Outline

- Chapter 9: Memory management
  - Swapping
  - Contiguous Allocation
  - Paging
  - Segmentation
  - Segmentation with Paging
- Chapter 10: Virtual Memory
  - Demand Paging
  - Process Creation
  - Page Replacement
  - Allocation of Frames
  - Thrashing

Virtual Memory

- Address binding:
  - Compile time - must recompile if starting memory addr changes (MS Dos .com file)
  - Load time (relocatable)
  - Execution time: H/w support needed

- Logical and Physical address
  - Process sees logical addresses, processor sees physical addr
  - MMU for run time mapping of virtual to physical address

Dynamic loading

- Routine is not loaded until it is called
- Better memory-space utilization; unused routine is never loaded.
- Useful when large amounts of code are needed to handle infrequently occurring cases.
- No special support from the operating system is required implemented through program design

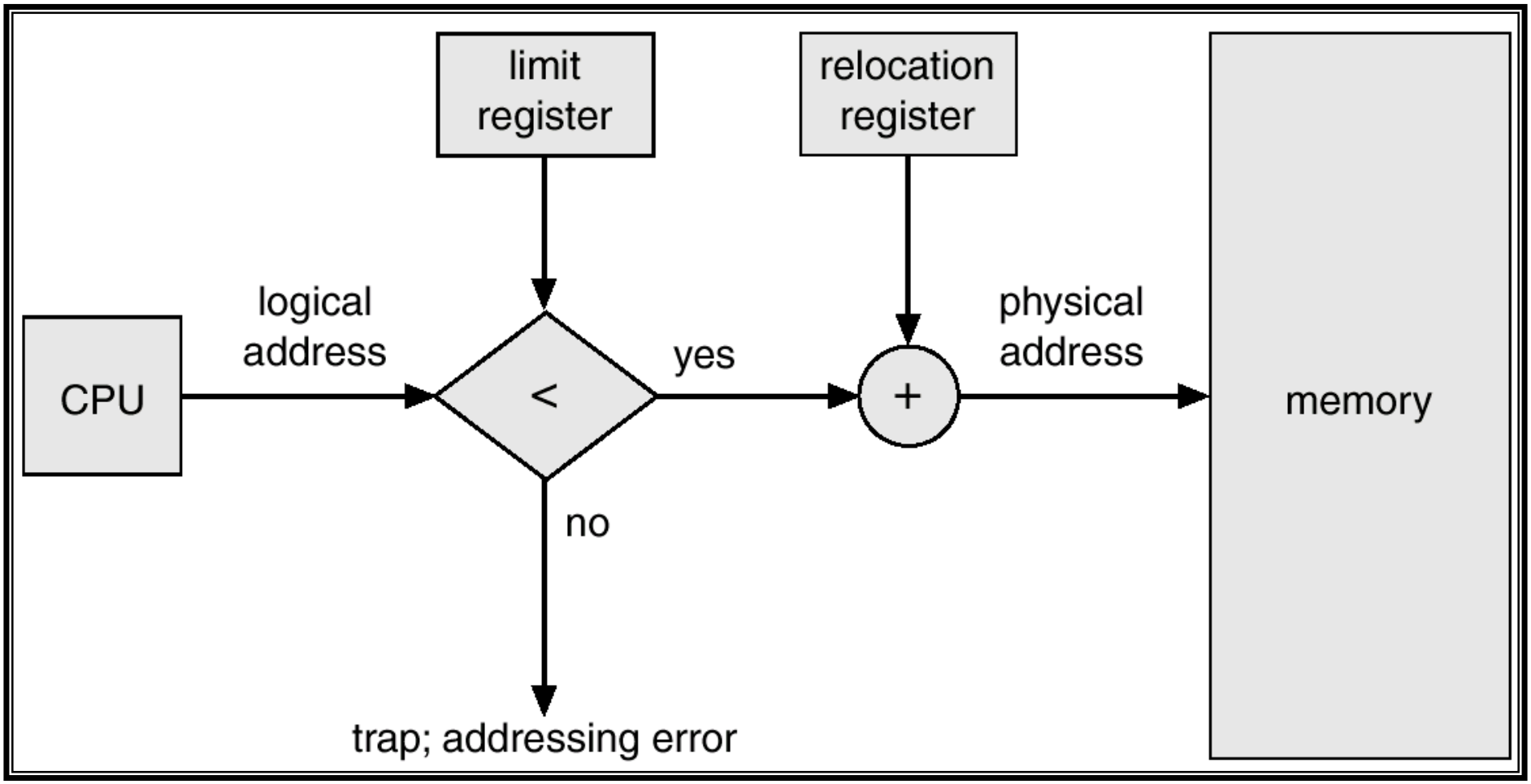
Dynamic linking

- Linking postponed until execution time.
- Small piece of code, *stub*, used to locate the appropriate memory-resident library routine.
- Stub replaces itself with the address of the routine, and executes the routine.
- Operating system needed to check if routine is in processes' memory address.
- Dynamic linking is particularly useful for libraries.

Swapping

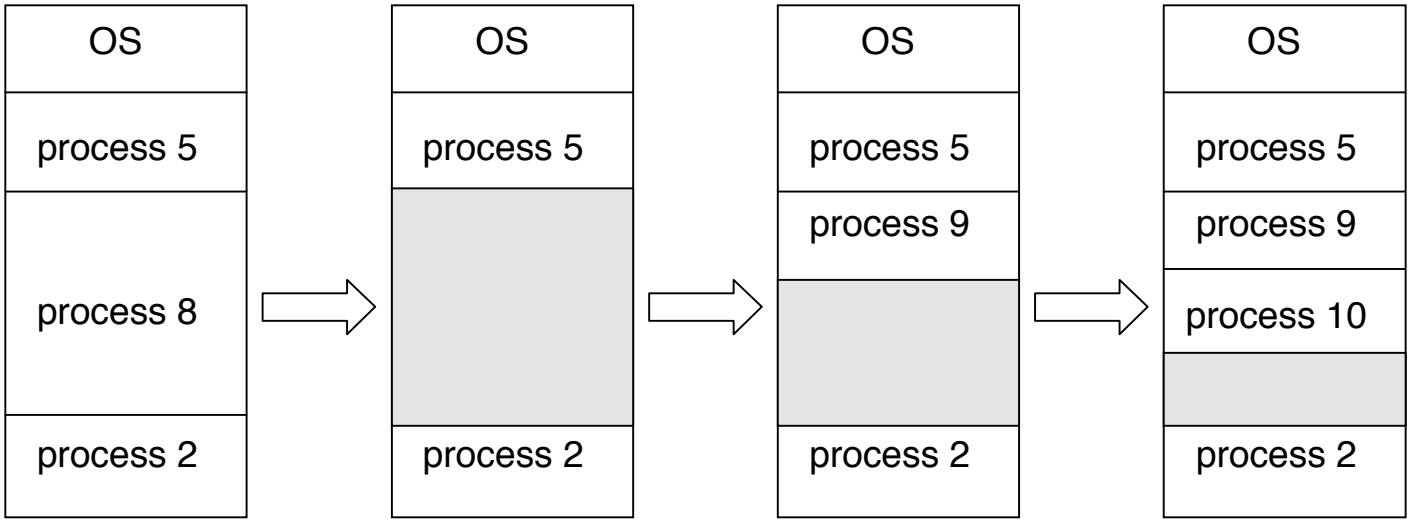
- A process can be *swapped* temporarily out of memory to a *backing store*, and then brought back into memory for continued execution
- Backing store – fast disk large enough to accommodate copies of all memory images for all users; must provide direct access to these memory images
- *Roll out, roll in* – 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
- Modified versions of swapping are found on many systems, i.e., UNIX, Linux, and Windows

Hardware support for relocation and limit reg.



Contigous allocation

- **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 ordered by size. Produces the smallest leftover hole.
- **Worst-fit**: Allocate the *largest* hole; must also search entire list. Produces the largest leftover hole.

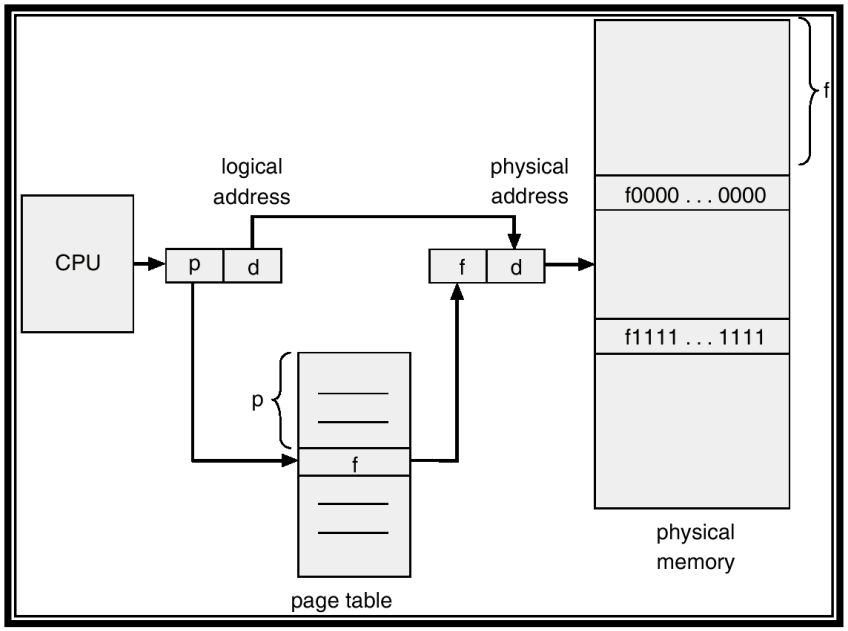


Fragmentation

- **External Fragmentation** – total memory space exists to satisfy a request, but it is not contiguous
- **Internal Fragmentation** – allocated memory may be slightly larger than requested memory; this size difference is memory internal to a partition, but not being used
- Reduce external fragmentation by compaction
  - Shuffle memory contents to place all free memory together in one large block
  - Compaction is possible *only* if relocation is dynamic, and is done at execution time
  - I/O problem
    - Latch job in memory while it is involved in I/O
    - Do I/O only into OS buffers

Paging

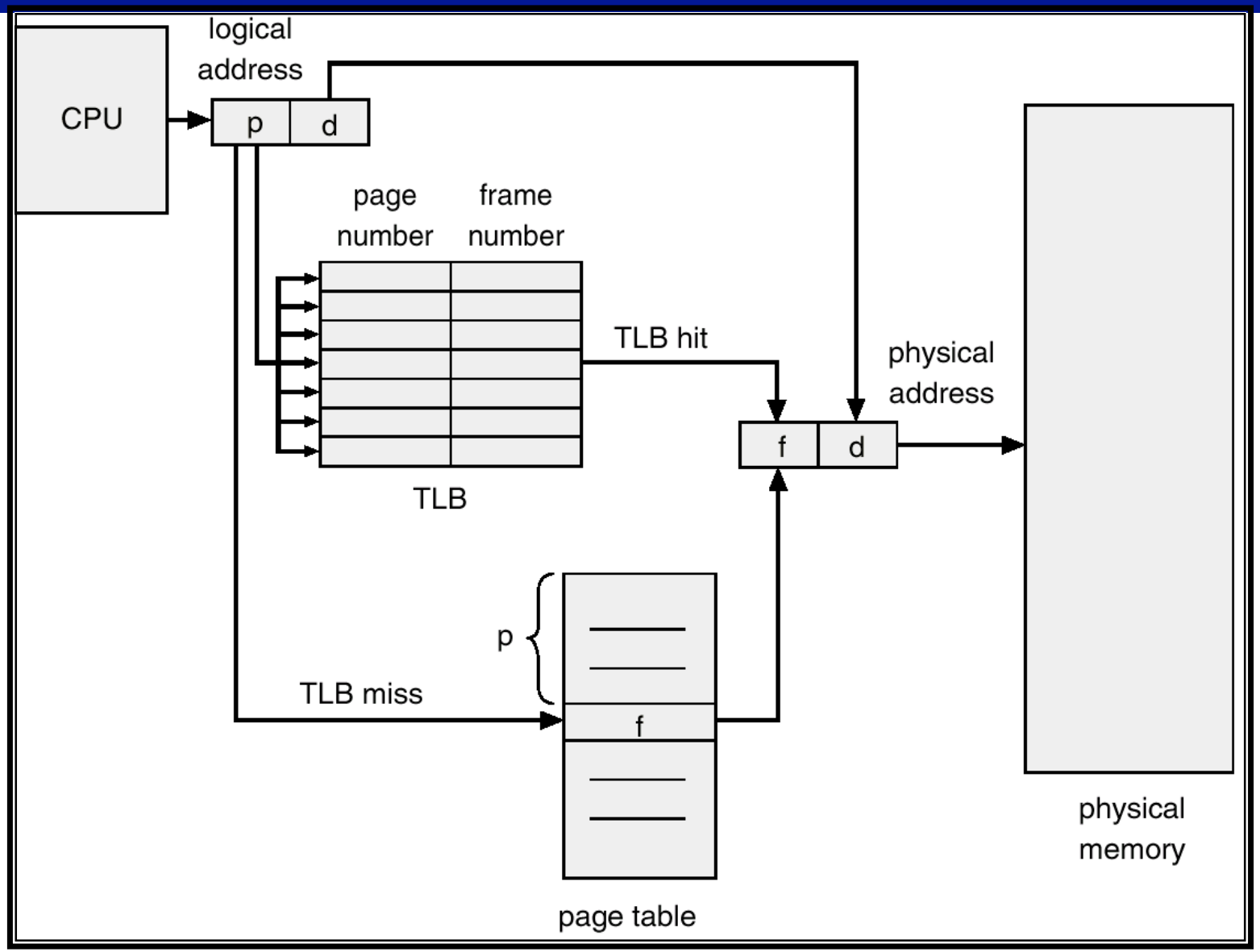
- Allows process physical page to be non-contiguous
  - Avoid fragmentation
  - Physical memory is broken into frames (h/w)
  - Logical memory is broken into pages (h/w)
  - Every address consists of page number and page offset. Page number is used as an index into page table



Page table implementation

- Page table is kept in main memory
- *Page-table base register (*PTBR) points to the page table
- *Page-table length register* (PRLR) 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)*

TLB


CPU
logical address
p
d
page number
frame number
TLB hit
physical address
f
d
TLB
p
TLB miss
f
page table
physical memory

Memory Protection

- Memory protection implemented by associating protection bit with each 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

Page table structure

- Hierarchical Paging
  - Break up the logical address space into multiple page tables
- Hashed Page Tables
  - The virtual page number is hashed into a page table. This 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
- Inverted Page Tables
  - 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

Segmentation

- Memory-management scheme that supports user view of memory.
- A program is a collection of segments. A segment is a logical unit such as:

  main program,

  procedure,

  function,

  method,

  object,

  local variables, global variables,

  common block,

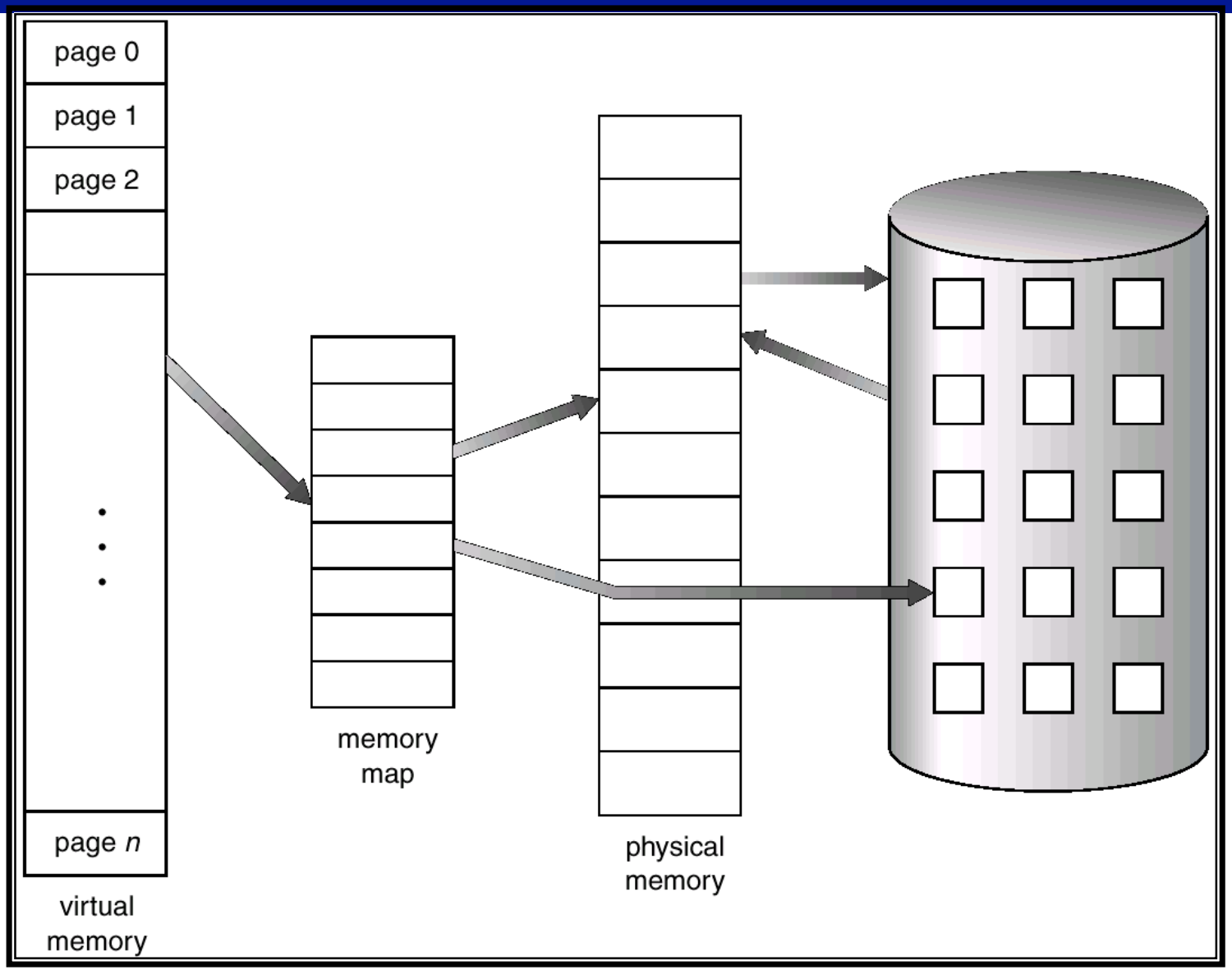
  stack,

  symbol table, arrays

Virtual memory

- separation of user logical memory from physical memory
  - Only part of the program needs to be in memory for execution
  - Logical address space can therefore be much larger than physical address space
  - Allows address spaces to be shared by several processes
  - Allows for more efficient process creation
- Virtual memory can be implemented via:
  - Demand paging
  - Demand segmentation

Virtual Memory



Demand paging

- Lazy swapper
  - Invalid pages generate page-fault trap
  - OS checks to see if it is indeed illegal
    - If valid, page it in
  - Once disk IO completely, schedule the process and resume execution
  - Performance implications

Process creation

- Copy-on-write
  - Fork model duplicates process. OS shares the same copy and marks the pages invalid for write. Duplicate when any process tries to modify a page (trap to the OS)

- Memory mapped files:
  - Virtual address space is logically associated with a file

- Page replacement:
  - Memory reference string
  - FIFO page replacement
    - Belady's anomaly

Optimal page replacement

- Replace the page that will not be used for the longest period of time
- Approximations
  - FIFO - Belady's anamoly
  - Least recently used (LRU)
    - Counters: Update every page access
    - Stack of page numbers used
    - Additional reference bit algorithm
    - Second chance algorithm
    - Enhanced second-chance algorithm:
      - Second chance+reference bit
  - Least Frequently used
  - Most Frequently used
  - Global vs local replacement
- Thrashing

Page rates

- Paging rates affected by code (compiler)
- Locking memory for IO completion