22: swapping policy

First of all, cache management

- 교체 정책의 목표
  - 캐시 미스 횟수를 최소화
- measurement: 평균 메모리 접근 시간(average memory access time, AMAT)

$$AMAT = (P_{Hit} \cdot T_M) + (P_{Miss} \cdot T_D)$$

최적 교체 정책

- 가장 나준에 접근될 페이지를 교체하는 것이 최적이다
- 실제 구현은 어려움

접근	히트/미스?	내보냄	결과적인 캐시 상태
0	미스		0
1	미스		0, 1
2	미스		0, 1, 2
0	히트		0, 1, 2
1	히트		0, 1, 2
3	미스	2	0, 1, 3
0	히트		0, 1, 3
3	히트		0, 1, 3
1	히트		0, 1, 3
2	미스	3	0, 1, 2
1	히트		0, 1, 2

〈그림 25.1〉 최적의 교체 정책의 흐름

FIFO

- first in first out

접근	히트/미스?	내보냄	결과적인 캐시 상태
0	미스		선입 0
1	미스		선입 0, 1
2	미스		선입 0, 1, 2
0	히트		선입 0, 1, 2
1	히트		선입 0, 1, 2
3	미스	0	선입 1, 2, 3
0	미스	1	선입 2, 3, 0
3	히트		선입 2, 3, 0
1	미스	2	선입 3, 0, 1
2	미스	3	선입 0, 1, 2
1	히트		선입 0, 1, 2

〈그림 25.2〉 **FIFO 정책의 흐름**

Random select policy

- 불안정함

접근	히트/미스?	내보냄	결과적인 캐시 상태
0	미스		0
1	미스		0, 1
2	미스		0, 1, 2
0	히트		0, 1, 2
1	히트		0, 1, 2
3	미스	0	1, 2, 3
0	미스	1	2, 3, 0
3	히트		2, 3, 0
1	미스	3	2, 0, 1
2	히트		2, 0, 1
1	히트		2, 0, 1

〈그림 25.3〉 **무작위 선택 정책의 흐름**

Least Recently Used(LRU)

- 가장 오래 전에 사용했던 페이지 교체

접근	히트/미스?	내보냄	결과적인 캐시 상태
0	미스		LRU 0
1	미스		LRU 0, 1
2	미스		LRU 0, 1, 2
0	히트		LRU 1, 2, 0
1	히트		LRU 2, 0, 1
3	미스	2	LRU 0, 1, 3
0	히트		LRU 1, 3, 0
3	히트		LRU 1, 0, 3
1	히트		LRU 0, 3, 1
2	미스	0	LRU 3, 1, 2
1	히트		LRU 3, 2, 1

〈그림 25.5〉 **LRU 정책의 흐름**

구현: LRU

- 가장 오래된 페이지를 찾으려면 O(log n)의 시간복잡도가 소요됨→ 이거도 비싸다!
- 근사해보자

시계 알고리즘

1. 현재 바늘이 가리키는 page P의 use bit가 1인가?
    a. 예
        i. use bit를 0으로
        ii. 바늘을 P+1로
    b. 아니오
        i. 너 교체.

Dirty page 고려

- 변경된 페이지는 내보내려면 변경 내용을 저장해야함
    - 비쌈
    - modified bit 추가해서 얘도 같이 보자

Thrashing

- 프로세스 요구 메모리 > 가용 물리 메모리일 때 계속 페이징하는 경우

- 다른 프로세스 죽여서 공간 확보하는 정책
  - 자세한건 찾아봐서 ..