CSE 4/521 Introduction to Operating Systems

Lecture 13 – Main Memory II

(Contiguous Memory Allocation, Segmentation)

Summer 2018

Overview

Objective:

1. To explore various techniques of allocating memory to processes.

- Contiguous Memory Allocation
- Segmentation

Recap

- Memory Management Background
  - Base registers and limit registers, address binding, logical vs. physical addresses (and spaces)
- Swapping
  - Swapping principles, swapping on mobile systems

Questions

1. Name 2 differences between logical and physical address. (Easy)

2. Explain why mobile OSes such as iOS and Android do not support swapping? (Medium)

3. Why the concept of logical and physical address? Are there alternatives to this approach? (Open-ended)

Overview

- Contiguous Memory Allocation
- Segmentation

Overview

- Contiguous Memory Allocation
- Segmentation

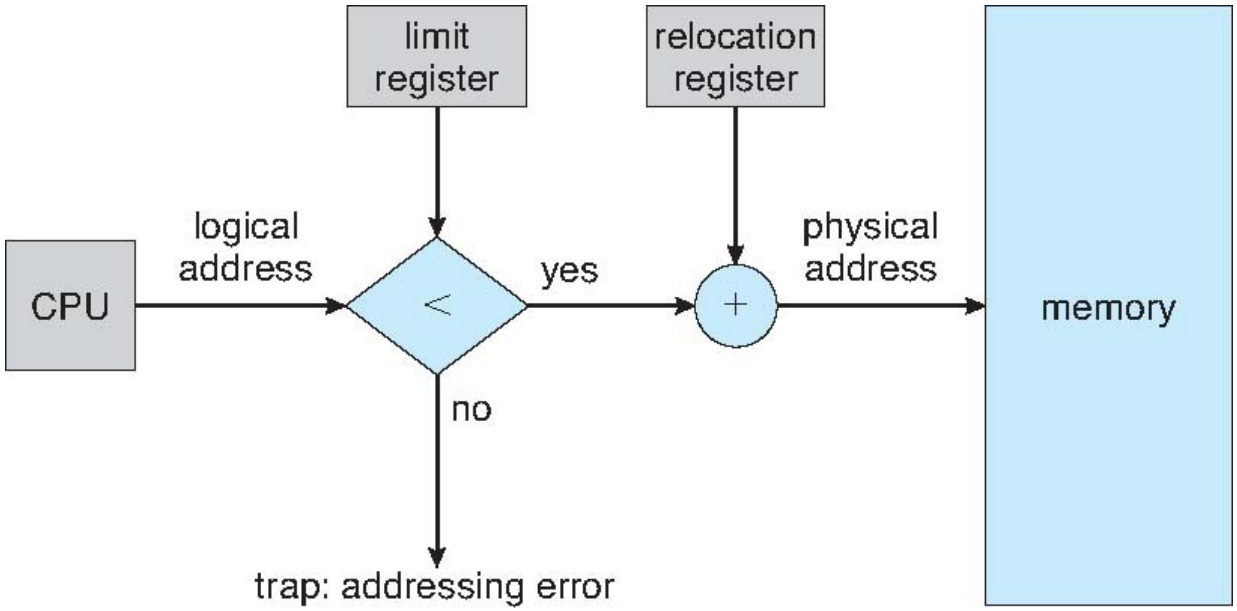
Contiguous Memory Allocation

- Main memory must support both OS and user processes
- Limited resource, must allocate efficiently
- Contiguous allocation is one early method
- Main memory usually into two partitions:
  - Resident operating system, usually held in low memory
  - User processes then held in high memory
  - Each process contained in single contiguous section of memory

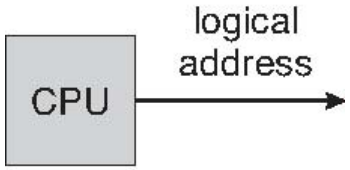
Contiguous Memory Allocation

- 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*
  - Can then allow actions such as kernel code being transient and kernel changing size

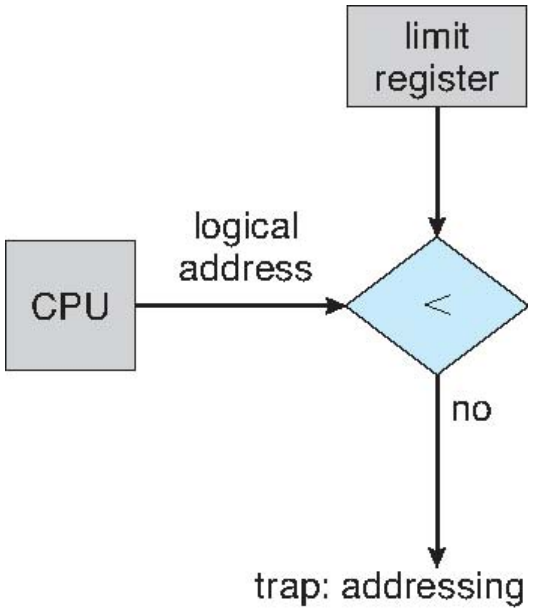
Contiguous Memory Allocation : Hardware Support for Relocation and Limit Registers



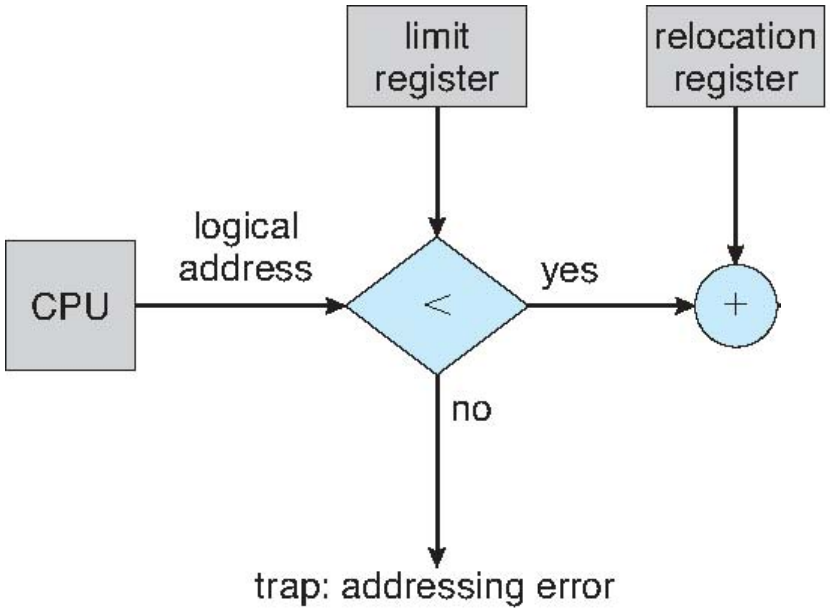
Contiguous Memory Allocation : Hardware Support for Relocation and Limit Registers



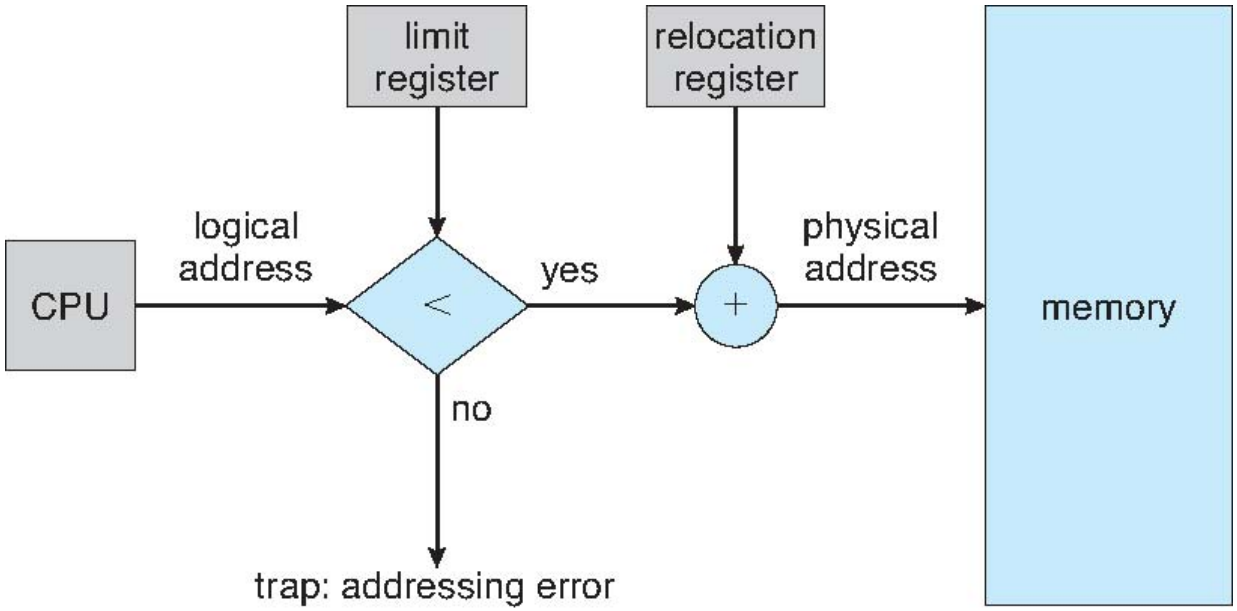
Contiguous Memory Allocation : Hardware Support for Relocation and Limit Registers



Contiguous Memory Allocation : Hardware Support for Relocation and Limit Registers

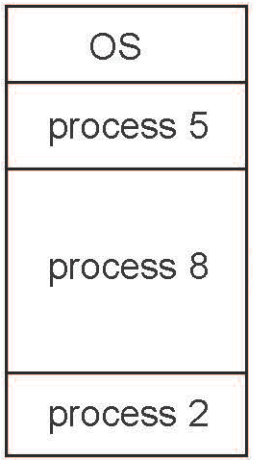


Contiguous Memory Allocation : Hardware Support for Relocation and Limit Registers



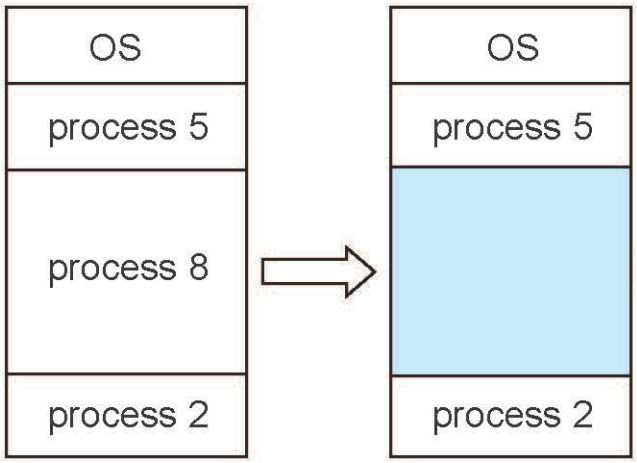
Contiguous Memory Allocation : Multiple-partition Allocation

- Multiple-partition allocation
  - Degree of multiprogramming limited by number of partitions
  - Variable-partition sizes for efficiency (sized to a given process' needs)
  - 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
  - Process exiting frees its partition, adjacent free partitions combined
  - Operating system maintains information about: a) allocated partitions b) free partitions (hole)



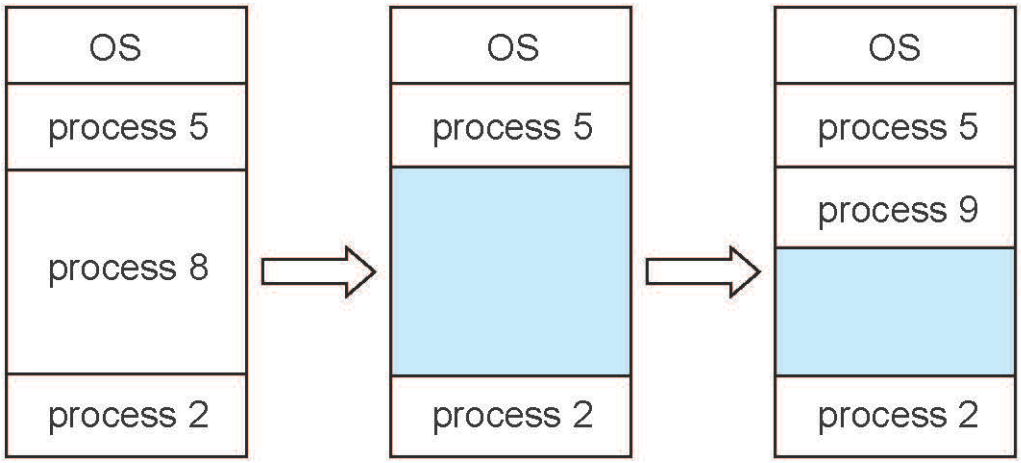
Contiguous Memory Allocation : Multiple-partition Allocation

- Multiple-partition allocation
  - Degree of multiprogramming limited by number of partitions
  - Variable-partition sizes for efficiency (sized to a given process' needs)
  - 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
  - Process exiting frees its partition, adjacent free partitions combined
  - Operating system maintains information about: a) allocated partitions b) free partitions (hole)



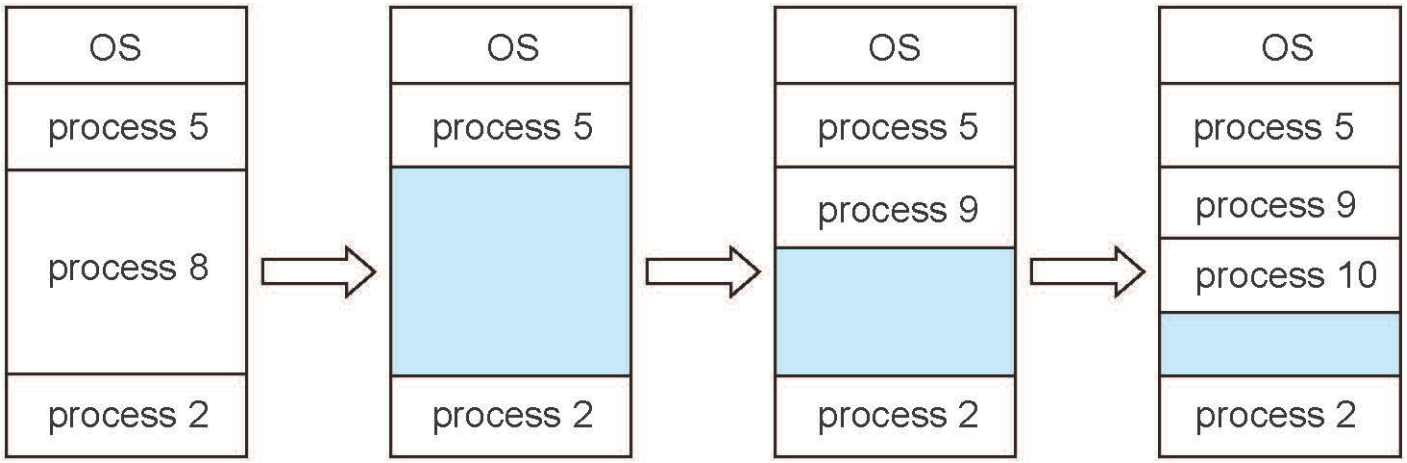
Contiguous Memory Allocation : Multiple-partition Allocation

- Multiple-partition allocation
  - Degree of multiprogramming limited by number of partitions
  - Variable-partition sizes for efficiency (sized to a given process' needs)
  - 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
  - Process exiting frees its partition, adjacent free partitions combined
  - Operating system maintains information about: a) allocated partitions b) free partitions (hole)



Contiguous Memory Allocation : Multiple-partition Allocation

- Multiple-partition allocation
  - Degree of multiprogramming limited by number of partitions
  - Variable-partition sizes for efficiency (sized to a given process' needs)
  - 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
  - Process exiting frees its partition, adjacent free partitions combined
  - Operating system maintains information about: a) allocated partitions b) free partitions (hole)



Contiguous Memory Allocation : 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
  - 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

Contiguous Memory Allocation : 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 wasted memory internal to a partition
- First fit analysis reveals that given *N* blocks allocated, 0.5 *N* blocks lost to fragmentation
  - 1/3 may be unusable -> 50-percent rule

Continuous Memory Allocation : Fragmentation

- 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
- Backing store also experiences same fragmentation problems

Overview

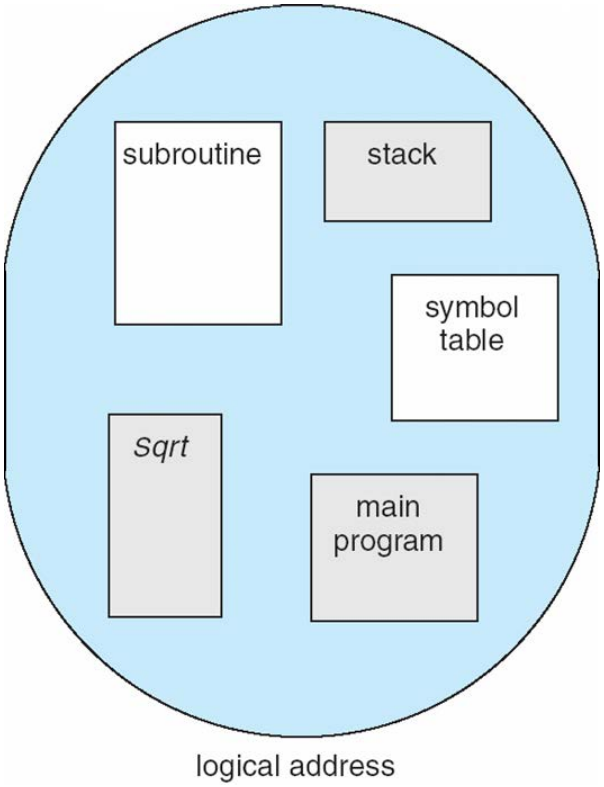
- Contiguous Memory Allocation
- Segmentation

Segmentation

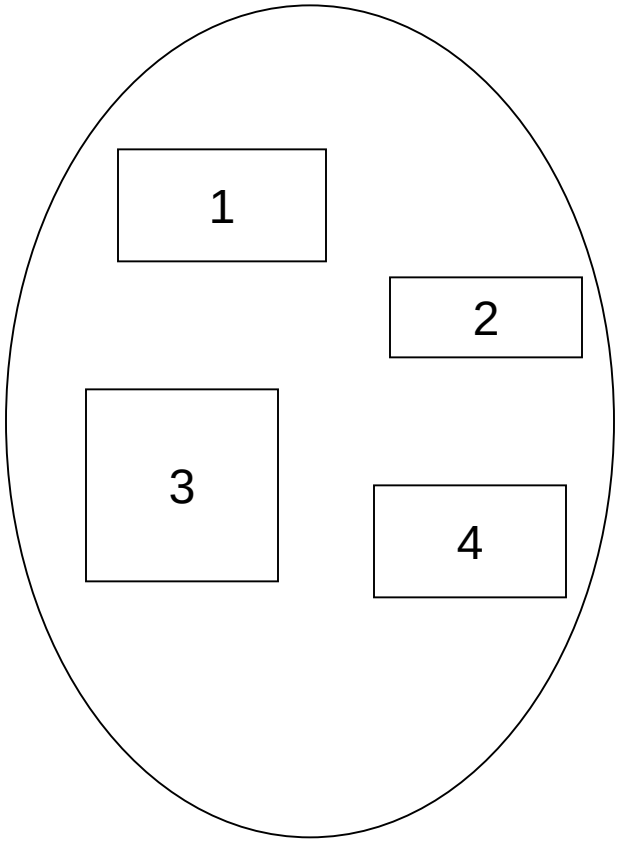
- Memory-management scheme that supports user view of memory
- A program is a collection of segments

  main program
  procedure
  function
  method
  object
  local variables, global variables
  common block
  stack
  symbol table
  arrays

Segmentation : User's View of a Program



Segmentation : Logical View of Segmentation



user space

physical memory space

Segmentation : Segmentation Architecture

- Logical address consists of a two tuple:

  <segment-number, offset>

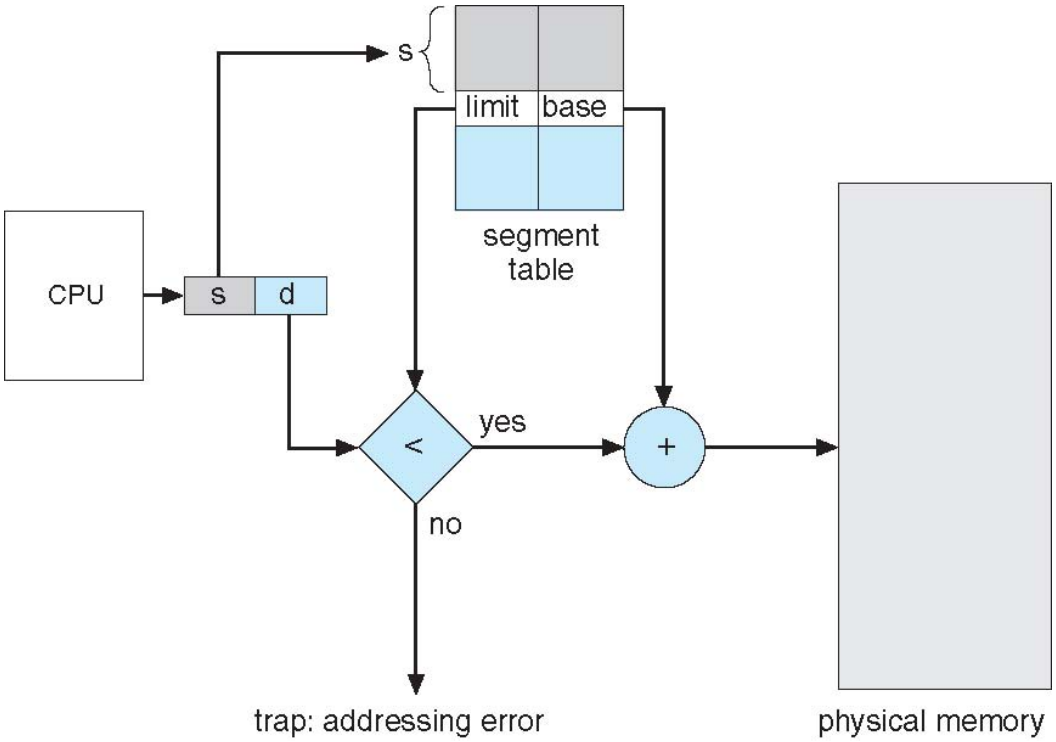
- Segment table – maps two-dimensional physical addresses; each table entry has:
  - base – contains the starting physical address where the segments reside in memory
  - limit – specifies the length of the segment
- Segment-table base register (STBR) points to the segment table's location in memory
- Segment-table length register (STLR) indicates number of segments used by a program;

  segment number *s* is legal if *s* < STLR

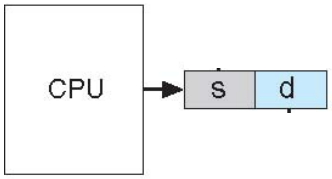
Segmentation : Segmentation Architecture

- 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

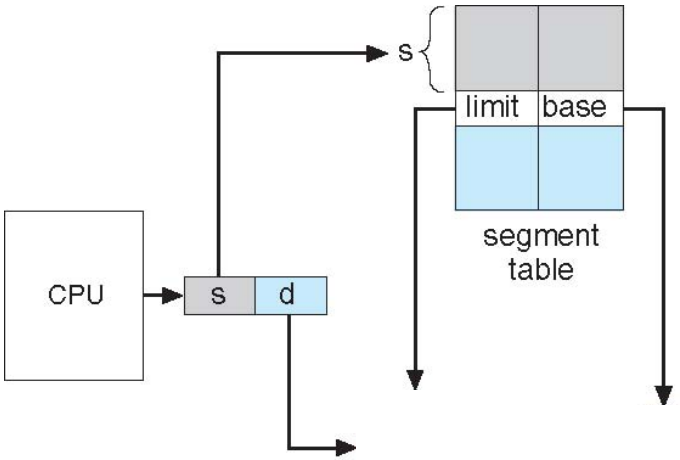
Segmentation : Segmentation Hardware



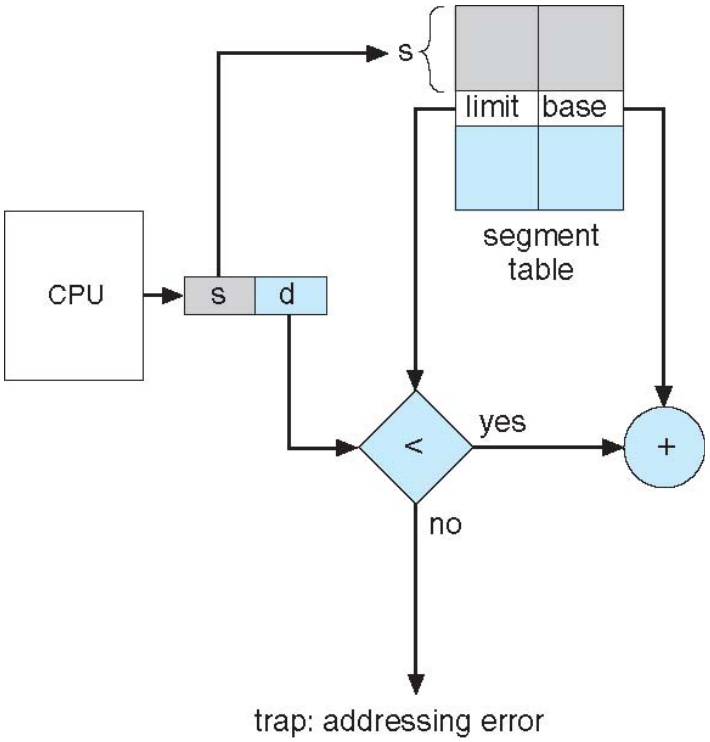
Segmentation : Segmentation Hardware



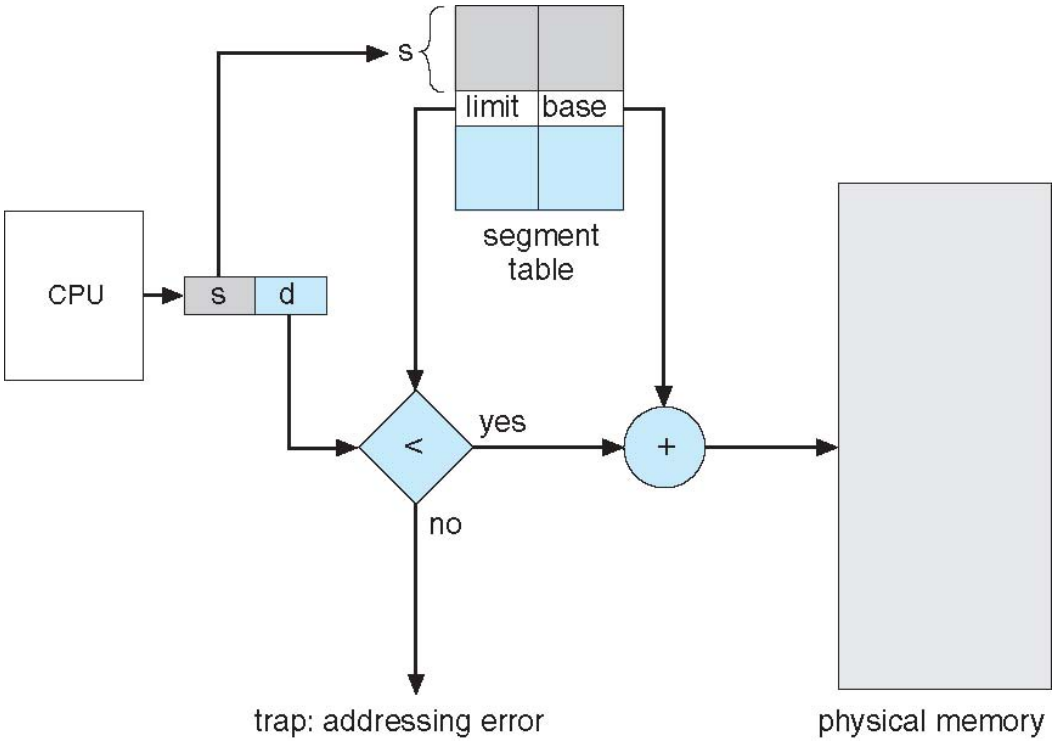
Segmentation : Segmentation Hardware



Segmentation : Segmentation Hardware



Segmentation : Segmentation Hardware



Credits for slides

Silberschatz, Galvin and Gagne