OPERATING SYSTEMS

Memory Management

Suresh Jamadagni
Department of Computer Science

OPERATING SYSTEMS

Slides Credits for all the PPTs of this course

- The slides/diagrams in this course are an **adaptation**, **combination**, and **enhancement** of material from the following resources and persons:

1. Slides of Operating System Concepts, Abraham Silberschatz, Peter Baer Galvin, Greg Gagne - 9th edition 2013 and some slides from 10th edition 2018
2. Some conceptual text and diagram from Operating Systems - Internals and Design Principles, William Stallings, 9th edition 2018
3. Some presentation transcripts from A. Frank – P. Weisberg
4. Some conceptual text from Operating Systems: Three Easy Pieces, Remzi Arpaci-Dusseau, Andrea Arpaci Dusseau

OPERATING SYSTEMS

Segmentation

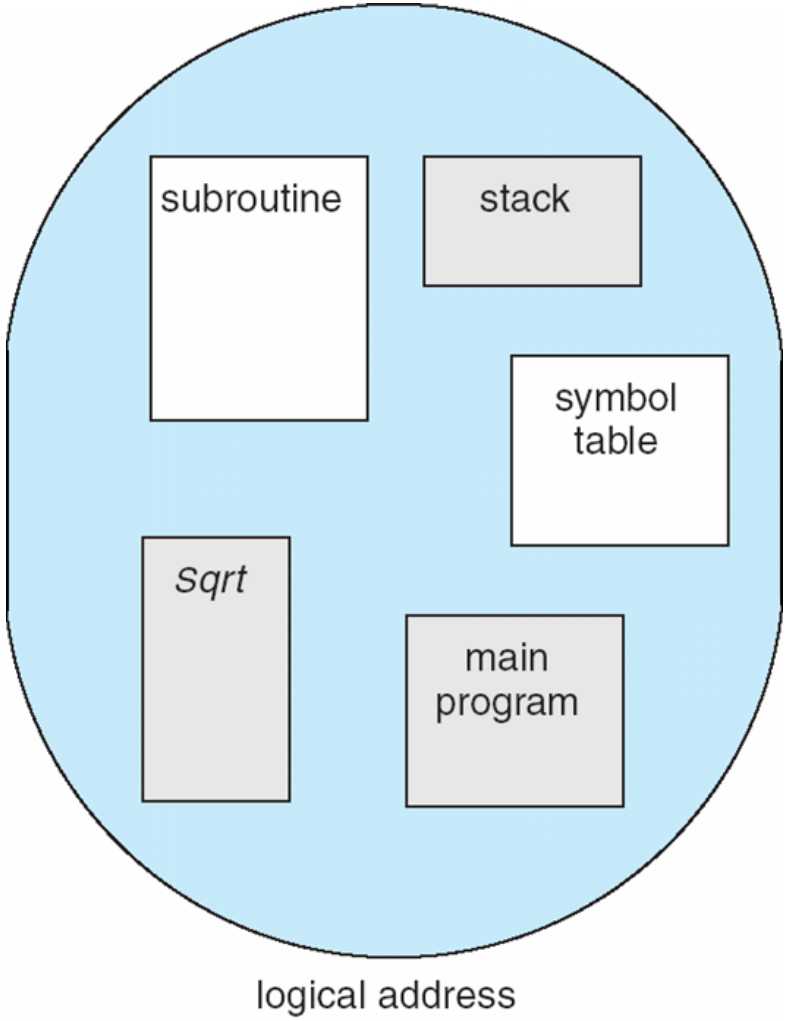
Suresh Jamadagni
Department of Computer Science

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

OPERATING SYSTEMS

User's View of a Program

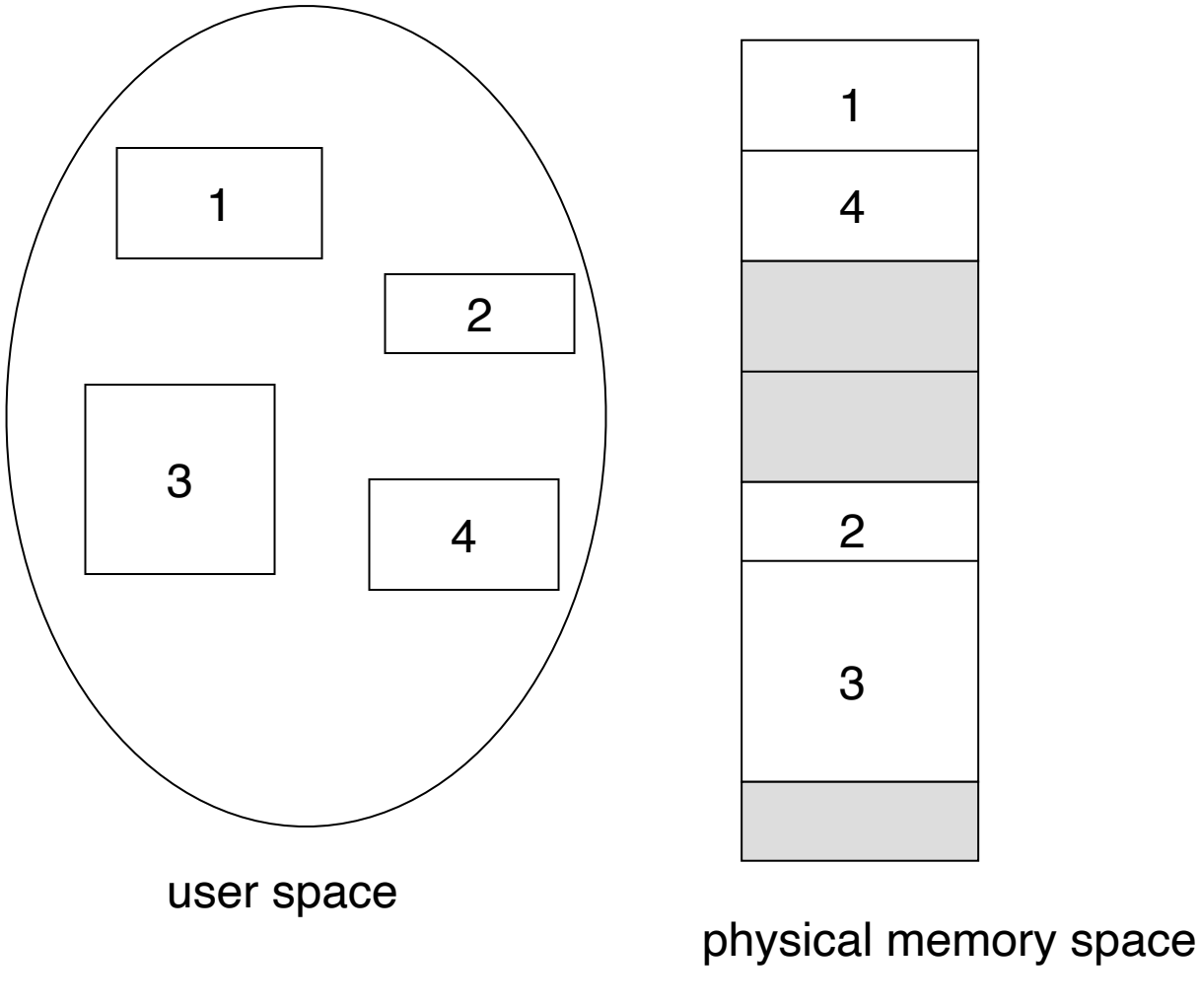


➢Segmentation is a memory-management scheme that supports this programmer view of memory

➢Each segment has a name and a length.

➢Addresses specify segment name + offset within the segment

Logical View of Segmentation



Segmentation Basics

- Segmentation is a technique for breaking memory up into logical pieces
- Each “piece” is a grouping of related information
  - data segments for each process
  - code segments for each process
  - data segments for the OS
  - etc.
- Segmentation permits the physical address space of a process to be non-contiguous.
- Like paging, use virtual addresses and use disk to make memory look bigger than it really is
- Segmentation can be implemented with or without paging

OPERATING SYSTEMS

Segmentation Architecture

- Logical address consists of a two tuple:

  <segment-number s, offset d>,

- **Segment table** – maps two-dimensional physical addresses; each table entry has:
  - **Segment base** – contains the starting physical address where the segments reside in memory
  - **Segment 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 ⇒ illegal segment
    - read/write/execute privileges
- Protection bits associated with segments
- Since segments vary in length, memory allocation is a dynamic storage-allocation problem

OPERATING SYSTEMS

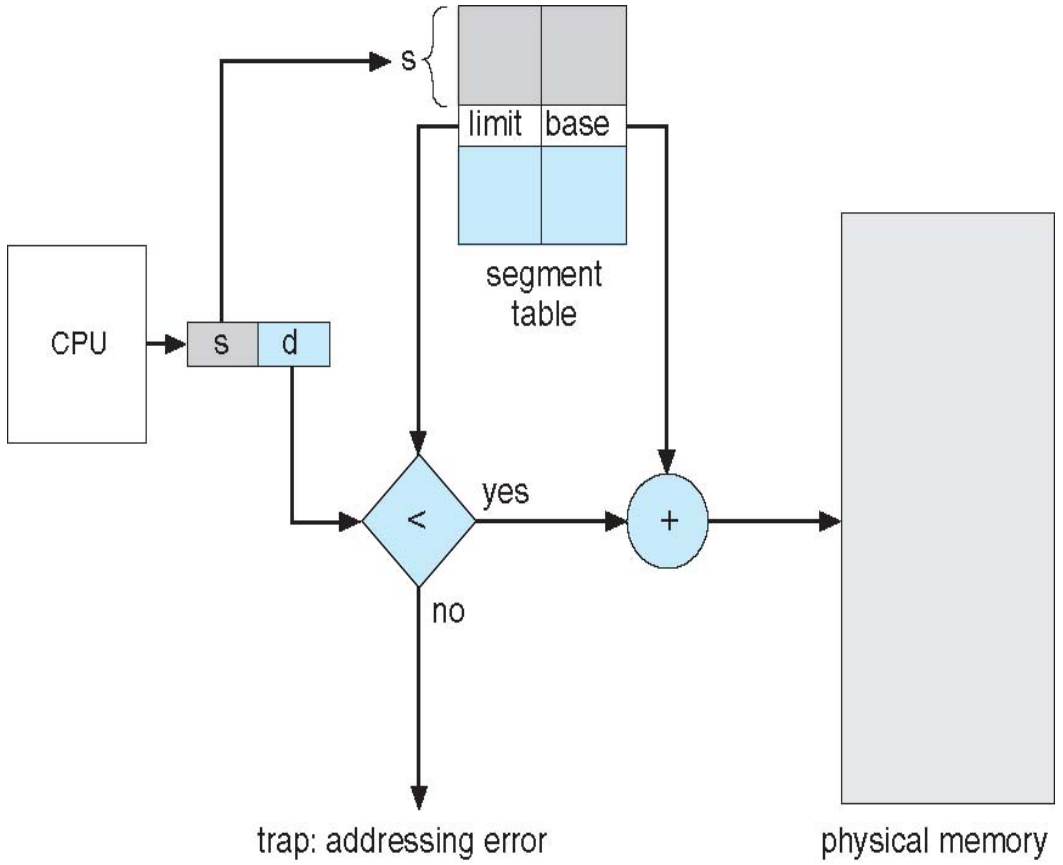
Segmentation Hardware

- Sounds very similar to paging
- Big difference – segments can be variable in size
- As with paging, to be effective hardware must be used to translate logical address
- Most systems provide segment registers
- If a reference isn't found in one of the segment registers
  - trap to operating system
  - OS does lookup in segment table and loads new segment descriptor into the register
  - return control to the user and resume

OPERATING SYSTEMS

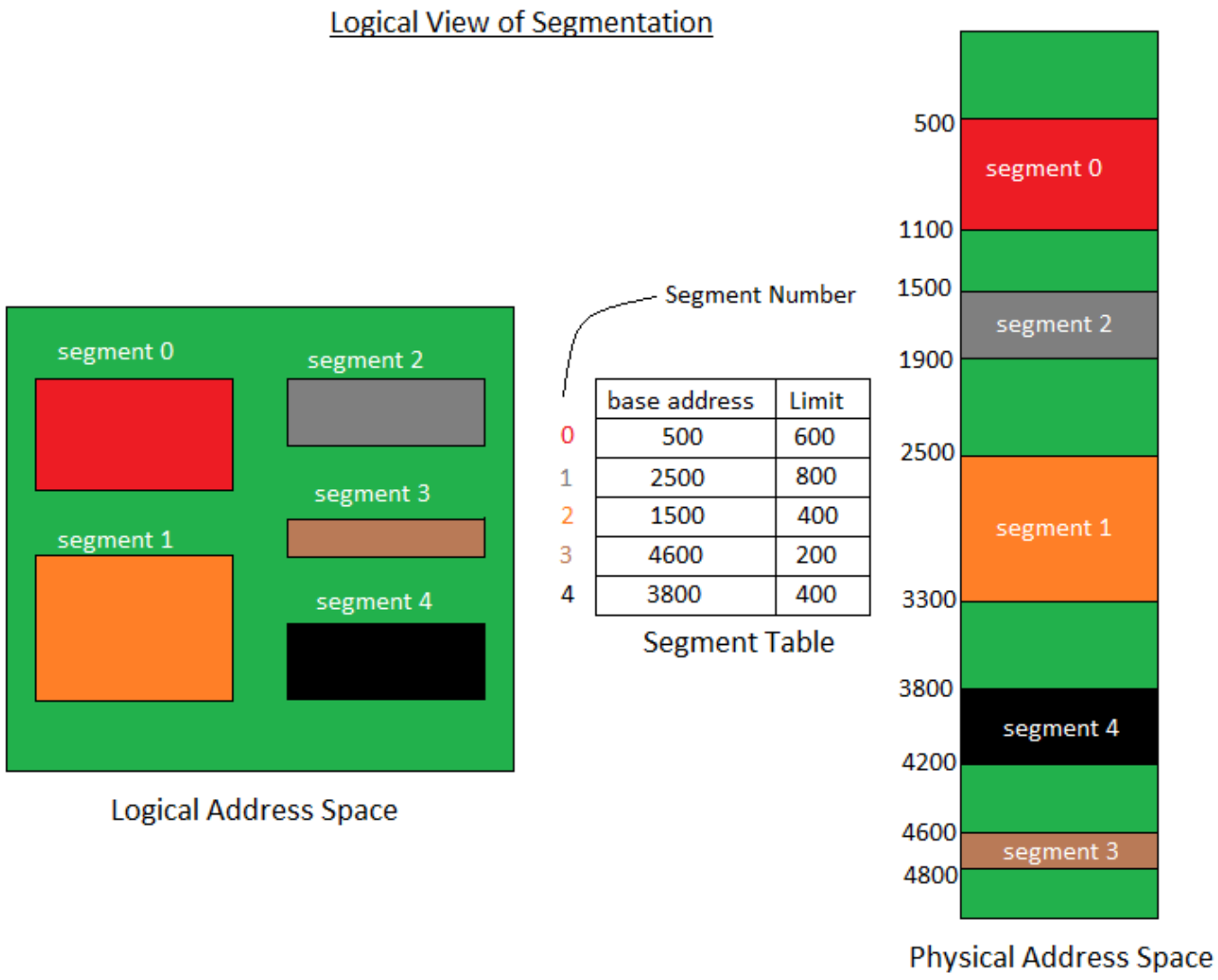
Segmentation Hardware (Cont.)

- A logical address consists of two parts: a segment number, s, and an offset into that segment, d.
- The segment number is used as an index to the segment table. The offset d of the logical address must be between 0 and the segment limit. If it is not, we trap to the operating system (logical addressing attempt beyond end of segment).
- When an offset is legal, it is added to the segment base to produce the address in physical memory of the desired byte. The segment table is thus essentially an array of base–limit register pairs.



OPERATING SYSTEMS

Segmentation Example



Segmentation Example (Cont.)

- We have five segments numbered from 0 through 4. The segments are stored in physical memory as shown.
- The segment table has a separate entry for each segment, giving the beginning address of the segment in physical memory (or base) and the length of that segment (or limit).
- For example, segment 2 is 400 bytes long and begins at location 1500.
  - Thus, a reference to byte 53 of segment 2 is mapped onto location 1500 + 53 = 1553.
- A reference to segment 3, byte 852, is mapped to 4600 (the base of segment 3) + 852 = 5452.
- A reference to byte 722 of segment 0 would result in a trap to the operating system, as this segment is only 600 bytes long.

Segmentation Comments

- In each segment table entry, we have both the starting address and length of the segment; the segment can thus dynamically grow or shrink as needed.
- But variable length segments introduce external fragmentation and are more difficult to swap in and out.
- It is natural to provide protection and sharing at the segment level since segments are visible to the programmer (pages are not).
- Useful protection bits in segment table entry:
  - read-only/read-write bit
  - Kernel/User bit

Segmentation Issues

- Entire segment is either in memory or on disk
- Variable sized segments leads to external fragmentation in memory
- Must find a space big enough to place segment into
- May need to swap out some segments to bring a new segment in.

Segmentation: Examples

- 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. 0,430

b. 1,10

c. 2,500

d. 3,400

e. 4,112

Segmentation: Examples

❑ 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. 0,430
b. 1,10
c. 2,500
d. 3,400
e. 4,112

Solution:

a. 0, 430 →219 + 430 = 649
b. 1, 10 →2300 + 10 = 2310
c. 2, 500 Illegal address since size of segment 2 is 100 and the offset in logical address is 500.
d. 3, 400 →1327 + 400 = 1727
e. 4, 112 Illegal address since size of segment 4 is 96 and the offset in logical address is 112.

THANK YOU

Suresh Jamadagni
Department of Computer Science and Engineering

sureshjamadagni@pes.edu