CSE 421/521 - Operating Systems
Fall 2012

Lecture - XII
Main Memory Management

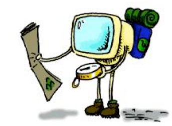
Tevfik Koşar

University at Buffalo

October 18th, 2012

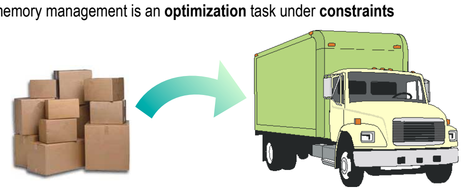
Roadmap

- Main Memory Management
  - Fixed and Dynamic Memory Allocation
  - External and Internal Fragmentation
  - Address Binding
  - HW Address Protection
  - Paging
  - Segmentation



Memory Management Requirements

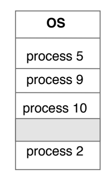
- **The O/S must fit multiple processes in memory**
  - memory needs to be subdivided to accommodate multiple processes
  - memory needs to be allocated to ensure a reasonable supply of ready processes so that the CPU is never idle
  - memory management is an **optimization** task under **constraints**



Fitting processes into memory is like fitting boxes into a fixed amount of space

Memory Allocation

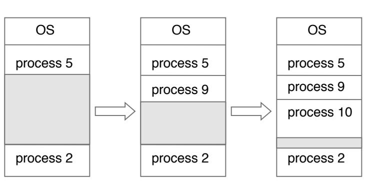
- Fixed-partition allocation
  - Divide memory into fixed-size partitions
  - Each partition contains exactly one process
  - The degree of multi programming is bound by the number of partitions
  - When a process terminates, the partition becomes available for other processes



➔no longer in use

Memory Allocation (Cont.)

- Variable-partition Scheme (Dynamic)
  - When a process arrives, search for a hole large enough for this process
  - Hole - block of available memory; holes of various size are scattered throughout memory
  - Allocate only as much memory as needed
  - 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 is faster.

Best-fit is better in terms of storage utilization.

Worst-fit may lead less fragmentation.

Example

Given five memory partitions of 100 KB, 500 KB, 200 KB, 300 KB, and 600 KB (in order), how would each of the first-fit, best-fit, and worst-fit algorithms place processes of 212 KB, 417 KB, 112 KB, and 426 KB (in order)? Which algorithm makes the most efficient use of memory?

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 is faster.

Best-fit is better in terms of storage utilization.

Worst-fit may lead less fragmentation.

Fragmentation

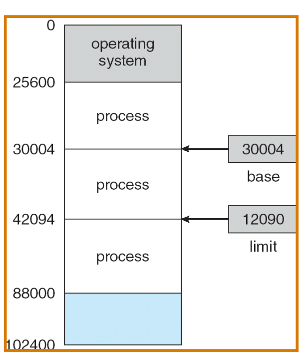
- **External Fragmentation** – total memory space exists to satisfy a request, but it is not contiguous (in average ~50% lost)
- **Internal Fragmentation** – allocated memory may be slightly larger than requested memory; this size difference is memory internal to a partition, but not being used
- Reduce 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

Address Binding

- Addresses in a source program are generally symbolic
  - eg. int count;
- A compiler binds these symbolic addresses to relocatable addresses
  - eg. 100 bytes from the beginning of this module
- The linkage editor or loader will in turn bind the relocatable addresses to absolute addresses
  - eg. 74014
- Each binding is mapping from one address space to another

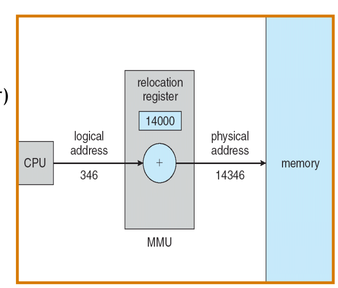
Logical Address Space

- Each process has a separate memory space
- Two registers provide address protection between processes:
- Base register: smallest legal address space
- Limit register: size of the legal range



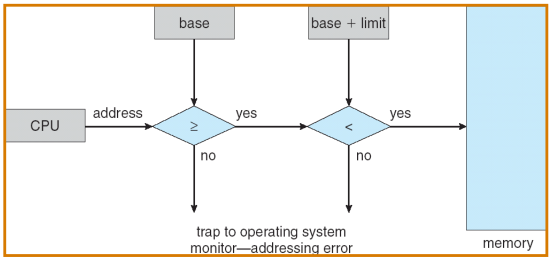
Memory-Management Unit (MMU)

- Hardware device that maps logical to physical address
- In MMU scheme, the value in the relocation register (base 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



HW Address Protection

- CPU hardware compares every address generated in user mode with the registers
- Any attempt to access other processes' memory will be trapped and cause a fatal error



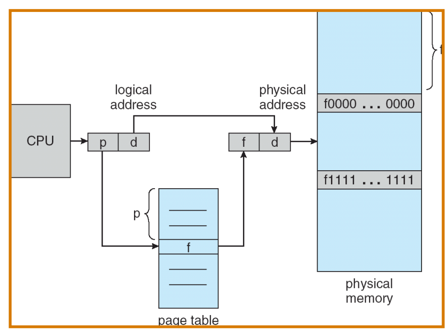
Paging - noncontiguous

- Physical address space of a process can be noncontiguous
- Divide physical memory into fixed-sized blocks called **frames** (size is power of 2, between 512 bytes and 16 megabytes)
- 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

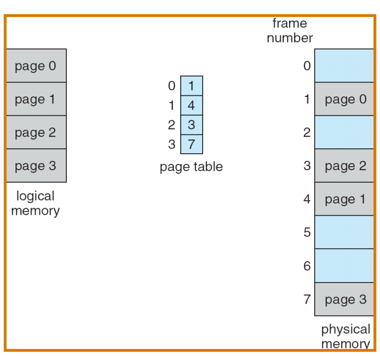
Address Translation Scheme

- 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

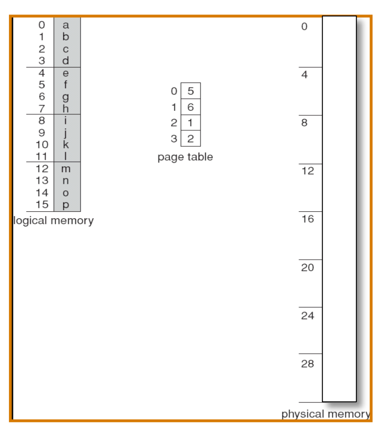
Address Translation Architecture



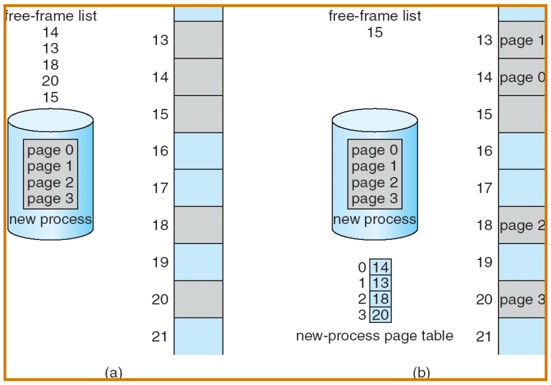
Paging Example



Paging Example



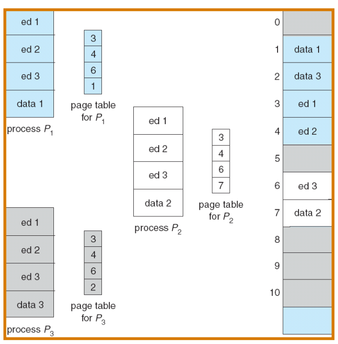
Free Frames



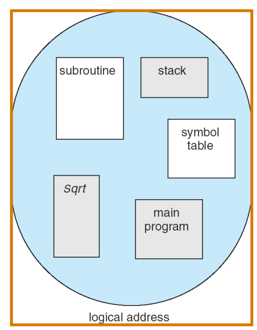
Shared Pages

- **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

Shared Pages Example



User's View of a Program

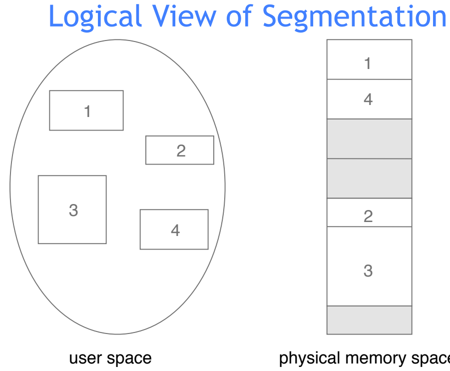


Segmentation

- 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

Logical View of Segmentation



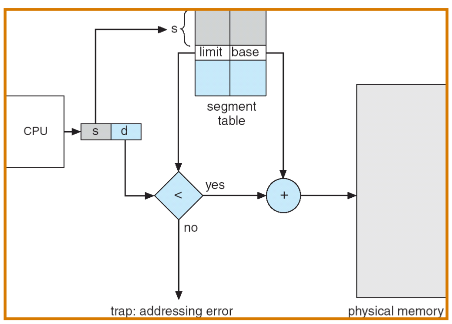
Segmentation Architecture

- 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 the length (limit) of the segment
- segment addressing is d (offset) < STLR

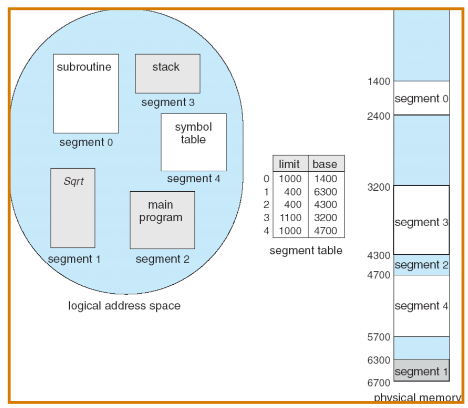
Segmentation Architecture (Cont.)

- Protection. With each entry in segment table associate:
  - validation bit = 0 ⇒ illegal segment
  - read/write/execute privileges
- Protection bits associated with segments; code sharing occurs at segment level
- Since segments vary in length, memory allocation is a dynamic storage-allocation problem
- A segmentation example is shown in the following diagram

Address Translation Architecture



Example of Segmentation



Exercise

- Consider the following segment table:

Segment	Base	Length
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

What are the physical addresses for the following logical addresses?

a. 1, 100

b. 2, 0

c. 3, 580

Solution

- Consider the following segment table:

Segment	Base	Length
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

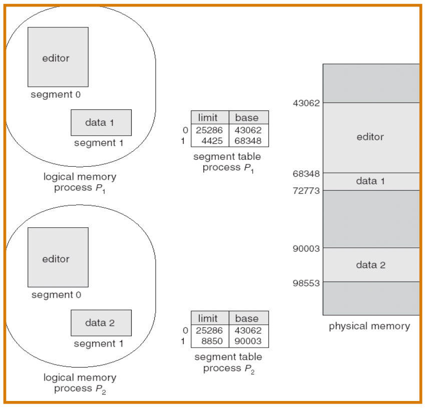
What are the physical addresses for the following logical addresses?

a. 1, 100
illegal reference (2300+100 is not within segment limits)

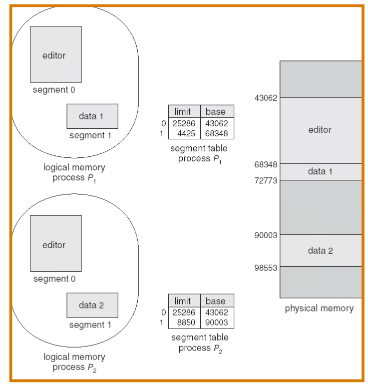
b. 2, 0
physical address = 90 + 0 = 90

c. 3, 580
illegal reference (1327 + 580 is not within segment limits)

Sharing of Segments

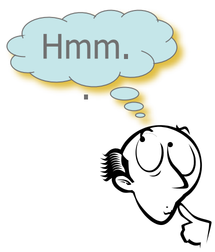


Sharing of Segments



Summary

- Main Memory Management
  - Memory Allocation
  - Fragmentation
  - Address Binding
  - HW Address Protection
  - Paging
  - Segmentation



- Next Lecture: Virtual Memory

Acknowledgements

- "Operating Systems Concepts" book and supplementary material by A. Silberschatz, P. Galvin and G. Gagne
- "Operating Systems: Internals and Design Principles" book and supplementary material by W. Stallings
- "Modern Operating Systems" book and supplementary material by A. Tanenbaum
- R. Doursat and M. Yuksel from UNR