

# Virtual Memory

2018년 5월 21일 월요일 오후 3:40

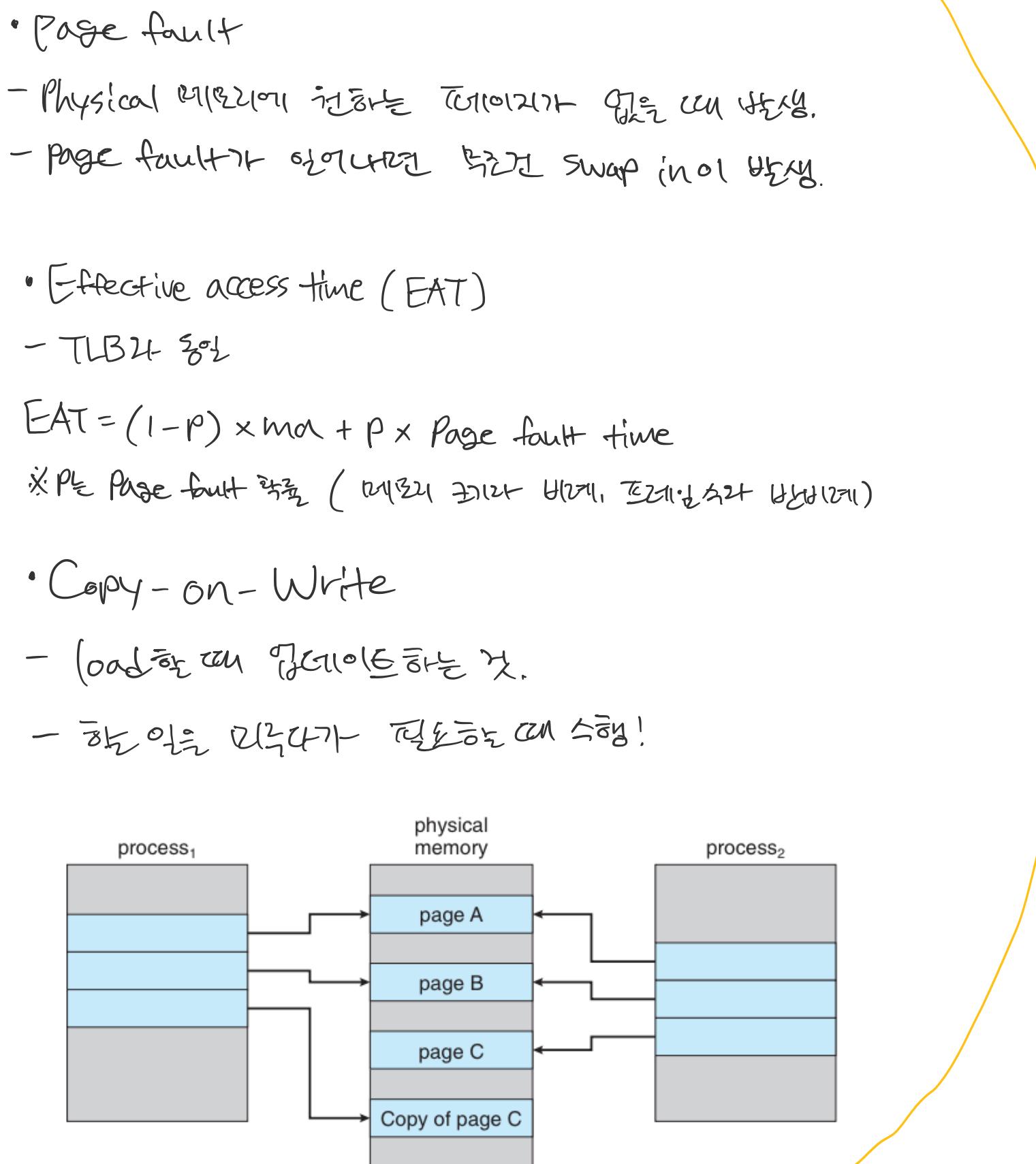
- 주소공간 메모리는 크기가 작다.
- 메모리를 커버시켜줄 핵심하는 것이 Virtual memory.
- Swap이 중요.

## • Background

- 한 프로세스 안에도 여러 사용하는 부분이 있고, 그중지 않은 부분이 있을 때.  
! 여러 데이터를 일관성 있는 메모리에 코드하고 나머지는 Virtual 메모리에 한다.

## • Demand Paging

- 요청성을 고려한 환경에서의 접근을 코드하는 방식



## • Page fault

- Physical 메모리에 현하는 데이터가 없을 때 발생.
- Page fault가 일어나면 블록과 swap-in이 발생.

## • Effective access time (EAT)

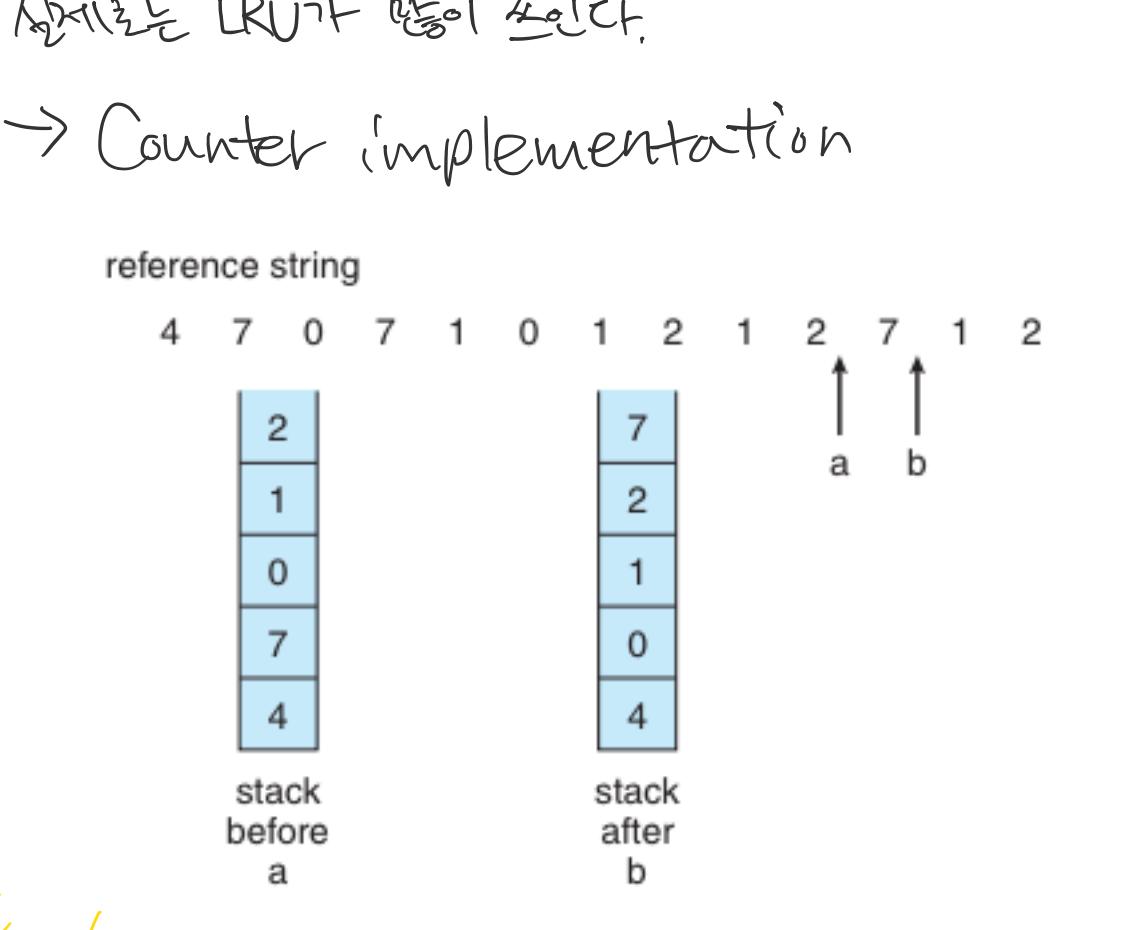
- TLBhit 동안

$$EAT = (1 - p) \times \text{Mol} + p \times \text{Page fault time}$$

\* p: Page fault 확률 (메모리 카터리, 프레임 스왑, 디스크 접근)

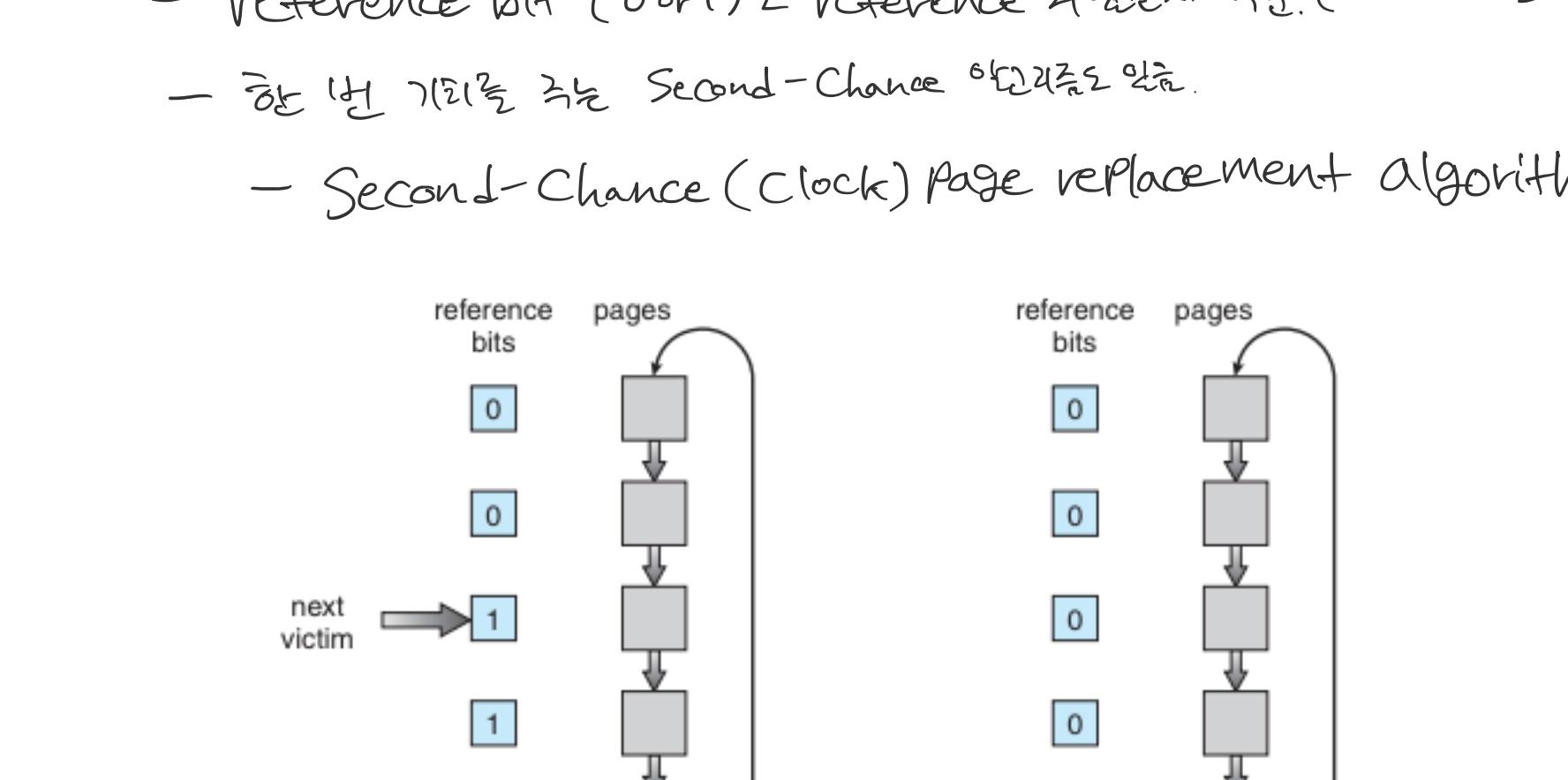
## • Copy-on-Write

- load할 때 복제하는 것.
- 할 일을 미루다가 필요할 때 스탑!



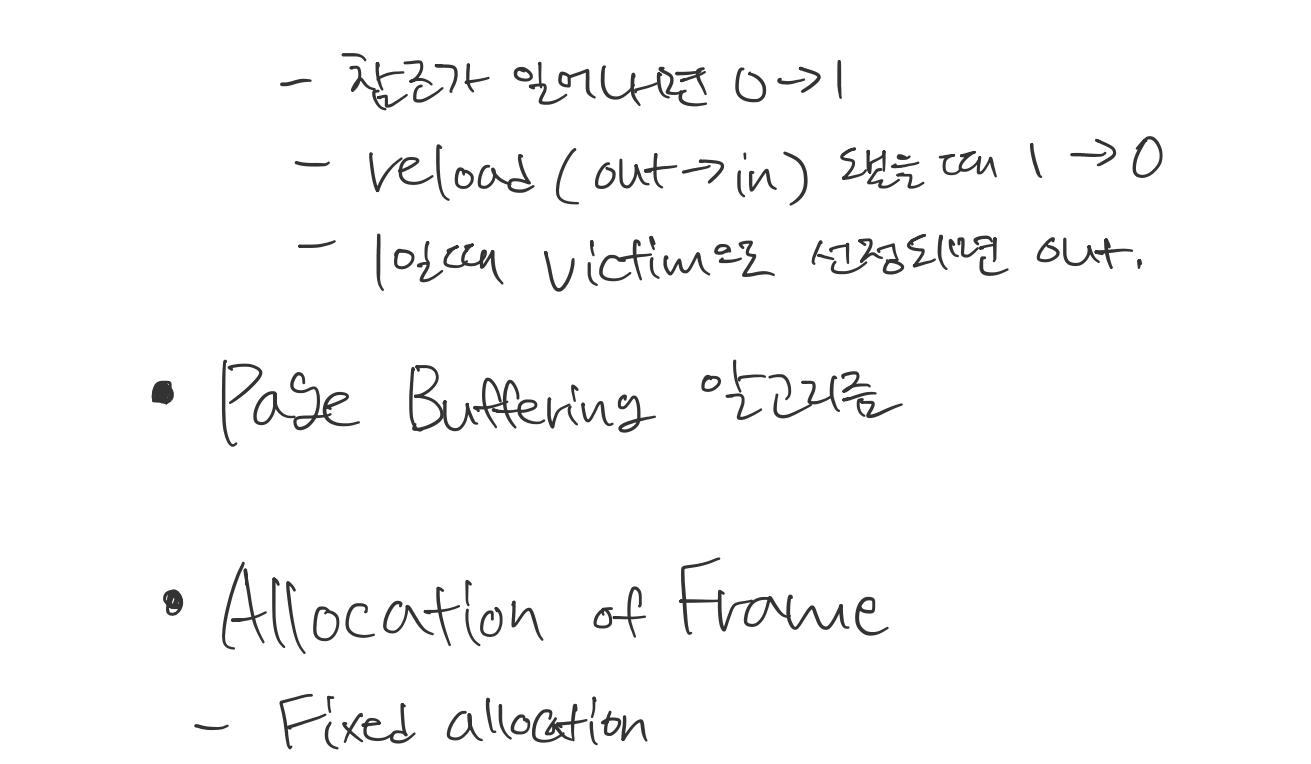
## • Page replacement

- FIFO: 제일 먼저 코드된 프레임을 Swap-out

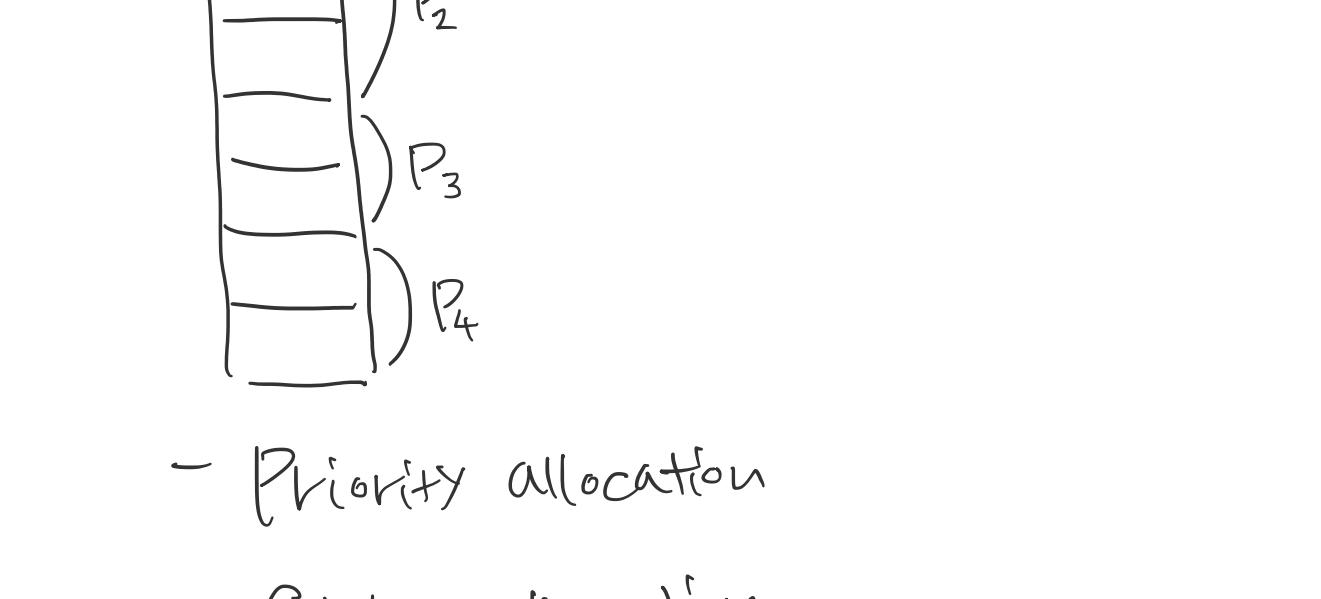


- Optimal: 가장 오래 사용되지 않은 프레임을 Swap-out

↳ SPT(Shortest Job First) 예측.



- Least recently used (LRU):



- ① 구현할 수 없음. - 한정 속의 힘든 알고리즘.
- ② 대신 자주 가는 주현한 알고리즘의 optimization에 의해 어떤지 비교하는 때 사용한다.

실제에는 LRU가 많이 쓰인다.

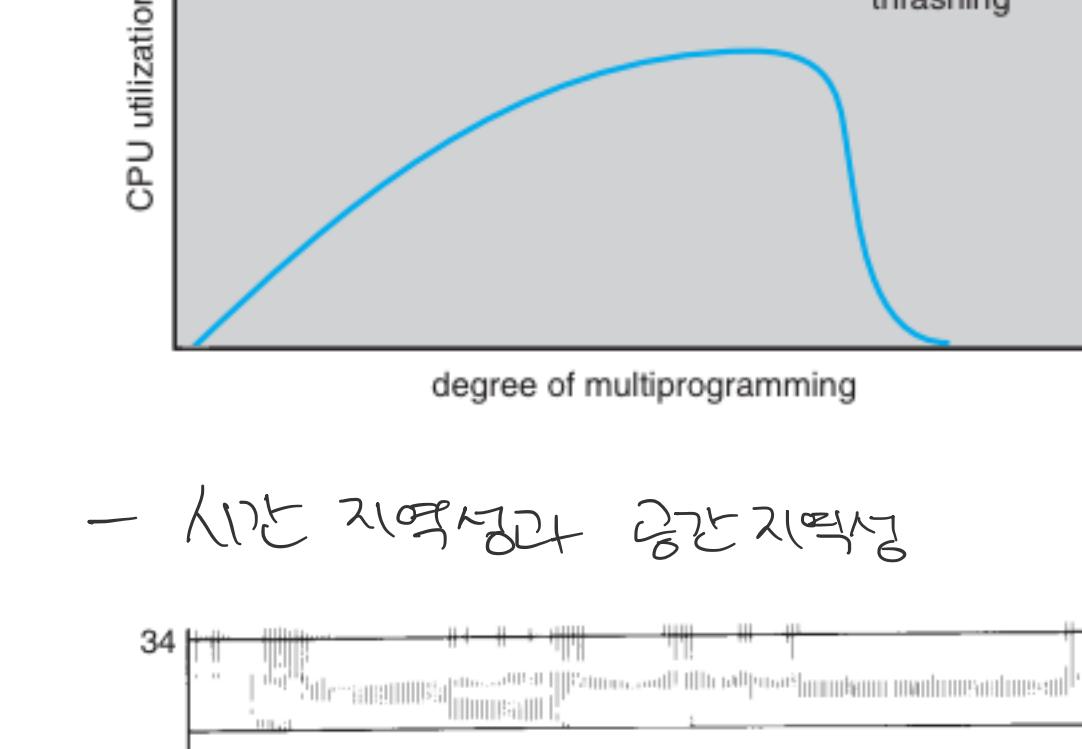
X: temp 값이 같거나

$$b = b + a$$

$$a = b - a$$

$$b = b - a$$

## → Counter implementation

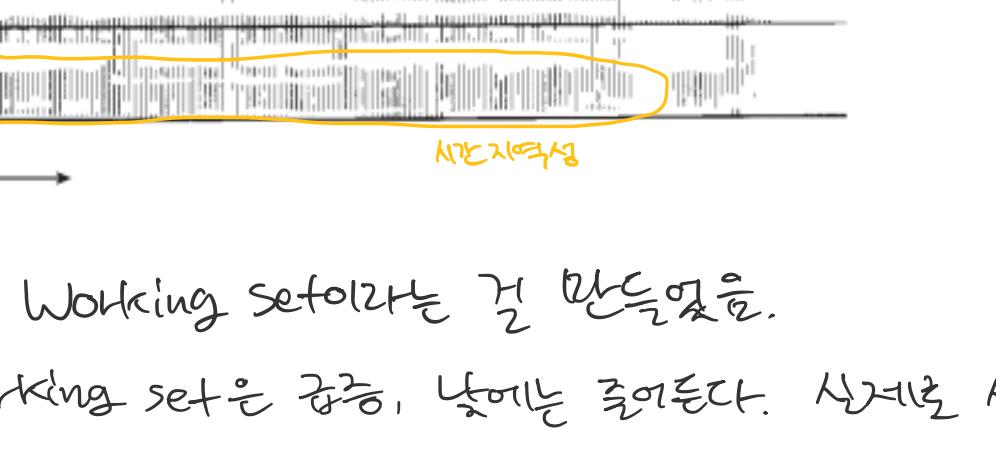


- LFU: 최근 참조가 가장 많은 것, 대체로 효율적.
- MFU: 최근 참조가 가장 많은 것, 한 번 본 데이터를 다시 참조했을 때 빈출이 난을 때 (정렬, 배포 등) 효과적.

→ Windows에서는 LRU Approximation을 사용.

- 1~2번 reference되자 압축되면 Victim이 될 가능성↑
- Reference bit (0 or 1)로 reference 되었는지 구분. (Counter보다 작은 용량)
- 한 번 기록을 주는 Second-Chance 알고리즘을 갖는다.

- Second-Chance (Clock) Page replacement algorithm



- 참조가 일어나면 0 → 1
- reload (out → in) 때는 1 → 0
- 1일 때 Victim으로 선정되면 out.

## • Page Buffering 알고리즘

## • Allocation of Frame

- Fixed allocation



- Priority allocation

- Global allocation

- Non-Uniform memory access

## • Thrashing

- 프로세스가 충분한 데이터를 가지 못하면 Page fault 확률이 높아진다.
- 동시에 사용하는 프로세스 수가 늘어나면 CPU utilization이 높아지는데, 이 현상을 Thrashing이라고 보는.



- 시간 지역성과 공간 지역성

↳ 시간 Working set은 짧은, 낮에는 줄어든다. 시스템은 서버가 여기에 맞춰 브레이크 앤드 푸시를 한다.

## • Page fault frequency

- 네트워크도 한 번하고 네트워크도 한 번.

## • Allocating Kernel Memory

- 접근이 일어나는 때마다 page table을 하지 않는다.

- pages는 page table overhead가 있다.

- Contiguous를 사용하면 초기화 초기화가 빠르다.

- **Buddy System**을 사용

## → Buddy System allocator

- 신시간으로 compaction이 일어나는 시스템.

## • Page Buffering 알고리즘

## • Allocation of Frame

- Fixed allocation



- Priority allocation

- Global allocation

- Non-Uniform memory access

## • Thrashing

- 프로세스가 충분한 데이터를 가지 못하면 Page fault 확률이 높아진다.
- 동시에 사용하는 프로세스 수가 늘어나면 CPU utilization이 높아지는데, 이 현상을 Thrashing이라고 보는.



- 시간 지역성과 공간 지역성

↳ 시간 Working set은 짧은, 낮에는 줄어든다. 시스템은 서버가 여기에 맞춰 브레이크 앤드 푸시를 한다.

## • Page fault frequency

- 네트워크도 한 번하고 네트워크도 한 번.

## • Allocating Kernel Memory

- 접근이 일어나는 때마다 page table을 하지 않는다.

- pages는 page table overhead가 있다.

- Contiguous를 사용하면 초기화 초기화가 빠르다.

- **Buddy System**을 사용

## → Buddy System allocator

- 신시간으로 compaction이 일어나는 시스템.

## • Page Buffering 알고리즘

## • Allocation of Frame

- Fixed allocation



- Priority allocation

- Global allocation

- Non-Uniform memory access

## • Thrashing

- 프로세스가 충분한 데이터를 가지 못하면 Page fault 확률이 높아진다.
- 동시에 사용하는 프로세스 수가 늘어나면 CPU utilization이 높아지는데, 이 현상을 Thrashing이라고 보는.



- 시간 지역성과 공간 지역성

↳ 시간 Working set은 짧은, 낮에는 줄어든다. 시스템은 서버가 여기에 맞춰 브레이크 앤드 푸시를 한다.

## • Page fault frequency

- 네트워크도 한 번하고 네트워크도 한 번.

## • Allocating Kernel Memory

- 접근이 일어나는 때마다 page table을 하지 않는다.

- pages는 page table overhead가 있다.

- Contiguous를 사용하면 초기화 초기화가 빠르다.

- **Buddy System**을 사용

## → Buddy System allocator

- 신시간으로 compaction이 일어나는 시스템.

## • Page Buffering 알고리즘

## • Allocation of Frame

- Fixed allocation



- Priority allocation

- Global allocation

- Non-Uniform memory access

## • Thrashing

- 프로세스가 충분한 데이터를 가지 못하면 Page fault 확률이 높아진다.
- 동시에 사용하는 프로세스 수가 늘어나면 CPU utilization이 높아지는데, 이 현상을 Thrashing이라고 보는.



- 시간 지역성과 공간 지역성

↳ 시간 Working set은 짧은, 낮에는 줄어든다. 시스템은 서버가 여기에 맞춰 브레이크 앤드 푸시를 한다.

## • Page fault frequency

- 네트워크도 한 번하고 네트워크도 한 번.

## • Allocating Kernel Memory

- 접근이 일어나는 때마다 page table을 하지 않는다.

- pages는 page table overhead가 있다.

- Contiguous를 사용하면 초기화 초기화가 빠르다.</