CS 3305A

Virtual Memory and Page Replacement Algorithms

Lecture 18

Agenda

- Virtual Memory
- Demand Paging
- Page Fault
- Page Replacement

Virtual Memory: Main Idea

- We already discussed about it – logical address space!
  - Processes use a virtual (logical) address space
- Every process has its own address space
- The virtual address space can be larger than physical memory.
  - Only part of the virtual address space is mapped to physical memory at any time.
- Parts of processes' memory content is on disk.
- Hardware & OS collaborate to move memory contents to and from disk (swapping)

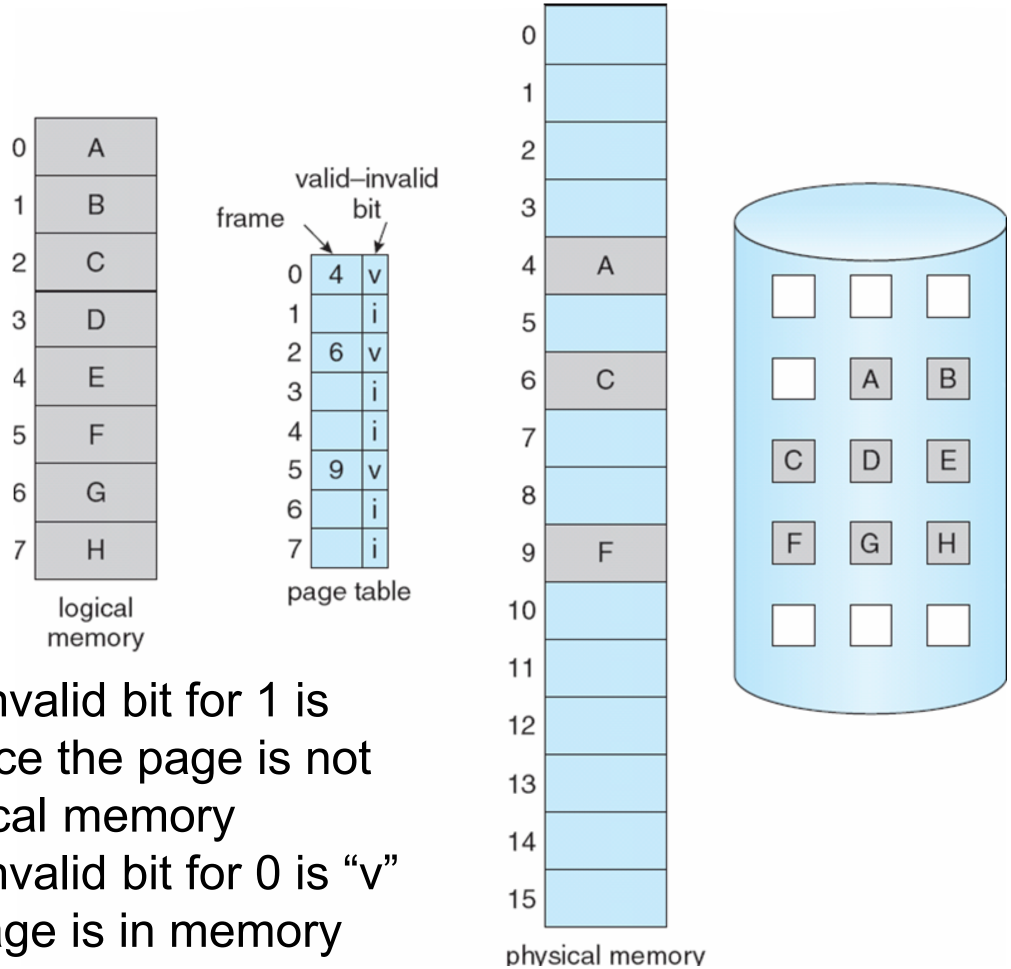
Demand Paging

- Bring a page into memory only when it is needed
  - Why?
    - Less I/O needed i.e., faster response
    - Less memory needed

Demand Paging

- We need to distinguish between pages that are in memory and the pages that are on disk
- A valid-invalid bit is part of each page entry
  - When the bit is set to “valid” the associated page is in memory
  - If the bit is set to “invalid” the page is on the disk

Demand Paging



- The valid-invalid bit for 1 is set to "i" since the page is not in the physical memory
- The valid-invalid bit for 0 is "v" since the page is in memory

Page Fault

- What happens if a process tries to access a page that was not brought into memory?
- Access to a page marked invalid causes a page fault

Page Replacement

- Let's assume that our physical memory consists of 40 frames
- We have 8 processes with 10 pages. That is 80 pages.
  - Obviously 80 pages is more than 40 frames

Page Replacement

- What do we do when a process needs a frame and there isn't one free?
- Essentially we choose a frame and free it

Page Replacement

- A page replacement algorithm describes which frame becomes a victim.
- Designing an appropriate algorithm is important since disk I/O is expensive
- Slight improvements in algorithms yield large gains in system performance

Page Replacement

- We will discuss several algorithms
- The examples assume:
  - 3 frames
  - Reference string:
    7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1
  - Each of the numbers above refers to a specific page number

Page Replacement Algorithms

- Optimal Page Replacement Algorithm
- FIFO
- Least Recently Used (LRU)
- Least frequently used (LFU)
- Most OS's use LRU

Optimal Page Replacement Algorithm

- Replace page needed at the farthest point in future i.e. replace the page that will not be used for the longest period of time

- This should have the lowest page fault rate

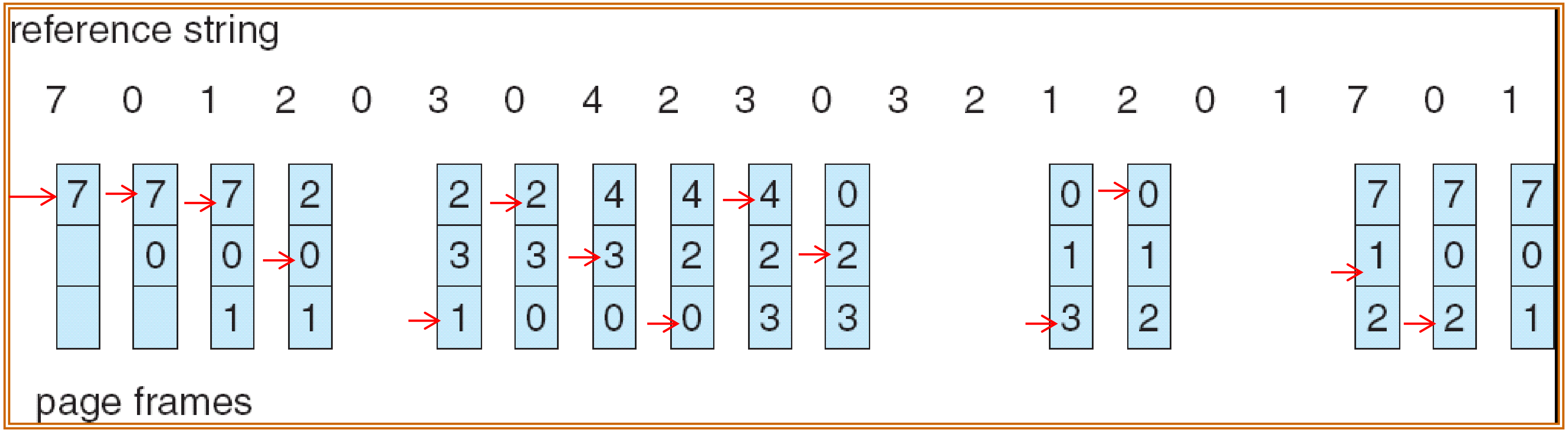
Optimal Page Replacement

- Optimal is easy to describe but impossible to implement
- At the time of the page fault, the OS has no way of knowing when each of the pages will be referenced next

FIFO Page Replacement Algorithm

- Maintain a linked list of all pages
  - Each page is associated with the time when that page was brought into memory
- Page chosen to be replaced is the oldest page

FIFO Page Replacement

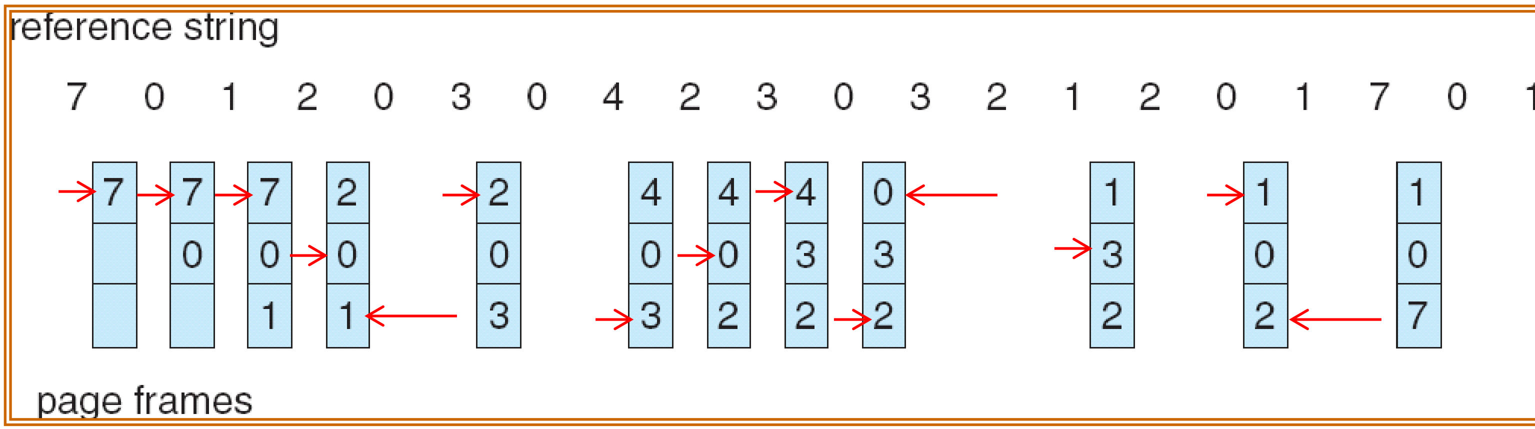


Note: The red arrow is pointing to the oldest page

LRU Replacement Algorithm

- LRU replacement associates with each page the time of that page's last use
- When a page must be replaced, LRU chooses the page that has not been used for the longest period of time.

LRU Page Replacement



Note: The red arrow is pointing to the LRU page

Summary

- We have studied the need for page replacement algorithms
- Several algorithms have been discussed including:
  - Optimal
  - FIFO
  - LRU