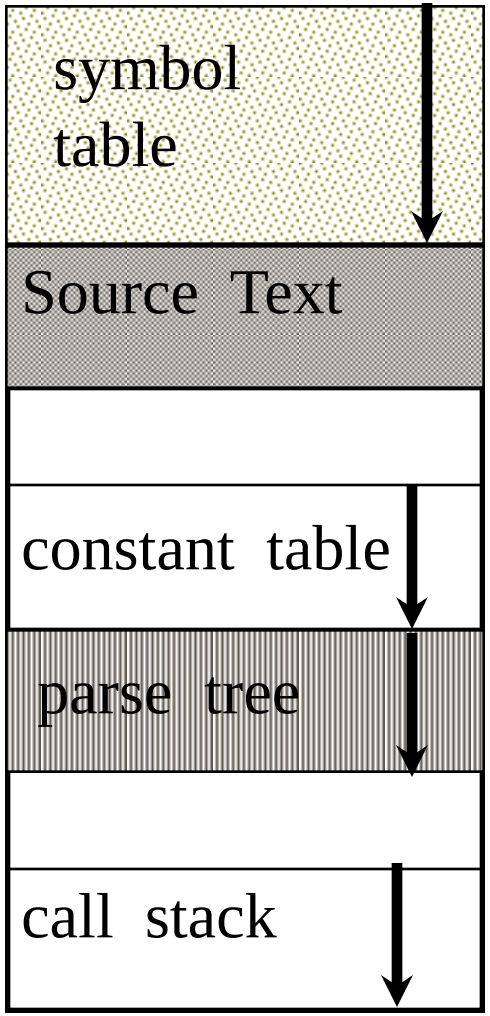
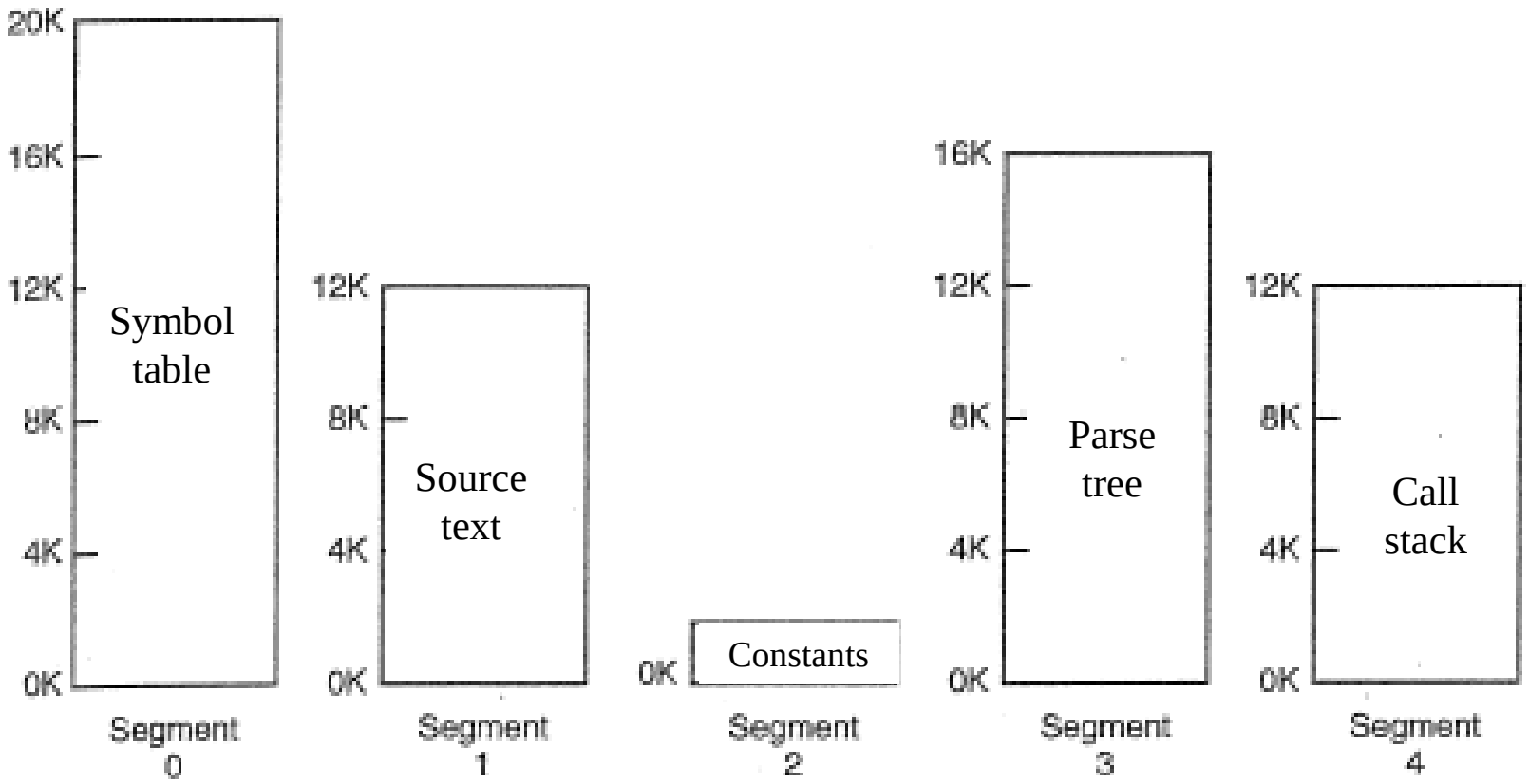
Segmentation

- ***Several*** address spaces ***per process***
- a compiler needs segments for
  - source text
  - symbol table
  - constants segment
  - stack
  - parse tree
  - compiler executable code
- Most of these segments ***grow*** during execution



Users view of segments



A segmented memory allows each table to grow or shrink independently of the other tables.

Segmentation - segment table

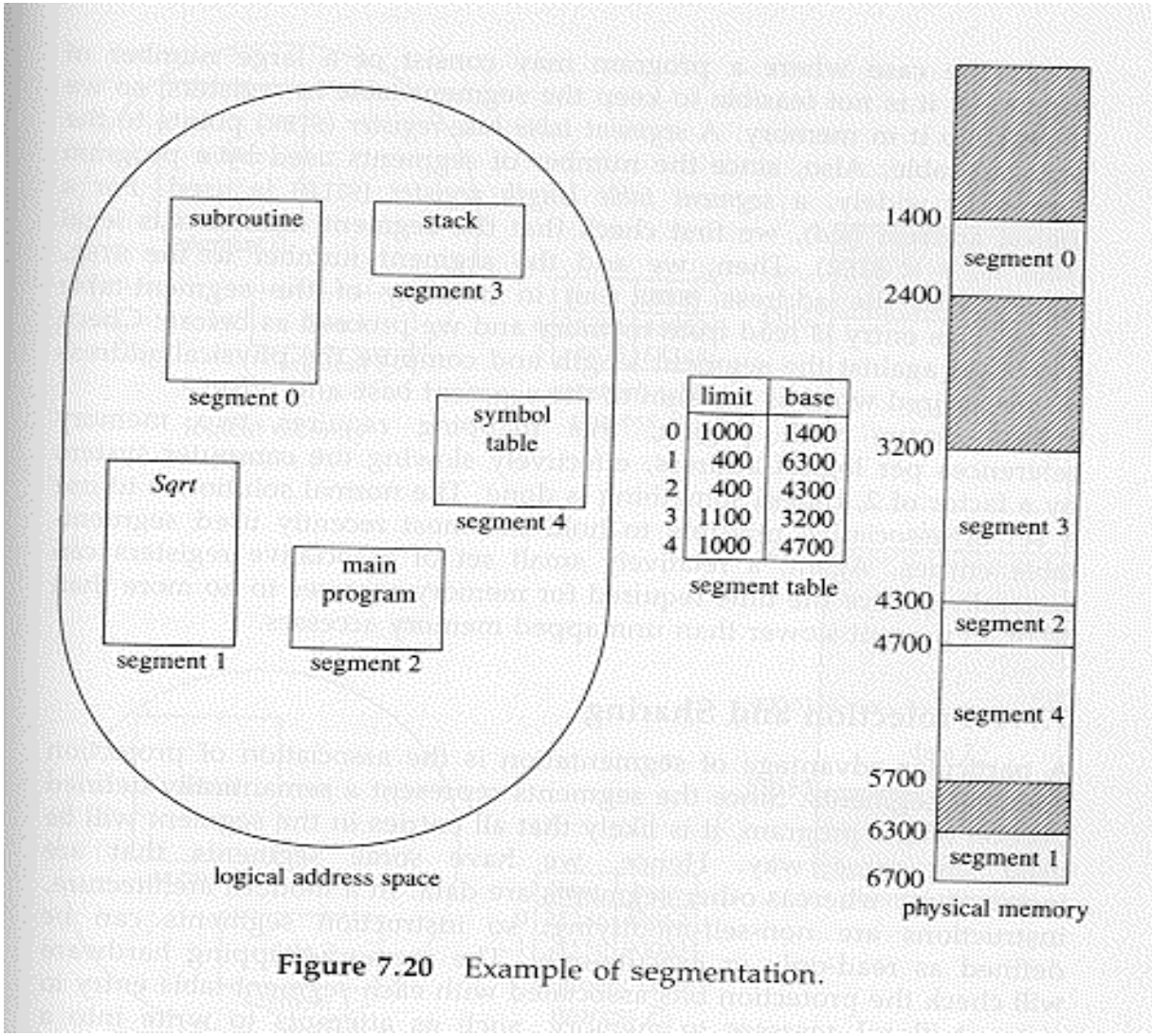


Figure 7.20 Example of segmentation.

Segmentation Hardware

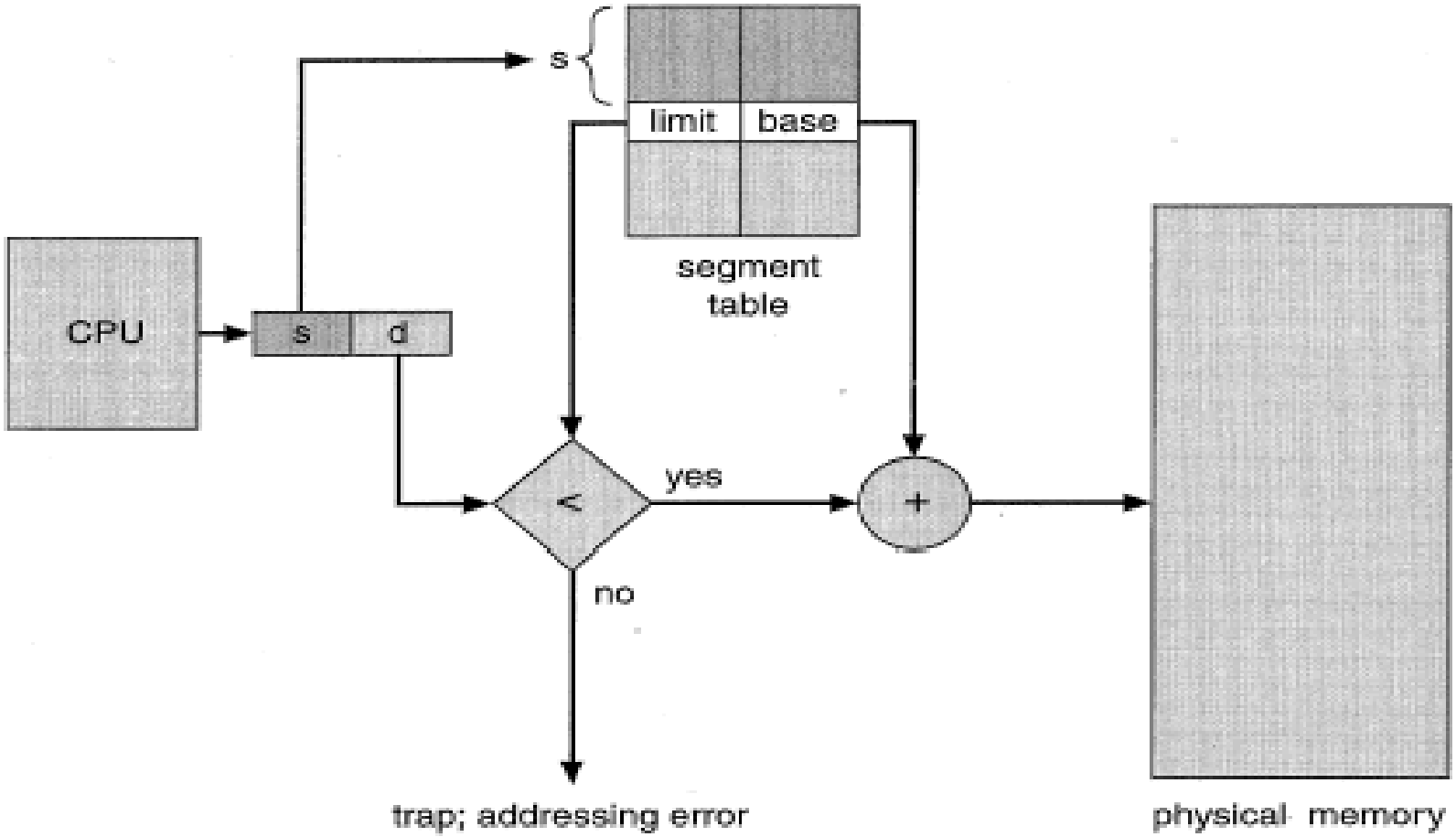


Figure 8.23 Segmentation hardware.

Segmentation vs. Paging

consideration	Paging	Segmentation
Need the program be aware of the technique ?	no	yes
How many per-process virtual address spaces ?	1	many
Can the total address space exceed physical memory ?	yes	yes
Can procedures and data be distinguished ?	no	yes
Sharing of procedures among users facilitated ?	no	yes
Motivation for the technique	Get larger linear space, eliminate external fragmentation	Programs and data in logical independent address spaces, sharing and protection made simpler

Segmentation pros and cons

- Advantages:
  - Growing and shrinking independently.
  - Sharing between processes simpler
  - Linking is easier
  - Protection easier
- Disadvantages:
  - Pure segmentation --> external Fragmentation revisited
  - Segments may be very large. What if they don't fit into physical memory?

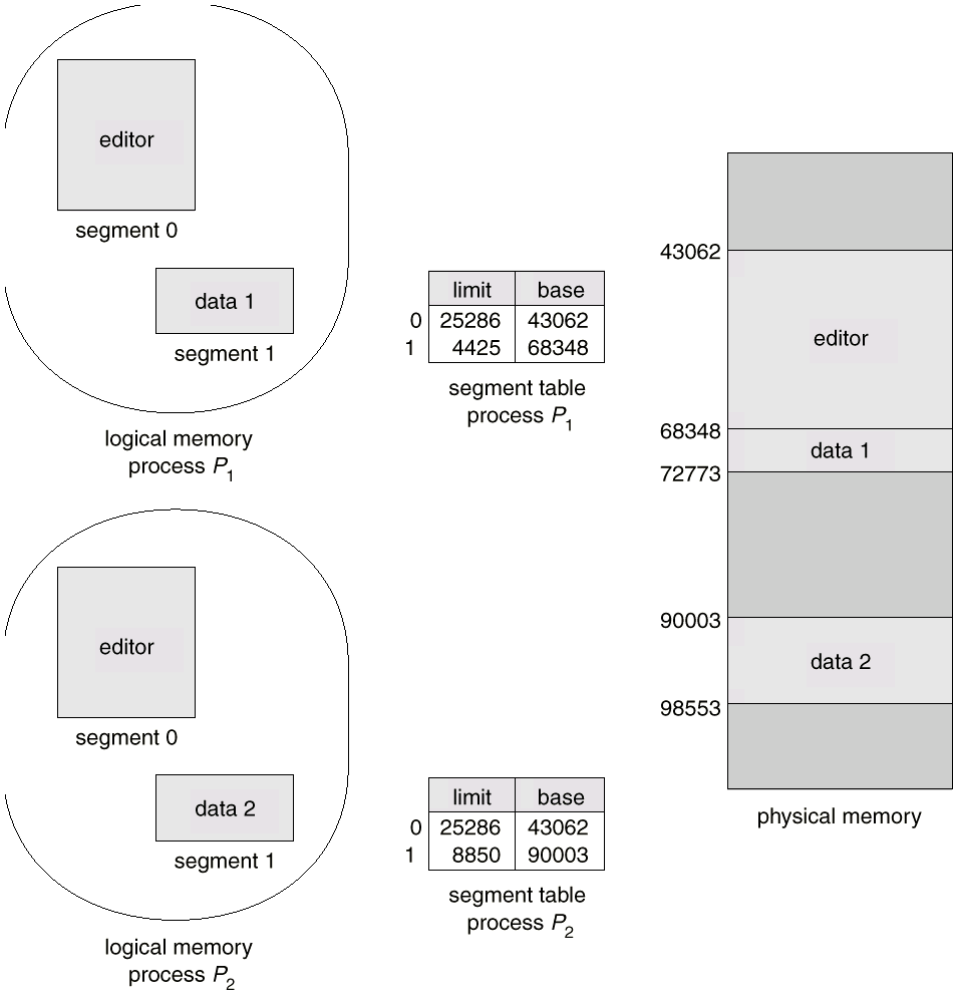
Segmentation Architecture

- Logical address composed of the pair <segment-number, offset>
- *Segment table* – maps to linear address space; each table entry has:
  - *base* – contains the starting linear address where the segment resides 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 (Cont.)

- *Protection*: each segment table entry contains:
  - 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 (external fragmentation problem)

Sharing of segments



Segmentation with Paging

- Segments may be too large
- Cause external fragmentation
- The two approaches may be combined:
  - Segment table.
  - Pages inside a segment.
  - Solves fragmentation problems.
- Most systems today provide a combination of segmentation and paging