Physical Memory

Daniel Bosk[1]

Department of Information and Communication Systems (ICS),
Mid Sweden University, Sundsvall.

physmem.tex 233 2018-02-18 21:20:05Z jimahl

Mittuniversitetet
MID SWEDEN UNIVERSITY

Overview

1. Hardware
   - Basic hardware
2. Address Binding
   - Binding memory in a process
   - Relocation registers
3. Memory Allocation
   - Contiguous Allocation
   - Non-Contiguous Allocation
   - Paging
   - Segmentation

Literature

This lecture will give an overview of physical memory management. It gives an overview of Chapter 8 “Memory-Management Strategies"in [SGG13a]

Overview

Basic hardware

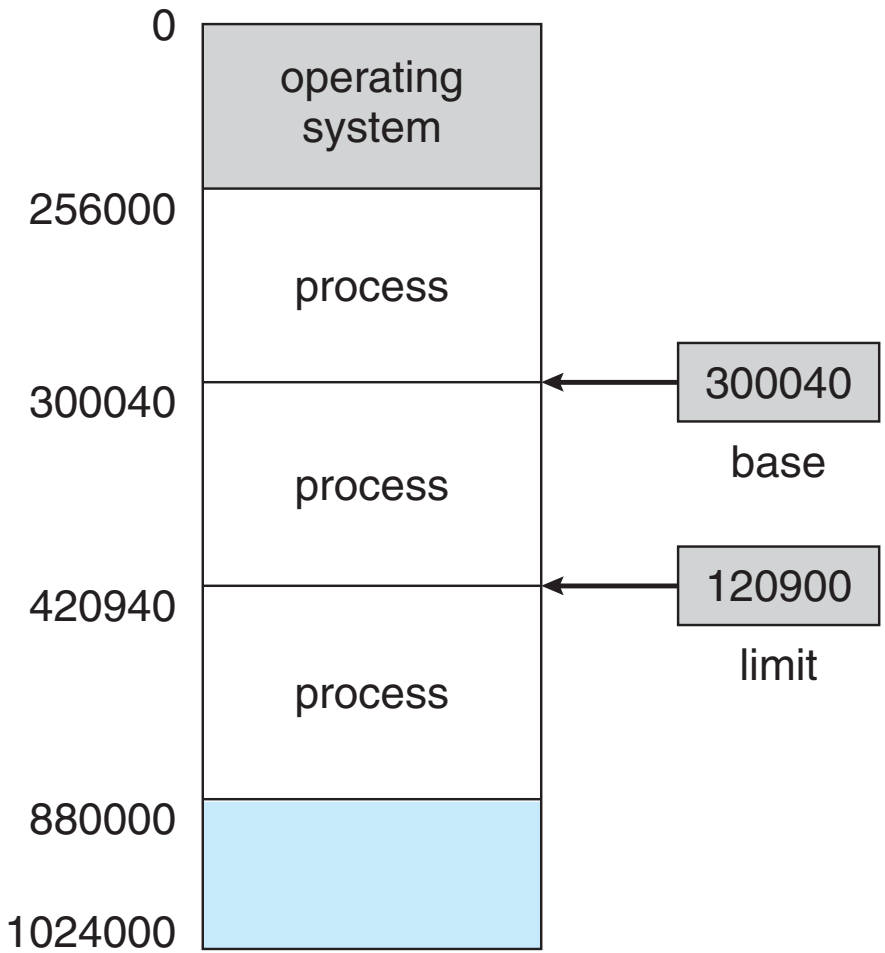

Figure: Memory layout with base and limit for logical address space. Image: [SGG13b].

Basic hardware

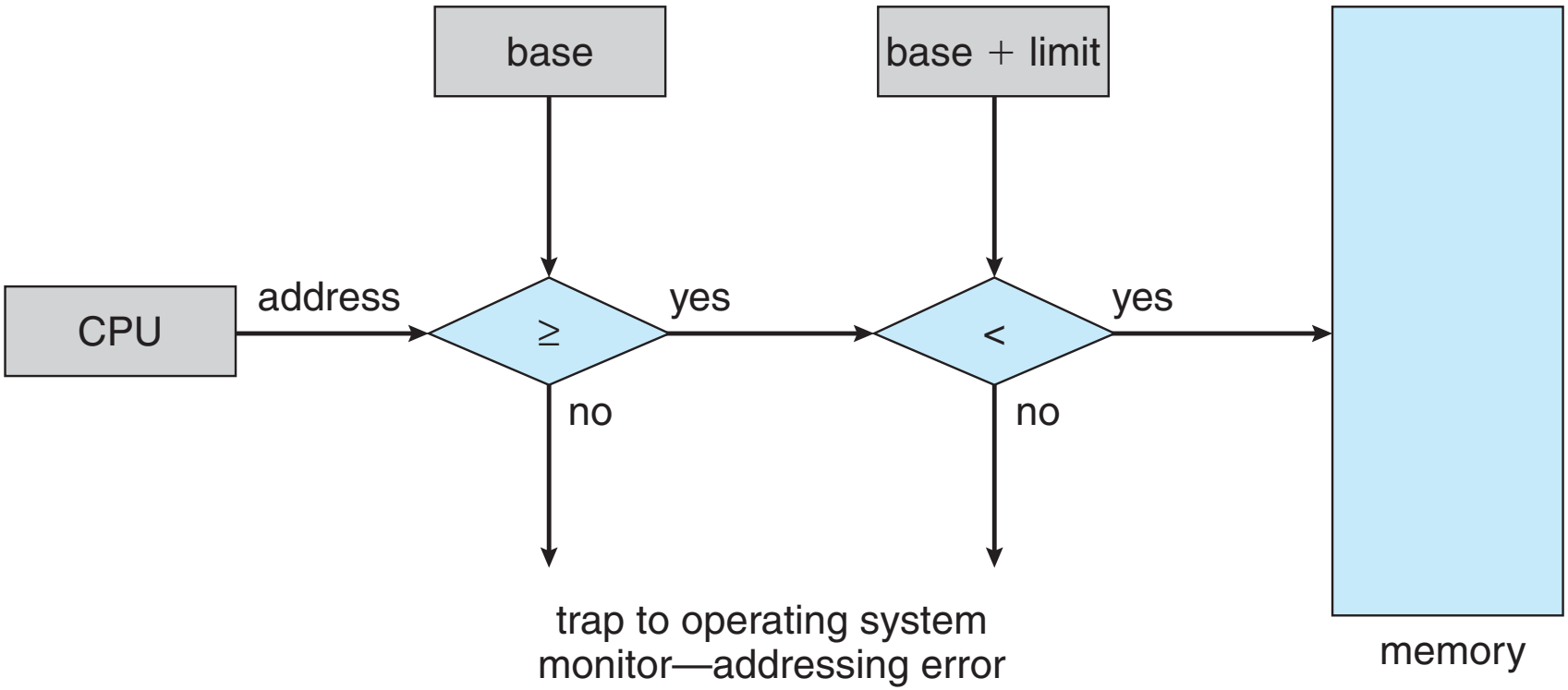

Figure: A schematic of hardware protection for base and limit memory protection. Image: [SGG13b].

Overview

Binding memory in a process

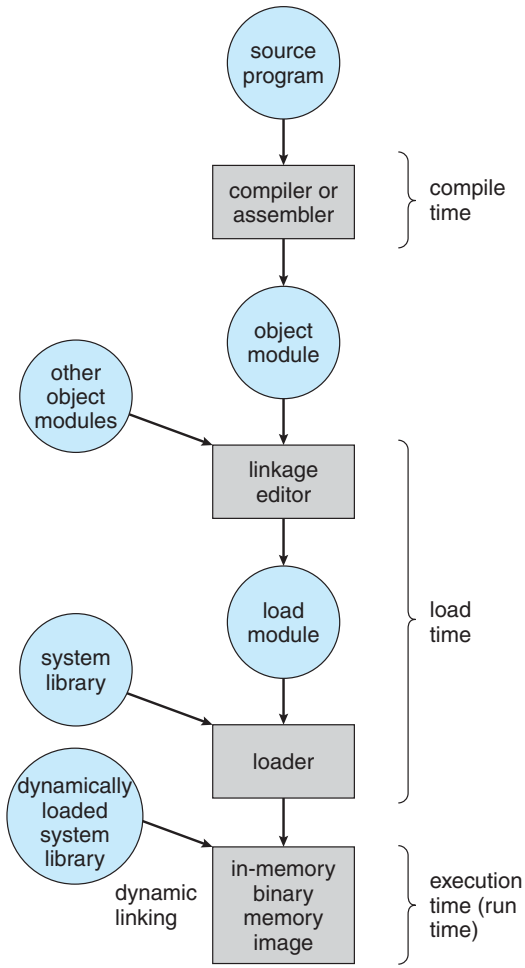

Figure: An overview of all steps to create a program. Image: [SGG13b].

Relocation registers

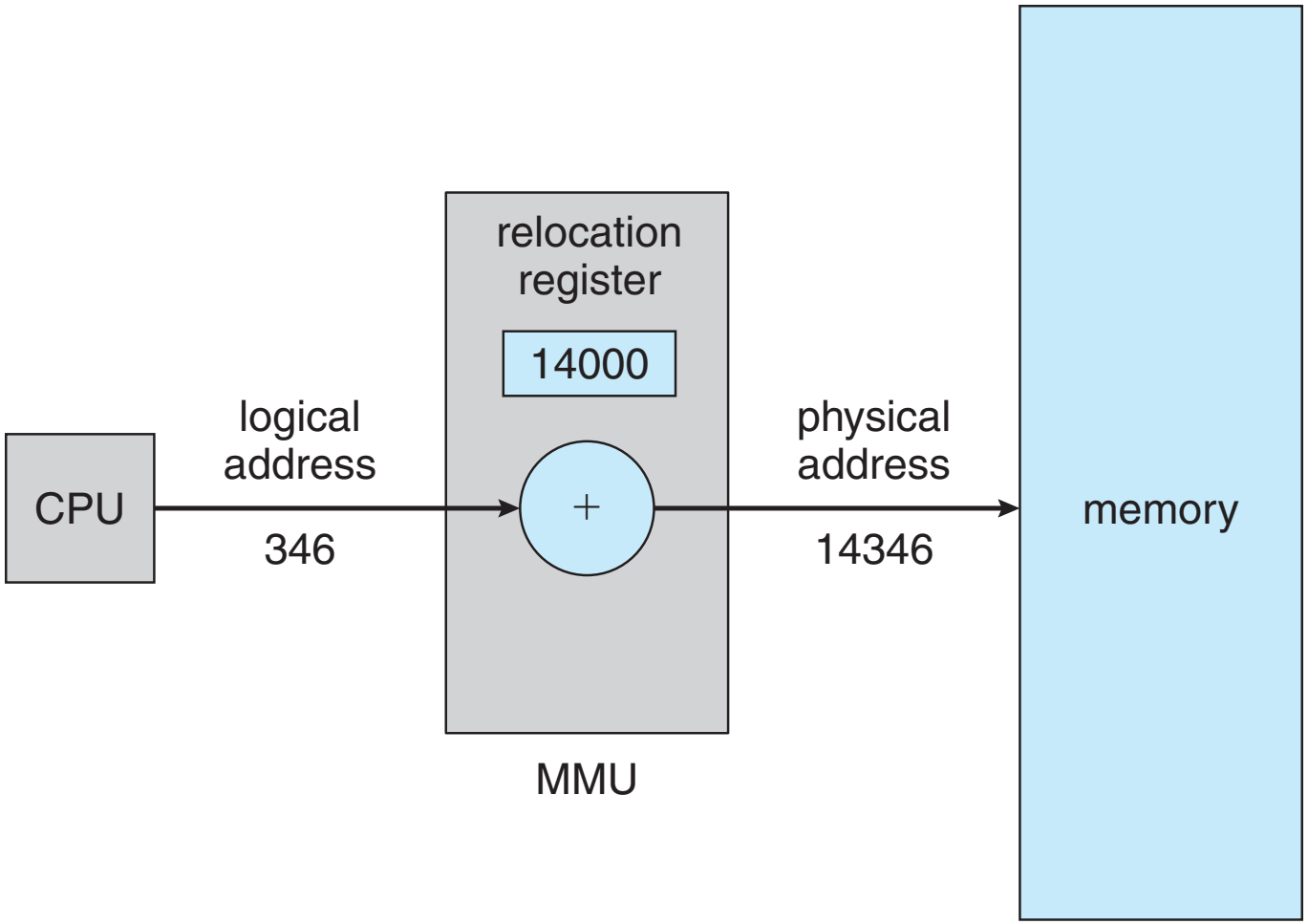

Figure: An illustration of a relocation register. Image: [SGG13b].

Relocation registers

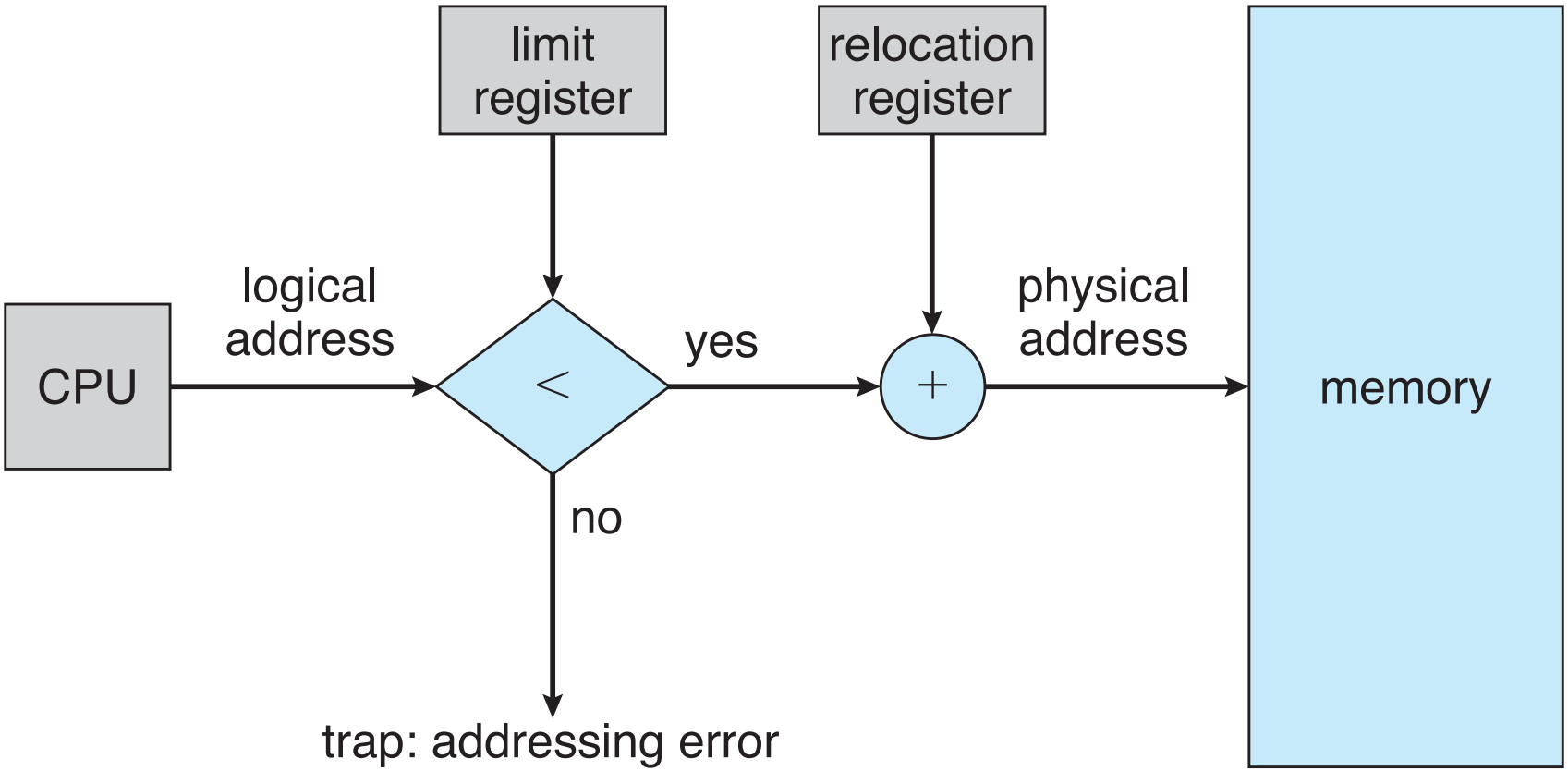

Figure: A relocation register with base and limit properties. Image: [SGG13b].

Overview

Contiguous Allocation

- Processes are placed in contiguous memory spans.
- External fragmentation, and
- the 50-percent rule.

Migitate the problem with:

- Compaction, and
- Swapping.

Contiguous Allocation

- First-fit
- Best-fit
- Worst-fit

Non-Contiguous Allocation

- Paging
- Segmentation

Paging

- Split the logical address into two parts:
  - page number, and
  - page offset (displacement).
- Fix the size of the pages in bits of the address, i.e. the size is a power of two. E.g. page size is the last 10 bits, then each page will be 2^{10} bytes or 1 KiB in size.
- The size of the page must be equal to the size of the physical counterpart, the frame. The frame size, and hence the page size, are determined by the hardware.
- Each page is mapped to a frame, this mapping is kept in a page table.

Paging

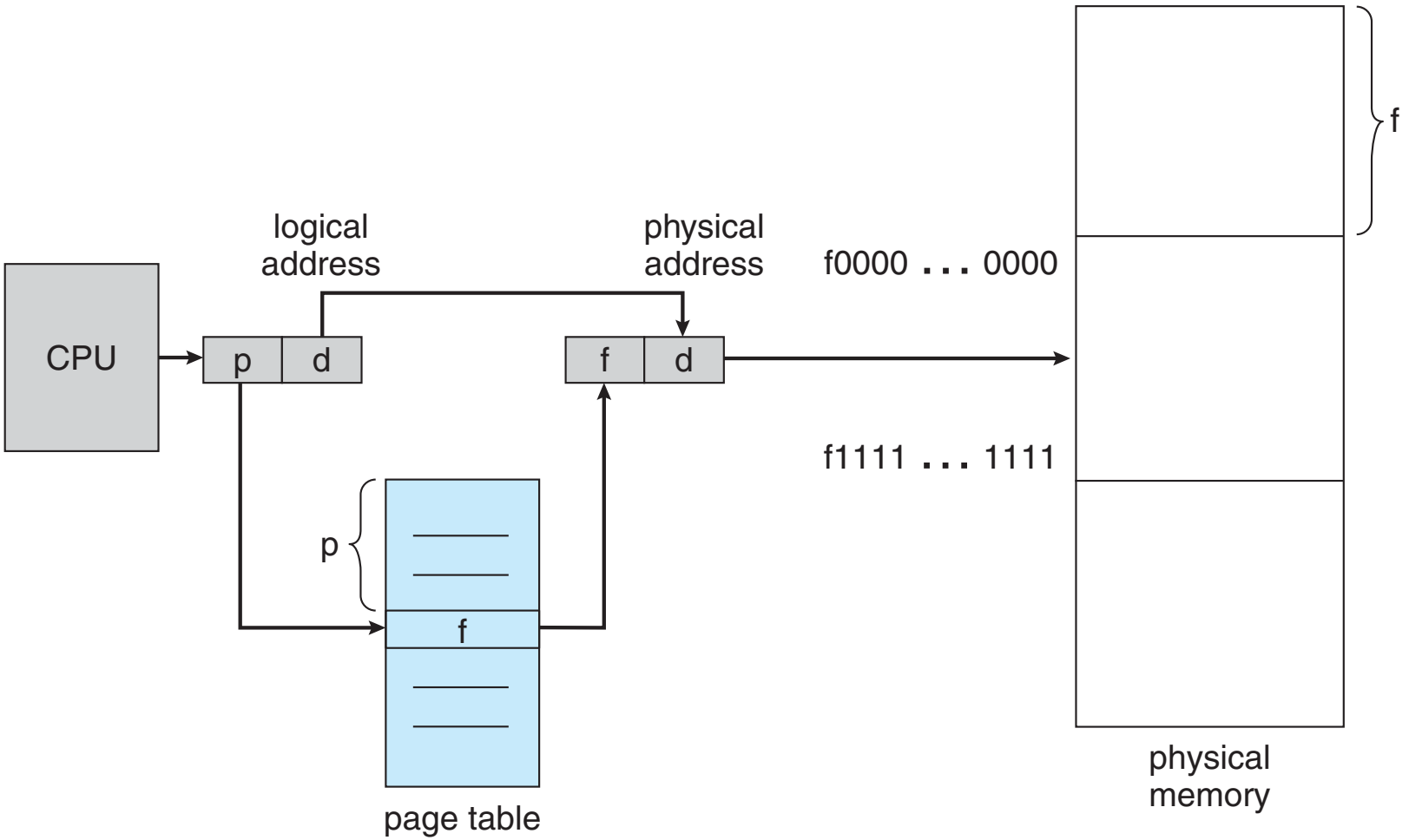

Figure: An overview of paging. Image: [SGG13b].

Paging

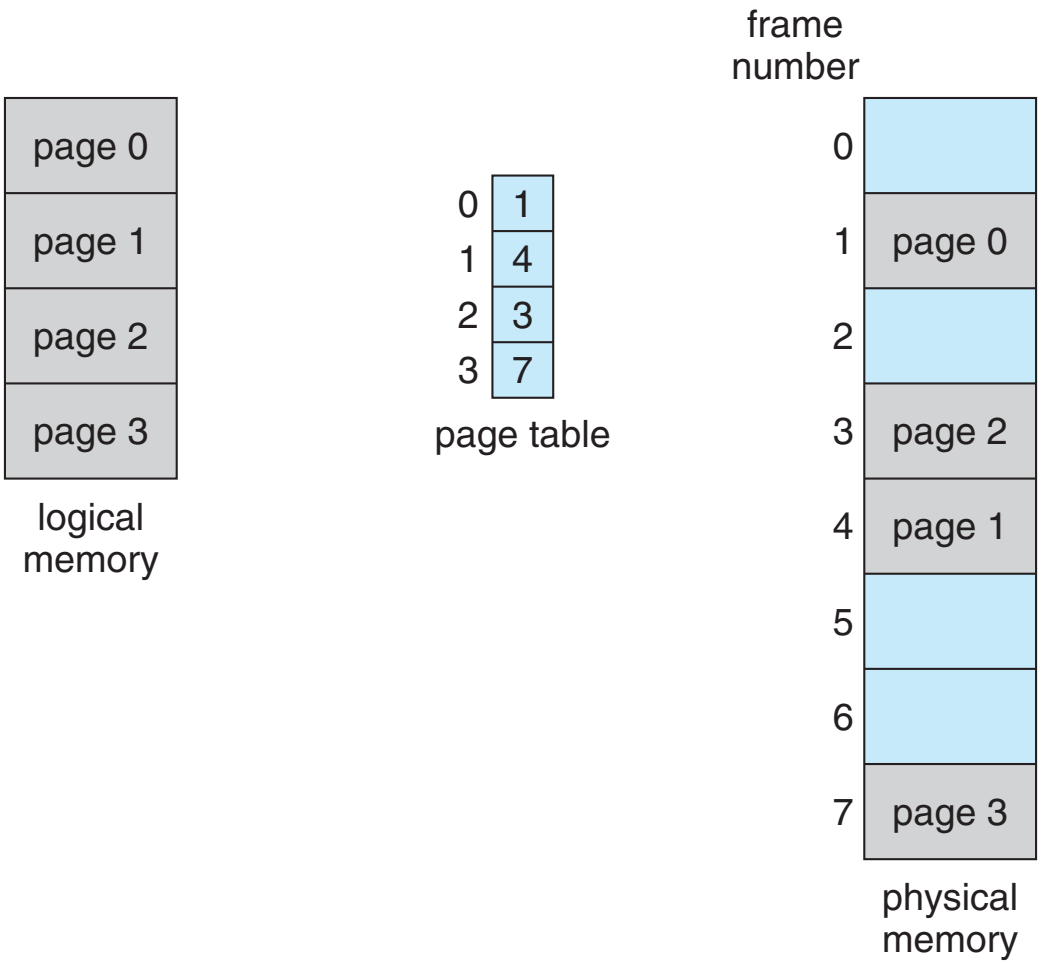

Figure: A pagetable. Image: [SGG13b].

Paging

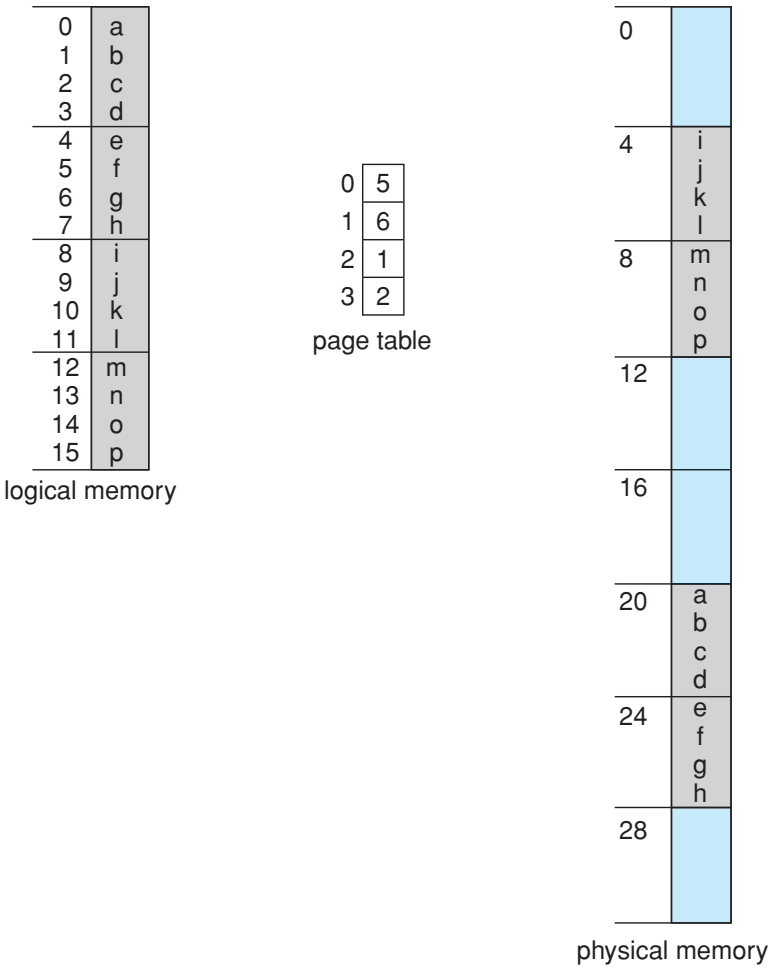

Figure: Allocated memory using paging. Image: [SGG13b].

Paging

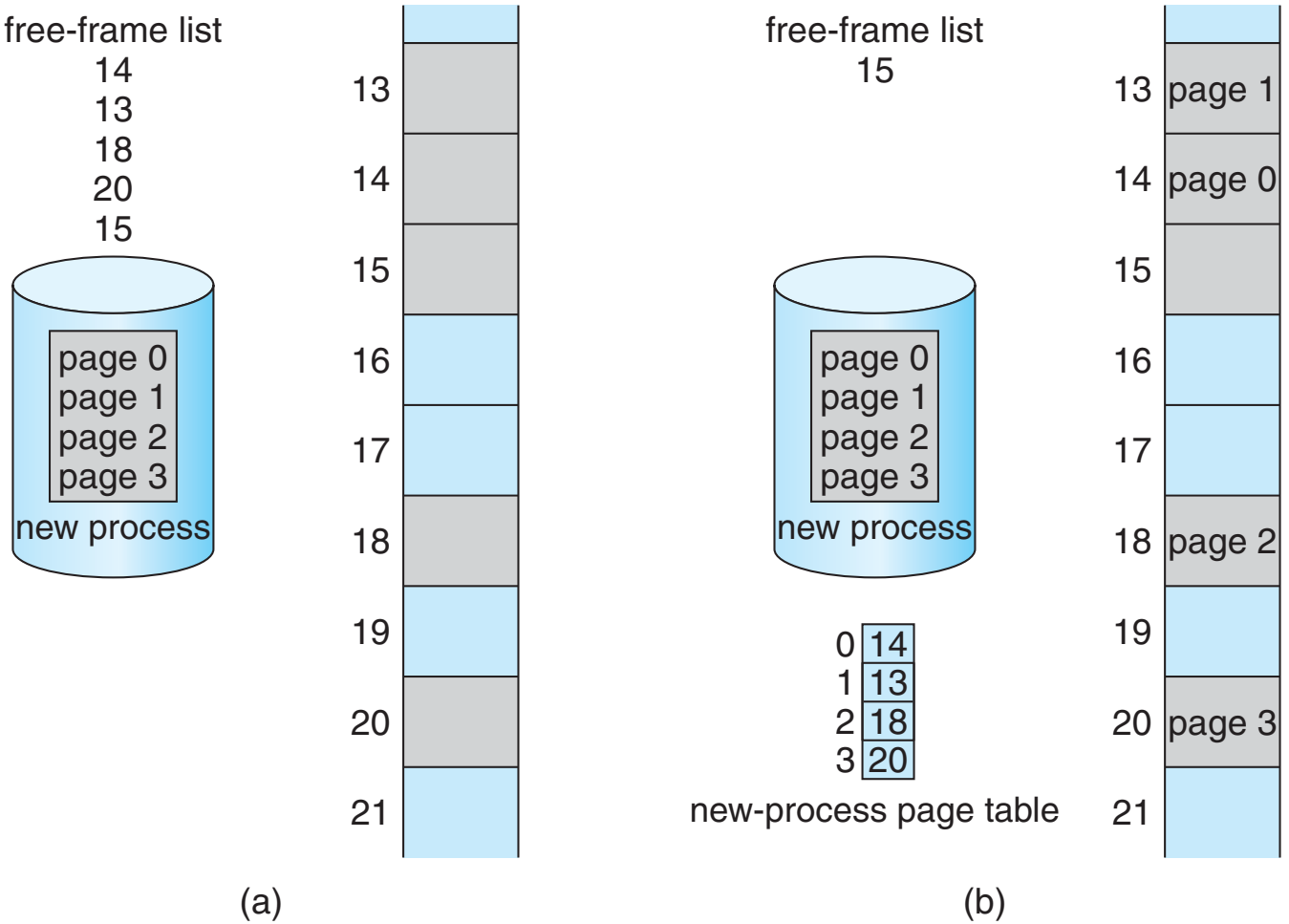

Figure: Before and after allocating a new process to memory. Image: [SGG13b].

Paging

- The page table is usually stored by the OS in the process's PCB.
- The dispatcher loads the process's page table into the hardware, if the page table is sufficiently small.
- Bigger tables must be kept in the memory. A page table base register then points to the page table's position in memory.
- However, this requires a memory lookup, i.e. each memory reference requires two references!
- The solution is to add a translation look-aside buffer (TLB), a very fast hardware cache.

Paging

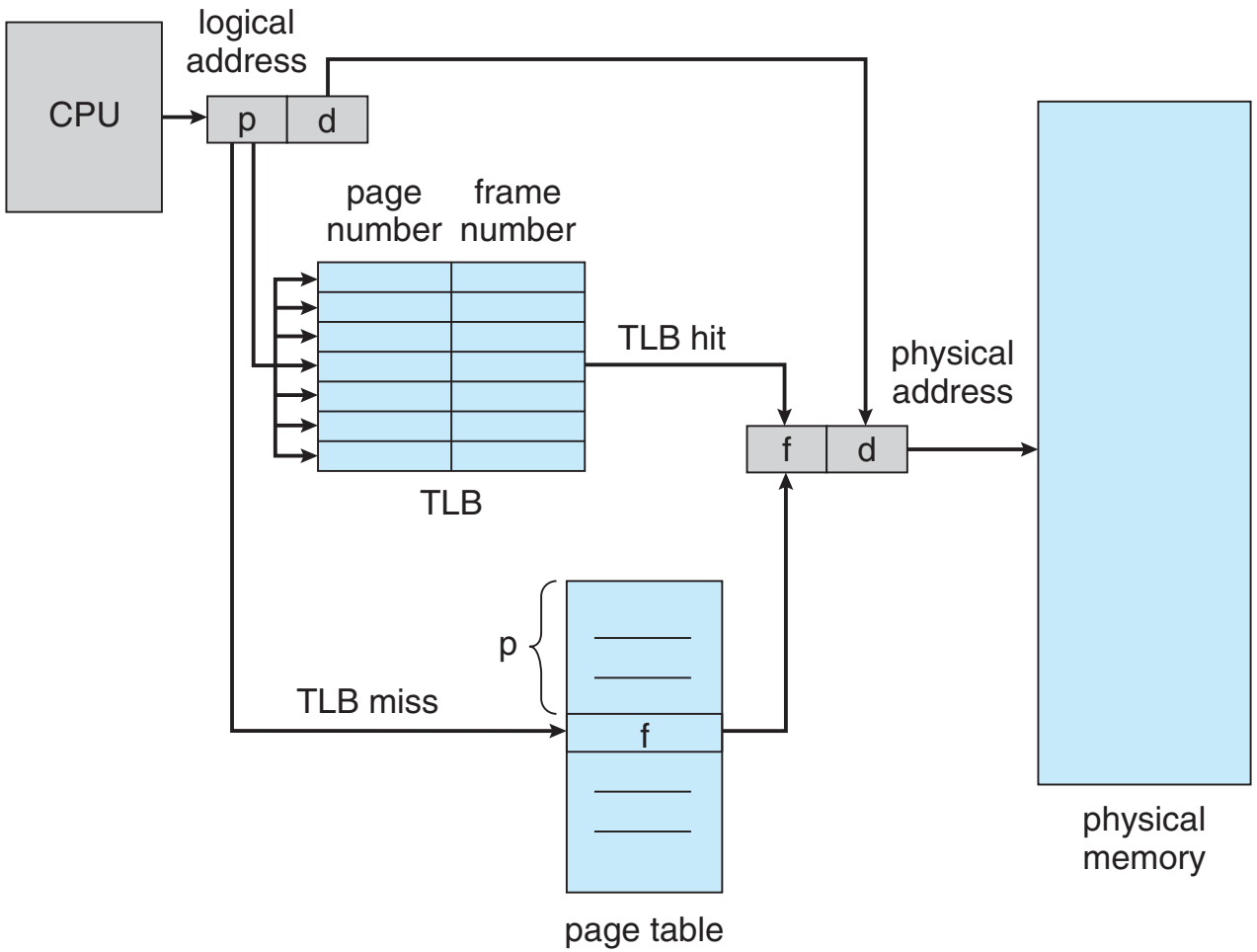

Figure: Paging with a TLB. Image: [SGG13b].

Paging

- Some TLBs have address-space identifications (ASIDs).
- ASIDs allows the TLB to handle pages for different processes simultaneously.
- Hence we don't need to clear it every context switch.

Paging

- One can compute the effective access time for memory.
- Effective memory-access time is the average time it takes for each memory access, i.e. with regards to TLB hits and misses.

Paging

Effective memory-access time

We can compute the effective access time t by

$$t = (1 - p) \times (t_t + t_m) + p \times (t_t + t_m + t_m),$$

where p is the probability of a TLB miss, t_t is the time for a TLB lookup, t_m is the time for a memory access.

Paging

- We must have some protection for the pages of a process.
- What happens when the page table is larger than necessary for a specific process?
- Consider a system with a 14-bit address space, a process with the logical address range 0 to 10468, and a page size of 2 KiB ...

Paging

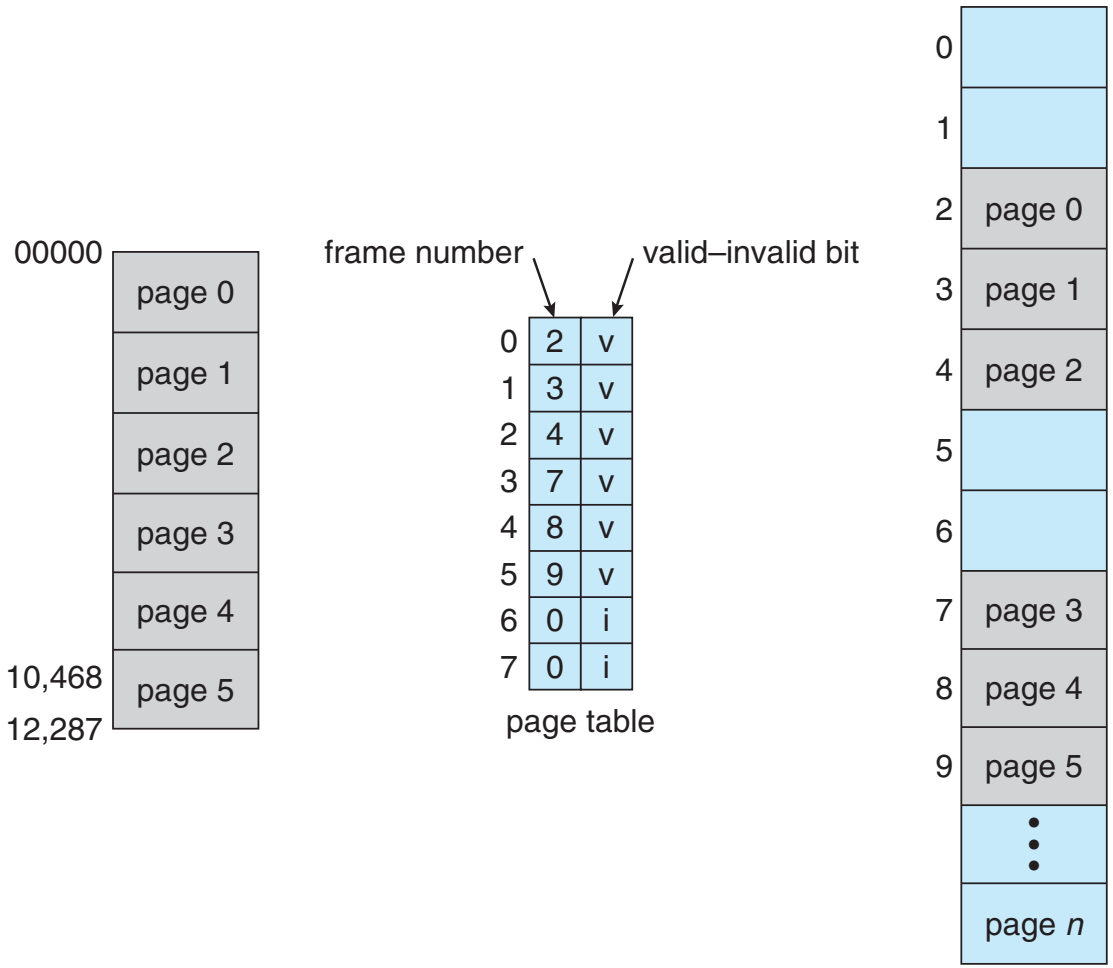

Figure: Paging with valid-bits. Image: [SGG13b].

Paging

- We can also add additional bits like this.
- We can have bits for specifying whether reading, writing or execution is allowed on these pages.
- This way we can implement execution prevention for certain memory regions, e.g. where the web browser stores data downloaded from the Web.
- This way several processes can share read-only pages, hence requiring less memory.

Paging

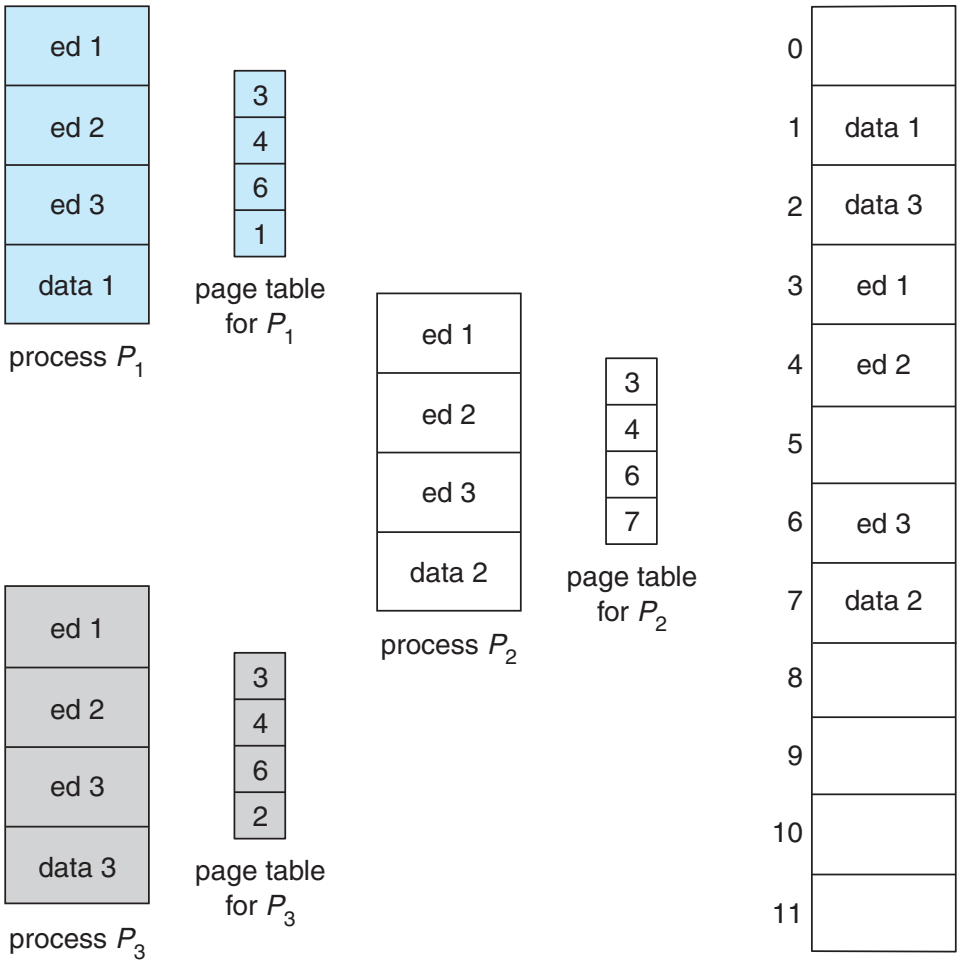

Figure: An example of shared pages. Image: [SGG13b].

Paging

- What happens when we get huge logical address spaces, e.g. 32 or 64-bit addresses?
- With a page size of 4 KiB (i.e. 12 bits) the worst case scenario in a 32-bit system is a page table of 4 MiB in size – for each process!
- If we need to allocate this contiguously in memory we're back to the original problem we wanted to solve.
- Well, we have a solution for that: paging!
- We page the page table ...

Paging

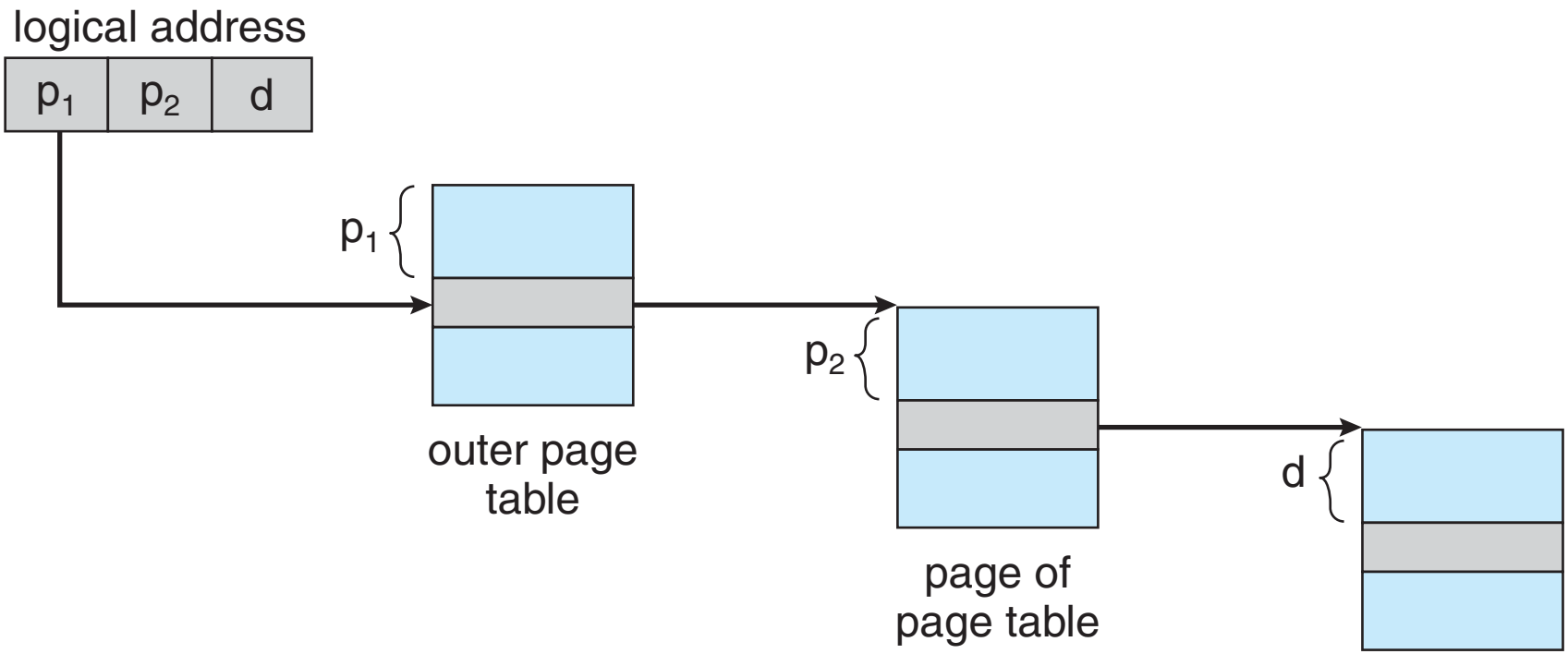

Figure: Hierarchical paging. Image: [SGG13b].

Paging

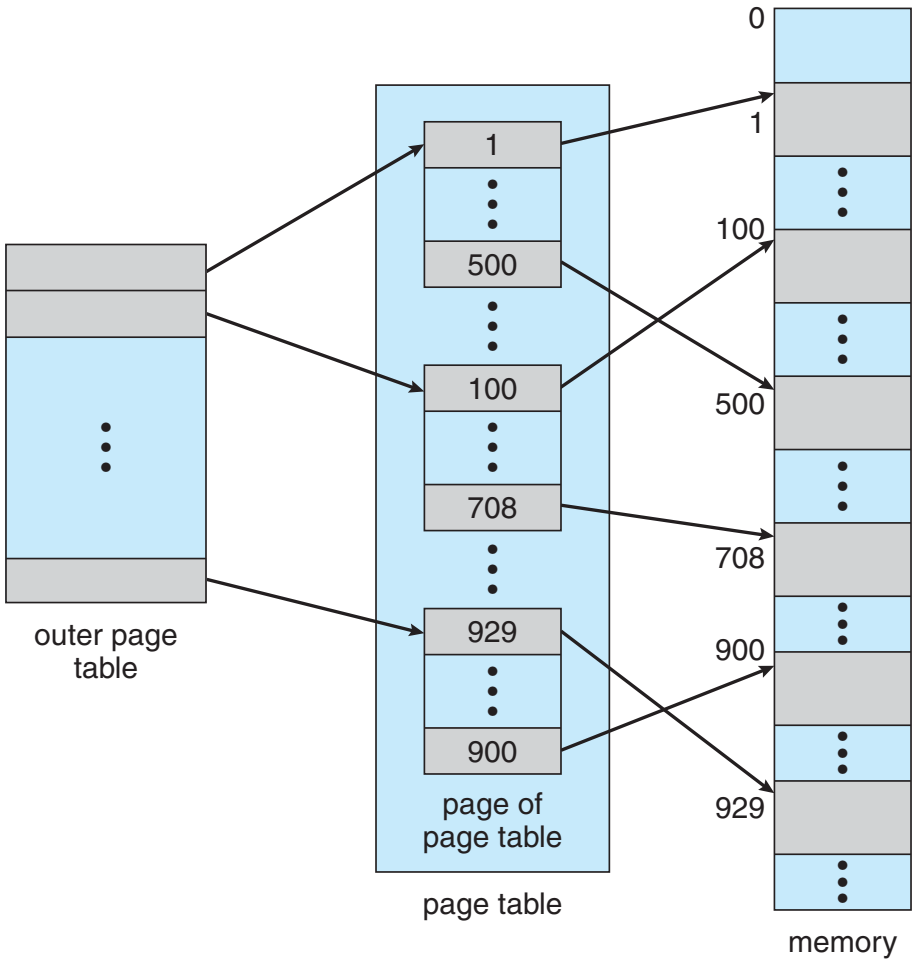

Figure: Hierarchical paging, overview. Image: [SGG13b].

Paging

- Now each part of the page table fits in a page.
- Hence we don't need contiguous allocation anymore.
- However, this doesn't work for spaces even as small as 64 bits.

Paging

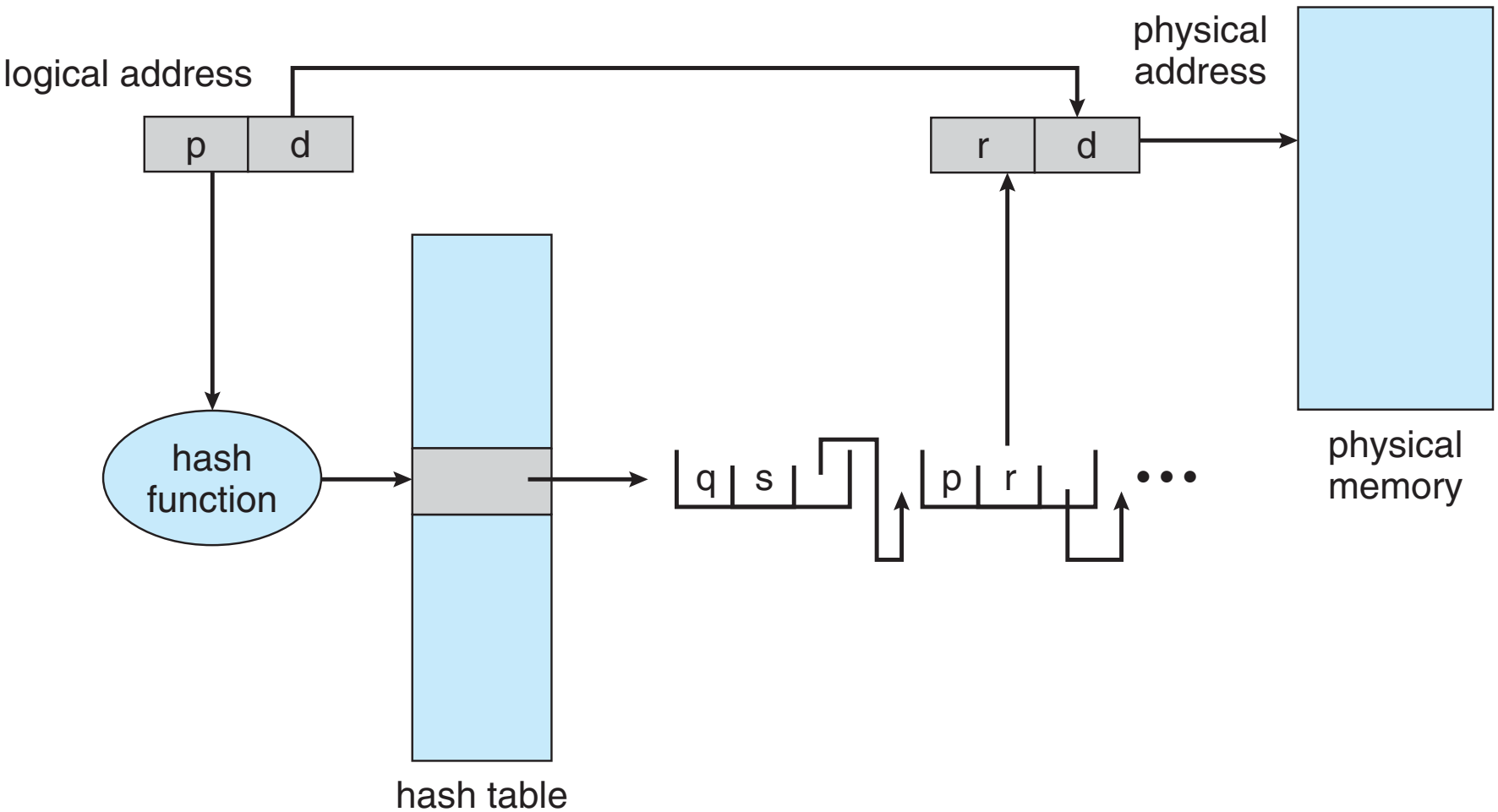

Figure: Paging using hashing. Image: [SGG13b].

Paging

- Another way of saving space is to use an inverted page table.
- Here we have one entry per frame, not per page.
- Then we map which process has allocated it and to which page.

Paging

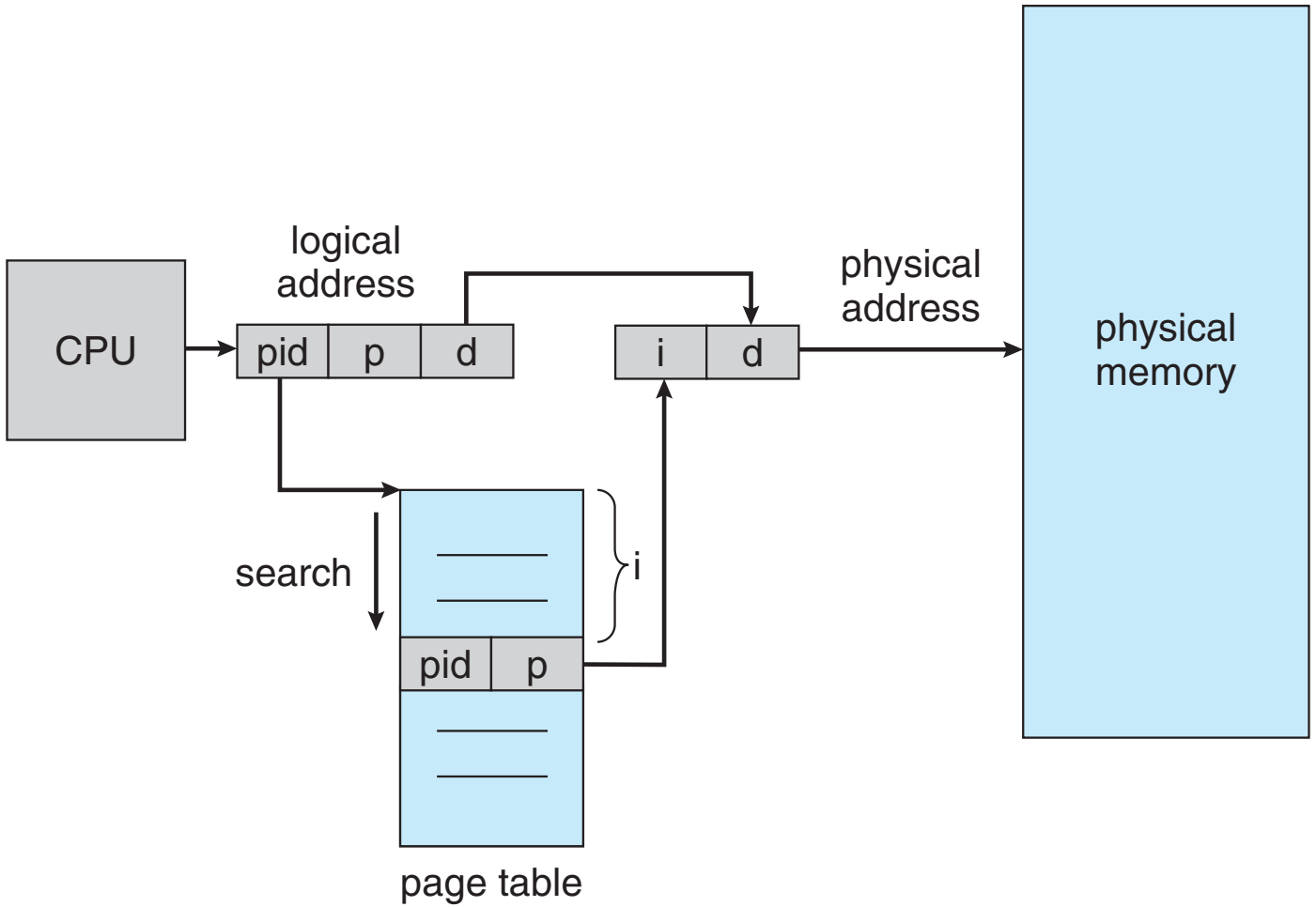

Figure: Inverted paging. Image: [SGG13b].

Segmentation

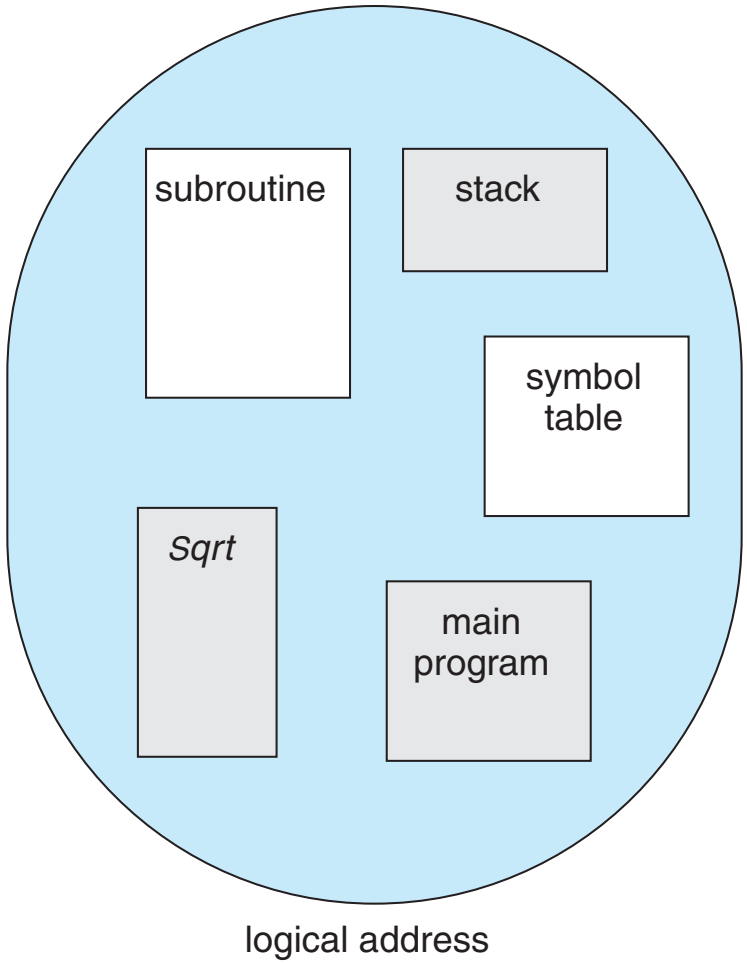

Figure: Segments. Image: [SGG13b].

Segmentation

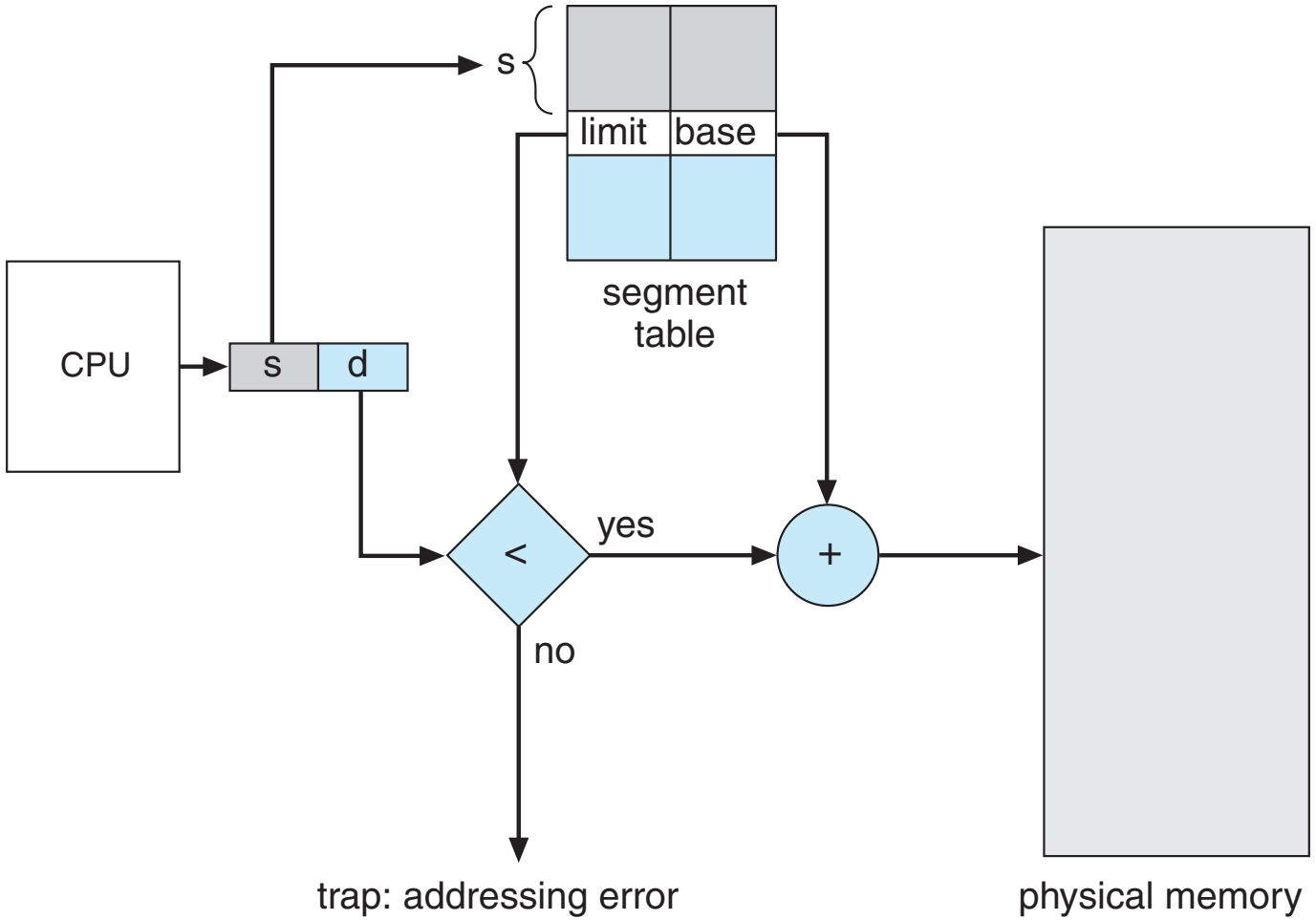

Figure: Segmentation hardware. Image: [SGG13b].

Segmentation

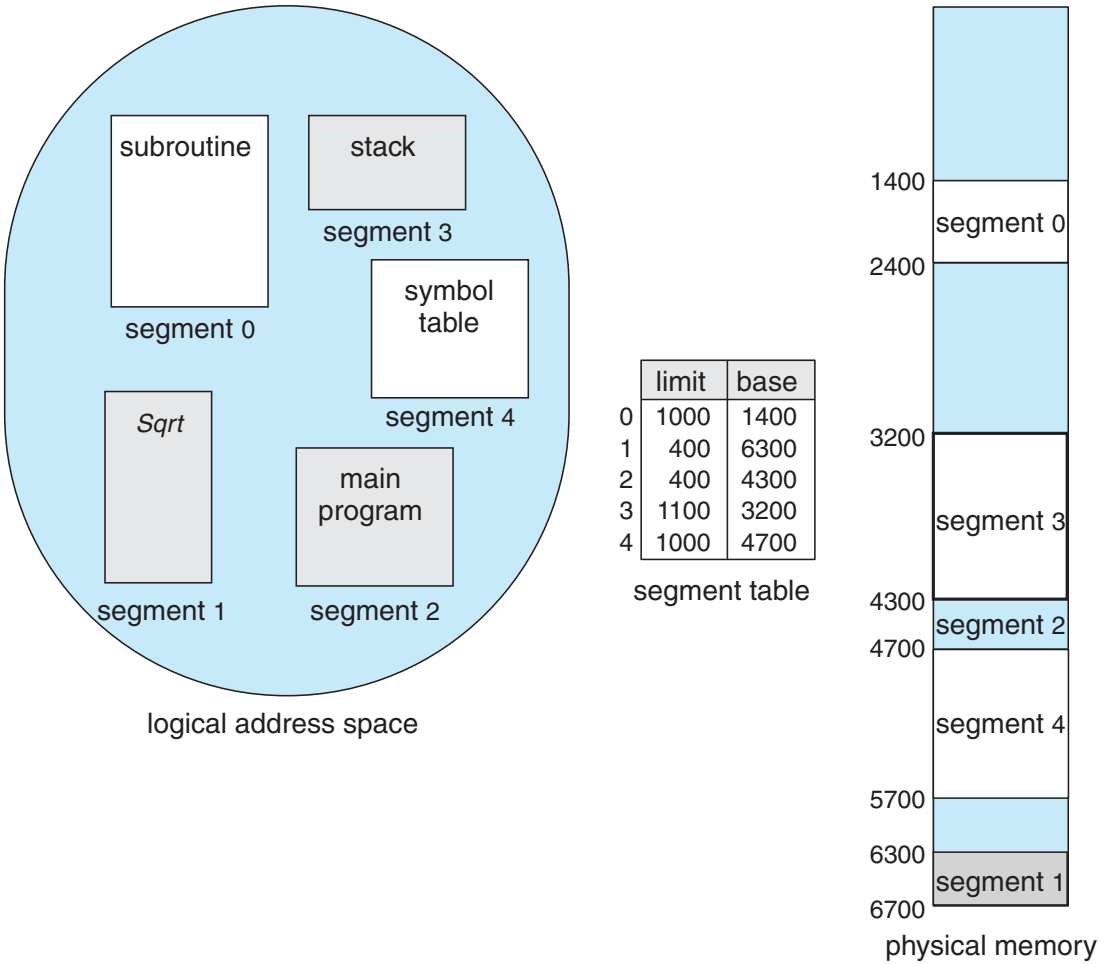

Figure: Segment mapping. Image: [SGG13b].

Referenser I

Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne. *Operating System Concepts*. 9th ed. International Student Version. Hoboken, N.J.: John Wiley & Sons Inc, 2013.

Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne. *Operating System Concepts*. 9th ed. Hoboken, N.J.: John Wiley & Sons Inc, 2013.