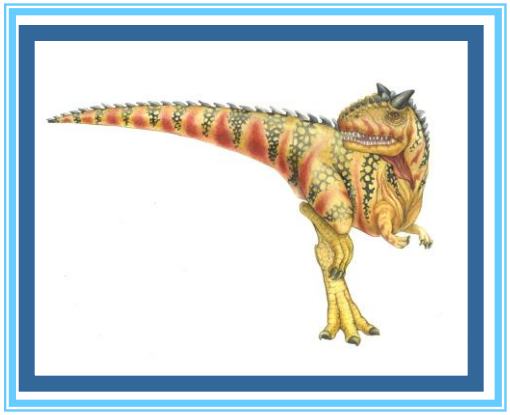
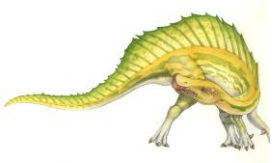
Chapter 8: Main Memory



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.
  - The two memory access problem can be solved by the use of a special fast-lookup hardware cache called **translation look-aside buffers (TLBs)**



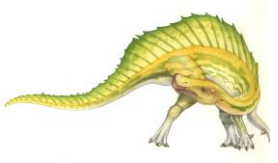
Associative Memory

- Associative memory – parallel search

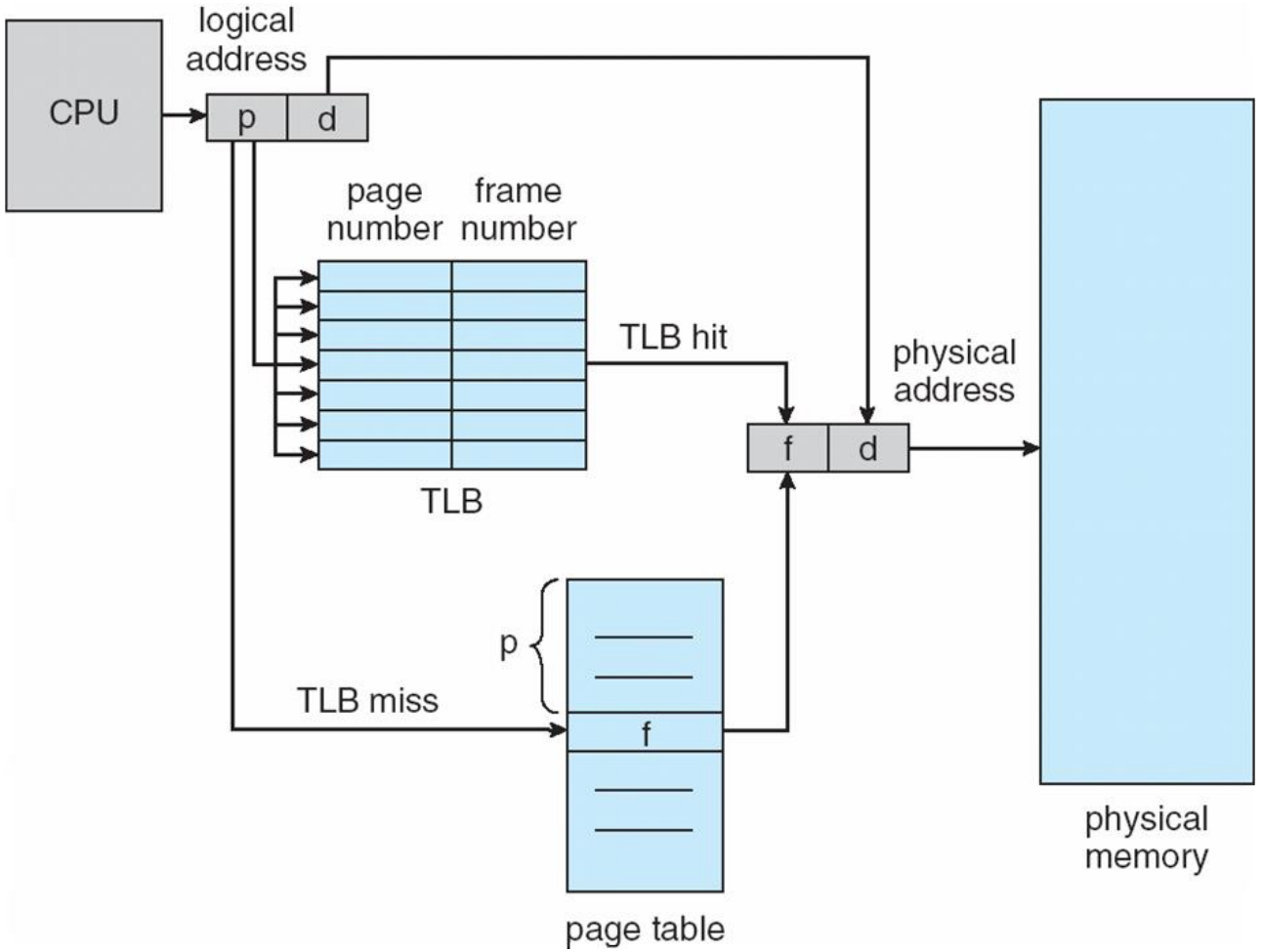
Page #	Frame #

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

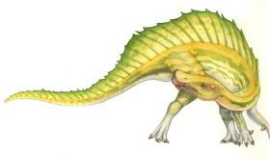


Effective Access Time

- Associative Lookup = ε time unit
- Assume memory cycle time is 1 microsecond
- Hit ratio α – percentage of times that a page number is found in the associative registers; ratio related to number of associative registers
- **Effective Access Time** (EAT)

$$EAT = (1 + \varepsilon)\,\alpha + (2 + \varepsilon)(1 - \alpha)$$

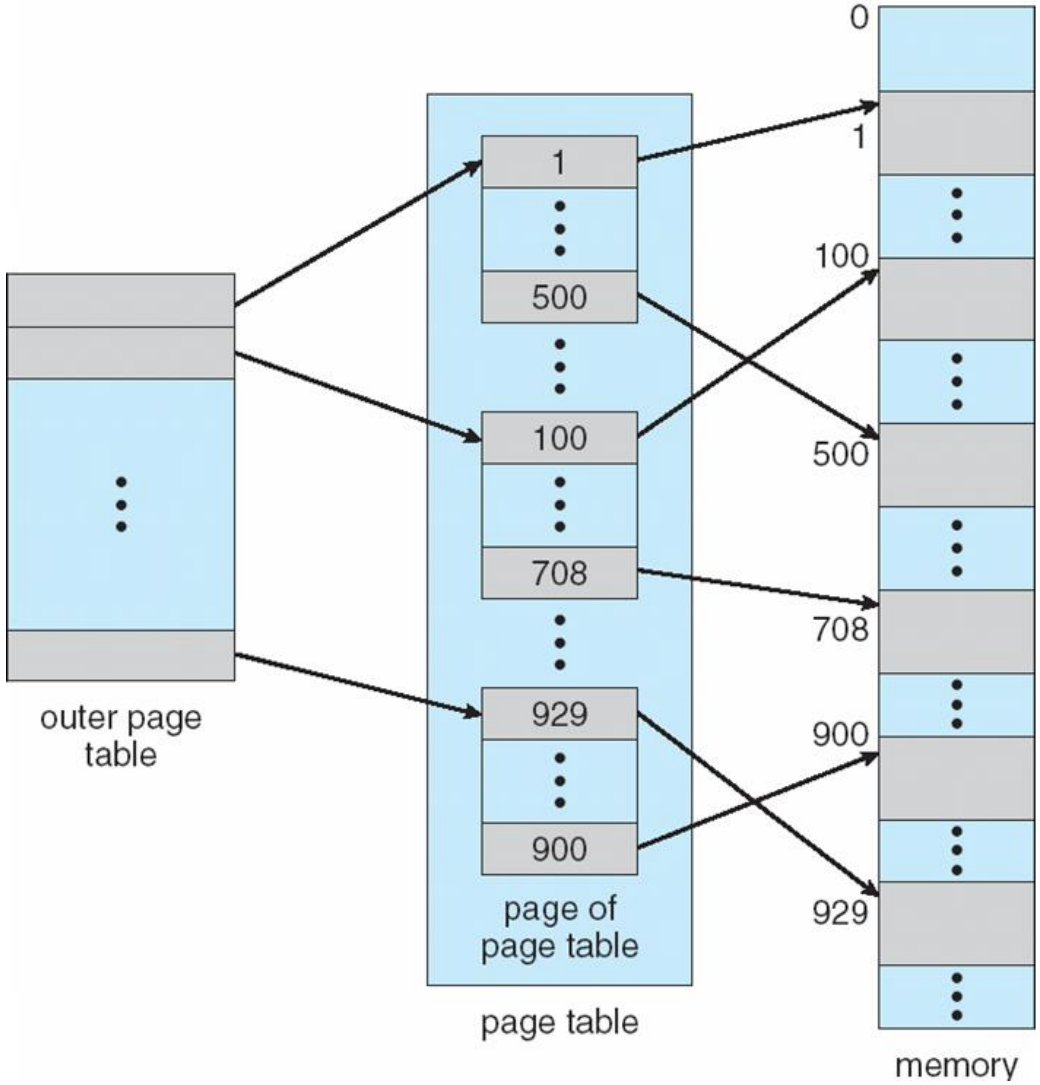
$$= 2 + \varepsilon - \alpha$$



Hierarchical Page Tables

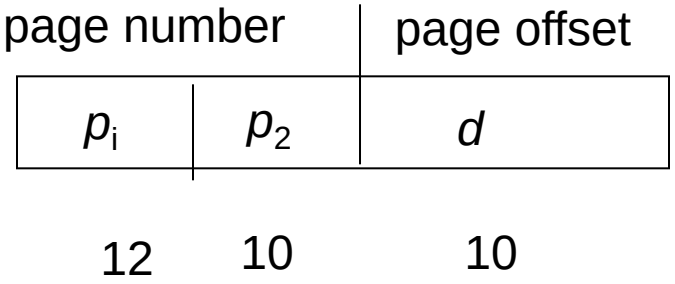
- Break up the logical address space into multiple page tables
- A simple technique is a two-level page table

Two-Level Page-Table Scheme

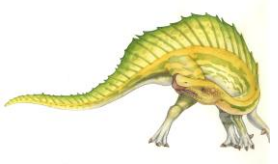


Two-Level Paging Example

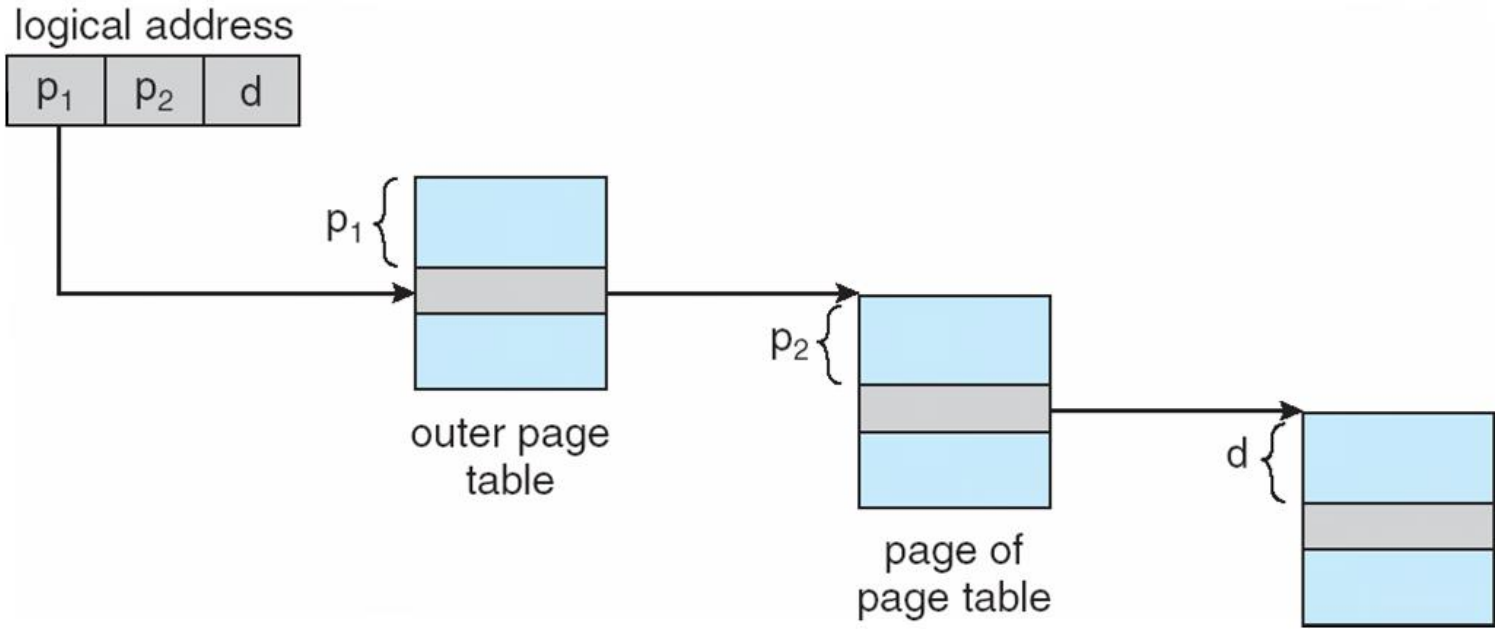
- A logical address (on 32-bit machine with 1K page size) is divided into:
  - a page number consisting of 22 bits
  - a page offset consisting of 10 bits
- Since the page table is paged, the page number is further divided into:
  - a 12-bit page number
  - a 10-bit page offset
- Thus, a logical address is as follows:

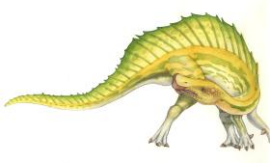


where p_i is an index into the outer page table, and p_2 is the displacement within the page of the outer page table

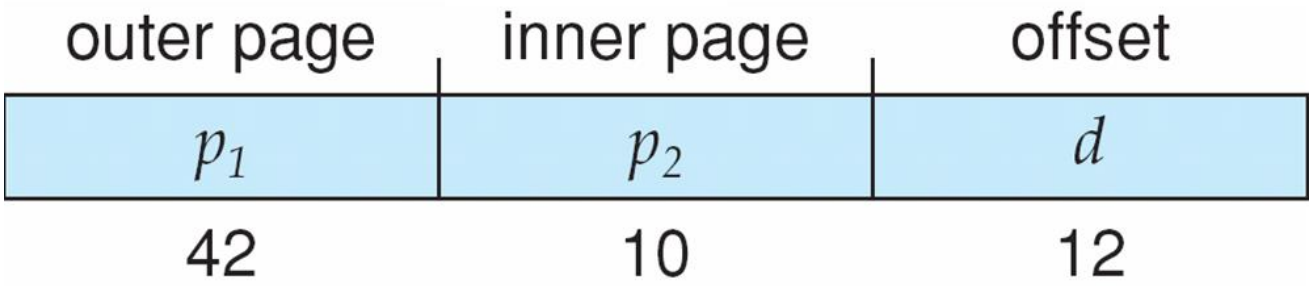


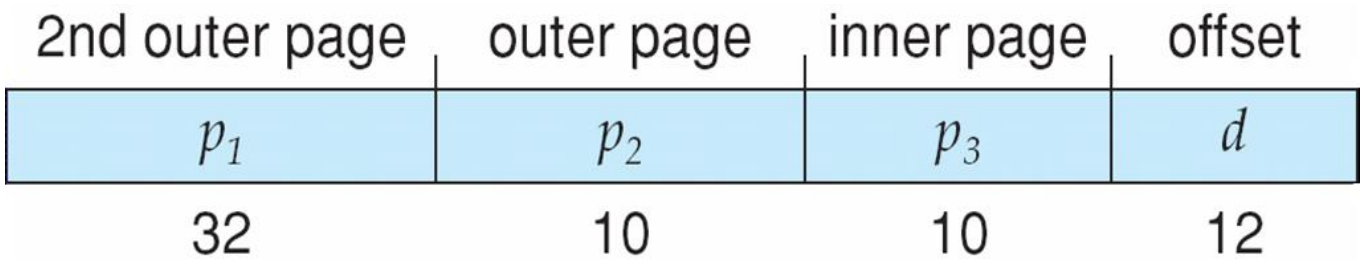
Address-Translation Scheme





Three-level Paging Scheme





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

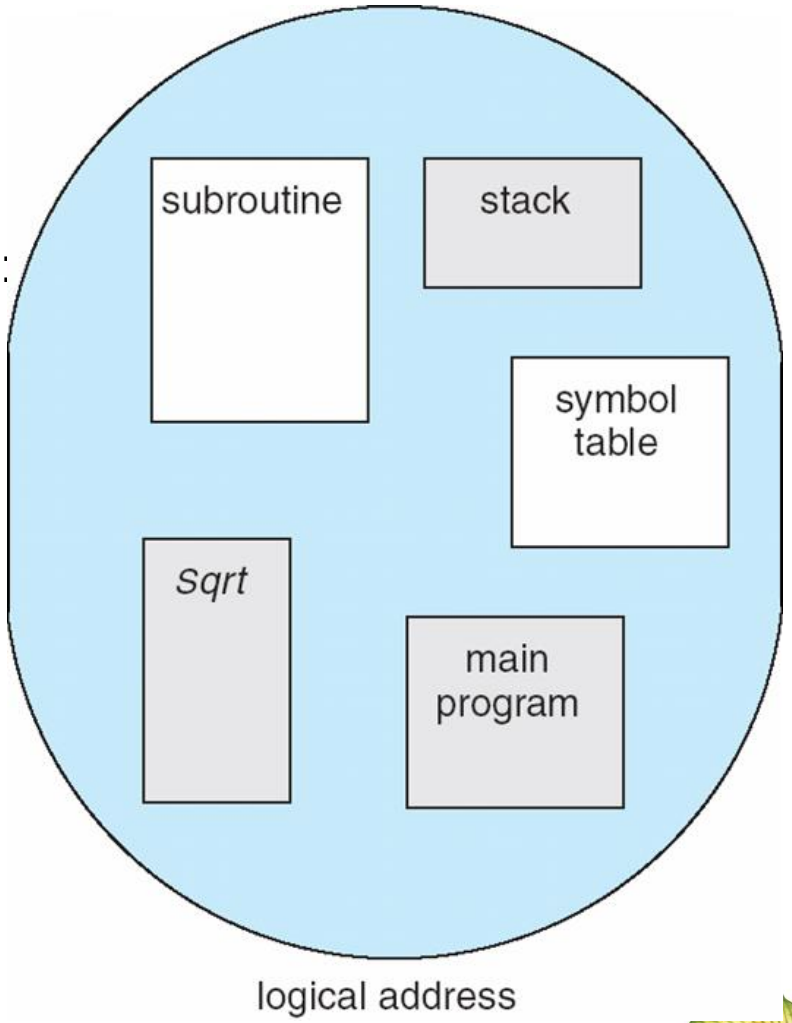
local variables, global variables

common block

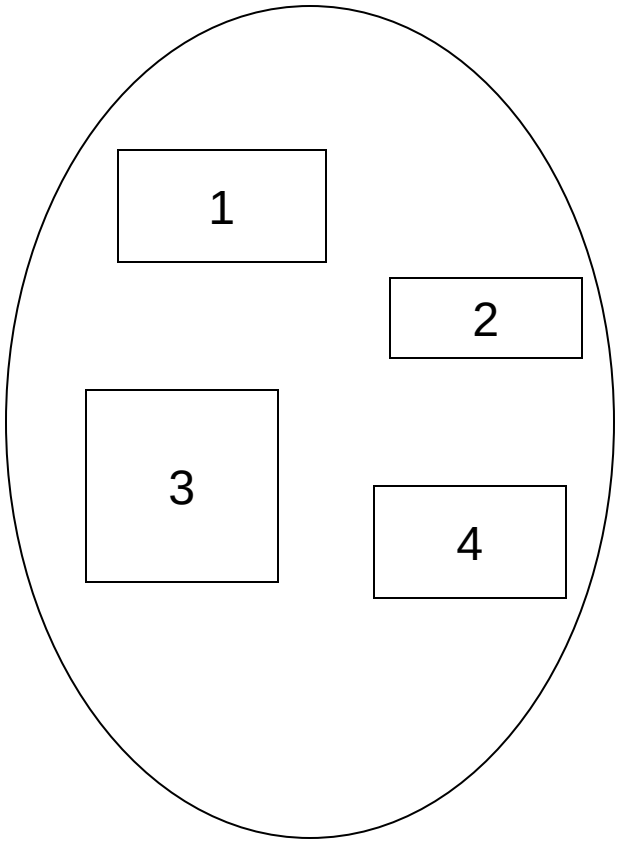
stack

symbol table

arrays

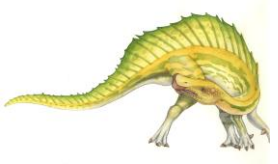


Logical View of Segmentation



user space

physical memory space



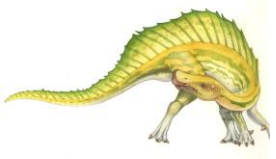
Segmentation Architecture

- Logical address consists of a two tuple:

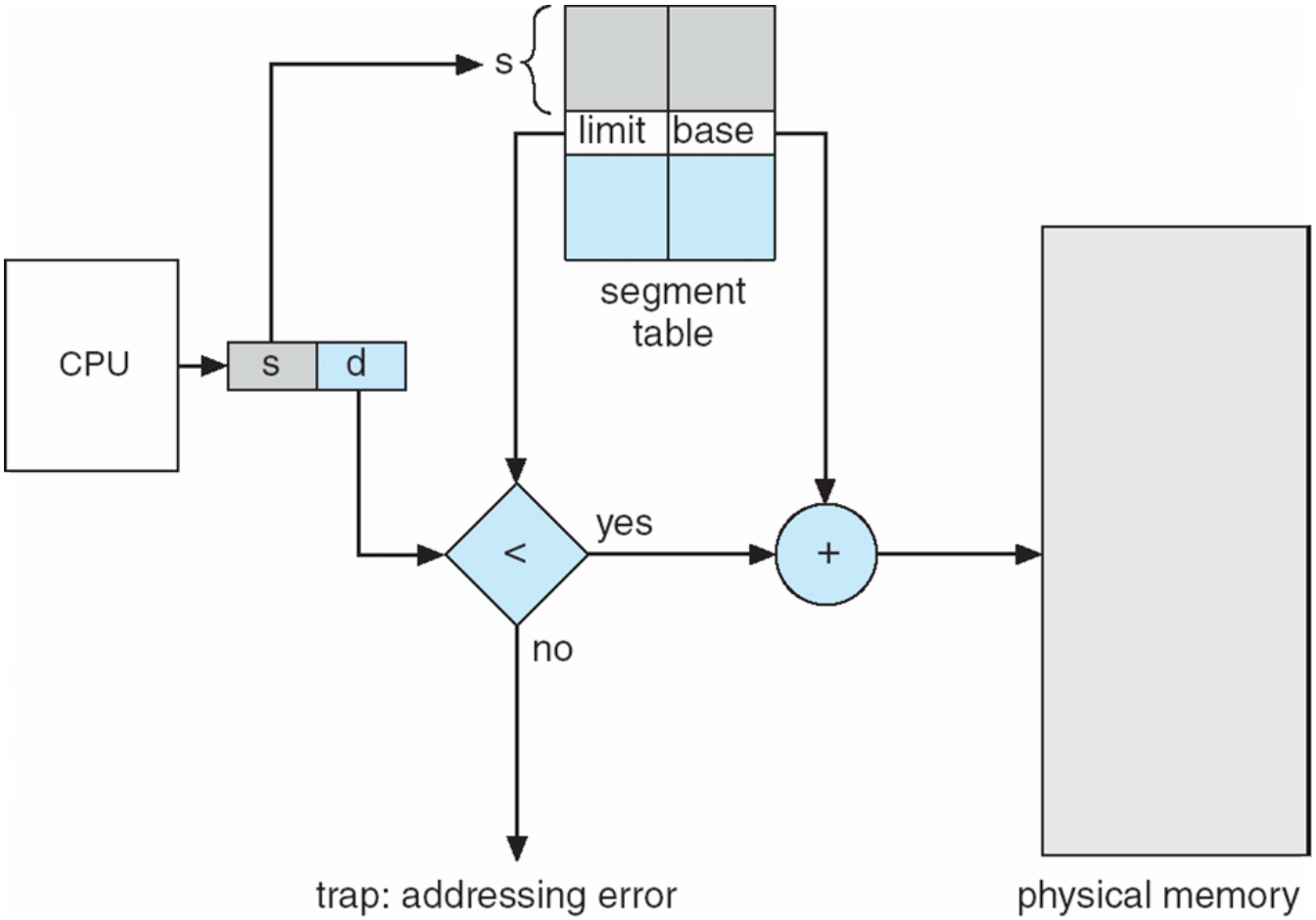
  <segment-number, offset>,

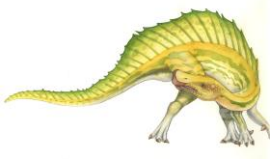
- **Segment table** – maps two-dimensional physical addresses; each table entry has:
  - **base** – contains the starting physical address where the segments reside in memory
  - **limit** – specifies the length of the segment
- **Segment-table base register (STBR)** points to the segment table's location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

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

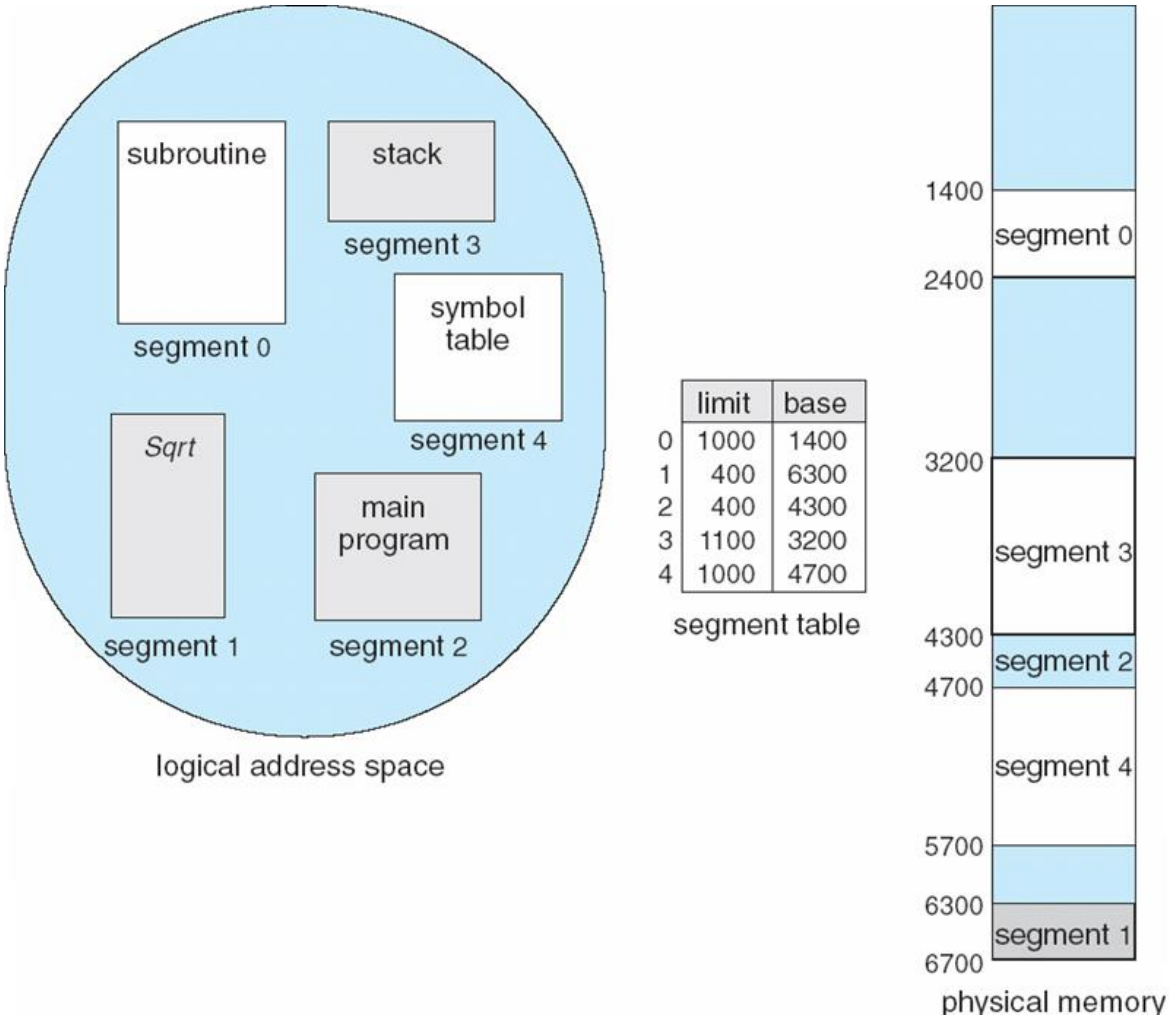


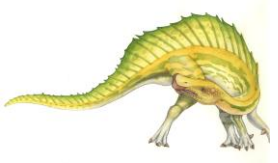
Segmentation Hardware





Example of Segmentation





End of Chapter 8