ECE437/CS481

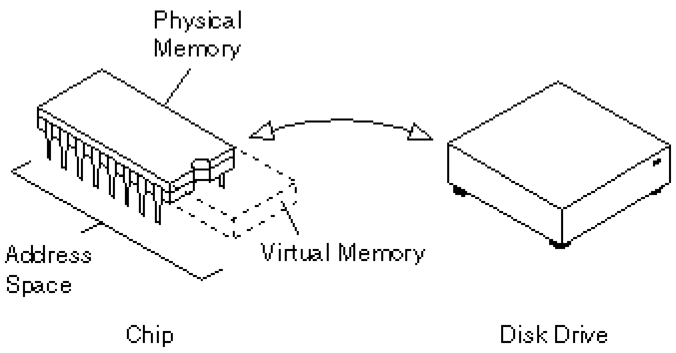
M07C: VIRTUAL MEMORY

CHAPTER 9.1-9.4

Xiang Sun

The University of New Mexico

Virtual Memory

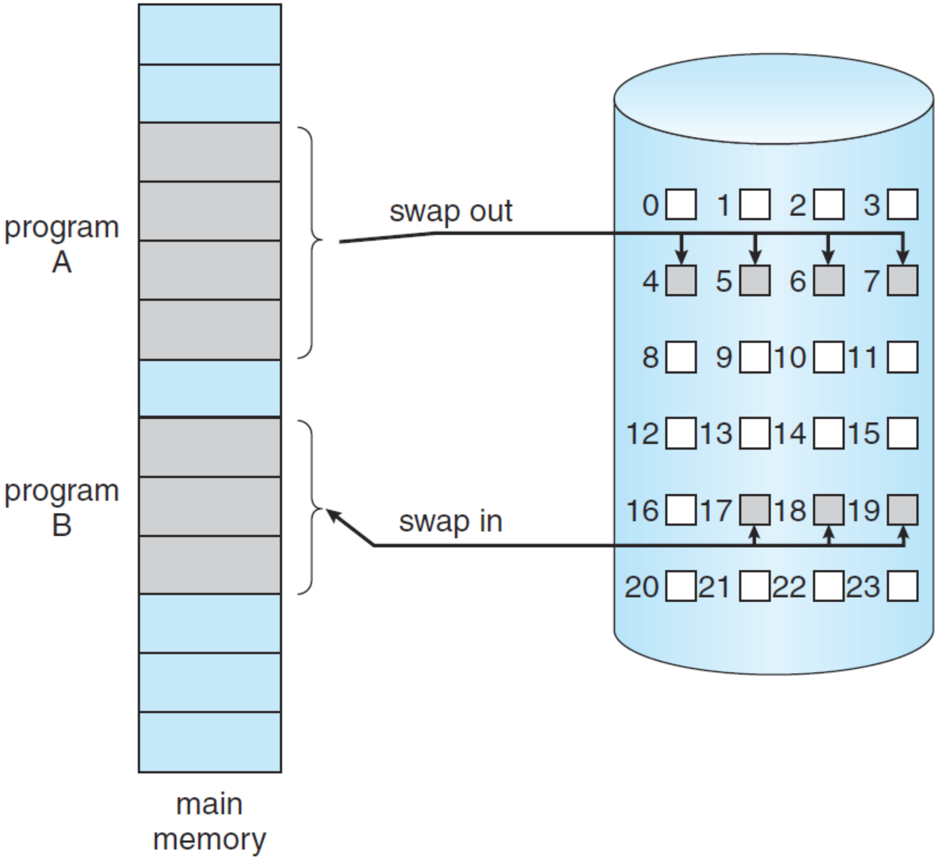


- Paging forms a foundation of virtual memory
  - ✓ Paging allows noncontiguous memory space allocation
- Virtual memory is more than paging
  - ✓ Virtual memory is a technique to allow the execution of processes not be completed in the main memory. That is, secondary memory (e.g., hard disk) can be used as a part of main memory.
  - ✓ Virtual memory technology frees programmers from concern over memory storage limitations.
  - ✓ Virtual memory can be implemented via Demand Paging.

Virtual Memory

❑ Demand Paging

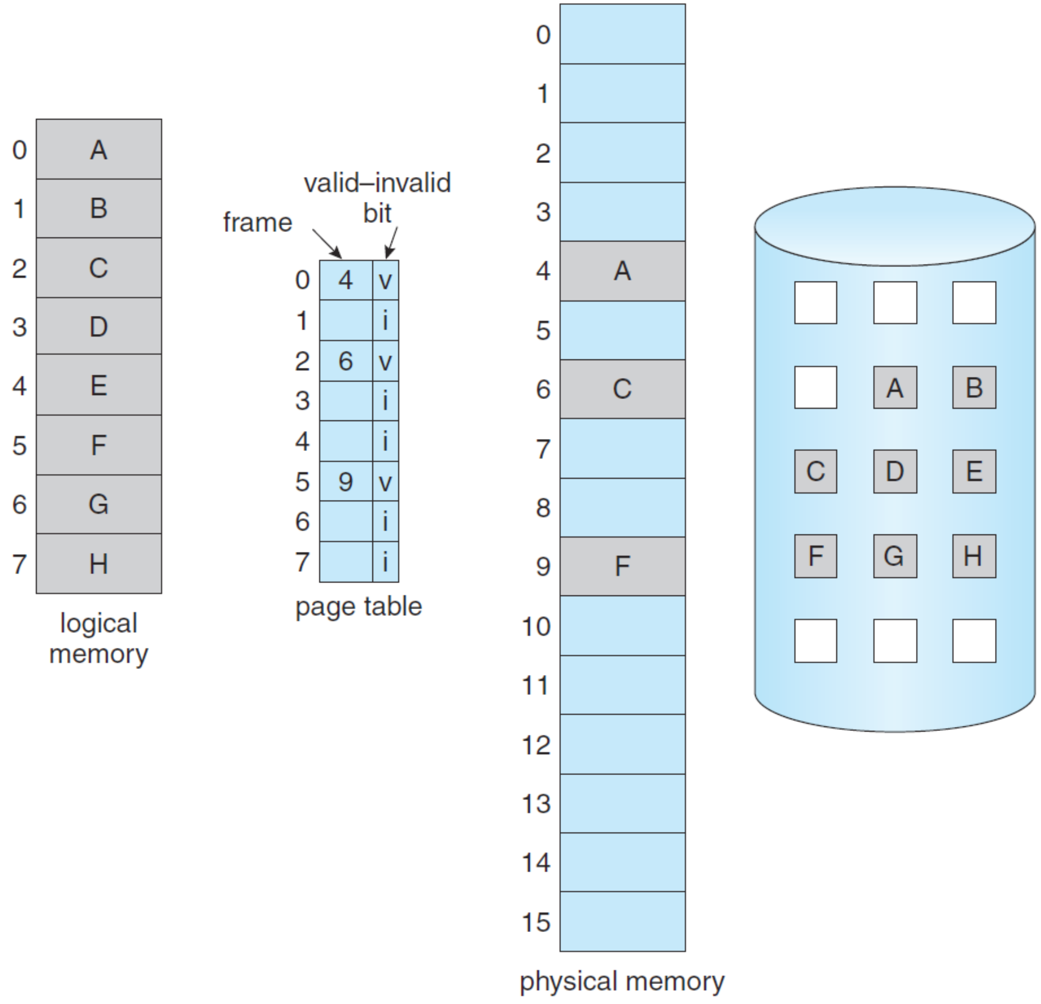
- Recall that: the pages can be swapped in/out between the main memory and disk.
  - ✓ For example, when the memory fills up, allocating a page in memory requires some other page to be swapped out from memory
  - ✓ The unit for swapping in/out is page



- Demand Paging: when a page is referenced (e.g., a page is needed by the CPU), and the page is NOT in the main memory, then the page is swapped into the main memory.

❑ Demand Paging

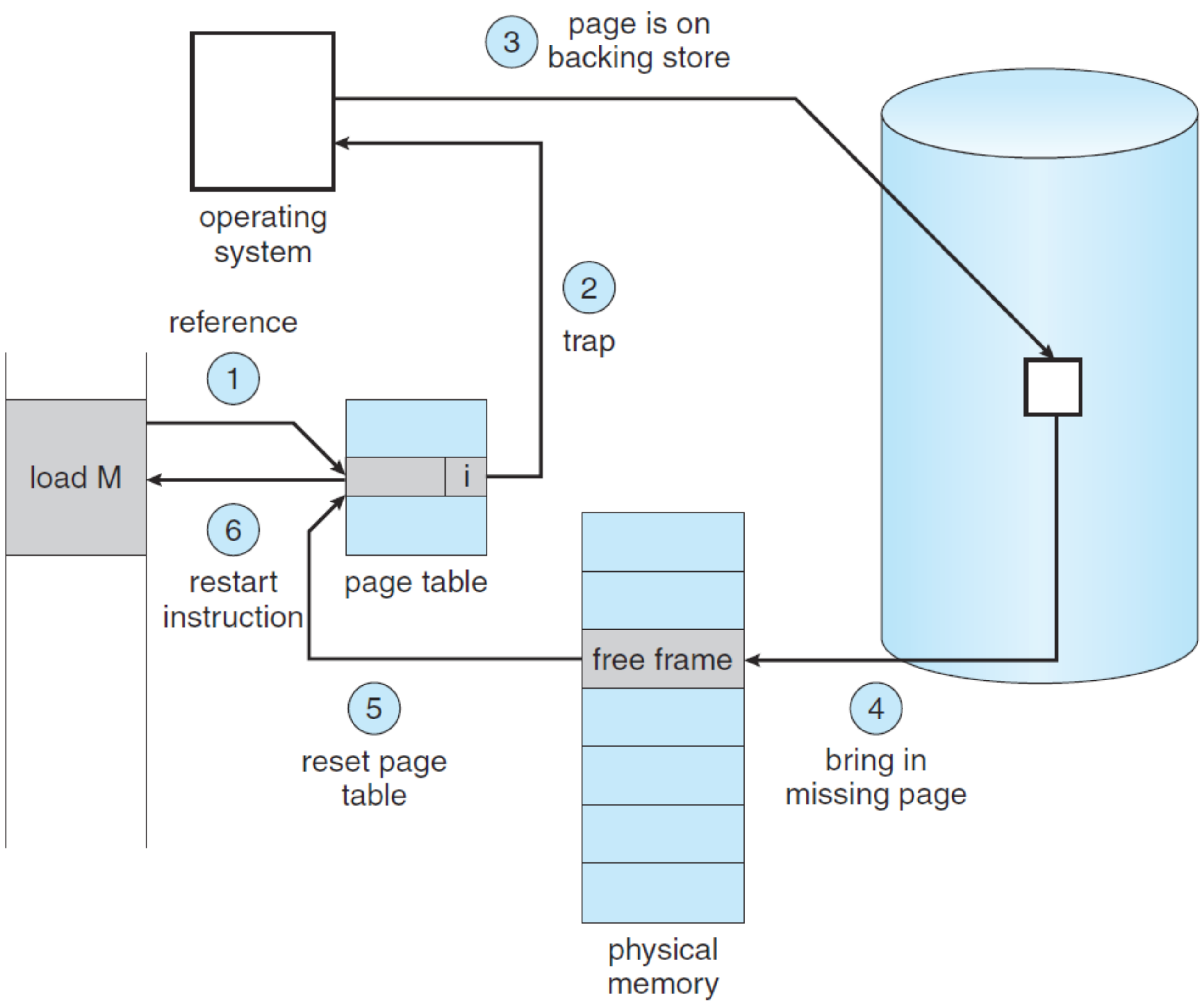
- Demand paging requires hardware support to distinguish between pages in memory and those in disk.
  - ✓ Recall that: in a page table, there is a valid/invalid bit to check if the referenced page is a legal page (w.r.t. the running process) or not.
  - ✓ It can be used to identify the location of the referenced page.
    - v → in-memory
    - i → not-in-memory



- During the address translation, if the valid/invalid bit is "i", a page fault will be raised.
  - ✓ A page fault is a type of exception raised by MMU
  - ✓ After raising the page fault, the kernel (interrupt handler) will deal with the page fault.

❑ Procedures in handling page fault

1. The page is looked up in the page table to determine if the referenced page is valid or not (hardware).
2. If the page is valid, the CPU can access the related frame in the main memory. If the page is not valid, a page fault is generated.
3. Find the referenced page in the hard disk.
4. Find a free frame in the main memory (may need to free up a page). Schedule a disk operation to read the referenced page into the newly allocated frame.
5. When the frame is filled, modify the page table.
6. Restart the instruction that was interrupted.



❑ Effective access time (EAT) in demand paging

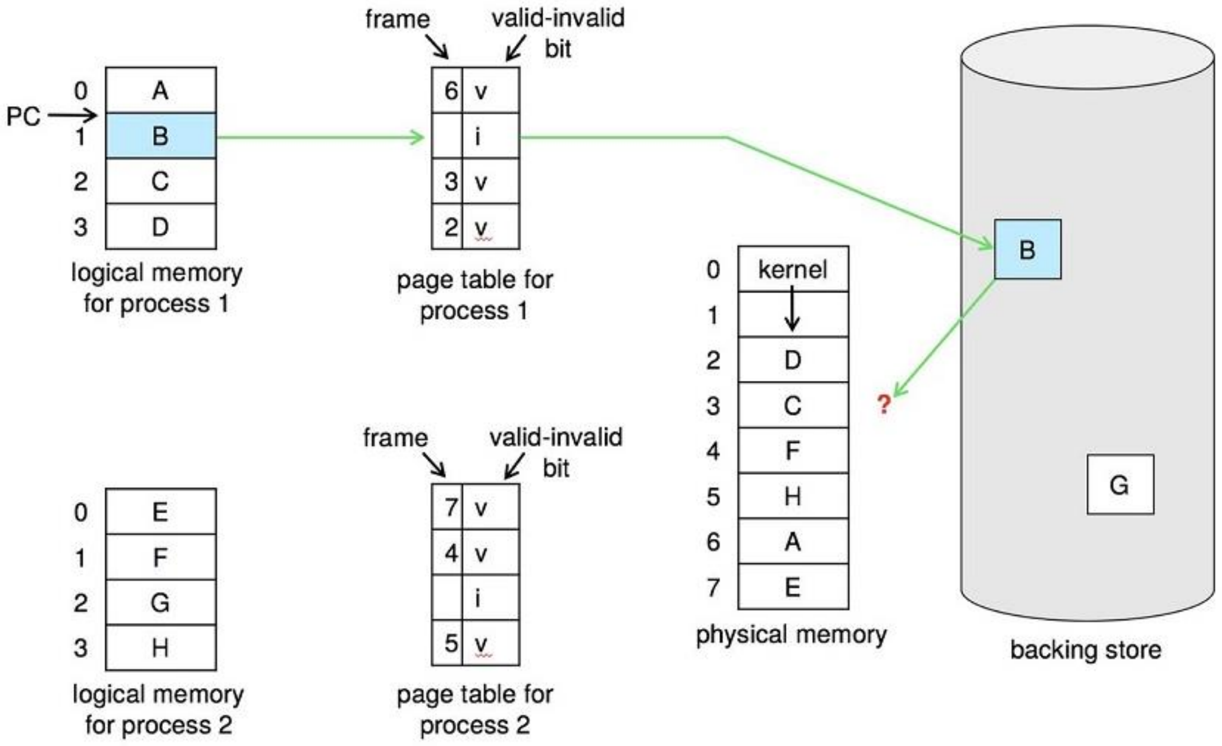
- EAT: the average time of accessing a memory page.
- Assume that
  1) memory access time = 200 ns (nanoseconds)
  2) average page-fault service time = 8 ms (milliseconds)
  3) probability of having a page fault is *p*—page-fault rate

- EAT = 200ns × 2 + (8ms+200ns × 2)×p
  =(400 + 8,000,400×p) ns

- If p=1/1,000,
  EAT=400+8,000.4=8,400.4 ns ≈ 8.4 μs (microsecond)
  This is a slowdown by a factor of 21!!

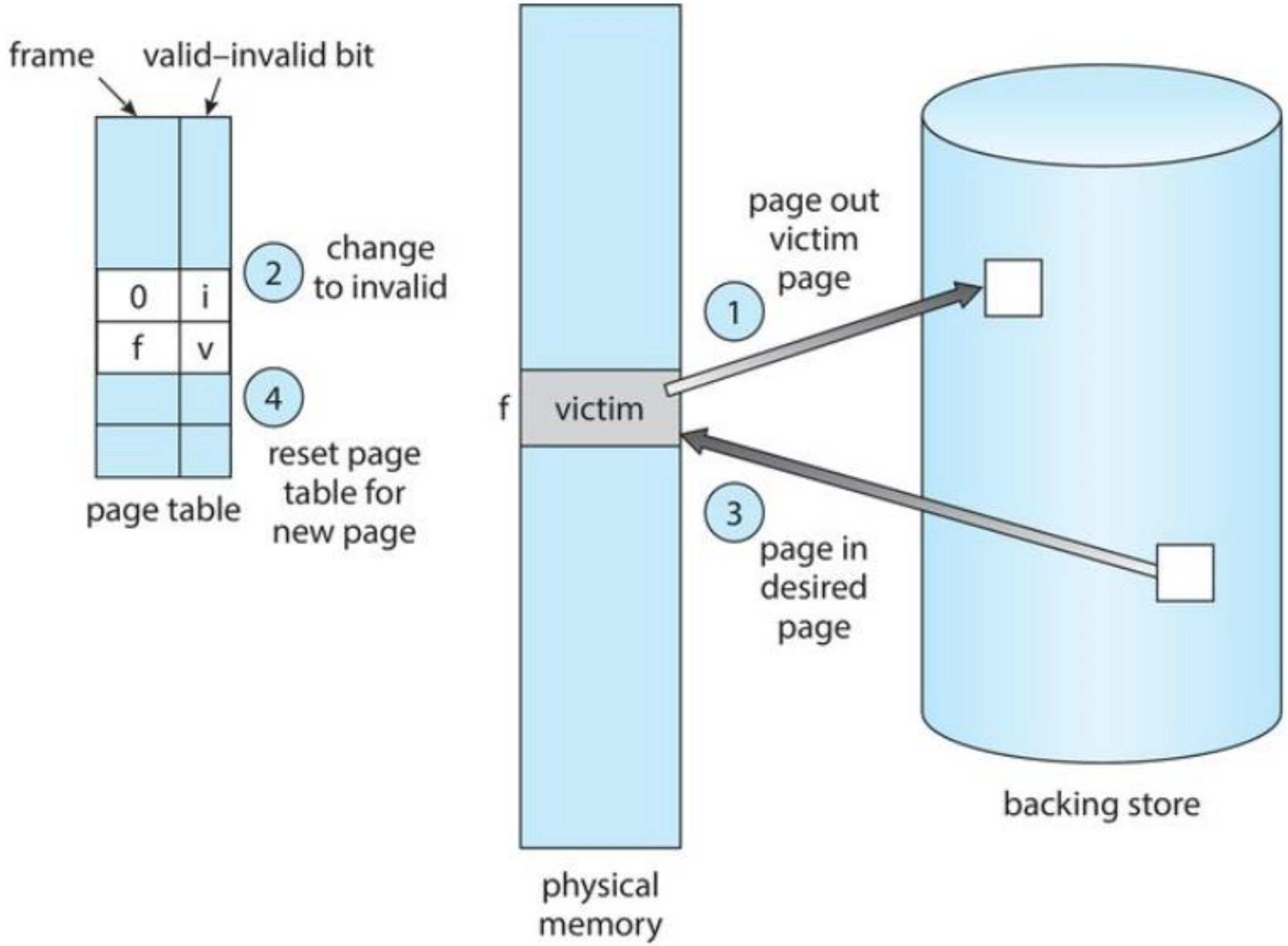
❑ Page replacement in virtual memory

- In order to increase the degree of multiprogramming, we are over-allocating memory to processors. So, some pages are swapped out of the physical memory.
- If a page fault occurs and there is no free frame in the main memory, then how to swap the referenced page in the memory?---page replacement
- Page replacement: find a suitable page in memory, and swap it out to the hard disk.



Virtual Memory

- Basic steps of page replacement
  - Victim frame is the selected frame (by the page replacement algorithm) to be swapped out



Page replacement algorithm

- Design metric
  - will result in the lowest page-fault rate
  - will introduce the minimum runtime overhead
    - overhead of executing the replacement algorithm

- Evaluate algorithm by running it on
  - a particular string of memory references (reference string)
  - computing the number of page faults on that string

- In all our examples, the reference string is

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

❑ Page replacement algorithm

➢ FIFO Page Replacement---If a page must be replaced, the oldest page is chosen.

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

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

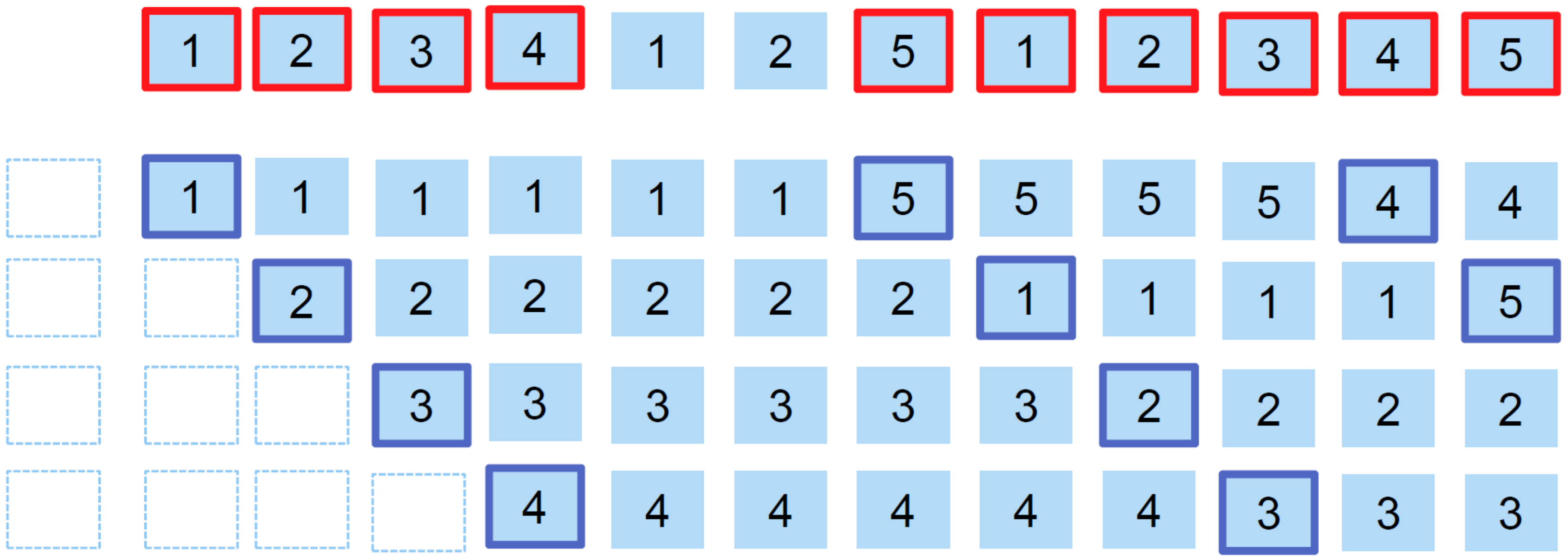
- Page fault = 9/12

indicate page fault

❑ Page replacement algorithm

➢ FIFO Page Replacement---If a page must be replaced, the oldest page is chosen.

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



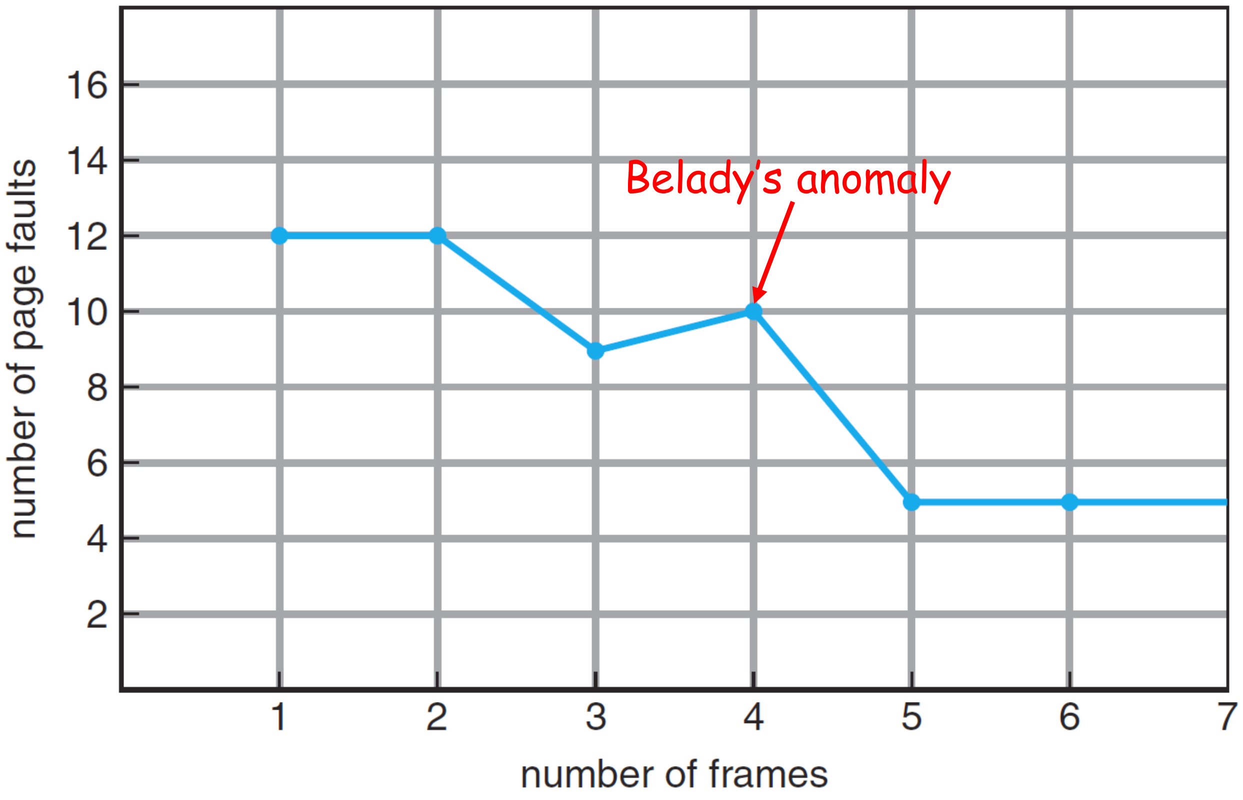
- Page fault = 10/12 indicate page fault

❑ Page replacement algorithm

- FIFO Page Replacement---If a page must be replaced, the oldest page is chosen.
  - observation: more frames NOT → lower page faults.

    + easy to implement

    - not satisfactory performance



Page replacement algorithm

- Optimal Page Replacement ---looking forward to the future: replace the page which will not be referenced for the longest time.

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

- Page fault = 6/12

❑ Page replacement algorithm

- Optimal Page Replacement ---looking forward to the future: replace the page which will not be used for the longest time.

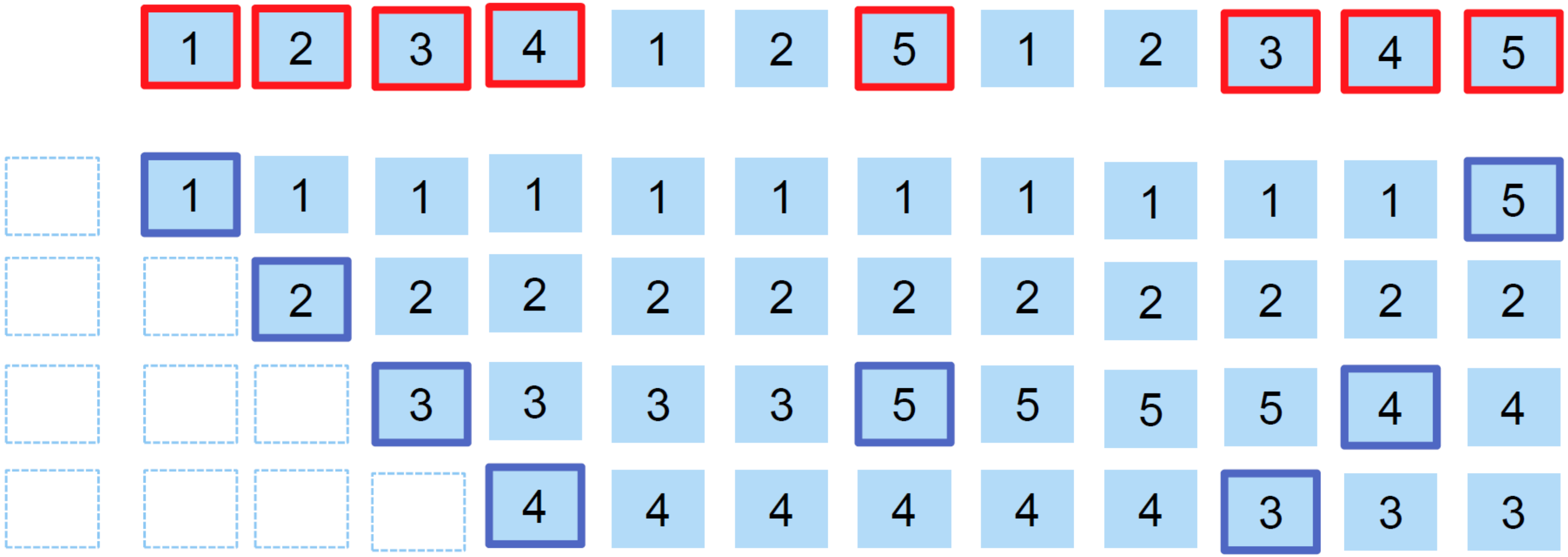
  + Optimal
  - Almost impossible to implement

  - Used for measuring other algorithms' performance

  - If looking forward is not implementable, then let's looking backwards

Page replacement algorithm

- Least Recently Used (LRU) Page Replacement ---Looking backwards: replace the page which was least recently used.

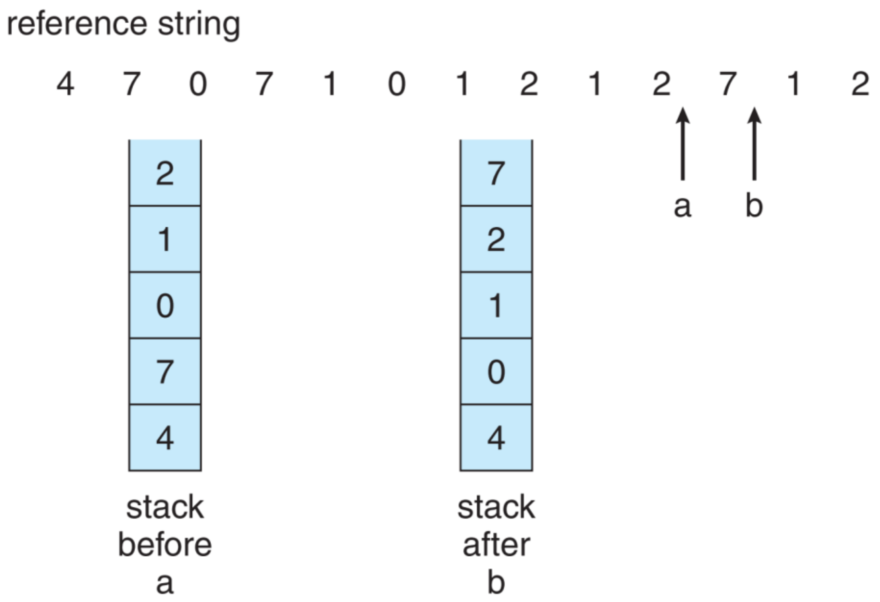


- Page fault = 8/12

LRU (least Recently Used) replacement algorithm implementation

- **Counters**
  - ✓ Each page-table entry is associated with a time-of-use field.
  - ✓ Whenever a refence to a page is made, the current time (in the clock register) are copied to the time-of-use field.
  - ✓ The page with the smallest value in the time-of-use field (i.e., least recent used) will be replaced.

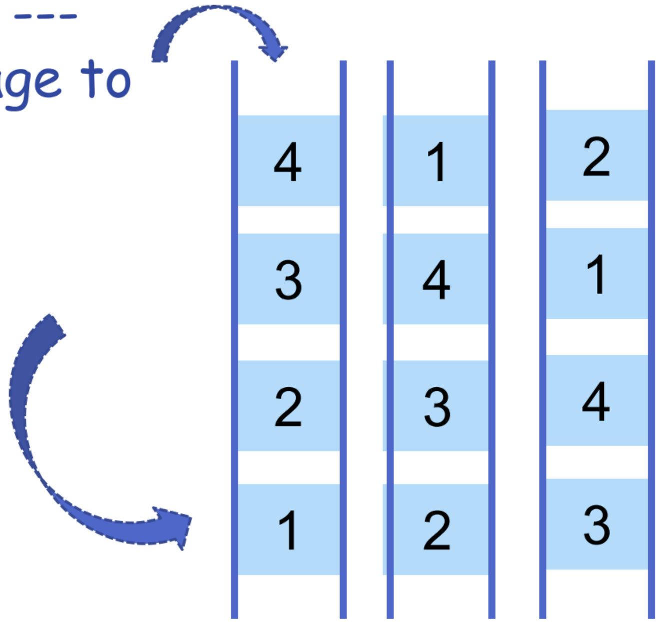
  \+ Simple to understand
  \- High overhead: 1) conduct a write to memory (page table) for each memory access; 2) search the entire page table to find the LRU page.



- **Stack**
  - ✓ Use a stack to keep page numbers being referenced.
  - ✓ Whenever a page is referenced, it is removed from the stack and put on the top.
  - ✓ The LRU page is always at the bottom.
  - ✓ Lower overhead, but entries need to be removed from the middle of the stack (e.g., "7")

LRU (least Recently Used) replacement algorithm implementation

- Link List—keep a double-linked list of page frames
  - memory reference time --- move the referenced page to the tail
  - replacement time --- pick up one at the head
  - link list operations are expensive



Reference string: **1, 2, 3, 4, 1, 2**

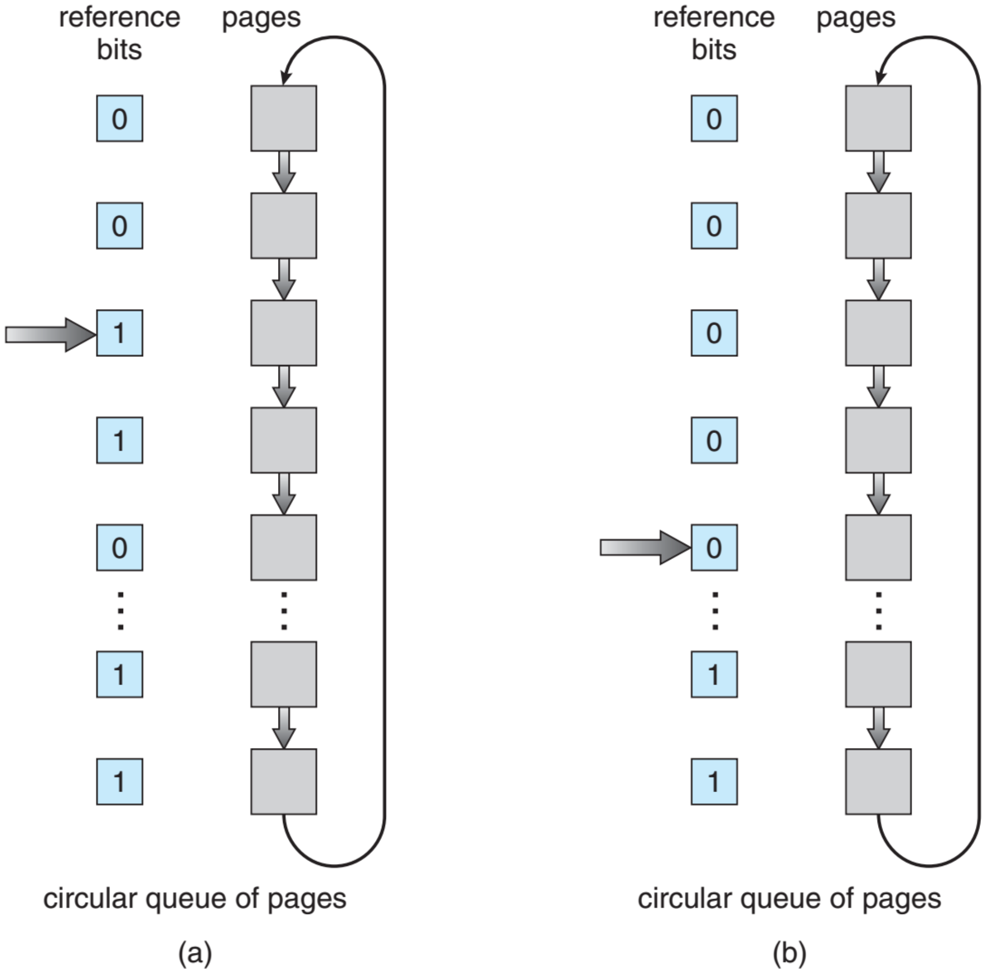
- Some Approximated-LRU Algorithms ---Not-recently-used

  - Reference bit – R only
    - ✓ one reference bit R per page table entry
    - ✓ memory reference time --- set R = 1
    - ✓ replacement time --- randomly pick up one with R is 0, reset all Rs to 0

  - Reference bit – R & Modification bit - M
    - ✓ reference bit R + modification bit M per page table entry
    - ✓ memory reference time --- set R = 1, and set M = 1 if write
    - ✓ replacement time ---
      - randomly pick up one with the minimal value of R&M
      - reset all Rs to 0

Some Approximated-LRU Algorithms --- Not-recently-used

- Reference bit + FIFO - second chance
  - ✓ keep a circular list of all page frames
  - ✓ one reference bit R per page table entry
  - ✓ keep a current inspection point
  - ✓ memory reference time --- set R=1
  - ✓ replacement time ---
    - if the current entry with R=1
      a) set R=0;
      b) advance the point, "give it a second chance"
    - else
      the current page is the "victim page"; replace the "victim page" with the new page.



Some Approximated-LRU Algorithms ---Not-recently-used

- Reference bit + aged history
  - one reference bit R + counter C (8 bits; initially, “00000000””) per page table entry
  - memory reference time --- set R=1
  - at each regular interval (e.g., 100 ms)---1) shift the reference bit into the high-order bit of its count C for each page entry; 2) shift the other bits right by 1 bit; 3) discard the low-order bit; 4) set R=0;
    - e.g., if C=00000000 and R=1, after one interval, C=10000000 and R=0.
  - replacement time ---
    - replace the page with the smallest value of C

Other page replacement algorithms

- Counting-based page replacement
  - ✓ One counter per page table entry. The counter records the number of references that have been made to its page.
    - Least frequently used algorithm: pick the page with the smallest value of counter being replaced.
    - Most frequently used algorithm: pick the page with the maximum value of counter being replaced
- Page-Buffering Algorithm
  - ✓ Construct a pool of free page pool in advance (before a page fault incurs).