Operating Systems: Internals and Design Principles

Chapter 8
Virtual Memory

Eighth Edition
William Stallings

Hardware and Control Structures

- Two characteristics fundamental to memory management:
  1) all memory references are logical addresses that are dynamically translated into physical addresses at run time
  2) a process may be broken up into a number of pieces that don't need to be contiguously located in main memory during execution

- If these two characteristics are present, it is not necessary that all of the pages or segments of a process be in main memory during execution

Background

- **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

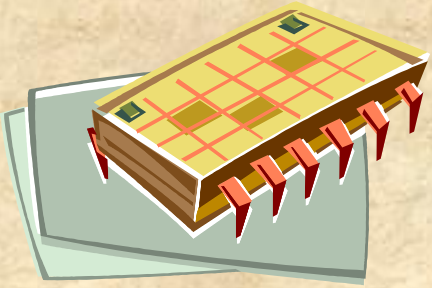
Virtual Memory vs. Physical Memory

Physical memory

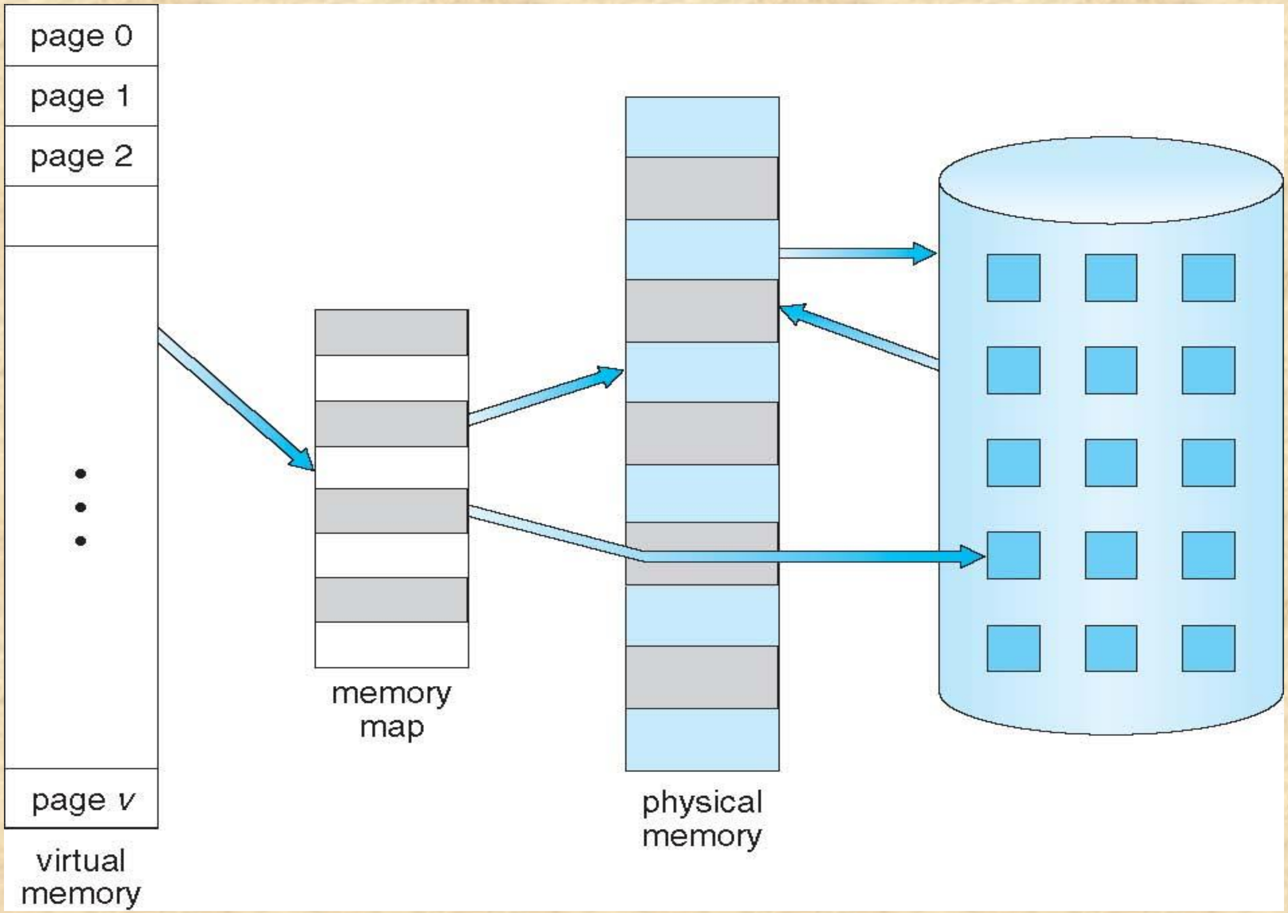
- main memory, the actual RAM

Virtual memory

- memory on disk
- allows for effective multiprogramming and relieves the user of tight constraints of main memory



Virtual Memory That is Larger Than Physical Memory

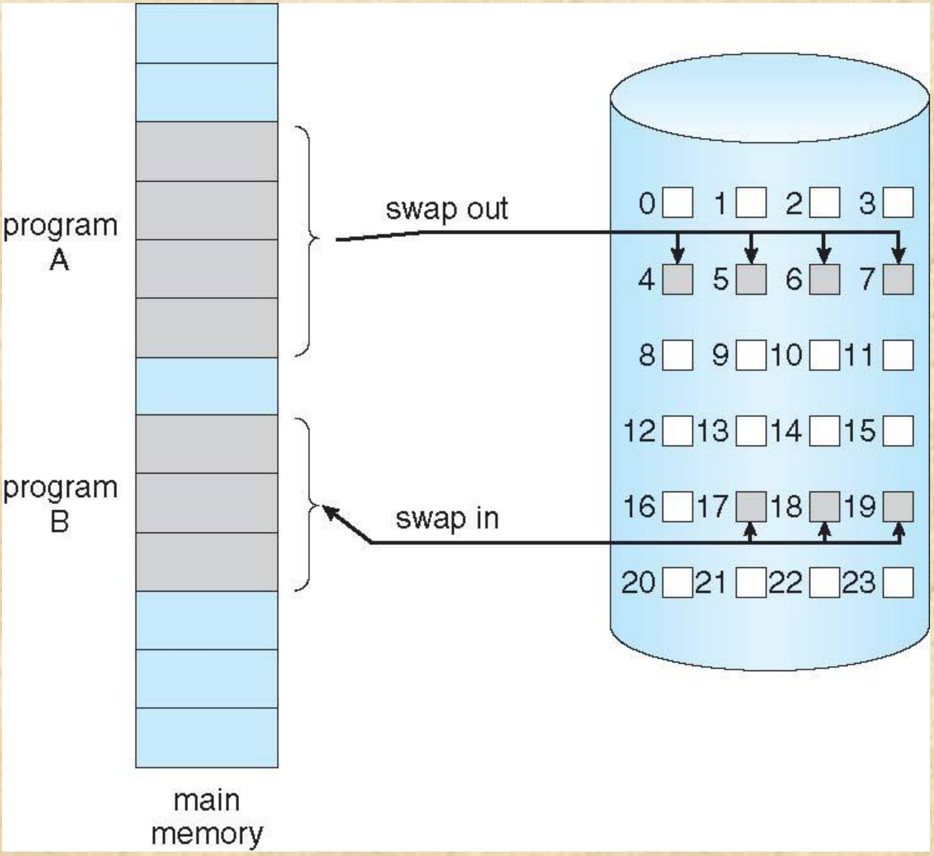


Adapted from Silberschatz, A. Galvin, P. and Gagne, G. (2013), Operating System Concepts

Demand Paging

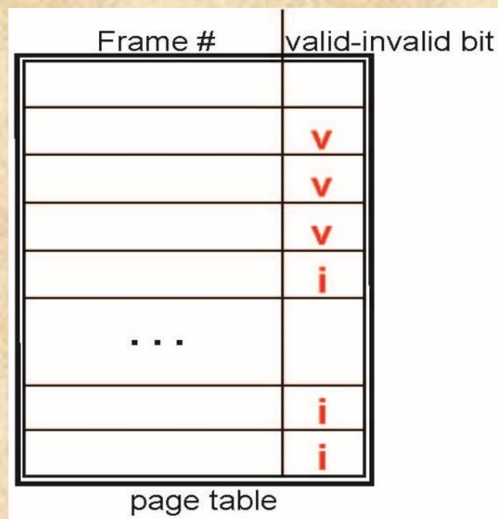
- Bring a page into memory only when it is needed.
  - Less I/O needed
  - Less memory needed
  - Faster response
  - More users
- Page is needed $\Rightarrow$ reference to it
  - invalid reference $\Rightarrow$ abort
  - not-in-memory $\Rightarrow$ bring to memory

Transfer of a Paged Memory to Contiguous Disk Space



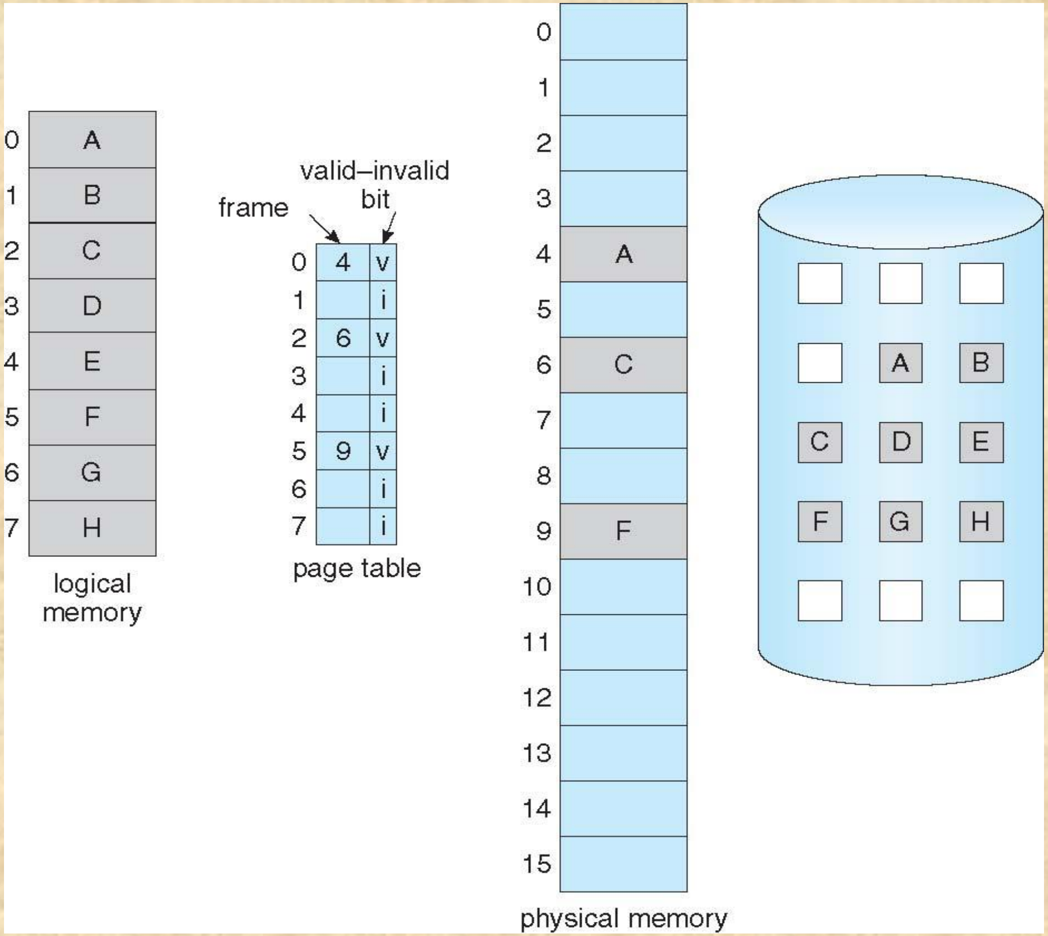
Valid-Invalid Bit

- With each page table entry a valid–invalid bit is associated (**V** ⇒ in-memory, **i** ⇒ not-in-memory)
- Initially valid–invalid but is set to **i** on all entries.
- Example of a page table snapshot.



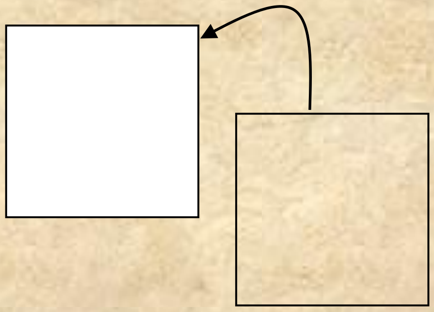
- During address translation, if valid–invalid bit in page table entry is **i** ⇒ page fault.

Page Table When Some Pages Are Not in Main Memory

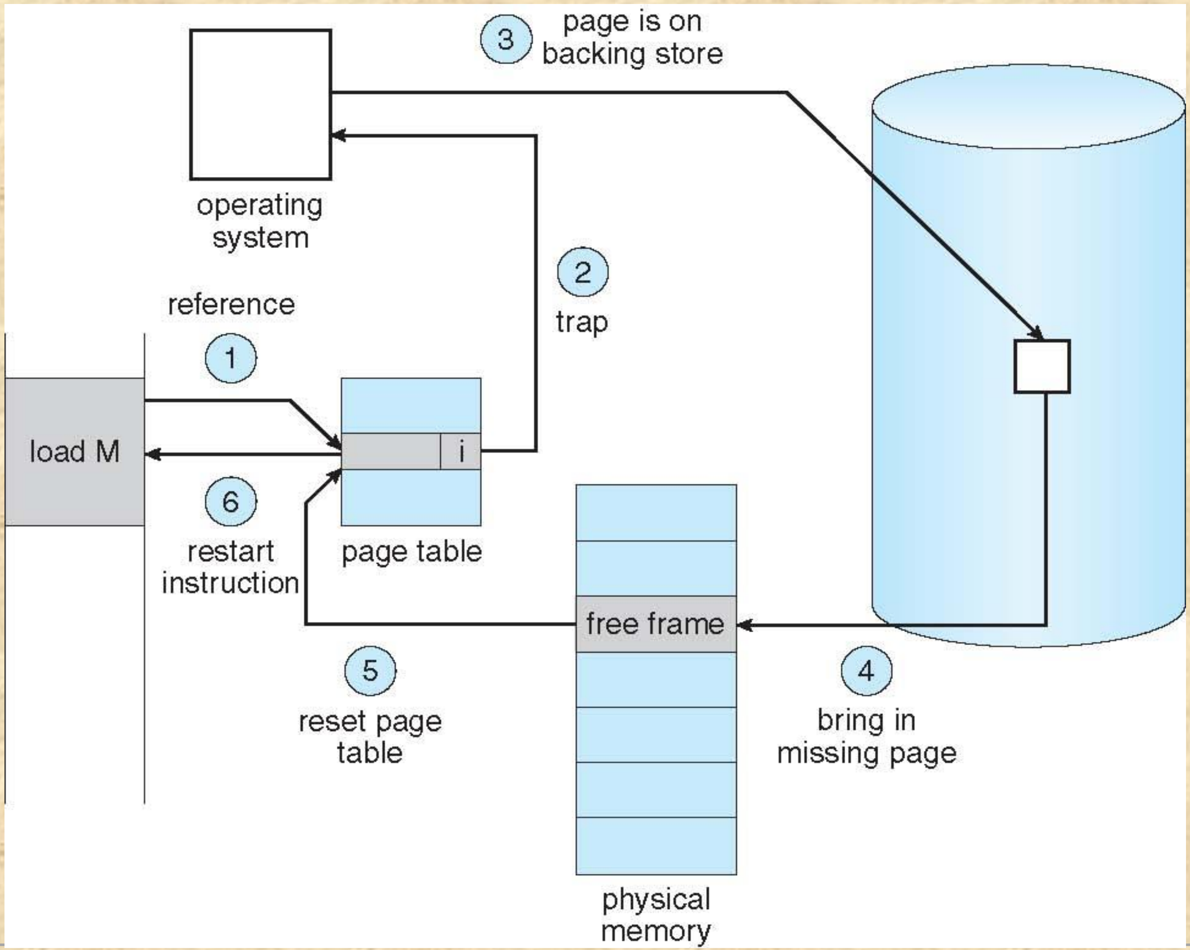


Page Fault

- If there is ever a reference to a page, first reference will trap to OS →page fault
- OS looks at another table to decide:
  - Invalid reference → abort.
  - Just not in memory.
- Get empty frame.
- Swap page into frame.
- Reset tables, validation bit = 1.
- Restart instruction: Least Recently Used



Steps in Handling a Page Fault



Adapted from Silberschatz, A. Galvin, P. and Gagne, G. (2013), Operating System Concepts

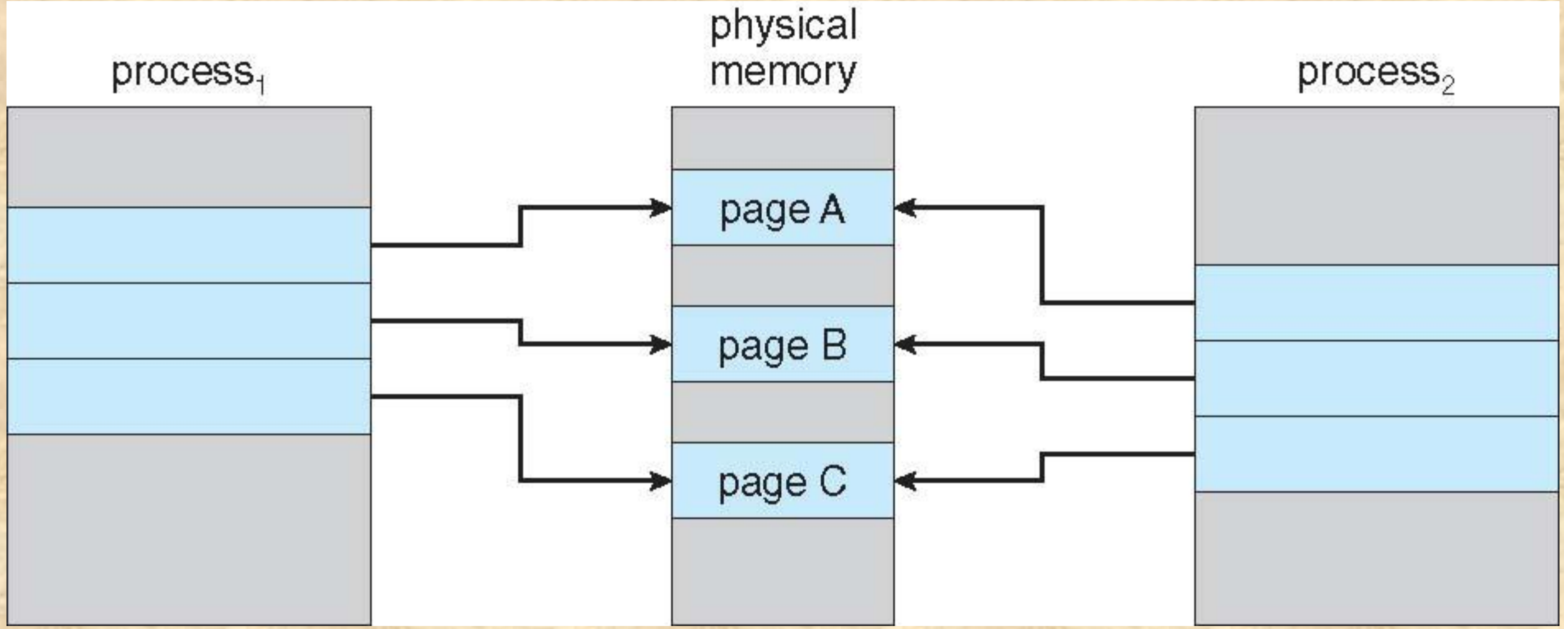
Process Creation

- Virtual memory allows other benefits during process creation:
  - Copy-on-Write
  - Memory-Mapped Files

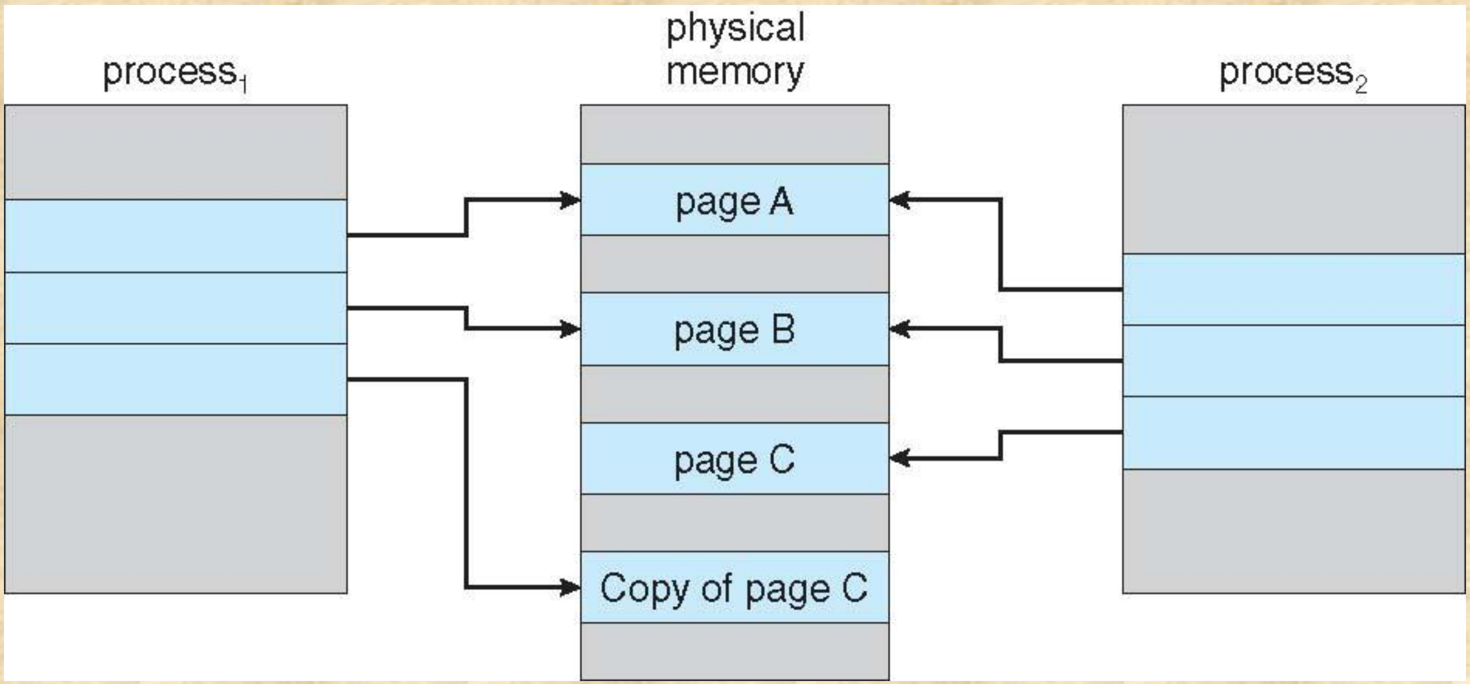
Copy-on-Write

- Copy-on-Write (COW) allows both parent and child processes to initially share the same pages in memory.
- If either process modifies a shared page, only then is the page copied.
- COW allows more efficient process creation as only modified pages are copied.
- Free pages are allocated from a pool of zeroed-out pages.

Before Process 1 Modifies Page C



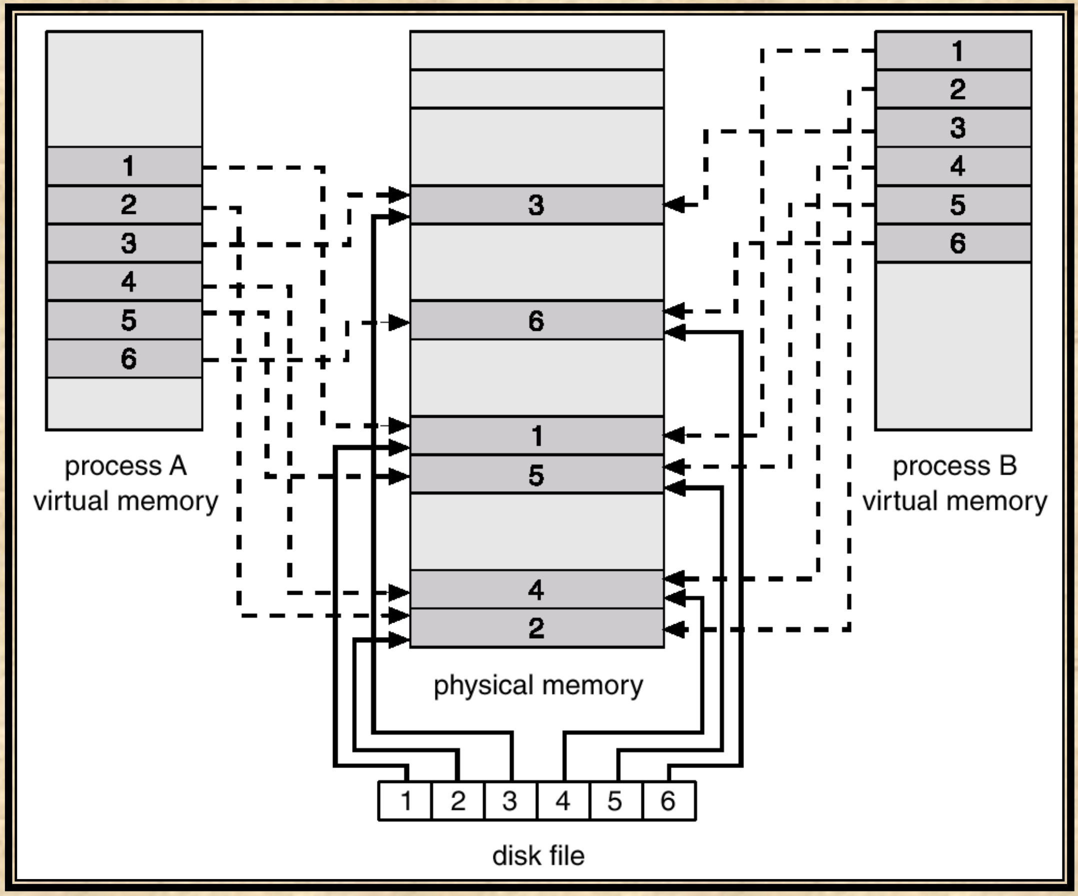
After Process 1 Modifies Page C



Memory-Mapped Files

- Memory-mapped file I/O allows file I/O to be treated as routine memory access by mapping a disk block to a page in memory.
- A file is initially read using demand paging. A page-sized portion of the file is read from the file system into a physical page. Subsequent reads/writes to/from the file are treated as ordinary memory accesses.
- Simplifies file access by treating file I/O through memory rather than **read()** **write()** system calls.
- Also allows several processes to map the same file allowing the pages in memory to be shared.

Memory-Mapped Files



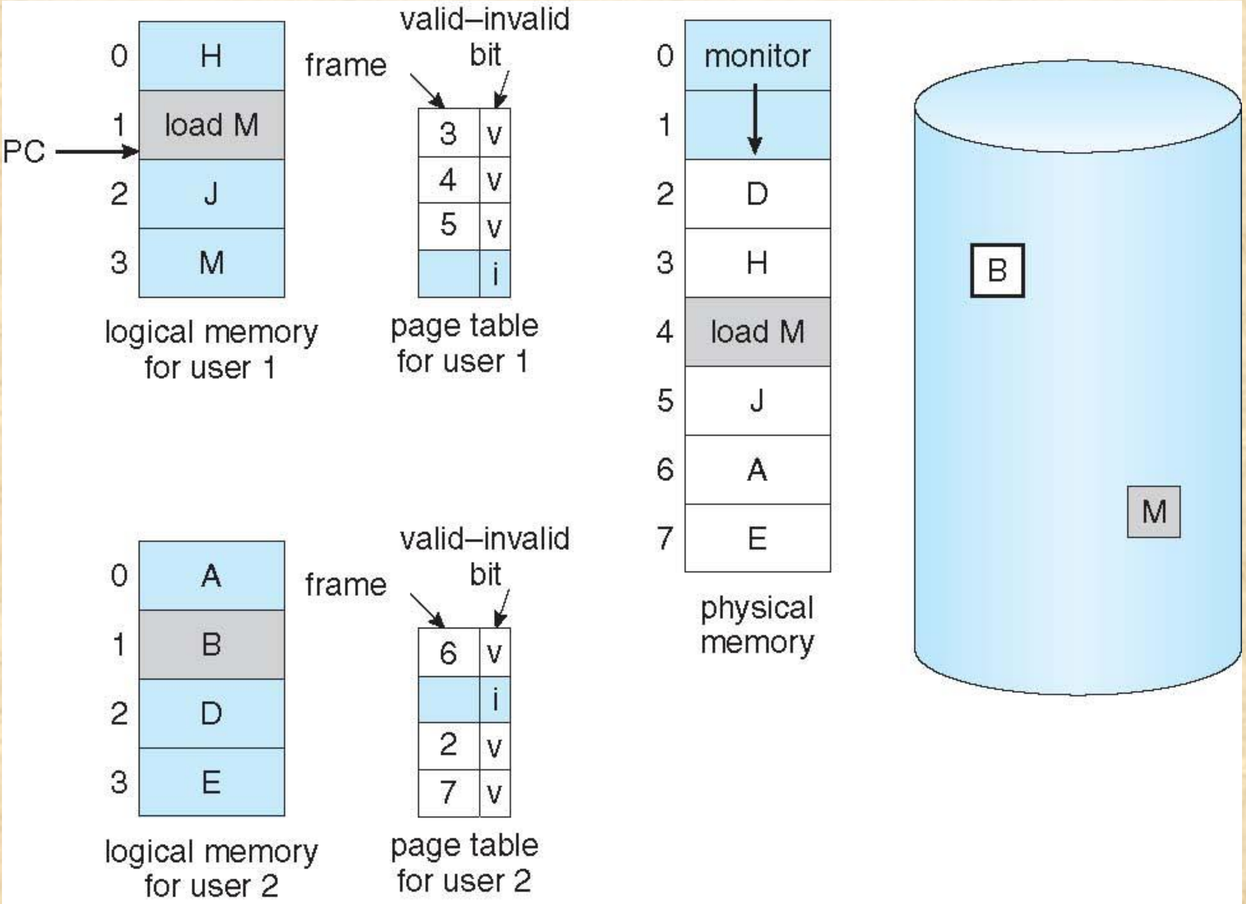
Page Replacement

- Prevent over-allocation of memory by modifying page-fault service routine to include page replacement.
- Use modify (dirty) bit to reduce overhead of page transfers
  - only modified pages are written to disk.
- Page replacement completes separation between logical memory and physical memory – large virtual memory can be provided on a smaller physical memory.

What happens if there is no free frame?

- Page replacement – find some page in memory, but not really in use, swap it out.
  - algorithm
  - performance – want an algorithm which will result in minimum number of page faults.
- Same page may be brought into memory several times.

Need For Page Replacement



Adapted from Silberschatz, A. Galvin, P. and Gagne, G. (2013), Operating System Concepts

Basic Page Replacement

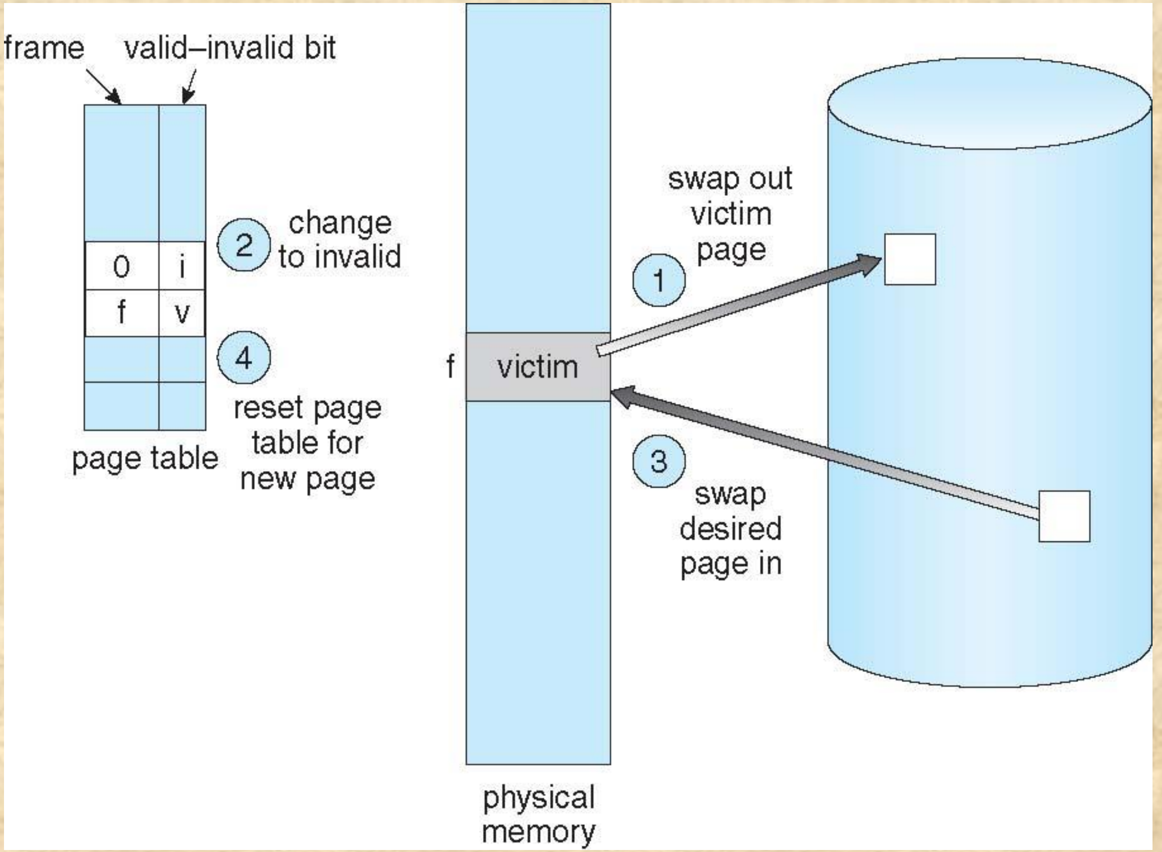
- Find the location of the desired page on disk.

- Find a free frame:
  - If there is a free frame, use it.
  - If there is no free frame, use a page replacement algorithm to select a *victim* frame.

- Read the desired page into the (newly) free frame. Update the page and frame tables.

- Restart the process.

Page Replacement



Page Replacement Algorithms

- Want lowest page-fault rate.
- Evaluate algorithm by running it on a particular string of memory references (reference string) and computing the number of page faults on that string.

Exercise

Consider the following page reference string:

1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5.

- Assuming a paging scheme with 3 frames is initially empty. Trace the allocation of pages to frames using
  - ***FIRST IN FIRST OUT (FIFO)***
  - ***OPTIMAL***
  - ***LEAST RECENTLY USED (LRU)*** algorithm.

Exercise

- Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- 3 frames (3 pages can be in memory at a time per process)

1	1	4	5
2	2	1	3
3	3	2	4

9 page faults

- 4 frames

1	1	5	4
2	2	1	5
3	3	2	
4	4	3	

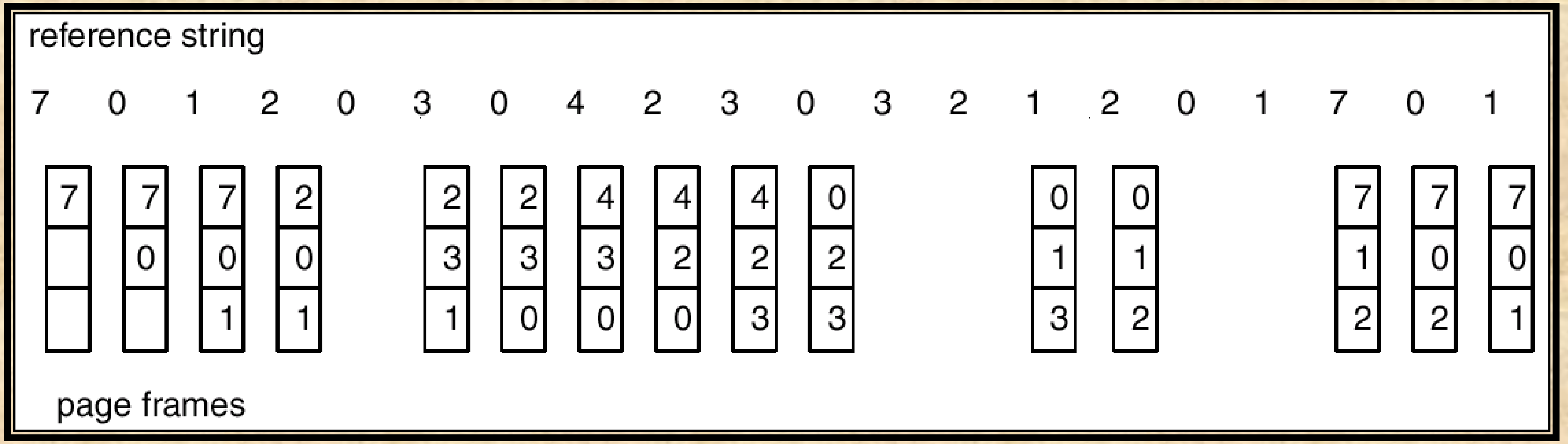
10 page faults

- FIFO Replacement – Belady's Anomaly
  - more frames $\Rightarrow$ more page faults

FIFO Page Replacement

Reference string:

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1



Optimal Algorithm

- Replace page that will not be used for longest period of time.
- 4 frames example
- 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

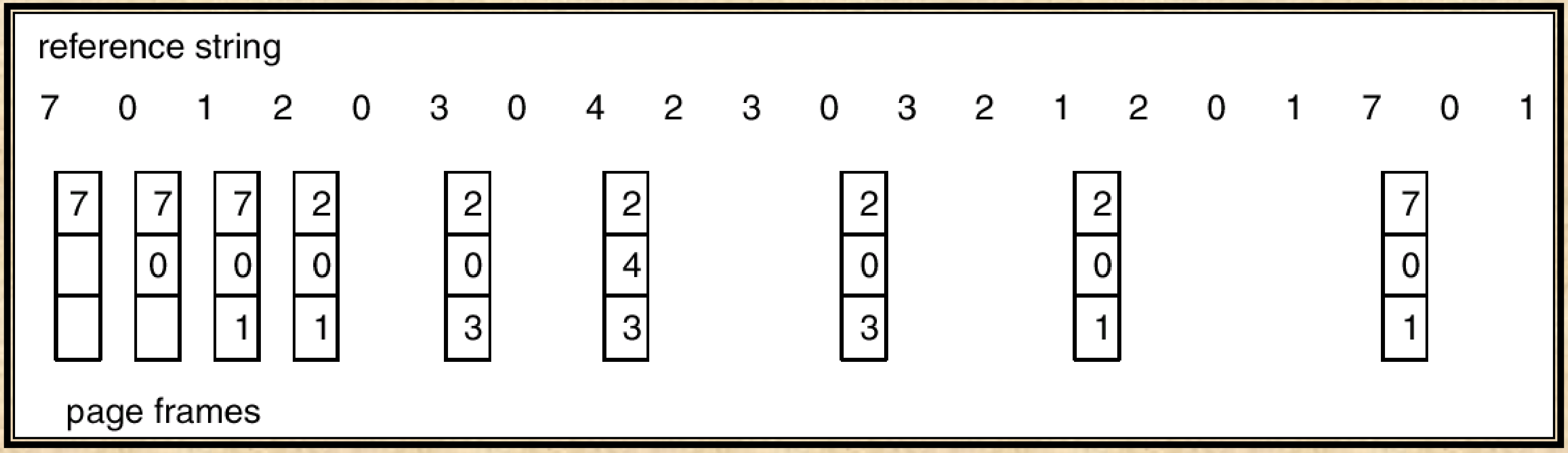
6 page faults

- How do you know this?
- Used for measuring how well your algorithm performs.

Optimal Page Replacement

Reference string:

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1



Least Recently Used (LRU) Algorithm

- Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

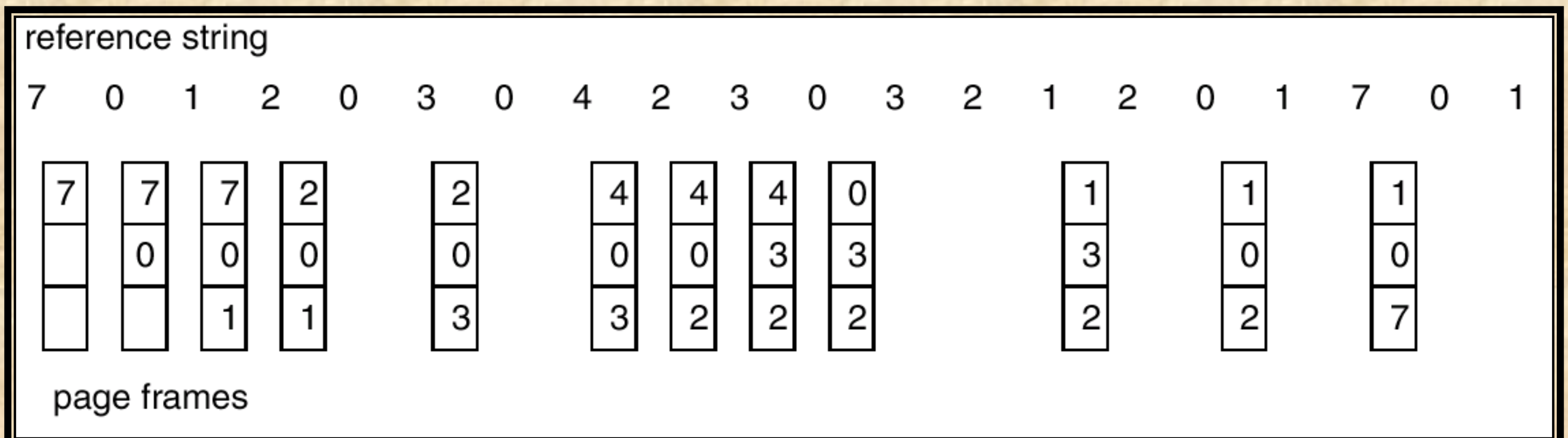
1	5	
2		
3	5	4
4	3	

- Counter implementation
  - Every page entry has a counter; every time page is referenced through this entry, copy the clock into the counter.
  - When a page needs to be changed, look at the counters to determine which are to change.

LRU Page Replacement

Reference string:

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1



Adapted from Silberschatz, A. Galvin, P. and Gagne, G. (2013), Operating System Concepts

Thrashing

Thrashing occurs when a computer's virtual memory subsystem is in a constant state of high paging activity.

A state in which the system spends most of its time swapping process pieces rather than executing instructions

This causes the performance of the computer to degrade or collapse (performance problems).

Solutions

To resolve thrashing due to excessive paging, a user can do any of the following:

- Increase the amount of RAM in the computer (generally the best long-term solution).
- Decrease the number of programs being run on the computer.
- Replace programs that are memory-heavy with equivalents that use less memory.
- Improve spatial locality.
- If a single process is too large for memory, there is nothing the OS can do. That process will simply thrash.
- If the problem arises because of the sum of several processes:
  - Figure out how much memory each process needs.
  - Change scheduling priorities to run processes in groups that fit comfortably in memory: must shed load.

Principle of Locality

- The operating system tries to guess, based on recent history, which pieces are least likely to be used in the near future
- Program and data references within a process tend to cluster
- Only a few pieces of a process will be needed over a short period of time
- Therefore it is possible to make intelligent guesses about which pieces will be needed in the future
- Avoids thrashing

Summary

- Virtual Memory
- Demand paging
- Valid invalid bit
- Page fault
- Page replacement algorithms
  - FIFO
  - LRU
  - Optimal
- Thrashing