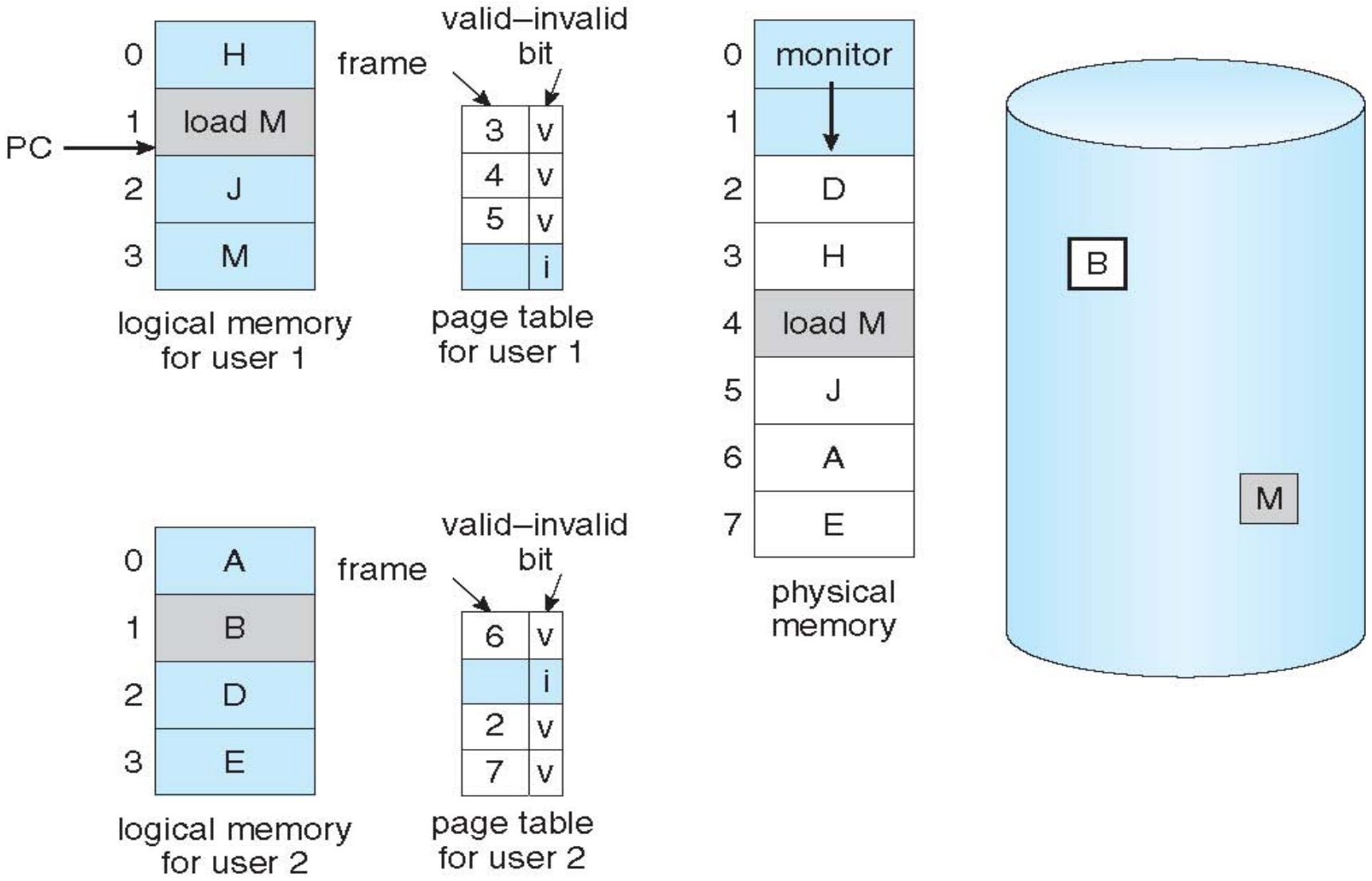
운영체제 13주차

부산가톨릭 대학교, 컴퓨터공학과
변상선

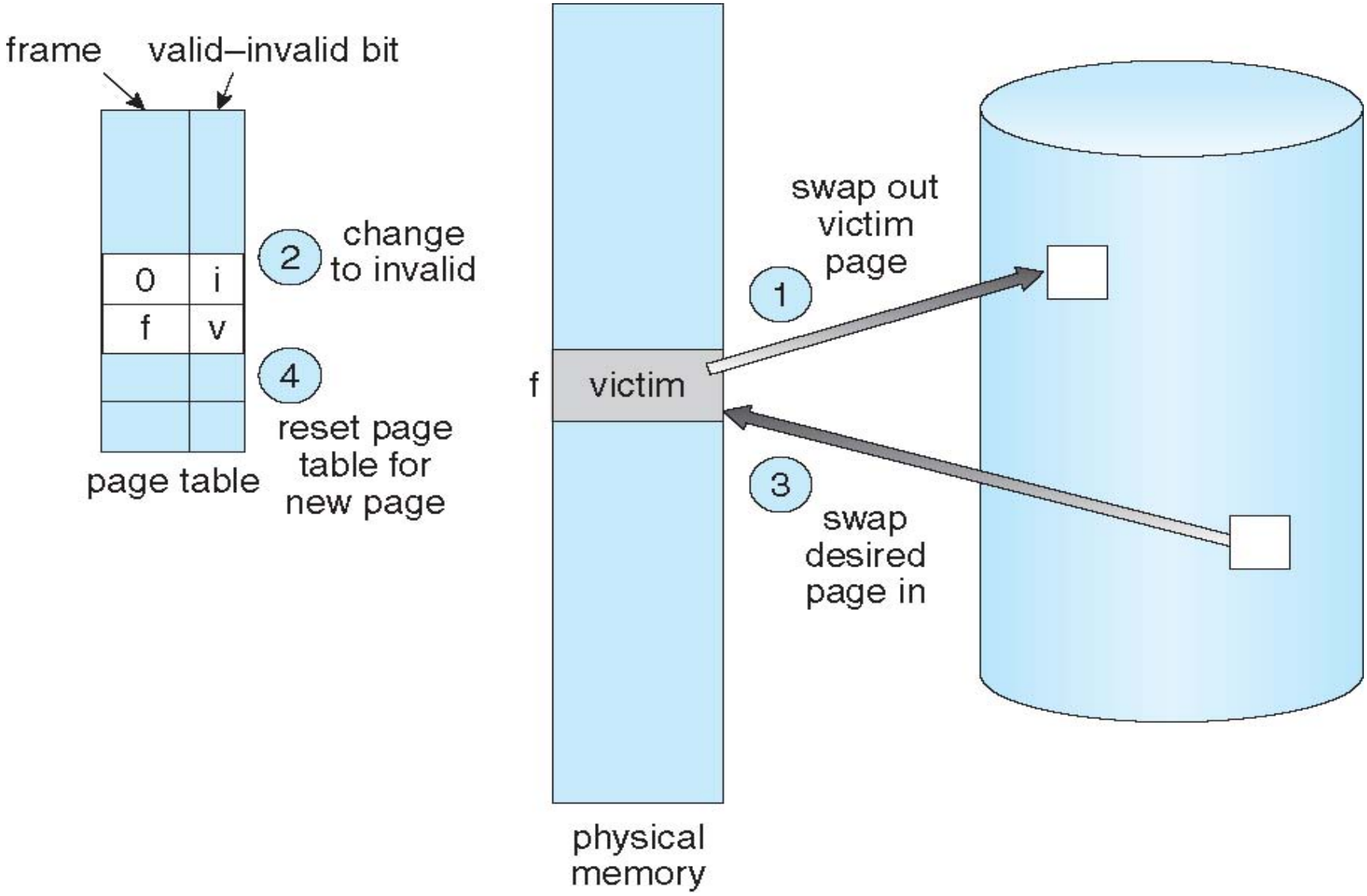
Page replacement

- Find some pages in physical memory, but not really in use, page it out
- 사용자 프로그램에게는 page 시스템이 보이지 않게 한다
- Page fault 가 발생했을 때의 처리
  - 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
    - Write the victim frame to disk if dirty
    3. Bring the desired page into the free frame; update the page and frame tables
    4. Continue the process by restarting the instruction that caused the trap
  - 디스크에 접근이 두번 발생 => 한번은 프레임을 비울 때, 한번을 읽어들일 때

Page replacement



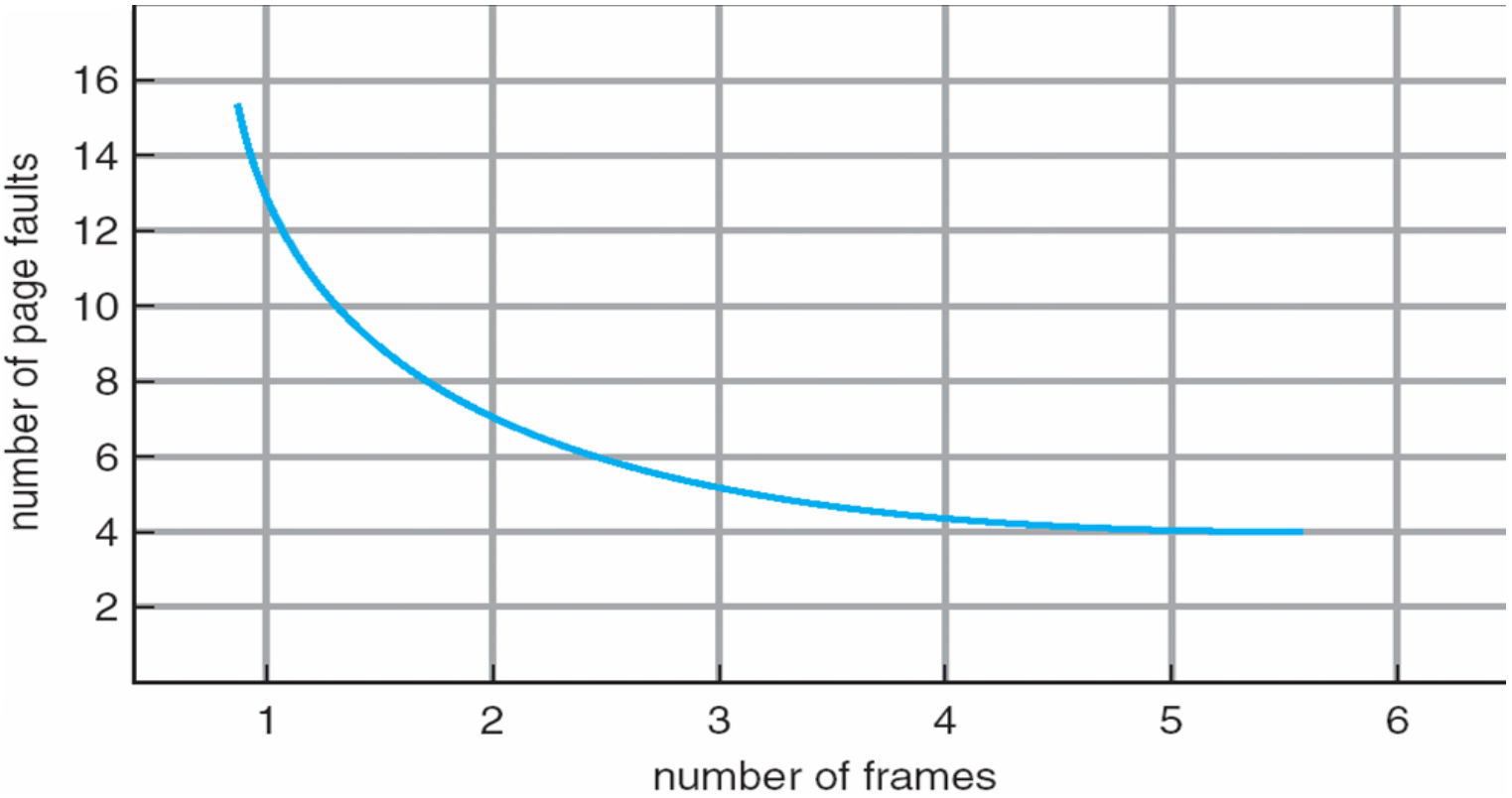
Page replacement



Page and frame replacement algorithm

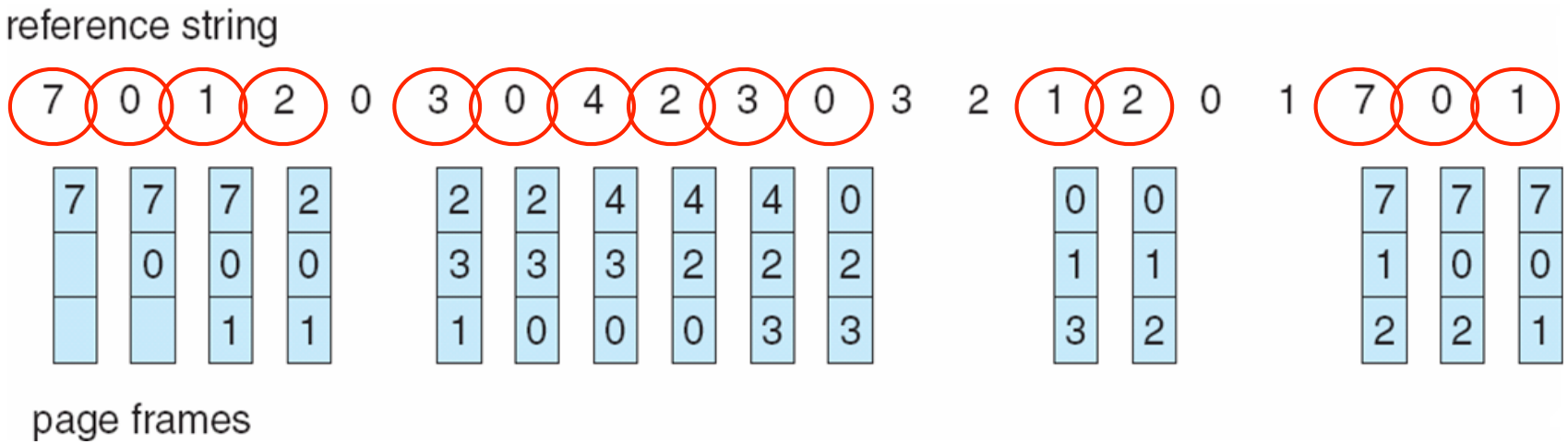
- Frame-allocation algorithm
  - 각 프로세스에게 몇 개의 프레임을 할당할 것인가?
- Page-replacement algorithm
  - 어떤 프레임을 victim으로 선택할 것인가?
  - Page 부재율이 가장 낮은 것이 적합
- 페이지 교체 알고리즘의 성능 평가
  - reference string
    - CPU가 참조하는 페이지 번호의 순서를 나타내는 문자열
  - 페이지 부재의 횟수를 측정

Page fault vs. number of frames

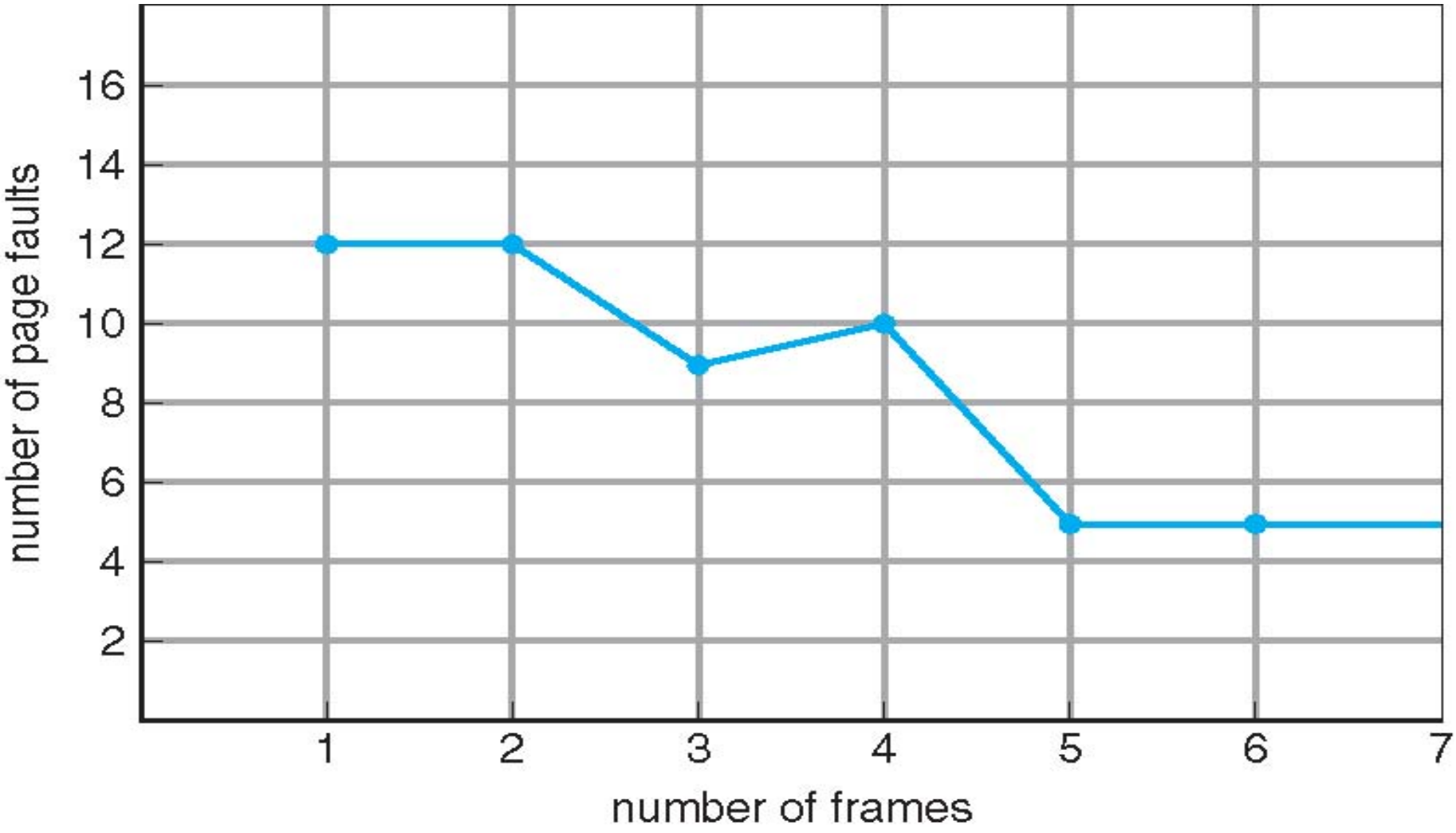


FIFO

- Refrence string: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1
- 3 frames
  - 3 pages can be in memory at a time per process
  - Belady's anomaly
    - 프로세스에게 프레임을 더 주었는데도 페이지 부재가 증가

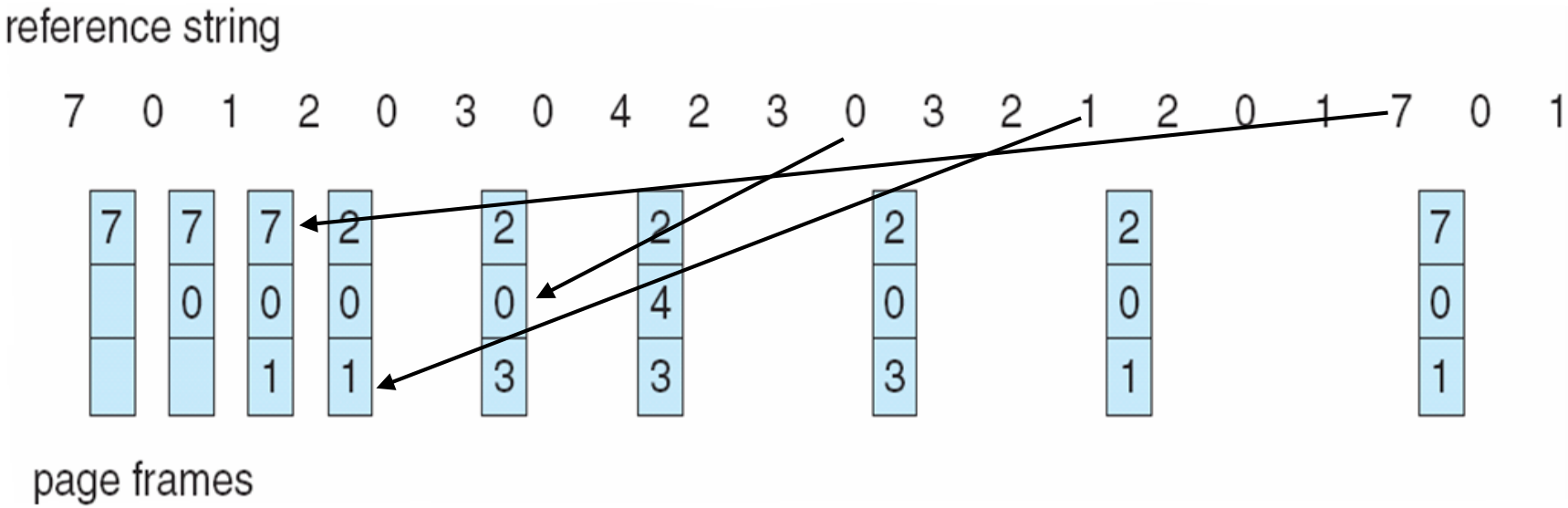


Belady's anomaly



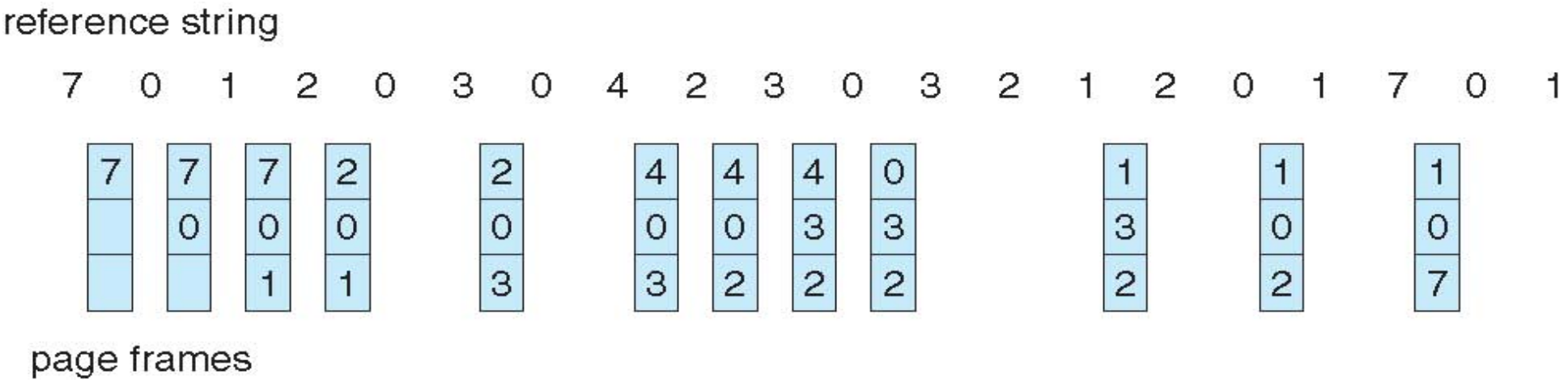
Optimal algorithm

- Replace page that WILL not be used for longest period of time
  - 실제 구현이 불가능
  - 오로지 비교평가 대상으로 사용



LRU

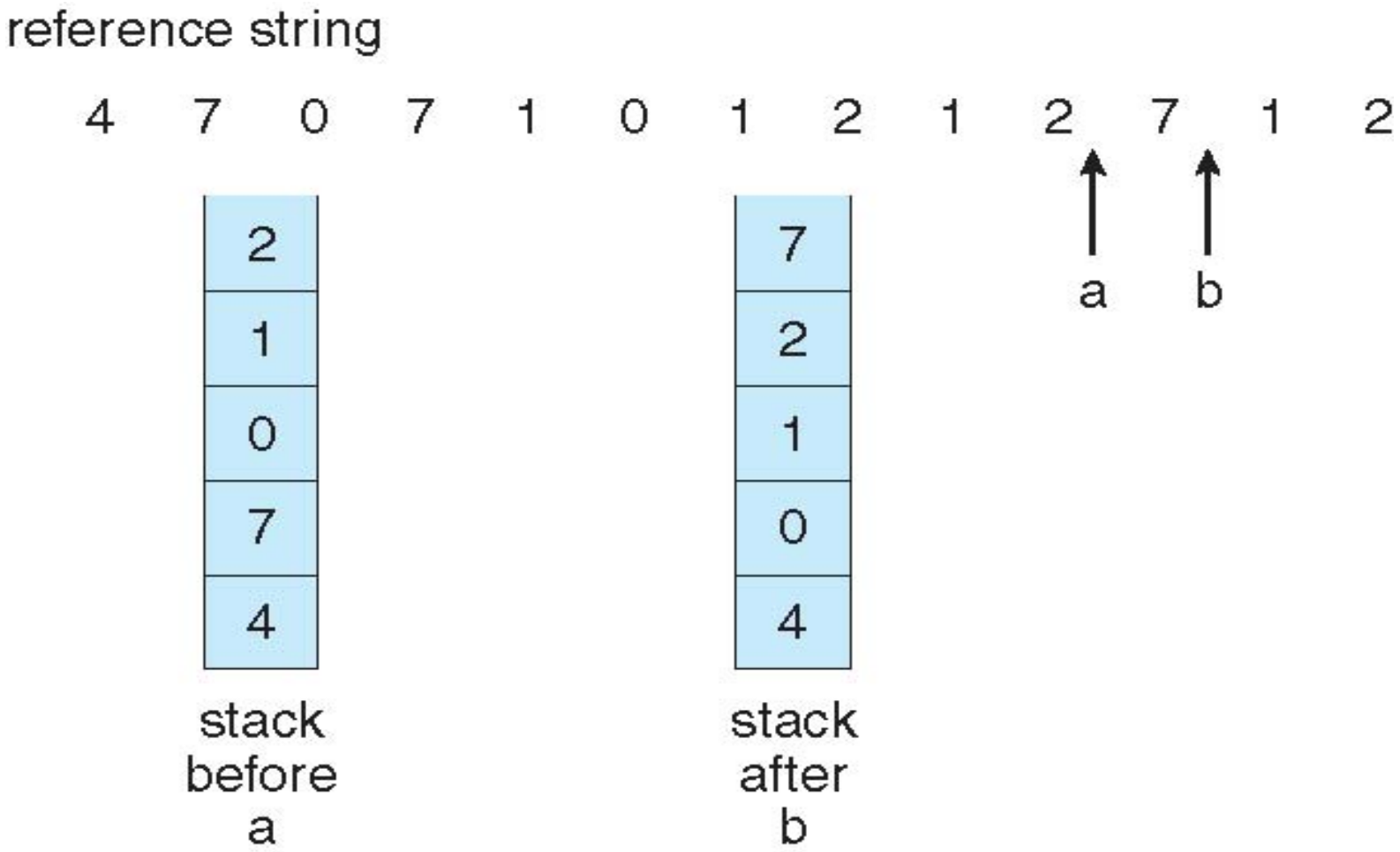
- Use past knowledge rather than future
- 가장 오랫동안 사용되지 않은 페이지를 교체
- No Belady's anomaly



LRU

- Counter implementation
  - 페이지 접근이 발생할 때마다 counter를 유지
  - Counter 값이 가장 적은 페이지를 교체
  - 모든 페이지 테이블을 다 뒤져야 함
- Stack implementation
  - Doubly liked list의 형태로 스택을 유지
  - 스택에 페이지 번호를 저장
  - 접근된 페이지의 번호를 top으로 이동
  - 항상 bottom에 위치한 page가 victim
  - 페이지 테이블을 뒤질 필요가 없음
- 반드시 MMU/TLB의 도움을 받아야 함
  - Page table을 뒤지거나 stack을 뒤지는 행위를 CPU가 메모리에서 (소프트웨어를 통해) 직접 수행하게 되면 지나친 메모리 참조로 인해 시스템 성능이 매우 떨어짐

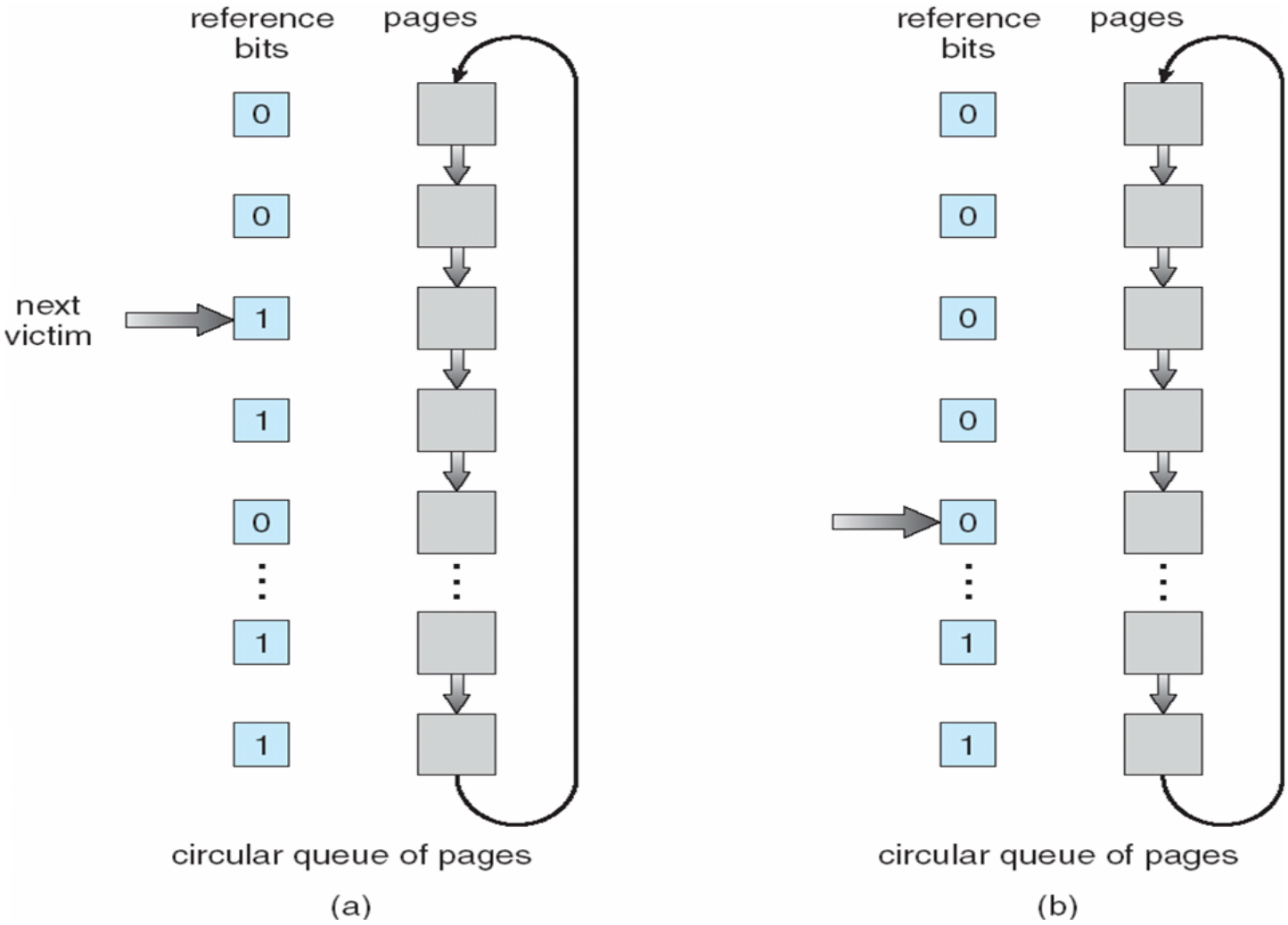
LRU - stack implementation



LRU approximation algorithm

- MMU와 TLB를 통한 LRU의 동작은 여전히 느림
- Additional-reference bits algorithm
  - Access history
  - 초기 모든 페이지는 = 00000000 (8 bit 인 경우)
  - 참조가 발생하면 해당 페이지의 reference bits = 10000000
  - 가장 작은 reference bits를 갖는 페이지가 victim
- Second-chance algorithm
  - 기본적으로 FIFO
  - 도착시간이 가장 오래된 reference bit가 0이면 교체, 1이면 0으로 바꾸고 (도착시간이 현재로 재설정) 메모리에 유지

Second-chance algorithm



Counting-based algorithm

- 참조횟수를 유지
- LFU
  - 가장 적게 참조된 페이지를 교체
- MFU
  - 가장 많이 참조된 페이지를 교체
    - 이미 많이 참조되었으므로 앞으로 참조될 확률이 적다고 판단
- LFU, MFU 모두 성능이 좋지 않음

Page-buffering algorithm

- Free frame의 pool을 항상 유지
- 페이지 부재시 교체될 페이지를 디스크에 쓰기 전에 free frame 에 할당
- 프로세스가 가능한한 빨리 시작하도록 함
- 그리고, 교체될 페이지를 디스크에 다 쓰고나면 그 페이지가 free frame pool에 추가
- 변형된 방법
  - Dirty page의 리스트를 항상 유지
  - 디스크가 유휴상태일때 몰아서 디스크에 dirty page를 기록하고 변경 비트를 0으로 reset, 재차 변경이 발생하면 변경 비트는 1
  - 페이지 부재가 발생되었을 때, 변경비트가 0인거 아무나 overwrite
- 또 다른 변형된 방법
  - Free frame pool을 유지하되 free 되기 전에 어떤 프로세스가 사용했는지 기록 (페이지 번호와 함께)
  - 만약, 이전에 사용했던 프로세스가 같은 페이지번호를 갖는 frame이 pool에서 발견된다면, 그대로 할당

Application and page replacement

- 특정 응용을 위한 페이지 교체 알고리즘
  - Database
    - MFU가 LRU나 LFU 보다 더 효과적일 수 있음
    - 대량의 데이터를 순차적으로 메모리로 읽어들인 후 같은 요구가 들어왔을 때, 재차 읽어들이게 되므로

Allocation of frames

- 각 프로세스가 가져야할 최소 프레임의 수
  - 하나의 명령어가 참조하게 될 페이지가 모두 물리 메모리에 있어야 한다
  - 하나의 주소만을 피연산자로 갖고, 간접주소 지정을 허용하는 경우
    - 3개의 프레임이 필요
  - 결과적으로 명령어 아키텍쳐 (주소 피연산자 개수, 간접주소 지정의 깊이 등)에 의해 결정

Fixed allocation

- Equal allocation
  - 모든 프로세스에게 동일하게 할당
    - 93개의 프레임, 5개의 프로세스
      - 각 프로세스가 18개의 프레임, 나머지 3개는 free frame으로 보관
- Proportional allocation
  - 62 free frames, 1KB frame size, 10KB process and 127KB process
    - 31개씩 할당하면 10KB process에게 할당된 21개는 낭비
    - 62 * 10KB / (10KB + 127KB) = 4
    - 62 * 127KB / (10KB + 127KB) = 57
- 두 방법 모두 동시에 실행중인 프로세스의 수가 많아지면 덜 받고, 적어지면 더 받게 된다

Global vs. local replacement

- 전역교체
  - 페이지 부재시 victim을 모든 프로세스의 프레임을 대상으로 찾는것
  - 한 프로세스에게 할당된 프레임의 수가 변할 수 있음
  - 더 나은 메모리 사용율, 하지만, 각 프로세스의 페이지 부재율이 시시각각 변함
- 지역교체
  - victim을 자신에게 할당된 프레임에서만 찾는것
  - 한 프로세스에게 할당된 프레임 수가 고정됨
  - 프로세스의 페이지 부재율이 일정, 낮은 메모리 사용율
  - 따라서, 전역교체를 사용함

Non-uniform memory access (NUMA)

- 클러스터링 시스템에서 CPU는 다른 컴퓨터에 있는 메모리의 접근 보다 같은 컴퓨터에 있는 메모리 접근이 더 빠름
- Frame 할당과 프로세스 스케쥴링
  - 가급적 같은 컴퓨터에 있는 메모리에서 프레임 할당
  - 같은 컴퓨터에서 계속 실행되도록 함 => cache affinity