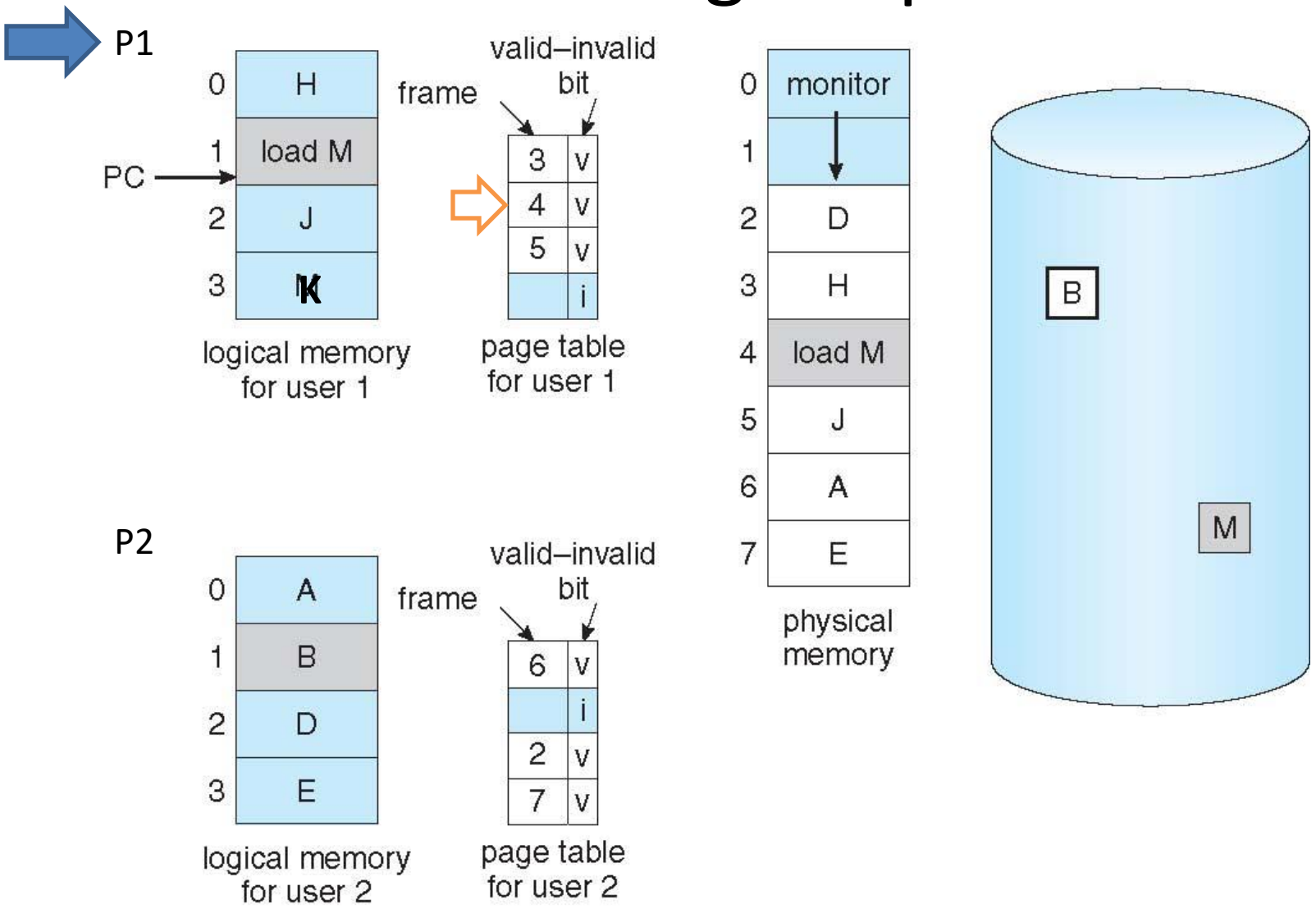
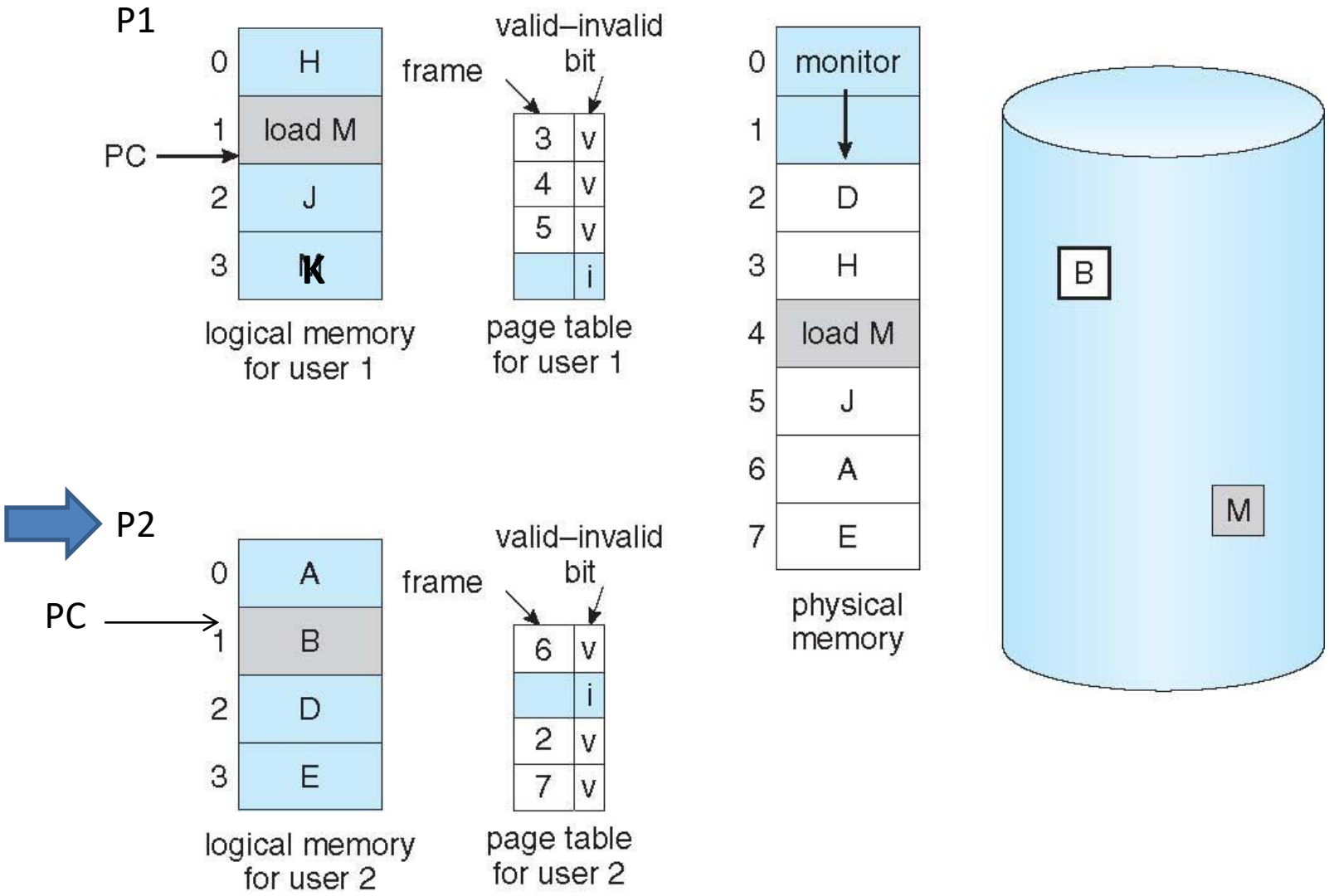
Page Replacement

Need For Page Replacement



Need For Page Replacement



Basic Page Replacement

1. Find the location of the desired page on disk

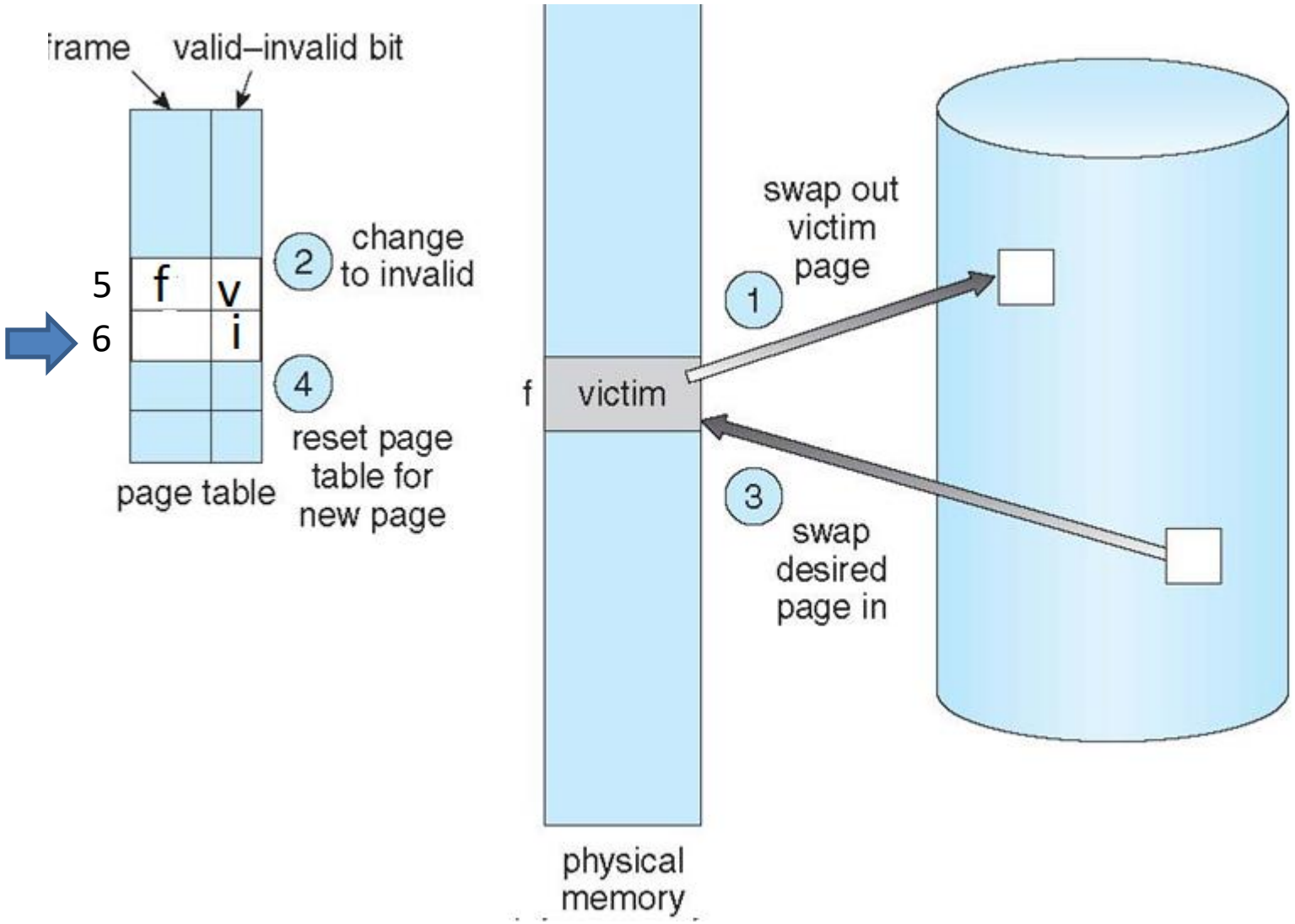
2. Find a free frame:
   - If there is a free frame, use it
   - If there is no free frame, use a page replacement algorithm to select a **victim frame (of that process)**
     - Write victim frame to disk

3. Bring the desired page into the (newly) free frame; update the page and frame tables

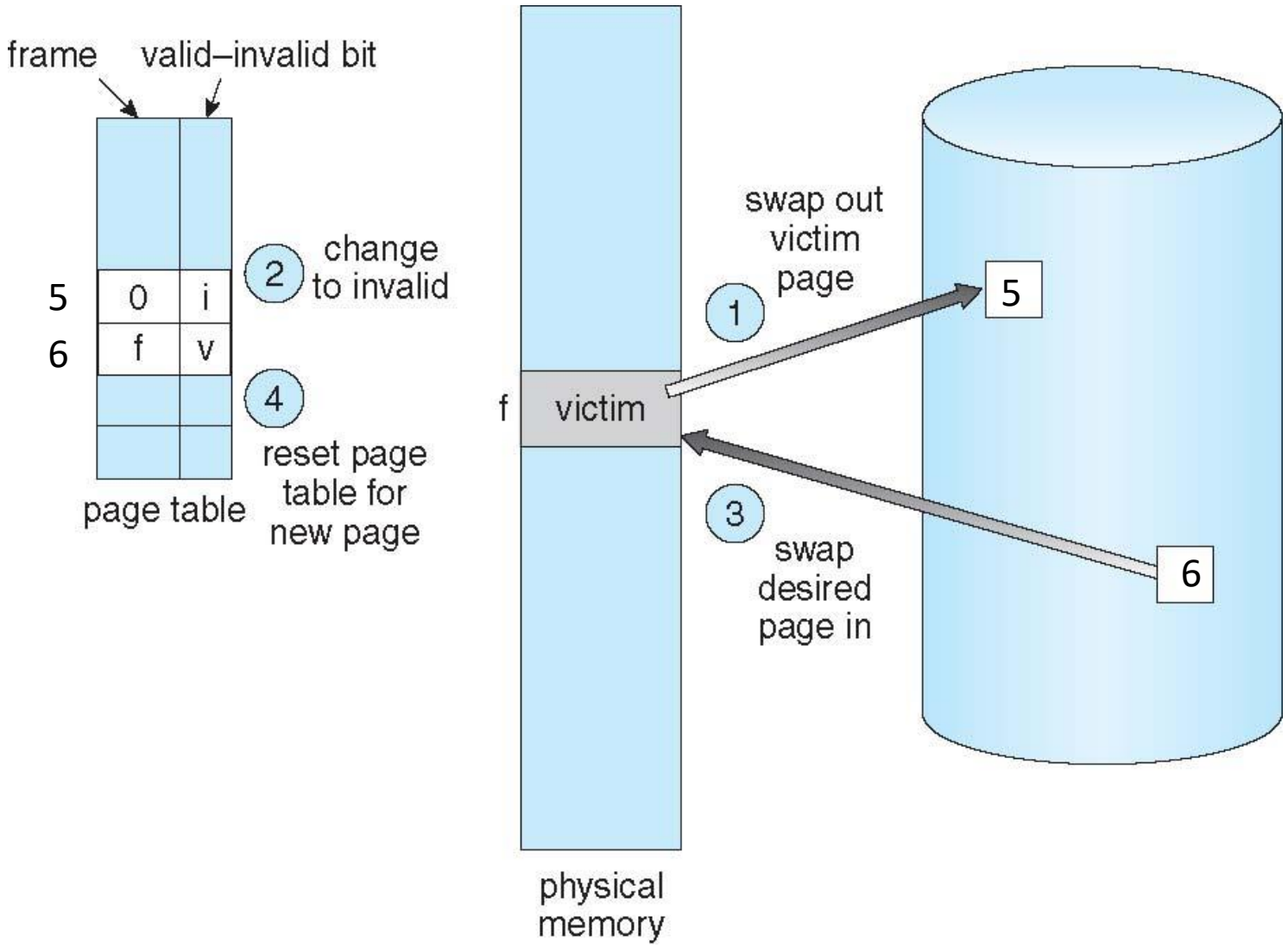
4. Continue the process by restarting the instruction that caused the trap

Note now potentially 2 page transfers for page fault – increasing Effective memory access time

Page Replacement



Page Replacement



Evaluation

- Evaluate algorithm by running it on a particular string of memory references (reference string) and computing the number of page faults on that string
  - String is just page numbers, not full addresses
  - Repeated access to the same page does not cause a page fault
- Trace the memory reference of a process

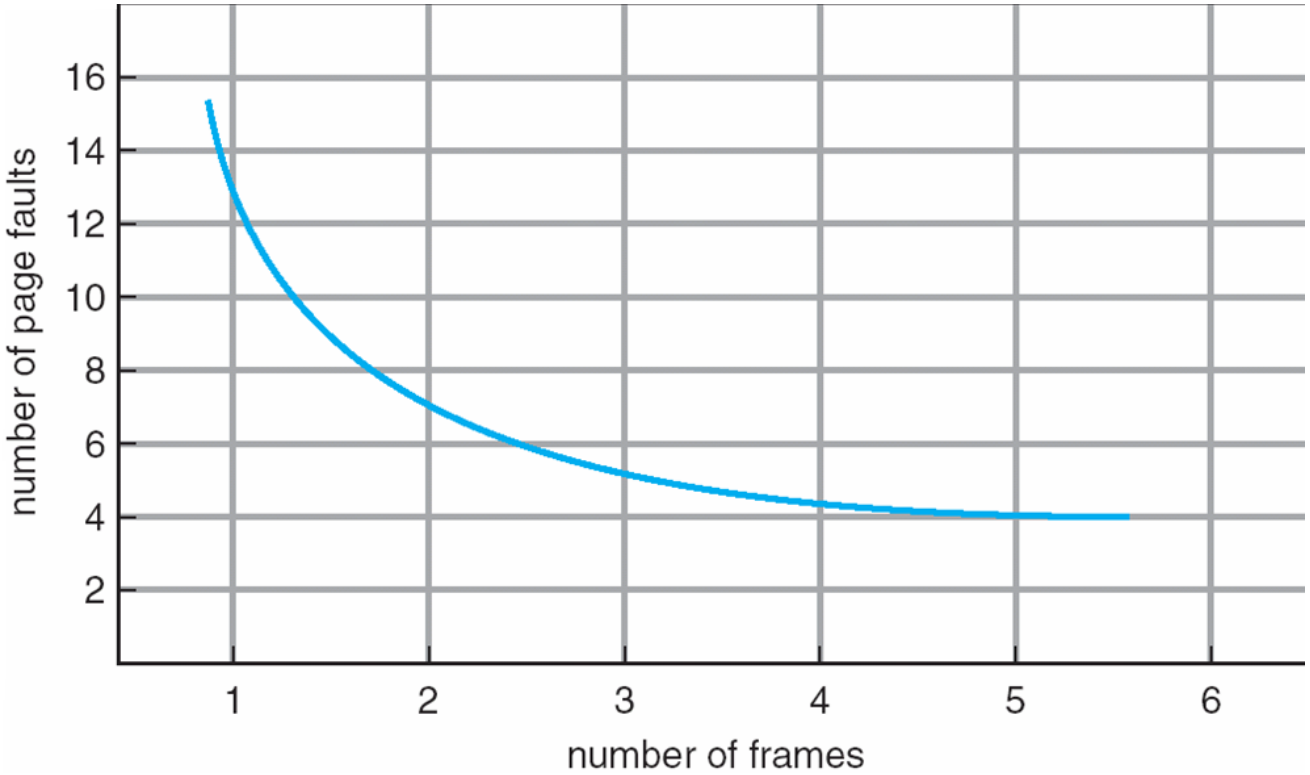
0100, 0432, 0101, 0612, 0102, 0104, 0101, 0611, 0102

- Page size 100B
- Reference string 1, 4, 1, 6, 1, 6

- In all our examples, the reference string is

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

Graph of Page Faults Versus The Number of Frames



First-In-First-Out (FIFO) Algorithm

Associates a time with each frame when the page was brought into memory

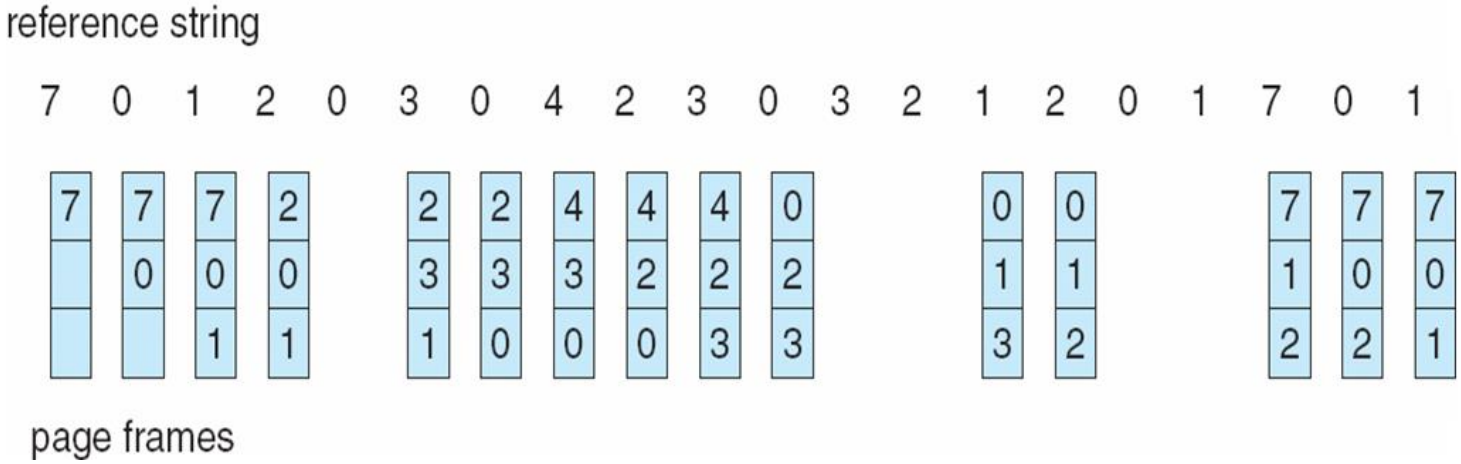
When a page must be replaced, the oldest one is chosen

Reference string: **7,0,1,2,0,3,0,4,2,3,0,3,0,3,2,1,2,0,1,7,0,1**

Limitation:
A variable is initialized early and constantly used

FIFO Page Replacement

3 frames (3 pages can be in memory at a time)



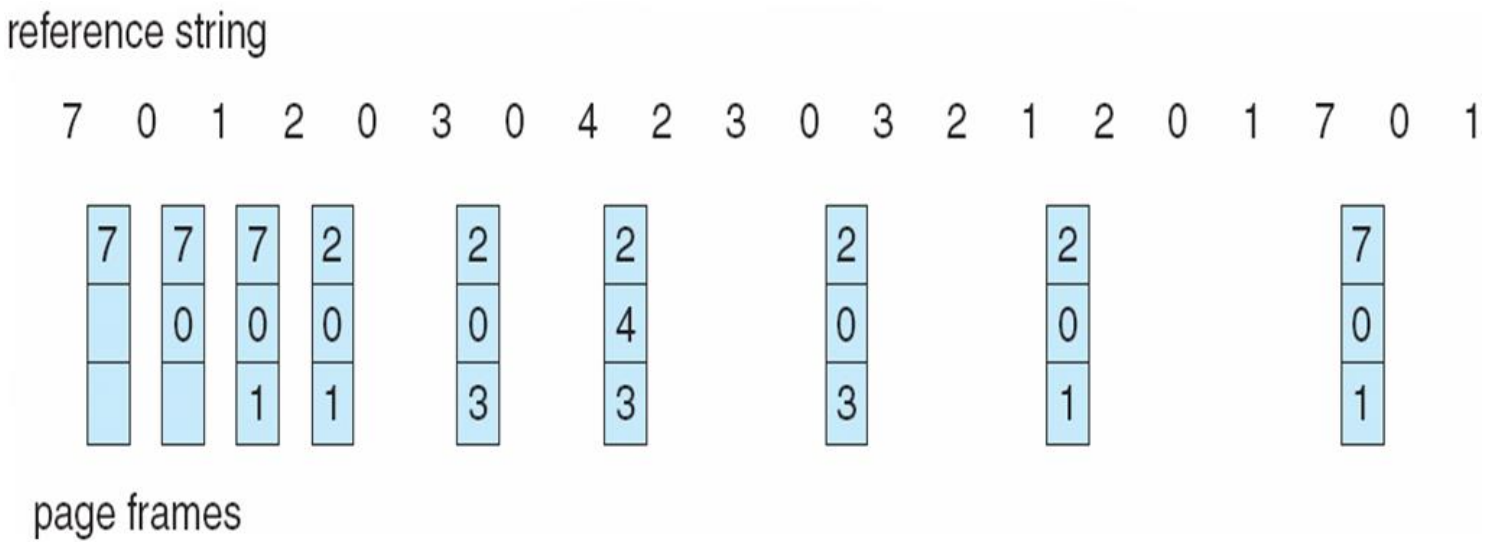
15 page faults

- How to track ages of pages?
  - Just use a FIFO queue to hold all the pages in memory
  - Replace the page at the head
  - Insert at tail

Optimal Algorithm

- Replace page that will not be used for longest period of time
- How do you know this?
  - Can’t read the future
- Used for measuring how well your algorithm performs

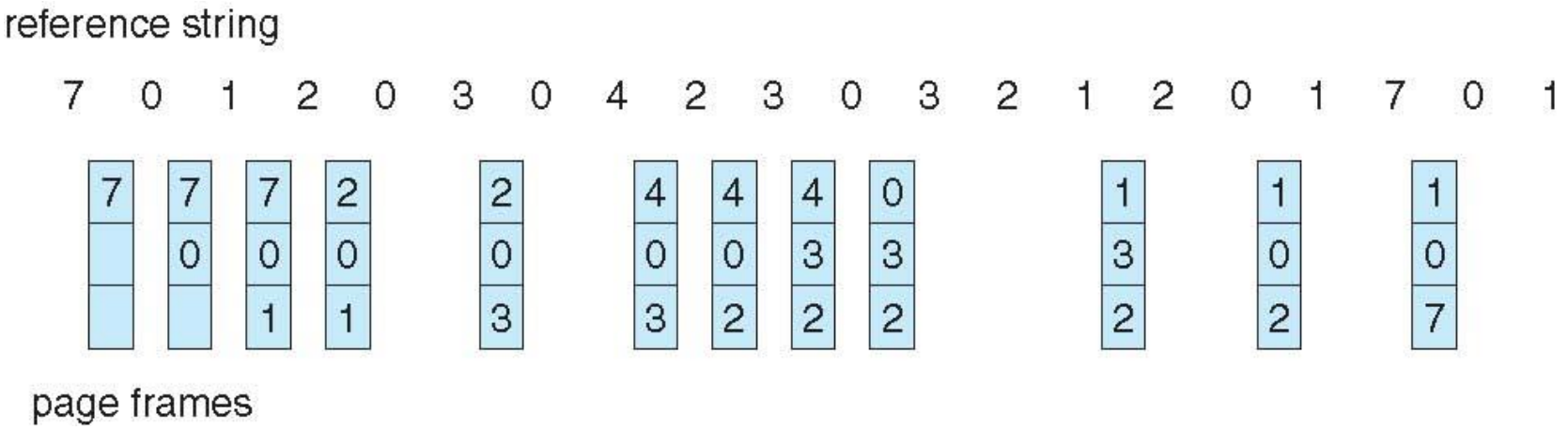
Optimal Page Replacement



9 page faults

Least Recently Used (LRU) Algorithm

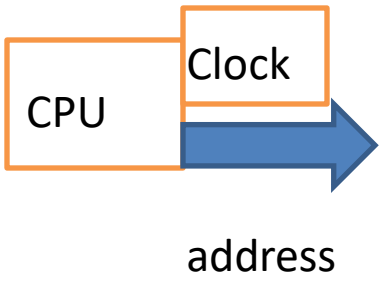
- **Use past knowledge rather than future**
  - Past is the proxy of future
- Replace page that has not been used in the most of the time
- Associate time of last use with each page



- 12 faults – better than FIFO but worse than OPT
- Generally good algorithm and frequently used
- But how to implement?

LRU Algorithm-Implementation

- **Counter implementation**
  - CPU maintains a clock
  - Every page entry has a Time of use;
    - every time page is referenced, copy the clock into the time of use
  - When a page needs to be replaced, look at the "Time of use" to find smallest value
    - Search through table needed

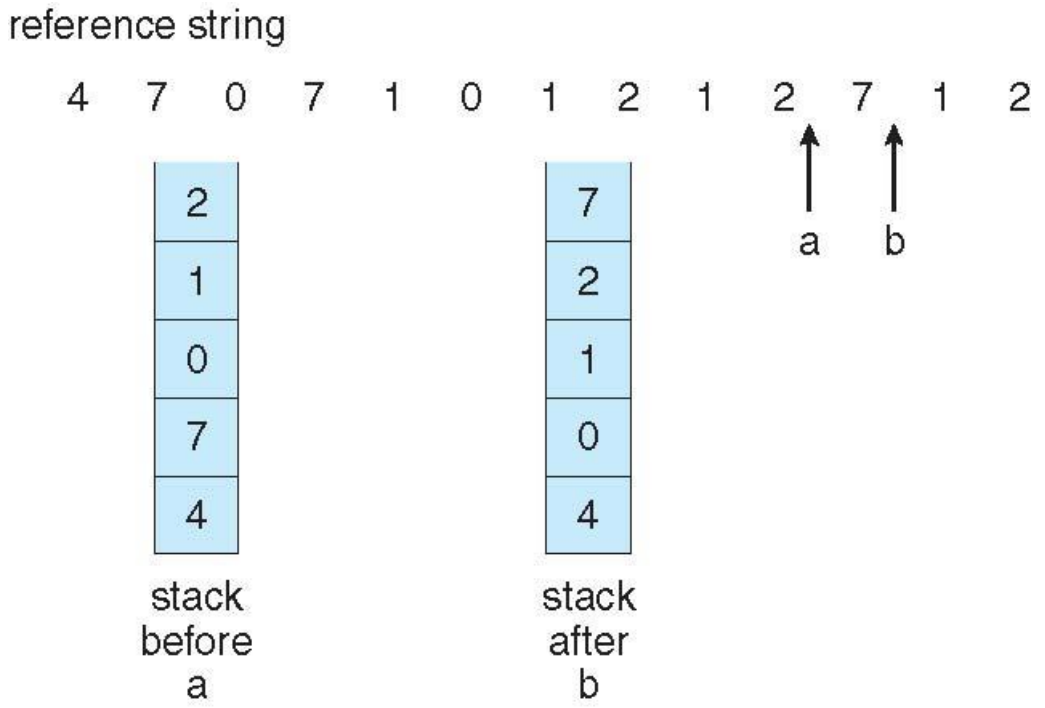


Page table

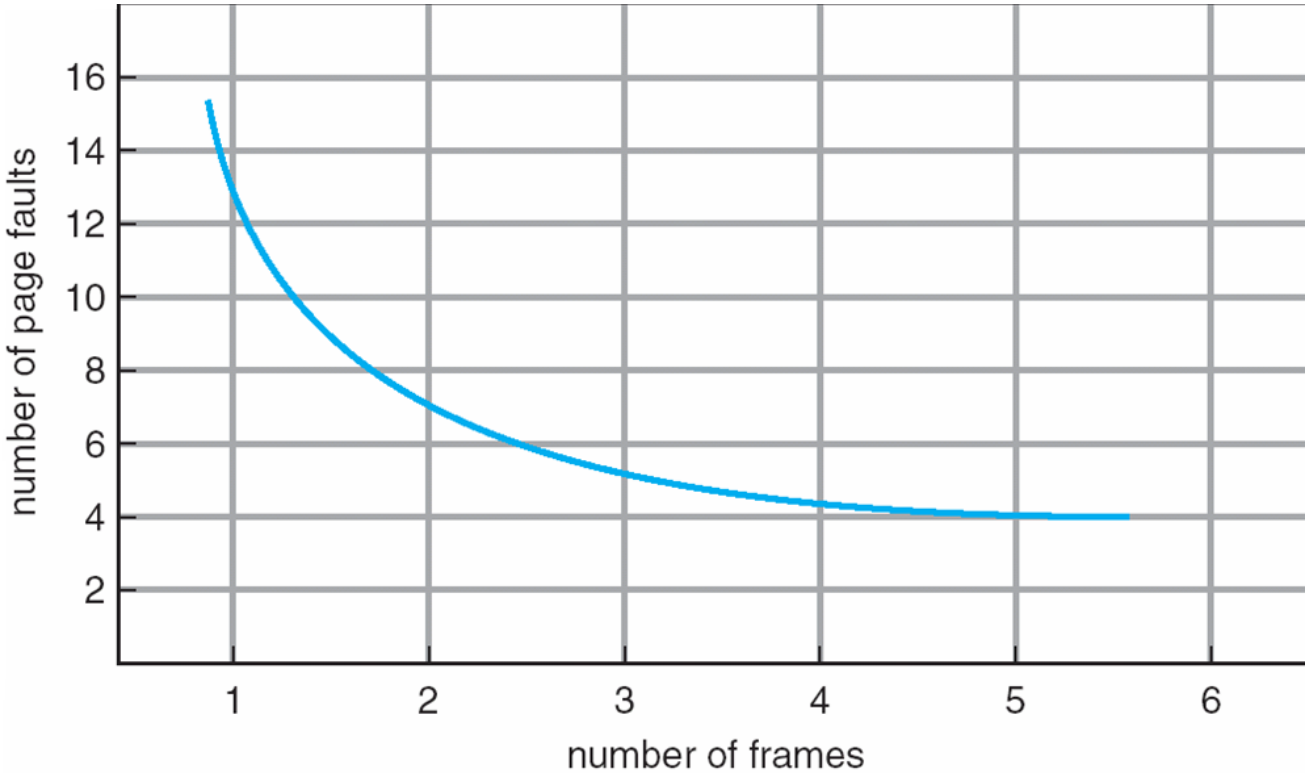
			Time of Use
0			
1			
2			
3			

LRU Algorithm-Implementation

- **Stack implementation**
  - Keep a stack of page numbers in a double linked list form:
  - Page referenced:
    - move it to the top
  - Victim page is the bottom page

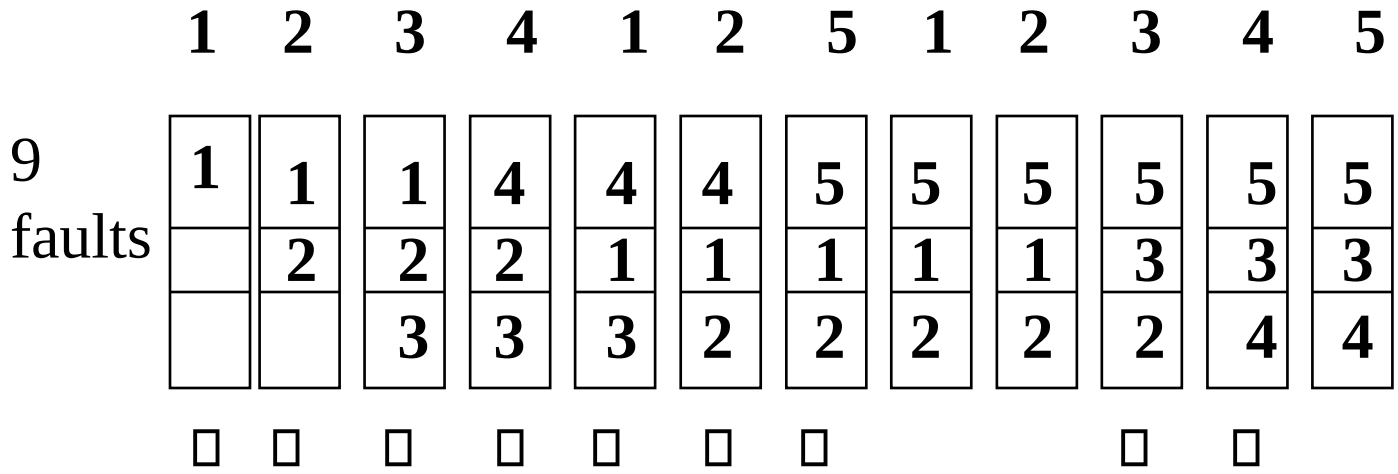


Graph of Page Faults Versus The Number of Frames

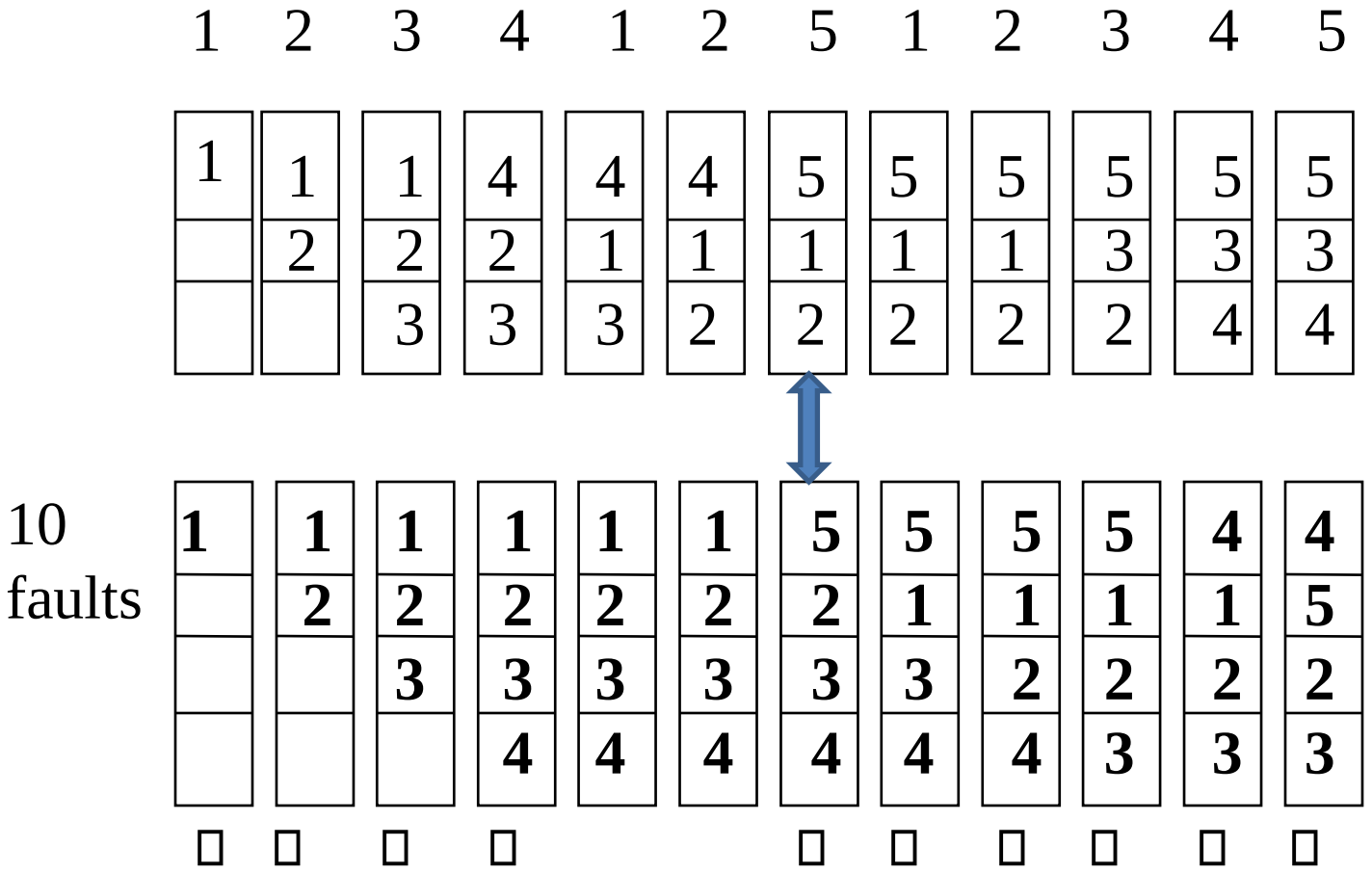


Increase in page frame decreases page fault rate?

FIFO Example – Program - 5 pages, 3 frames of Memory allocated



FIFO Example – Program - 5 pages, 4 frames of Memory allocated



Belady's Anomaly