Memory Management

Operating Systems (ECEg-4181)

Mequanent Argaw Muluneh

Wednesday, April 29, 2020

Outline

- Swapping
- Contiguous Memory Allocation
- Segmentation
- Paging
- Structure of the Page Table

Objectives

- To provide a detailed description of various ways of organizing memory hardware.
- To explore various techniques of allocating memory to processes.
- To discuss in detail how paging works in contemporary computer systems.

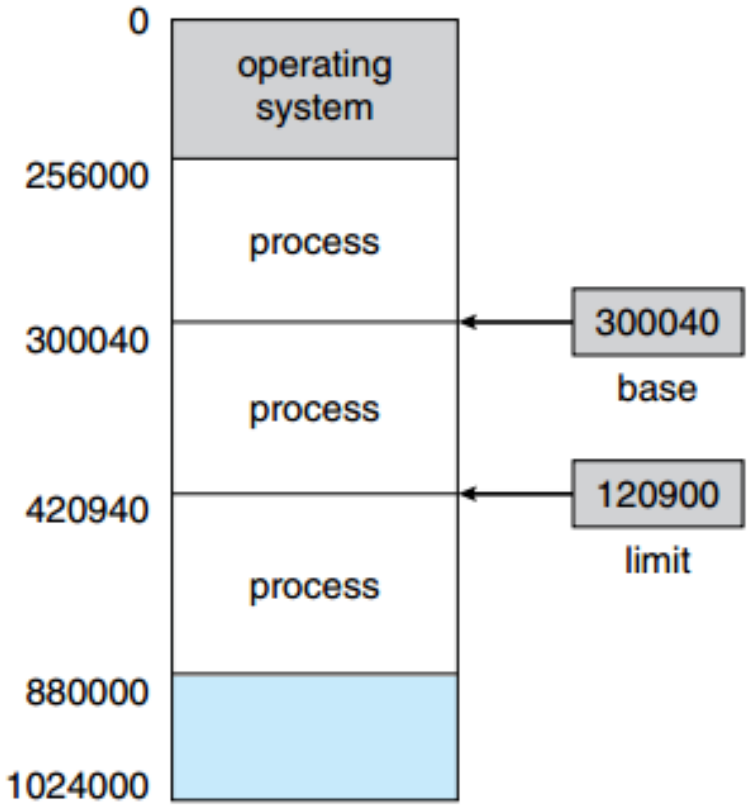
Background

- Program must be brought (from disk) into memory and placed within a process for it to be run.
- Main memory and registers are the only storages CPU can access directly.
- Memory unit only sees a stream of addresses + read requests, or address + data and write requests.
- Register access is done in one CPU clock cycle (or less).
- Main memory can take many cycles, causing a stall.
- Cache sits between main memory and CPU registers.
- Protection of memory required to ensure correct operation.

Background ...

Base and Limit Registers

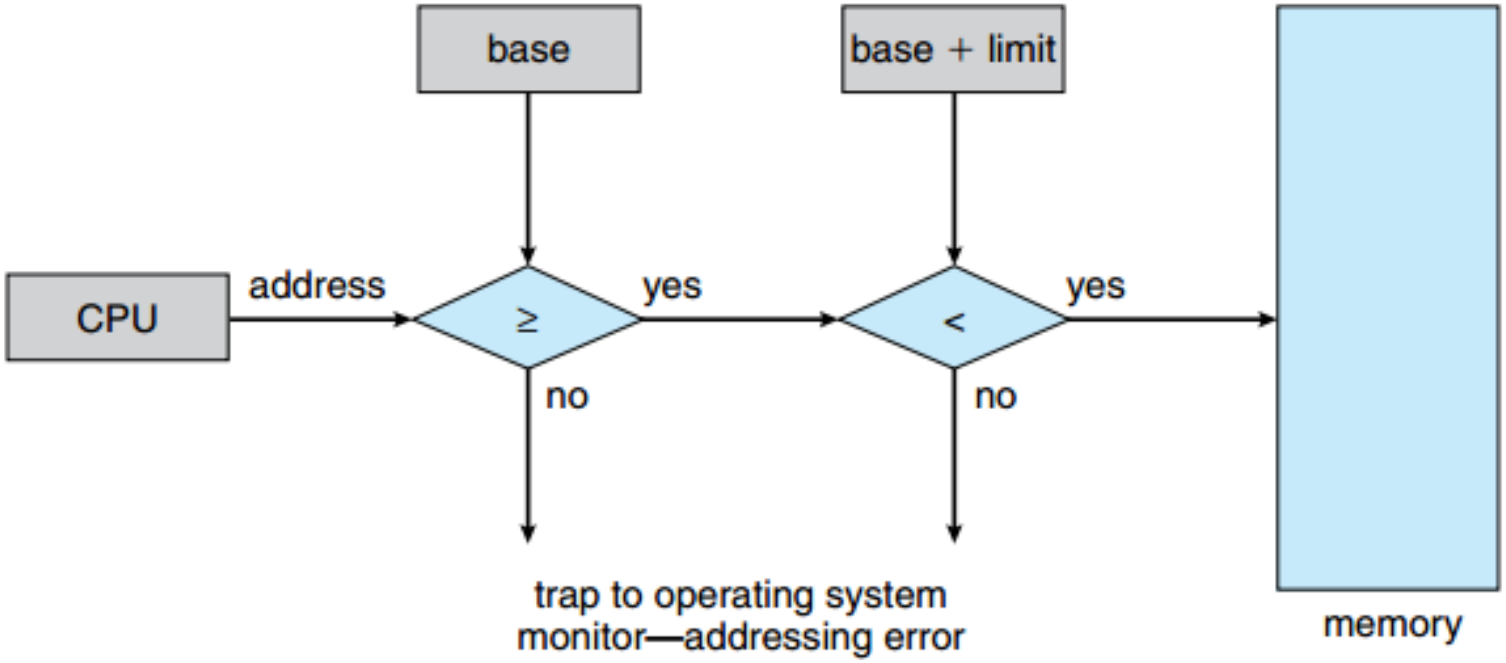
- A pair of base and limit registers define the logical address space.
- CPU must check every memory access generated in user mode to be sure it is between base and limit for that user.



Background ...

Hardware Address Protection

- Hardware address protection with base and limit registers.



Background …

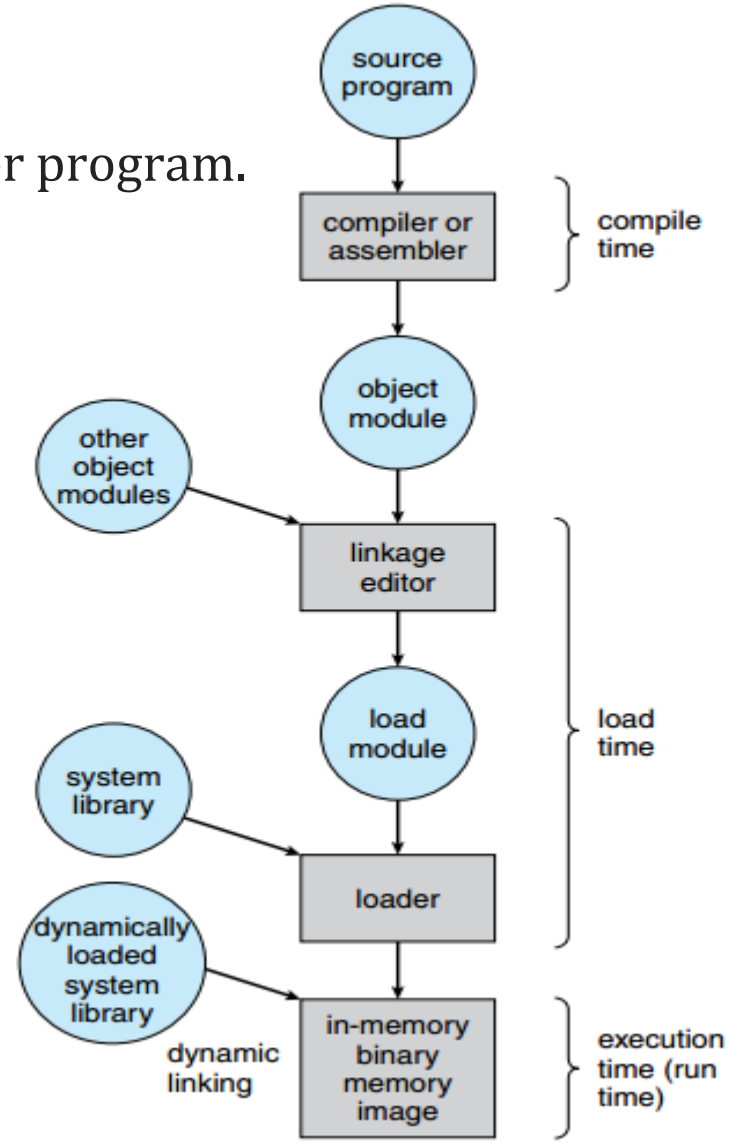
Address Binding

- Programs on disk which are ready to be brought into memory to execute form an **input queue**.
- Most systems allow a user process to reside in any part of the physical memory.
  - Thus, although the address space of the computer may start at 00000, the first address of the user process need not be 00000.
- Addresses may be represented in different ways while a user program goes through several steps before being executed (*see the figure in the next slide*).
- Addresses in the source program are generally symbolic (such as `count`).
- A compiler typically binds these symbolic addresses to relocatable addresses (such as "14 bytes from the beginning of this module").
- The linkage editor or loader in turn binds the relocatable addresses to absolute addresses (such as 74014).
- Each binding is a mapping from one address space to another.

Background ...

Address Binding ...

- Multistep processing of a user program.



Background ...

Address Binding ...

- Address binding of instructions and data to memory addresses can happen at three different stages.
  - **Compile time.** If memory location of a process is known at compile time, **absolute code** can be generated; we must recompile the code if starting location changes.
  - **Load time. Relocatable code** must be generated if memory location is not known at compile time.
  - **Execution time.** Binding delayed until run time if the process can be moved during its execution from one memory segment to another.
    - Needs hardware support for address maps (e.g., base and limit registers)

Background …

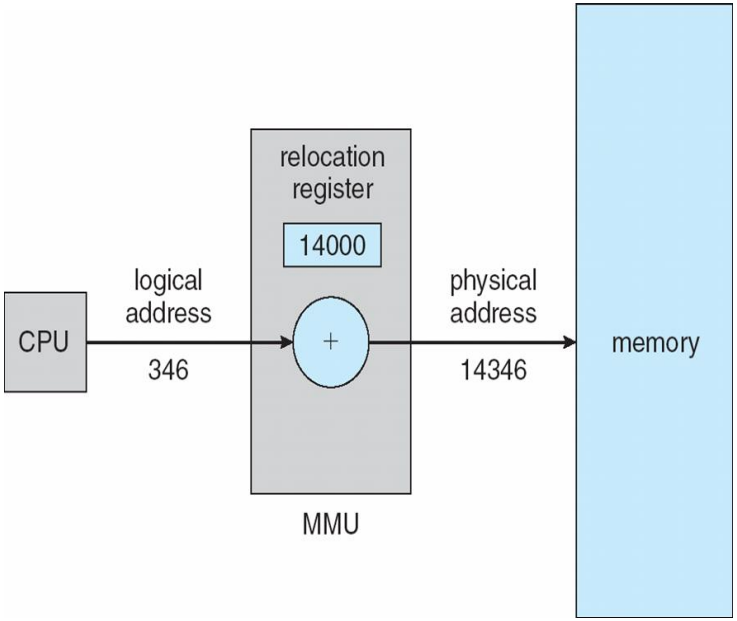
Logical vs. Physical Address Space

- The concept of a logical address space that is bound to a separate **physical address space** is central to proper memory management.
  - **Logical address (virtual address)** – is generated by the CPU.
  - **Physical address** – address seen by the memory unit.
- Logical and physical addresses are the same in compile-time and load-time address-binding schemes; logical (virtual) and physical addresses differ in execution-time address-binding scheme.
- **Logical address space** is the set of all logical addresses.
- **Physical address space** is the set of all physical addresses.

Background ...

Memory-Management Unit (MMU)

- MMU is a hardware device that maps logical address to physical address at run time.
- Consider a simple scheme where the value in the **relocation** (**base**) **register** is added to every address generated by a user process at the time it is sent to memory.
- The user program deals with *logical* addresses; it never sees the *real* physical addresses.
  - The memory-mapping hardware converts logical addresses into physical addresses.
  - Execution-time binding occurs when reference is made to location in memory.



Background ...

Dynamic Loading

- With dynamic loading a routine is not loaded until it is invoked.
- It has a better memory-space utilization since unused routine is never loaded.
- All routines are kept on disk in a relocatable load format.
- It is useful when large amounts of code are needed to handle infrequently occurring cases.
- Dynamic loading requires no special support from the operating system.
  - It is implemented through program design by the programmers.
  - OS can help by providing libraries to implement dynamic loading.

Background ...

Dynamic Linking

- **Dynamically linked libraries:** are system libraries that are linked to user programs when the programs are *run*.
- **Static linking:** system libraries are treated like any other object module and are combined by the loader into the binary program image.
- A **stub** indicates how to locate the appropriate memory-resident library routine or how to load the library if the routine is not already present.
- The stub, a small piece of code, replaces itself with the address of the routine, and executes the routine.
- Thus, the next time that particular code segment is reached, the library routine is executed directly, incurring no cost for dynamic linking.
- Under this scheme, all processes that use a language library execute only one copy of the library code.

Background ...

Dynamic Linking ...

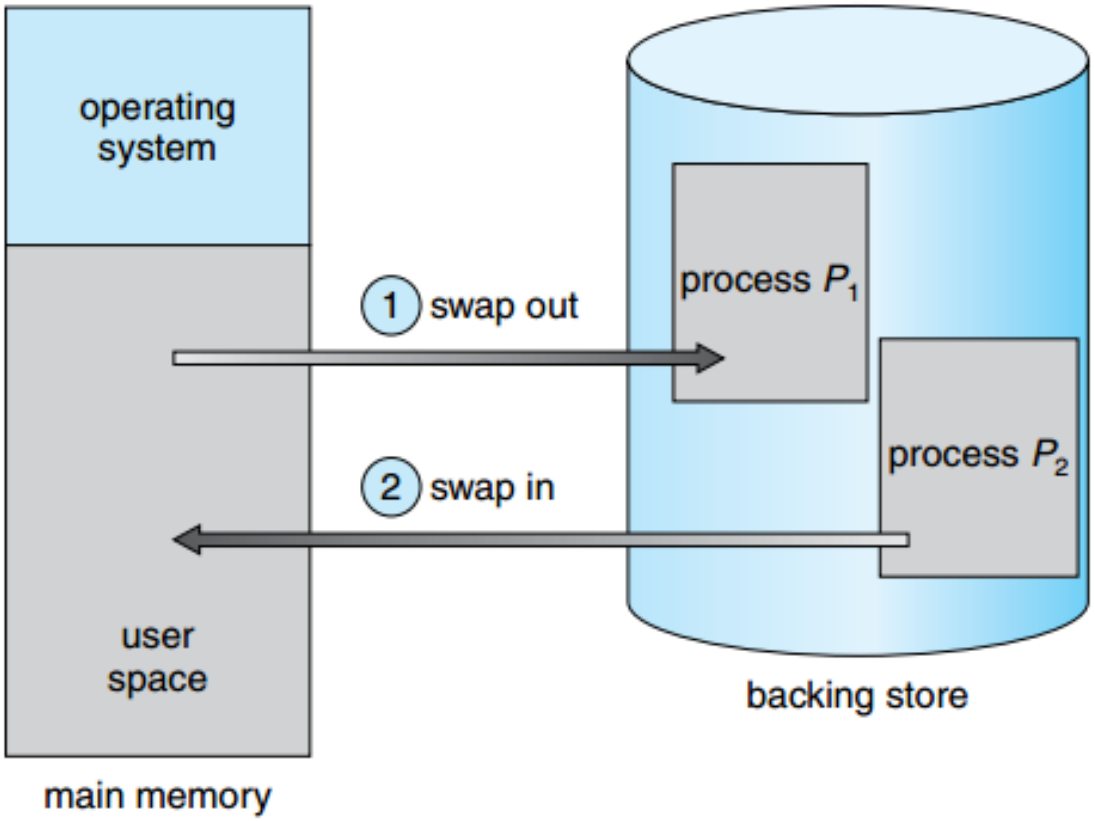
- This feature can be extended to library updates (such as bug fixes).
- A library may be replaced by a new version, and all programs that reference the library will automatically use the new version.
- Without dynamic linking, all such programs would need to be relinked to gain access to the new library.
- So, programs will not accidentally execute new, incompatible versions of libraries, version information is included in both the program and the library.
- More than one version of a library may be loaded into memory, and each program uses its version information to decide which copy of the library to use.
- Programs linked before the new library was installed will continue using the older library. This system is also known as **shared libraries**.

Swapping

- A process can be **swapped** temporarily out of memory to a backing store, and then brought back into memory for continued execution.
  - By swapping, total physical memory space of processes can exceed physical memory which increases degree of multiprogramming.
- **Backing store** is a fast disk which must be large enough to accommodate copies of all memory images for all users; and it must provide direct access to these memory images.
- The system maintains a **ready queue** consisting of all ready processes whose memory images are on the backing store or in memory.
- **Roll out, roll in** – is a swapping variant used for priority-based scheduling algorithms; lower-priority process is swapped out so higher-priority process can be loaded and executed.
- Major part of swap time is transfer time; total transfer time is directly proportional to the amount of memory swapped.

Swapping …

- Schematic View of Swapping



Swapping …

- Standard swapping is not used in modern operating systems.
- It requires too much swapping time and provides too little execution time to be a reasonable memory-management solution.
- Modified versions of swapping, however, are found on many systems, including UNIX, Linux, and Windows.
  - In one common variation, swapping is normally disabled but will start if the amount of free memory falls bellow a threshold amount.
    - Swapping is halted when the amount of free memory increases.
  - Another variation involves swapping portions of processes—rather than entire processes—to decrease swap time.

Swapping …

Context Switch Time including Swapping

- If next processes to be put on CPU is not in memory, need to swap out a process and swap in the target process.
- Context switch time can then be very high.
- Assume that a user process is 100 MB in size and the backing store is a standard hard disk with a transfer rate of 50 MB/second.
  - The actual transfer of the 100-MB process to or from main memory takes 100 MB/50 MB per second = 2 seconds
  - Swap out time of 2 sec + swap in time of 2 sec = 4 seconds.
  - The total context switch swapping component time is then 4 seconds.

Swapping ...

Context Switch Time including Swapping ...

- Swapping is constrained by other factors as well.
- If we want to swap a process, we must be sure that it is completely idle. Of particular concern is any pending I/O.
- A process may be waiting for an I/O operation when we want to swap that process to free up memory.
- A process pending for I/O operation cannot be swapped.

Contiguous Memory Allocation

- Main memory must accommodate both OS and user processes.
- Contiguous memory allocation is one of the early methods used to allocate memory in the most efficient way possible.
- Main memory is usually divided into two **partitions**:
  - One for the resident operating system and one for the user processes.
- The operating system is mostly placed nearest to the interrupt vector in the low memory.
- In contiguous memory allocation, each process is contained in a single section of memory that is contiguous to the section containing the next process.

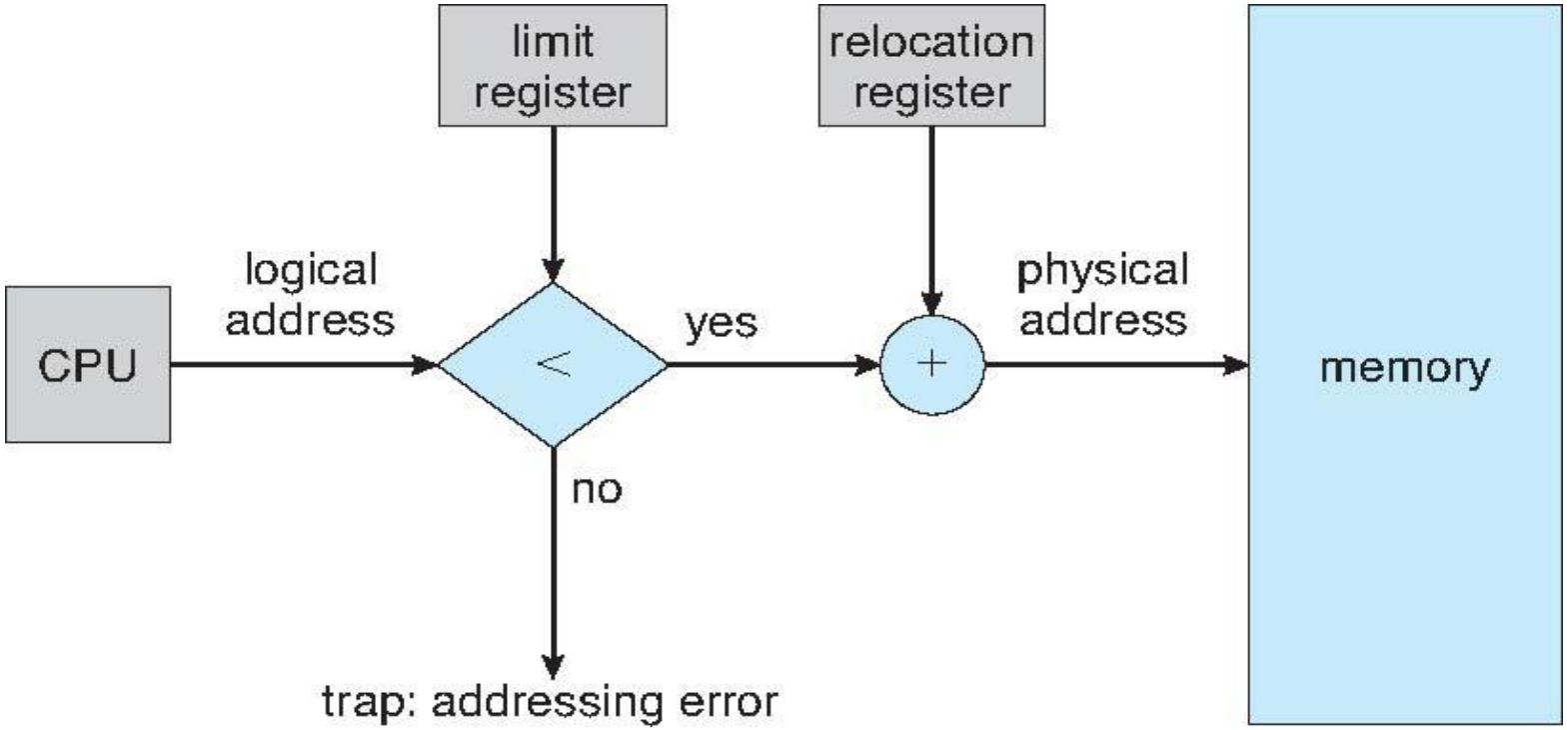
Contiguous Memory Allocation …

Memory Protection

- Relocation registers used to protect user processes from each other, and from changing operating-system code and data.
  - The relocation register (base register) contains value of the smallest physical address.
  - The limit register contains range of logical addresses – each logical address must be less than the value in the limit register.
  - MMU maps logical address *dynamically* by adding the value in the relocation register.

Contiguous Memory Allocation …

Memory Protection …



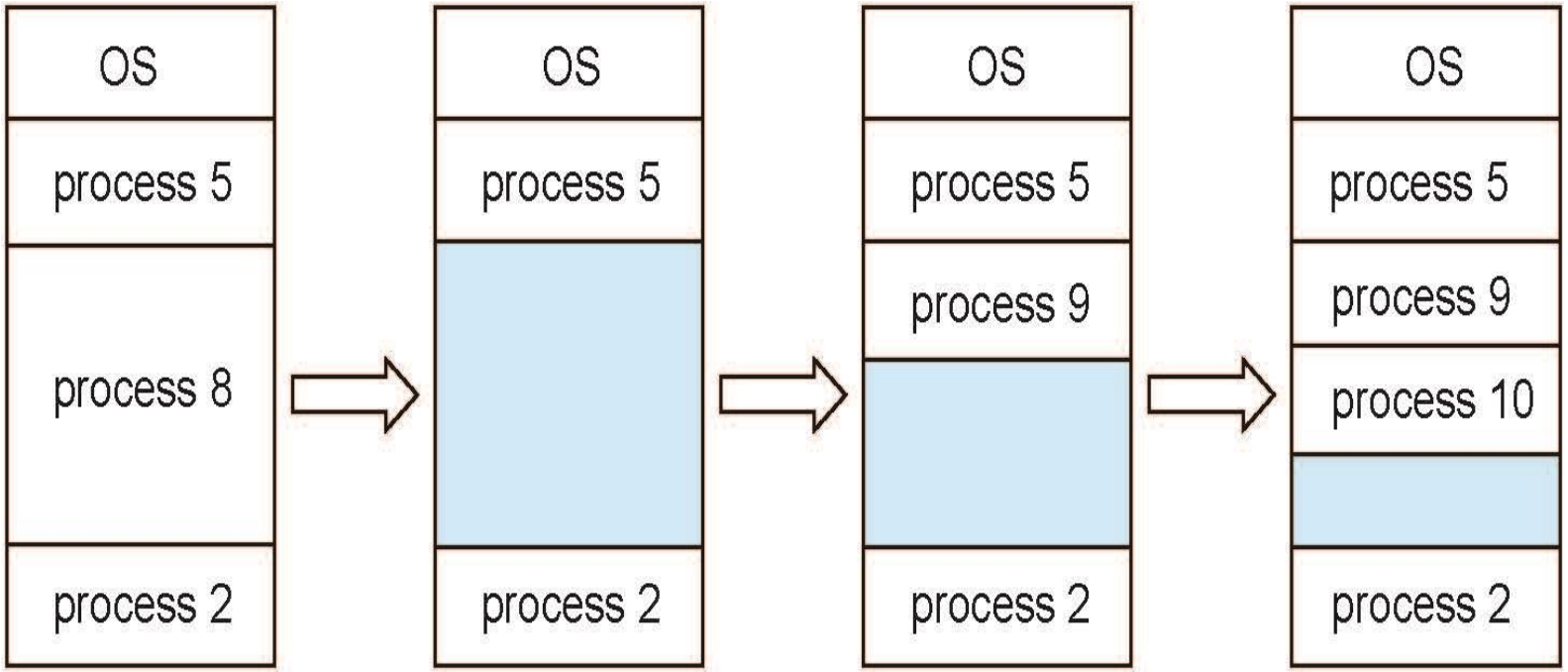
Hardware Support for Relocation and Limit Registers

Contiguous Memory Allocation …

Memory Allocation

- The simplest method for allocating memory is to divide it into several fixed-sized partitions.
- The degree of multiprogramming is bound by the number of partitions.
- In the variable-partition scheme, the operating system keeps a table indicating which parts of memory are available and which are occupied.
- Initially, all memory is available for user processes and is considered one large block of available memory, a hole.
  - When a process arrives, it is allocated with memory from a hole large enough to accommodate it.
  - Exiting process frees its partition and adjacent free partitions combined with the freed partition.
  - Operating system maintains information about:
    - a) allocated partitions b) free partitions (holes)

Contiguous Memory Allocation …



Contiguous Memory Allocation …

Dynamic Storage-Allocation Problem

- How to satisfy a request of size n from a list of free holes?
- **First-fit**. Allocate the ***first*** hole that is big enough.
- **Best-fit**. Allocate the ***smallest*** hole that is big enough; must search entire list, unless the list is ordered by size.
  - This strategy produces the smallest leftover hole.
- **Worst-fit**. Allocate the ***largest*** hole; must also search entire list unless the list is ordered by size.
  - This strategy produces the largest leftover hole.

Contiguous Memory Allocation …

Fragmentation

- Memory fragmentation can be internal as well as external.
- **Internal Fragmentation** – allocated memory may be slightly larger than requested memory; this size difference is unused memory internal to a partition.
- **External Fragmentation** – total memory space exists to satisfy a request, but it is not contiguous.
- Fragmentation can be a severe problem. E.g. first fit analysis reveals that given *N* blocks allocated, 0.5 *N* blocks lost to fragmentation.
  - 1/3 of memory may be unusable which is known as the **50-percent rule.**

Contiguous Memory Allocation …

Fragmentation …

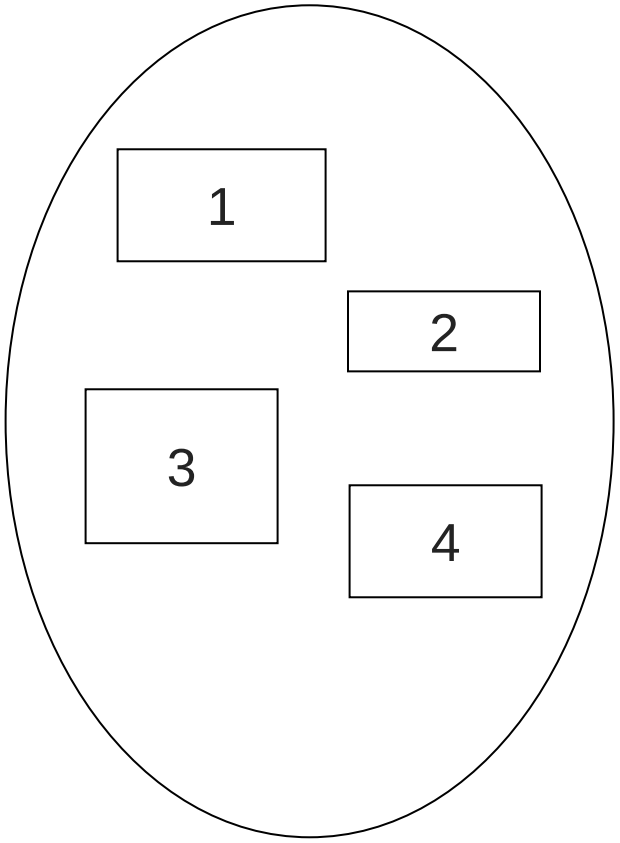
- One solution to the problem of external fragmentation is **compaction.**
  - The goal is to shuffle the memory contents so as to place all free memory together in one large block.
  - Compaction is possible *only* if relocation is dynamic, and is done at execution time.
- Another possible solution to the external-fragmentation problem is to permit the logical address space of the processes to be noncontiguous.
  - Allow a process to be allocated physical memory wherever such memory is available.
  - Two complementary techniques achieve this solution: segmentation and paging.

Segmentation

- The user's or programmer's view of memory is not the same as the actual physical memory.
- What if the hardware could provide a memory mechanism that mapped the programmer's view to the actual physical memory?
  - The system would have more freedom to manage memory, while the programmer would have a more natural programming environment.
  - Segmentation provides such a mechanism.
- A logical address is a collection of segments. Each segment has a name and a length.

Segmentation …

Logical View of Segmentation



user space

physical memory space

Segmentation ...

Segmentation Hardware

- For simplicity of implementation, segments are numbered and are referred to by a segment number, rather than by a segment name.
- Thus, a logical address consists of a two tuple:

  <segment-number, offset>

- Although the programmer can now refer to objects in the program by a two-dimensional address, the actual physical memory is still, of course, a one-dimensional sequence of bytes.
- Thus, we must define an implementation to map two-dimensional user-defined addresses into one-dimensional physical addresses.

Segmentation ...

Segmentation Hardware ...

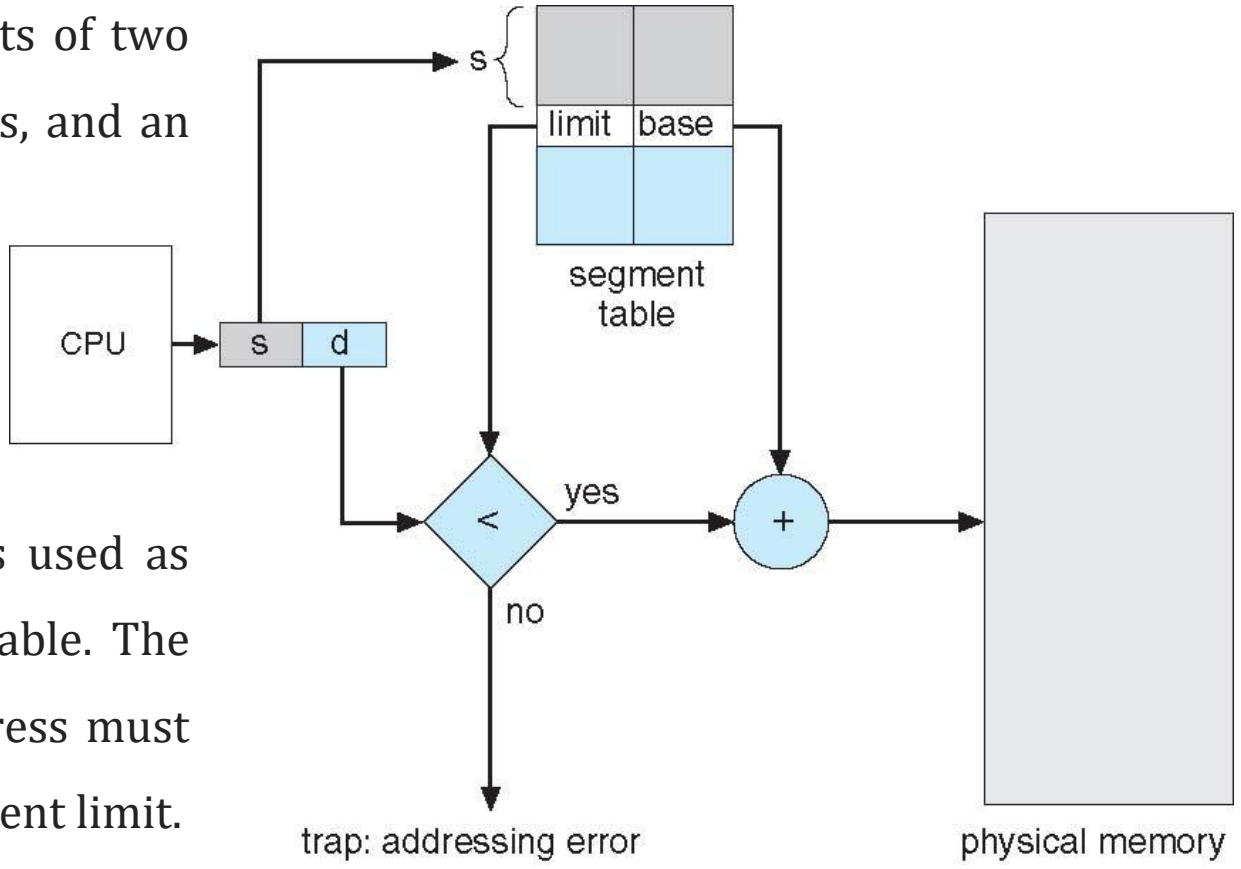
- This mapping is effected by a **segment table**.
- Each entry in the segment table has a **segment base** and a **segment limit**.
- The segment base contains the starting physical address where the segment resides in memory.
- The segment limit specifies the length of the segment.
- The use of a segment table is illustrated in the figure shown in the next slide.

Segmentation ...

Segmentation Hardware ...

❖ A logical address consists of two parts: a segment number, s, and an offset into that segment, d.

❖ The segment number is used as an index to the segment table. The offset d of the logical address must be between 0 and the segment limit.



Paging

- Segmentation permits noncontiguous physical address space for a process.
- **Paging** is another memory-management scheme that offers this advantage.
- However, paging avoids external fragmentation and the need for compaction, whereas segmentation does not.
- It also solves the considerable problem of fitting memory chunks of varying sizes onto the backing store.
- Paging is implemented through cooperation between the operating system and the computer hardware.

Paging …

Basic Method

- The basic method for implementing paging involves:
  - breaking physical memory into fixed-sized blocks, **frames** and
  - breaking logical memory into blocks of the same size, **pages**.
- When a process is to be executed, its pages are loaded into any available memory frames from their source.
- The backing store is divided into fixed-sized blocks that are the same size as the memory frames or clusters of multiple frames.

Paging …

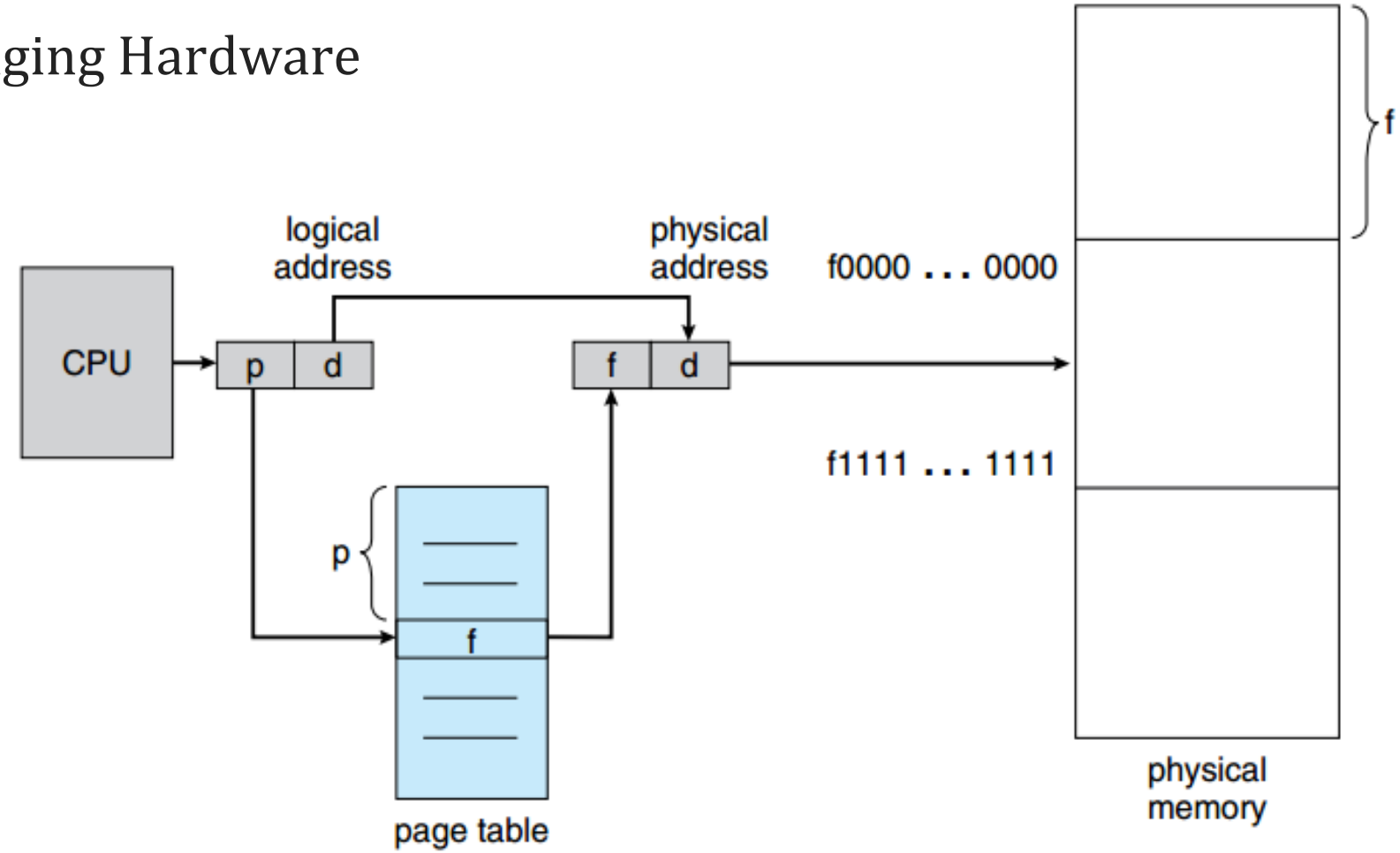
Basic Method …

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

Paging ...

Basic Method ...

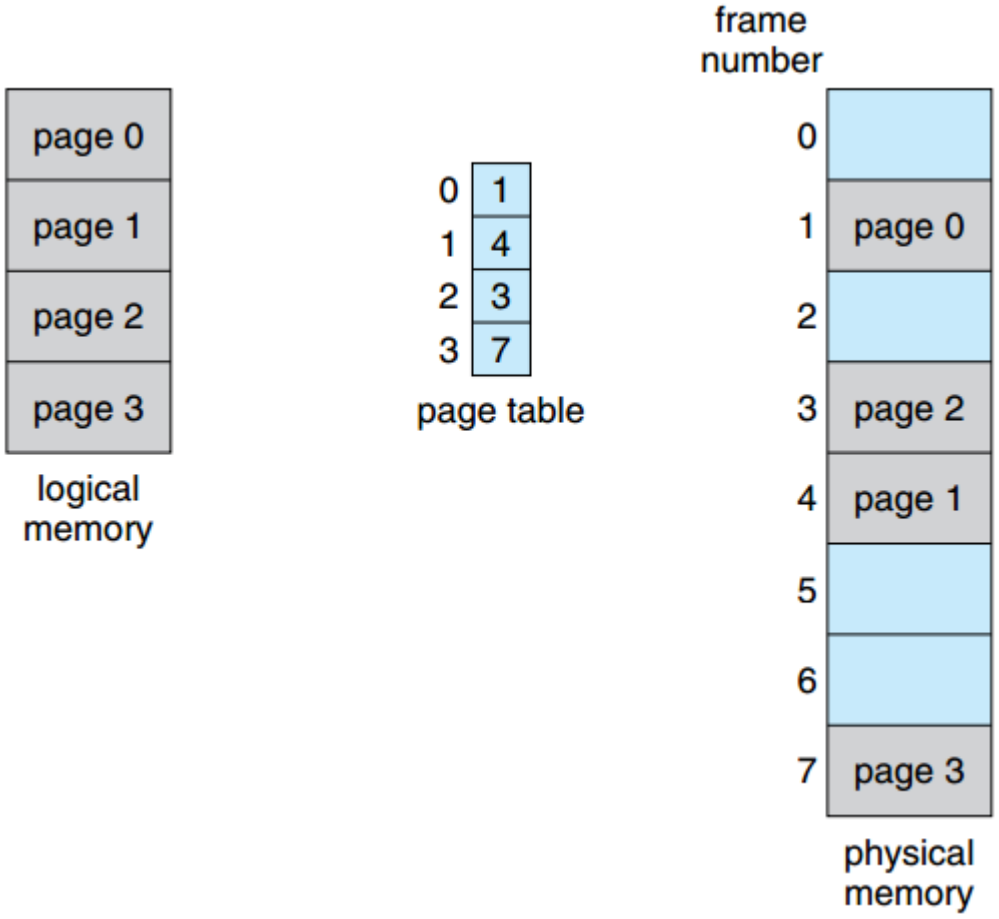
- Paging Hardware



Paging …

Basic Method …

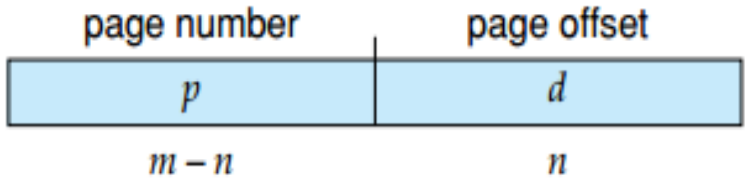
- Paging Model of Logical and Physical Memory



Paging …

Basic Method …

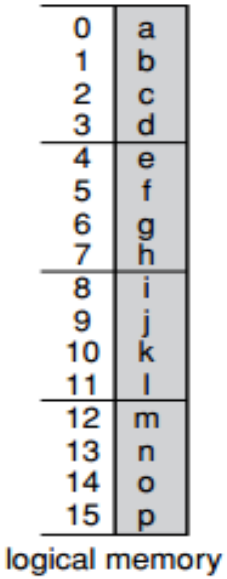
- The page size (like the frame size) is defined by the hardware.
- The size of a page is a power of 2, varying between 512 bytes and 1 GB per page, depending on the computer architecture.
- If the size of the logical address space is 2^m, and a page size is 2^n bytes, then the high-order m − n bits of a logical address designate the page number, and the n low-order bits designate the page offset.
- Thus, the logical address is as follows:

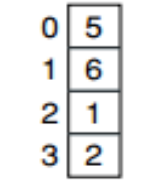


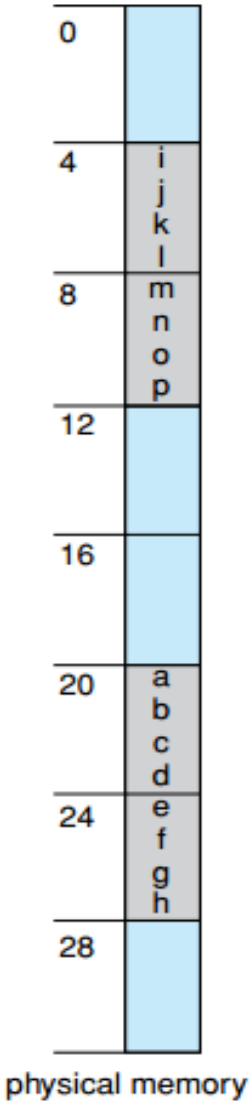
where p is an index into the page table and d is the displacement within the page.

Paging …

Basic Method …







❖ Paging example for a 32-byte memory with 4-byte pages.

n=2 and *m*=4 32-byte memory and 4-byte pages

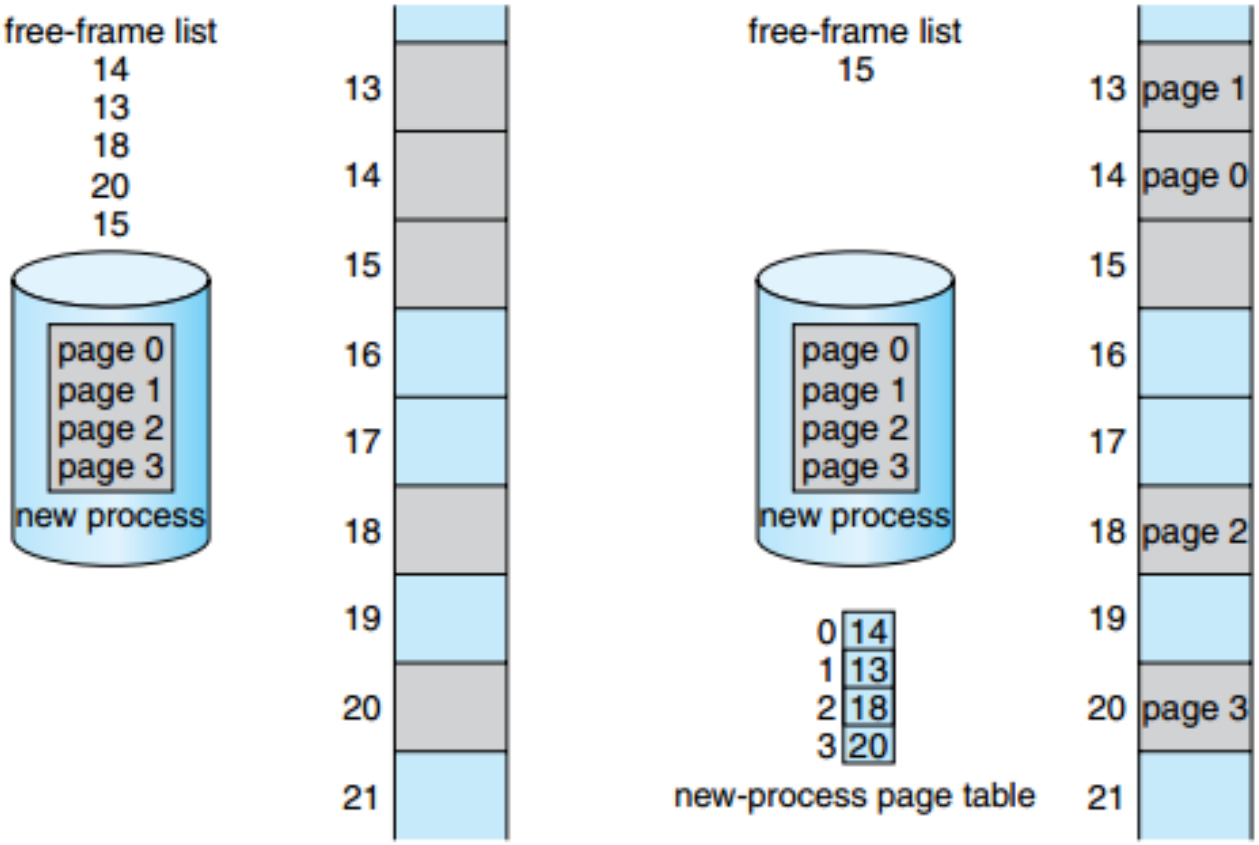
Paging …

Basic Method …

- Calculating internal fragmentation
  - Page size = 2,048 bytes
  - Process size = 72,766 bytes
  - 35 pages + 1,086 bytes
  - Internal fragmentation of 2,048 - 1,086 = 962 bytes
  - Worst case fragmentation = 1 frame for only 1 byte requirement
  - If process size is independent of page size, we expect internal fragmentation to average one-half page per process.
  - This consideration suggests that small page sizes are desirable.

Paging …

Basic Method …



❖ **Free Frames** (a) Before Allocation (b) After Allocation

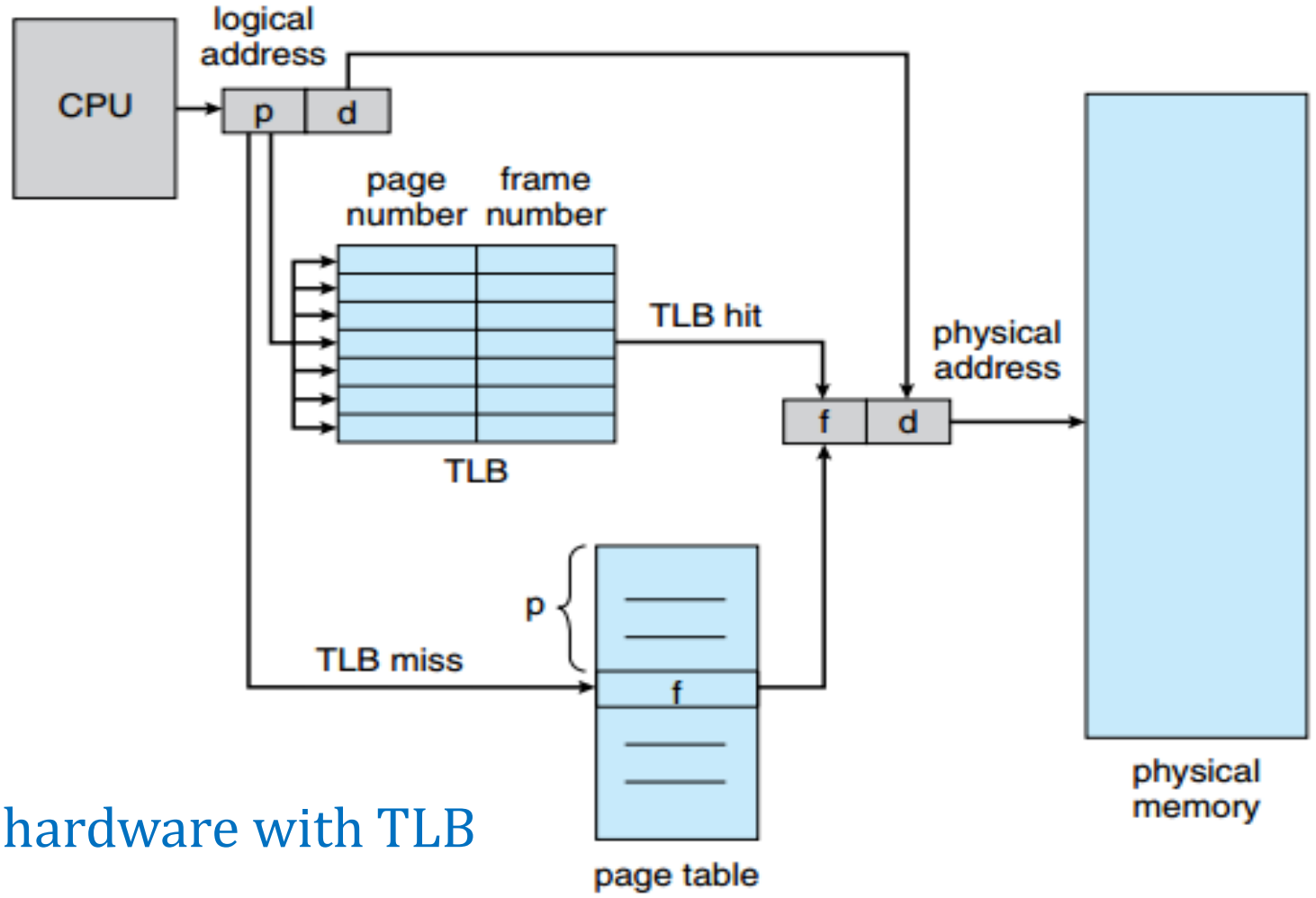
Paging …

Hardware Support

- 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.
- In this scheme every data/instruction access requires two memory accesses.
  - One for the page table and one for the data / instruction.
- 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**)

Paging …

Hardware Support …



Paging hardware with TLB

Paging …

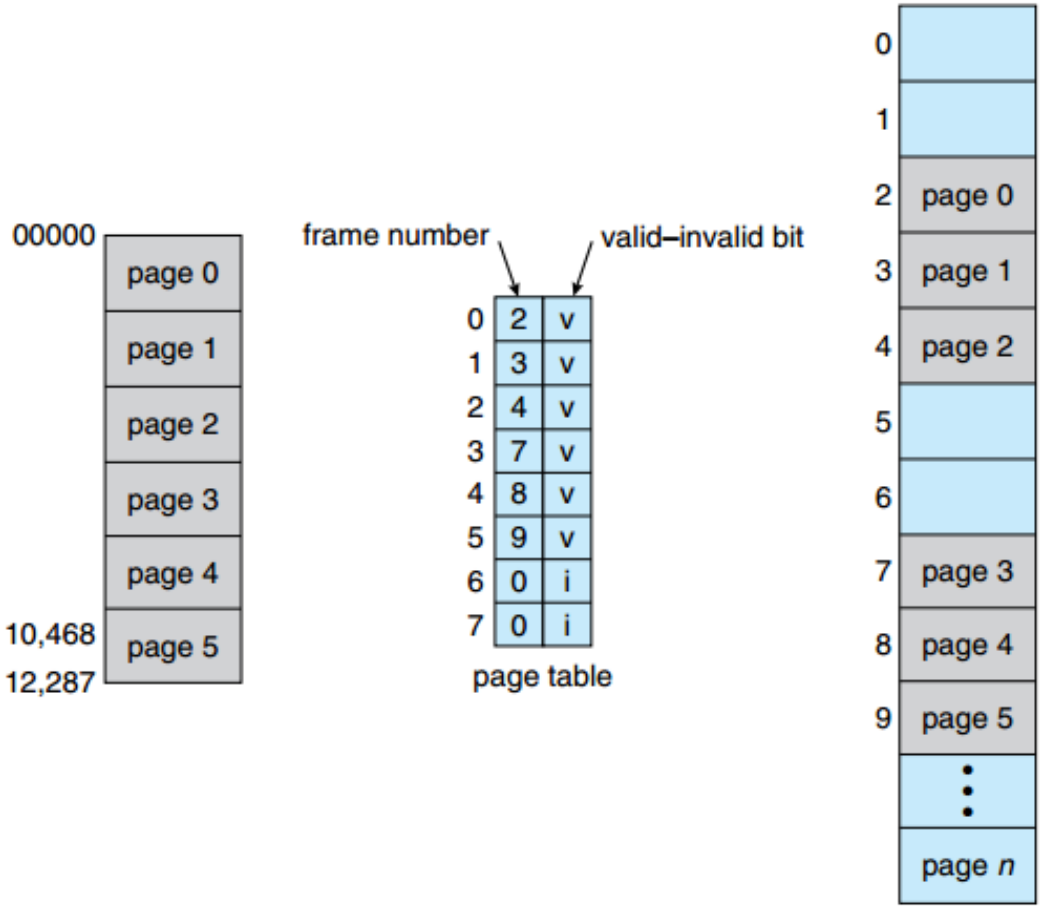
Memory Protection

- Memory protection is implemented by associating a protection bit with each frame to indicate if read-only or read-write access is allowed.
  - We can also add more bits to indicate page execute-only, and so on.
- **Valid-invalid** bit is 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.
- Any violations result in a trap to the kernel.

Paging …

Memory Protection …

❖ **Valid (v) or Invalid (i) Bit in a Page Table**



Structure of the Page Table

- Memory structures for paging can get huge using straight-forward methods.
  - Consider a 32-bit logical address space as on modern computers
  - Page size of 4 KB (2^{12})
  - Page table would have 1 million entries (2^{32} / 2^{12})
  - If each entry is 4 bytes -> 4 MB of physical address space / memory is required for page table alone.
    - That amount of memory used costs a lot.
    - We don't want to allocate that contiguously in main memory.
- Hierarchical Paging
- Hashed Page Tables
- Inverted Page Tables

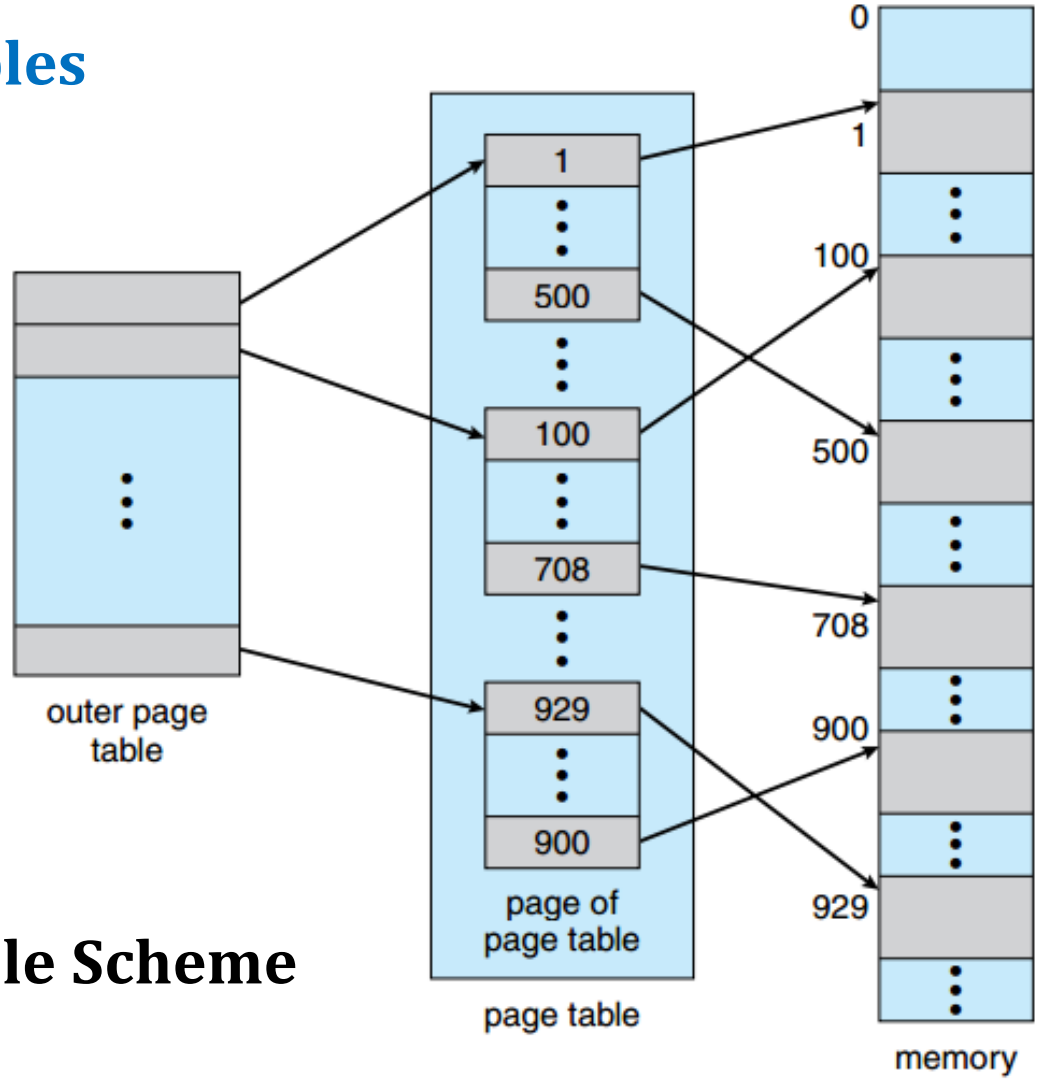
Structure of the Page Table …

Hierarchical Page Tables

- Break up the logical address space into multiple page tables.
- A simple technique is a two-level page table.
- We then page the page table.

Structure of the Page Table …

Hierarchical Page Tables



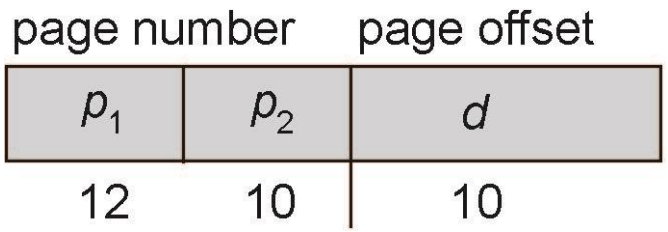
❖ **Two-Level Page-Table Scheme**

Structure of the Page Table …

Hierarchical Page Tables

❖ Two-Level Paging Example

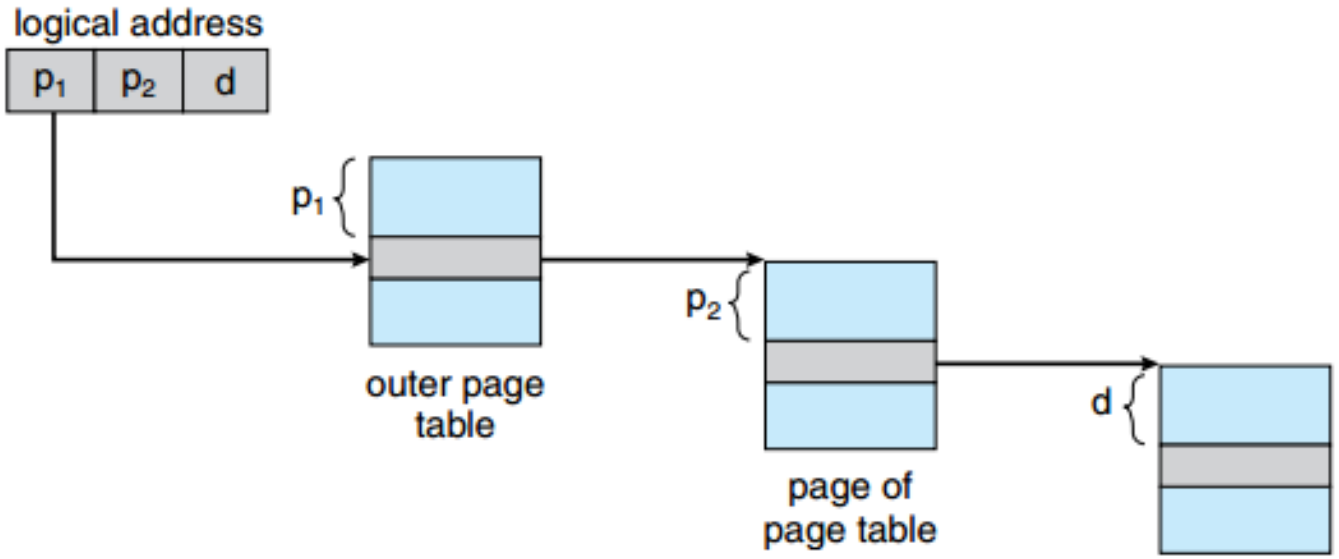
- ❖ A logical address (on 32-bit machine with 1K page size) is divided into:
  - ❖ a page number consisting of 22 bits
  - ❖ a page offset consisting of 10 bits
- ❖ Since the page table is paged, the page number is further divided into:
  - ❖ a 12-bit page number
  - ❖ a 10-bit page offset
- ❖ Thus, 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 inner page table
- ❖ Known as **forward-mapped page table**

Structure of the Page Table …

Address-Translation Scheme



Structure of the Page Table …

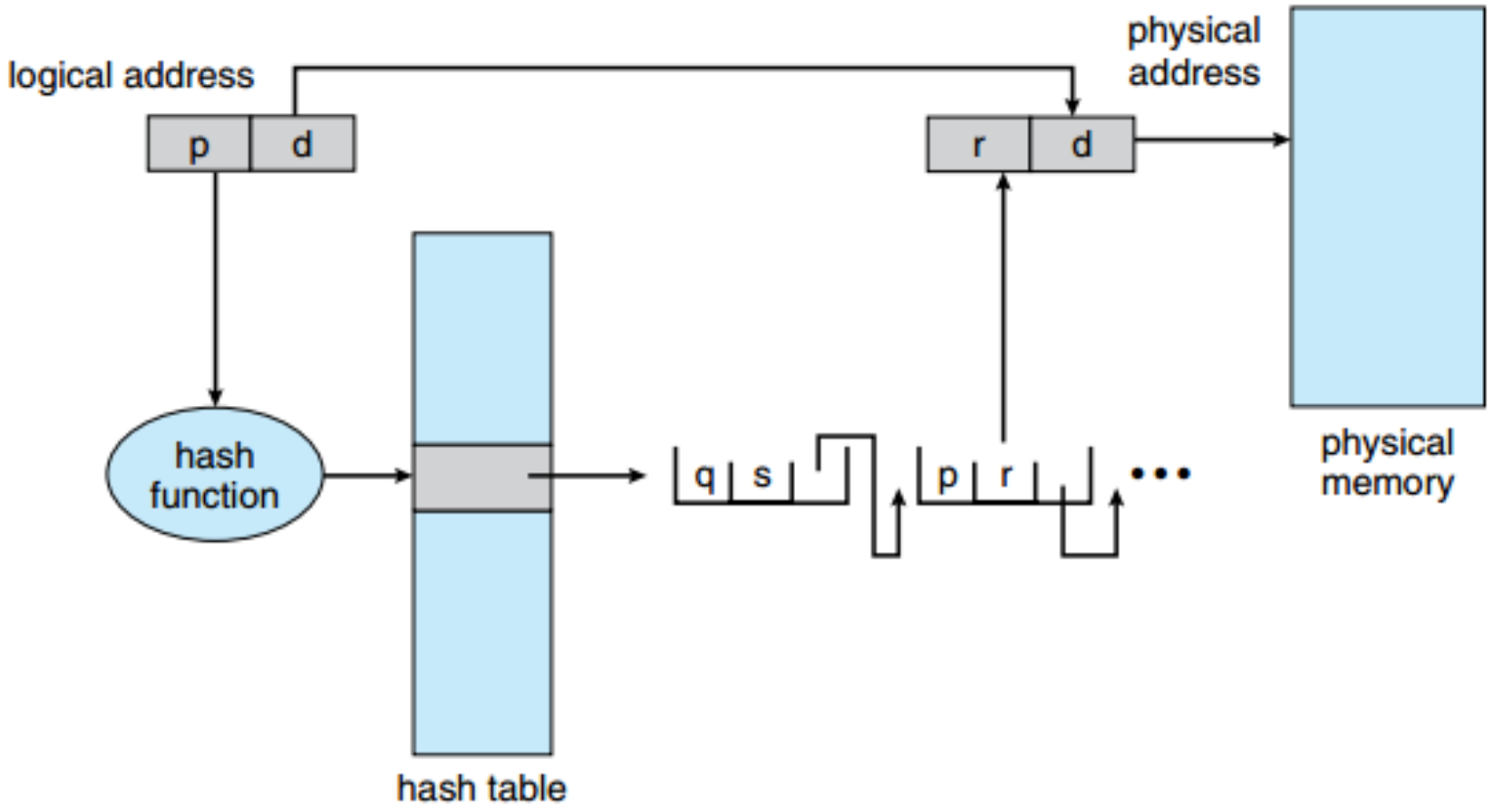
Hashed Page Tables

- Common in address spaces > 32 bits
- The virtual page number is hashed into a page table
  - This page table contains a chain of elements hashing to the same location
- Each element contains (1) the virtual page number (2) the value of the mapped page frame (3) a pointer to the next element
- Virtual page numbers are compared in this chain searching for a match
  - If a match is found, the corresponding physical frame is extracted

Structure of the Page Table …

Hashed Page Tables

❖



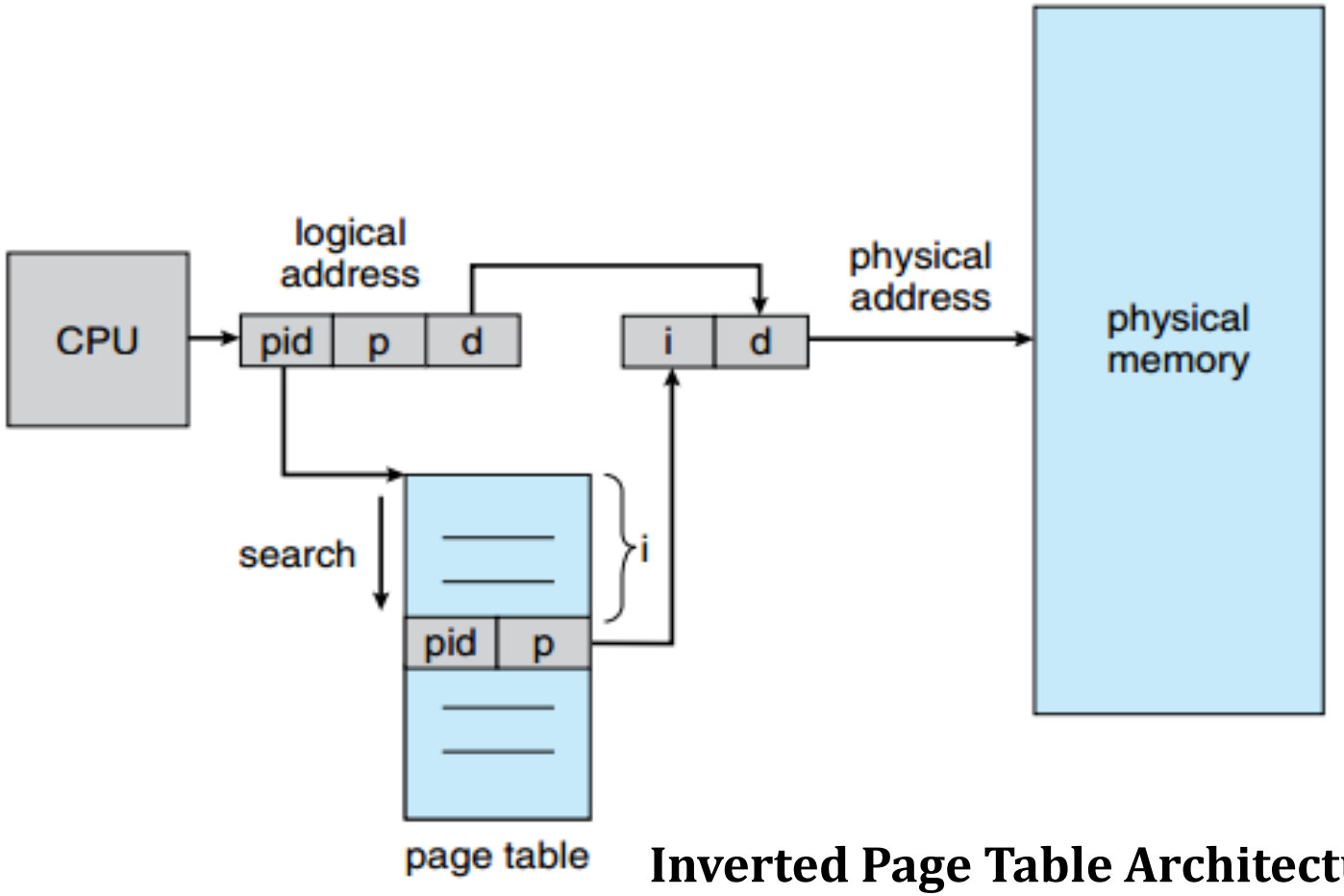
Structure of the Page Table ...

Inverted Page Table

- Rather than each process having a page table and keeping track of all possible logical pages, track all physical pages.
- One entry for each real page of memory.
- Entry consists of the virtual address of the page stored in that real memory location, 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.
  - TLB can accelerate access.
- But how to implement shared memory?
  - One mapping of a virtual address to the shared physical address.

Structure of the Page Table ...

Inverted Page Table ...



Inverted Page Table Architecture

Reference: Silberschatz et al., Operating System Concepts, Ninth Edition, 2013.

End of Chapter 6

Questions???