1 가상기억장치 관리 기법

1 가상기억장치 관리 기법

1 가상 기억장치의 개념

- 사용자와 논리적 주소를 물리적으로 분리하여 사용자가 메인 메모리 용량을 초과한 프로세스에 주소를 지정해서 메모리를 제한 없이 사용할 수 있도록 하는 것
- 프로그램 전체를 동시에 실행하지 않으므로 요구한 메모리 전체가 아닌 일부만 적재해도 실행 가능

1 가상기억장치 관리 기법

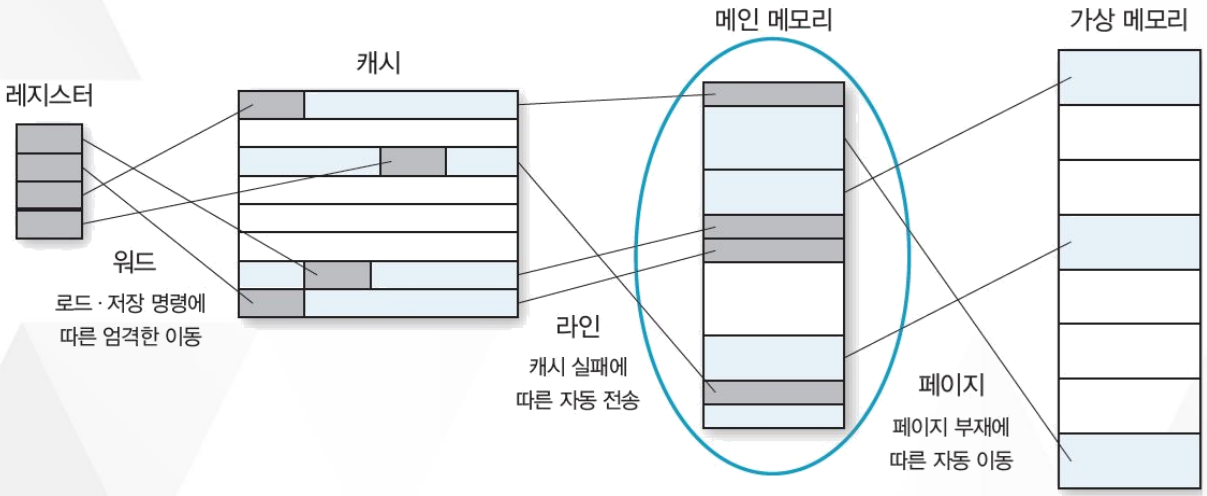
1 가상 기억장치의 개념

- 활동 영역을 메인 메모리에 유지하면서 필요할 때는 디스크와 메모리 사이에 프로세스 코드와 데이터 저장, 다시 자동으로 전송하는(스왑 인, 스왑 아웃) 과정을 거쳐 프로세스를 재할당, 디스크에 저장된 주소 공간은 캐시로 처리하여 메인 메모리 효율적 사용 가능

- 메인 메모리의 제한된 용량과 중첩 사용 문제 해결

1 가상기억장치 관리 기법

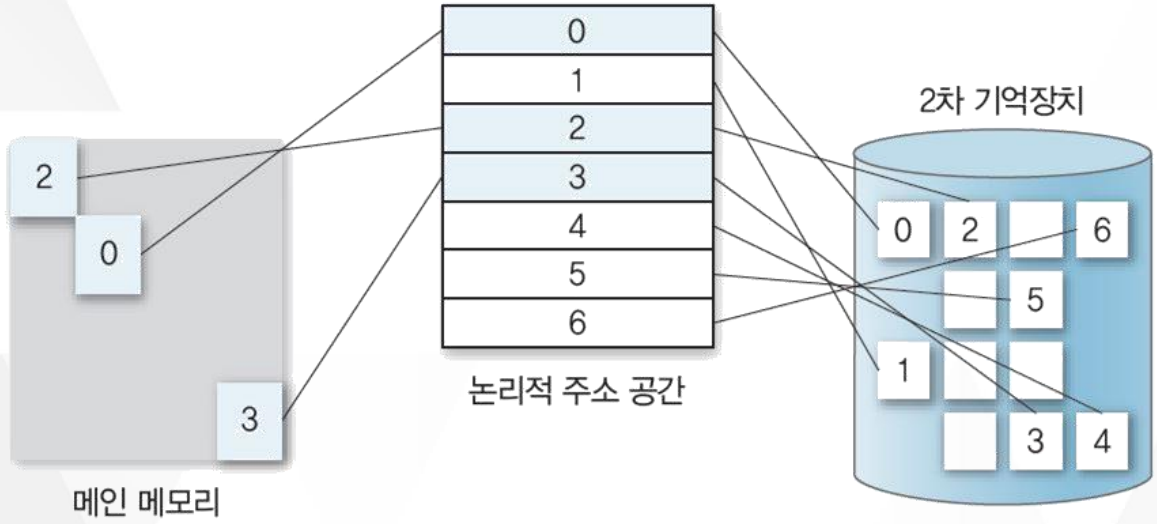
2 페이징으로 구현한 가상메모리



※출처: 그림으로 배우는 구조와 원리 운영체제, 구현회, 한빛아카데미, 2016

1 가상기억장치 관리 기법

3 가상 메모리를 이용한 효율적인 메인 메모리 운영



※출처: 그림으로 배우는 구조와 원리 운영체제, 구현회, 한빛아카데미, 2016

4 메모리에 프로그램의 일부만 적재 수행 시 장점

- 중첩을 고려하여 프로그래밍할 필요가 없으므로 프로그래밍 용이
- 프로세서의 이용률과 처리율 향상 (응답시간이나 반환시간은 향상되지 않음)

5 메모리에 프로그램의 일부만 적재 수행 시 문제점

- 메모리와 디스크 사이에 이동량 증가, 스와핑 공간 필요, 페이지 적재와 복귀 할 페이징 알고리즘 결정해야 함
- 요구된 프로세스의 페이지가 없을 때 처리하는 방안 등

6 문제점 해결 방법

- 실행 중인 프로세스가 참조하는 주소와 메인 메모리에서 사용하는 물리적 주소 분리
  - 실행 중인 프로세스가 참조하는 주소
    : 가상 주소(논리적 주소, 프로그램 주소)
  - 가상 주소를 물리 적 주소로 변환하는 과정 : 매핑
  - 매핑은 가능한 빨리 수행하며 그렇지 않으면 시스템 성능 떨어지고 가상 메모리 사용 효과 낮아짐
  - 매핑은 변환 함수로 표현

1 가상기억장치 관리 기법

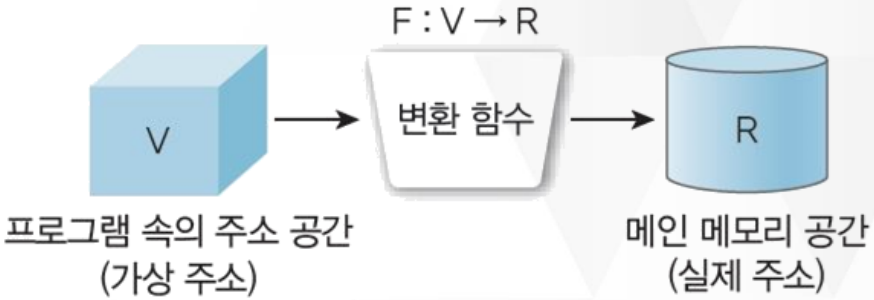
6 문제점 해결 방법

[가상 주소와 물리적 주소의 매핑]

❶ F : V → R || 가상 주소(V)와 실제 주소(R)의 분리

❷ R = F(V)

(a) 정의식

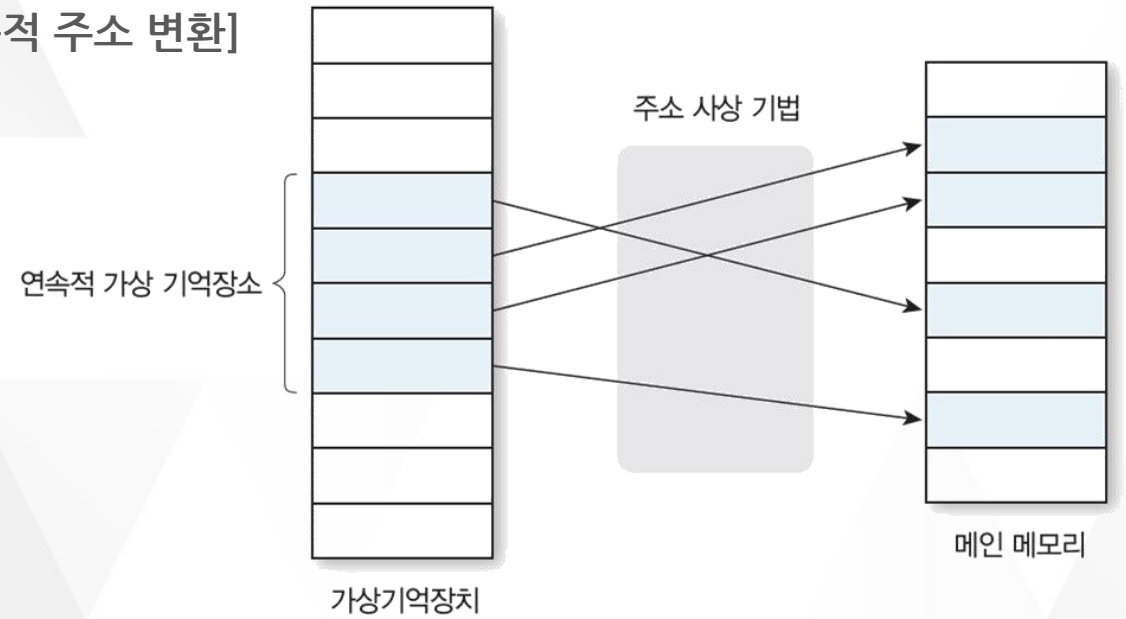


(b) 관계도

※출처: 그림으로 배우는 구조와 원리 운영체제, 구현회, 한빛아카데미, 2016

7 가상 주소와 물리적 주소를 매핑하는 방법

[동적 주소 변환]



※출처: 그림으로 배우는 구조와 원리 운영체제, 구현회, 한빛아카데미, 2016

2 요구 페이징 기법

2 요구 페이징 기법

1 요구 페이징 개념

- 가상 메모리에서 많이 사용하는 메모리 관리 방법
- 스와핑을 사용하는 페이징 시스템과 비슷
- 프로그램을 실행하려고 프로그램의 일부만 메인 메모리에 적재하되, 순차적으로 작성되어 있는 프로그램의 모듈을 처리할 때 다른 부분은 실행하지 않음을 이용

2 요구 페이징 기법

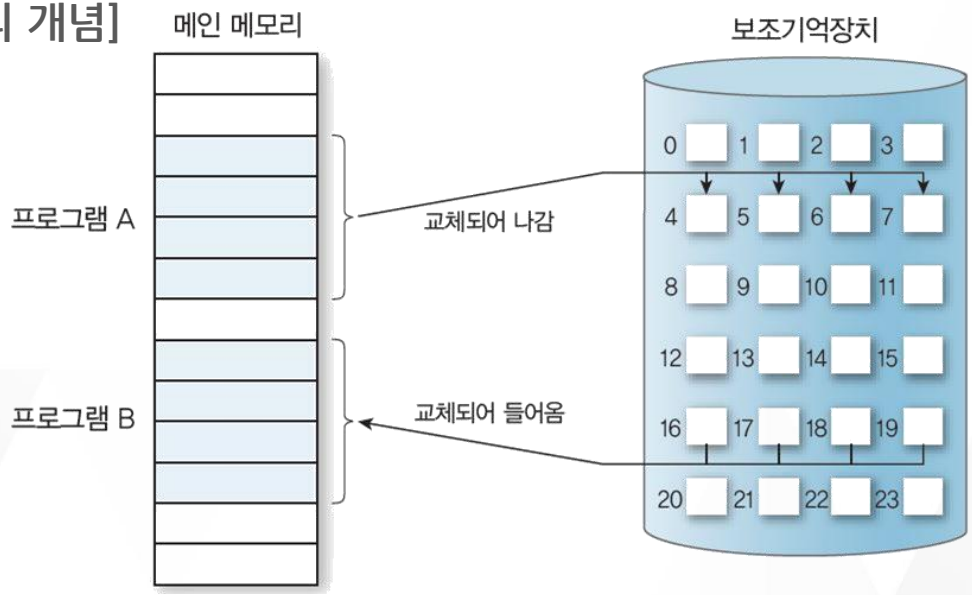
1 요구 페이징 개념

- 프로세스 시작할 때 디스크에서 메인 메모리로 스와핑하는 순수 스와핑과 다르게, 실행 중인 프로세스들의 요구 페이지만 메모리에 반입하여 프로세스의 모든 페이지를 메모리에 동시에 적재하지 않으며 이 과정을 지연 기술 또는 지연 스와퍼라 하고, 이를 위해 페이지 테이블 유지

- 동적 주소 변환에서는 가상 메모리와 메인 메모리의 매핑을 매핑 테이블로 표시하고 페이지 방법에서는 페이지 테이블로 표시함

2 요구 페이징의 개념도

[요구 페이징의 개념]



※출처: 그림으로 배우는 구조와 원리 운영체제, 구현회, 한빛아카데미, 2016

2 요구 페이징 기법

3 페이지 부재 개념

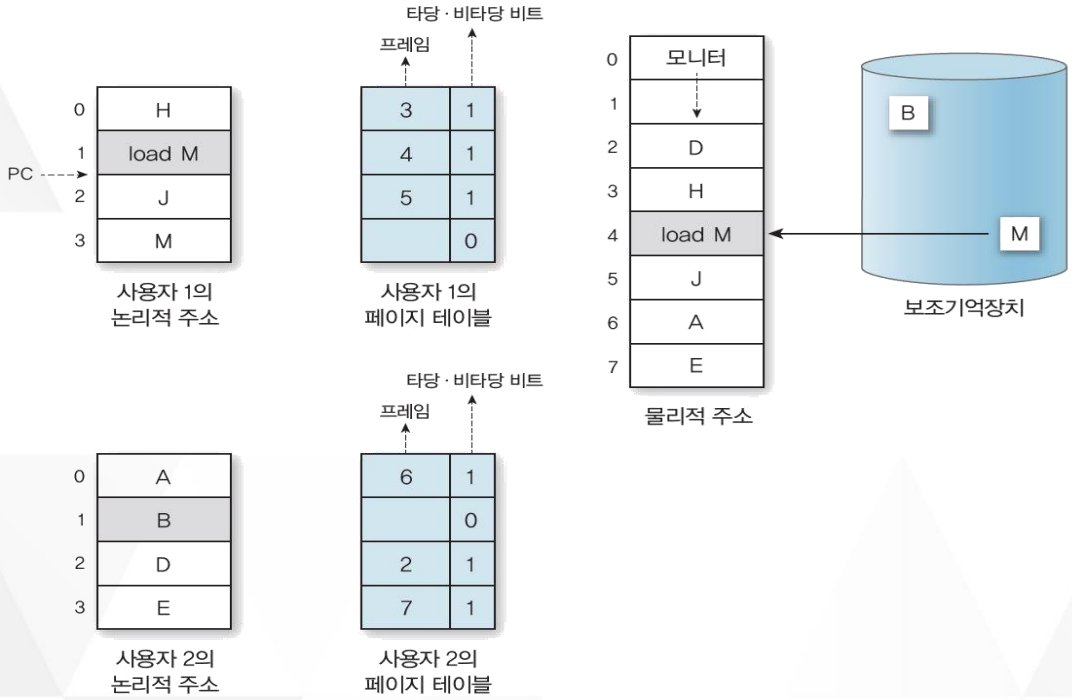
- 프로세스가 비타당 비트로 표시된 페이지에 액세스하지 않는다면 비타당 비트 여부가 영향을 주지 않으나 메모리에 적재되지 않은 페이지에 액세스하려고 한다면 페이지 부재 발생

2 요구 페이징 기법

4 페이지 대치의 개념

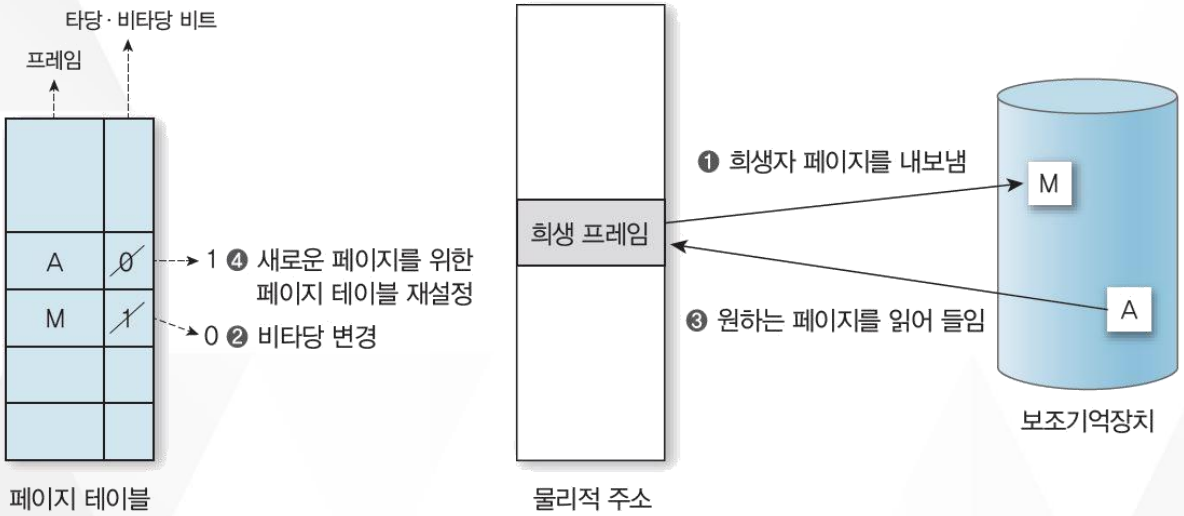
- 페이지 부재 발생 시 메인 메모리에 있으면서
  사용하지 않는 페이지 없애 새로운 페이지로 바꾸는 것

5 페이지 대치의 필요성



※출처: 그림으로 배우는 구조와 원리 운영체제, 구현회, 한빛아카데미, 2016

6 페이지 대치 과정



※출처: 그림으로 배우는 구조와 원리 운영체제, 구현회, 한빛아카데미, 2016

6 페이지 대치 과정

① 프레임 리스트에서 빈 프레임을 찾는데,
빈 프레임이 없으면 현재 사용하지 않는 희생자
프레임을 선정하려고 페이지 대치 알고리즘을 사용,
프레임(M)을 선정하면 프레임의 내용을 디스크에
저장하여(스왑 아웃) 프레임을 비움

② 페이지 테이블의 비타당(0) 비트로 변경하여
페이지가 메모리에 더는 존재하지 않음을 알려 주며
프페임을 비움

③ 원하는 페이지(A)를 디스크에서 읽어 프레임에 저장

④ 새로운 페이지 (A)를 위해 페이지 테이블을 수정

3 페이지 대치 알고리즘

3 페이지 대치 알고리즘

1 선입 선출 대치 알고리즘

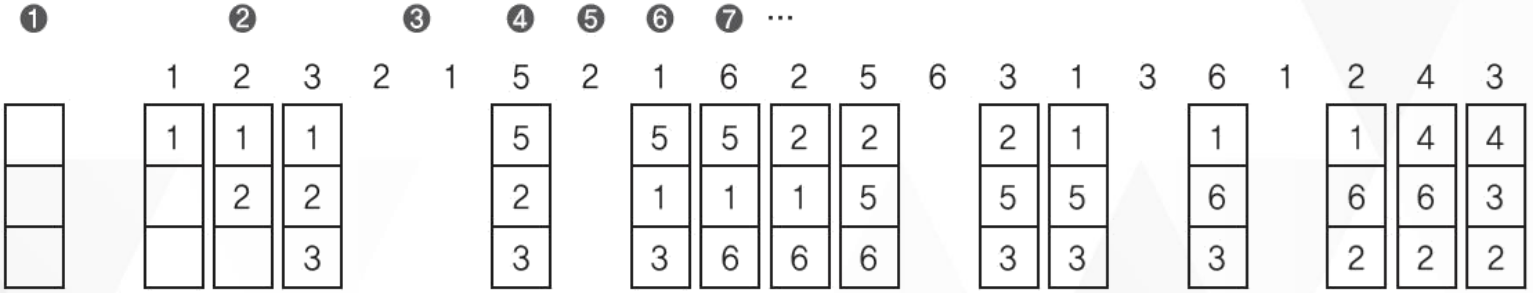
- 가장 간단한 알고리즘
- 메모리에 있는 페이지는 모두
  선입선출 큐 FIFO queue 가 관리
- 큐의 헤드 부분에 있는 페이지를 먼저 대치
- 큐에 있는 페이지가 메모리로 들어갈 때
  큐의 끝에 페이지 삽입
- 큐의 크기는 사용 가능한 메모리 프레임의 수

3 페이지 대치 알고리즘

1 선입 선출 대치 알고리즘

1, 2, 3, 2, 1, 5, 2, 1, 6, 2, 5, 6, 3, 1, 3, 6, 1, 2, 4, 3

(a) 참조 문자열 예



(b) 선입선출 대치 알고리즘 실행 과정

※출처: 그림으로 배우는 구조와 원리 운영체제, 구현회, 한빛아카데미, 2016

3 페이지 대치 알고리즘

2 최적 페이지 대치 알고리즘

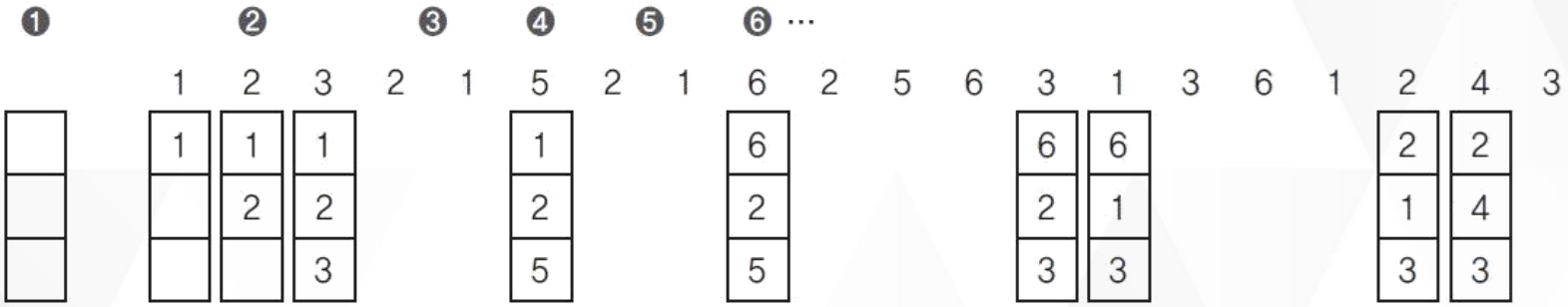
- 벨래디의 알고리즘
- 모든 알고리즘 중 페이지 부재 비율이 가장 낮음
- 기본 아이디어
  - 앞으로 가장 오랫동안 사용하지 않을 페이지를 대치

3 페이지 대치 알고리즘

2 최적 페이지 대치 알고리즘

1, 2, 3, 2, 1, 5, 2, 1, 6, 2, 5, 6, 3, 1, 3, 6, 1, 2, 4, 3

(a) 참조 문자열 예



(b) 최적의 페이지 알고리즘 실행 과정

※출처: 그림으로 배우는 구조와 원리 운영체제, 구현회, 한빛아카데미, 2016

3 페이지 대치 알고리즘

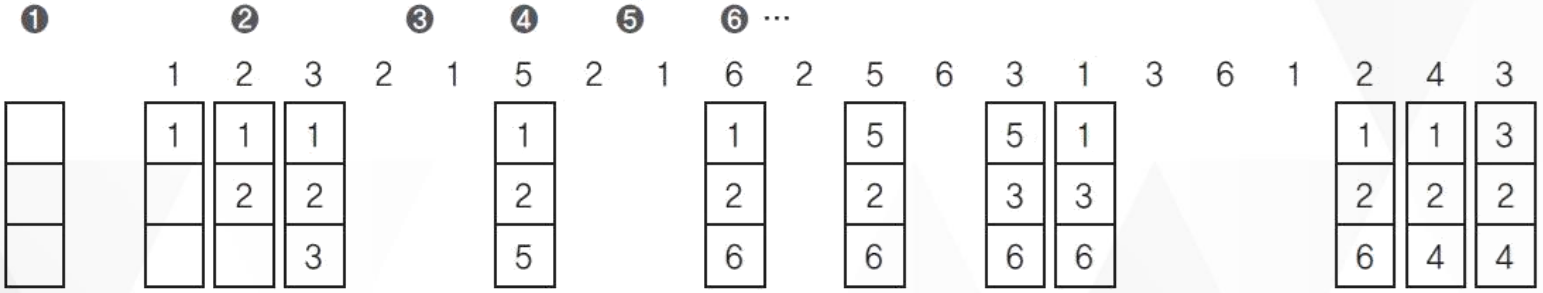
3 최근 최소 사용 대치 알고리즘

- 프로세스가 가장 최근 페이지에 액세스했다는 것은 멀지 않아 다시 액세스할 가능성이 있다는 의미하며 과거 오랫동안 사용하지 않은 페이지로 대치하는 효과로 생각
- 과거의 데이터를 이용하여 미래를 예측하려는 통계적 개념
- 메모리의 지역성을 이용한 알고리즘으로 각 페이지에 마지막으로 사용한 시간을 연관
- 페이지를 대치할 때 오랫동안 사용하지 않은 페이지를 선택하므로 시간적으로 거꾸로 찾는 최적 페이지 대치 알고리즘이라고 할 수 있음

3 최근 최소 사용 대치 알고리즘

1, 2, 3, 2, 1, 5, 2, 1, 6, 2, 5, 6, 3, 1, 3, 6, 1, 2, 4, 3

(a) 참조 문자열 예



(b) 최근 최소 사용 알고리즘 실행 과정

※출처: 그림으로 배우는 구조와 원리 운영체제, 구현회, 한빛아카데미, 2016

3 페이지 대치 알고리즘

4 계수기를 이용한 순서 결정 방법

- 각 페이지 테이블 항목에 사용 시간 레지스터를 연관시키고 프로세서에 논리 클록을 추가한 후 카운터 필드를 덧붙여 프레임 순서 결정
- 메모리관리장치MMU 는 페이지 참조 가 있을 때마다 모든 참조의 프로세서 클록을 각 페이지 테이블 항목에 업데이트
- 페이지를 참조할 때마다 클록은 증가하고, 클록 레지스터의 내용은 페이지의 해당 페이지 테이블에 있는 사용 시간 레지스터에 복사하여 각 페이지의 최후 참조 시간 갖음