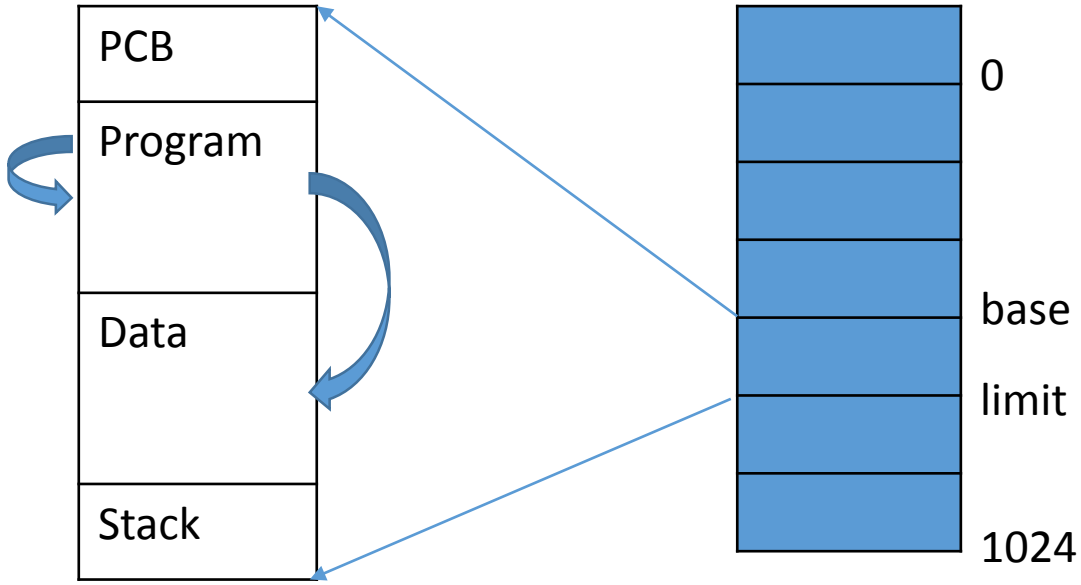
Agenda

- Assignment 4 due Friday Wednesday November 29
- Today's lecture
  - Memory management
  - Paging
  - Segmentation
  - Textbook Reading: 8.1-8.4, 8.6
- Next lecture
  - Course Evaluations
  - Virtual Memory (chapter 9)

What is Memory?

- Computer memory consists of a linear array of addressable storage cells that are similar to registers
- Program must be brought (from disk) into memory and placed within a process for it to be run
  - Each process has a base/limit address to specify its accessible addresses
  - ”segmentation fault” error generated when trying to access memory outside of the allowable address space



Memory Management Goals

- Make sure each process has sufficient memory
- Keep as many processes in memory as possible
- Allocate memory efficiently
  - Allocate as much as is needed for a process – not more
  - Leave some free space for new starting processes
  - Maximize memory utilization
- Memory is a finite resources
- These goals cannot be simultaneously met

OS Strategies

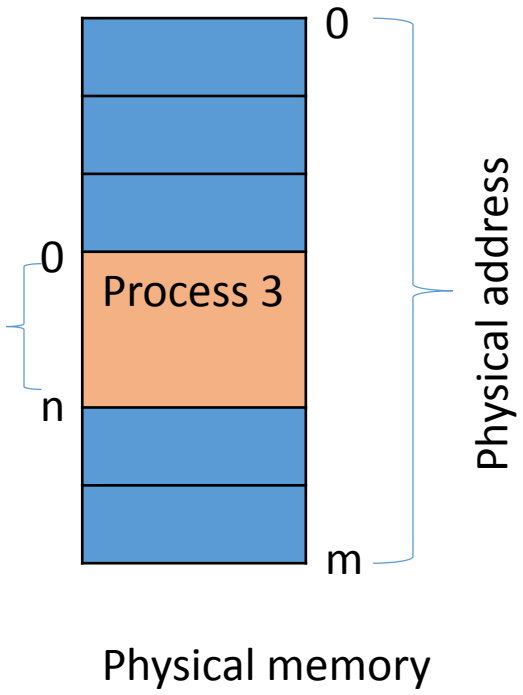
- Swapping
- Segmentation
- Paging
- Virtual memory

Memory Management Requirements

- Logical organization
- Physical organization
- Relocation
- Protection
- Sharing

Physical vs Logical Address Spaces

- The concept of a logical address space that is bound to a separate **physical address space** is central to proper memory management
  - **Logical address** – generated by the CPU;
    - Referred to as **virtual address**
    - Address space visible to a process (0-n)
  - **Physical address** – address seen by the memory unit
    - Address range of the physical memory (0-m)
- In primitive systems, the physical and logical address spaces were the same



Memory Management Techniques

- Fixed partitioning
- Dynamic partitioning
- Simple Paging
- Simple Segmentation
- Virtual Memory Paging
- Virtual Memory Segmentation

Fragmentation

- **External Fragmentation** – total memory space exists to satisfy a request, but it is not contiguous
  - Memory utilization declines
- **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**

Tradeoffs

- All partitioning systems lead to fragmentation (assuming contiguous allocation of processes)
- One can reduce external fragmentation for at the cost of internal fragmentation
- Can we do better?

Terminology

- **Frame**
  - A fixed-length block of main memory
- **Page**
  - A fixed-length block of data residing in secondary memory
- **Paging**
  - A page of data may temporarily be copied into a frame of main memory
- **Segment**
  - A variable-length block of data residing in secondary memory
- **Segmentation**
  - An entire segment may temporarily be copied into a region of main memory
- **Segmentation and Paging**
  - An entire segment may be divided into pages
  - Individual pages temporarily copied into main memory

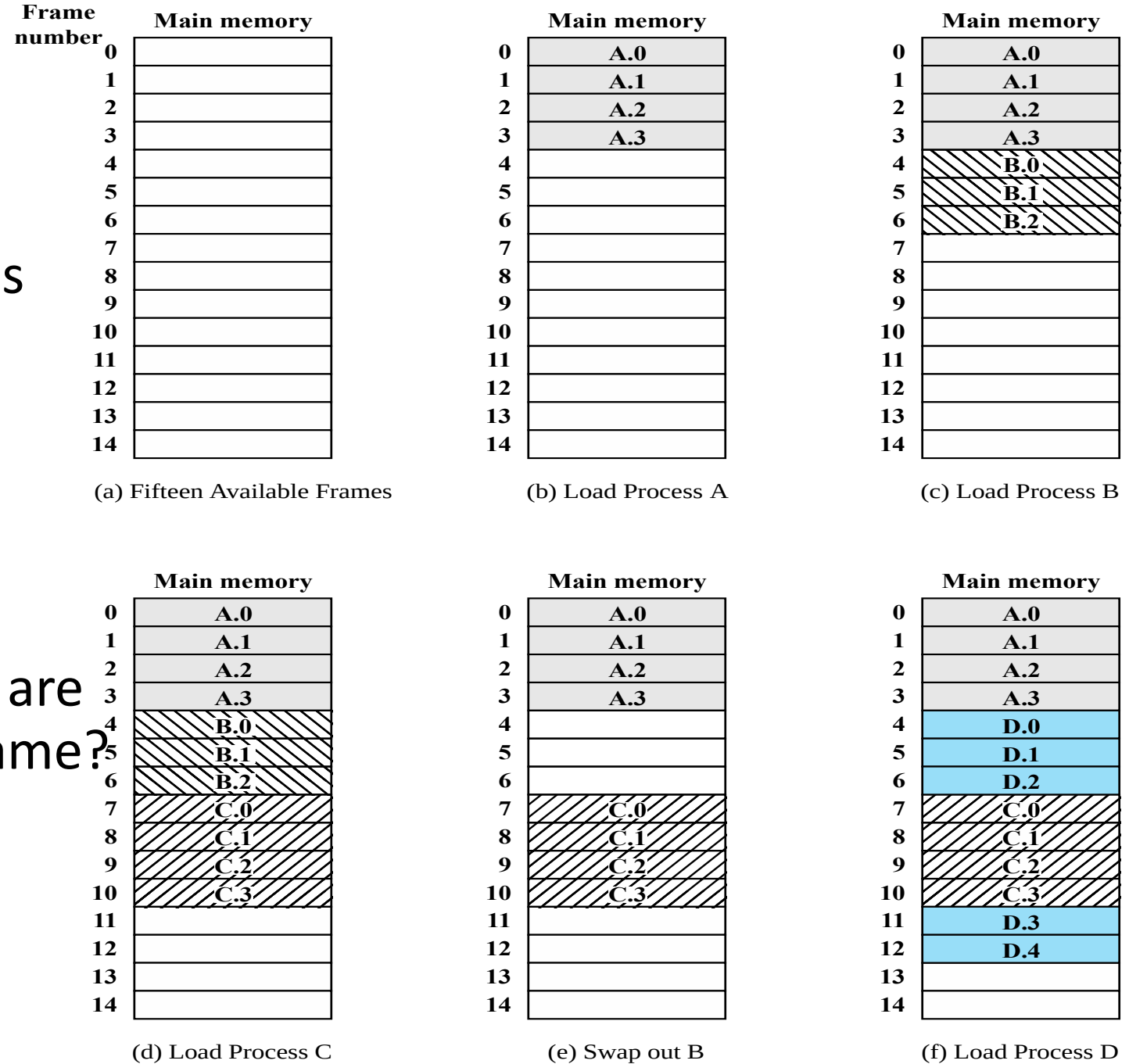
Paging

- Logical address space of a process can be noncontiguous; process is allocated physical memory whenever the latter is available
- Divide physical memory into fixed-sized blocks called **frames** (size is power of 2, between 512 bytes and 8,192 bytes)
- Divide logical memory into blocks of same size called **pages**

Pages	Frames
• Chunks of a process	• Available chunks of memory

Paging

- Keep track of all free frames
- To run a program of size ***n*** pages, need to find *n* free frames and load program
  - E.g., process A needs 4 free frames
- What happens when there are no sufficient contiguous frame?
  - E.g., process D needs 5 free frames



Paging

- Set up a *page table* to translate logical to physical addresses
  - Role of the MMU
  - There is a page table for each process, managed and held by the OS
- Protection: with each entry in page table associate:
    - validation bit = 0 $\Rightarrow$ illegal page
    - read/write/execute privileges
- Keeps track of how main memory is used (free/available) using a *frame table*
  - usually one frame table managed by the OS

Page Table Example

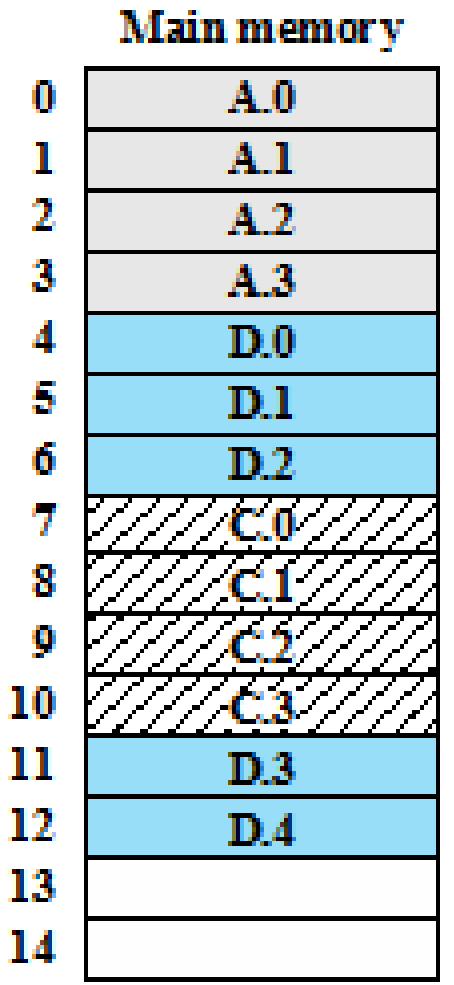
Process A
0
1
2
3

Process B

Process C
7
8
9
10

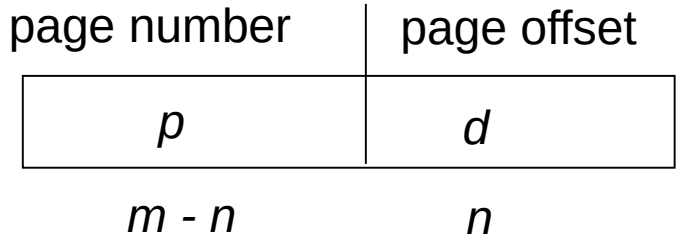
Process D
4
5
6
11
12

Frame Table
13
14



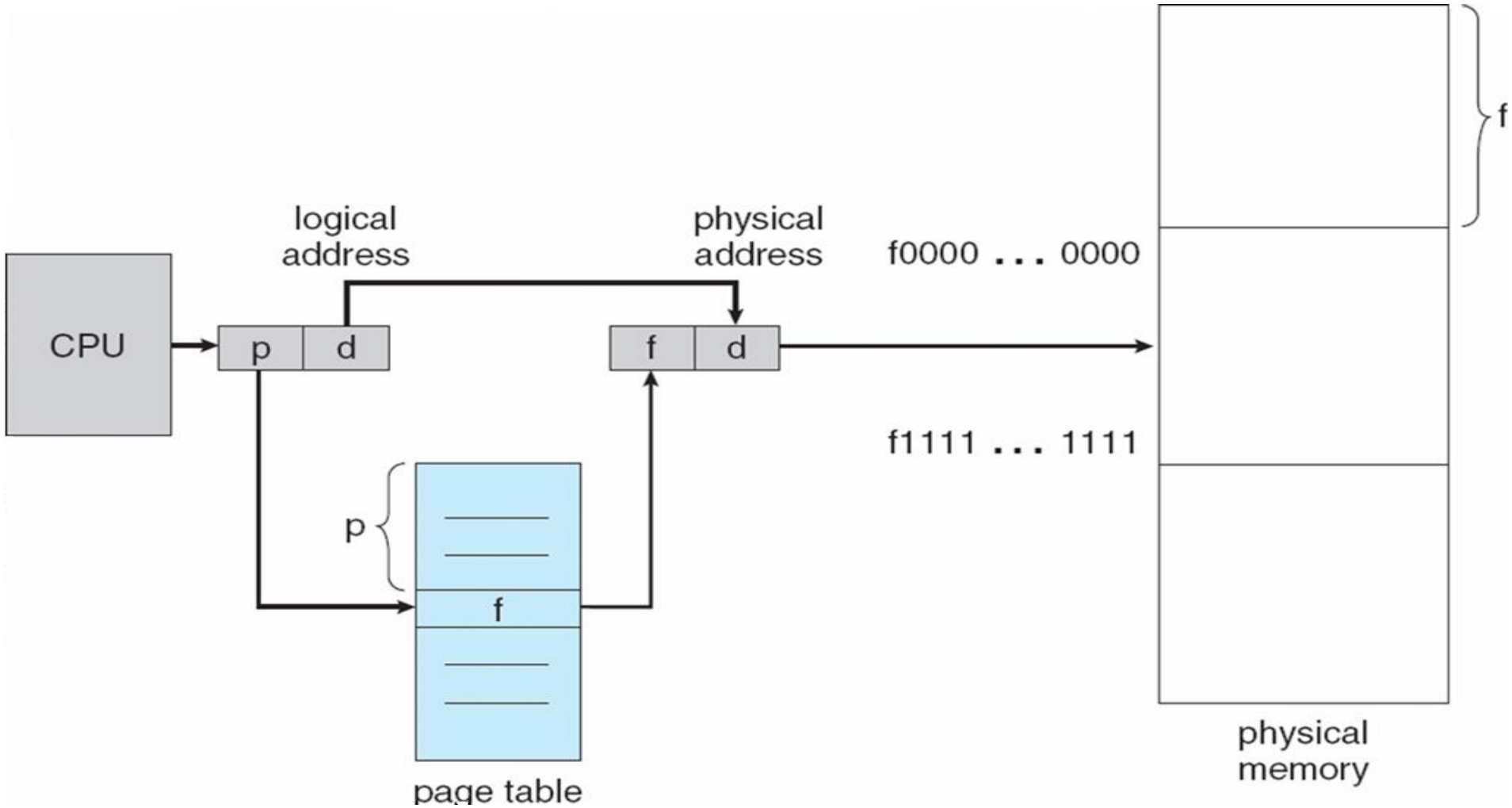
Address Translation Scheme

- The 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

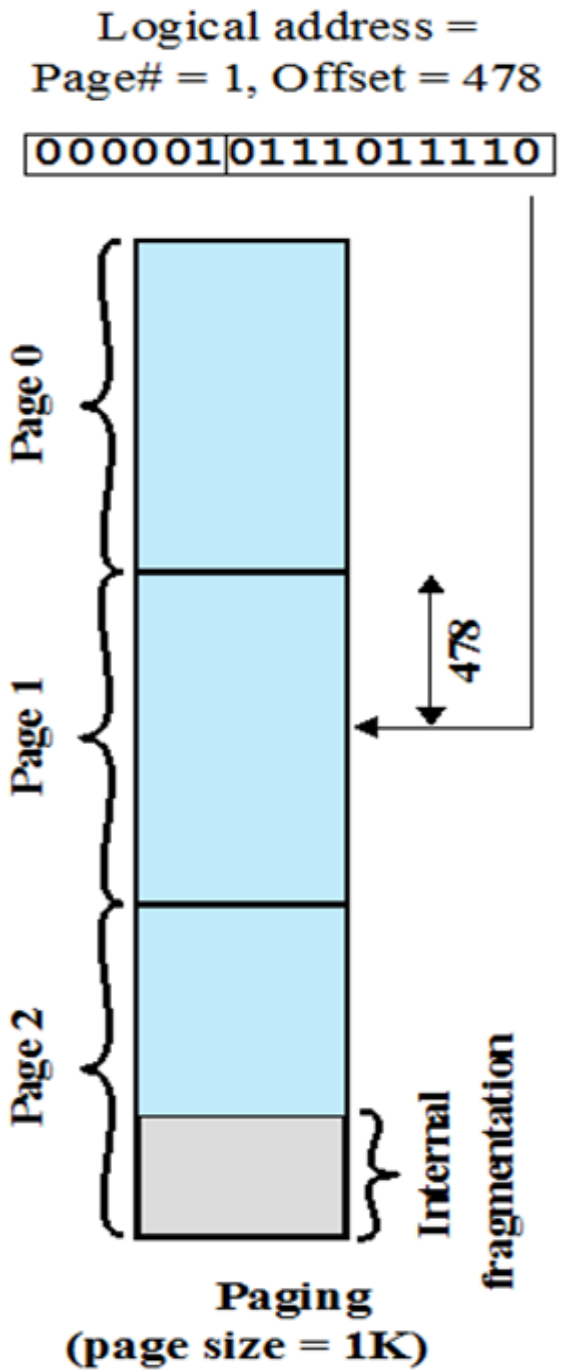


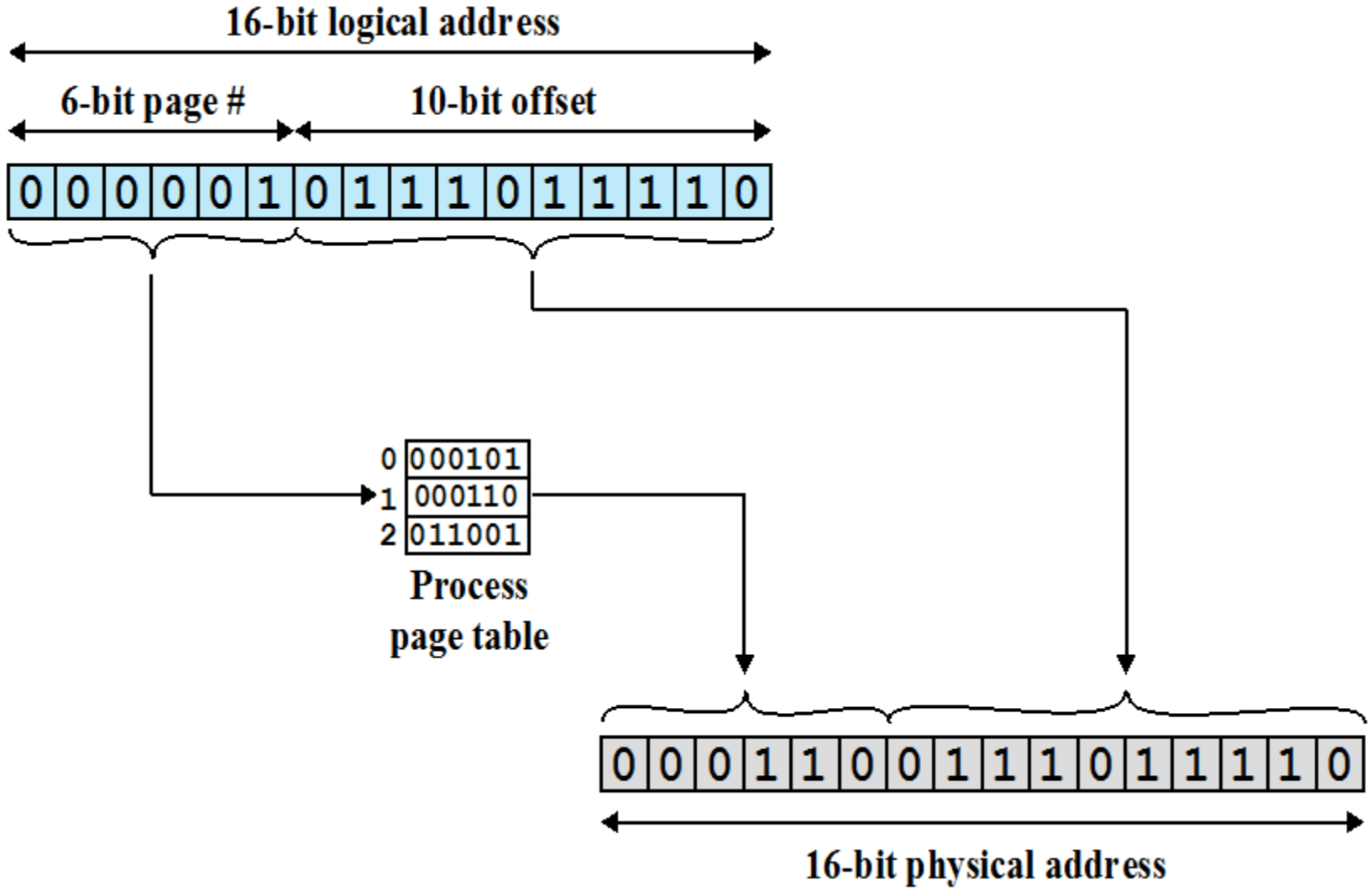
  - For a given logical address space 2^m *and page size* 2^n

Address Translation



Address Translation





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 **associative memory** or **translation look-aside buffers (TLBs)**
- Some TLBs store **address-space identifiers (ASIDs)** in each TLB entry – uniquely identifies each process to provide address-space protection for that process

Paging

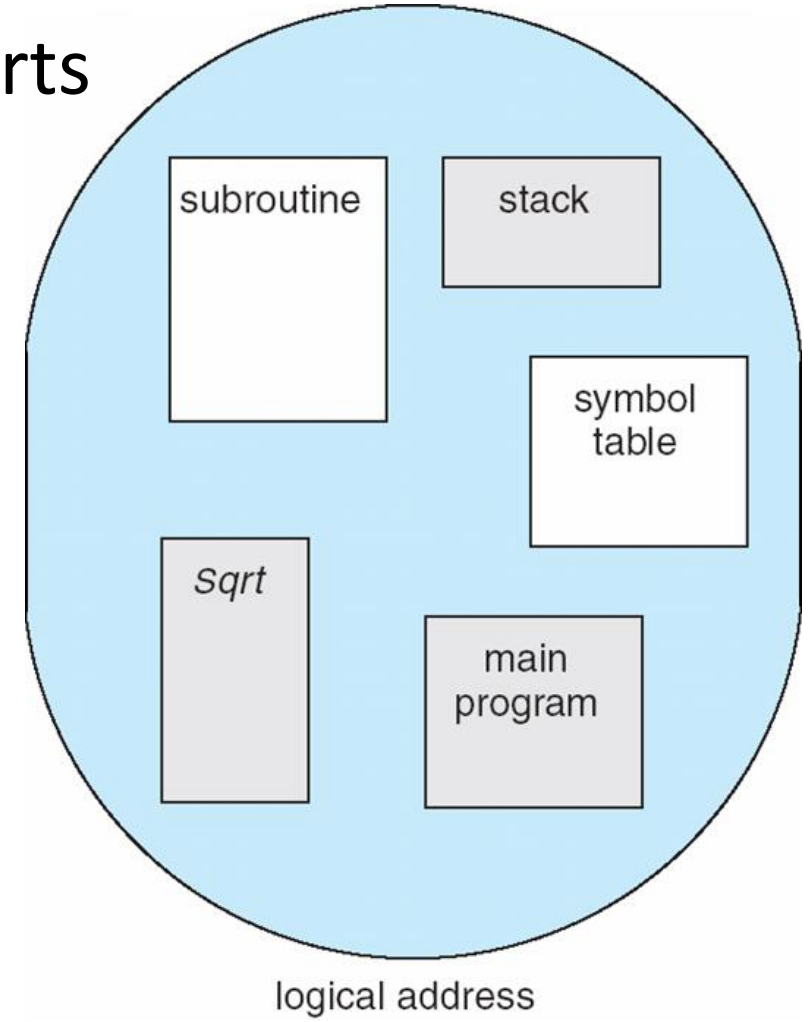
- Similar to fixed partitioning
- Partitions are smaller
- A program may occupy more than one partition
  - Partitions need not be contiguous
- May lead to internal fragmentation
- Slower address translation
  - Can be solved using Multi-level page tables

User’s view of a program

- As a programmer, can you describe your view of the memory?
- Is this view visible using Paging?
- Would it help if the memory management scheme used supported user’s view of a program?

Segmentation

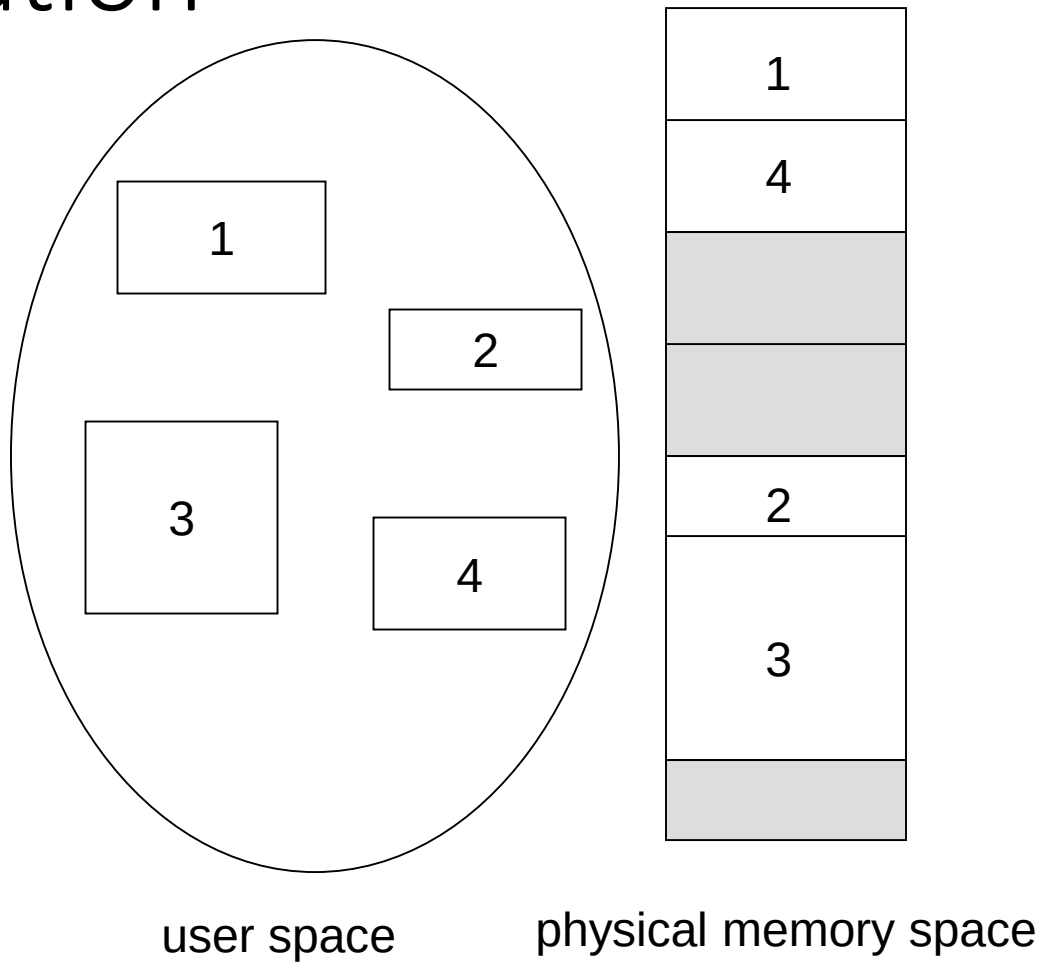
- Memory-management scheme that supports user view of memory
- A program can be subdivided into segments
  - May vary in length
  - There is a maximum length
- Addressing consists of two parts:
  - Segment number
  - An offset
- Similar to dynamic partitioning
  - Non-contiguous allocation
- Eliminates internal fragmentation



User's view of a program

Logical View of Segmentation

- Usually visible
- Provided as a convenience for organizing programs and data
- Typically the programmer will assign programs and data to different segments
- For purposes of modular programming the program or data may be further broken down into multiple segments
  - The principal inconvenience of this service is that the programmer must be aware of the maximum segment size limitation



Segmentation Architecture

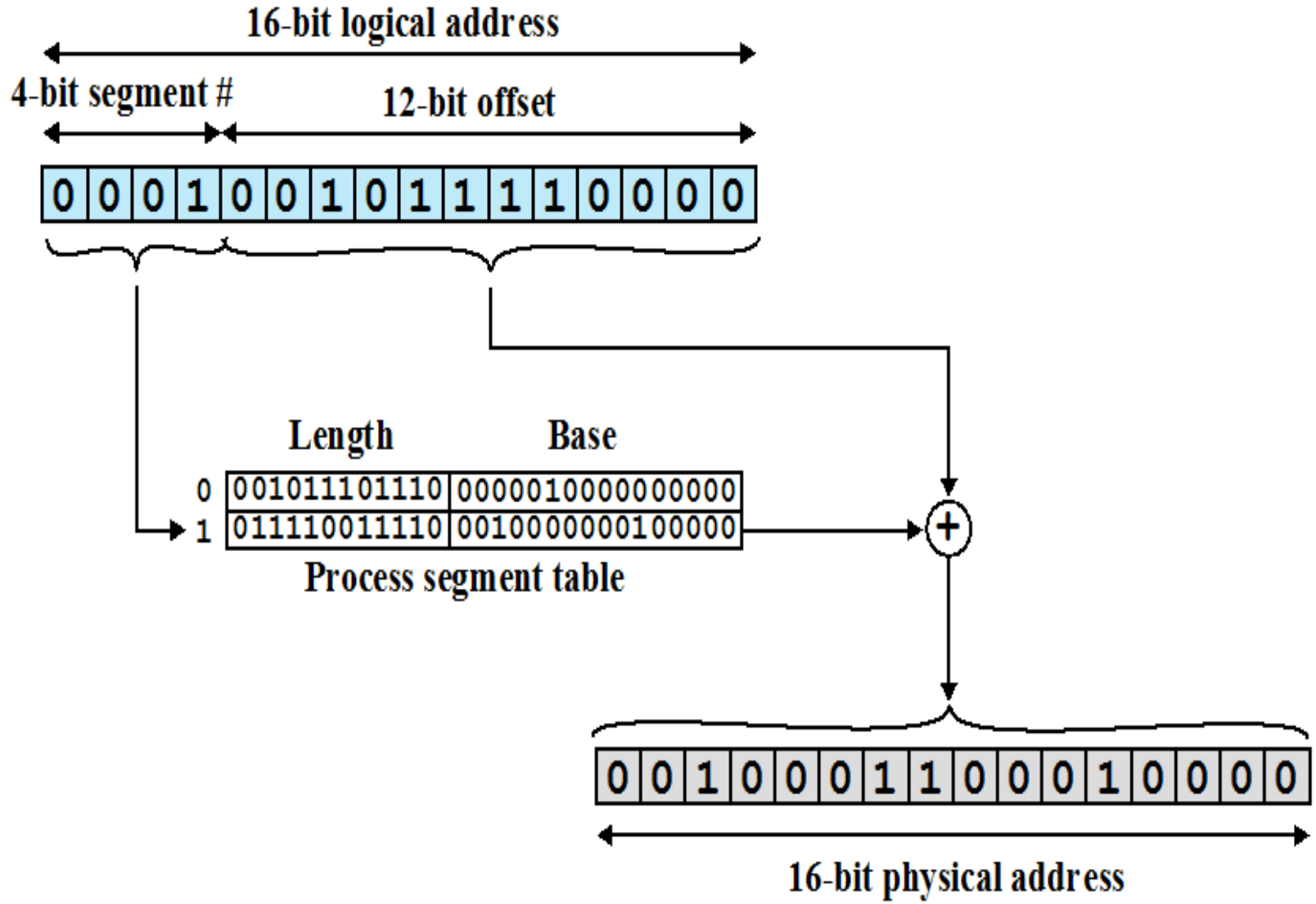
- Logical address consists of a two tuple:

  <segment-number, offset>,

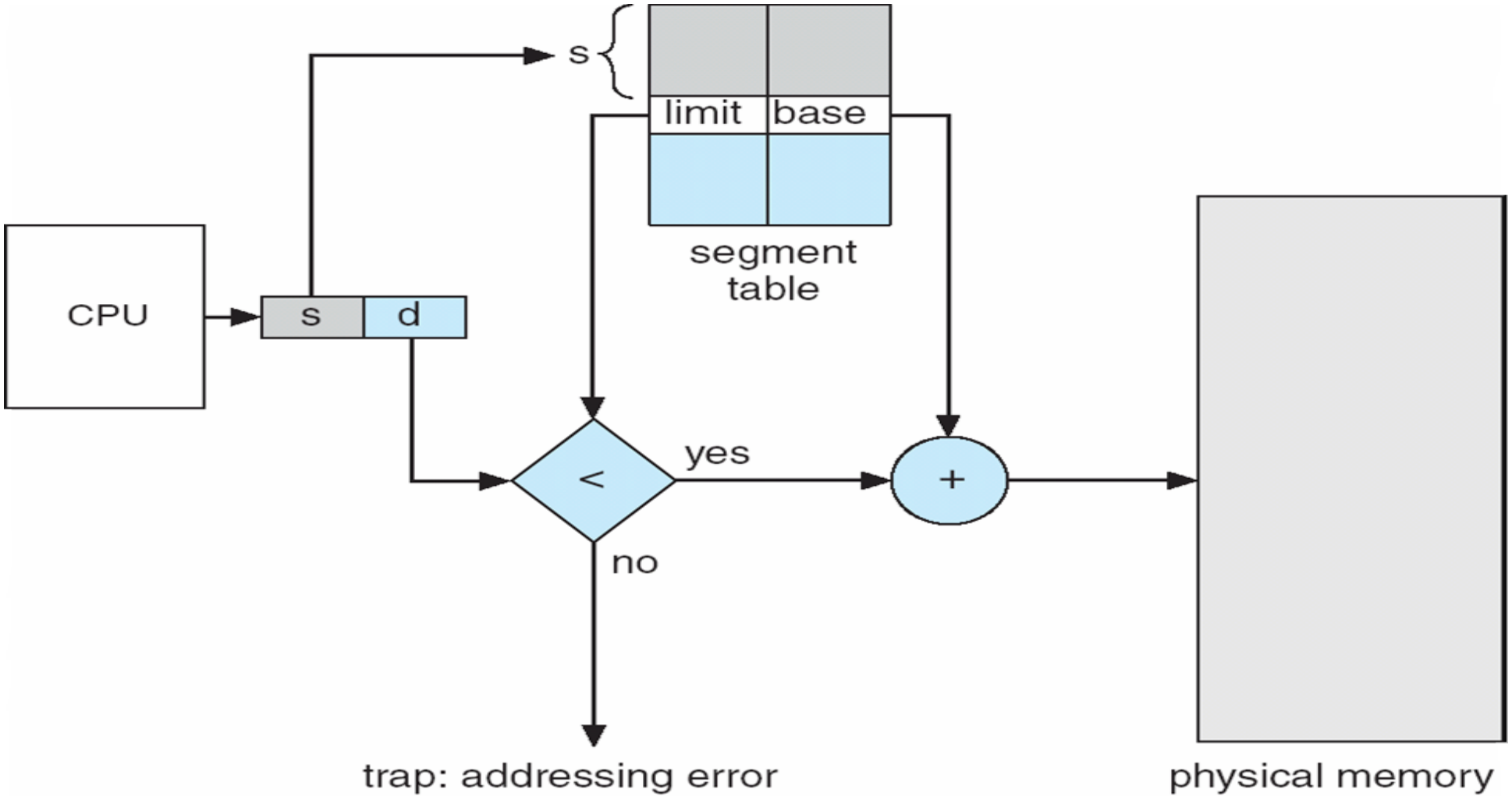
- **Segment table** – maps two-dimensional physical addresses; each table entry has:
  - **base** – contains the starting physical address where the segments reside in memory
  - **limit** – specifies the length of the segment
- **Segment-table base register (STBR)** points to the segment table's location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

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

Address Translation

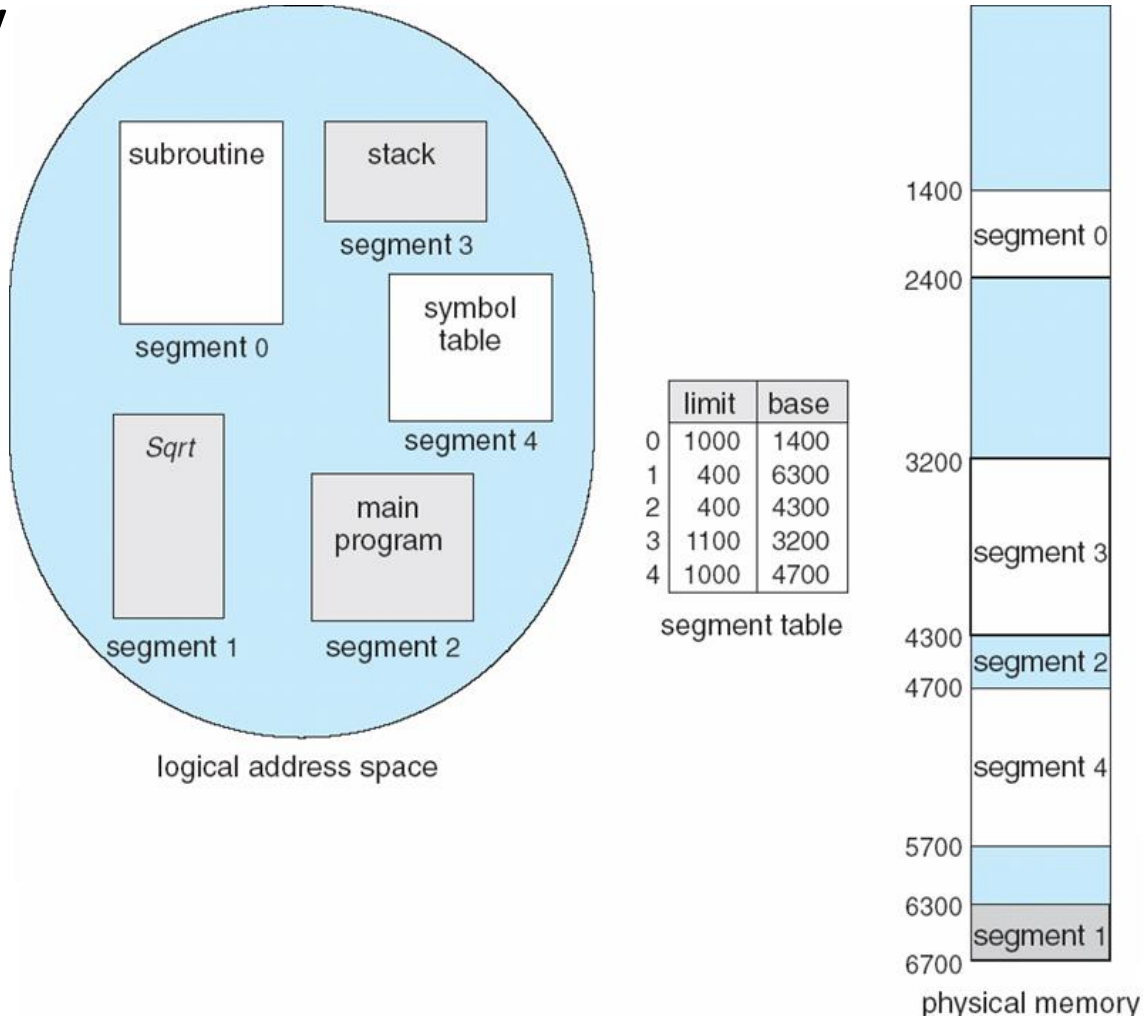


Segmentation Hardware



Segmentation Example

- Since segments vary in length, memory allocation is a dynamic storage-allocation problem
- A segmentation example is shown in the following diagram



Reflections on Segmentation

- Advantages
  - Allocation flexibility
  - Easier to increase the size of a segment than a process
  - Fast address translation
  - Segments swapping cheaper than process swapping
- Disadvantages
  - Processes are no longer contiguous
    - Requires a more complex pointer arithmetic
  - Processes may only access a fixed number of segments at a time
    - Data movement between multiple segments becomes more complex

Reflections on Segmentation and Paging

- Two characteristics fundamental to memory management:
    1) All memory references are logical addresses that are dynamically translated into physical addresses at run time
    2) A process may be broken up into a number of pieces that don't need to be contiguously located in main memory during execution
- If these two characteristics are present, it is not necessary that all of the pages or segments of a process be in main memory during execution