

CSC 4103 - Operating Systems
Fall 2009

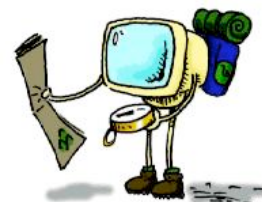
LECTURE - XVII
MAIN MEMORY

Tevfik Koşar

Louisiana State University
October 27th, 2009

Roadmap

- **Paging**
 - Address Translation Scheme
 - Shared Pages
- **Segmentation**
 - Address Translation Scheme
 - Shared Segments



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 8192 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

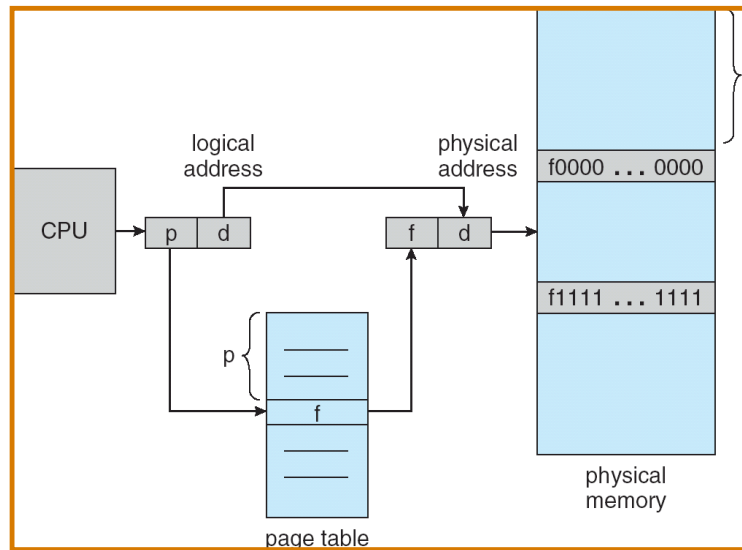
16

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

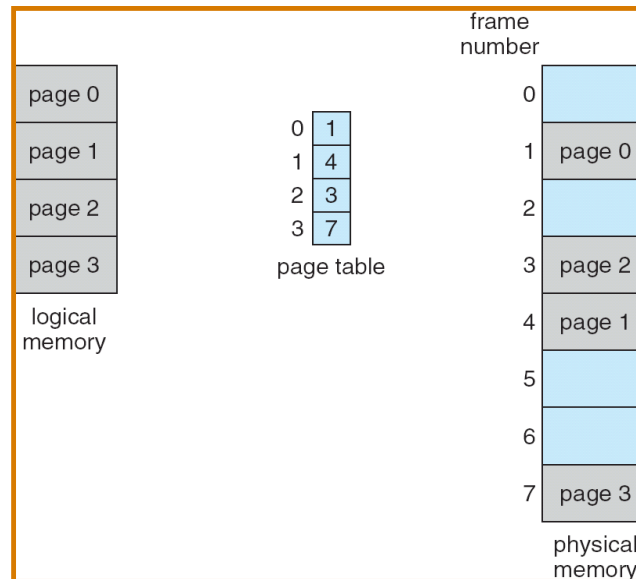
17

Address Translation Architecture



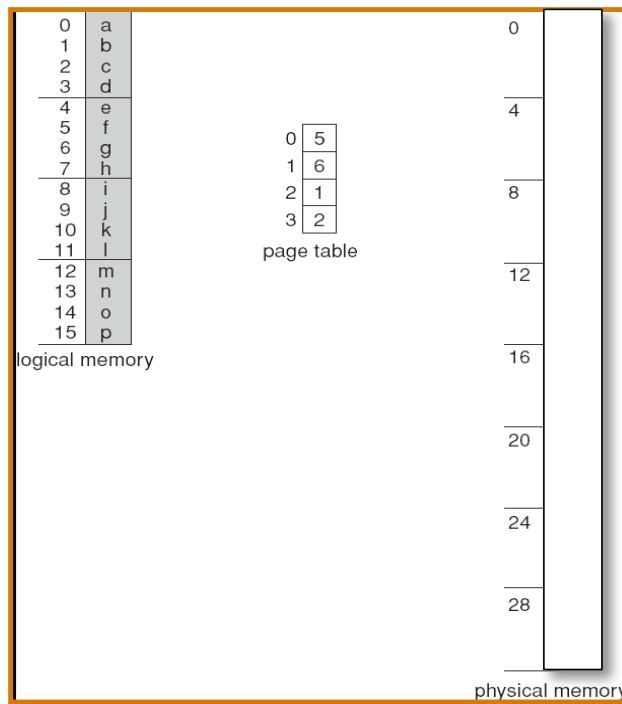
18

Paging Example



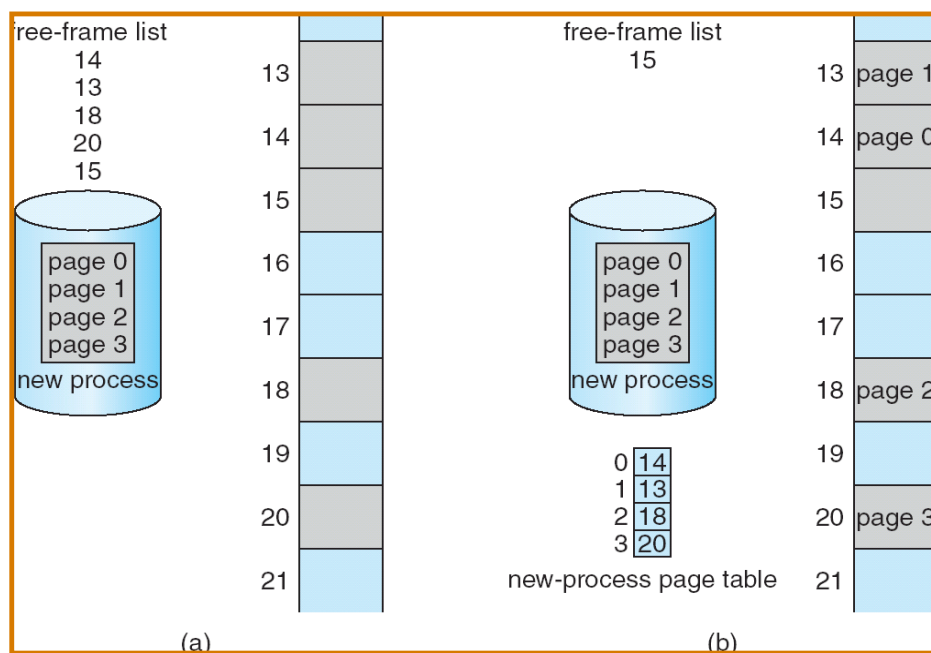
19

Paging Example



20

Free Frames



21

Shared Pages

- **Shared code**

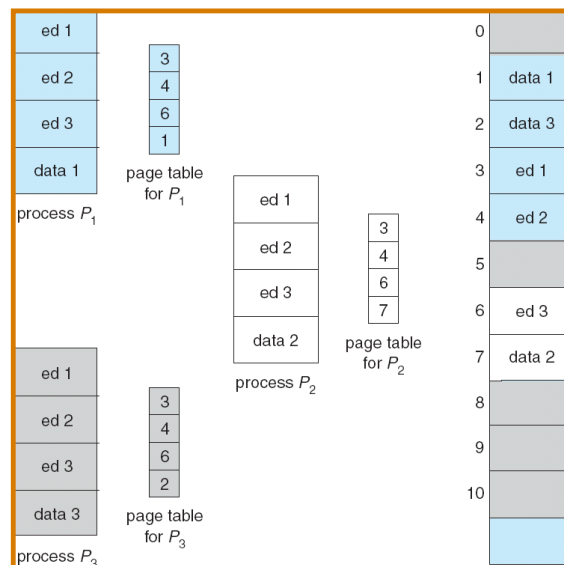
- One copy of read-only (reentrant) code shared among processes (i.e., text editors, compilers, window systems).
- Shared code must appear in same location in the logical address space of all processes

- **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

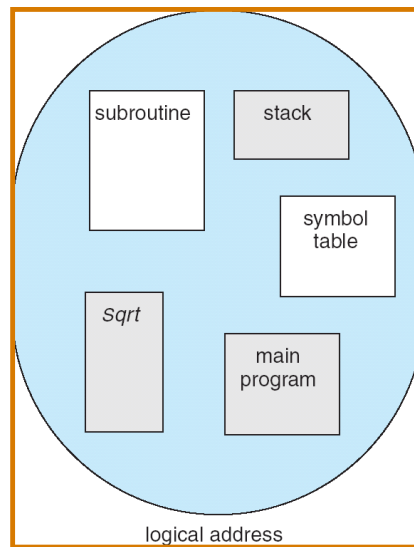
22

Shared Pages Example



23

User's View of a Program



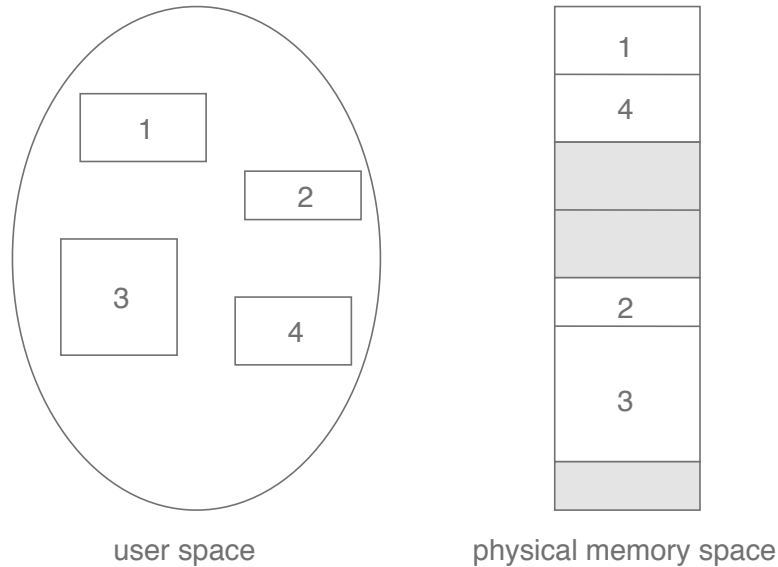
24

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,

25

Logical View of Segmentation



26

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;

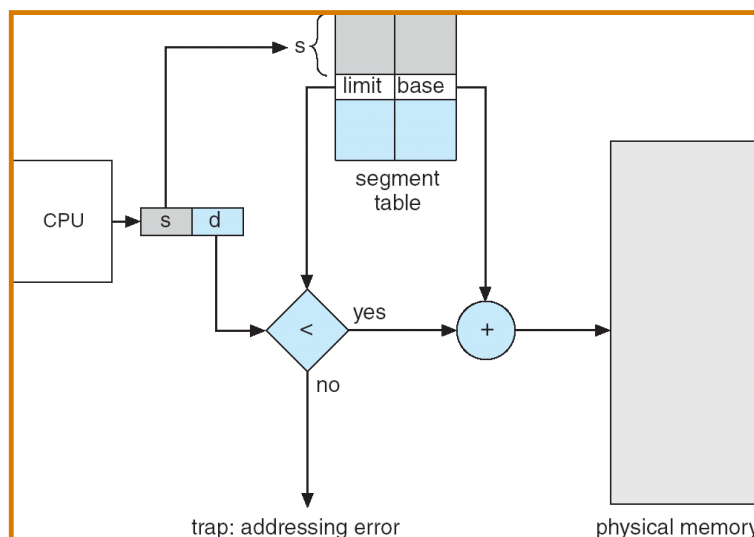
27

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; 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

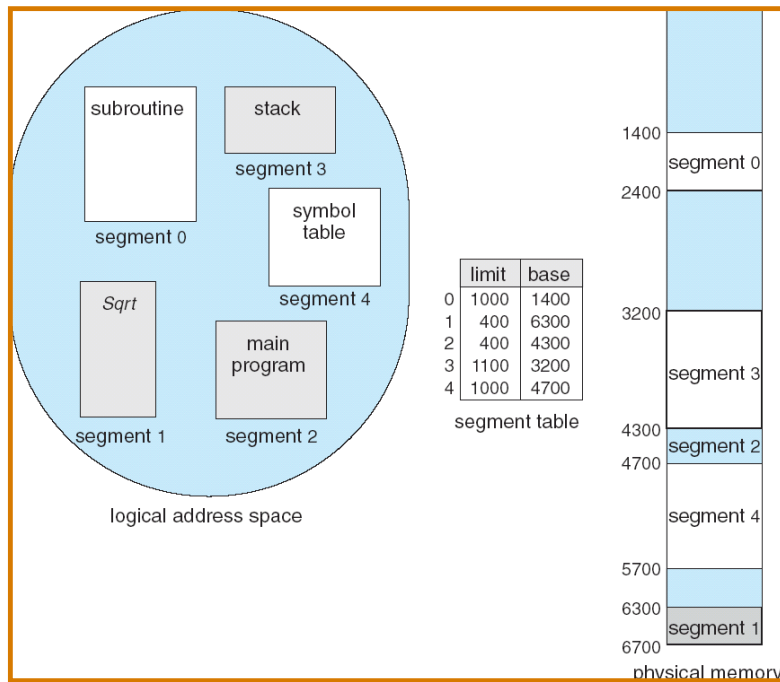
28

Address Translation Architecture



29

Example of Segmentation



30

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?

- 1, 100
- 2, 0
- 3, 580

Solution

- Consider the following segment table:

<u>Segment</u>	<u>Base</u>	<u>Length</u>
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

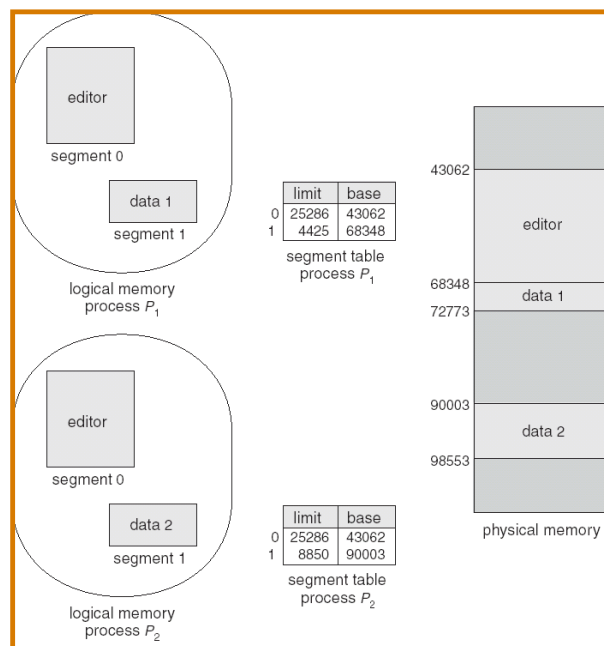
physical address = $90 + 0 = 90$

c. 3, 580

illegal reference (1327 + 580 is not within segment limits)

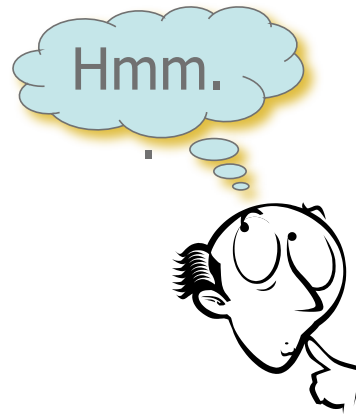
19

Sharing of Segments



Summary

- Fragmentation
- Address Binding
- Address Protection
- Paging
- Segmentation



- Next Lecture: Virtual Memory
- Reading Assignment: Chapter 8 from Silberschatz.

21

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

22