Segmentation

Module 5 self study material

Operating systems 2020

1DT003, 1DT044 and 1DT096

Spring 2020 karl.marklund@it.uu.se Uppsala university

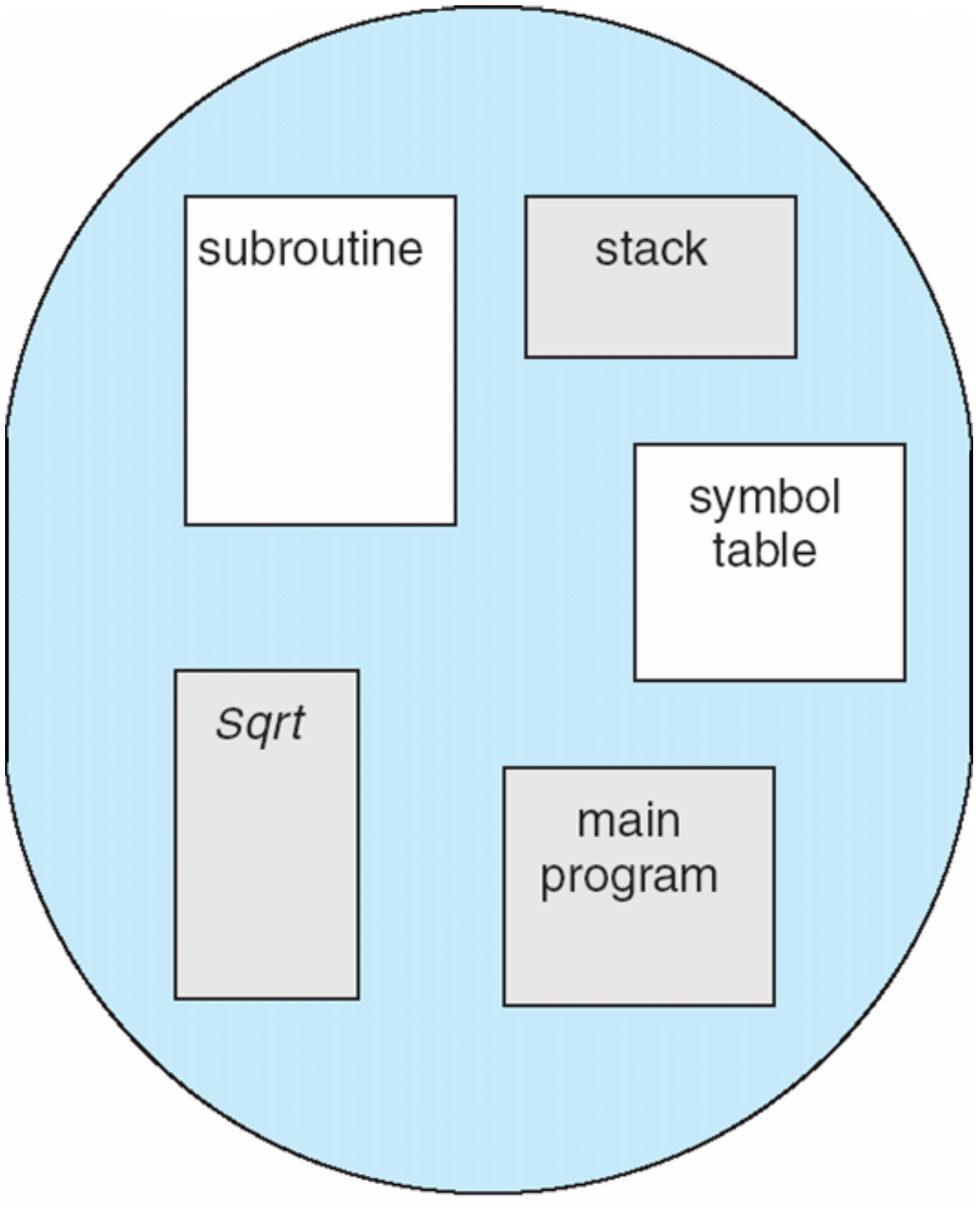
Segmentation

Memory segmentation is the division of
a computer's primary memory into segments or sections.

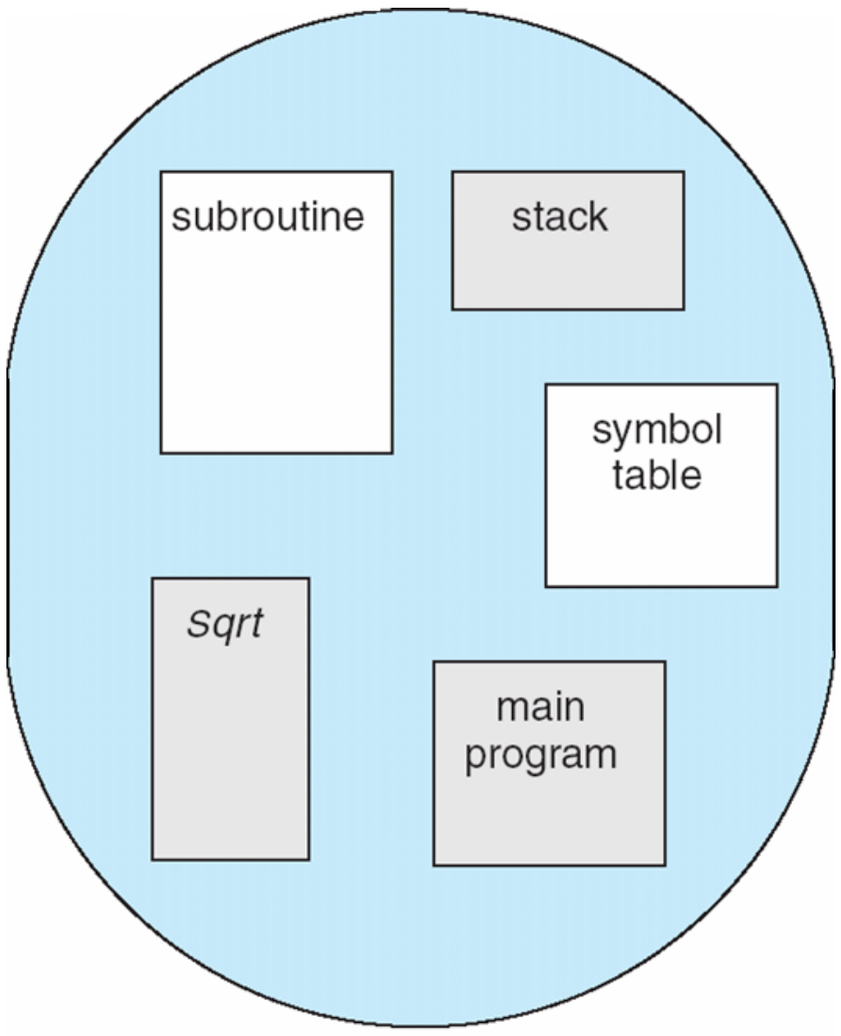
In a computer system using segmentation, a reference to a memory location includes a value that identifies a segment and an offset (memory location) within that segment.

Source: https://en.wikipedia.org/wiki/Memory_segmentation 2018-02-22

- 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

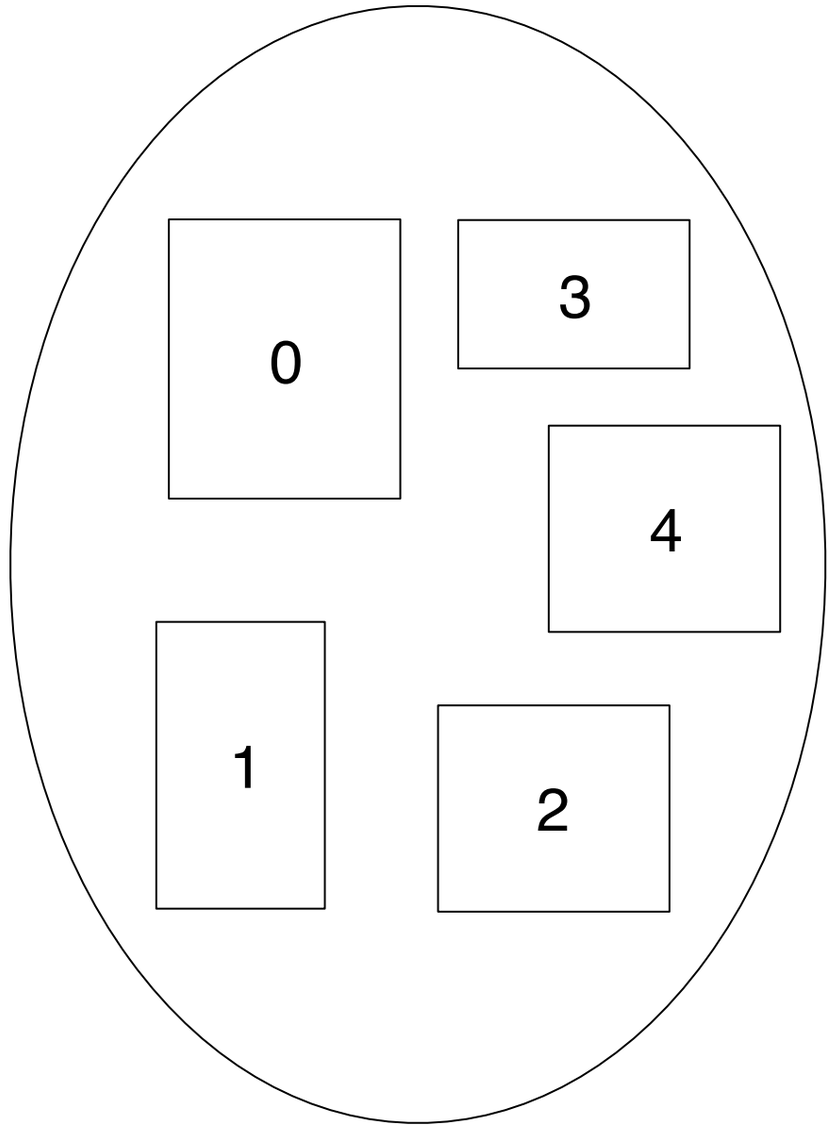


User's view of a program

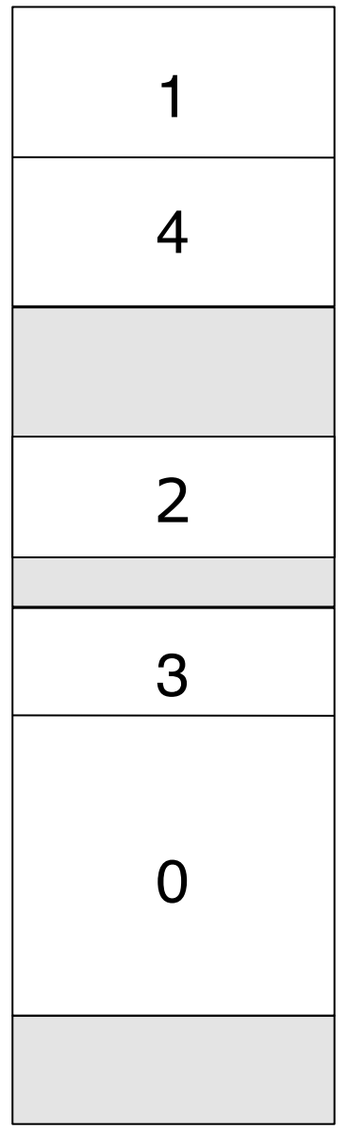


logical address

Logical view of segmentation



user space



physical memory space

Observation: segments can be of arbitrary length

Segmentation

In a system using segmentation computer memory addresses consist of a **segment id** and an **offset** within the segment.

Each **segment** has a **length** and set of **permissions** (for example, read, write, execute) associated with it. Sharing occurs at segment level.

A hardware memory management unit (**MMU**) is responsible for **translating** the **segment** and **offset** into a **physical memory address**, and for **performing checks** to make sure the translation can be done and that the reference to that segment and offset is permitted.

Segmentation fault

A process is only allowed to make a reference into a segment if the type of reference is allowed by the permissions, and if the offset within the segment is within the range specified by the length of the segment.

Otherwise, a hardware exception such as a **segmentation fault** is raised.

Segmentation architecture (1)

- 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 architecture (2)

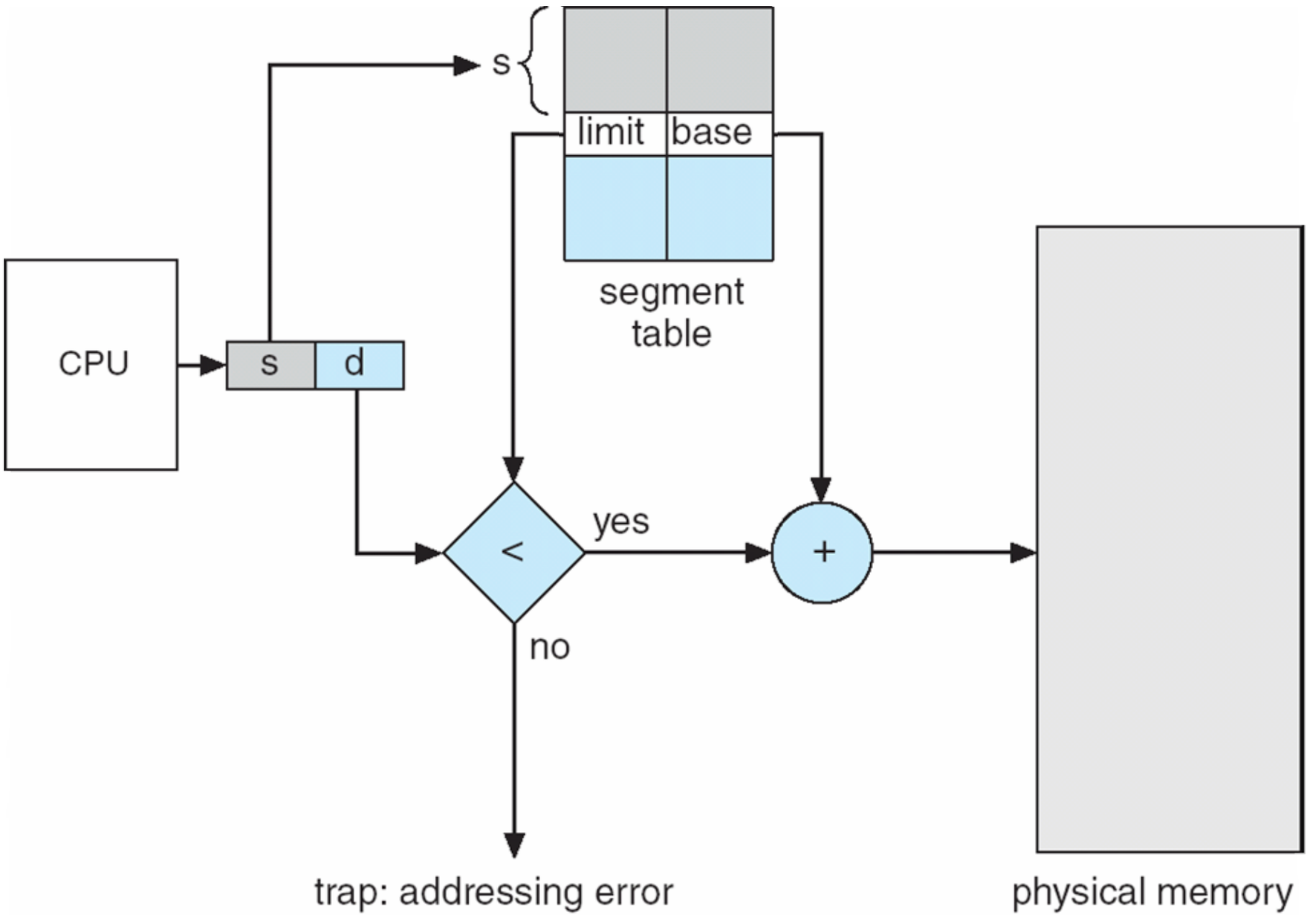
- **Protection** - with each entry in segment table associate:
  - validation bit = 0 ⇒ illegal segment
  - read/write/execute privileges
- Protection bits associated with segments; **code sharing** occurs at **segment level**.
- Since segments vary in length, memory allocation is a dynamic storage-allocation problem.

Segmentation hardware

s - segment id

d - segment offset

limit - segments can have individual lengths

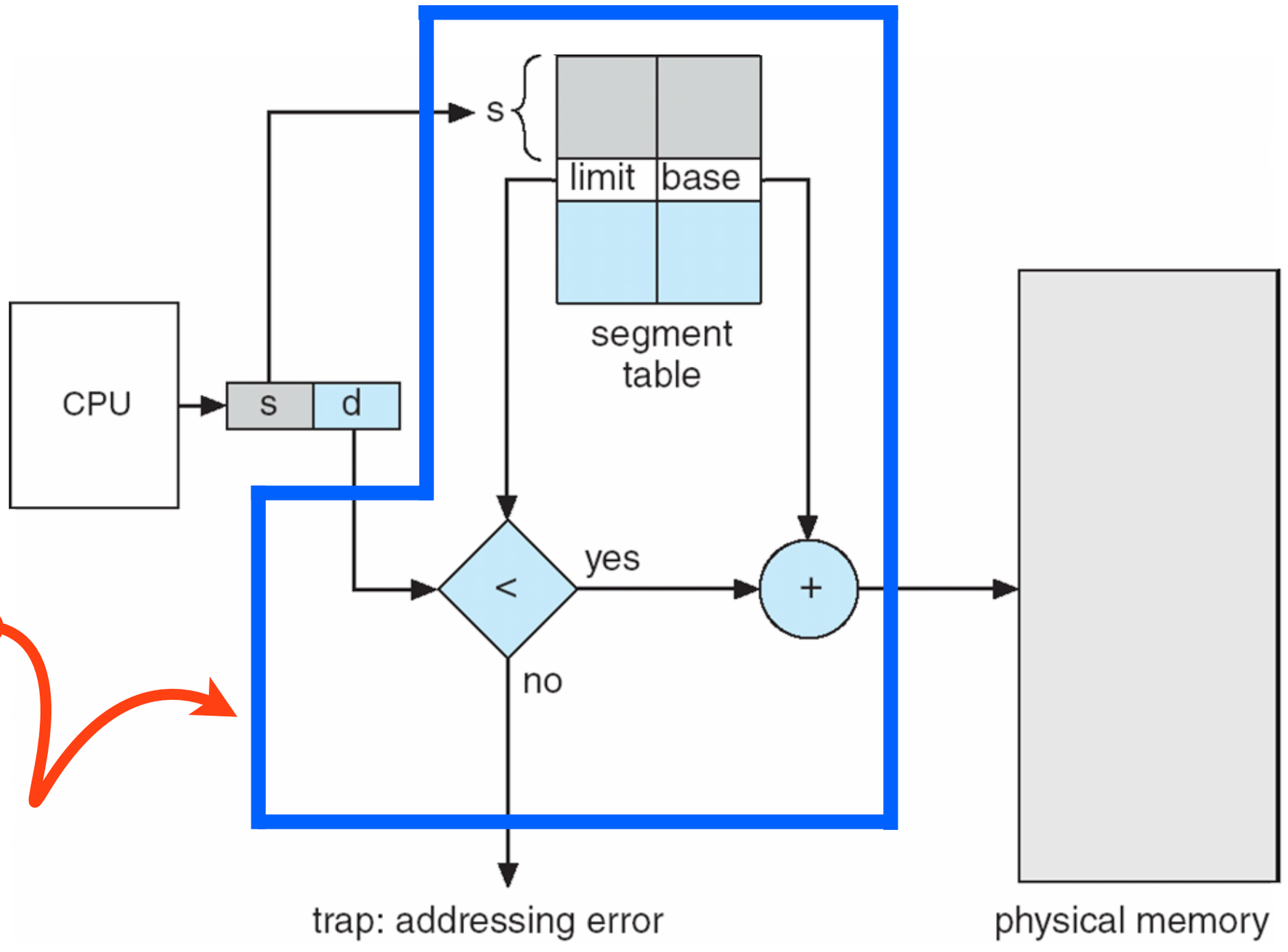


Segmentation hardware

s - segment id

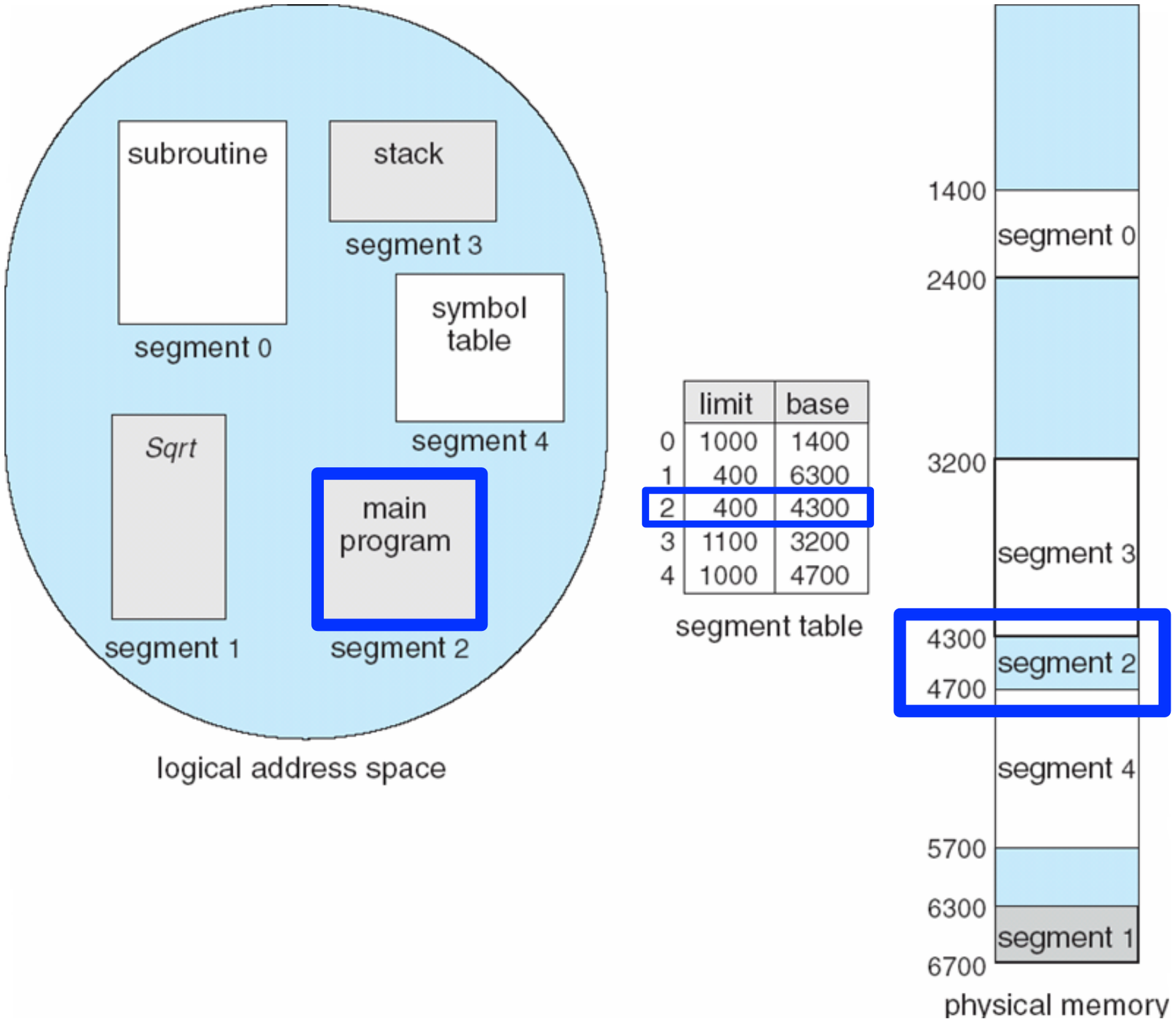
d - segment offset

limit - segments can have individual lengths



MMU

Segmentation (example)



Segmentation and virtual memory

Segments may also be used to implement **virtual memory**.

- Each segment has an associated flag indicating whether it is present in main memory or not.
- If a segment is accessed that is not present in main memory, an exception is raised, and the operating system will read the segment into memory from secondary storage.

Segmentation is one method of implementing **memory protection**. Paging is another, and they can be combined.

Segmentation without paging

An implementation of virtual memory on a system using segmentation without paging requires that:

- **entire segments** be **swapped** back and forth between main memory and secondary storage.

When a segment is swapped in, the operating system has to:

- **allocate** enough **contiguous free memory** to hold the entire segment.

Often memory fragmentation results in there being not enough contiguous memory even though there may be enough in total.

Segmentation with paging

Instead of an actual memory location the segment information includes the address of a page table for the segment.

- When a program references a memory location the offset is translated to a memory address using the page table.

- A segment can be extended simply by allocating another memory page and adding it to the segment's page table.

Pages of the segment can be located anywhere in main memory and **need not be contiguous**. This usually results in:

- A reduced amount of input/output between primary and secondary storage.

- Reduced memory fragmentation.

Intel Pentium segmentation (1)

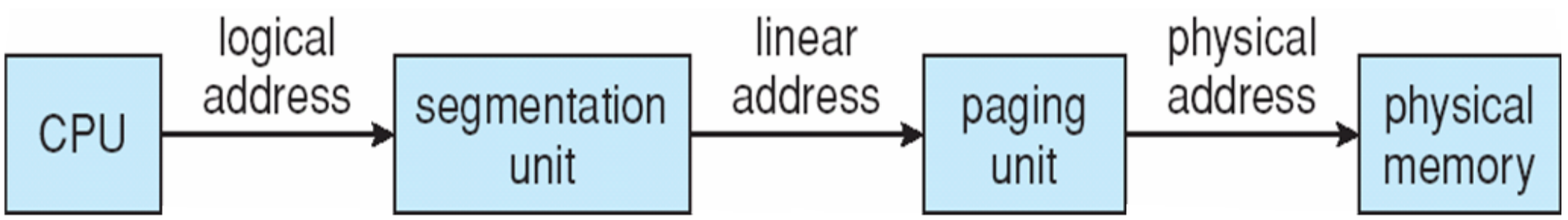
In the x86 Pentium implementation of segmentation the segment table, rather than pointing to a page table for the segment, contains the segment address in **virtual linear memory**.

- This address is then mapped to a physical address using a separate page table.

- Unlike other paged implementations of segmentation this prevents segments from dynamically growing in size.

Intel Pentium segmentation (2)

1. CPU generates logical address
2. Given to segmentation unit
3. Which produces linear addresses
4. Linear address given to paging unit
5. Which generates physical address in main memory
6. Paging units form equivalent of MMU



The Pentium uses a two-level paging scheme.