Secure Operating Systems

Nikos Tziritas

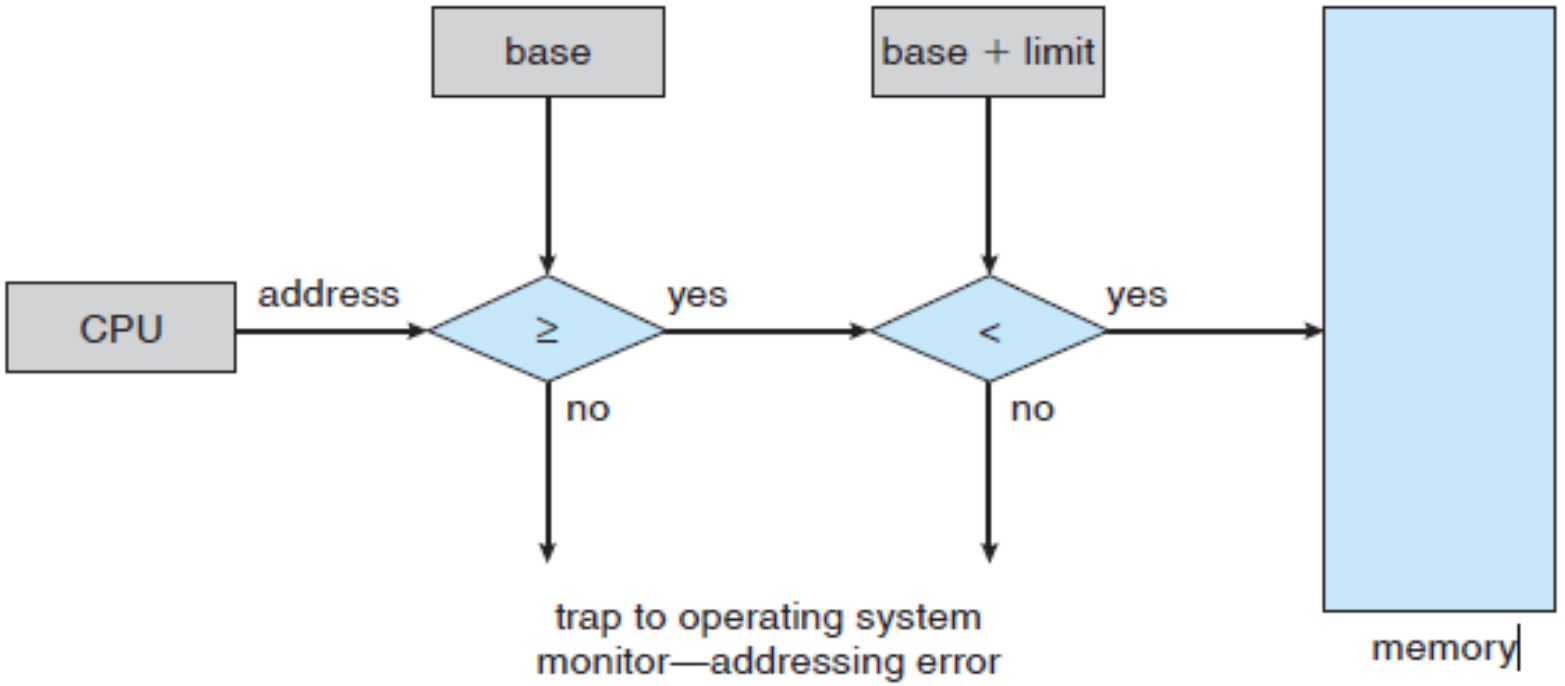
Memory Management

- CPU accesses data from main memory via a transaction on the memory bus
- Completing a memory access may take many cycles of the CPU clock. In such cases, the processor normally needs to stall.
- A remedy to the aforementioned problem (CPU stall) is to add fast memory between the CPU and main memory. This memory is called cache.

Legal addresses

- The operating system must make sure that each process has a separate memory space.
- Therefore there is a range of legal addresses that each process may access.
- The operating system provides the aforementioned protection of memory addresses through :
  - base register (smallest legal address)
  - limit register (the size of the range)

Hardware address protection



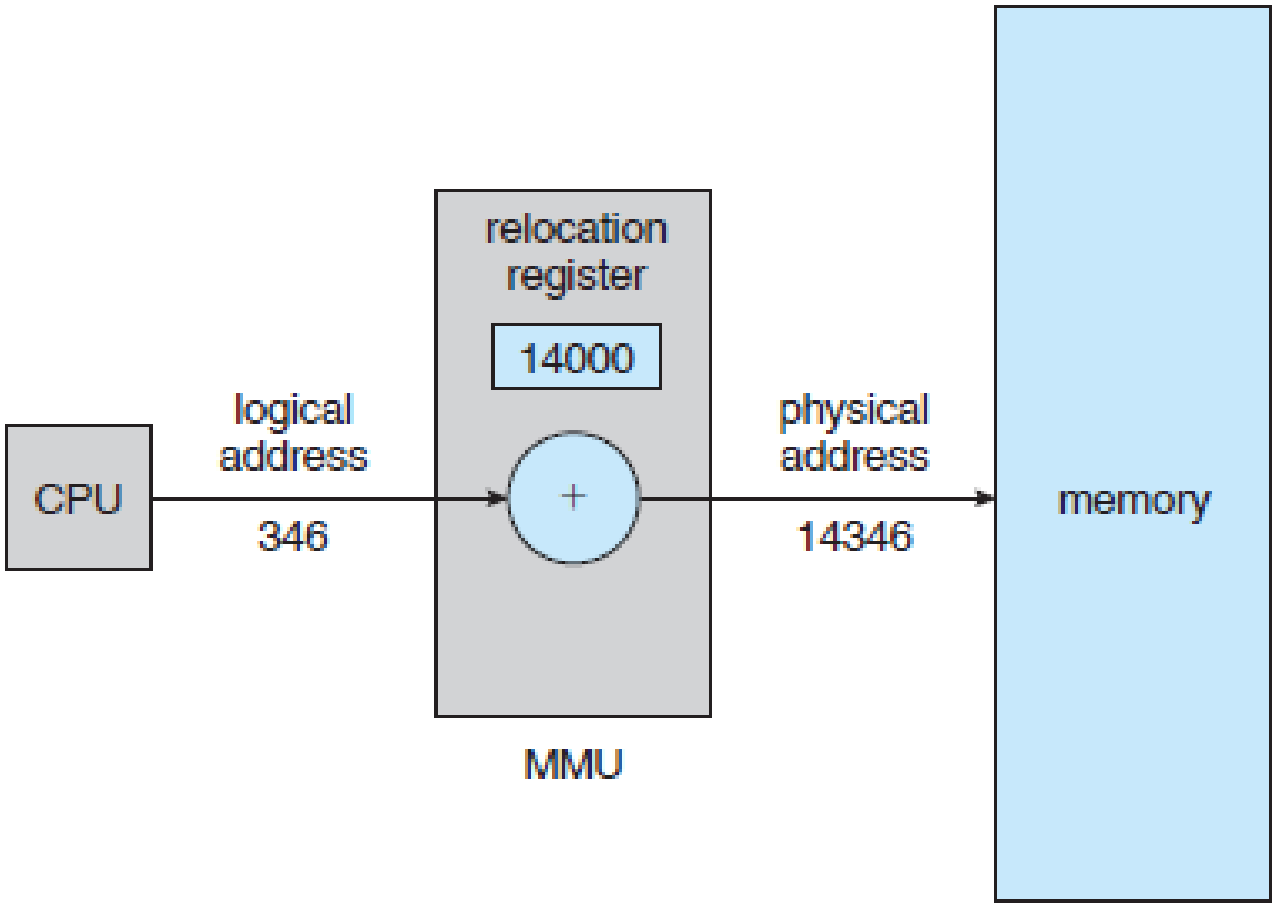
Virtual vs Physical Address Space

- An address generated by the CPU is referred to as a **virtual address**
- An address seen by the memory unit is referred to as a **physical address**
- The set of all virtual addresses generated by a program is a **virtual address space**. The set of all physical addresses corresponding to these virtual addresses is a physical address space

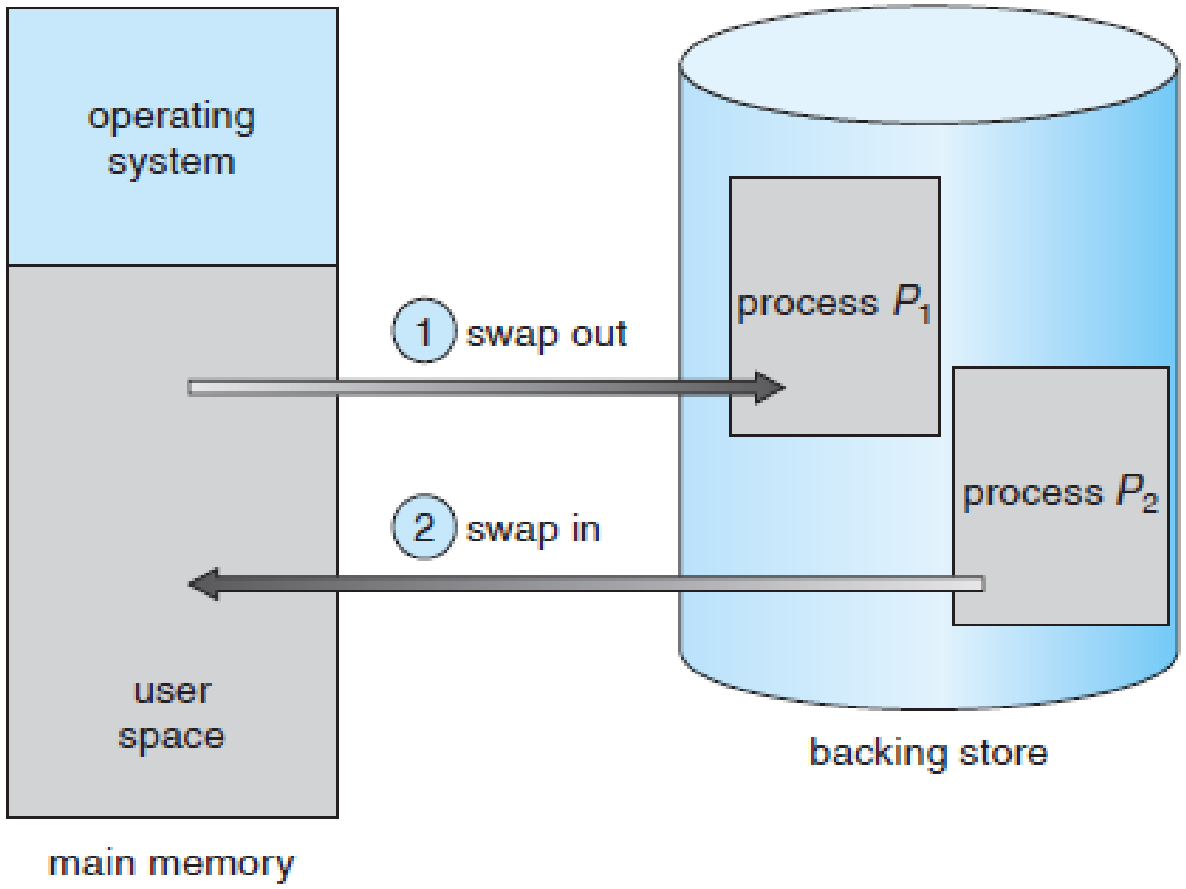
Runtime Memory Mapping

- The runtime mapping from virtual to physical addresses is done by a hardware device called the **Memory-Management Unit** (MMU)
- We illustrate this mapping with a simple MMU scheme that is a generalization of the base-register scheme. The base register in this scheme is called relocation register

Dynamic Relocation



Swapping


operating system
user space
main memory
1 swap out
2 swap in
process P_1
process P_2
backing store

Paging

- Paging is a memory-management scheme that permits the physical address of a process to be noncontiguous.
- Paging avoids external fragmentation (memory space broken into little pieces)

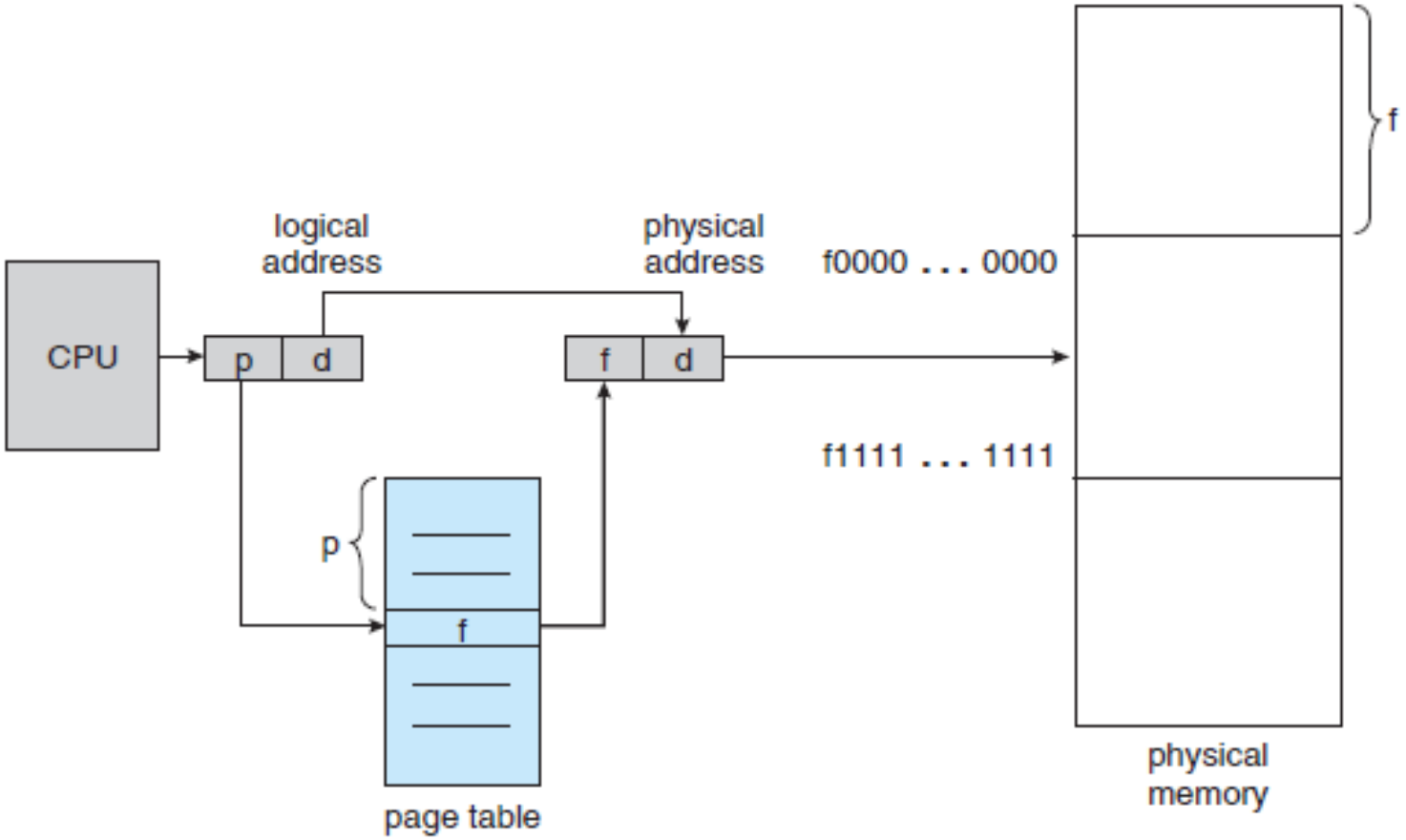
Paging – Basic Method (1)

- Breaking physical memory into fixed-sized blocks called **frames**.
- Breaking virtual memory into blocks of the same size called **pages**.
- The backing store is divided into fixed-sized blocks that are of the same size as the memory frames

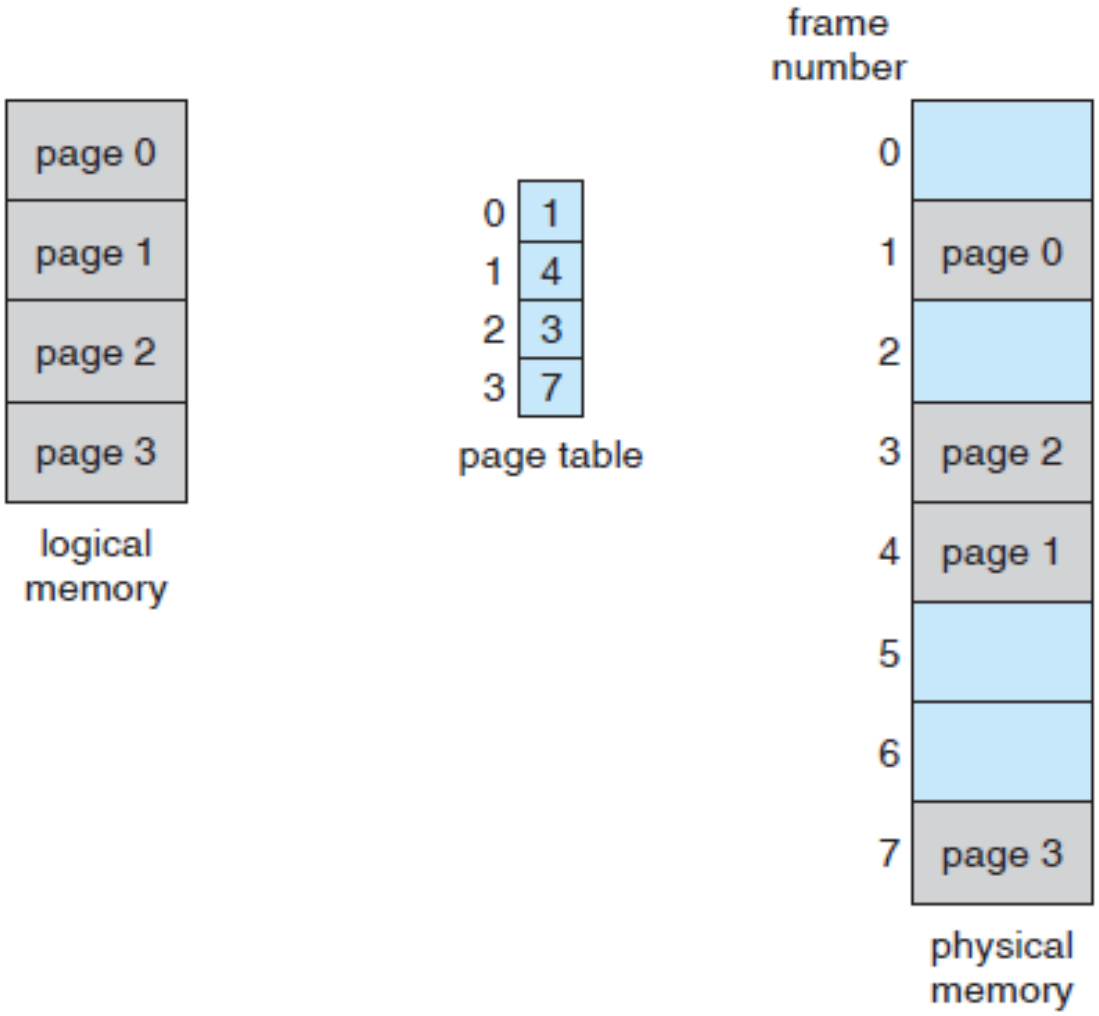
Paging – Basic Method (2)

- Every address generated by CPU is divided into two parts:
  - Page number (p)
  - Page offset (d)
- The page number is used as an index into a page table. The page table contains the base address of each page in physical memory. This base address is combined with the page offset to define the physical memory.

Paging Hardware

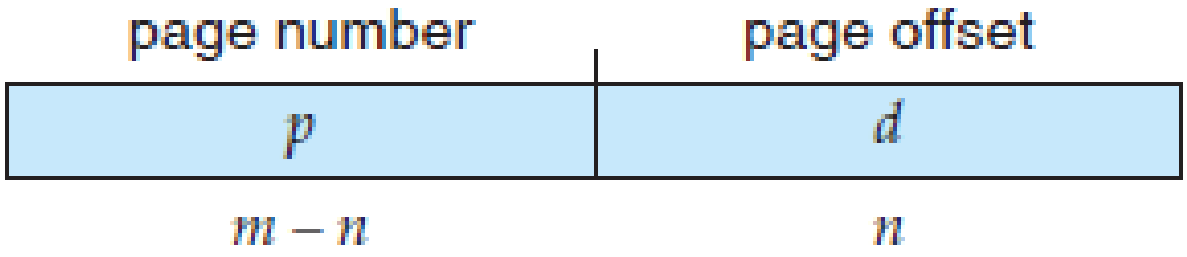


Paging Model (page table per process)



Page number and offset

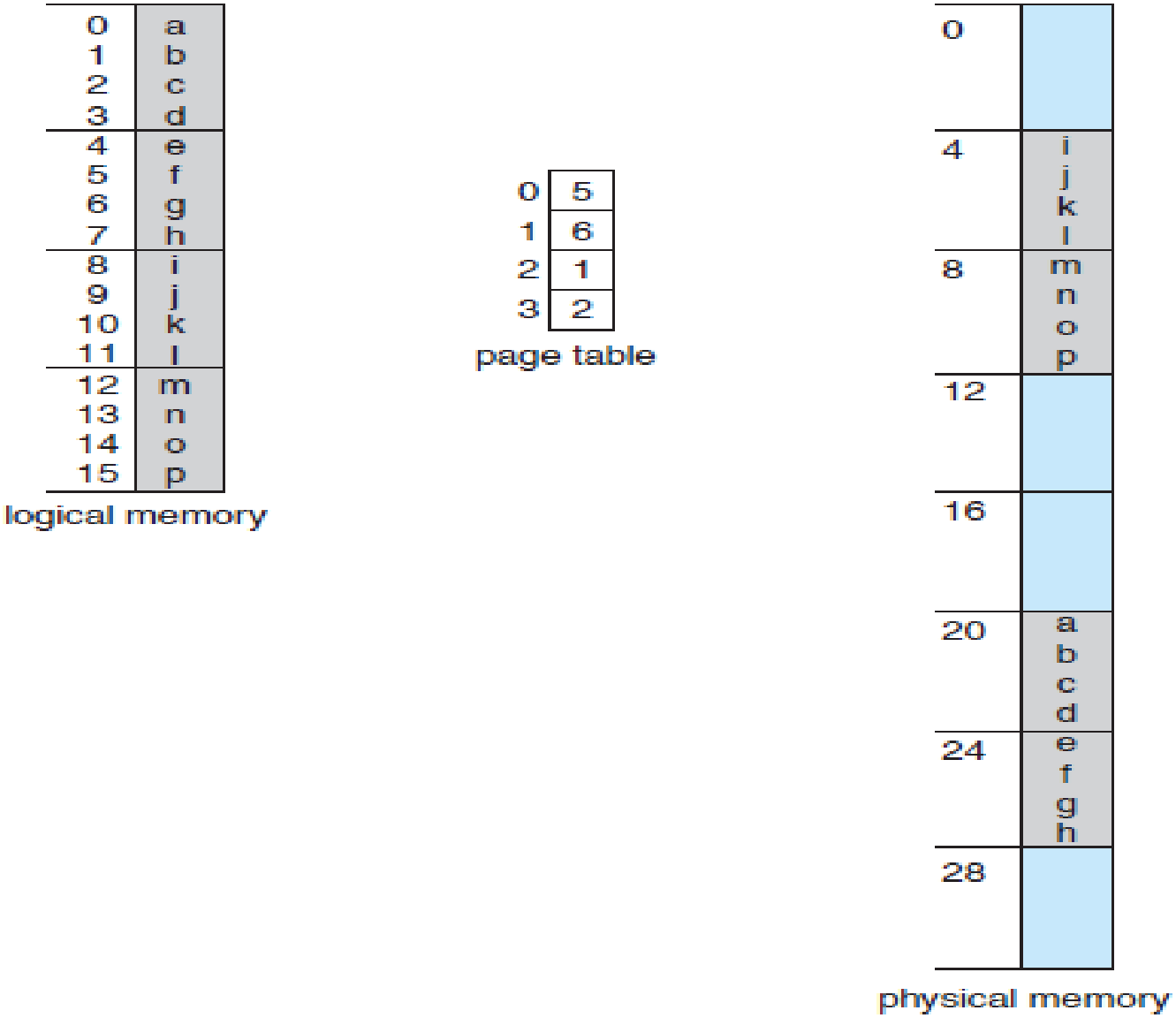
The size of the virtual address space is 2^m and a page size is 2^n addressing units (words). The high-order *m*-*n* bits of logical address designate the page number and the *n* low-order bits designate the page offset.



Example

Consider n=2 and m=4. Our physical memory is 32 bytes (8 pages).

Example (cont.)

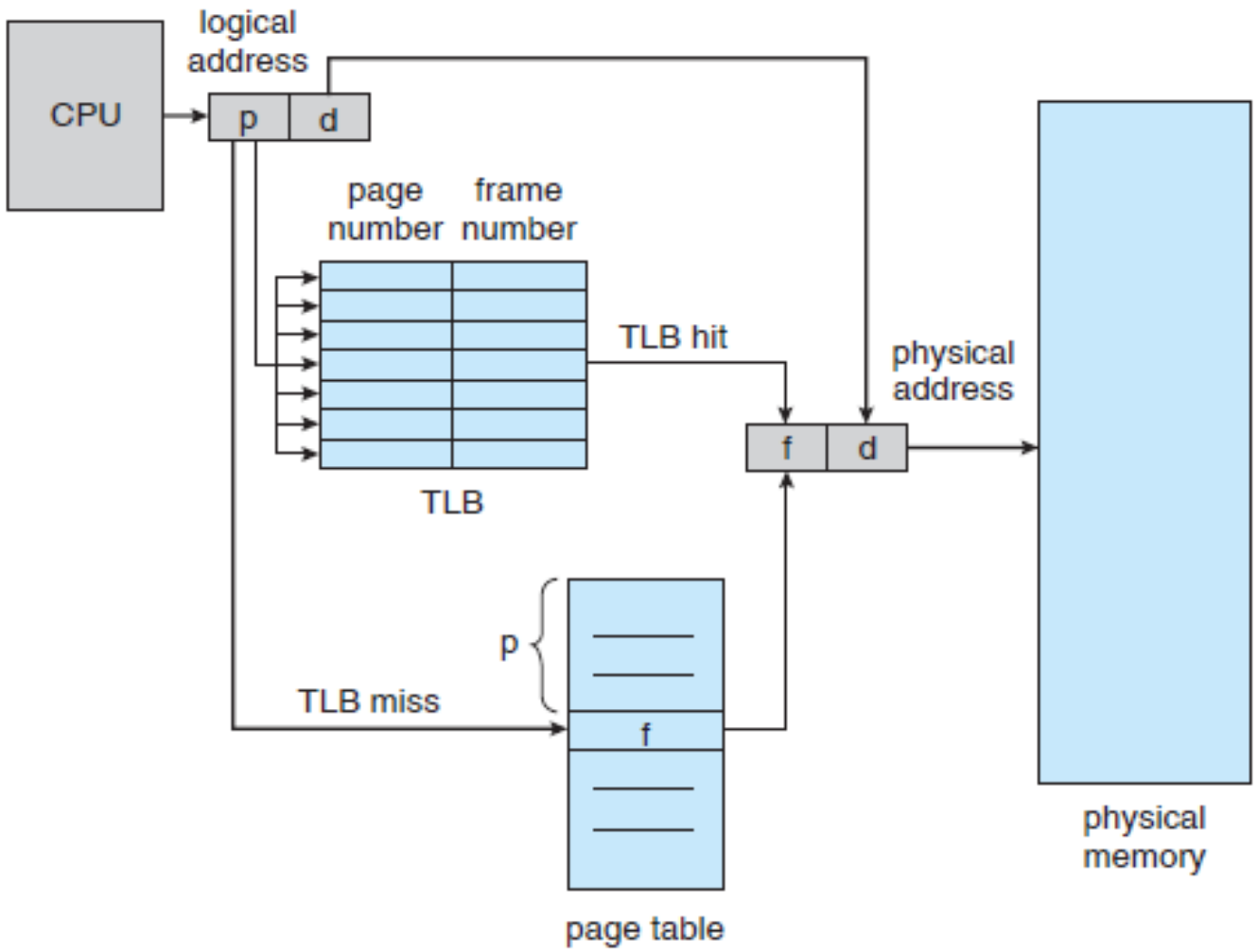


Example (cont.)

According to the page table shown in the previous slide we have that:

- Virtual address 0 is represented as (0000) page 0, offset 0. The page table maps page 0 to frame 5, which means that address 0 maps to physical address 20 [=5*4+0].
- Virtual address 4 is represented (0100) as page 1, offset 0. The page table maps page 1 to frame 6, which means that address 4 maps to physical address 24[=6*4+0]

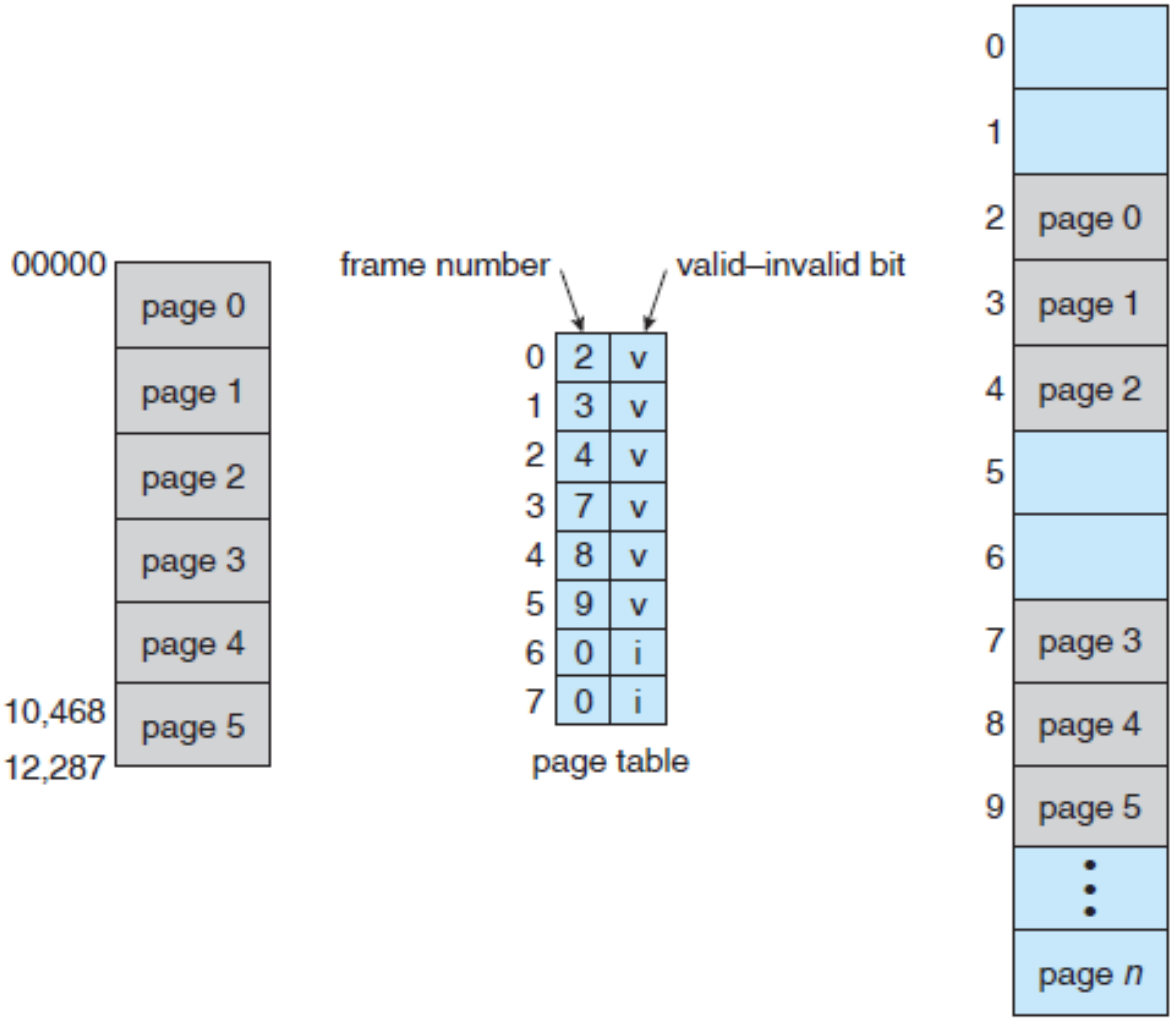
Translation Look-aside Buffer (TLB)



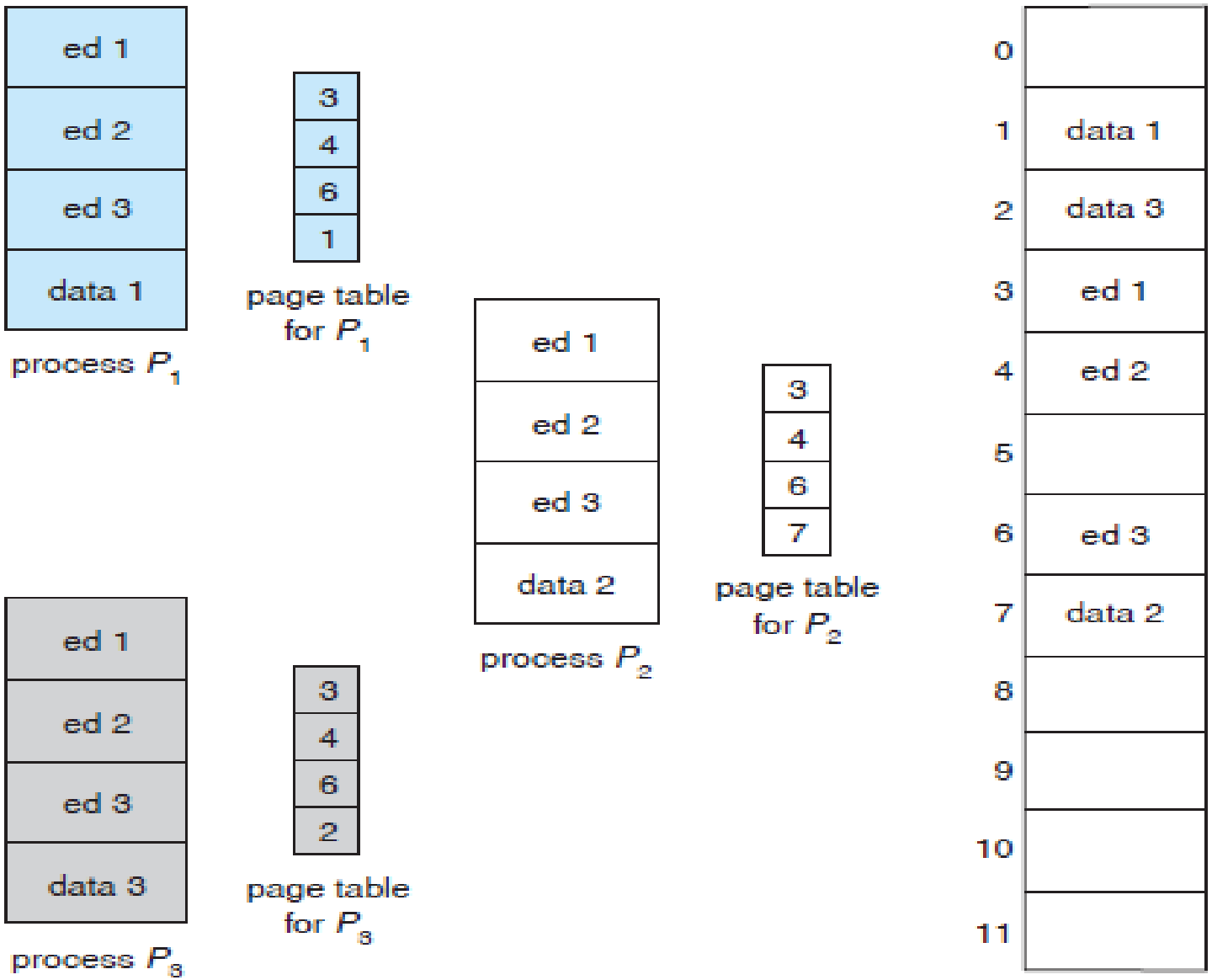
TLB Example

- The percentage of times that a particular page number is found in TLB is called **hit-ratio**.
- Consider that the hit ratio is 80%, the TLB search costs 20 ns, while the main memory search costs 100 ns. To access the desired byte in memory costs:
  - 120 ns in case of success
  - 220 ns in case of failure
- Therefore, the effective memory-access time is 0.80*120+0.20*220=140 ns

Memory protection



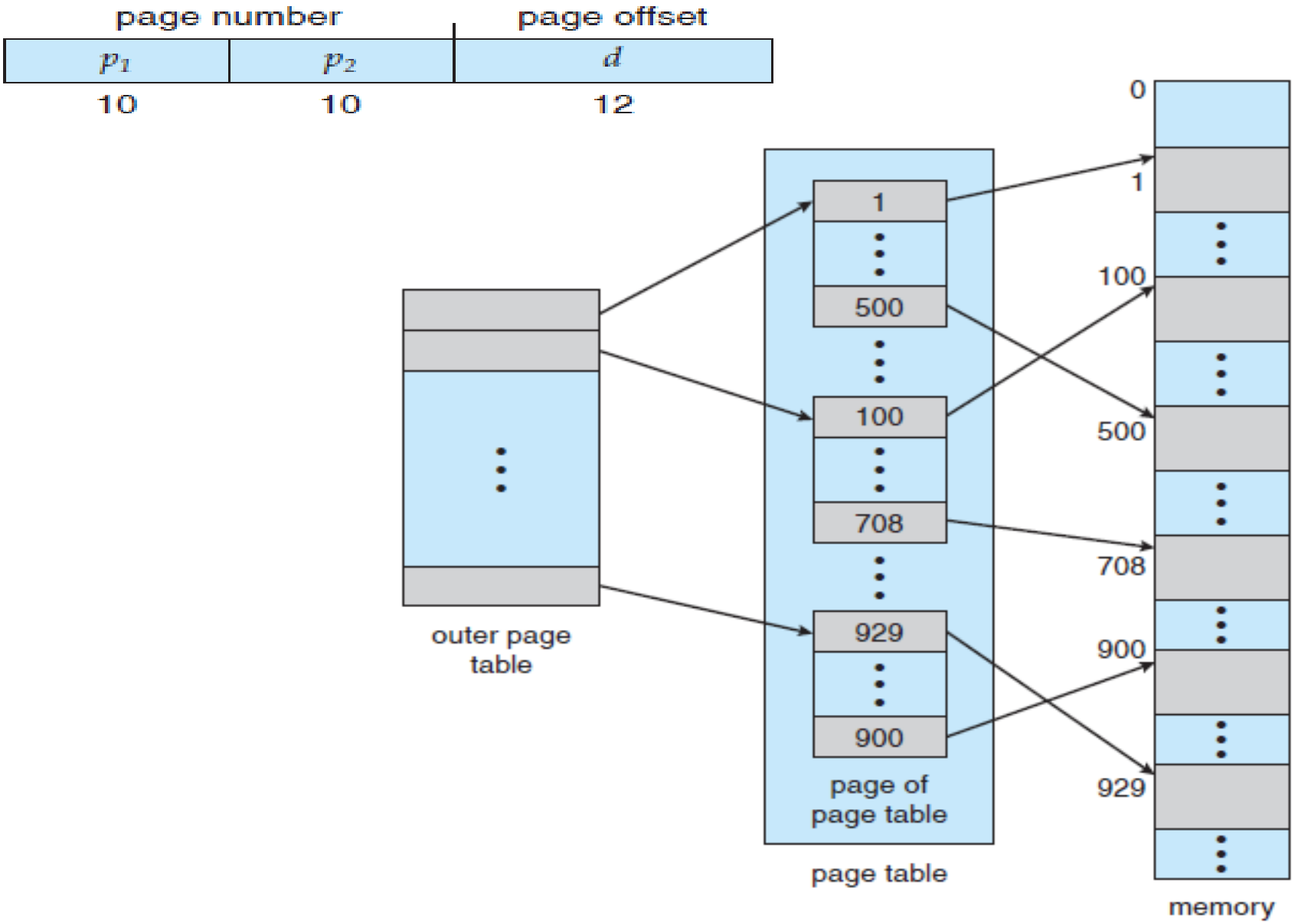
Shared pages



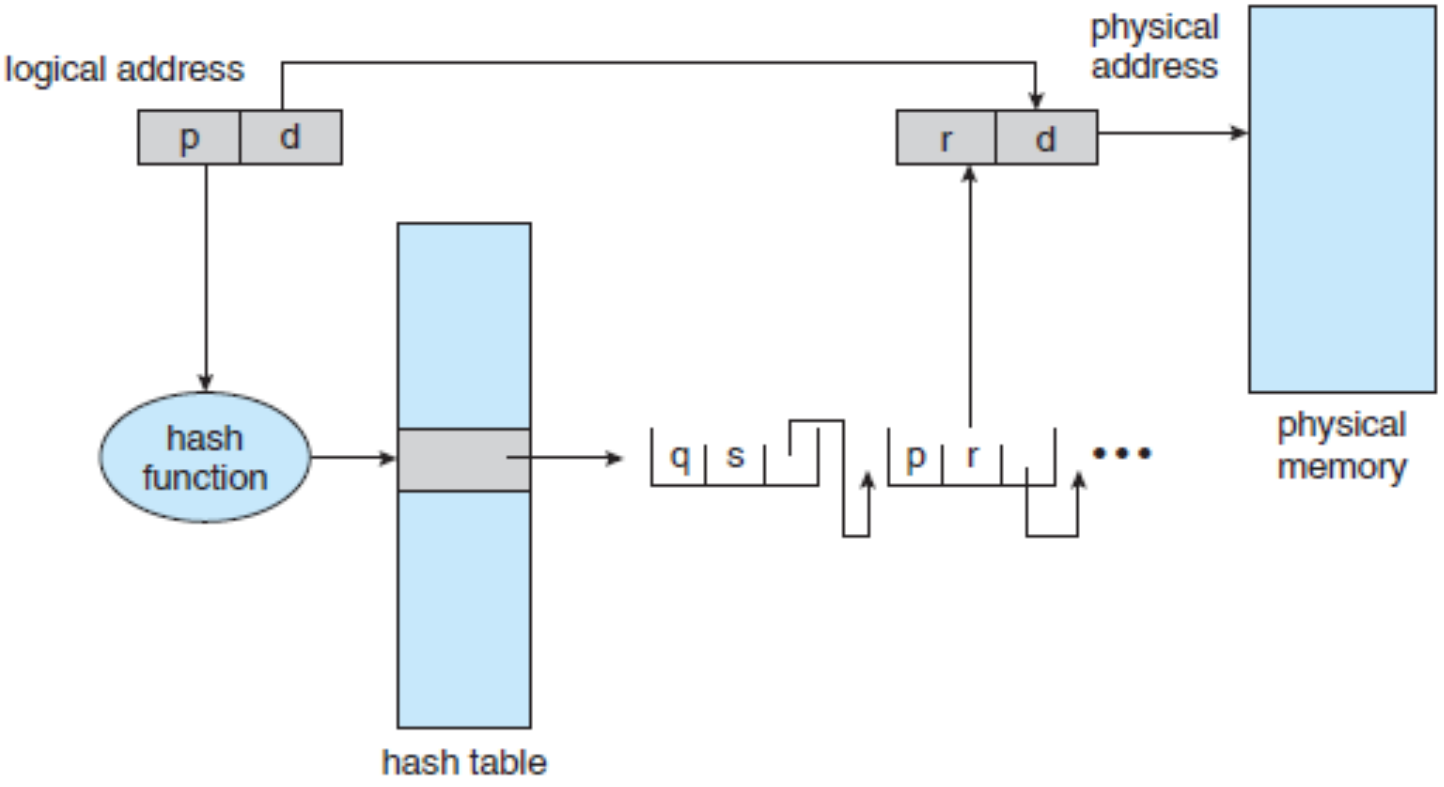
Hierarchical Paging

- Consider a 32-bit logical address space, with 4 KB (2^{12}) page size, then a page table may consist of up to 1 million entries (2^{32}/ 2^{12}). Assuming that each entry consists of 4 bytes, each process may need up to 4MB of physical address.

Hierarchical paging



Hash paged table



Inverted Page Table

- Each process has an associated page table. The table has one entry for each page the process is using. This results in millions of entries for each page table.
- To solve the aforementioned problem, we can use an inverted page table. An inverted page table has one entry for each real page of memory. Each entry consists of the process-id and page number.

Inverted Page Table

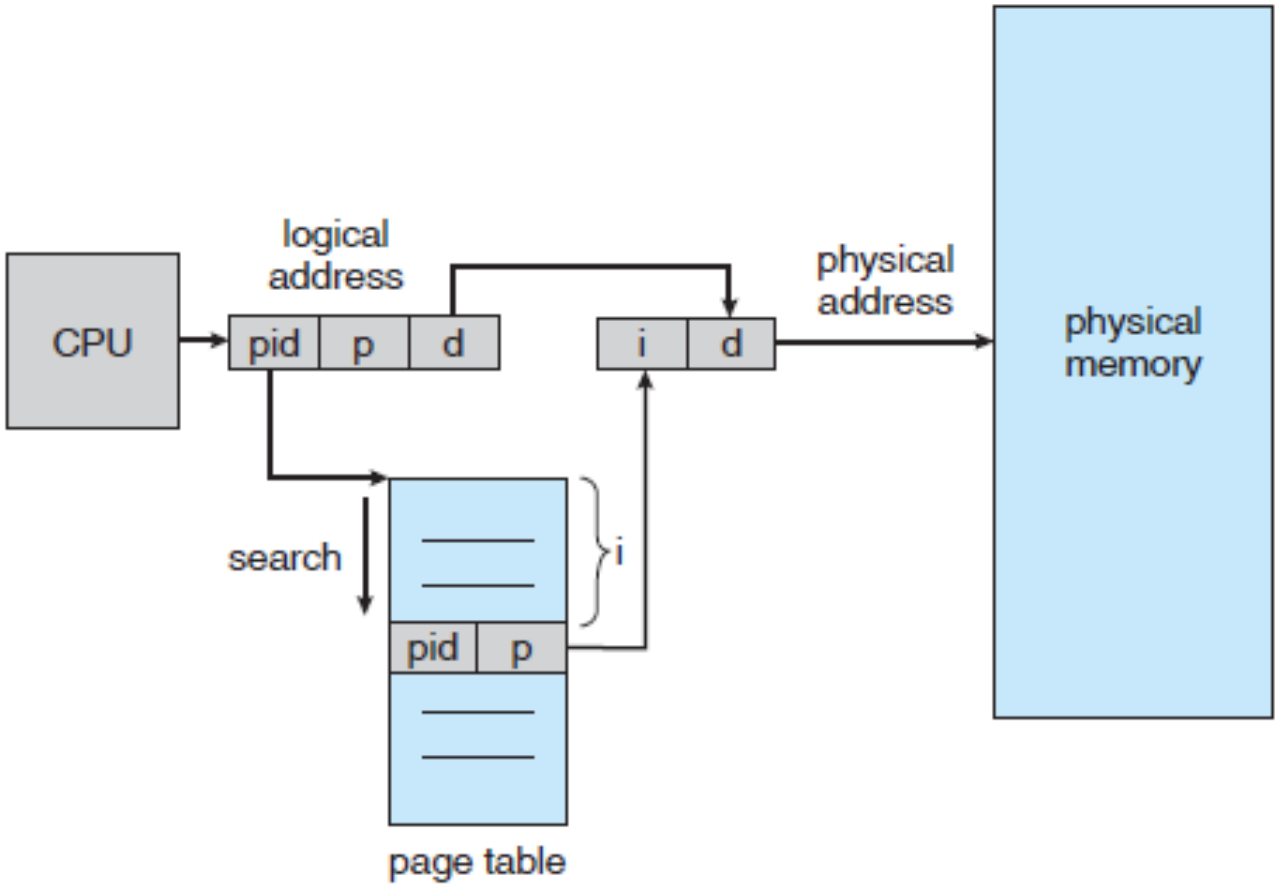
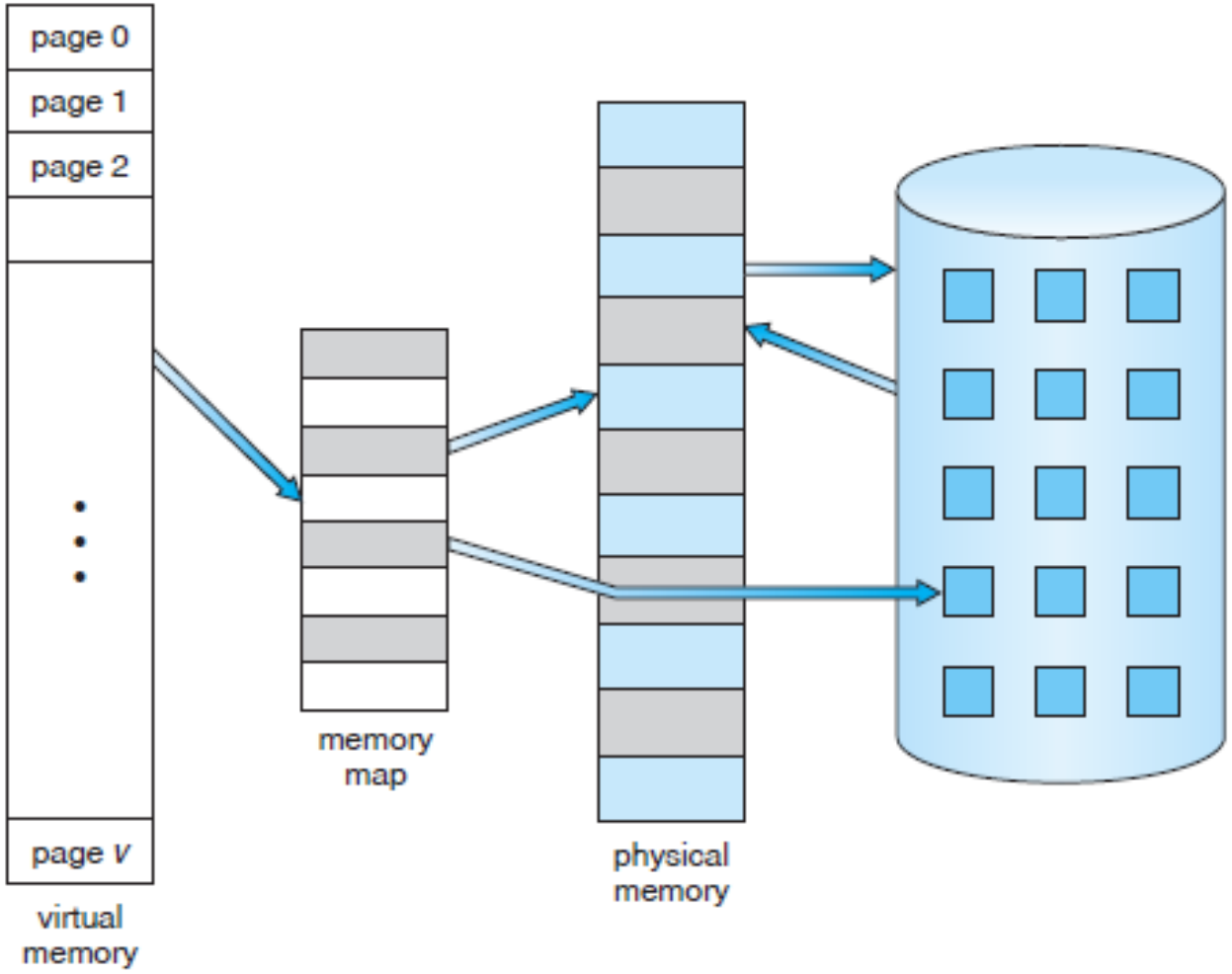
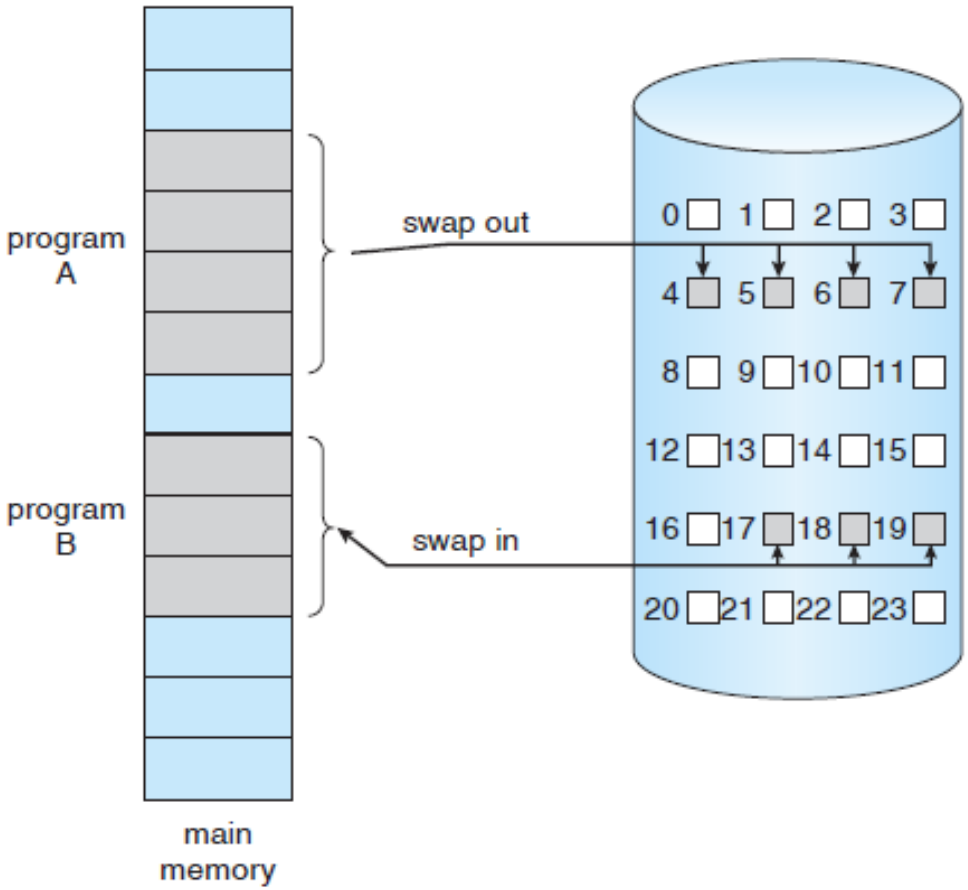


Figure 7.17 Inverted page table.

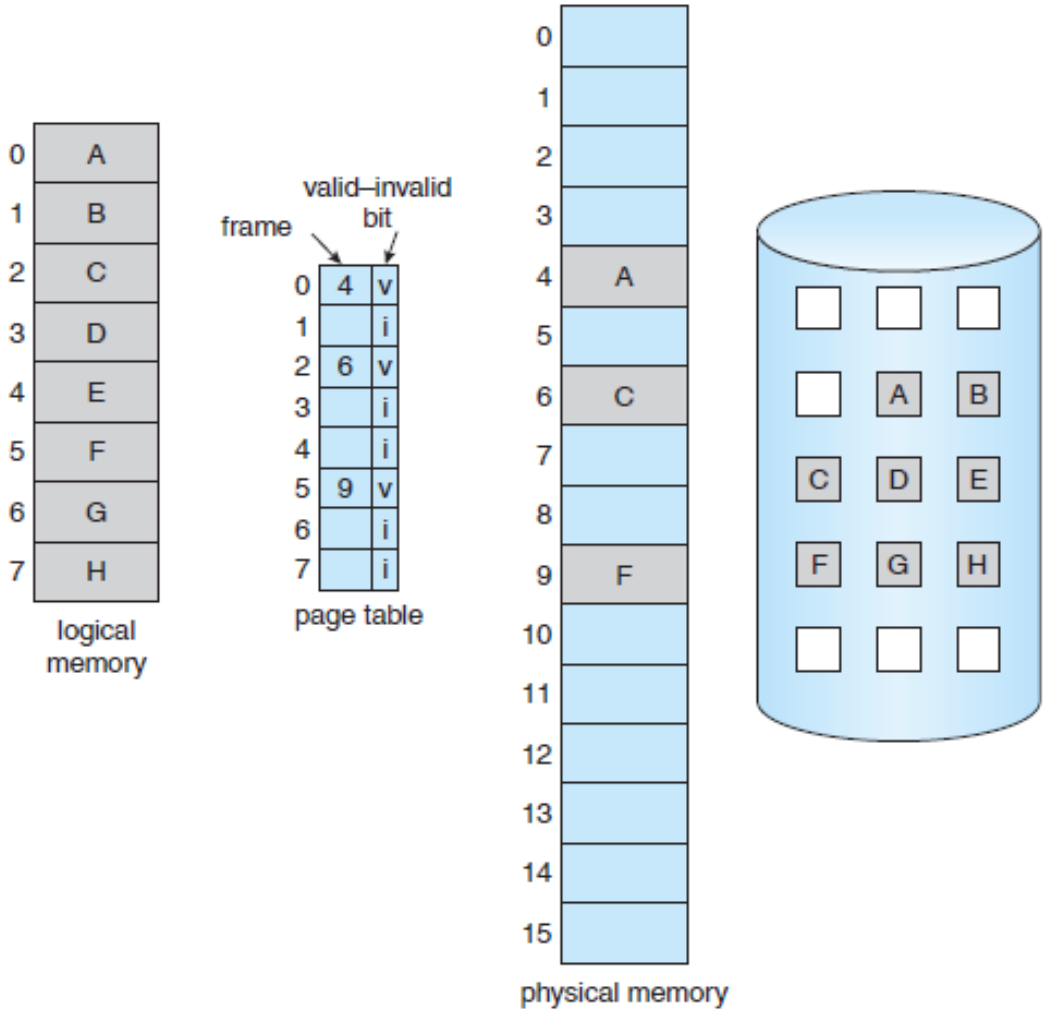
Virtual memory



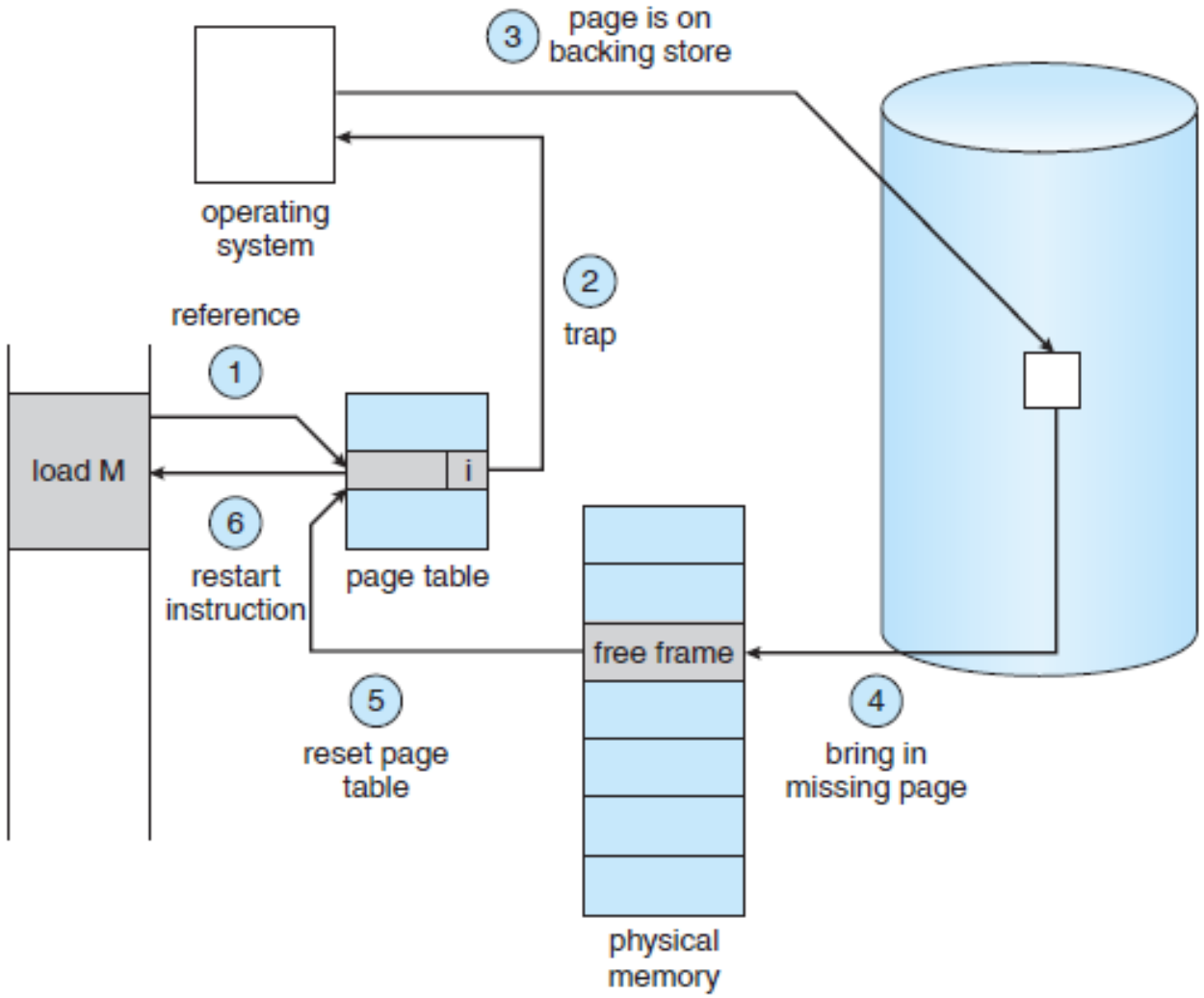
Transfer of paged memory to contiguous disk space



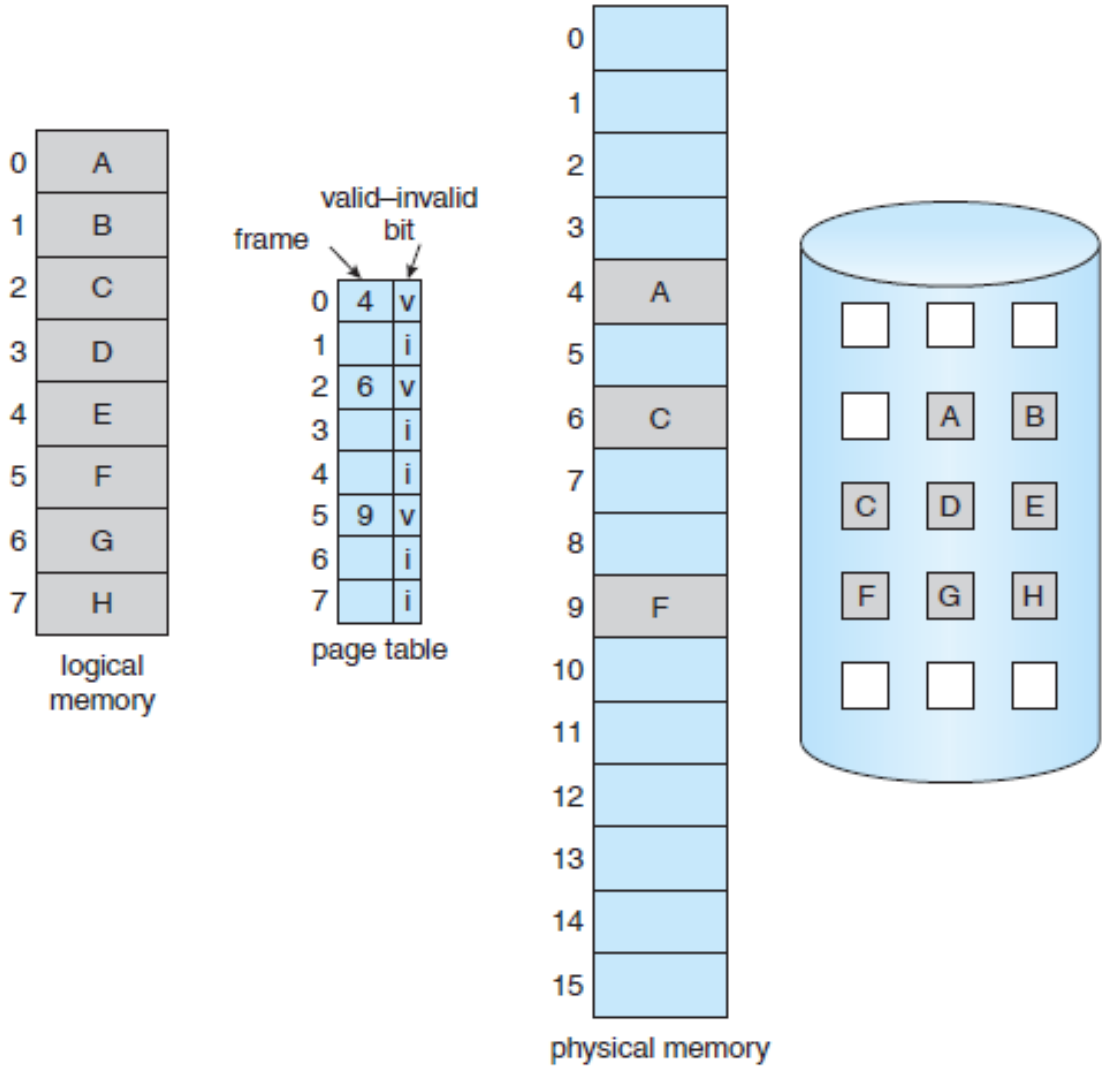
Page table when some pages are not in main memory



Handling a Page Fault



Demand Paging

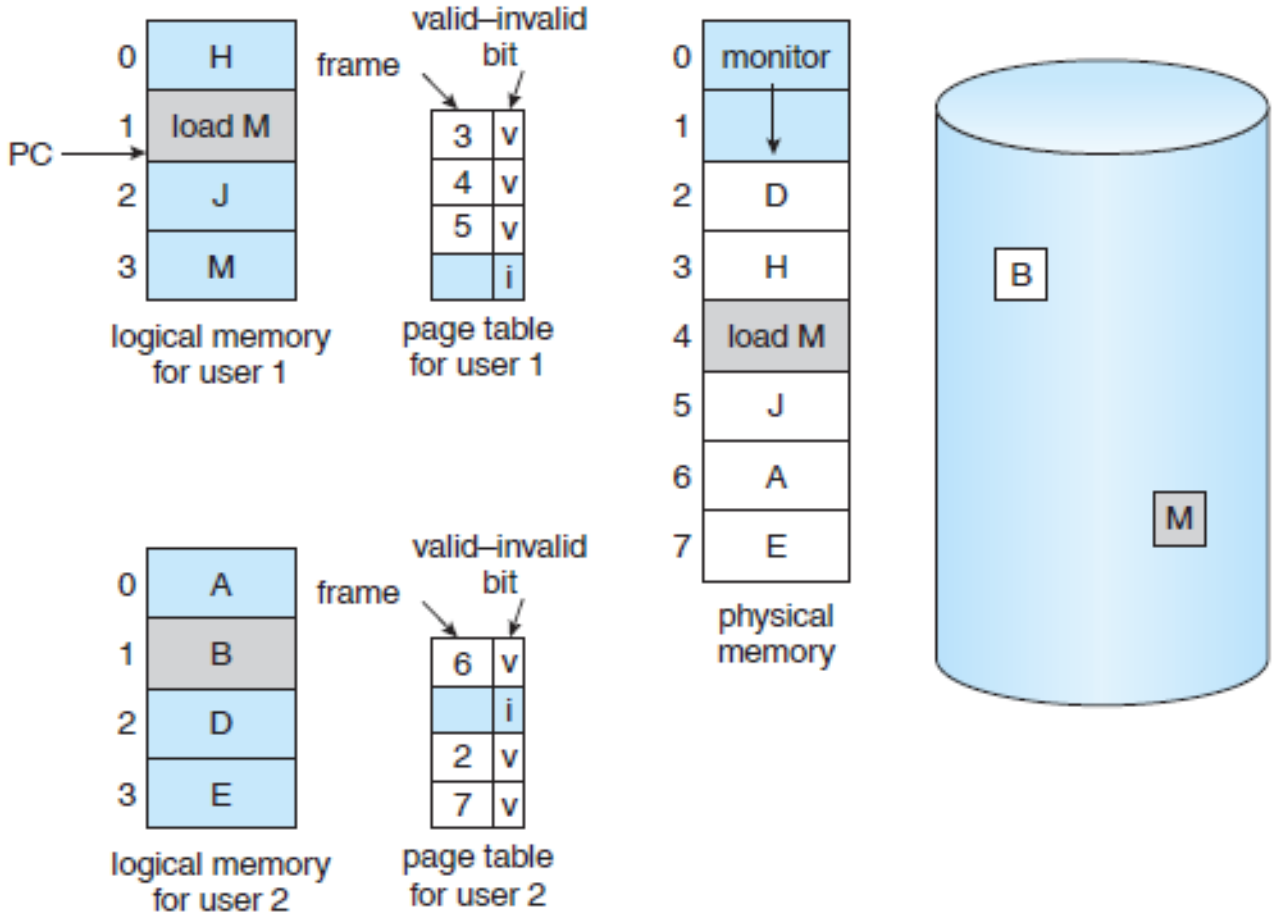


Performance of Demand Paging

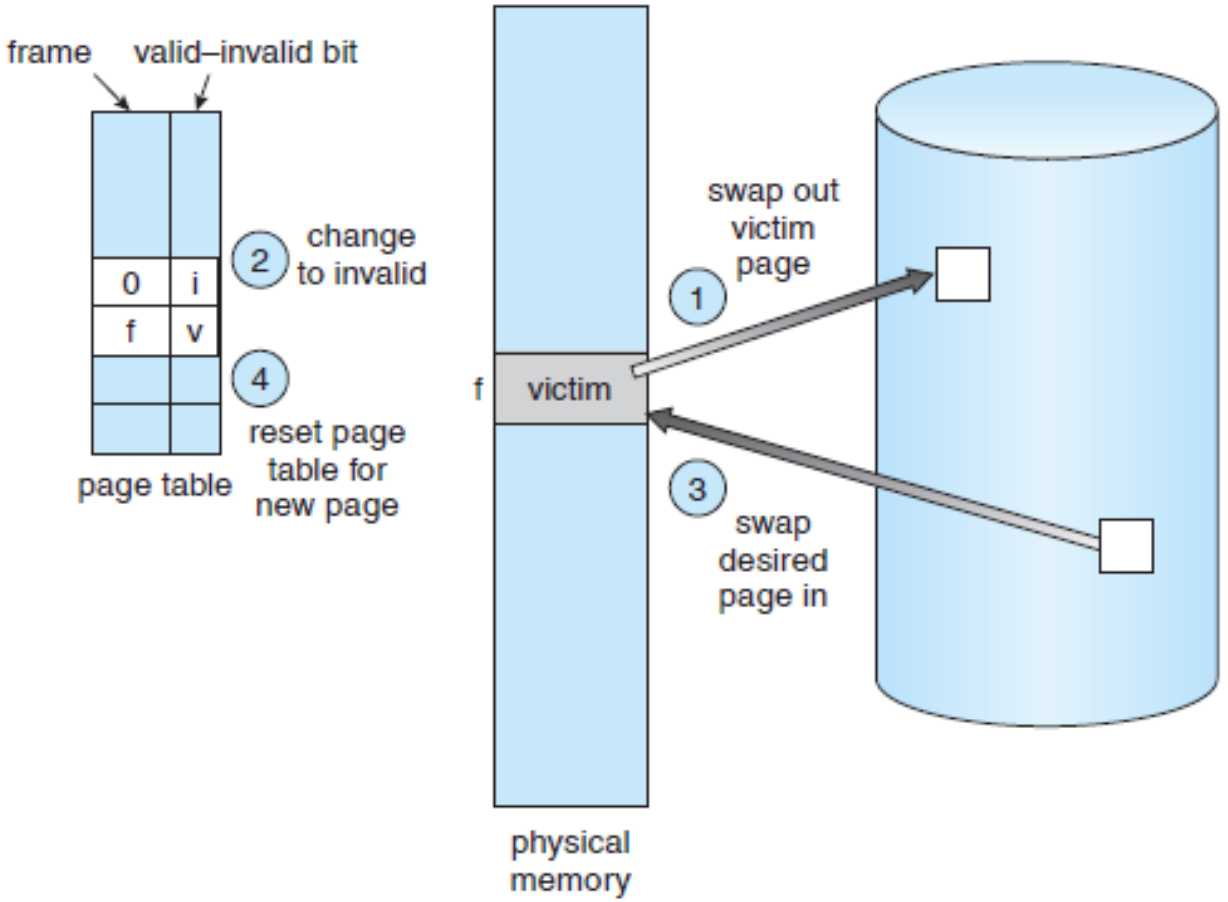
- Let p be the probability of a page fault. We would expect p to be close to zero. The effective access time becomes:

(1-p)*mem_acc_time+p*(page_fault_time)

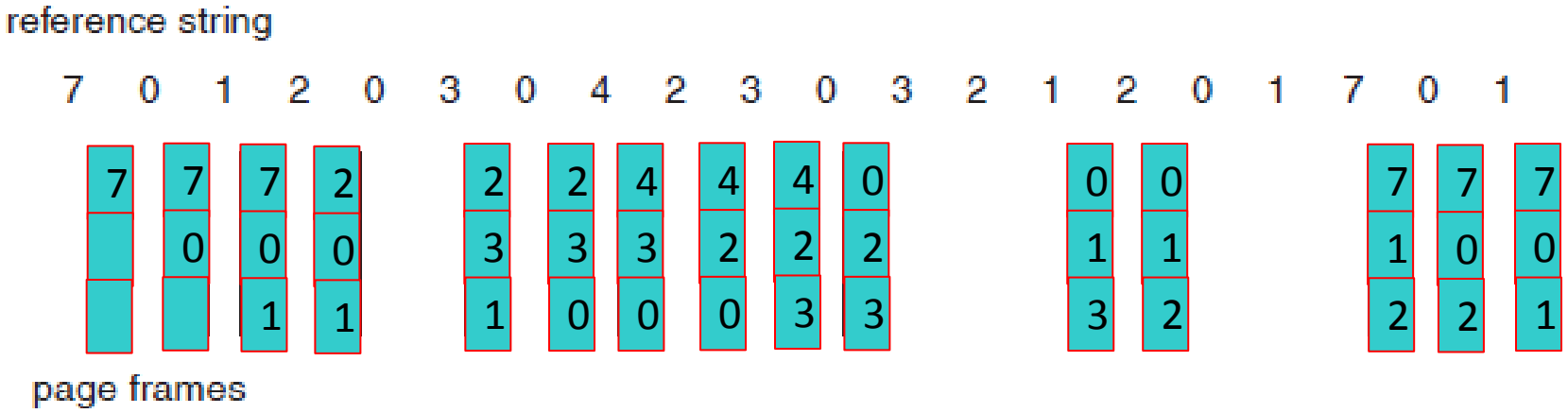
Need for Page Replacement



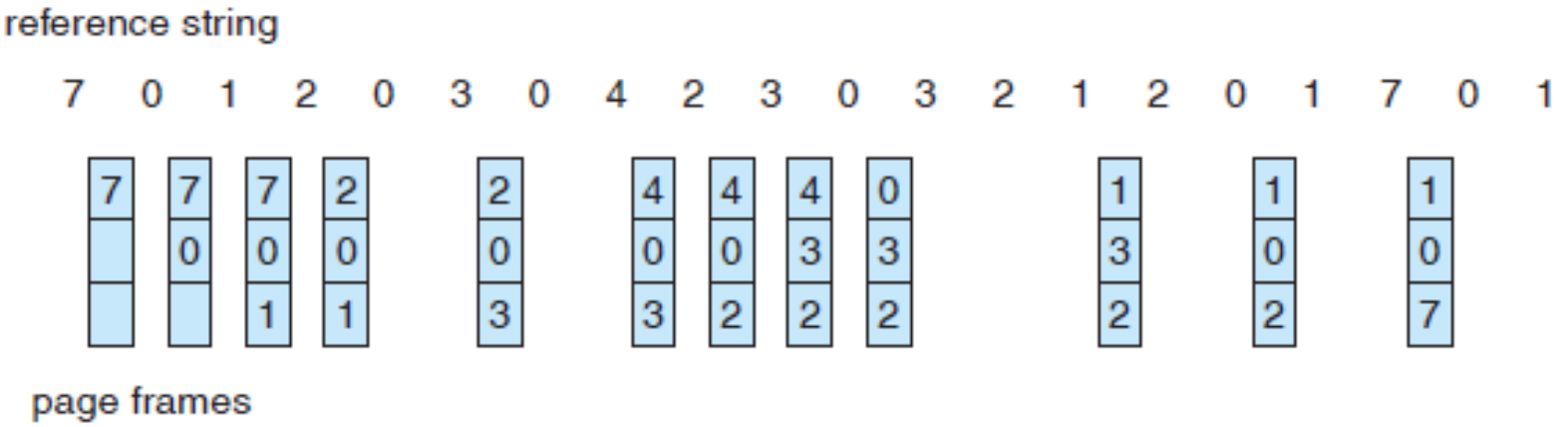
Page Replacement



FIFO page-replacement



LRU Page Replacement



Non Uniform Memory Access (NUMA)

- In systems with multiple CPUs, a given CPU can access some sections of main memory faster than it can access others.
- The performance differences are caused by how CPUs and memory are interconnected within the system.
- Systems in which memory accesses times vary significantly are known as non-uniform memory access (NUMA) systems.