KOÇ UNIVERSITY

Paging

Didem Unat

Lecture 17

COMP304 - Operating Systems (OS)

Logical vs. Physical Address Space

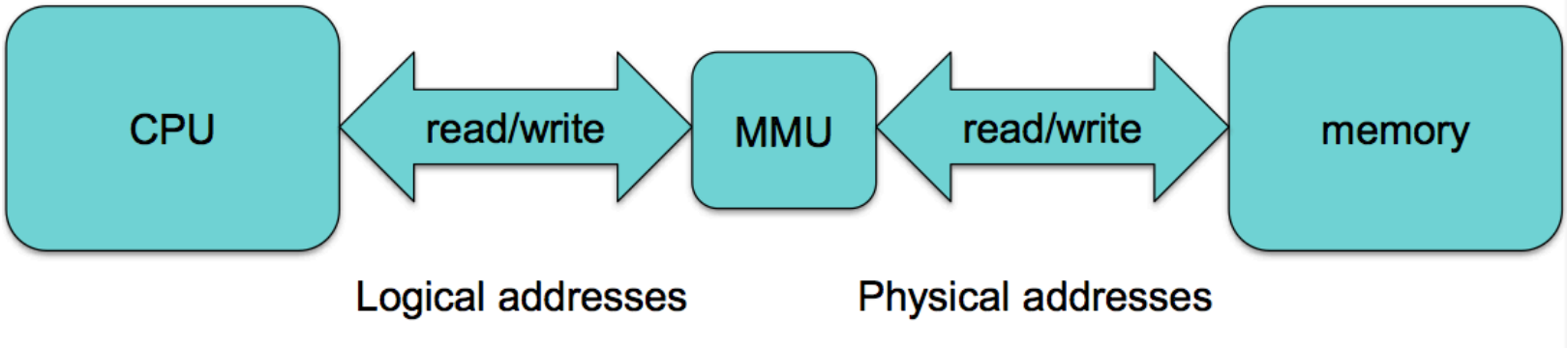
- The concept of a **logical address space** that is bound to a separate **physical address space** is central to proper memory management.
  - **Logical address** – generated by the CPU; also referred to as **virtual address**.
  - **Physical address** – address seen by the memory unit.

- The user program deals with logical addresses; it never sees the real physical addresses.

Logical Addressing

Memory management unit (MMU):

- Real-time, on-demand translation between *logical* (virtual) and *physical* addresses

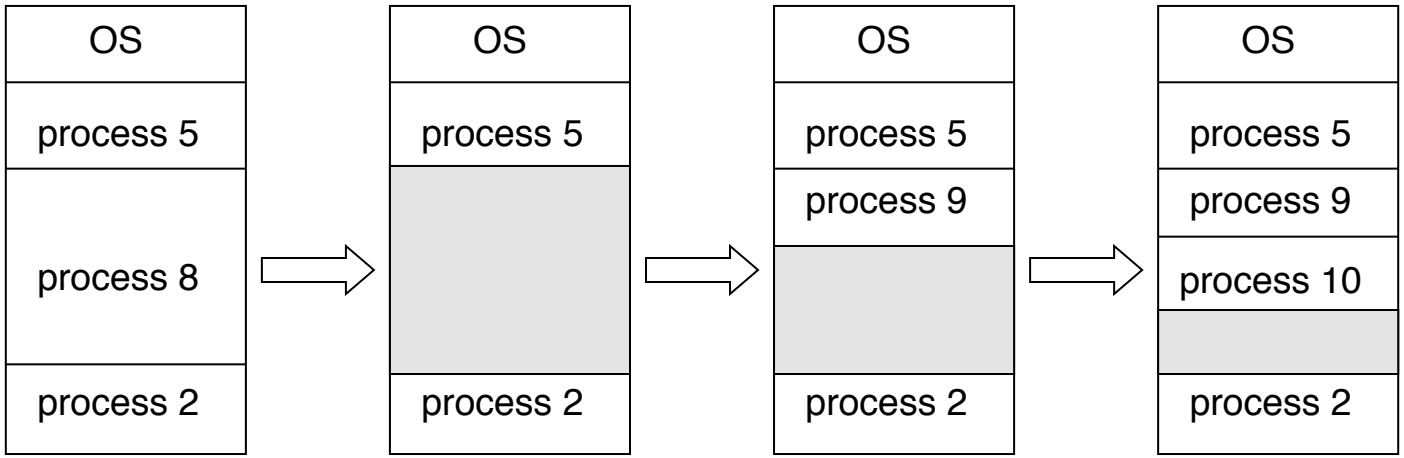


Memory Allocation

- Main memory must support both OS and user processes
- Limited resource, must allocate efficiently
- Three methods:
  - Contiguous memory allocation
  - Segmentation
  - Paging

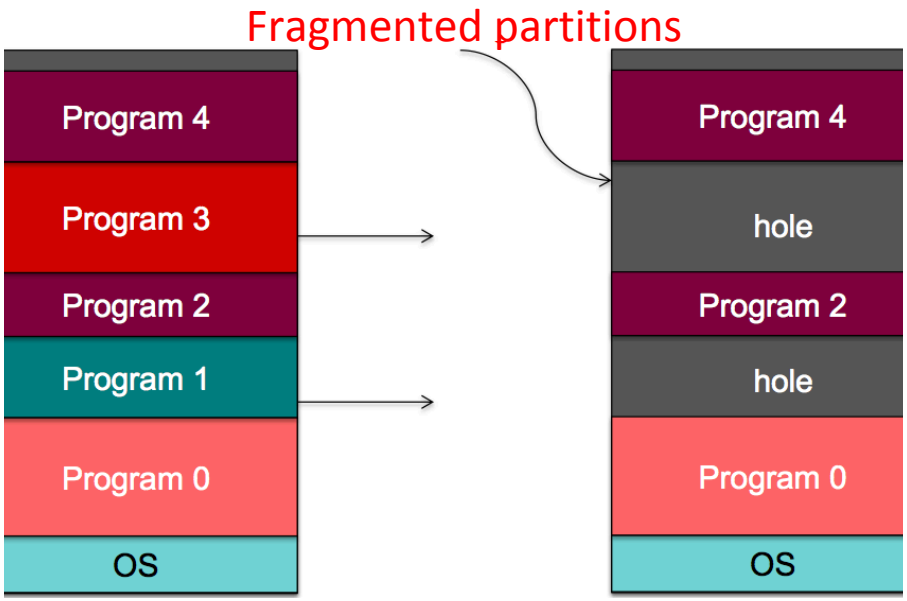
1. Contiguous Allocation

- Multiple-partition allocation
  - **Hole** – block of available memory; holes of various sizes are scattered throughout memory.
  - When a process arrives, it is allocated memory from a hole large enough to accommodate it.
  - Operating system maintains information about:
    allocated partitions and free partitions (holes)



Fragmentation

- **External Fragmentation**
  - total memory space exists to satisfy a request, but it is not contiguous
  - Also a common problem in disk as well



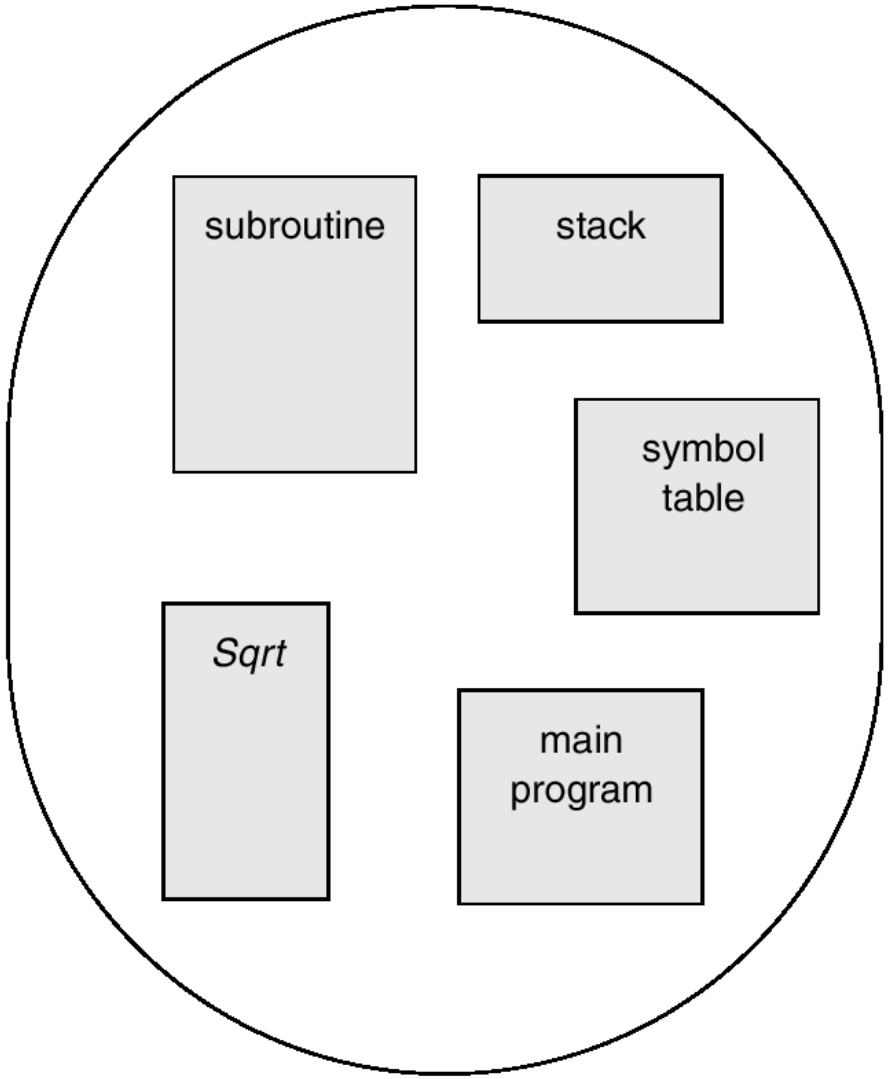
- **Internal Fragmentation**
  - Allocated memory may be slightly larger than requested memory; this size difference is memory internal to a partition, but not being used.

2. Segmentation

- Memory allocation mechanism that supports user view of memory.
- Users prefer to view memory as a collection of variable-sized segments – similar to programmer's view of memory
- A program is a collection of segments. A segment is a logical unit such as:

  main program,
  function,
  method,
  object,
  local variables, global variables,
  common block,
  stack,
  symbol table, arrays

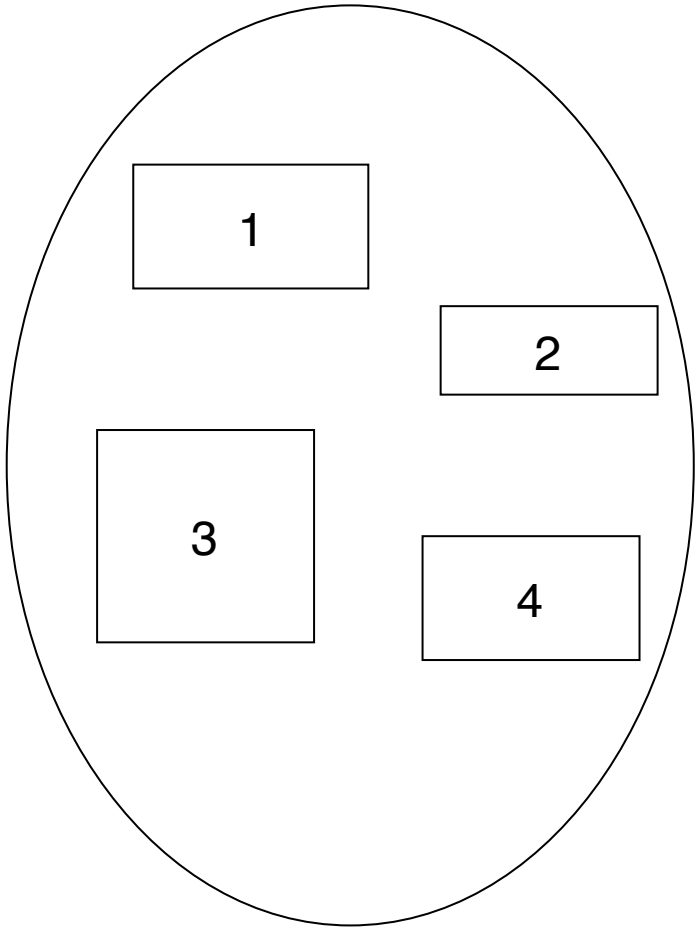
User's View of a Program



logical address space

Logical address space is a collection of segments

Logical View of Segmentation



user space

physical memory space

3. Paging

- Logical address space of a process can be noncontiguous; process is allocated physical memory.

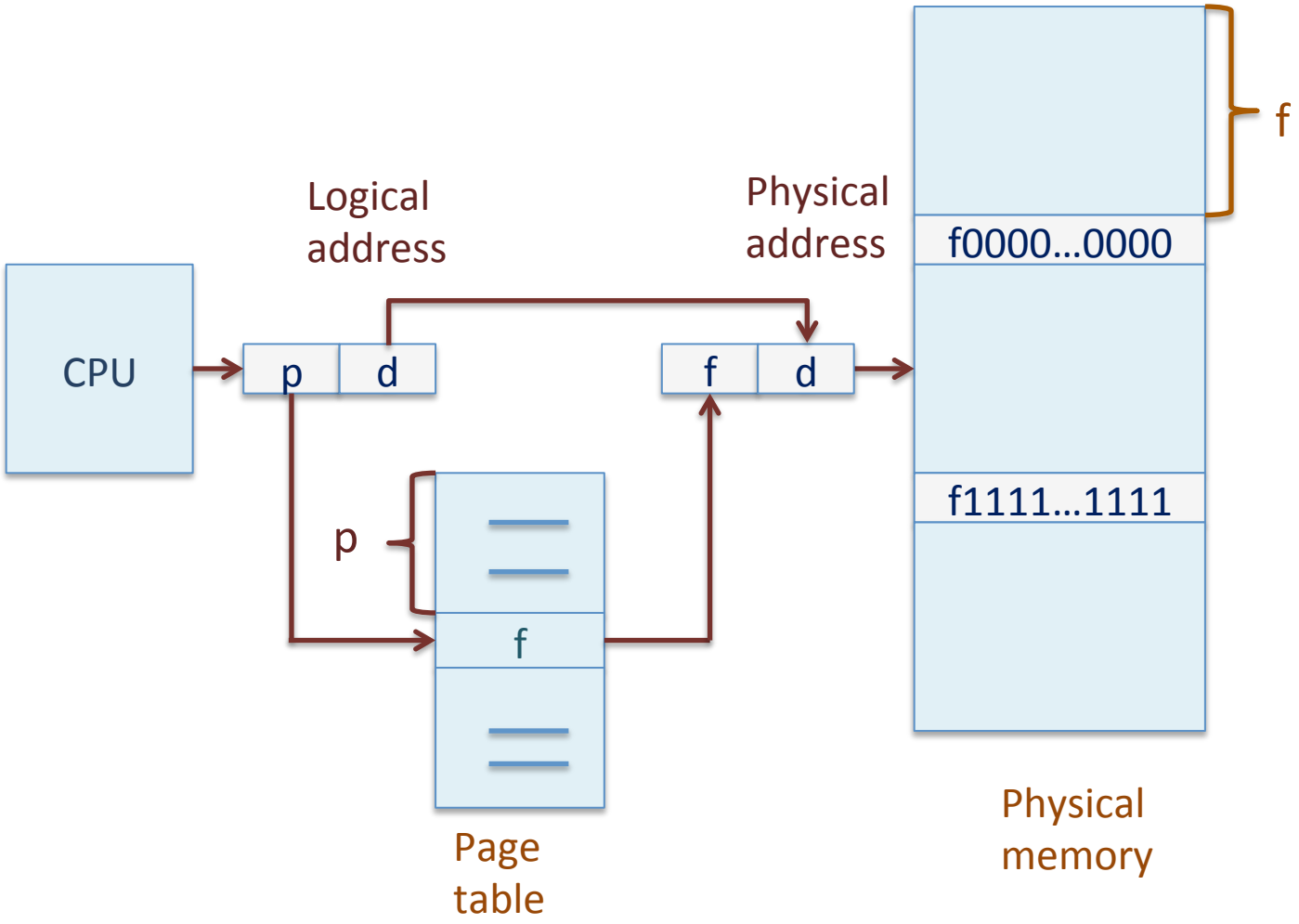
- Divide physical memory into fixed-sized blocks called **frames** (size is power of 2, between 512 bytes and 16MB).
- Divide logical memory into blocks of the same size regions called **pages**.

- Keep track of all free frames.
  - Set up a page table to translate logical to physical addresses.

Address Translation Scheme

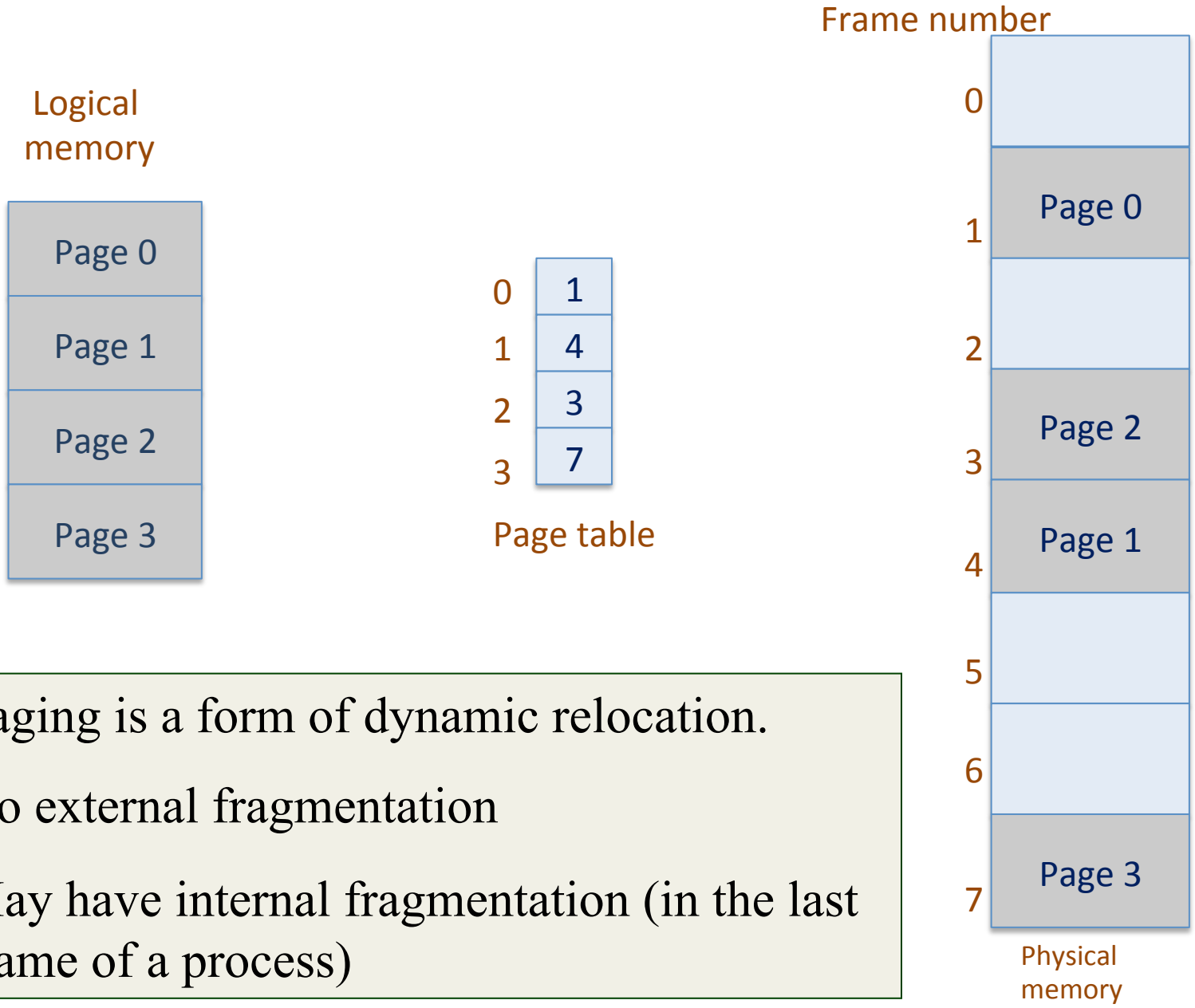
- Address generated by CPU (logical address) is divided into:
  - Page number (p) – used as an index into a page table which contains base address of each page in physical memory.
  - Page offset (d) – combined with base address to define the physical memory address that is sent to the memory unit.

Address Translation Architecture



Paging hardware

Paging Model of Logical and Physical Memory

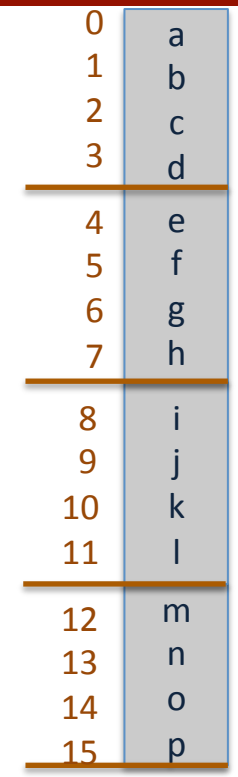


Paging is a form of dynamic relocation.

No external fragmentation

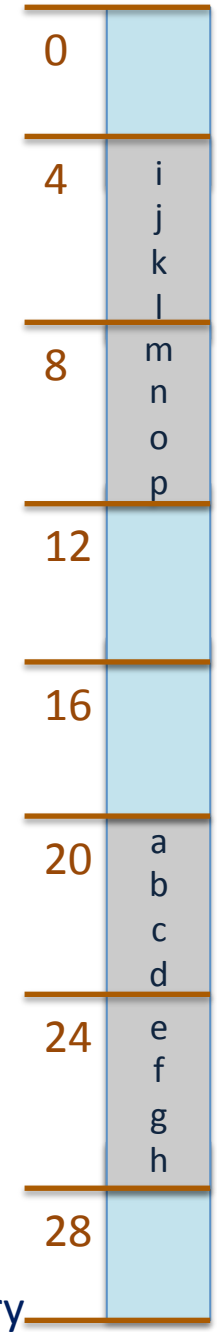
May have internal fragmentation (in the last frame of a process)

Paging Example



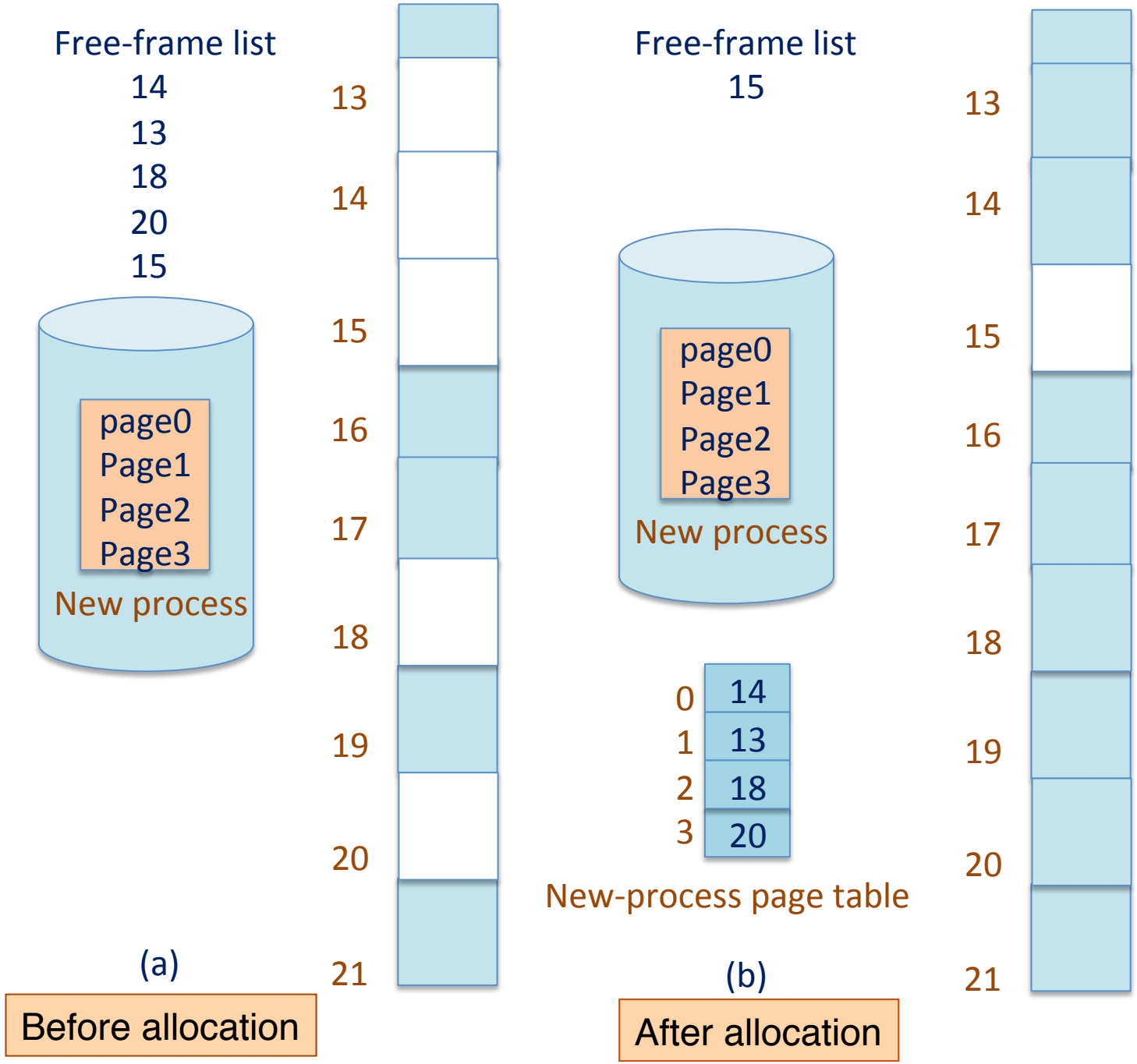
Logical memory

Page table



Physical memory

32-byte memory and 4-byte frames


Free-frame list
14
13
18
20
15
page0
Page1
Page2
Page3
New process
13
14
15
16
17
18
19
20
21
(a)
Before allocation
Free-frame list
15
page0
Page1
Page2
Page3
New process
0 14
1 13
2 18
3 20
New-process page table
13
14
15
16
17
18
19
20
21
(b)
After allocation

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 (PTLR) indicates size of the page table.
- Problem: Every data/instruction access requires two memory accesses:
  - one for the page table and
  - one for the data/instruction.
- Solution: 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)

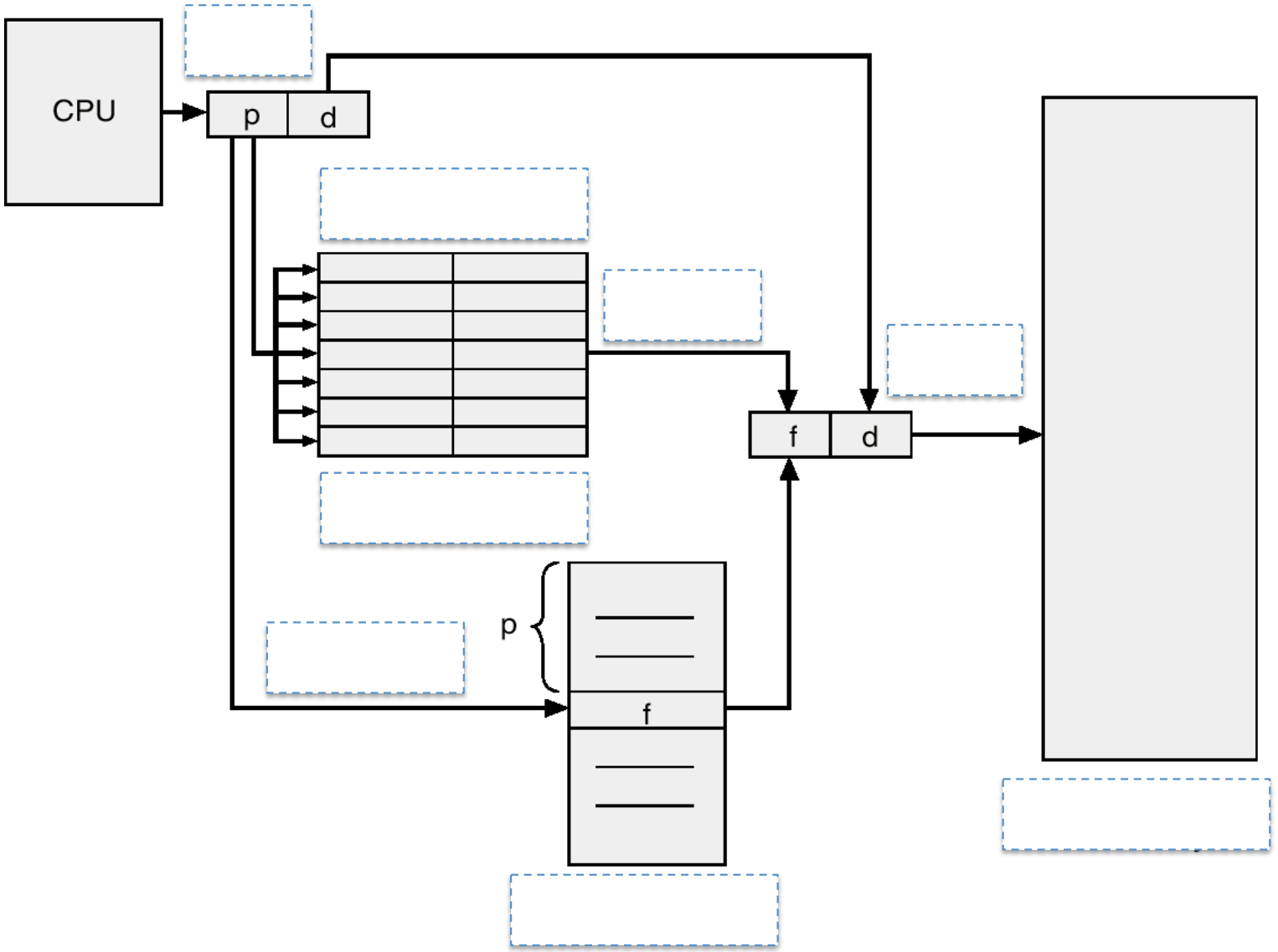
Associative Memory

- Associative memory – parallel search

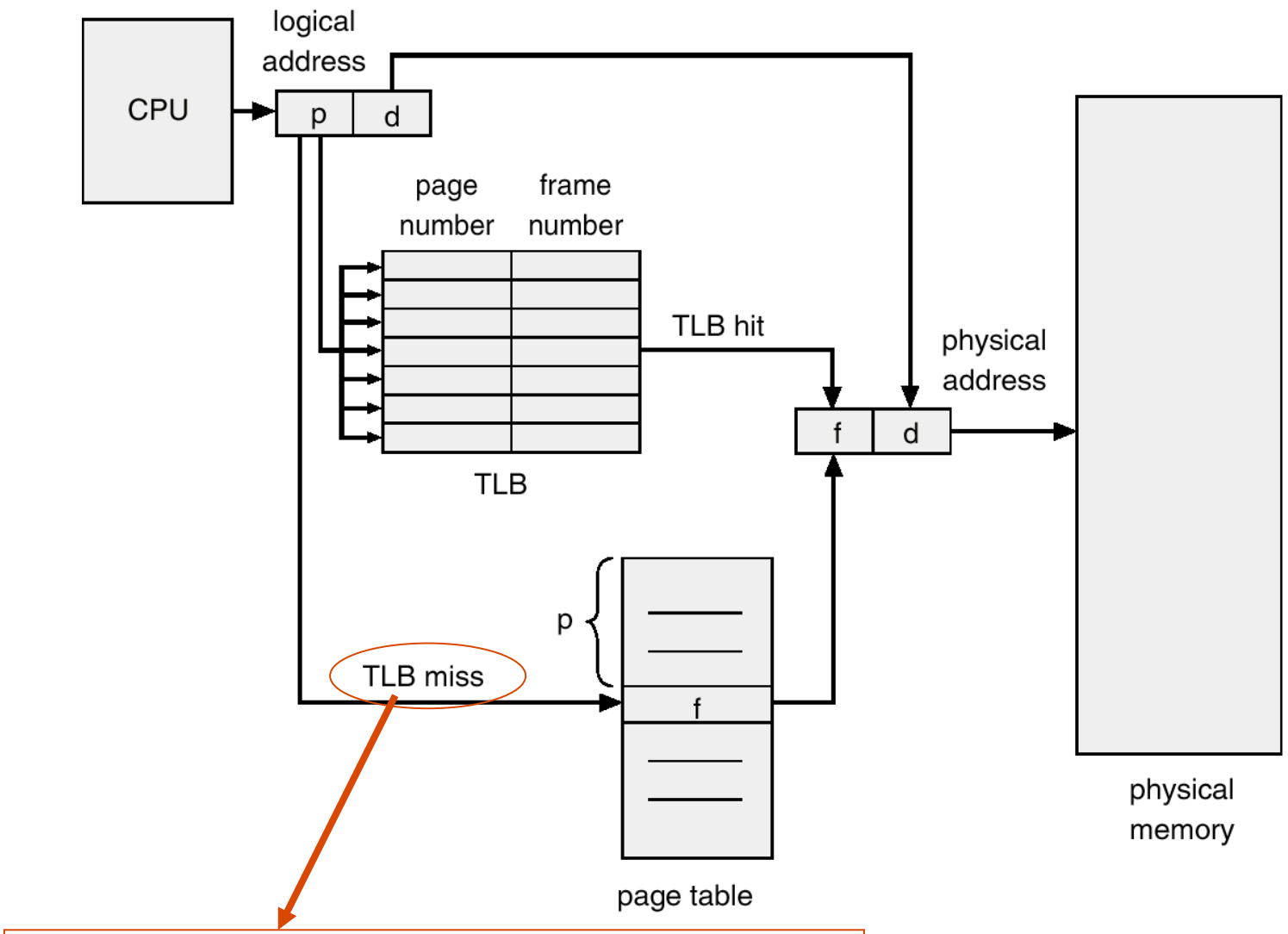
Page #	Frame #

- Address translation (p, d)
  - If p is in associative register, get frame # out.
  - Otherwise get frame # from page table in memory
- Search is fast
- TLB contains some of the page table entries (64 – 1024)

Paging Hardware with TLB



Paging Hardware with TLB



Page # and frame # is added to TLB

Effective Access Time

- Associative Lookup in TLB = ε time unit
- **Hit ratio** – percentage of times that a page number is found in the associative registers (TLB)
  - ratio related to number of associative registers.
  - assume a hit ratio α
- **Effective Access Time (EAT)**

  Example: memory cycle time is 1 time unit

  $EAT = (1 + \varepsilon)\,\alpha + (2 + \varepsilon)(1 - \alpha)$

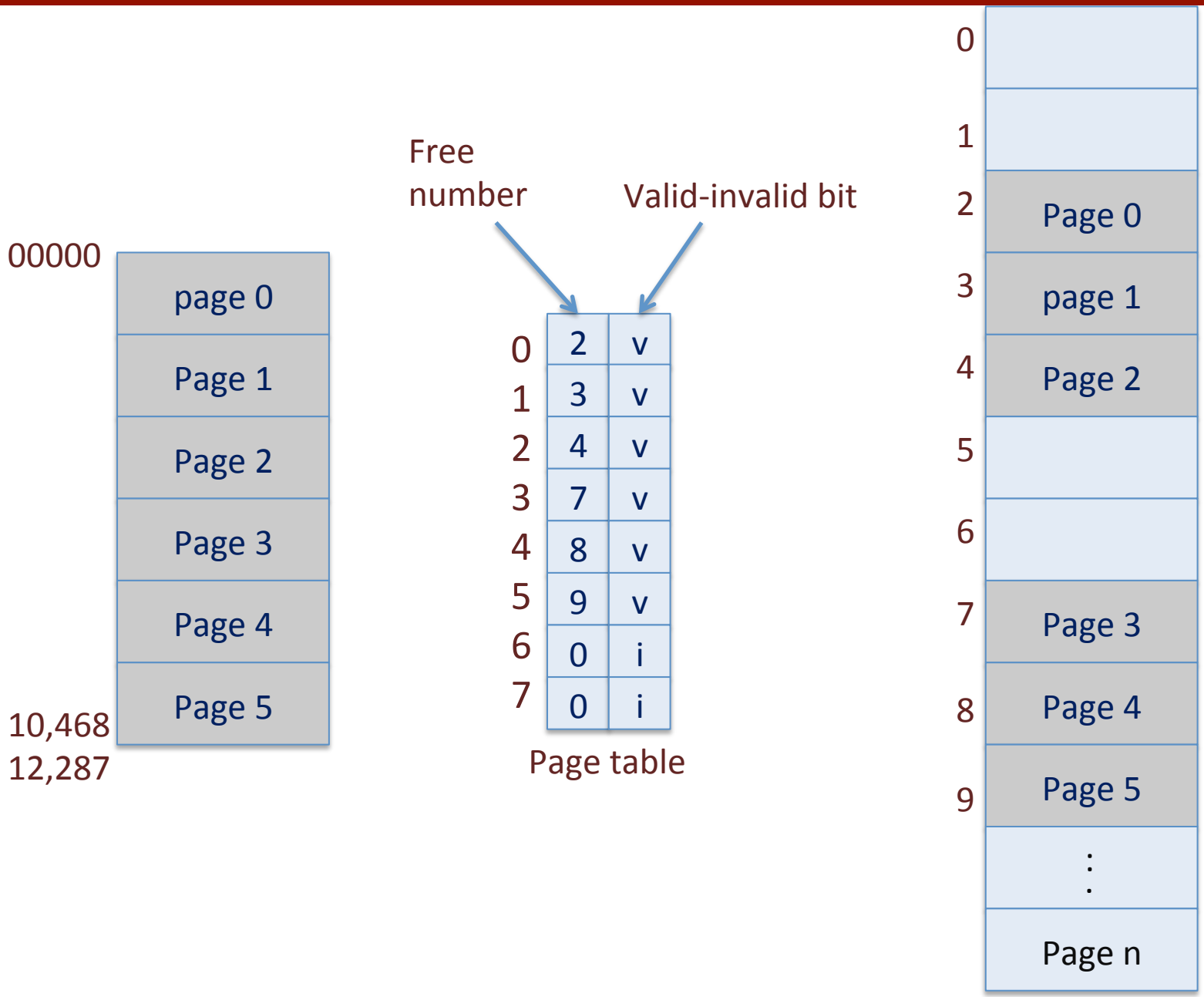
  $= 2 + \varepsilon - \alpha$

Memory Protection

- Memory protection implemented by associating protection bit with each page/frame.

- valid-invalid bit attached to each entry in the page table:
  - valid indicates that the associated page is in the process' logical address space, and is thus a legal page.
  - invalid indicates that the page is not in the process' logical address space.

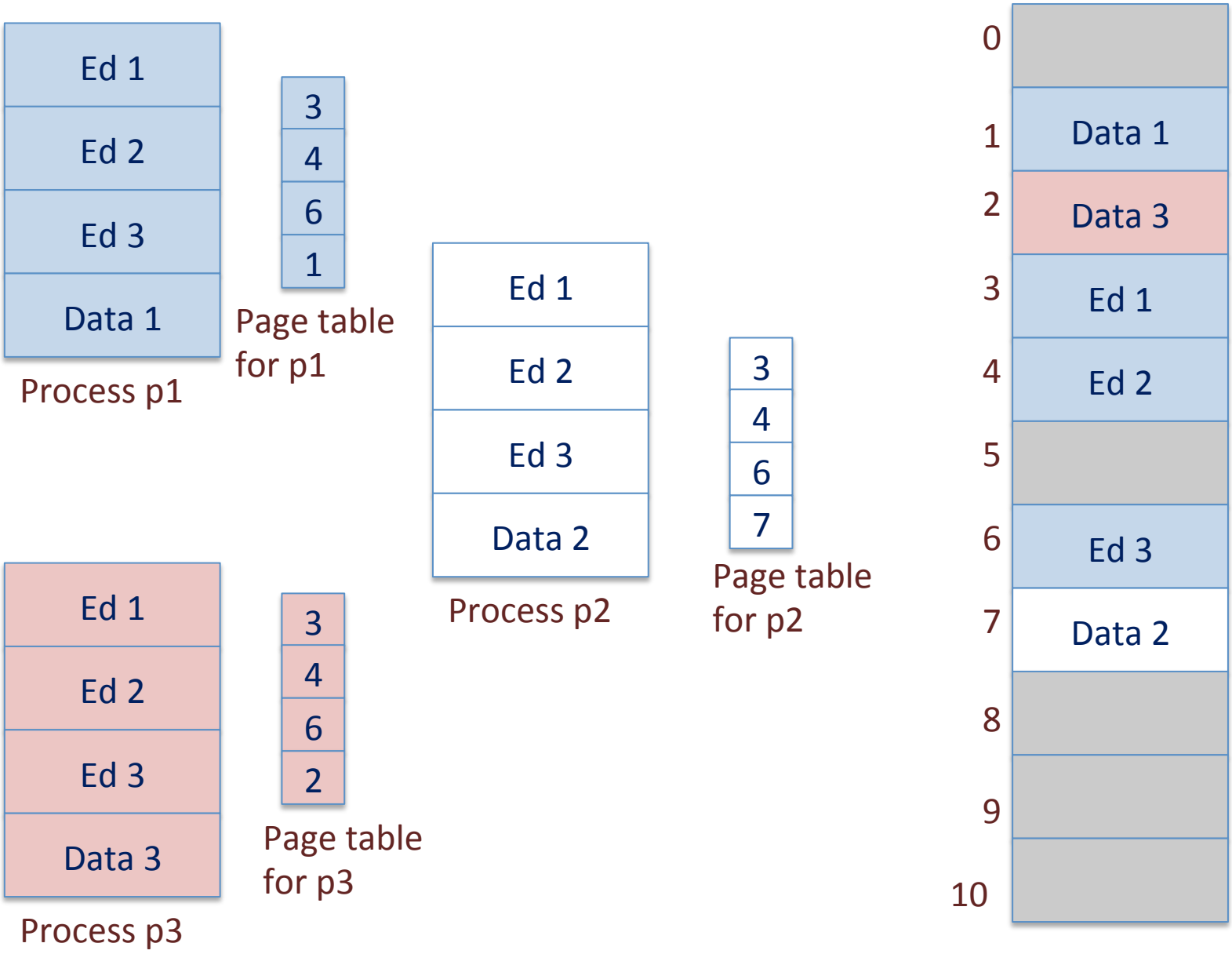
valid (v) or invalid (i) bit in a page table



Shared Pages

- An advantage of paging is the possibility of sharing common code
- Shared code
  - One copy of read-only (reentrant) code shared among processes (i.e., text editors, compilers, window systems).
  - Shared code must appear in same location in the logical address space of all processes.
- Private code and data
  - Each process keeps a separate copy of the code and data.
  - The pages for the private code and data can appear anywhere in the logical address space.

Shared Pages Example



Page Table Structure

- Modern computer systems support a large logical-address space
- Do not want to allocate the page table contiguously in main memory

- Hierarchical Page Tables

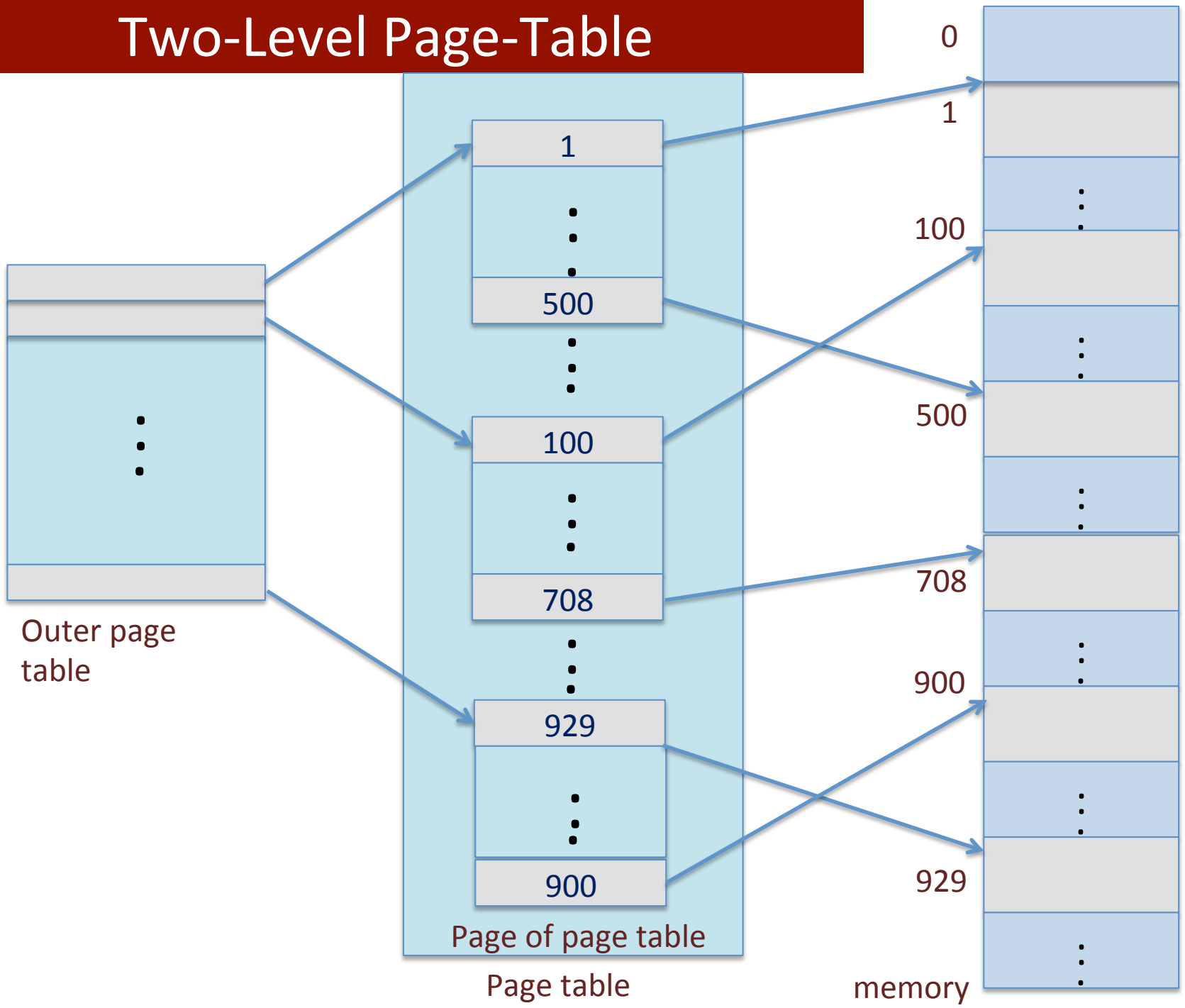
- Hashed Page Tables

- Inverted Page Tables

Hierarchical Page Tables

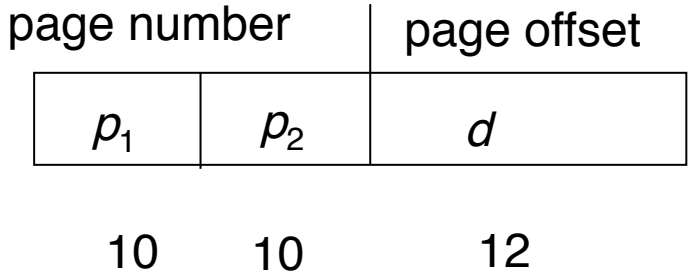
- Break up the logical address space into multiple page tables.
- A simple technique is a two-level page table.
- Page table itself is also paged

Two-Level Page-Table



Two-Level Paging Example

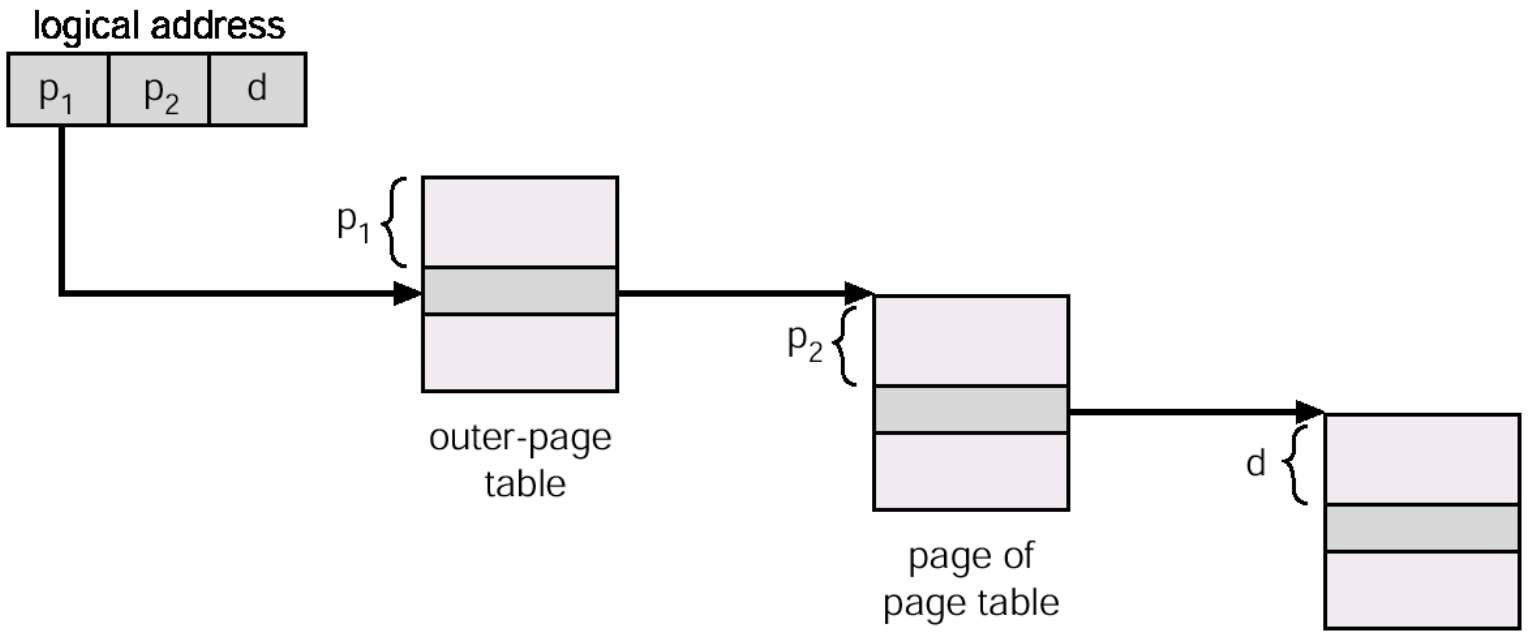
- A logical address (on 32-bit machine with 4K page size) is divided into:
  - a page number consisting of 20 bits.
  - a page offset consisting of 12 bits.
- Since the page table is paged, the page number is further divided into:
  - a 10-bit page number.
  - a 10-bit page offset.
- Thus, a logical address is as follows:



where p_1 is an index into the outer page table, and p_2 is the displacement within the page of the outer page table.

Address-Translation Scheme

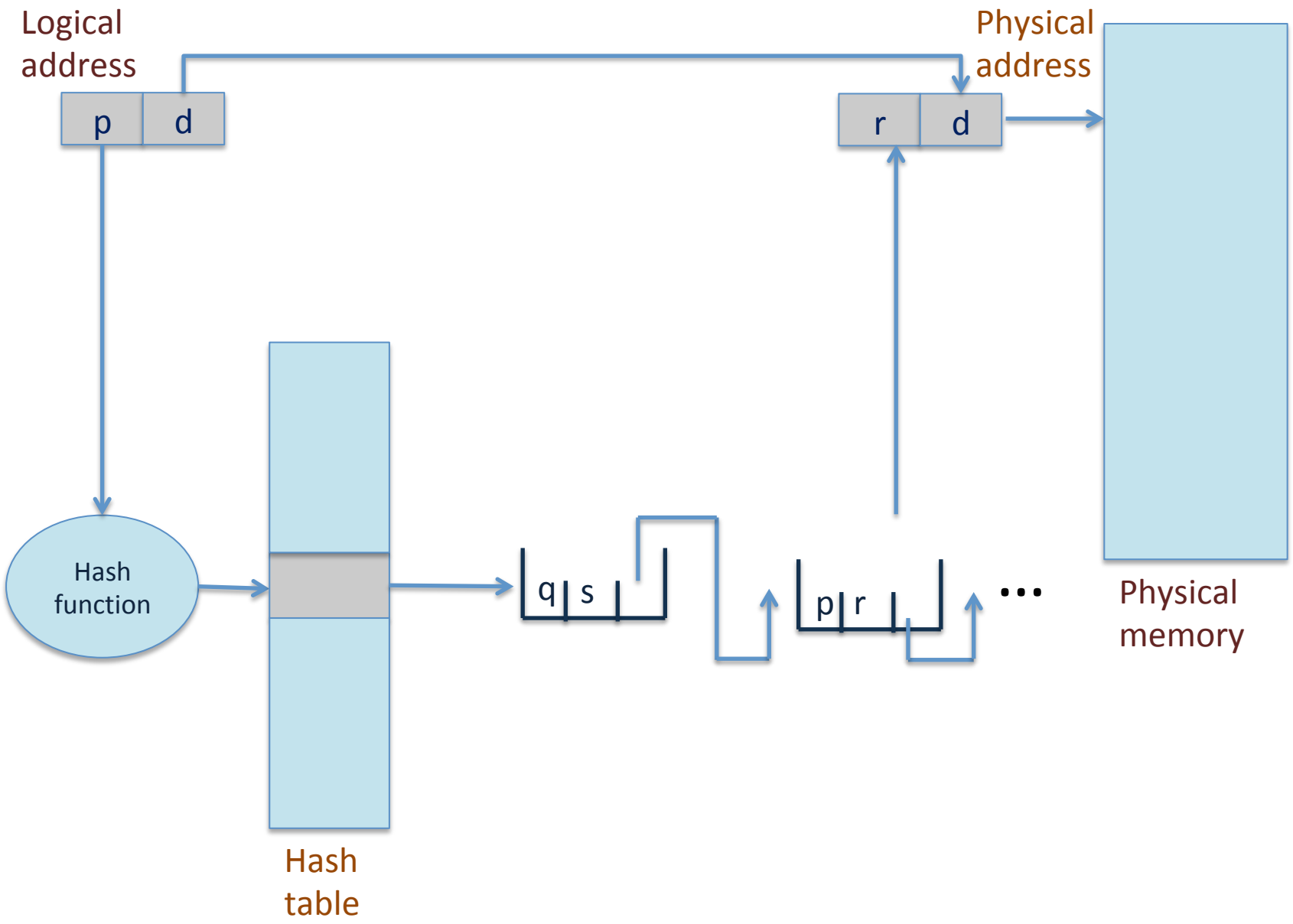
Address-translation for a two-level 32-bit paging architecture



Hashed Page Tables

- Common in address spaces > 32 bits.
- The virtual (logical) page number is hashed into a page table. This hashed page table contains a chain of elements hashing to the same location.
- Virtual page numbers are compared in this chain searching for a match. If a match is found, the corresponding physical frame is extracted.

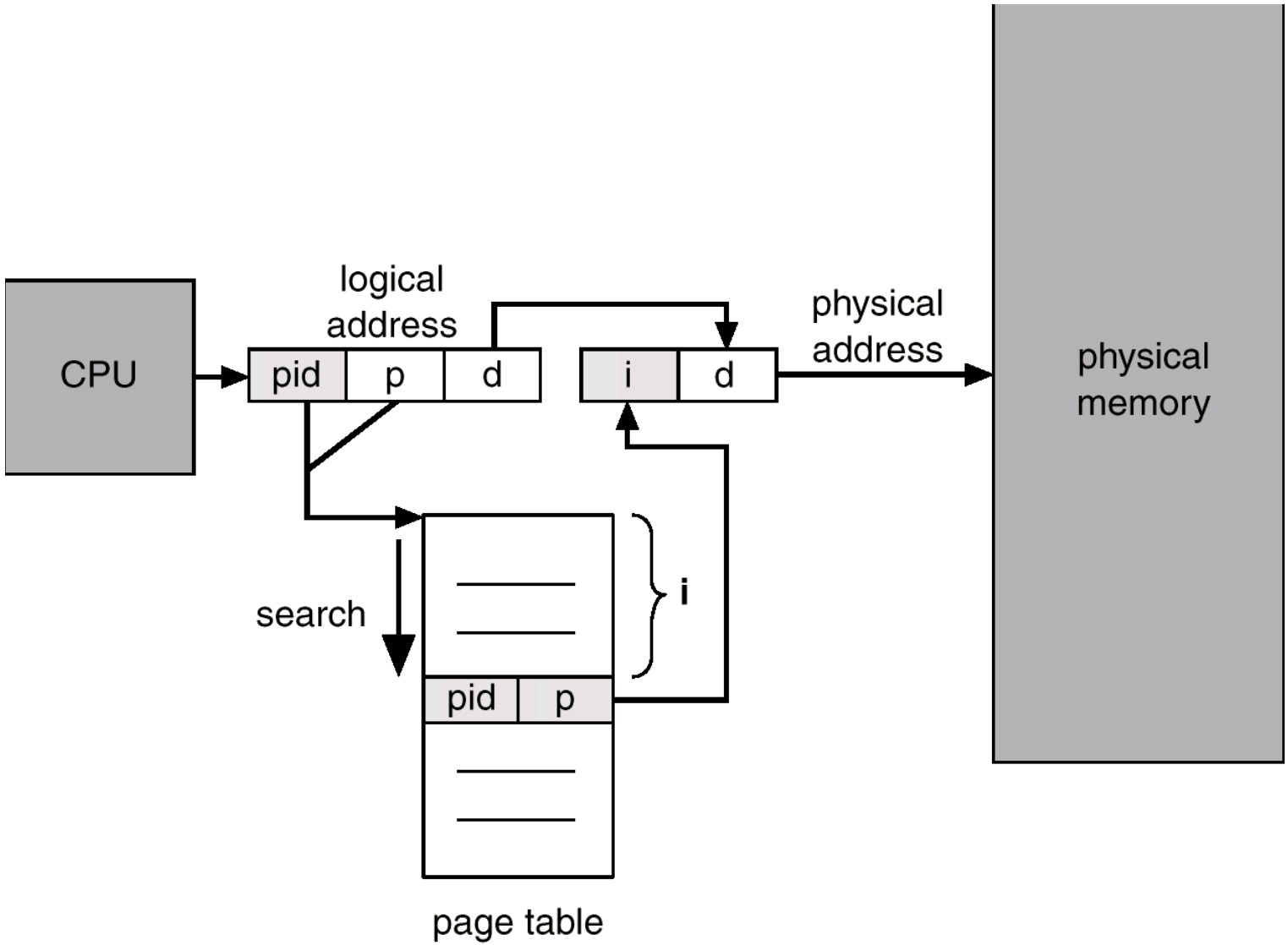
Hashed Page Table



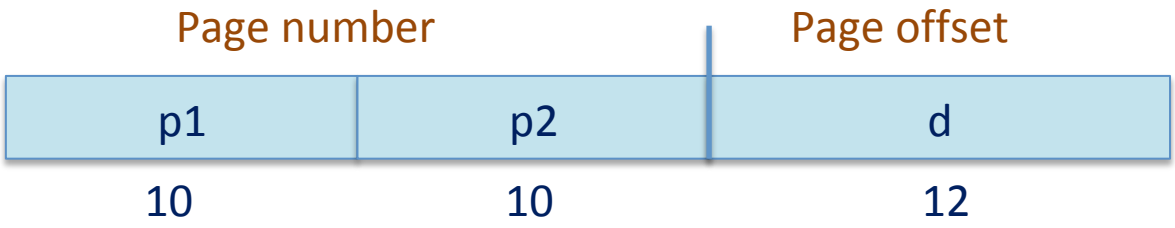
Inverted Page Table

- Only one page table in the system, with one entry for each frame of physical memory.
- Entry consists of the virtual address of the page, with information about the process that owns that page.
- Decreases memory needed to store each page table, but increases time needed to search the table when a page reference occurs.
- Use hash table to limit the search to one — or at most a few — page-table entries.

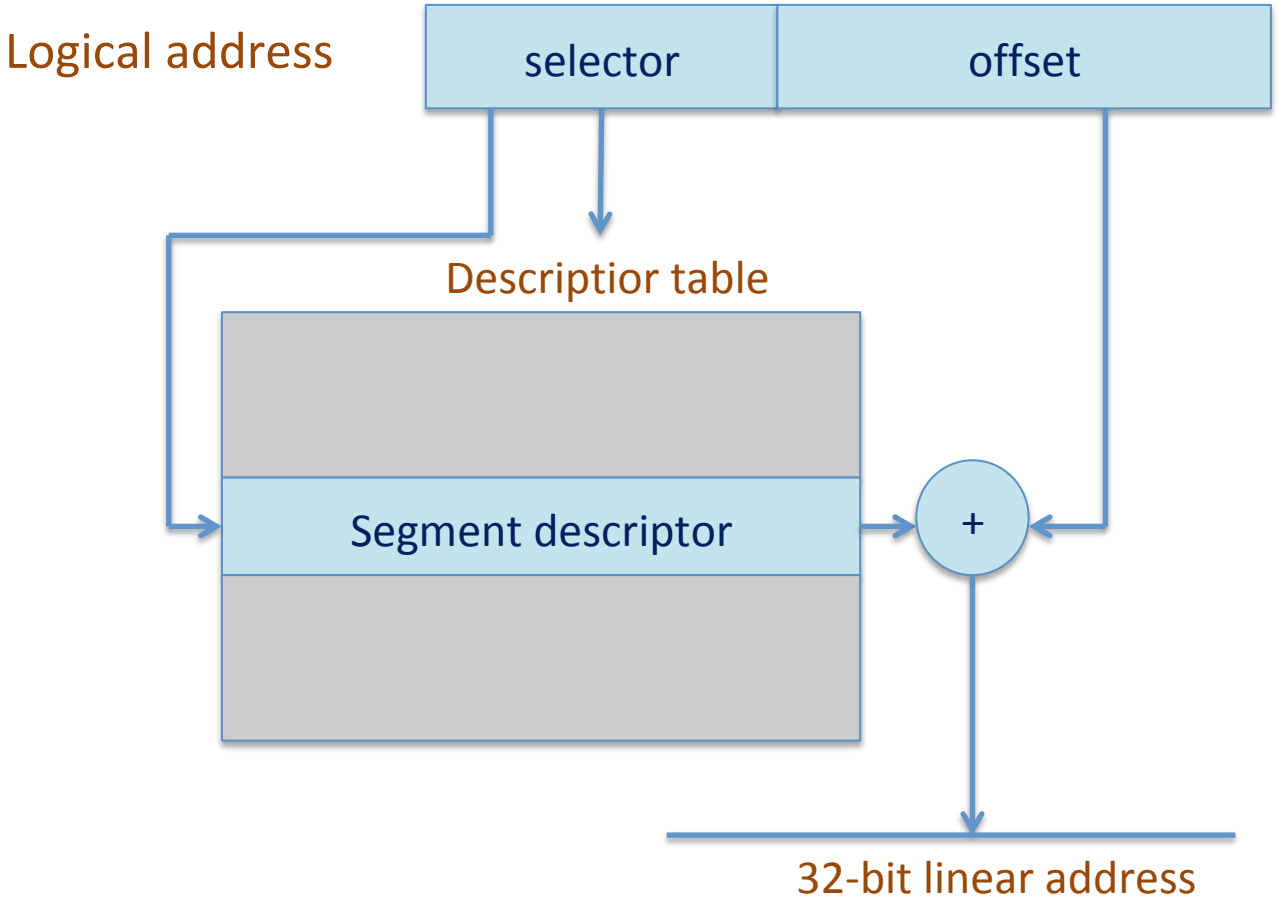
Inverted Page Table Architecture



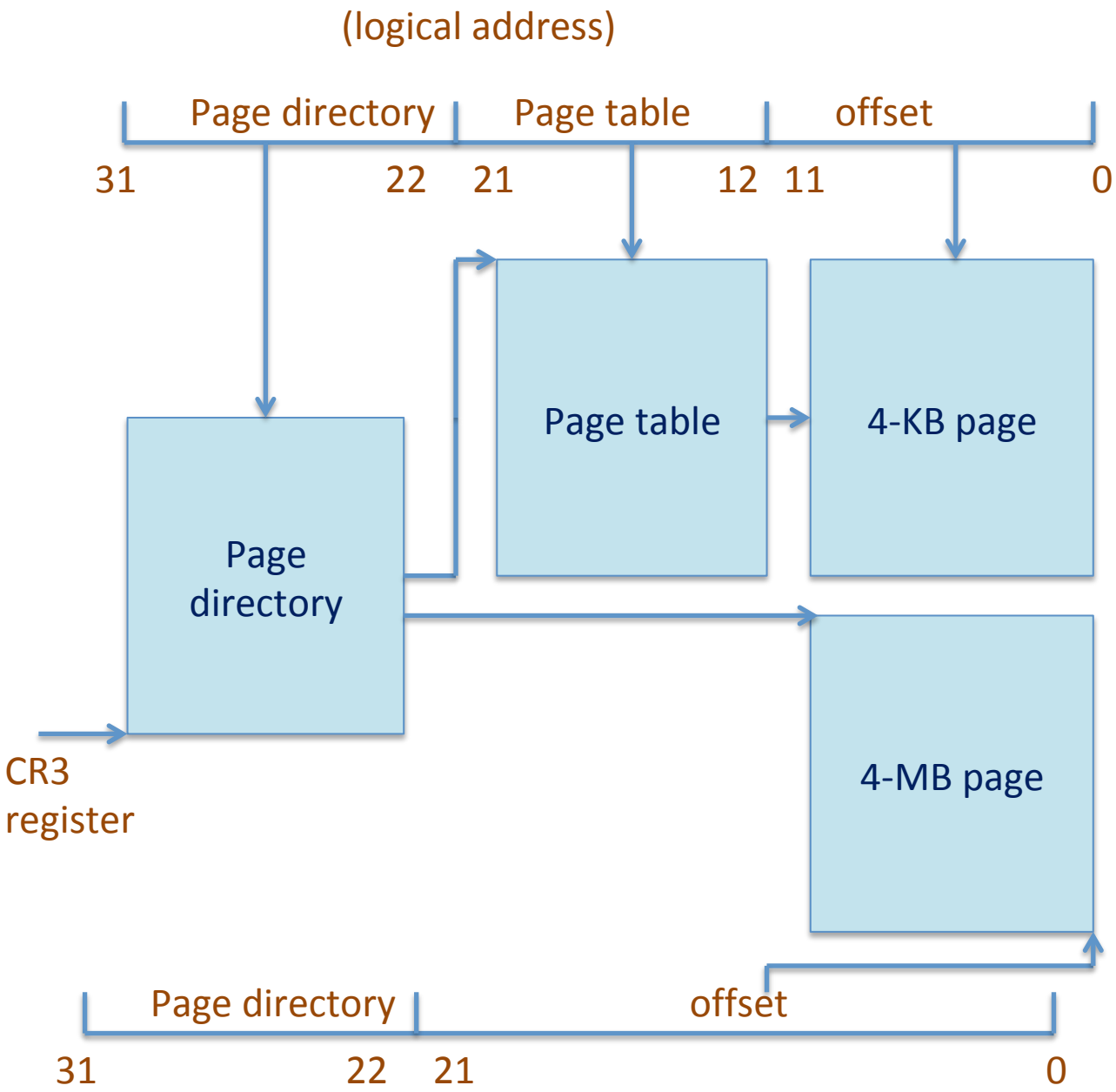
Logical to Physical Address Translation in Pentium



Intel Pentium Segmentation



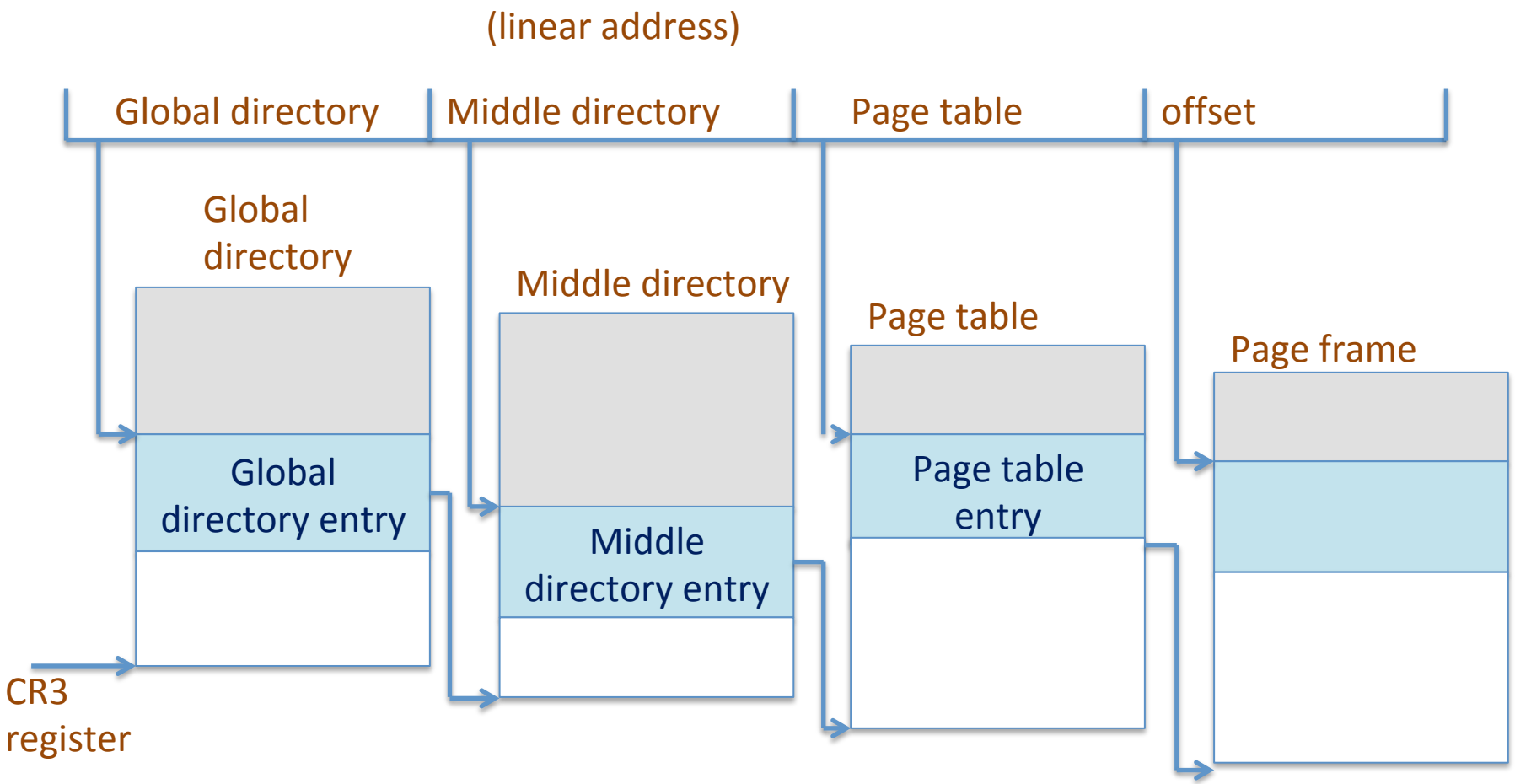
Pentium Paging Architecture



Linear Address in Linux

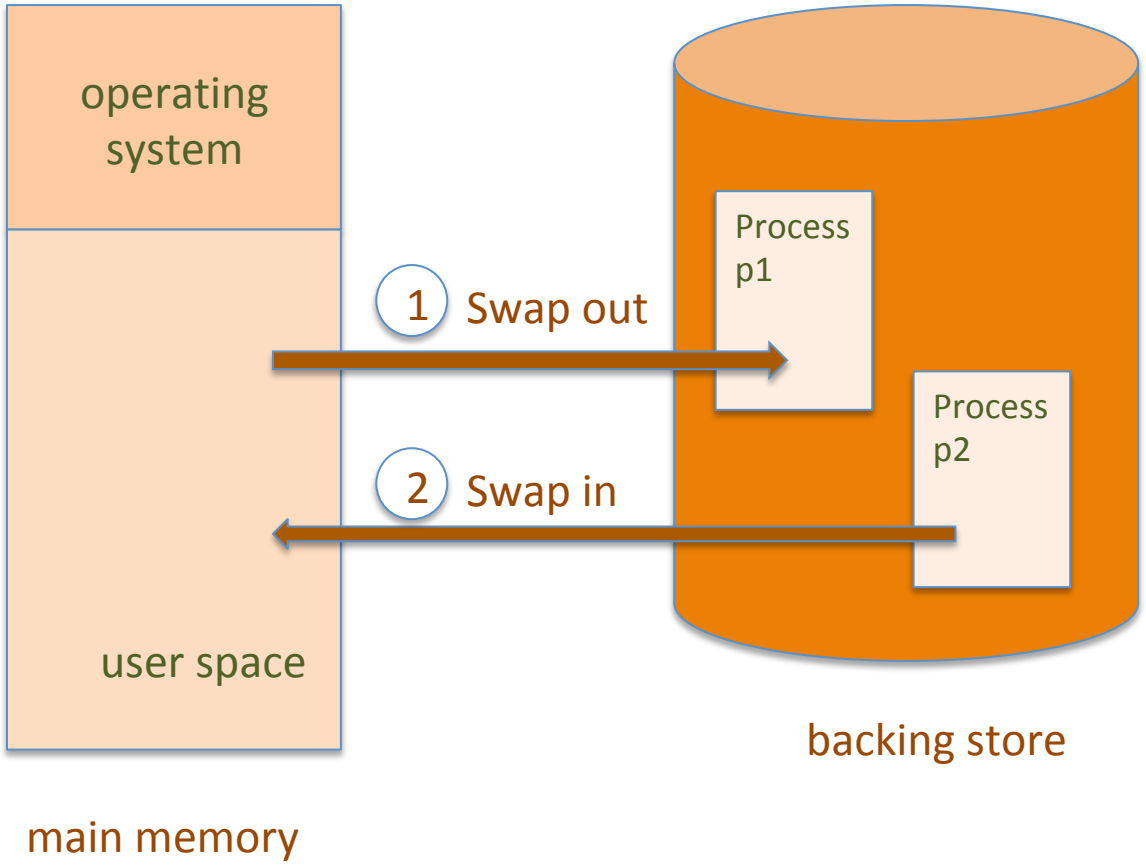
Broken into four parts:

Three-level Paging in Linux



Swapping

- A process can be swapped temporarily out of memory to a backing store, and then brought back into memory for continued execution.



Acknowledgments

- These slides are adapted from
  - Öznur Özkasap (Koç University)
  - Operating System and Concepts (9th edition) Wiley