제9장
가상기억장치 관리

컴퓨터 운영체제 OS Operating Systems

Network Security Technology Lab.

가상기억장치 관리 개념

개념

- 가상기억장치(virtual storage)
  - 사용자 프로그램을 분할하여 주기억장치에 적재/실행
  - 비연속 적재
  - 페이징/세그먼테이션
- 가상기억장치 관리 목적
  - 가상기억장치 시스템의 성능 최적화
  - 비용 모델(cost model) 설정 및 최적 기법 연구

가상기억장치 관리 개념

- 가상기억장치 관리 기법
  - 할당 기법(allocation strategies)
  - 호출 기법(fetch strategies)
  - 배치 기법(placement strategies)
  - 교체 기법(replacement strategies)
  - 소거 기법(cleaning strategies)
  - 부하 조정 기법(load control strategies)

비용 모델

- 가상기억장치 시스템의 비용 모델
  - 페이지 부재(page fault) 발생 빈도
  - 페이지 부재 발생율(page fault rate)

  - 운영비용, 즉, 각 프로세스들이 실행 중에 발생시키는 페이지 부재율을 최소한으로 줄일 수 있도록 설계되어야 함
  - Context switching이나 커널의 개입을 최소화
  - 시스템 성능의 향상

비용 모델

- 페이지 참조 스트링
  - 프로세스가 실행되는 동안 접근한 페이지들에 대해 이의 페이지 번호들을 접근 순서대로 나열
  - 페이지 참조 스트링(page reference string) : ω
    - $\omega = r_1 r_2 \cdots r_k \cdots r_T$
    - r_i = 페이지 번호, $r_i \in \{0, 1, 2, \cdots, N-1\}$
    - N : 프로세스의 페이지 수 (0 ~ N-1)
    - T : 프로세스의 주기억장치 접근 횟수
  - 페이지 부재율 : $F(\omega)$
    - 프로세스의 실행 중 페이지 부재의 발생 비율
    - $$F(\omega) = \frac{\text{(number of page faults during } \omega)}{|\ \omega\ |}$$

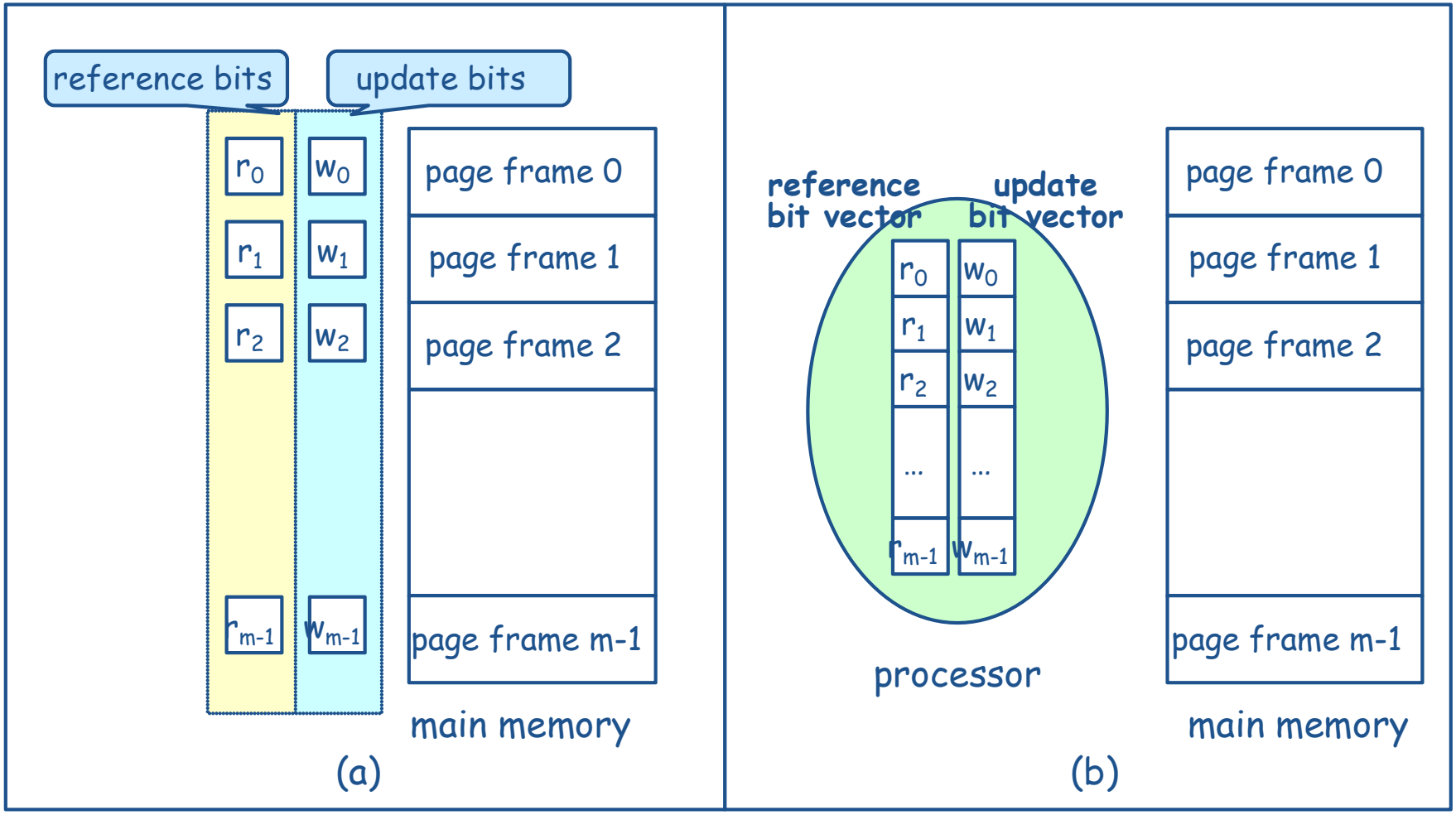
하드웨어 및 소프트웨어 요소

- 하드웨어 요소
  - 주소 사상 장치(address translation device)
    - 주소 사상 과정을 보다 효율적으로 수행하기 위한 하드웨어 추가
    - 주소 사상 장치의 예
      - 주소 사상 테이블 저장을 위한 캐쉬 기억장치
      - 연관 기억장치
      - 주소 사상 과정을 진행하기 위한 하드웨어 모듈 등

하드웨어 및 소프트웨어 요소

- 하드웨어 요소
  - 비트 벡터(bit vectors)
    - 주기억장치의 페이지 프레임에 적재되어 있는 각종 페이지들에 대한 최근 참조 여부 및 내용 갱신 여부 등을 기록
    - 종류
      - 참조 비트(reference bits, use bits)
      - 갱신 비트(update bits, modify bits, write bits, dirty bits)

가상 기억장치 시스템의 비트 벡터



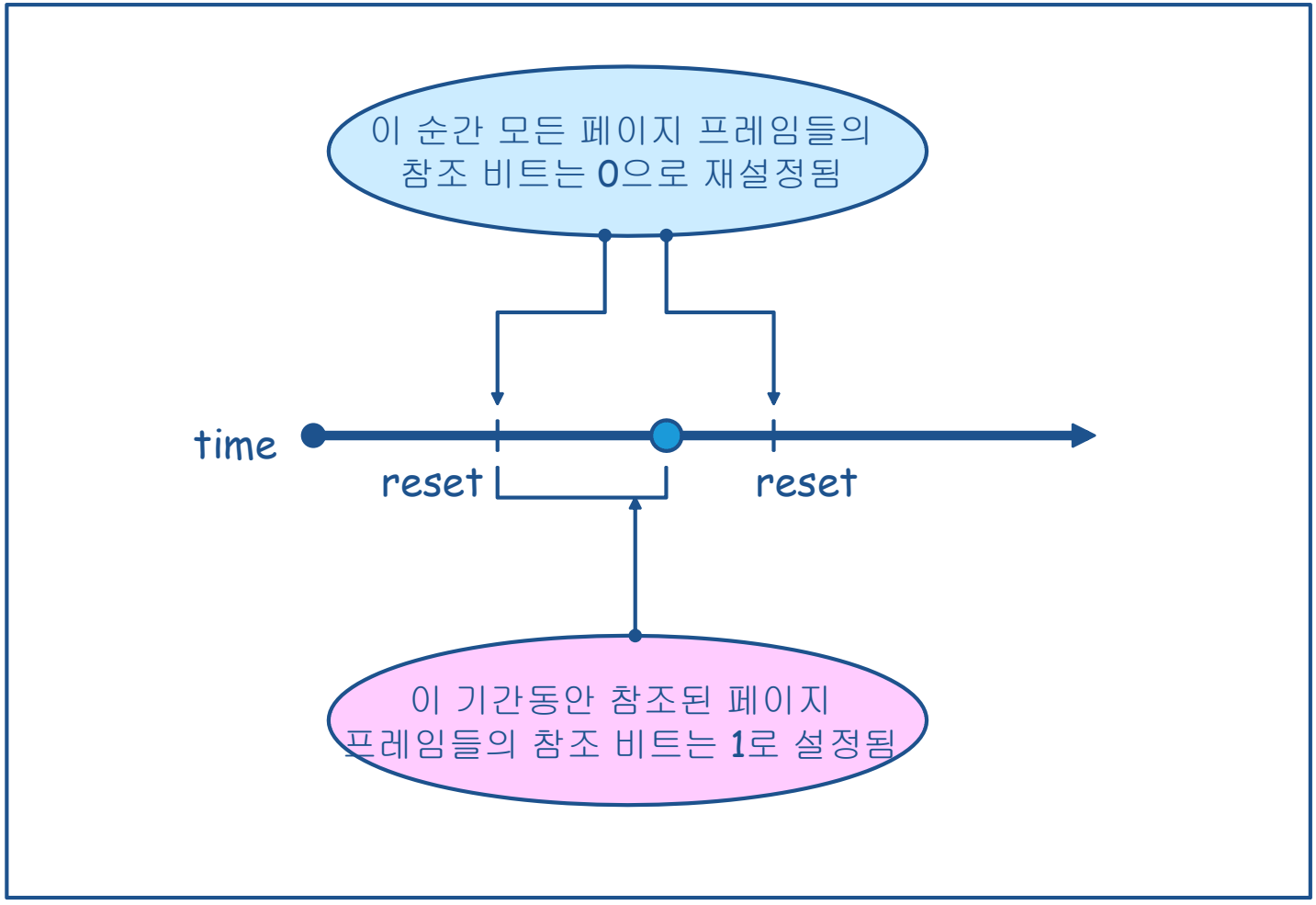
하드웨어 및 소프트웨어 요소

- 참조 비트 벡터
  - 각 페이지 프레임의 내용이 최근 참조되었는지에 대한 정보 기록
  - 과정
    ① 실행중이 프로세스에 의해 참조되면 참조 비트 1로 설정
    ② 정해진 주기가 되면 모든 참조 비트들이 0으로 재설정
    ③ 다시 참조된 페이지 프레임들의 참조 비트는 1로 설정
    ④ 이 과정을 반복
  - 임의의 순간에 참조 비트 = 1
    - 해당 페이지 프레임이 최근의 재설정 주기 후에 적어도 한 번 이상 참조 되었음

하드웨어 및 소프트웨어 요소

- 갱신 비트 벡터
  - 해당 페이지 프레임의 내용이 갱신되었는지에 대한 정보 기록
  - 주기적으로 0으로 재설정되지 않음
  - 갱신 비트가 1인 페이지 프레임의 내용
    - 언젠가 그 내용을 디스크로 다시 전송하여야 함을 의미함
    - write-back

참조 비트의 변화 과정


이 순간 모든 페이지 프레임들의
참조 비트는 0으로 재설정됨
time
reset
reset
이 기간동안 참조된 페이지
프레임들의 참조 비트는 1로 설정됨

하드웨어 및 소프트웨어 요소

- 소프트웨어 요소
  - 가상기억장치 시스템의 효율성 및 성능의 향상을 위해 필요한 가상기억장치 관리 기법들
    - 할당 기법(allocation strategies)
    - 호출 기법(fetch strategies)
    - 배치 기법(placement strategies)
    - 교체 기법(replacement strategies)
    - 소거 기법(cleaning strategies)
    - 부하 조정 기법(load control strategies)

하드웨어 및 소프트웨어 요소

- 할당 기법
  - 각 프로세스들에게 할당해 줄 주기억장치의 양을 결정하는 기법
  - 분류
    - 고정 할당(fixed allocation) 기법
      - » 메모리 할당량을 프로세스의 실행 동안 고정시키는 기법
    - 가변 할당(variable allocation) 기법
      - » 메모리 할당량을 프로세스의 실행 동안 변화시키는 기법
  - 고려사항
    - 해당 컴퓨터 시스템의 능력과 특성 고려해야 함
    - 프로세스들이 요구하는 양을 적절히 파악할 수 있어야 함
    - 너무 많게 할당하는 경우
      - » 주기억장치 공간이 낭비됨
    - 너무 적게 할당하는 경우
      - » 페이지 부재의 발생 빈도 증가, 시스템 성능 저하됨

하드웨어 및 소프트웨어 요소

- 호출 기법
  - 특정 페이지를 언제 주기억장치에 적재할 것인가를 결정하는 기법
  - 분류
    - 요구 호출(demand fetch, demand paging) 기법
      - » 실행중인 프로세스가 실제로 참조한 페이지들만을 적재
      - » 항상 필요한 페이지들만이 주기억장치에 적재됨
      - » 추가 오버헤드 필요하지 않음
      - » 페이지 부재시 오버헤드 발생
    - 예측 호출(anticipatory fetch, prepaging) 기법
      - » 가까운 미래에 참조될 가능성이 높은 페이지를 실제 참조되기 전에 미리 주기억장치에 적재
      - » 페이지 참조에 대한 예측 오버헤드 발생
      - » 예측의 성공률에 민감함

❑ 실제로 대부분의 시스템에서 요구 호출 기법 사용
- 예측 호출 기법 : 예측 오버헤드 필요함, 잘못된 예측으로 인한 시스템 자원 낭비 큼
- 요구 호출 기법 : 실제 시스템에서 크게 성능의 저하를 초래하지 않음

하드웨어 및 소프트웨어 요소

- 배치 기법
  - 주기억장치의 어느 빈 페이지 프레임에 적재할 것인지 결정하는 기법
  - 페이징 시스템에서의 배치 기법은 불필요함
  - 세그먼테이션 시스템에서의 배치기법
    - 최초 적합(first-fit) 전략
    - 최적 적합(best-fit) 전략
    - 최악 적합(worst-fit) 전략
    - 순환 최초 적합(next-fit) 전략

하드웨어 및 소프트웨어 요소

- 교체 기법
  - 새로운 페이지가 적재되어야 하는 상황에서 빈 페이지 프레임이 없을 때 어느 페이지를 교체시킬 것인지 결정
  - 분류
    - 고정 할당(fixed allocation) 기반의 교체 기법
      - » MIN(OPT, B_0) 알고리즘
      - » 무작위(Random) 알고리즘
      - » FIFO(First In First Out) 알고리즘
      - » LRU(Least Recently Used) 알고리즘
      - » LFU(Least Frequently Used) 알고리즘
      - » NUR(Not Used Recently) 알고리즘
      - » 클럭(Clock) 알고리즘
      - » Second chance 알고리즘
    - 가변 할당 (variable allocation) 기반의 교체 기법
      - » VMIN(Variable MIN) 알고리즘
      - » WS(Working Set) 알고리즘
      - » PFF(Page Fault Frequency) 알고리즘

하드웨어 및 소프트웨어 요소

- 소거 기법
  - 변경된 페이지의 내용을 언제 디스크에 저장(write-back)할 것인가를 결정하는 기법
  - 분류
    - 요구 소거(demand cleaning) 기법
      - » 변경된 페이지가 교체 대상이 되었을 때에만 디스크에 기입
      - » 주기억장치에서 변경된 페이지를 한 번만 디스크에 기입
    - 예측 소거(anticipatory cleaning, precleaning) 기법
      - » 더 이상 변경될 가능성이 없다고 판단될 때 미리 디스크에 기입
      - » 교체될 당시에 디스크에 기입함으로 인한 지연 시간을 줄임
      - » 디스크에 기입 후 교체되기 전에 재변경되면 오버헤드 초래

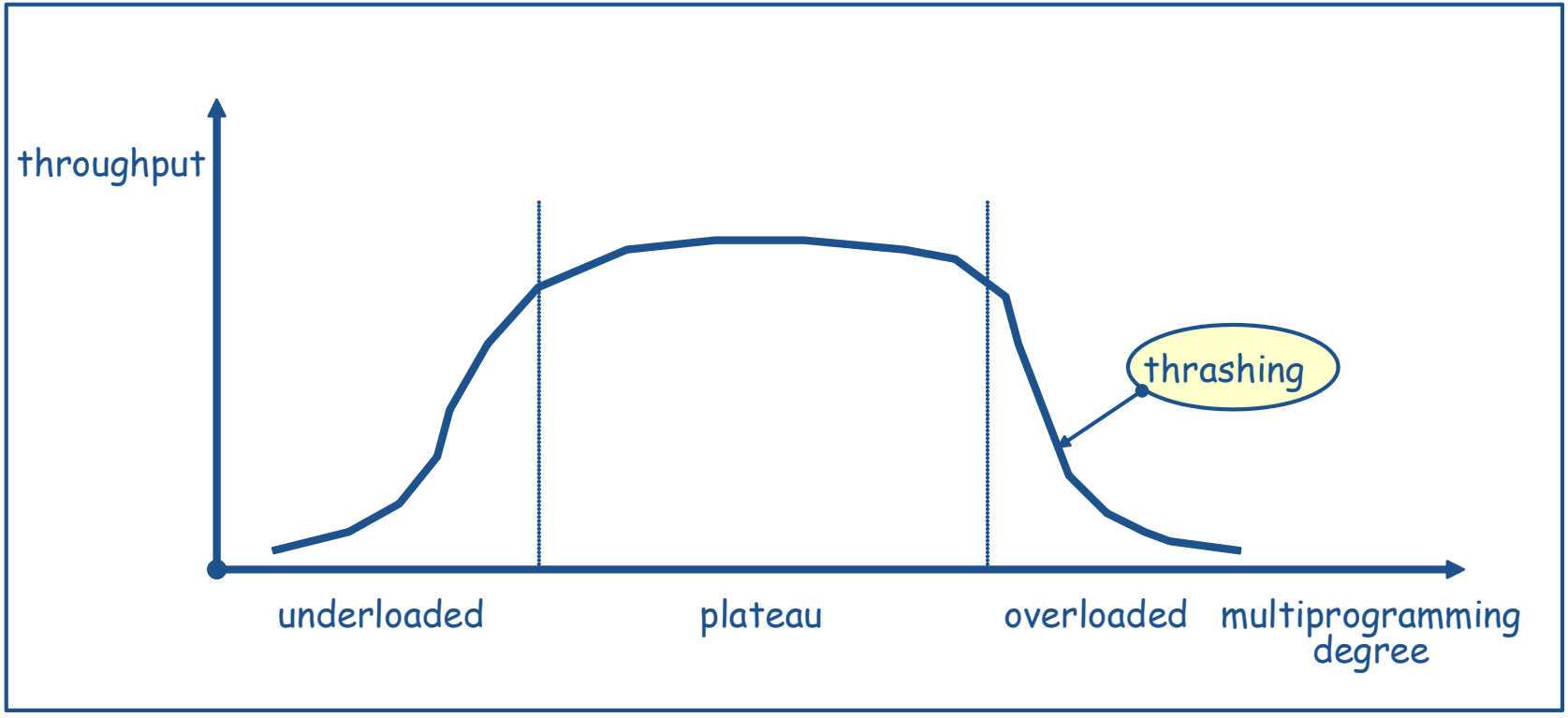
❑ 실제로 대부분의 시스템에서 요구 소거 기법 사용
- 예측 소거 기법 : 예측 오버헤드 필요함, 잘못된 예측으로 인한 시스템 자원 낭비 큼
- 요구 소거 기법 : 실제 시스템에서 크게 성능의 저하를 초래하지 않음

하드웨어 및 소프트웨어 요소

- 부하 조정 기법
  - 시스템의 다중프로그래밍 정도를 조절하는 기법
  - 할당 기법과 연계되어야 함
  - 컴퓨터 시스템은 항상 적절하게 다중프로그래밍 정도를 유지
    - plateau 영역에서 적절히 결정 (다음 슬라이드 참조)

- 저부하 상태 : 시스템 자원의 낭비, 성능 저하
- 고부하 상태 : 시스템 자원에 대한 경쟁, 성능 저하
  - 스래싱(thrashing) 현상 발생
    - 페이지 부재가 과도하게 발생하는 현상
- plateau 영역의 범위에 대한 수치 값이 시스템마다 다를 수 있음을 고려

다중프로그래밍 정도와 단위시간당 처리량과의 관계



- 소프트웨어적 가상기억장치 관리 기법의 목적
  - 페이지 부재의 발생 빈도 축소 등에 의한 시스템 운영 비용 극소화
  - 시스템 성능 극대화
- 시스템 성능에 많은 영향을 미치는 대표적인 기법
  - 할당 기법, 교체 기법

지역성(Locality)

- 정의
  - 프로그램 내의 일부 명령들이 집중적으로 실행되는 현상
  - 지역성의 원인
    - 프로그램의 반복 구조
    - 배열, 구조체 등의 자료 구조

- 지역성의 종류
  - 시간 지역성(temporal locality)
    - 한 번 실행된 명령은 이후 재실행될 가능성 높음
    - 한 번 접근된 데이터 영역은 이후 재접근될 가능성 높음
  - 공간 지역성(spatial locality)
    - 직전에 실행된 명령의 주변 명령들이 실행될 가능성 높음
    - 직전에 접근된 데이터의 부근 데이터들의 접근 가능성 높음

지역성의 예

프로그램 실행 환경

- 가정
  - 페이징 시스템
  - 한 페이지가 1000 워드를 저장할 수 있는 크기를 가짐
  - 모든 기계어 명령들은 하나의 워드에 해당하는 크기를 가짐
  - 주소 기정은 워드 단위로 이루어짐
  - 변수 n = 1000
  - 컴파일된 프로그램이 주기억장치 4000번지, 즉, 4번 페이지 프레임부터 연속적으로 적재됨

프로그램 코드

```
...
for ← 1 to n do
A[i] ← B[i] + C[i];
endfor
...
```

주기억장치에 적재된 실행 코드

주기억장치 주소	실행 코드
4000	(R1) ← ONE
4001	(R2) ← n
4002	compare R1, R2
4003	branch greater 4009
4004	(R3) ← B(R1)
4005	(R3) ← (R3) + C(R1)
4006	A(R1) ← (R3)
4007	(R1) ← (R1) + 1
4008	branch 4002
...	...
6000-6999	storage for array A
7000-7999	storage for array B
8000-8999	storage for array C
9000	storage for ONE
9001	storage for n

```
...
for ← 1 to n do
A[i] ← B[i] + C[i];
endfor
...
```

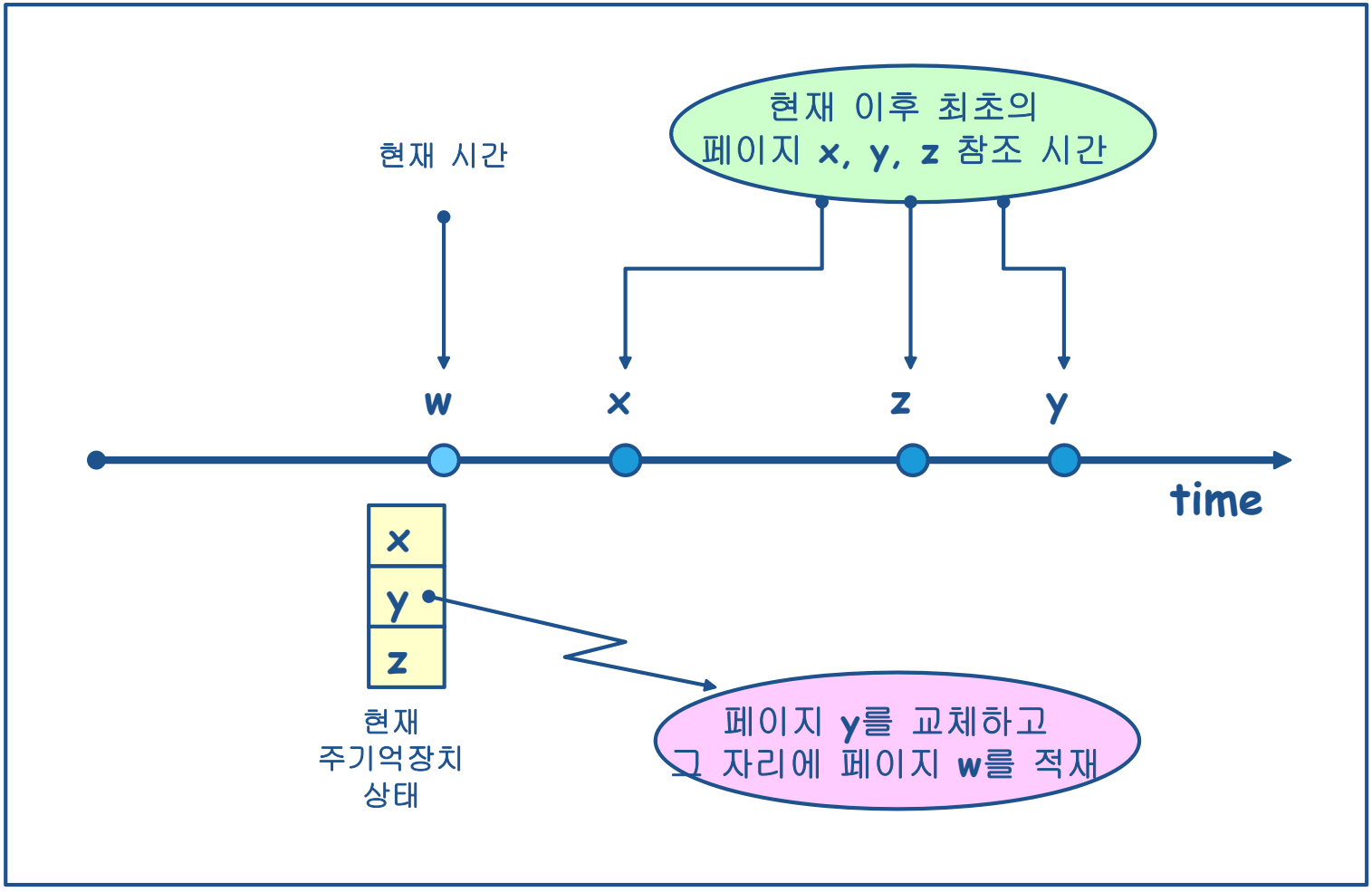
- $\omega = 494944(47484649444)^{1000}$

- 11000번의 주기억장치 참조동안 프로세스는 5개의 페이지만을 집중적으로 접근하게 됨

고정 할당 기반의 교체 기법

- MIN 알고리즘(OPT, B0 알고리즘)
  - 1966년 Belady에 의해 제시된 기법
  - 페이지 부재 발생 횟수를 최소화하는 최적의 기법 (proved)
  - 기법
    - 현재 시점 이후로 가장 오래동안 참조되지 않을 페이지를 교체
    - tie-breaking rule : 가장 작은 번호를 갖는 페이지를 교체
  - 실현 불가능한 기법
    - 프로세스의 참조 스트링이 미리 알려져 있어야 함
  - 의미
    - 각 교체 기법들에 대한 성능 평가 도구로 사용 가능함

MIN 알고리즘의 페이지 교체 예



MIN 알고리즘 사용시 페이지 부재 발생 횟수 분석

- 가정
  - 프로세스에게 4개의 페이지 프레임이 고정 할당됨
  - 초기에 4개의 페이지 프레임들이 모두 비어 있음

ω = 1 2 6 1 4 5 1 2 1 4 5 6 4 5

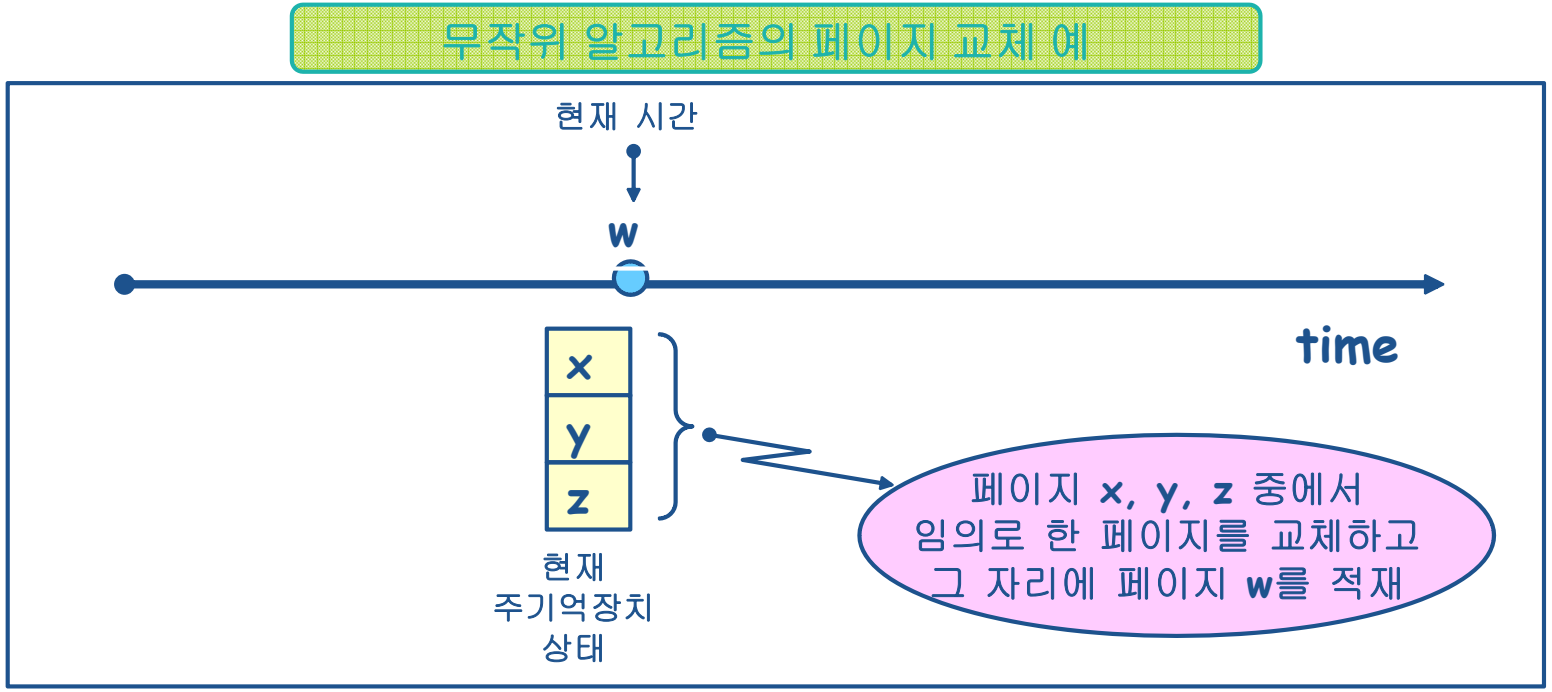
MIN 기법에 의한 주기억장치 상태 변화 과정 예

시간	1	2	3	4	5	6	7	8	9	10	11	12	13	14
참조 스트링	1	2	6	1	4	5	1	2	1	4	5	6	4	5
주기억장치 상태	1	1	1	1	1	1	1	1	1	1	1	6	6	6
		2	2	2	2	2	2	2	2	2	2	2	2	2
			6	6	6	5	5	5	5	5	5	5	5	5
					4	4	4	4	4	4	4	4	4	4
페이지 부재 발생 여부	F	F	F		F	F						F		

❑ 페이지 부재의 발생 횟수 : 총 6회

고정 할당 기반의 교체 기법

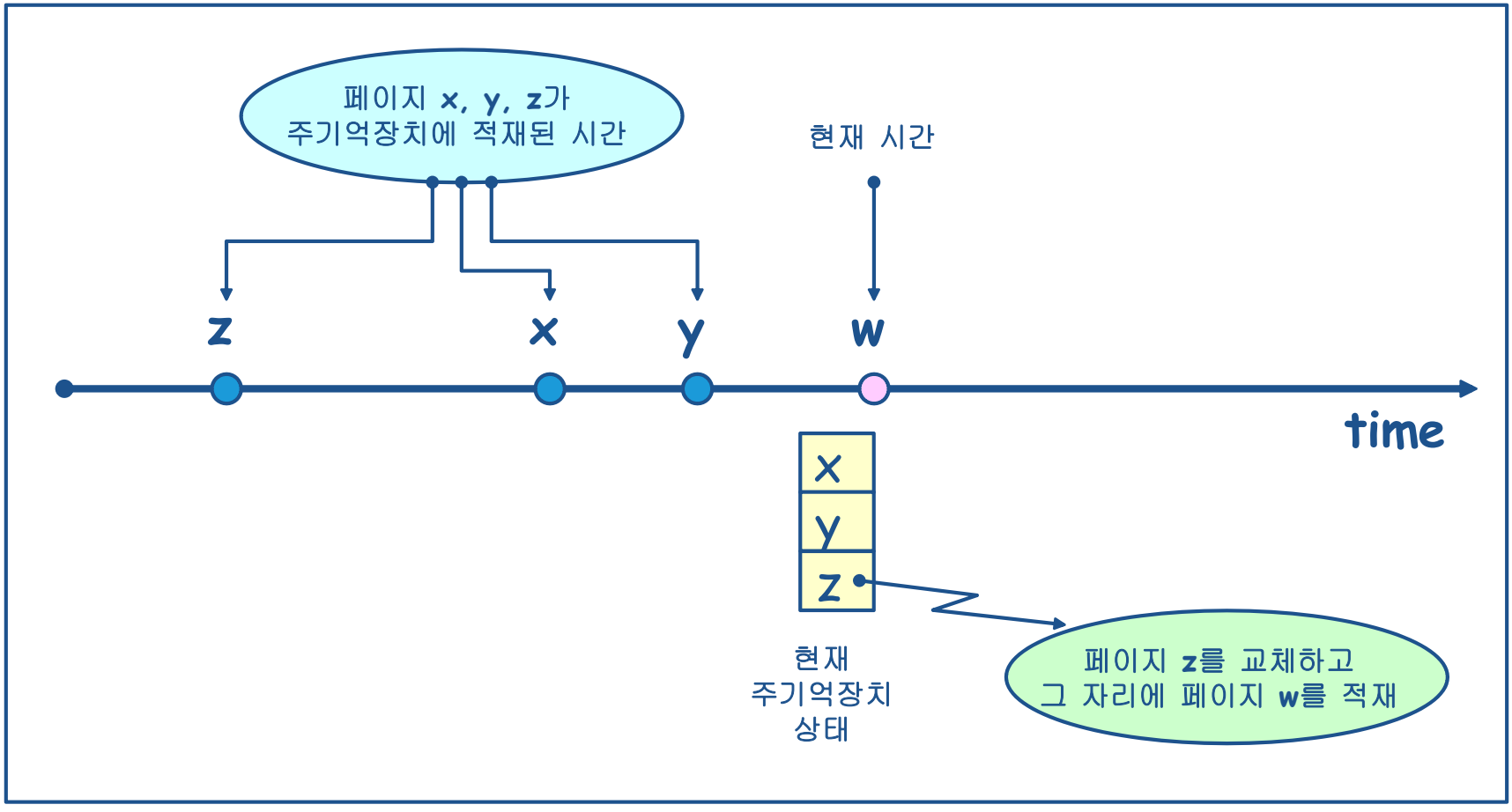
- 무작위(random) 알고리즘
  - 원칙 없이 임의로 교체 대상 페이지 선정하는 기법
  - 교체될 페이지를 선정하는 오버헤드 적음
  - 페이지 부재율을 줄이려는 어떠한 시도도 하지 않는 기법임



고정 할당 기반의 교체 기법

- FIFO 알고리즘
  - 페이지가 주기억장치에 적재된 시간을 기준으로 교체될 페이지를 선정하는 기법
  - 기법
    - 주기억장치에 가장 먼저 적재된 페이지를 교체함
  - 조건
    - 페이지들의 주기억장치 적재 시간을 기록해 두어야 함
  - 특성
    - 사용 빈도가 높은 프로그램의 페이지들이 교체될 가능성 높음
  - FIFO의 변칙(FIFO anomaly, Belady's anomaly)
    - FIFO 알고리즘을 사용할 경우 주기억장치 할당량을 늘려 주었는데도 페이지 부재의 발생 횟수가 증가하는 경우 있음

FIFO 알고리즘의 페이지 교체 예



FIFO 알고리즘 사용시 페이지 부재 발생 횟수 분석

- 가정
  - 프로세스에게 4개의 페이지 프레임이 고정 할당됨
  - 초기에 4개의 페이지 프레임들이 모두 비어 있음

ω = 1 2 6 1 4 5 1 2 1 4 5 6 4 5

FIFO 기법에 의한 주기억장치 상태 변화 과정 예

시간	1	2	3	4	5	6	7	8	9	10	11	12	13	14
참조 스트링	1	2	6	1	4	5	1	2	1	4	5	6	4	5
주기억장치 상태	1	1	1	1	1	5	5	5	5	5	5	5	4	4
		2	2	2	2	2	1	1	1	1	1	1	1	5
			6	6	6	6	6	2	2	2	2	2	2	2
					4	4	4	4	4	4	4	6	6	6
페이지 부재 발생 여부	F	F	F		F	F	F	F				F	F	F

⇨ 페이지 부재의 횟수 : 총 10번 발생

FIFO Anomaly의 예

- 가정
  - 프로세스의 프로그램이 5개의 페이지로 구성되어 있음

ω = 1 2 3 4 1 2 5 1 2 3 4 5

페이지 프레임을 3개 할당했을 경우

시간	1	2	3	4	5	6	7	8	9	10	11	12
참조 스트링	1	2	3	4	1	2	5	1	2	3	4	5
주기억장치 상태	1	1	1	4	4	4	5	5	5	5	5	5
		2	2	2	1	1	1	1	1	3	3	3
			3	3	3	2	2	2	2	2	4	4
페이지 부재 발생 여부	F	F	F	F	F	F	F			F	F	

페이지 프레임을 4개 할당했을 경우

시간	1	2	3	4	5	6	7	8	9	10	11	12
참조 스트링	1	2	3	4	1	2	5	1	2	3	4	5
주기억장치 상태	1	1	1	1	1	1	5	5	5	5	4	4
		2	2	2	2	2	2	1	1	1	1	5
			3	3	3	3	3	3	2	2	2	2
				4	4	4	4	4	4	3	3	3
페이지 부재 발생 여부	F	F	F	F			F	F	F	F	F	F

- 페이지 부재의 발생 횟수 : 총 9회
- 페이지 부재의 발생 횟수 : 총 10회

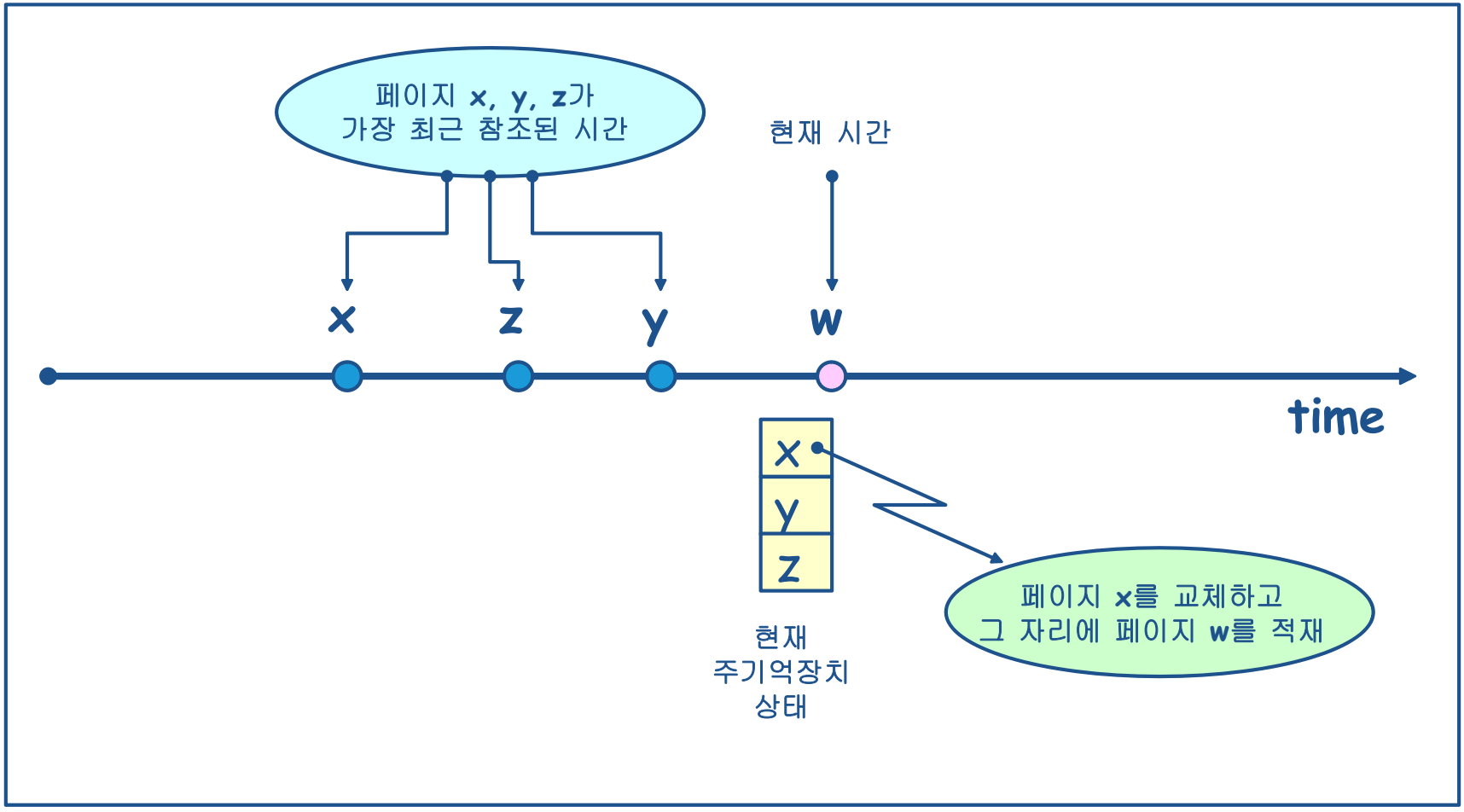
고정 할당 기반의 교체 기법

- LRU(Least Recently Used) 알고리즘
  - 참조된 시간을 기준으로 교체될 페이지를 선정하는 기법
  - 기법
    - 최근 가장 오래동안 참조되지 않은 페이지를 교체함
  - 조건
    - 적재되어 있는 페이지의 참조 시마다 참조 시간을 기록해야 함
  - 특성
    - 프로그램의 지역성(locality)에 기반을 둠
    - 최적의 기법인 MIN 알고리즘의 성능에 근사하게 제시된 기법
  - 실제로 가장 많이 사용되고 있는 기법

고정 할당 기반의 교체 기법

- LRU(Least Recently Used) 알고리즘
  - 단점
    - 매 번 참조 시간 기록해야 하는 오버헤드가 매우 큼
    - 대형 반복 구조를 실행하는데 그 보다 적은 크기의 기억장치 공간이 할당된 경우 페이지 부재의 발생 횟수가 급격히 증가함
      - (예, 프로그램상의 반복 구조 크기 : 4 페이지
        이를 실행하는 프로세스에게 할당된 페이지 프레임 : 3개)

LRU 알고리즘의 페이지 교체 예



LRU 알고리즘 사용시 페이지 부재 발생 횟수 분석

◆ 가정
- 프로세스에게 4개의 페이지 프레임이 고정 할당됨
- 초기에 4개의 페이지 프레임들이 모두 비어 있음

ω = 1 2 6 1 4 5 1 2 1 4 5 6 4 5

LRU 기법에 의한 주기억장치 상태 변화 과정 예

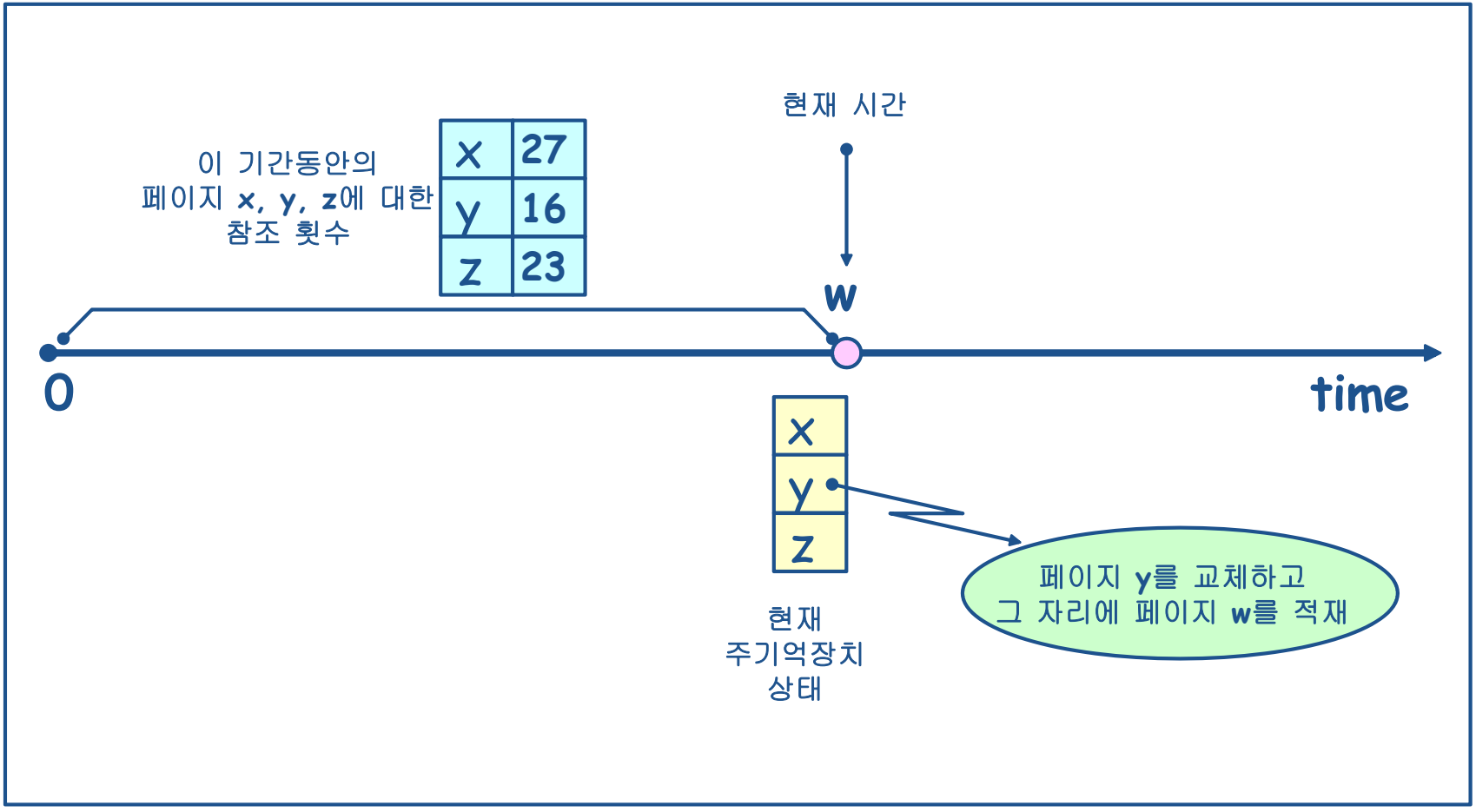
시간	1	2	3	4	5	6	7	8	9	10	11	12	13	14
참조 스트링	1	2	6	1	4	5	1	2	1	4	5	6	4	5
주기억장치 상태	1	1	1	1	1	1	1	1	1	1	1	1	1	1
		2	2	2	2		5	5	5	5	5	5	5	5
			6	6	6	6	6	2	2	2	2	6	6	6
					4	4	4	4	4	4	4	4	4	4
페이지 부재 발생 여부	F	F	F		F	F		F				F		

❑ 페이지 부재의 발생 횟수 : 총 7회

고정 할당 기반의 교체 기법

- LFU(Least Frequently Used) 알고리즘
  - 참조된 횟수를 기준으로 교체될 페이지를 선정하는 기법
  - 기법
    - 가장 참조 횟수가 적은 페이지를 교체함
    - tie-breaking rule : LRU 기법에 따름
  - 조건
    - 페이지들이 참조될 때마다 참조 횟수를 누적시켜야 함
  - 특징
    - LRU 기법의 오버헤드를 줄이면서 프로그램의 지역성 이용
    - 단점
      - 최근에 적재된 페이지가 이후 참조될 가능성이 많음에도 불구하고 교체될 가능성 있음
      - 각 페이지의 참조 시마다 해당 페이지의 참조 횟수를 누적시켜야 하는 오버헤드 발생

LFU 알고리즘의 페이지 교체 예



LFU 알고리즘 사용시 페이지 부재 발생 횟수 분석

- 가정
  - 프로세스에게 4개의 페이지 프레임이 고정 할당됨
  - 초기에 4개의 페이지 프레임들이 모두 비어 있음

ω = 1 2 6 1 4 5 1 2 1 4 5 6 4 5

LFU 기법에 의한 주기억장치 상태 변화 과정 예

시간	1	2	3	4	5	6	7	8	9	10	11	12	13	14
참조 스트링	1	2	6	1	4	5	1	2	1	4	5	6	4	5
주기억장치 상태	1	1	1	1	1	1	1	1	1	1	1	1	1	1
		2	2	2	2	5	5	5	5	5	5	5	5	5
			6	6	6	6	6	2	2	2	2	6	6	6
					4	4	4	4	4	4	4	4	4	4
페이지 부재 발생 여부	F	F	F		F	F		F				F		

❑ 페이지 부재의 횟수 : 총 7번 발생

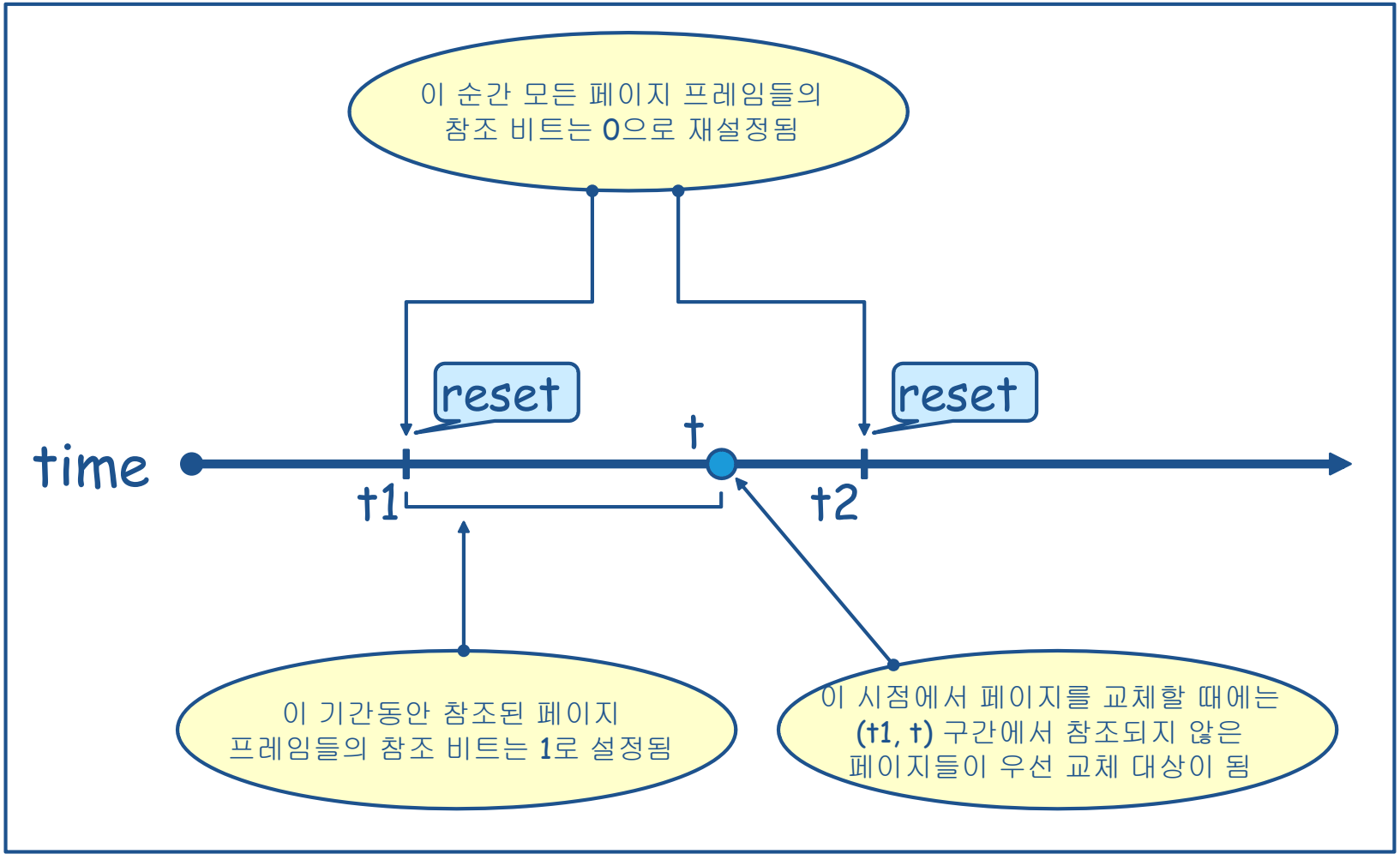
고정 할당 기반의 교체 기법

- NUR(Not Used Recently) 알고리즘
  - 적은 오버헤드로 LRU 기법의 성능을 내기 위한 기법
  - 비트 벡터 이용
    - 참조 비트 벡터
      - 일정 주기마다 0으로 재설정 되며
        이후 해당 페이지가 참조되면 1로 설정됨
    - 갱신 비트 벡터
      - 내용이 갱신되었을 경우 1로 설정되어
        해당 페이지의 내용을 디스크에 기입하여야 함을 표시함

고정 할당 기반의 교체 기법

- NUR(Not Used Recently) 알고리즘
  - 기법
    - 참조 비트와 갱신 비트를 검사하여 페이지를 교체함
    - 교체 순서(참조비트 : r, 갱신 비트 m)
      ① (r, m) = (0, 0)인 페이지가 있으면 이를 교체함
      ② (r, m) = (0, 1)인 페이지가 있으면 이를 교체함
      ③ (r, m) = (1, 0)인 페이지가 있으면 이를 교체함
      ④ (r, m) = (1, 1)인 페이지를 교체함

NUR 알고리즘의 페이지 교체 예



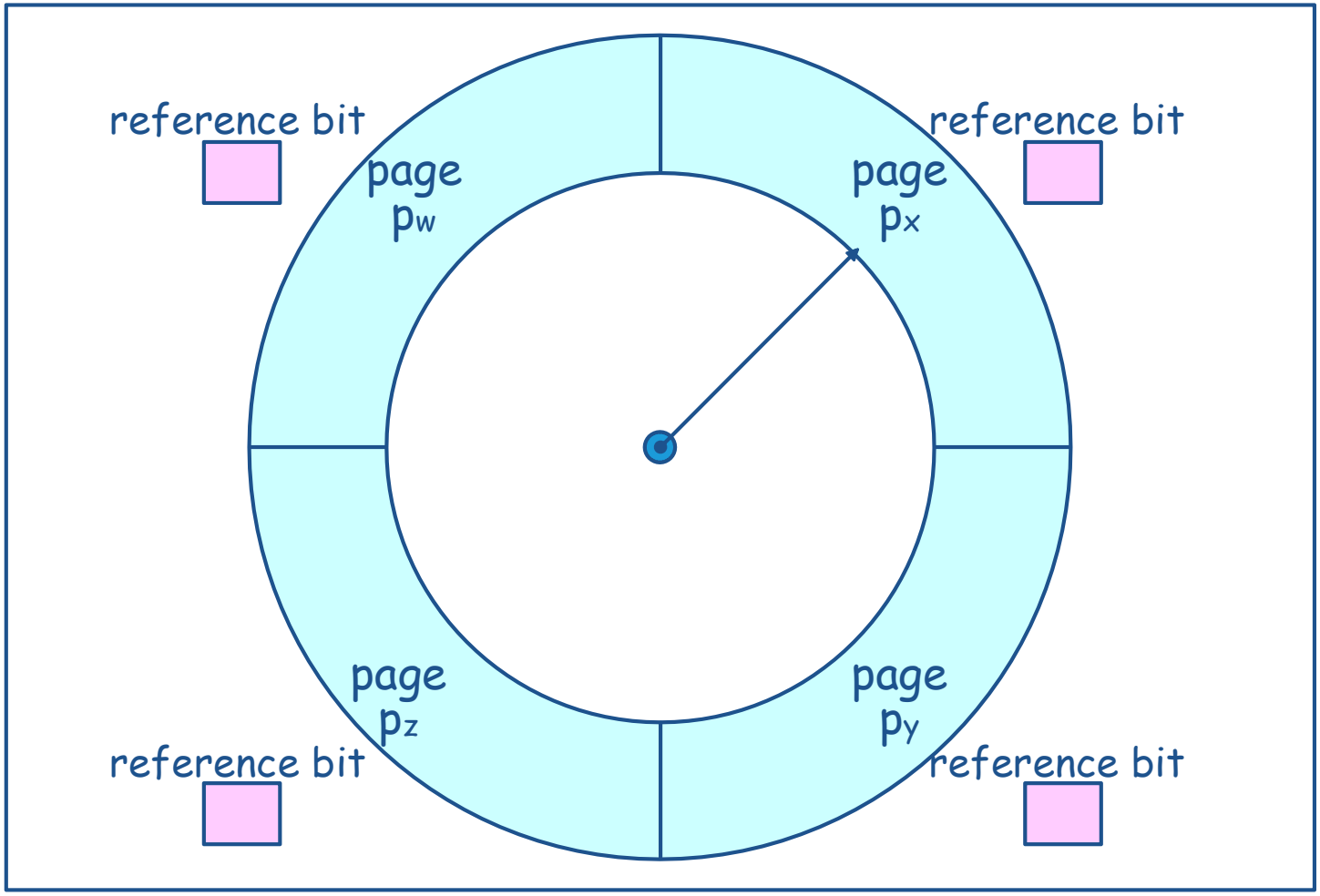
고정 할당 기반의 교체 기법

- 클럭(clock) 알고리즘
  - IBM VM/370 운영체제에서 사용된 교체 기법
  - 참조 비트를 사용함
    - 참조 비트를 주기적으로 0으로 재설정하지 않음
  - 주기억장치에 적재되어 있는 페이지들을 환형 리스트로 보고

    이 페이지들을 따라 움직이는 포인터를 사용하여

    교체될 페이지를 선정하는 기법

클럭 알고리즘의 개념


reference bit
page
p_w
reference bit
page
p_x
page
p_z
reference bit
page
p_y
reference bit

고정 할당 기반의 교체 기법

- 기법
  - 포인터를 시계 방향으로 이동시키면서 교체될 페이지 선정
  - 교체 단계
    ① 현재 포인터가 가리키고 있는 페이지의 참조 비트 검사
    ② 그 값이 0이면 교체 대상으로 선정하고 포인터를 시계 방향으로 한 단계 진행시킨 후 선정 과정을 종료
    ③ 그 값이 1이면 참조 비트를 0으로 재설정하고 포인터를 시계 방향으로 한 단계 진행시킨 후 ① 부터 반복

- 특징
  - 먼저 적재된 페이지가 교체의 대상이 될 가능성 높음
    – FIFO 알고리즘과 유사
  - 참조 비트를 검사하여 그 값이 0인 페이지가 교체 대상이 됨
    – LRU 알고리즘(또는 NUR 알고리즘)과 유사

클럭 알고리즘 사용시 페이지 부재 발생 횟수 분석

- 가정
  - 프로세스에게 4개의 페이지 프레임이 고정 할당됨
  - 초기에 4개의 페이지 프레임에 페이지 a, b, c, d가 적재되어 있음
  - 이 4개의 페이지 프레임의 참조 비트가 모두 1로 설정되어 있음

ω = c a d b e b a b c d

클럭 기법에 의한 주기억장치 상태 변화 과정 예

시간		0	1	2	3	4	5	6	7	8	9	10
참조 스트링			c	a	d	b	e	b	a	b	c	d
주기억 장치 상태	frame 0		→a/1	→a/1	→a/1	→a/1	e/1	e/1	e/1	e/1	→e/1	d/1
	frame 1		b/1	b/1	b/1	b/1	→b/0	→b/1	b/0	b/1	b/1	→b/0
	frame 2		c/1	c/1	c/1	c/1	c/0	c/0	a/1	a/1	a/1	a/0
	frame 3		d/1	d/1	d/1	d/1	d/0	d/0	→ d/0	→ d/0	c/1	c/0
페이지 부재 발생 여부							F		F		F	F
Pclock							e		a		c	d
Qclock							a		c		d	e

- Pclock : 해당 시점에 주기억장치에 적재되는 페이지
- Qclock : 해당 시점에 주기억장치로부터 교체되는 페이지

고정 할당 기반의 교체 기법

- Second chance 알고리즘
  - 클럭 알고리즘과 유사
  - 클럭 알고리즘과 차이점
    - 참조 비트뿐만 아니라 갱신 비트도 함께 고려함

고정 할당 기반의 교체 기법

❖ Second chance 알고리즘

- 기법
  - 포인터를 시계 방향으로 이동시키면서 교체될 페이지 선정
  - 교체 원칙 (참조비트 : r, 갱신 비트 m)

① 현재 포인터가 가리키고 있는 페이지의 r과 m을 검사

② (r, m) = (0, 0) 이면 해당 페이지를 교체 대상으로 선정, 포인터를 한 단계 진행시킨 후 선정 과정 종료

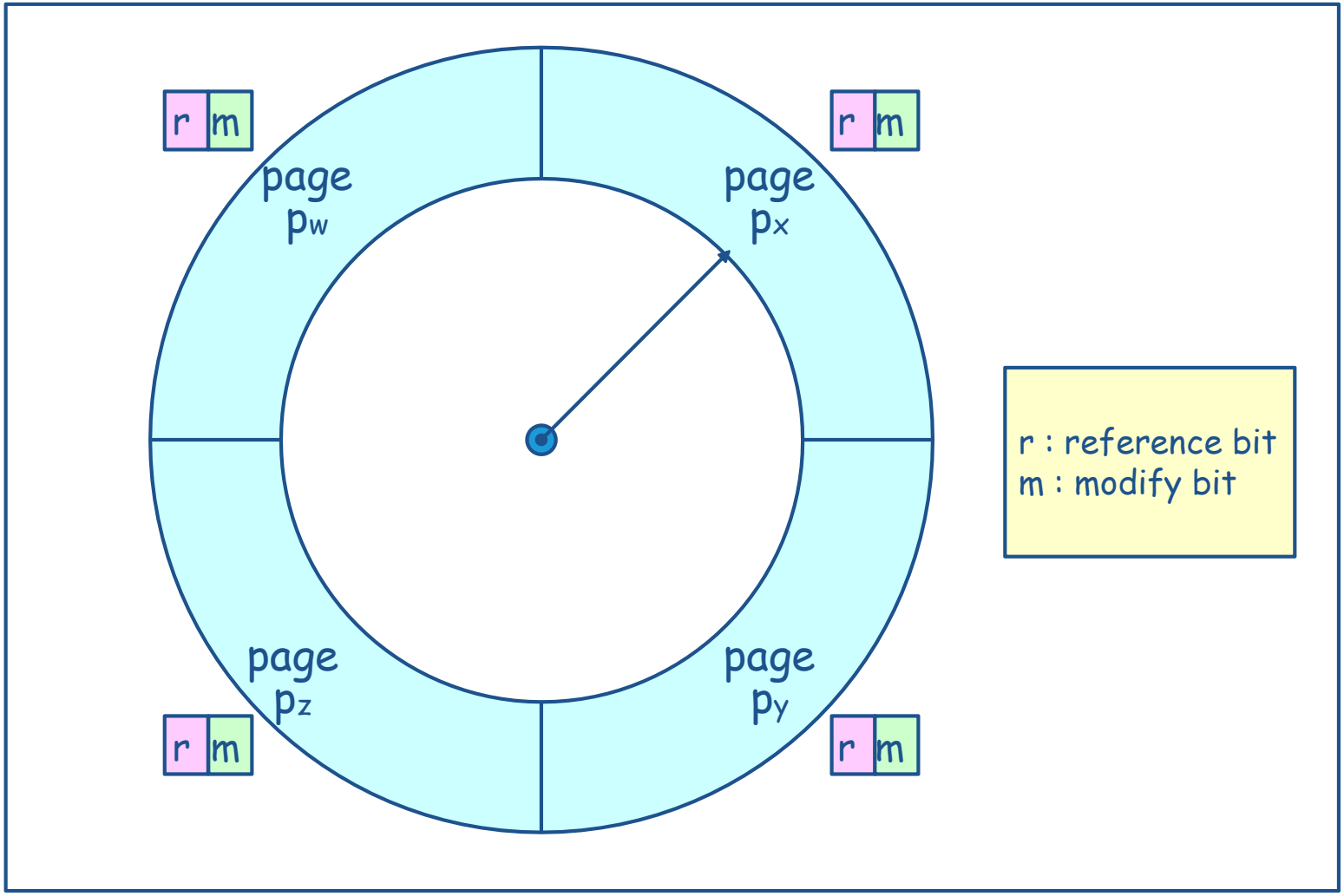
③ (r, m) = (0, 1) 이면 (0, 0)으로 설정하고 해당 페이지를 소거 대상 리스트에 추가한 뒤 ⑥ 단계로 이동

④ (r, m) = (1, 0) 이면 (0, 0)으로 설정하고 ⑥ 단계로 이동

⑤ (r, m) = (1, 1) 이면 (0, 1)로 설정하고 ⑥ 단계로 이동

⑥ 포인터를 한 단계 진행 시킨 후 ① 단계부터 반복

Second chance 알고리즘의 개념



Second chance 알고리즘 사용시 페이지 부재 발생 횟수 분석

- 가정
  - 프로세스에게 4개의 페이지 프레임이 고정 할당됨
  - 초기에 4개의 페이지 프레임에 페이지 a, b, c, d가 적재되어 있음
  - 참조 비트는 모두 1로, 갱신 비트는 모두 0으로 설정되어 있음

$\omega = c\ a^W\ d\ b^W\ e\ b\ a^W\ b\ c\ d$

Second chance 기법에 의한 주기억장치 상태 변화 과정 예

시간		0	1	2	3	4	5	6	7	8	9	10
참조 스트링			c	a^W	d	b^W	e	b	a^W	b	c	d
주기억 장치 상태	frame 0	→a/10	→a/10	→a/11	→a/11	→a/11	a/00	a/00	a/11	a/11	→a/11	a/00
	frame 1	b/10	b/10	b/10	b/10	b/11	b/00	b/10	b/10	b/10	b/10	d/10
	frame 2	c/10	c/10	c/10	c/10	c/10	e/10	e/10	e/10	e/10	e/10	→e/00
	frame 3	d/10	d/10	d/10	d/10	d/10	→d/00	→d/00	→ d/00	→ d/00	c/10	c/00
페이지 부재 발생 여부 P2nd-chance Q2nd-chance							F e c				F c d	F d b

- W 표시 : 페이지를 갱신하는 형태의 접근인 경우 표시
- 밑줄친 부분 : 해당 페이지가 소거 대상 리스트에 추가되었음을 의미함
- P2nd-chance : 해당 시점에 주기억장치에 적재되는 페이지
- Q2nd-chance : 해당 시점에 주기억장치로부터 교체되는 페이지

고정 할당 기반의 교체 기법

기타 알고리즘

- MRU(Most Recently Used) 알고리즘
  - LRU 알고리즘에 상대되는 기법
- MFU(Most Frequently Used) 알고리즘
  - LFU 알고리즘에 상대되는 기법

❑ 이론상으로만 존재하며 현실적으로 사용되지 않는 기법들임

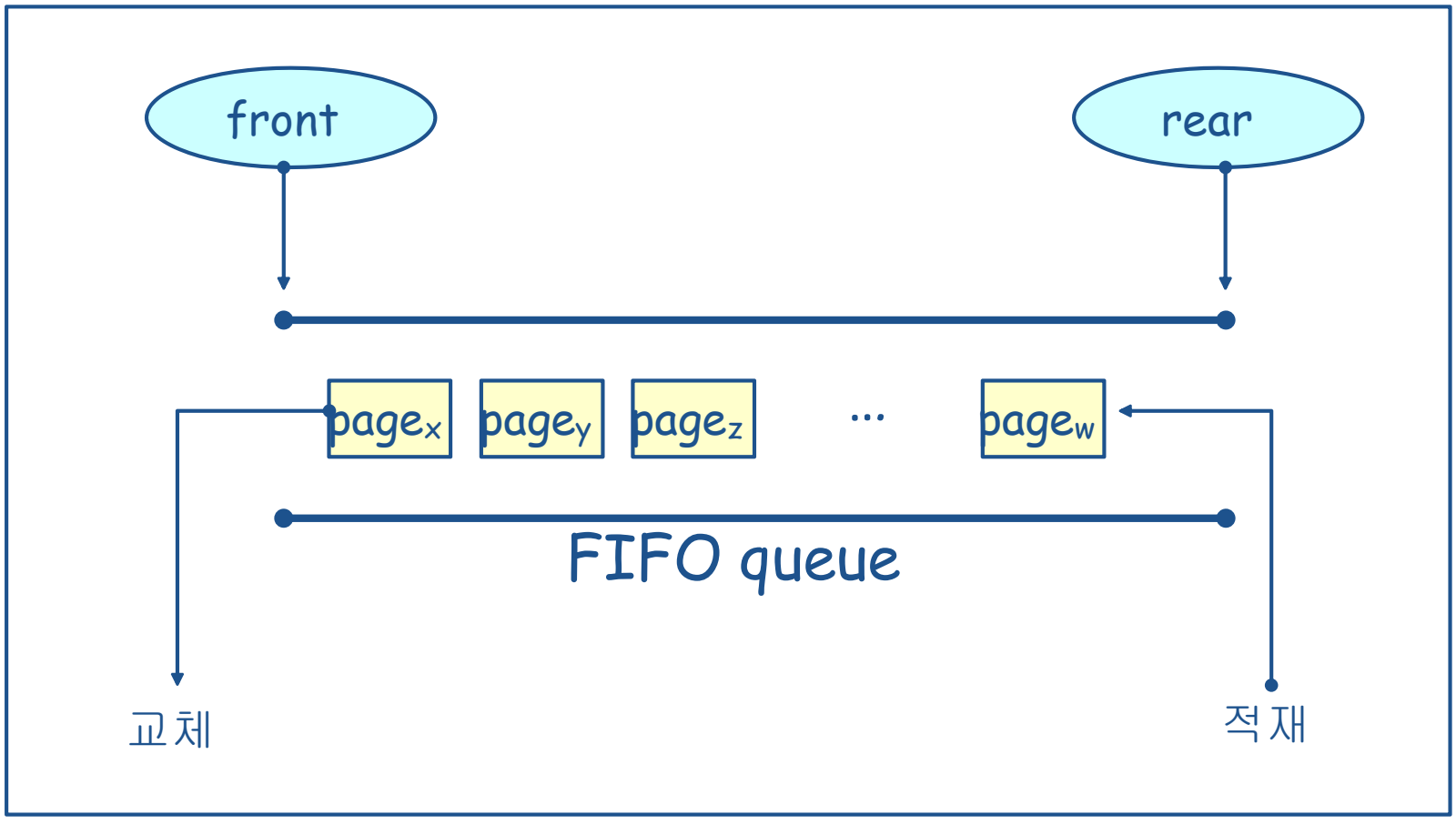
고정 할당 기반의 체 기법

- 페이지 교체 기법 선정시 고려사항
  - 합당한 기준을 사용하고 있는지
  - 선정을 위한 오버헤드가 크지 않은지 등
- 페이지 교체 기법의 구현
  - 각 기법 별로 차이 있음
  - 시스템마다 다를 수 있음

고정 할당 기반 교체 기법의 구현

- FIFO 알고리즘의 구현
  - 적재 시간 기록(time-stamping) 기법
    - 실제 적재 시간 기록
    - 적재된 상대적인 순서를 기록
      - 하나의 카운터를 두고 페이지 교체 시마다 카운터 값을 증가시켜 적재되는 페이지에 대하여 그 당시의 카운터 값을 기록
    - 시간의 기록은 프로세스의 PMT에 필드 추가로 가능
  - FIFO 큐(queue) 사용 기법
    - 프로세스마다 페이지 번호들을 기록하기 위한 큐를 둠
    - 페이지 교체가 필요한 경우
      - 프로세스 큐의 전방에 있는 페이지를 교체하고 큐에서 삭제
    - 새로 적재되는 페이지 번호
      - 프로세스 큐의 후방에 삽입

큐를 이용한 FIFO 교체 알고리즘의 구현



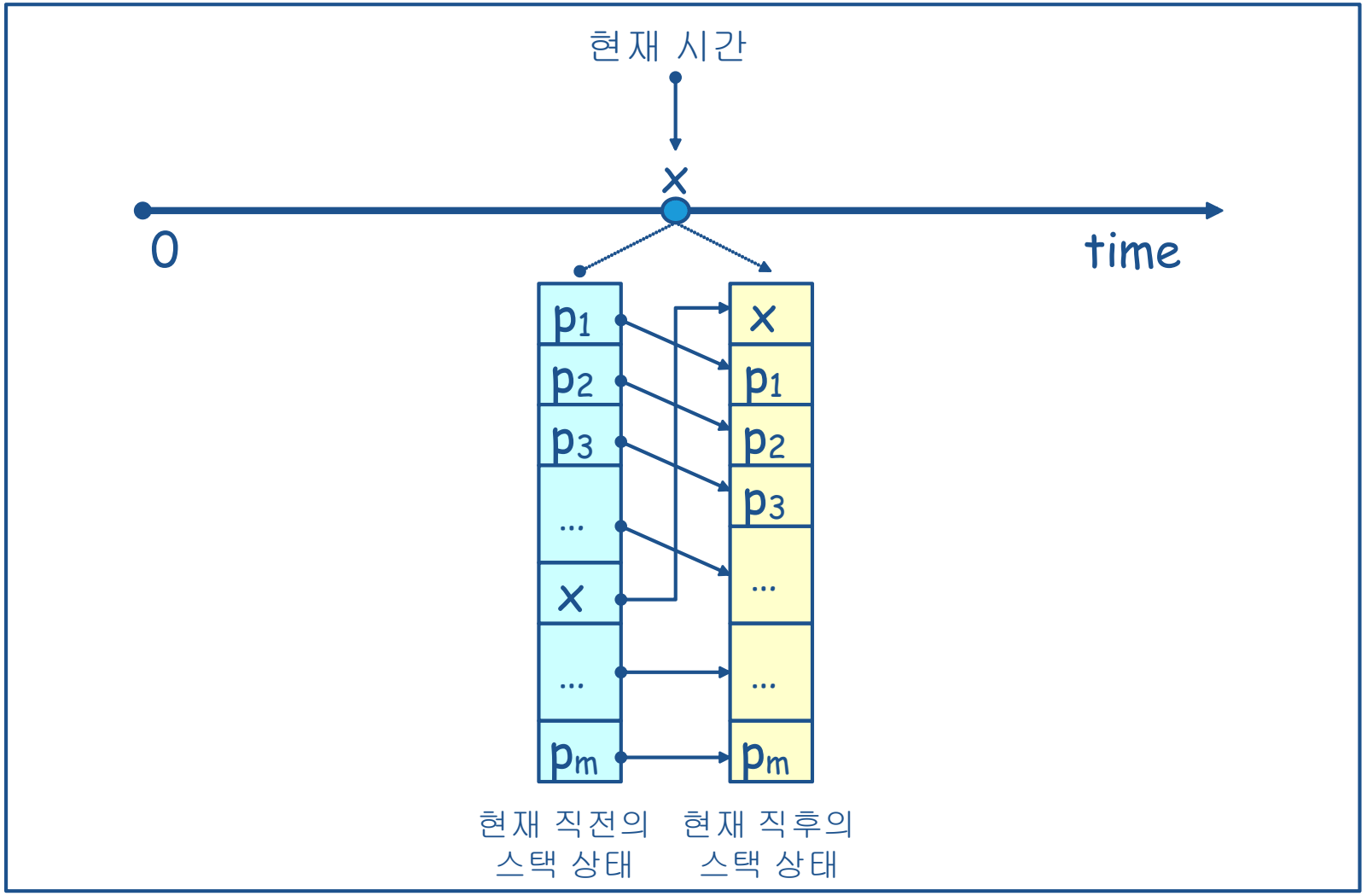
고정 할당 기반 교체 기법의 구현

- LRU 알고리즘의 구현
  - 카운터(counter) 사용기법
    - 각 프로세스마다 PMT에 카운트 필드 추가
    - 프로세서에 클럭 또는 카운터를 두고 매순간 값을 증가시킴
    - 각 페이지 참조 시마다 프로세서의 클럭(카운터) 값을 해당 프로세스의 PMT에 기록
    - 각 페이지들에 대해 최근 참조된 상대적인 순서 확인 가능
    - 교체 페이지를 선정하기 위해서 PMT에 대한 탐색이 필요함

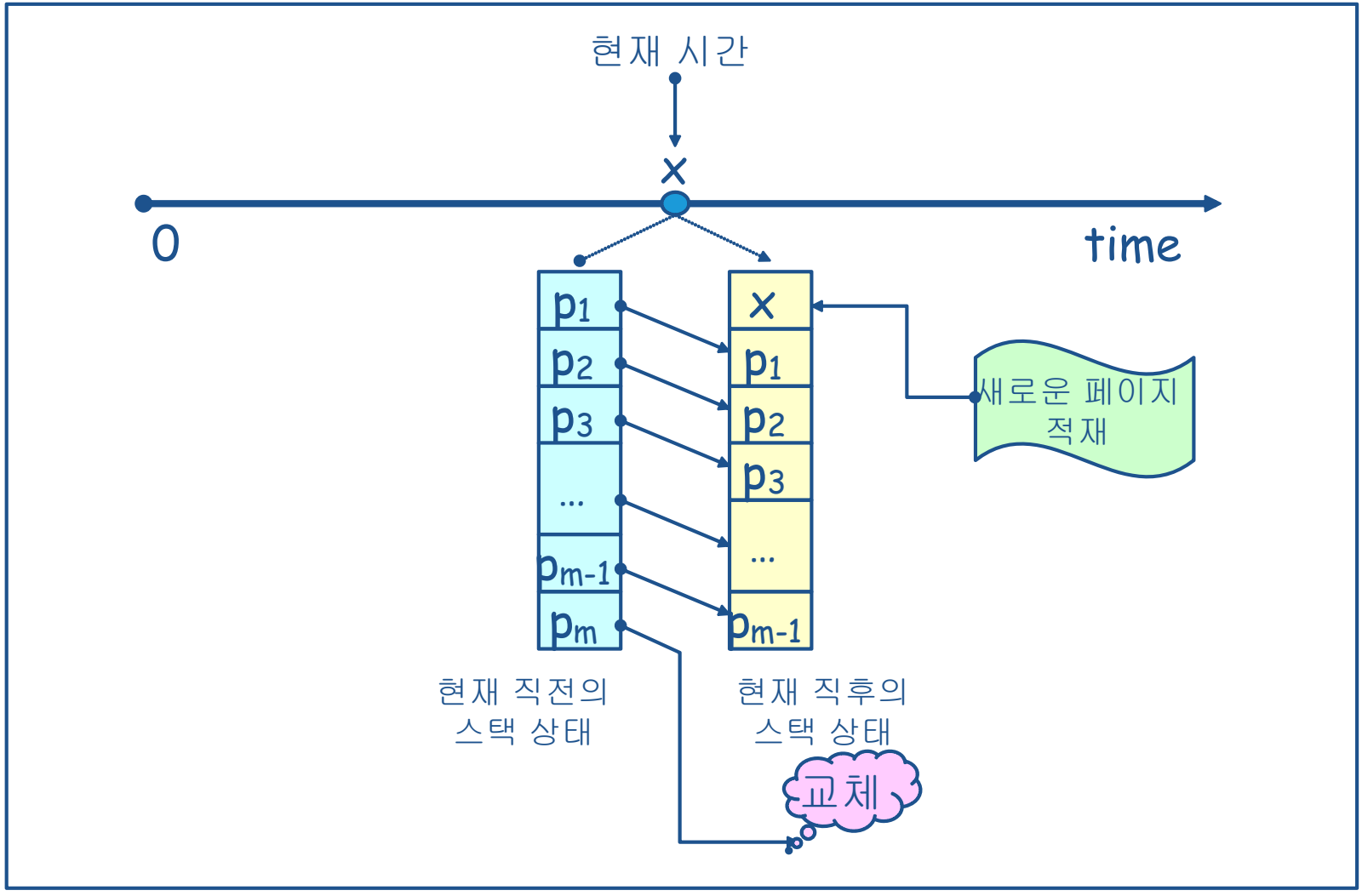
고정 할당 기반 교체 기법의 구현

- LRU 알고리즘의 구현
  - 스택(stack) 사용 기법
    - 스택
      - 각 프로세스마다 페이지 번호를 저장하는 스택을 둠
      - 스택의 중간에서도 페이지 번호의 삭제가 가능한 구조
      - 현재 적재된 페이지들의 번호를 최근 참조된 순서대로 유지
    - 페이지 부재가 발생하지 않은 경우
      - 스택의 해당 페이지 번호를 삭제하고 이를 다시 스택의 top에 삽입
    - 페이지 부재가 발생한 경우
      - 스택의 bottom에 있는 번호의 페이지를 교체 대상으로 선정하고 스택에서 삭제한 후, 새로 적재된 페이지 번호를 스택의 top에 삽입

페이지 부재가 발생하지 않은 경우의 스택의 변화



페이지 부재가 발생한 경우의 스택의 변화



LRU 알고리즘을 위한 스택을 구현할 경우 스택의 변화과정의 예

- 가정
  - 프로세스에게 4개의 페이지 프레임이 고정 할당됨
  - 초기에 4개의 페이지 프레임들이 모두 비어 있음

ω = 1 2 6 1 4 5 1 2 1 4 5 6 4 5

LRU 기법에서의 스택의 변화 과정 예

시간	1	2	3	4	5	6	7	8	9	10	11	12	13	14
참조 스트링	1	2	6	1	4	5	1	2	1	4	5	6	4	5
스택	1	2	6	1	4	5	1	2	1	4	5	6	4	5
		1	2	6	1	4	5	1	2	1	4	5	6	4
			1	2	6	1	4	5	5	2	1	4	5	6
					2	6	6	4	4	5	2	1	1	1
페이지 부재 발생 여부	F	F	F		F	F		F				F		

❑ 페이지 부재의 횟수 : 총 7번 발생

가변 할당 기반의 교체 기법

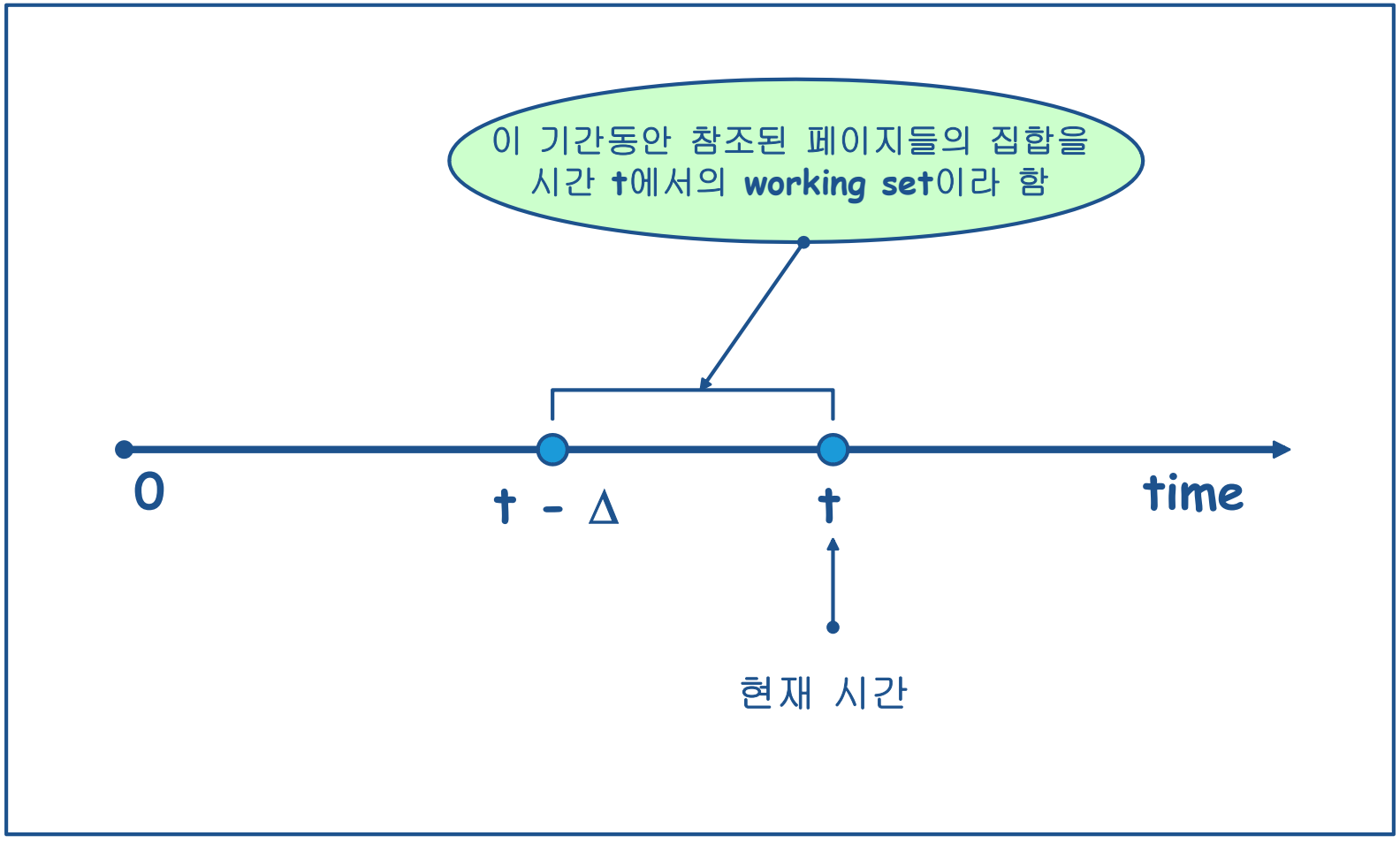
Working set 기법

- 1968년 Denning에 의해 제안된 개념
- working set
  - 프로세스가 특정 시점에 집중적으로 참조하는 페이지들의 집합
  - 최근 일정 시간동안 참조한 페이지들의 집합
  - 프로세스의 working set은 시간이 지남에 따라 변함

정의 : Working set

□ 특정 프로세스에 대해 시간 t에서의 **working set**, $W(t, \Delta)$
- 해당 프로세스가 시간 $[t-\Delta, t]$동안 참조한 페이지들의 집합
(Δ : **window size**, 시스템 파라메타, 시스템의 특성에 따라 미리 값이 결정됨)

Working set 의 개념



가변 할당 기반의 교체 기법

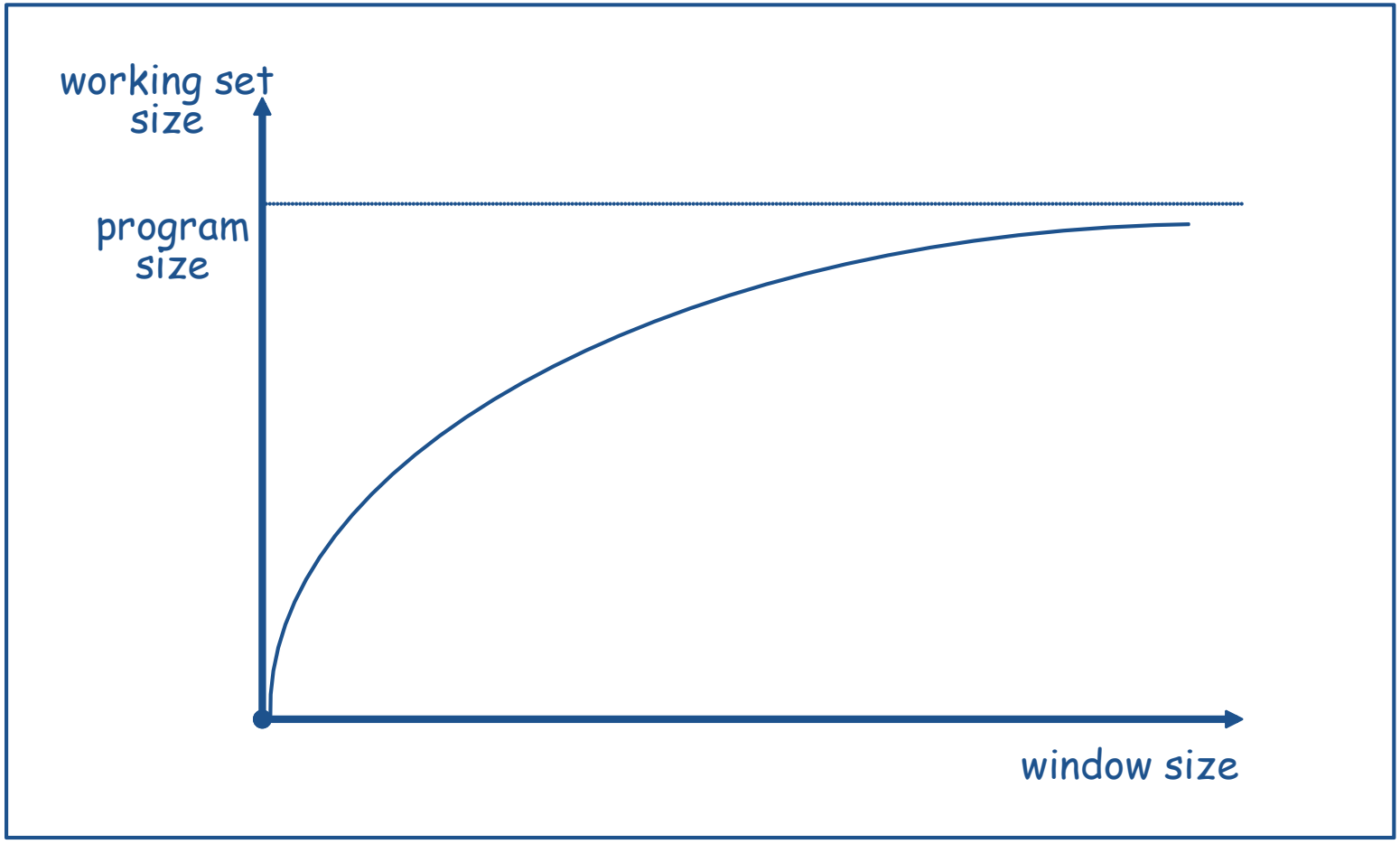
- Working set 기억장치 관리 기법
  - 지역성 기반
  - Working set을 모두 주기억장치에 적재시킴으로써 프로세스로 하여금 페이지 부재를 거의 발생시키지 않고 실행할 수 있도록 하는 기법
  - 스래싱(thrashing)현상으로 인한 시스템 성능 저하 방지 목적

정의 : **Working set** 기억장치 관리 기법

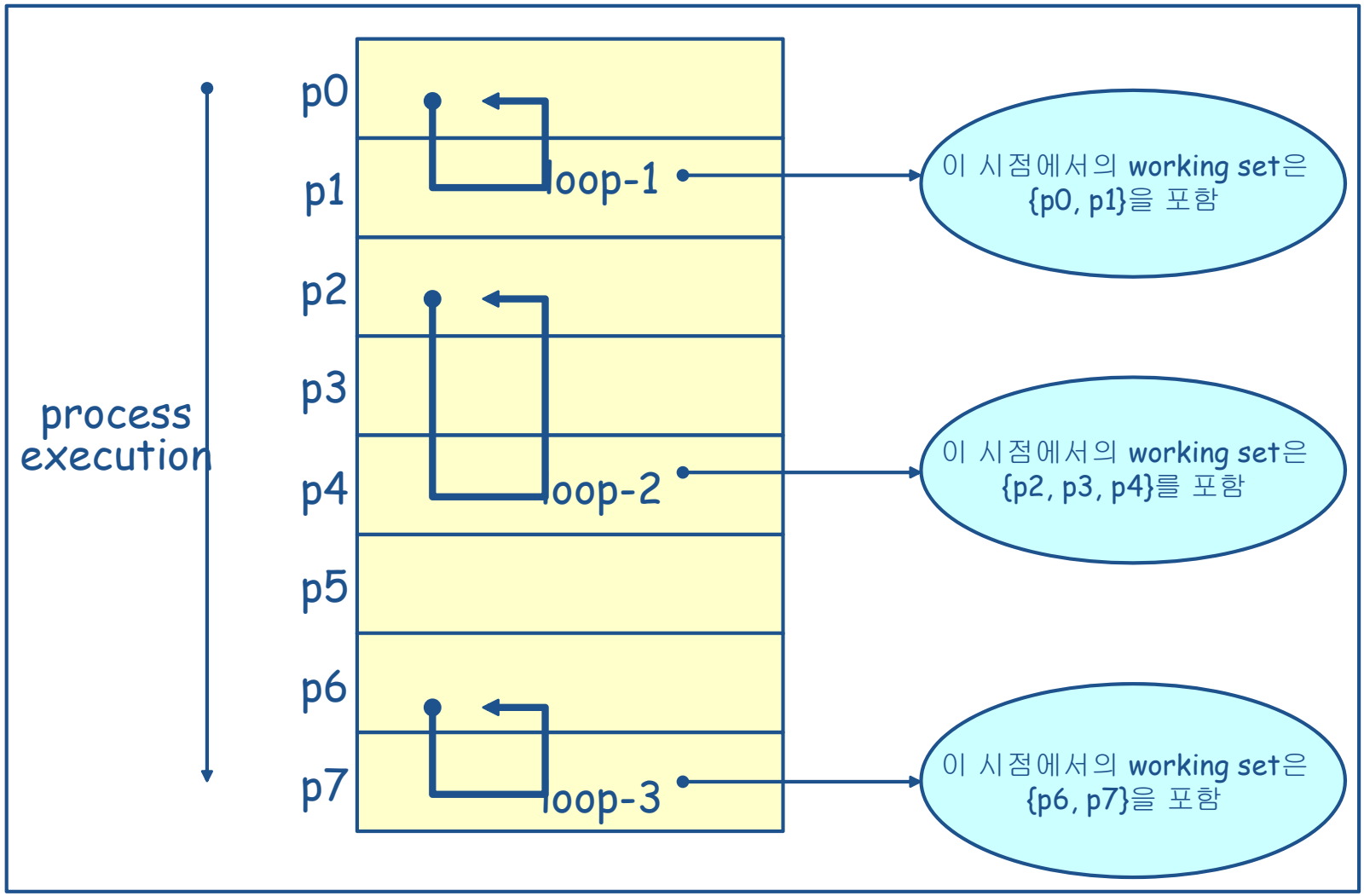
❑ 프로세스가 실행하는 동안 매순간 그 프로세스의 **working set**이 주기억장치에 적재되어 있도록 하는 기법

❑ **working set** 기억장치 관리 기법에서는 윈도우 크기 Δ의 결정이 중요함
- 일반적인 원칙 없음
- 해당 시스템의 주기억장치 용량이나 실행되는 프로세스들의 특성등 고려

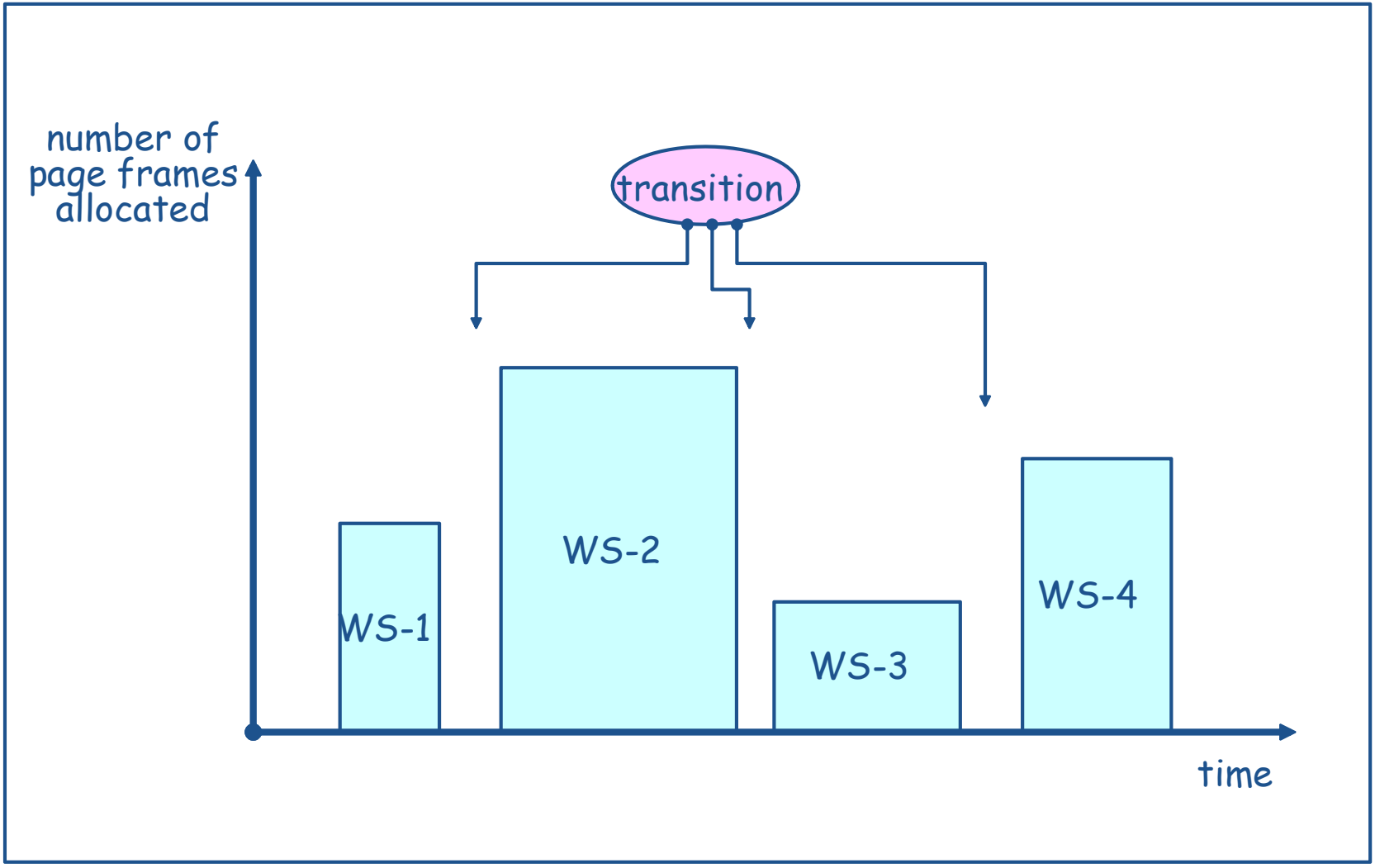
윈도우 크기와 working set 크기와의 관계



Working set 변화의 예



Working set 변화 과정



Working set 기억장치 관리 기법 하에서의 주기억장치 상태 변화과정 예

- 가정
  - 윈도우 크기(Δ) = 3, 총 5개의 페이지(0, 1, 2, 3, 4)로 구성
  - 임의의 시점(시간 0)에 3개의 페이지 {0, 3, 4}가 적재되어 있음
  - 시간 0 : 페이지 0, 시간 -1 : 페이지 3, 시간 -2 : 페이지 4 참조

ω = 2 2 3 1 2 4 2 4 0 3

시간		-2	-1	0	1	2	3	4	5	6	7	8	9	10
참조 스트링		4	3	0	2	2	3	1	2	4	2	4	0	3
주기억 장치 상태	페이지 0	?	?	0	0	0	0	-	-	-	-	-	0	0
	페이지 1	?	?	-	-	-	-	1	1	1	1	-	-	-
	페이지 2	?	?	-	2	2	2	2	2	2	2	2	2	2
	페이지 3	?	?	3	3	3	3	3	3	3	-	-	-	3
	페이지 4	?	?	4	4	-	-	-	-	4	4	4	4	4
페이지 부재 발생 여부		?	?	?	F			F		F			F	F
Pws		?	?	?	2			1		4			0	3
Qws		?	?	?		4		0			3	1		
주기억장치 할당량		?	?	-	4	3	3	3	3	4	3	2	3	4

- Pws : 주기억장치에 적재되는 페이지들
- Qws : 주기억장치로부터 교체되는 페이지들

가변 할당 기반의 교체 기법

- Working set 기억장치 관리 기법의 성능 평가
  - 단순히 페이지 부재의 발생 횟수만으로 평가 불가능

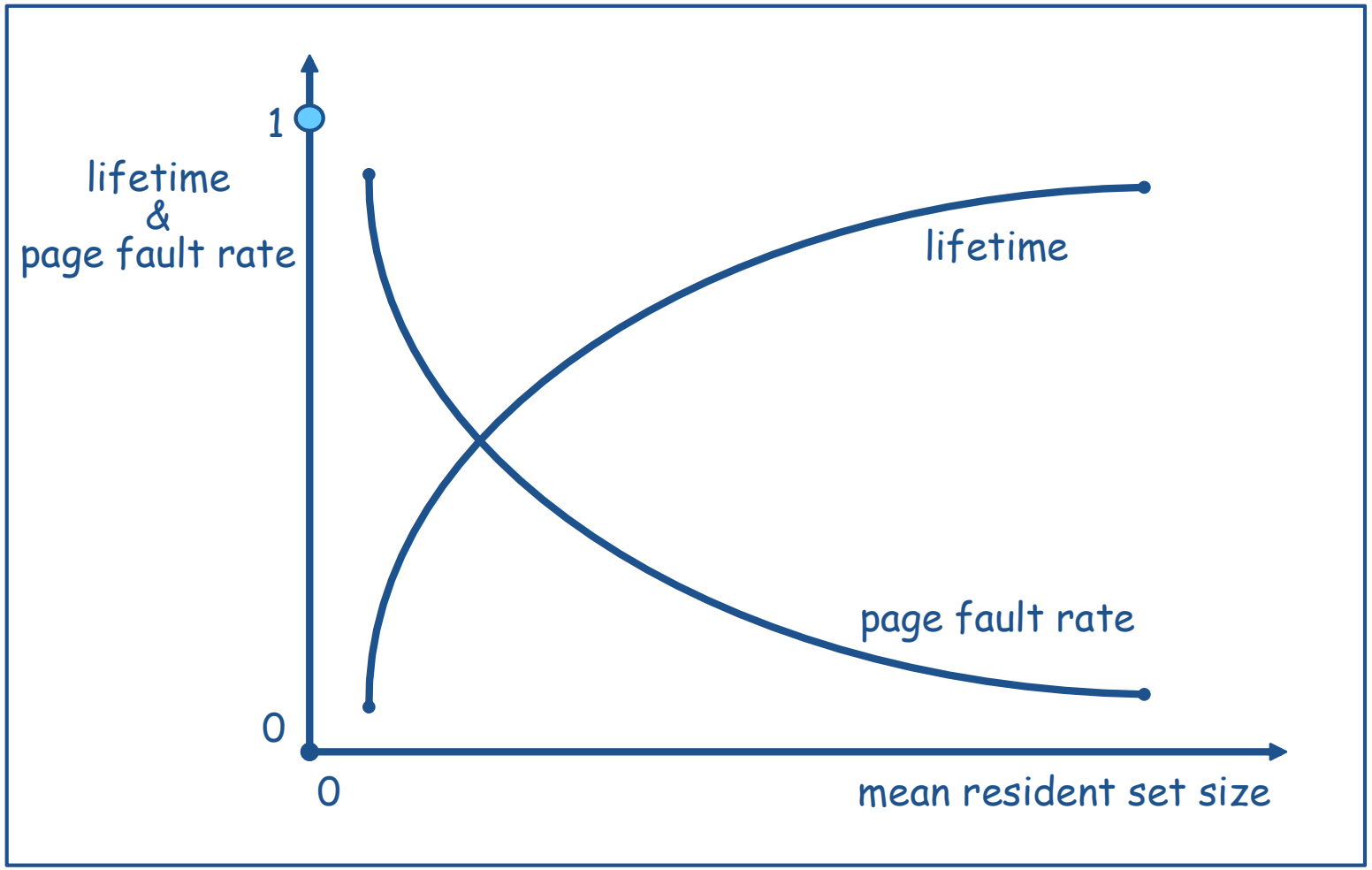
❑ 결과
- 시간 [1, 10]동안의 페이지 부재 발생 횟수 = 5
- 시간 [1, 10]동안의 평균 주기억장치 할당량 = 3.2

❑ 평가
- 평균 3.2개의 페이지 프레임을 할당받은 상태에서 5회의 페이지 부재를 발생시킴

가변 할당 기반의 교체 기법

- Working set 기억장치 관리 기법의 특성
  - 추가로 적재되는 페이지가 없어도 교체되는 페이지가 있을 수 있음
  - 추가로 적재되는 페이지가 있어도 교체되는 페이지가 없을 수 있음
- Working set 기억장치 관리 기법의 단점
  - 매순간 페이지 참조가 있을 때마다 상주 집합(residence set), 즉, 주기억장치에 적재되는 페이지들의 집합을 조정해야 하는 오버헤드 발생
  - 페이지 부재와 관계없이 상주 집합 조정

평균 주기억장치 할당량과 페이지 부재율과의 관계



가변 할당 기반의 교체 기법

- PFF(Page Fault Frequency) 기법
  - 상주 집합 결정시 프로세스의 페이지 부재율을 반영
    - 프로세스의 페이지 부재율이 낮을 경우
      (inter-fault time이 큰 경우)
      – 할당되는 페이지 프레임의수를 줄임
    - 프로세스의 페이지 부재율이 높을 경우
      (inter-fault time이 작은 경우)
      – 페이지 프레임을 추가로 할당
  - 주기억장치 할당 공간 양의 변경, 상주 집합의 조정
    - 페이지 부재가 발생했을 때에만 실행하여 오버헤드 줄임

가변 할당 기반의 교체 기법

PFF(Page Fault Frequency) 기법

- 페이지 부재율의 고저 판단 기준

페이지 부재율의 고저 판단	
IFT > τ	페이지 부재율이 낮다고 판단함
IFT <= τ	페이지 부재율이 높다고 판단함

- 임계값(threshold value, τ)
  - 시스템 파라메타
  - 미리 정해진 시간 간격
  - 페이지 부재 발생 간격(IFT)의 고저 판단을 위한 기준치

PFF 기법이 기억장치를 관리하는 구체적인 과정

⑴ 프로세스 실행도중 페이지 부재가 발생하면 IFT 계산

$$IFT = t_{c-1} - t_c$$

(IFT : 페이지 부재 발생 시간 간격(inter-fault time)
t_{c-1} : 직전 페이지 부재 발생 시간
t_c : 현재 페이지 부재 발생 시간)

⑵ $IFT > \tau$ 인 경우

- 시간 t_{c-1} 부터 시간 t_c 까지 참조된 페이지들만 주기억장치에 적재
- 나머지 페이지들은 모두 교체 시킴

⑶ $IFT \leq \tau$ 인 경우

- 기존의 상주 집합은 그대로 주기억장치에 유지
- 현재 참조된 페이지를 주기억장치에 추가로 적재
  (기존에 프로세스에게 할당되어 있던 페이지 프레임의 수를 증가시키는 효과 초래)

PFF 기억장치 관리 기법하에서의 주기억장치 상태 변화과정 예

- 가정
  - 임계값(τ) = 2, 총 5개의 페이지(0, 1, 2, 3, 4)로 구성
  - 임의의 시점(시간 0)에 3개의 페이지 {0, 3, 4}가 적재되어 있음
  - 시간 0 : 페이지 0, 시간 -1 : 페이지 3, 시간 -2 : 페이지 4 참조

ω = 2 2 3 1 2 4 2 4 0 3

시간		-2	-1	0	1	2	3	4	5	6	7	8	9	10
참조 스트링		4	3	0	2	2	3	1	2	4	2	4	0	3
주기억 장치 상태	페이지 0	0	0	0	0	0	0	-	-	-	-	-	0	0
	페이지 1	-	-	-	-	-	-	1	1	1	1	1	-	-
	페이지 2	-	-	-	2	2	2	2	2	2	2	2	2	2
	페이지 3		3	3	3	3	3	3	3	3	3	3	-	3
	페이지 4	4	4	4	4	4	4	-	-	4	4	4	4	4
페이지 부재 발생 여부 P_{PFF} Q_{PFF}		F 4 ?			F 2			F 1 0,4		F 4			F 0 1,3	F 3
주기억장치 할당량		-	-	-	4	4	4	3	3	4	4	4	3	4

- P_{PFF} : 주기억장치에 적재되는 페이지들
- Q_{PFF} : 주기억장치로부터 교체되는 페이지들

가변 할당 기반의 교체 기법

- PFF 기억장치 관리 기법의 성능 평가
  - 단순히 페이지 부재의 발생 횟수만으로 평가 불가능

  ❑ 결과
  - 시간 [1, 10]동안의 페이지 부재 발생 횟수 = 5
  - 시간 [1, 10]동안의 평균 주기억장치 할당량 = 3.7

  ❑평가
  - 평균 3.7개의 페이지 프레임을 할당받은 상태에서 5회의 페이지 부재 발생시킴

- PFF 기억장치 관리 기법의 특성
  - 페이지 부재가 발생한 경우에만 주기억장치 상태가 변함

가변 할당 기반의 교체 기법

- VMIN(Variable MIN) 알고리즘
  - 가변 할당 기반의 페이지 교체 기법들 중에서 최적의 성능을 내기 위한 기법
    - 최적 : 단순히 페이지 부재의 발생 횟수의 최소화뿐만 아니라 실행 시간 동안의 평균 주기억장치 할당량도 고려한 기준
  - 미래의 페이지 참조 스트링이 미리 알려져 있어야 하는 기법
  - 실현 불가능한 기법
  - 프로세스가 실행하는 과정의 매 시점 t마다 $[t, t+\Delta]$에 해당하는 미래의 시간 간격 동안의 페이지 참조 스트링을 보고 주기억장치 상태를 변화 시킴

VMIN 기법에 의한 기억장치 관리 기법의 구체적인 과정

⑴ 임의의 시점 t에 페이지 r이 참조되면 이후 [t, t+ Δ]에 해당하는 미래의 시간 간격 동안 페이지 r이 다시 참조되는지 검사

⑵ [t, t+ Δ]의 시간 동안 페이지 r이 다시 참조되는 경우 그 때까지 페이지 r을 주기억장치에 유지시킴

⑶ [t, t+ Δ]의 시간 동안 페이지 r이 다시 참조되지 않는 경우 페이지 r을 참조한 후 즉시 이를 교체시킴

VMIN 기억장치 관리 기법하에서의 주기억장치 상태 변화과정 예

- 가정
  - 윈도우 크기(Δ) = 4, 총 5개의 페이지(0, 1, 2, 3, 4)로 구성
  - 초기(시간 0)에 페이지 3만 적재되어 있음

ω = 2 2 3 1 2 4 2 4 0 3

시간		0	1	2	3	4	5	6	7	8	9	10
참조 스트링		0	2	2	3	1	2	4	2	4	0	3
주기억 장치 상태	페이지 0	-	-	-	-	-	-	-	-	-	0	-
	페이지 1	-	-	-	-	1	-	-	-	-	-	-
	페이지 2	-	2	2	2	2	2	2	2	-	-	-
	페이지 3	3	3	3	3	-	-	-	-	-	-	3
	페이지 4	-	-	-	-	-	-	4	4	4	-	-
페이지 부재 발생 여부 PVMIN QVMIN			F 2			F 1 3	 1	F 4		 2	F 0 4	F 3 0
주기억장치 할당량		-	2	2	2	2	1	2	2	1	1	1

- PVMIN : 주기억장치에 적재되는 페이지들
- QVMIN : 주기억장치로부터 교체되는 페이지들

가변 할당 기반의 교체 기법

- VMIN 기억장치 관리 기법의 성능 평가
  - 단순히 페이지 부재의 발생 횟수만으로 평가 불가능

❑ 결과
- 시간 [1, 10]동안의 페이지 부재 발생 횟수 = 5
- 시간 [1, 10]동안의 평균 주기억장치 할당량 = 1.6

❑ 평가
- 평균 1.6개의 페이지 프레임을 할당받은 상태에서 5회의 페이지 부재 발생시킴

가변 할당 기반의 교체 기법

- VMIN 기법이 최적의 성능을 내도록 하기 위한 Δ 값 결정

$$\Delta = \frac{R}{U}$$

  - U : 한 번의 주기억장치 참조 시간 동안 한 페이지를 주기억장치에 유지시키는데 드는 비용
  - R : 페이지 부재를 서비스하는데 드는 비용

기타 고려 사항

- 페이지 크기
  - 시스템마다 다르게 결정됨
    - 시스템의 특성에 따라
    - 실행되는 프로세스들의 특성에 따라
  - 일반적인 페이지 크기
    - 2^7 Bytes ~ 2^{14} Bytes
  - 장단점 비교
    - 페이지 크기를 작게 하는 경우
      - 테이블 단편화 유발로 주기억장치가 많이 소모됨
      - 페이지 프레임 개수 증가로 커널의 관리 오버헤드 발생
      - 내부 단편화를 줄일 수 있음
      - 전체적으로 디스크와 주기억장치간의 전송시간 증가
      - 사용자 프로그램중 필요한 부분만 적재될 가능성 많아짐
    - 페이지 크기를 크게 하는 경우
      - 불필요한 부분까지 적재되어 주기억장치 공간의 낭비 초래
      - etc

기타 고려 사항

- 프로그램 구조 재구성(program restructuring)
  - 개념
    - 사용자 또는 프로그래머가 시스템의 기억장치 관리 기법이나 구조에 대해 어느 정도의 지식을 가지고 있다면, 자신이 실행시키고자 하는 프로그램이 보다 빠르게 실행을 마칠 수 있도록 프로그램의 구조를 재구성할 수 있음
  - 필요 정보
    - 시스템의 페이지 크기
    - 배열의 저장 형태
    - 기억장치 관리 기법 등

프로그램 구조 재구성 개념의 예

- 가정
  - 페이지 크기가 1KB인 페이징 시스템
  - 정수형(int) 데이터의 크기 : 4 Bytes

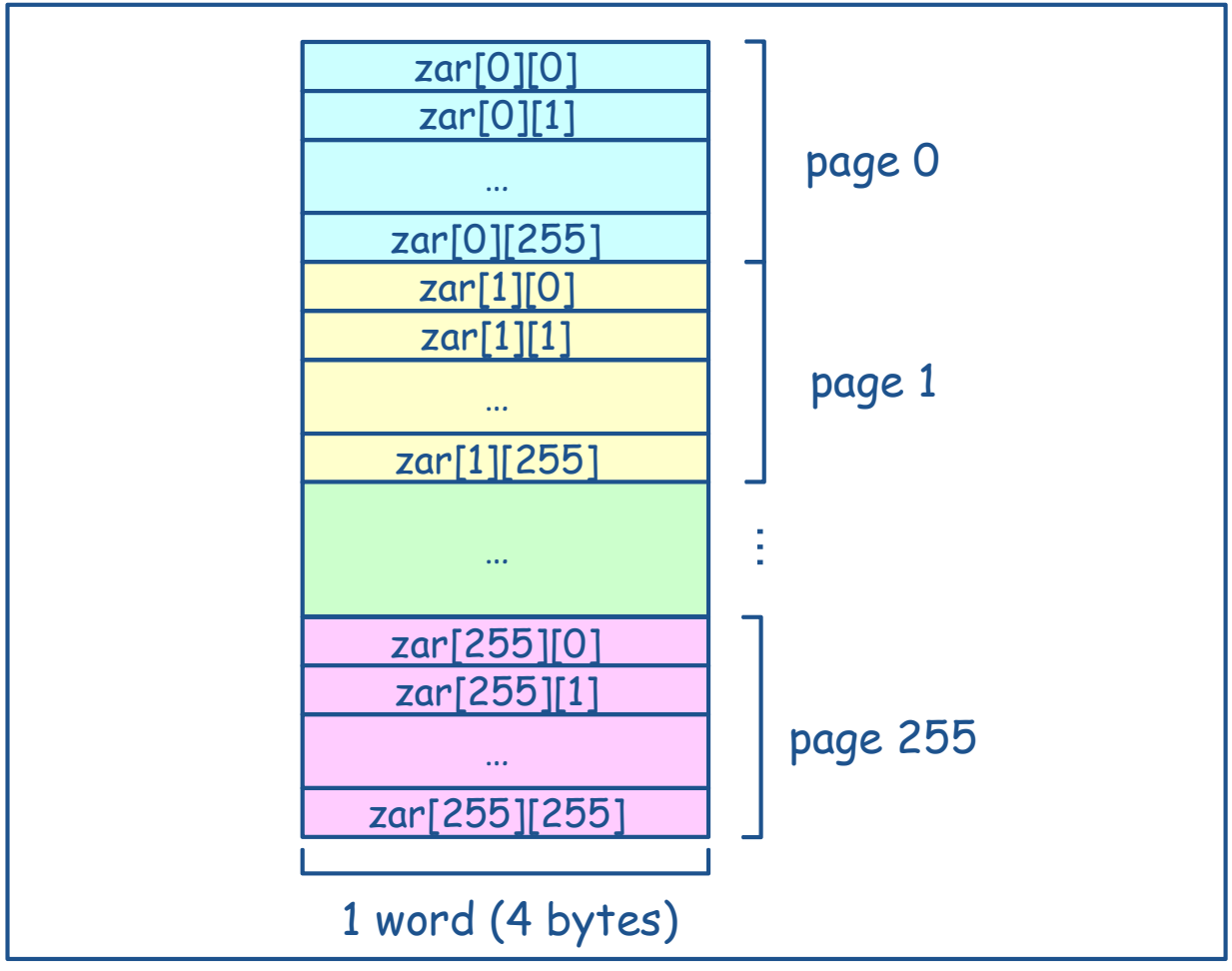
```
// 프로그램-1

int main()
{
  int zar[256][256];
  int i, j;

  for(j = 0; j < 256; j++)
      for(i = 0; i < 256; i++)
          zar[i][j] = 0;

  return 0;
}
```

C 언어 프로그램에서의 행우선 순위 배열 저장 형태



- 배열 zar이 0번 페이지부터 저장됨을 가정함

프로그램 구조 재구성

```
// 프로그램-1

int main()
{
  int zar[256][256] ;
  int i, j;

  for (j = 0; j < 256; j++)
      for (i = 0; i < 256; i++)
          zar[i][j] = 0;

  return 0;
}
```

```
// 프로그램-2

int main()
{
  int zar[256][256] ;
  int i, j;

  for (i = 0; i < 256; i++)
      for (j = 0; j < 256; j++)
          zar[i][j] = 0;

  return 0;
}
```

- 매번 반복 구문을 실행할 때마다
  0번 페이지부터 255번 페이지까지 번갈아 참조하며
  이 과정을 다시 256회 반복함
- 페이지 부재를 발생시키지 않고 효율적으로 실행되기 위해서는
  - 해당 프로세스에게 256개의 페이지 프레임 할당되어야 함
  - 매우 비현실적임

결론

- 가상기억장치의 관리 기법
  - 가상기억장치 관리 기법들의 목적
    - 운영 비용의 감소
    - 비용 모델 : 일반적으로 페이지 부재의 발생 횟수로 설정됨
    - 프로그램들이 갖는 지역성 이용
  - 시스템 성능에 영향을 미치는 주된 관리 기법
    - 할당 기법과 교체 기법
  - 고정 할당 기반의 교체 기법
    - 프로세스의 페이지 부재 발생 횟수를 줄이는 것이 일차적 목표
    - 실제로 많이 응용되고 있는 기법 : LRU 기법
  - 가변 할당 기반의 교체 기법
    - 페이지 부재의 발생 횟수뿐만 아니라
      각 프로세스에 대한 평균 주기억장치 할당량도 고려하여야 함
  - 기타 고려사항