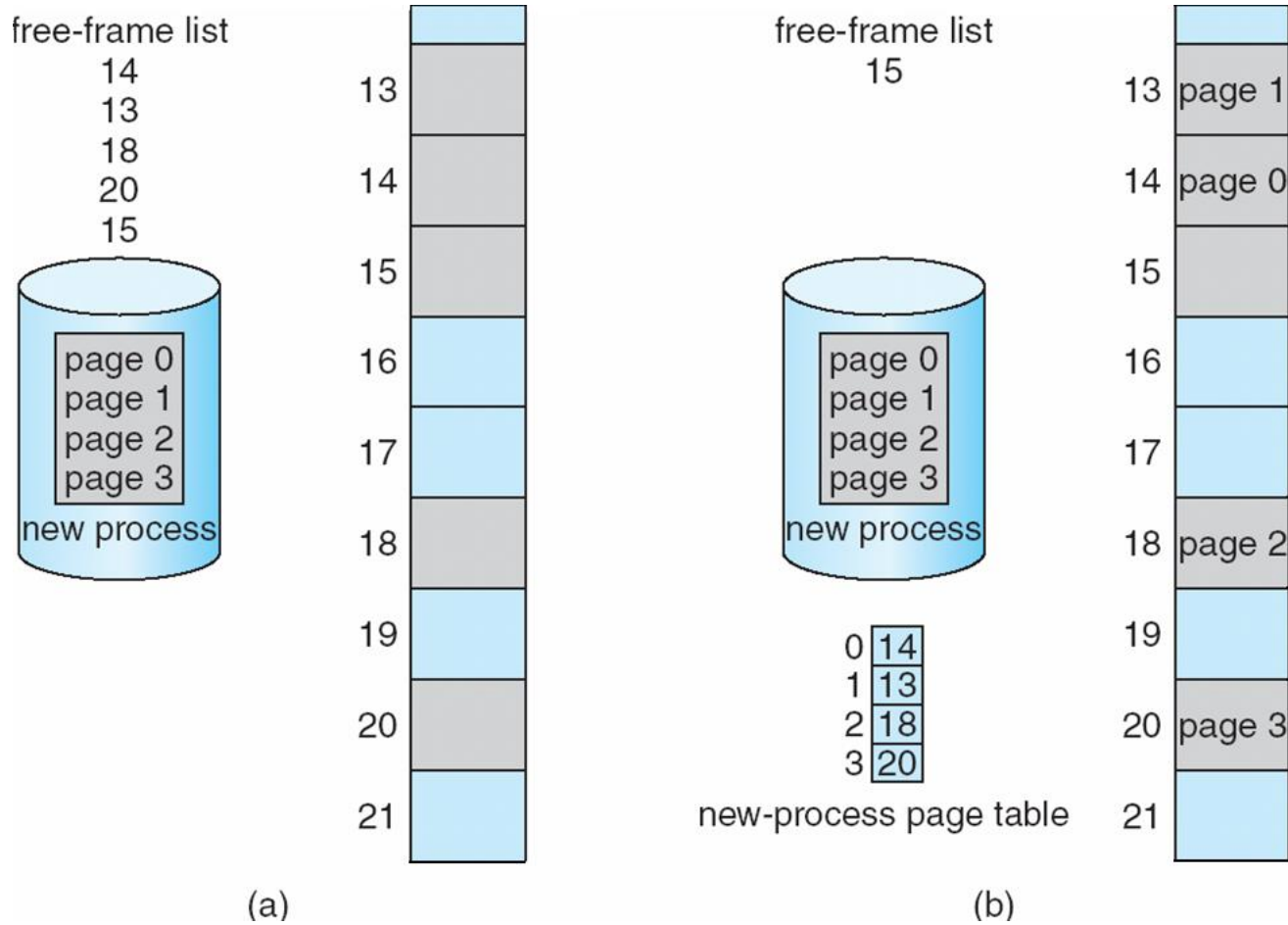


# Paging Example





# Free Frames



Before allocation

After allocation

