CSI 4500 Operating Systems

Main Memory

Up to This Point

- Threads provide the illusion of an infinite number of CPUs
  - On a single processor machine
- Memory management provides a different set of illusions
  - Protected memory
  - Infinite amount of memory
  - Transparent sharing

Review

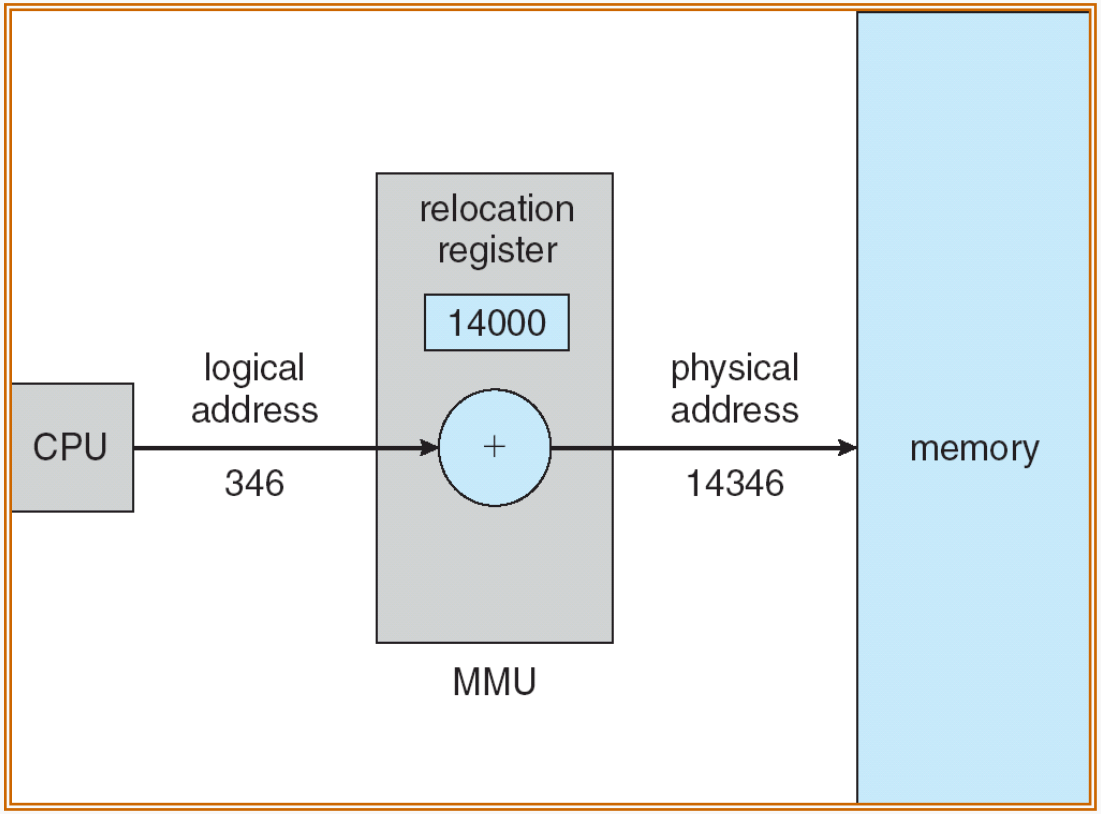
- **Main memory** and **registers** are only storage CPU can access directly
- Access speed
  - **Register** access in one CPU clock (or less)
  - **Main memory** can take many cycles
  - **Cache** sits between main memory and CPU registers
- Program must be brought (from disk) into memory and placed within a **process** for it to be runof ready-to-run processes which have memory images on disk
- System maintains a **ready queue**
- Protection of memory required to ensure correct operation

Logical vs. Physical Address Space

- **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
  - User program deals with logical addresses, it never sees real physical addresses

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.



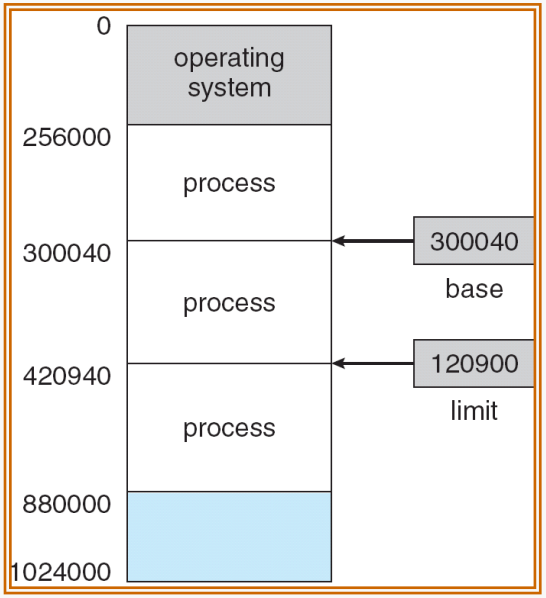
Physical vs. Virtual Memory

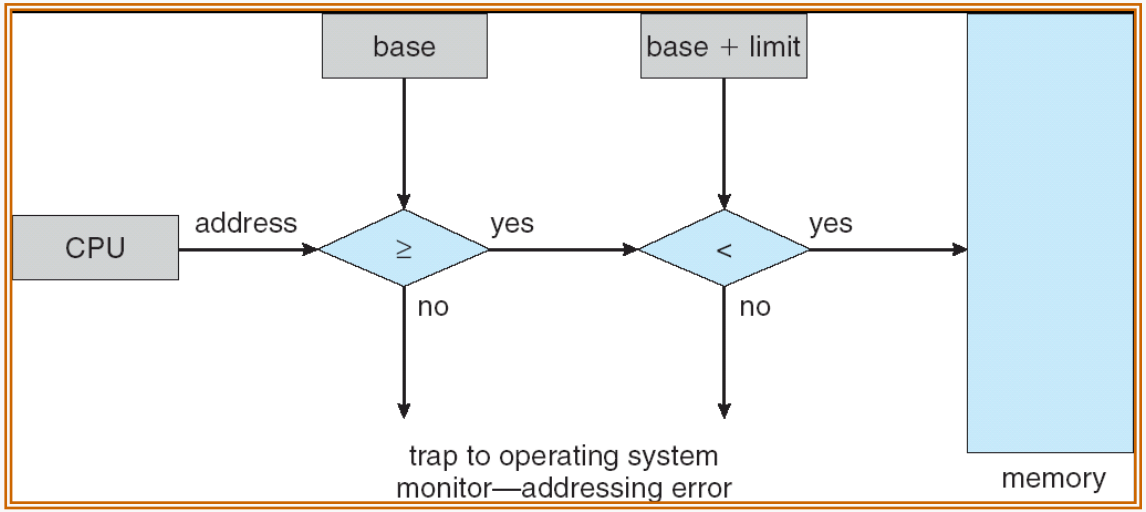
Physical memory	Virtual memory
No protection	Each process isolated from all others and from the OS
Limited size	Illusion of infinite memory
Sharing visible to processes	Each process cannot tell if memory is shared

- **Memory protection** keeps user programs from crashing one another and the OS
- Two hardware-supported mechanisms
  - Address Translation
  - Dual-mode Operation

Address Mapping and Protection

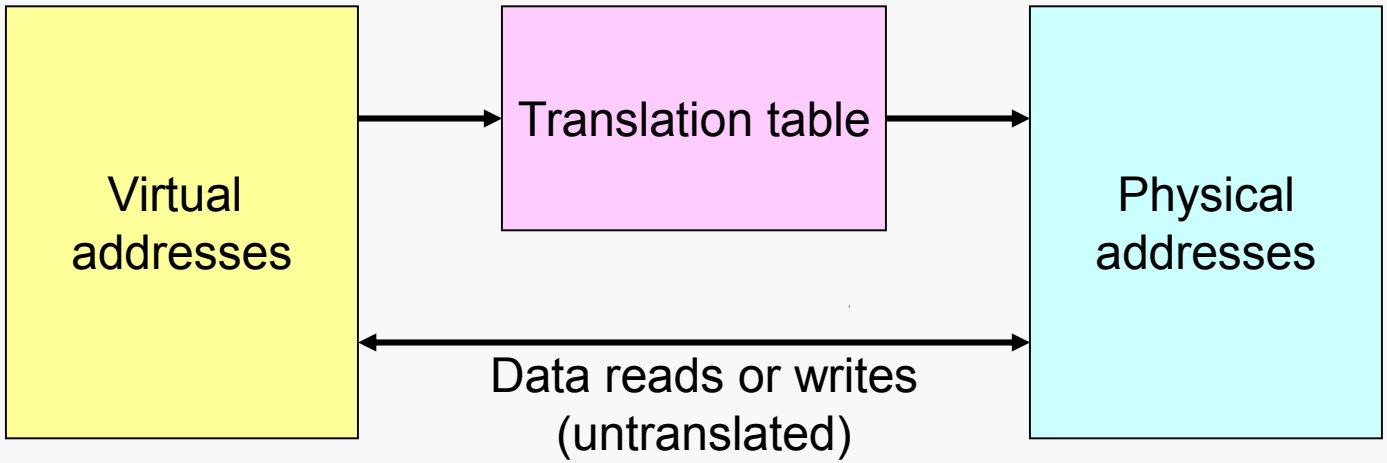
- Main memory usually into two partitions:
  - Resident operating system, usually held in low memory with interrupt vector
  - User processes then held in high memory
- Relocation registers used to protect user processes from each other, and from changing operating-system code and data
  - Base register contains value of smallest physical address
  - Limit register contains range of logical addresses





Address Translation

- Recall that each process is associated with an **address space**, or all the *physical* addresses a process can touch
- However, each process believes that it owns the entire memory, starting with the *virtual* address 0
- The missing piece is a translation table to translate every memory reference from virtual to physical addresses



- Translation provides protection
  - Processes cannot talk about other processes' addresses, nor about the OS addresses
  - OS uses physical addresses directly (No translations)

Dual-Mode Operation Revisited

- Translation tables offer protection if they cannot be altered by applications
- An application can only touch its address space under the user mode
- Hardware requires the CPU to be in the kernel mode to modify the address translation tables

Switching from the Kernel to User Mode

- To run a user program, the kernel
  - Creates a process and initialize the address space
  - Loads the program into the memory
  - Initializes translation tables
  - Sets the hardware pointer to the translation table
  - Sets the CPU to user mode
  - Jumps to the entry point of the program

Switching from User Mode to Kernel Mode

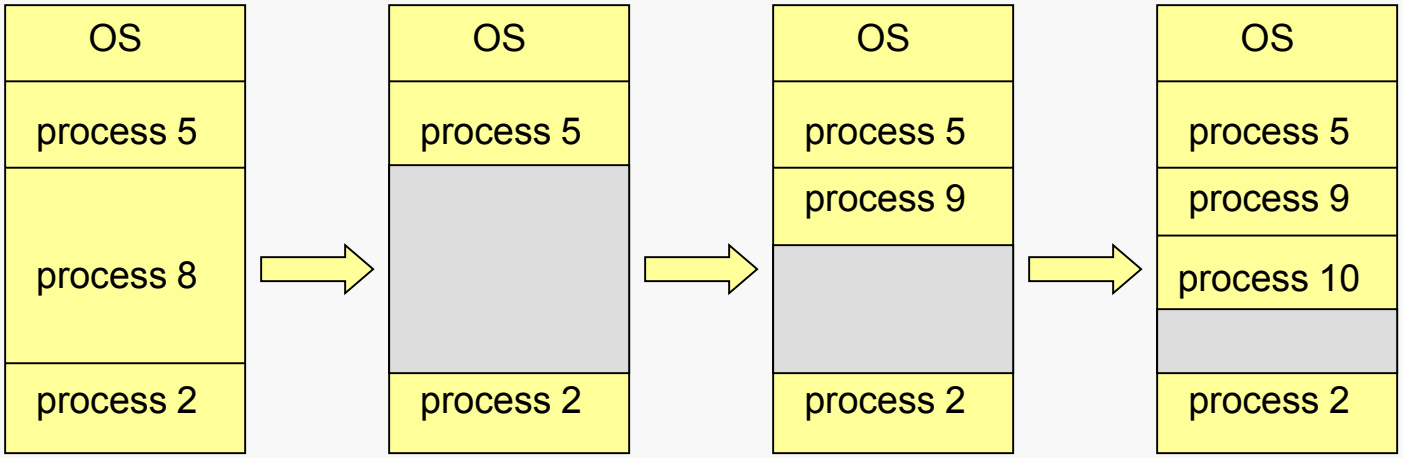
- Voluntary
  - **System calls:** a user process asks the OS to do something on the process's behalf
- Involuntary
  - Hardware interrupts (e.g., I/O)
  - Program exceptions (e.g., segmentation fault)
- For all cases, hardware atomically performs the following steps
  - Sets the CPU to kernel mode
  - Saves the current program counter
  - Jumps to the handler in the kernel
    - The handler saves old register values

Switching from User Mode to Kernel Mode

- Unlike context switching among threads, to switch among processes
  - Need to save and restore pointers to translation tables
- To resume process execution
  - Kernel reloads old register values
  - Sets CPU to user mode
  - Jumps to the old program counter

Contiguous Allocation

- Operating system maintains information about
  - allocated partitions
  - 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

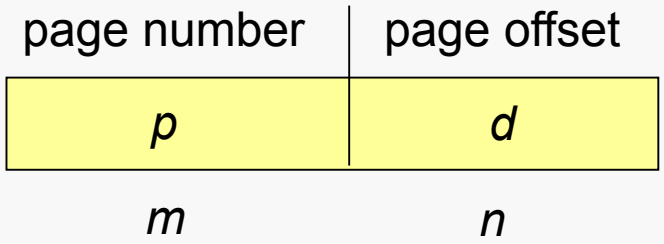
- Fragmentation Problem (complex memory allocation)
  - Not every process is the same size.
- **Internal Fragmentation** – allocated memory may be slightly larger than requested memory
- **External Fragmentation** – total memory space exists to satisfy a request, but it is not contiguous
  - Reduce external fragmentation by compaction
  - Shuffle memory contents to place all free memory together in one large block
- Another solution: to allow noncontiguous logical address space:
  - paging and segmentation.

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**
- OS keeps track of all free frames
- Set up a page table to translate logical to physical addresses
- To run a program of size ***n*** pages, need to find *n* free frames and load program
  - Internal fragmentation

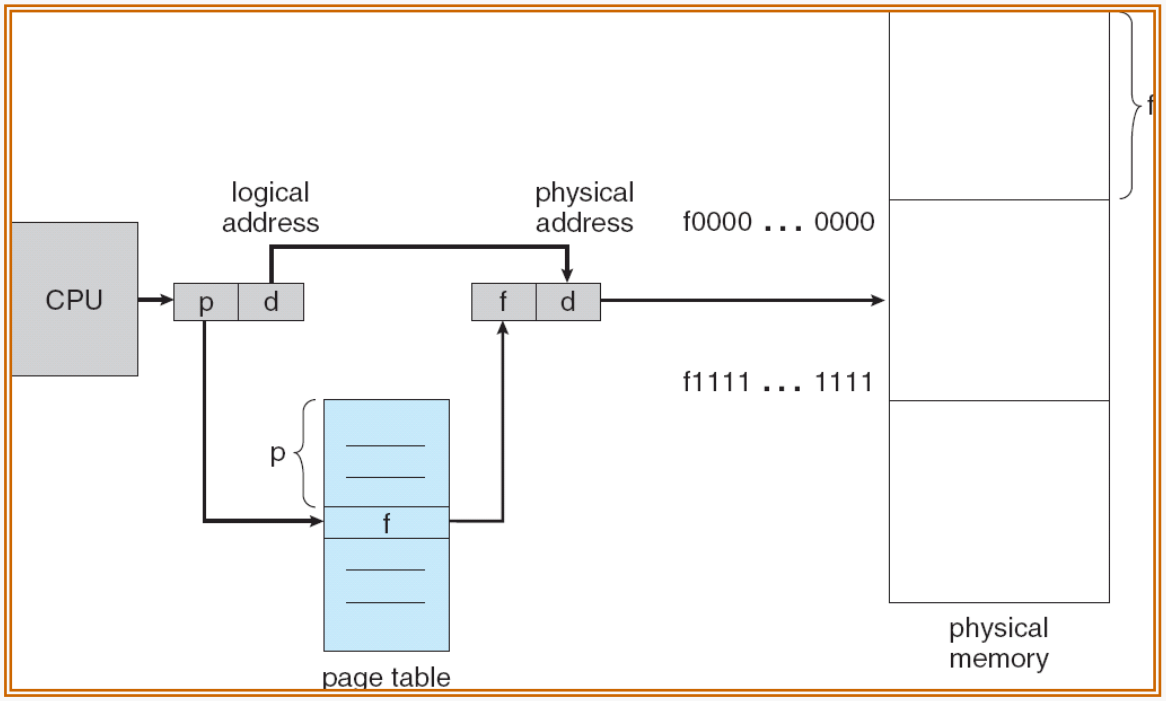
Address Translation Scheme

- 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

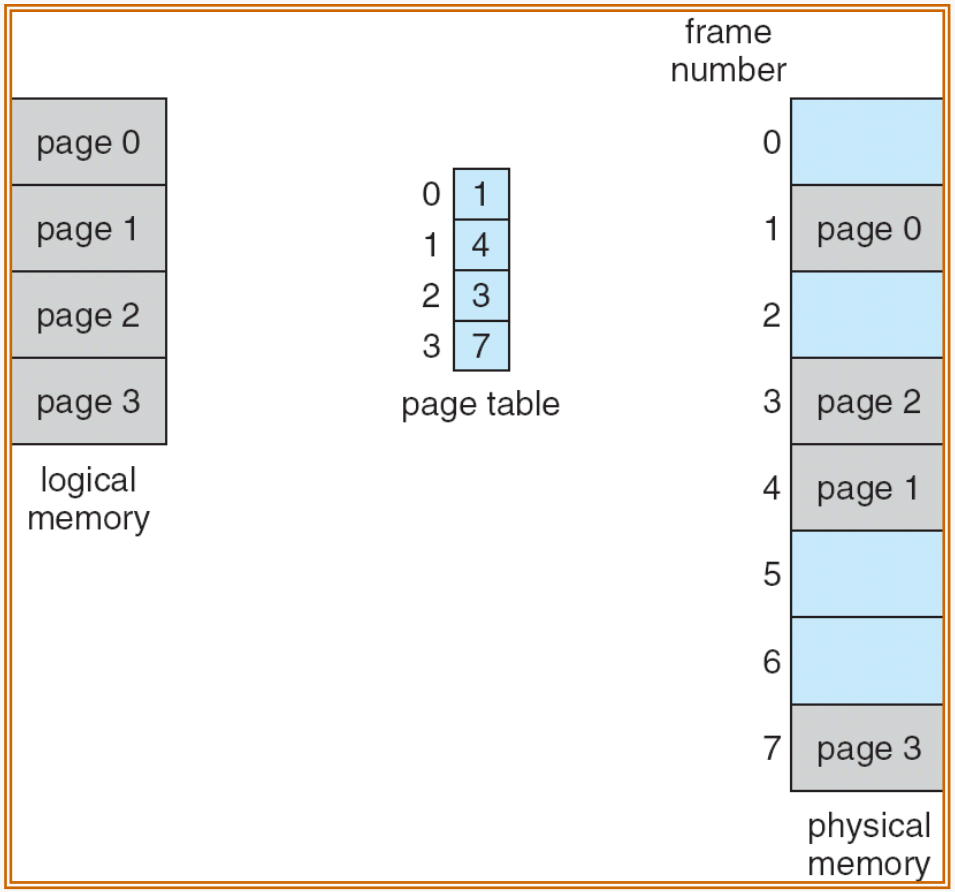


  - For given logical address space 2^{m+n} *and page size* 2^{n}

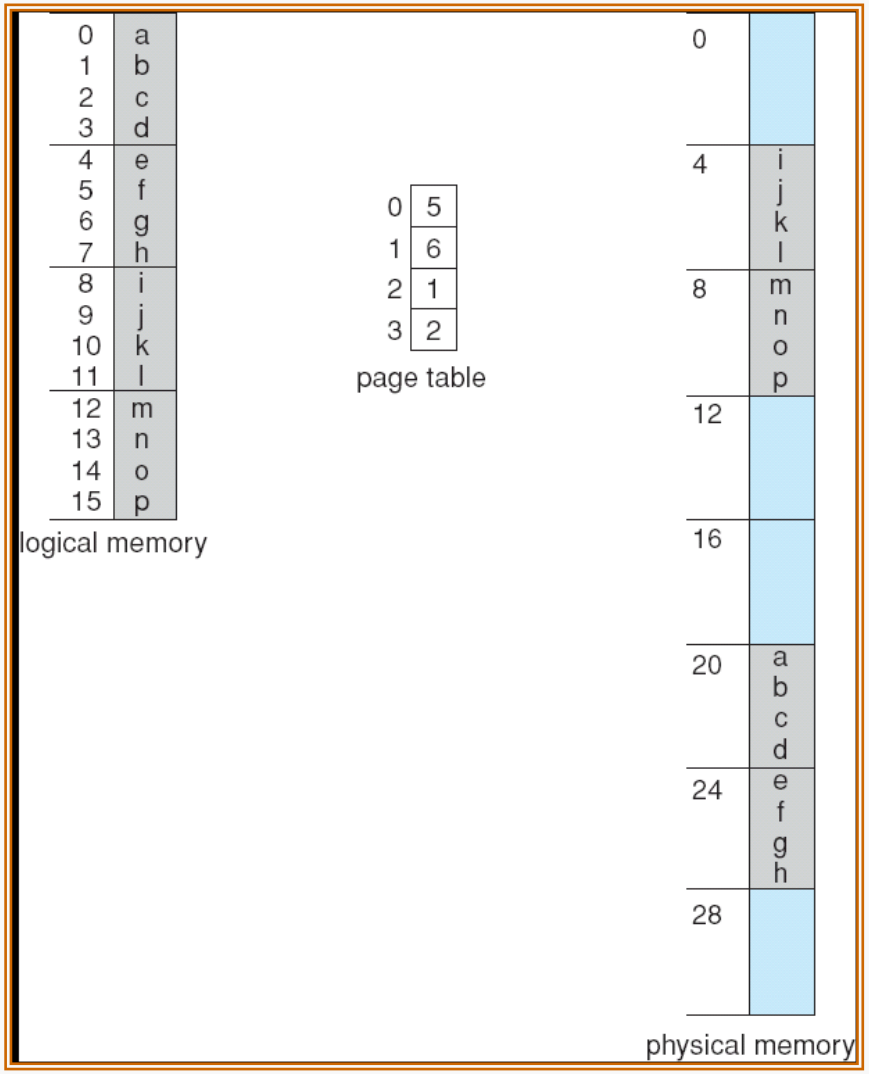
Hardware Support for Paging



Paging Model of Logical and Physical Memory

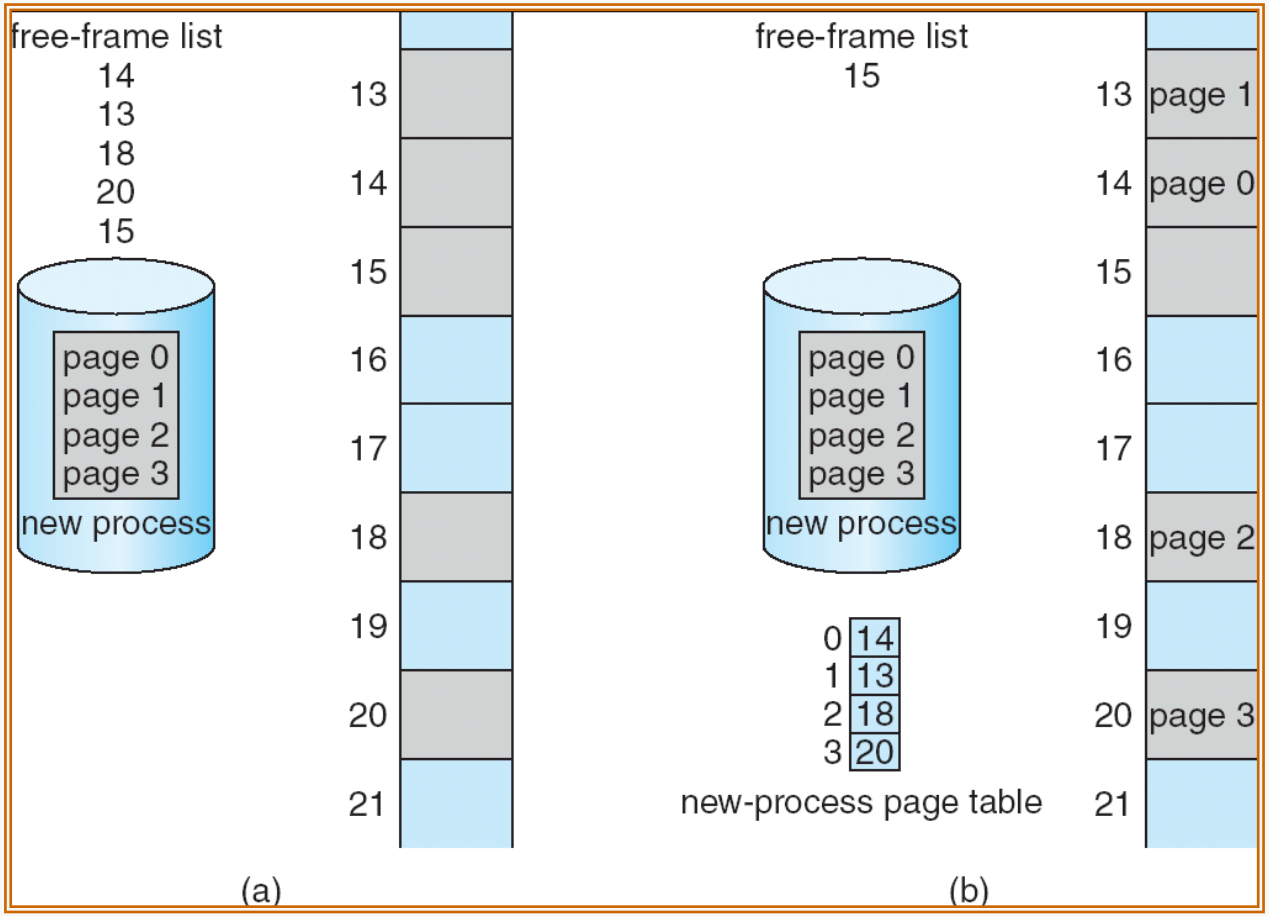


Paging Exercise



32-byte memory and 4-byte pages

Free Frames



Before allocation

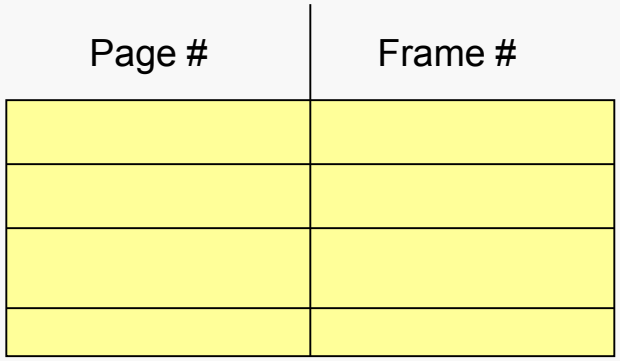
After allocation

Implementation of Page Table

- Page table is kept in main memory
- **Page-table base register (PTBR)** points to the page table
- **Page-table length register (PRLR)** indicates size of the page table
- In this scheme every data/instruction access requires two memory accesses.
  - One for the page table and one for the data/instruction.
  - Use of a special fast-lookup hardware cache called **associative memory** or **translation look-aside buffers (TLBs)**
  - Some TLBs store **address-space identifiers (ASIDs)** in each TLB entry – uniquely identifies each process to provide address-space protection for that process

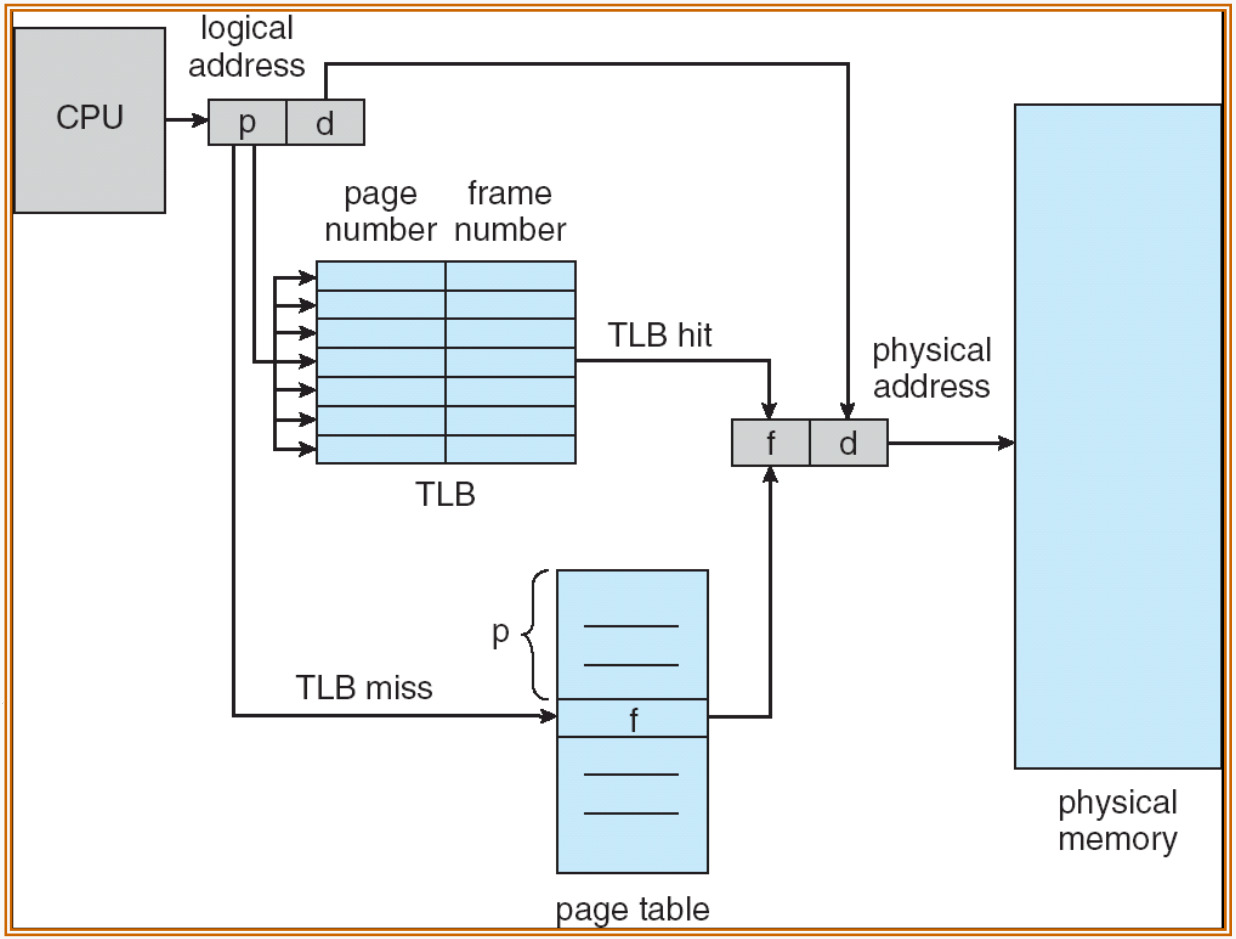
Associative Memory

- Associative memory – parallel search



- Address translation (p, d)
  - If p is in associative register, get frame # out
  - Otherwise get frame # from page table in memory

Paging Hardware With TLB



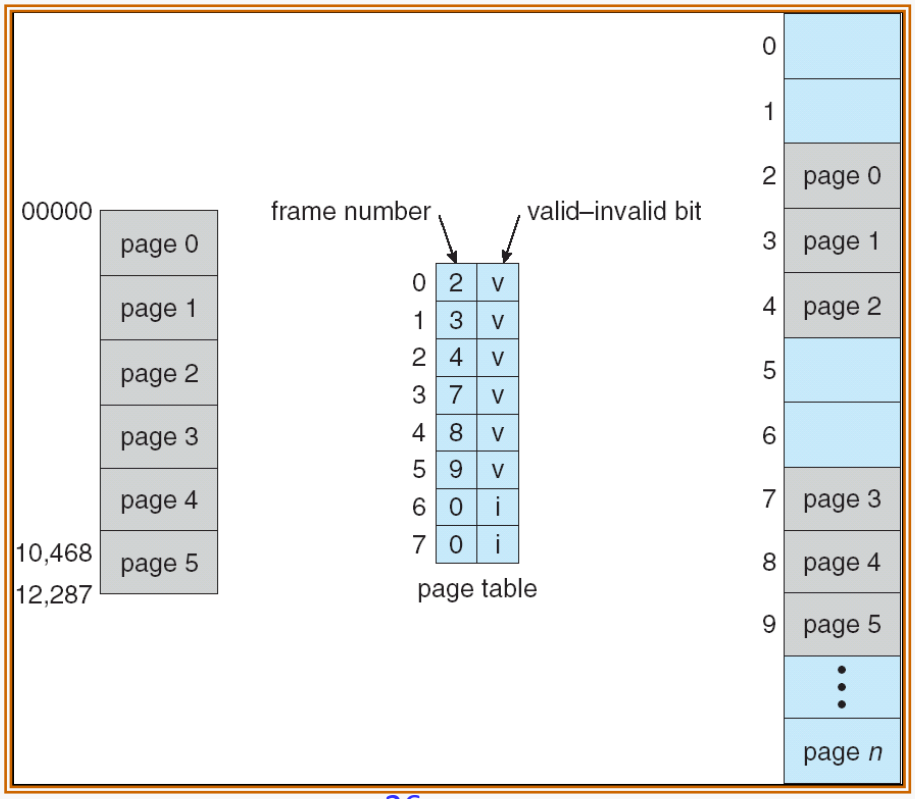
Effective Access Time

- Associative Lookup = a time unit
- Assume memory cycle time is b
- **Hit ratio** – percentage of times that a page number is found in the associative registers.
  - ratio related to number of associative registers
- Hit ratio = p
- **Effective Access Time** (EAT)

$$EAT = (a + b)\,p + (a + b + b)(1 - p)$$

Memory Protection

- Associating protection bit with each frame
- **Valid-invalid** bit attached to each entry in the page table:
  - "valid" indicates that the associated page is in the process' logical address space, and is thus a legal page
  - "invalid" indicates that the page is not in the process' logical address space



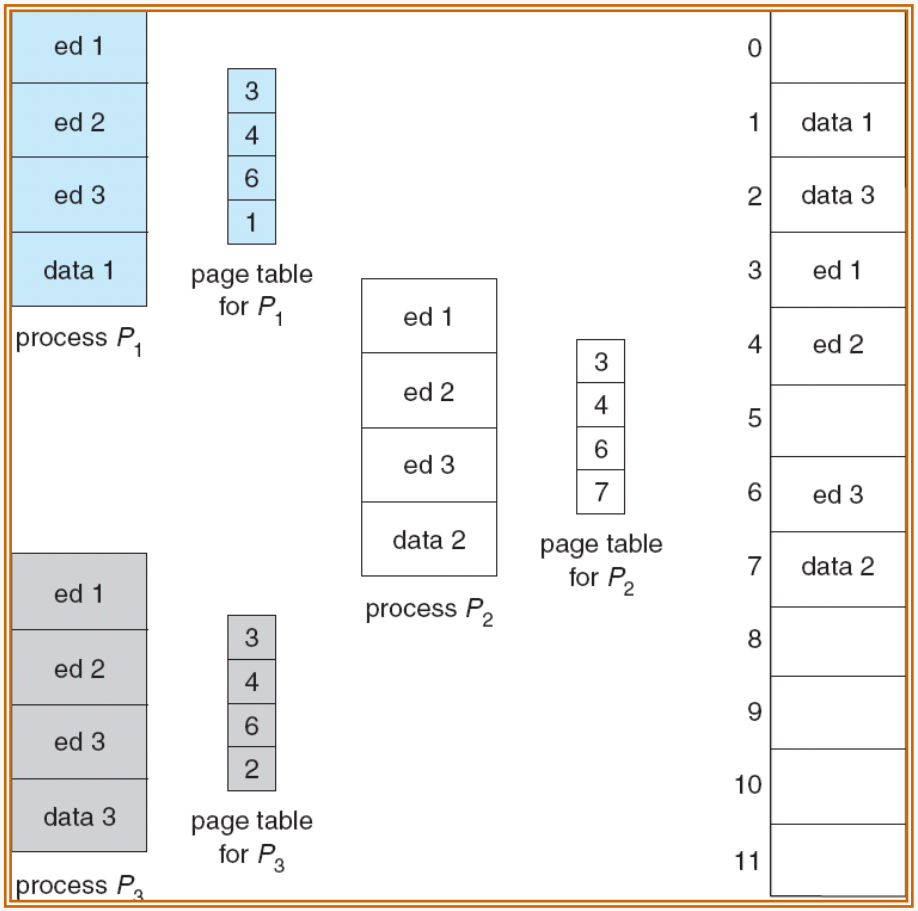
Shared Pages

Shared code

- One copy of read-only (reentrant) code shared among processes (i.e., text editors, compilers, window systems).
- Shared code must appear in same location in the logical address space of all processes

Private code and data

- Each process keeps a separate copy of the code and data
- The pages for the private code and data can appear anywhere in the logical address space

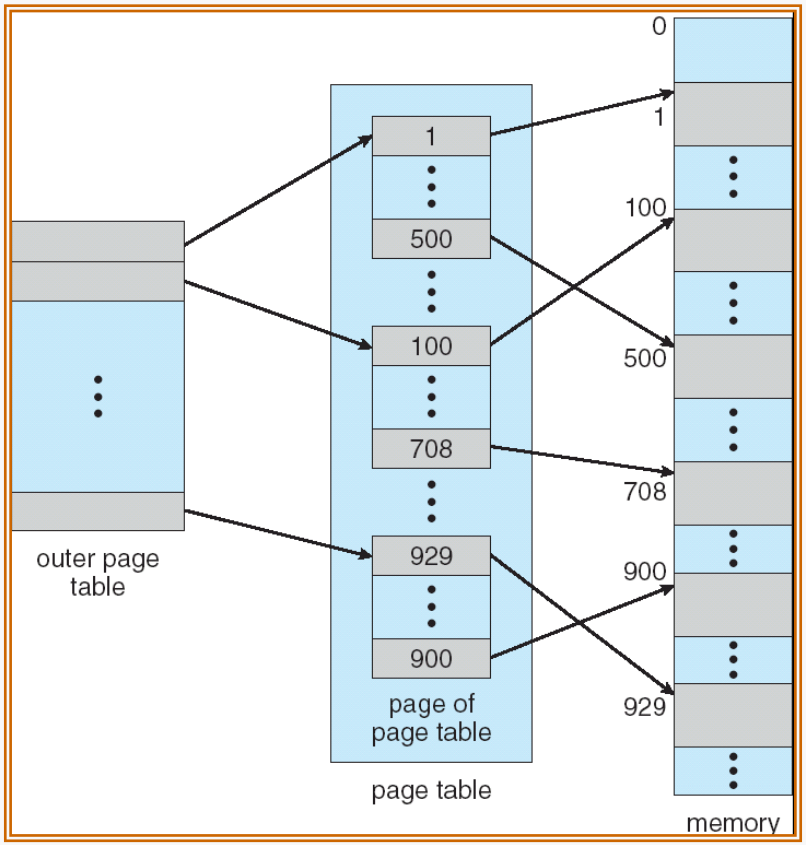


Structure of the Page Table

- Hierarchical Paging
- Hashed Page Tables
- Inverted Page Tables

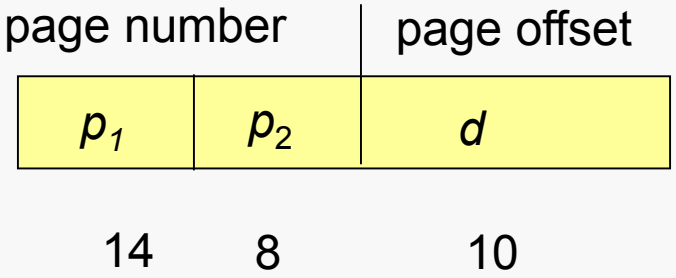
Hierarchical Page Tables

- Break up the logical address space into multiple page tables
- A simple technique is a two-level page table



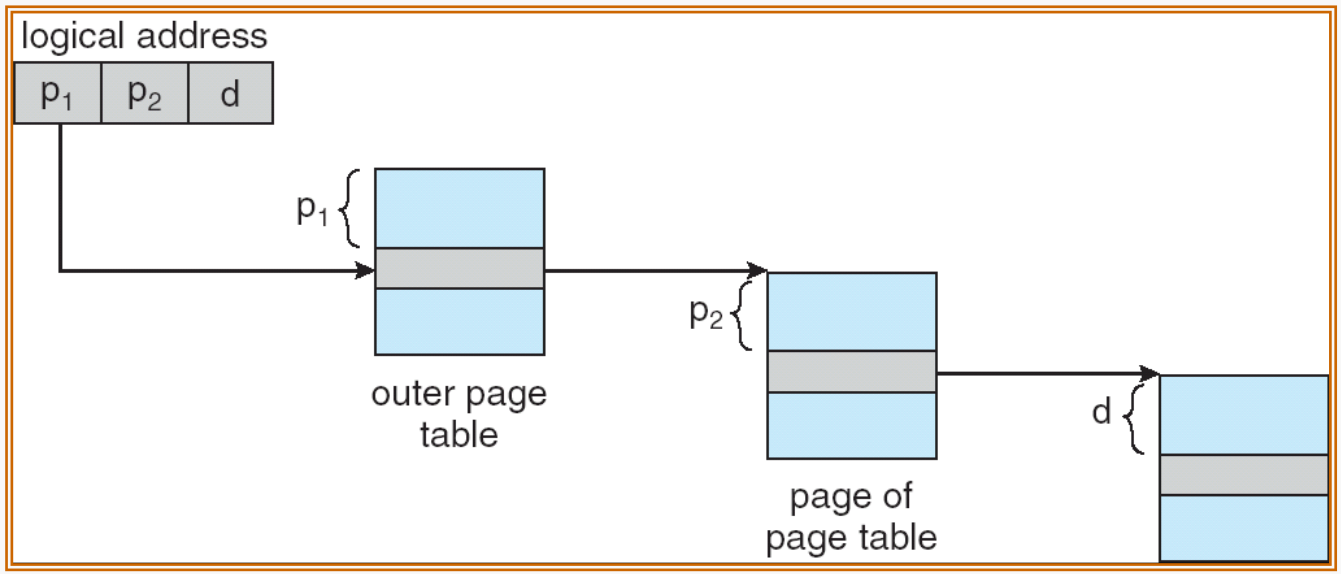
Two-Level Paging Example

- A logical address (on 32-bit machine with 1K page size) is divided into:
  - a page number consisting of 22 bits
  - a page offset consisting of 10 bits
- Since the page table is paged, the page number is further divided into:
  - a 14-bit page number and a 8-bit page offset



where p_1 is an index into the outer page table, and p_2 is the displacement within the page of the outer page table

Address-Translation Scheme



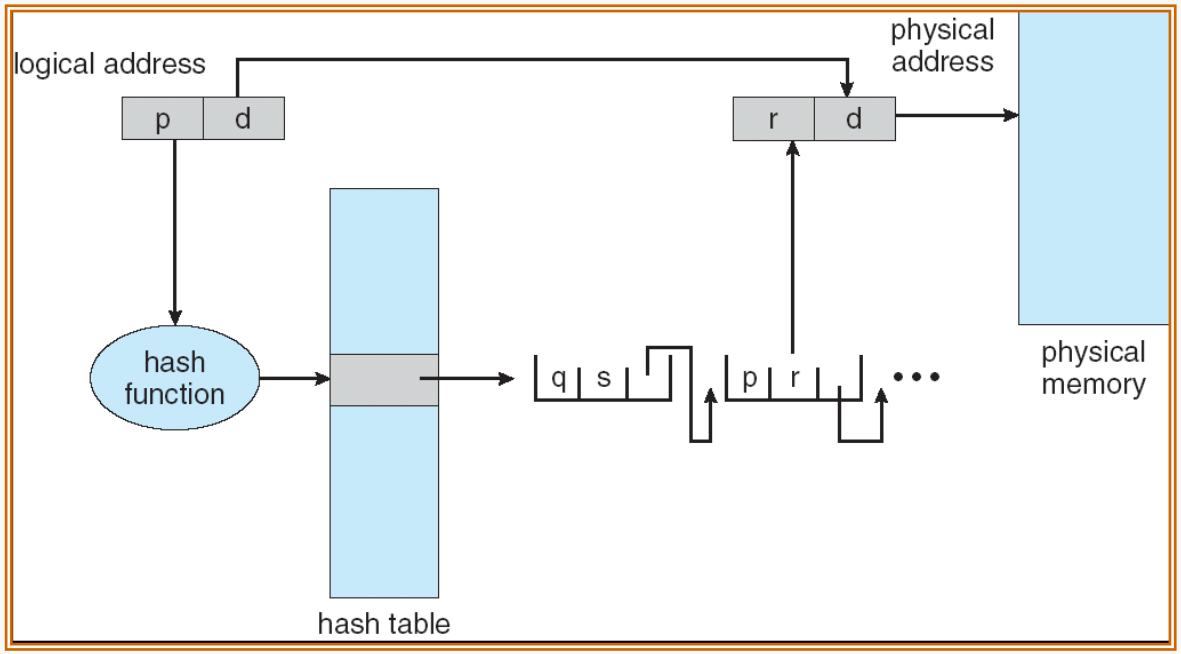
Three-level Paging Scheme

outer page	inner page	offset
p_1	p_2	d
42	10	12

2nd outer page	outer page	inner page	offset
p_1	p_2	p_3	d
32	10	10	12

Hashed Page Tables

- Common in address spaces > 32 bits
- The virtual page number is hashed into a page table.
  - This page table contains a chain of elements hashing to the same location.
  - Virtual page numbers are compared in this chain searching for a match.
  - If a match is found, the corresponding physical frame is extracted.



Inverted Page Table

- One entry for each real page of memory
- Entry consists of the virtual address of the page stored in that real memory location, with information about the process that owns that page
- Decreases memory needed to store each page table, but increases time needed to search the table when a page reference occurs
- Use hash table to limit the search to one — or at most a few — page-table entries