쉽게 배우는

운영체제

2판

Chapter 08 가상 메모리의 기초

차례

학습목표

- 가상 메모리 시스템의 동작 원리와 매핑 테이블의 역할을 이해한다.
- 페이징 기법의 구현과 동적 주소 변환 과정을 알아본다.
- 세그먼테이션 기법의 구현과 동적 주소 변환 과정을 알아본다.

1-1 가상 메모리 시스템

■ 주방 규모를 고려한 레시피 개발의 어려움

- 새로운 레시피 개발할 때 요리사가 주방 크기까지 고려해야 한다면 제약 사항이 너무 많음 → 레시피대로 어떻게 조리할지는 레스토랑에서 해결할 문제
- 마찬가지로 프로그래머가 메모리 크기를 고려하여 프로그래밍하기는 어려움!

그림 8-1 주방 규모를 고려한 레시피 개발의 어려움

■ 가상 메모리

- 물리 메모리 크기와 상관없이 프로세스에 커다란 메모리 공간을 제공하는 기술
- 가상 메모리를 이용하면 프로세스는 운영체제가 어디에 있는지, 물리 메모리의 크기가 어느 정도인지 신경 쓰지 않고 메모리를 마음대로 사용할 수 있음

1-1 가상 메모리 시스템

■ 가상 메모리의 크기와 주소

- 프로세스가 바라보는 메모리 영역과 메모리 관리자가 바라보는 메모리 영역으로 나뉨
- 가상 메모리에서 메모리 관리자는 물리 메모리의 부족한 부분을 스왑 영역으로 보충

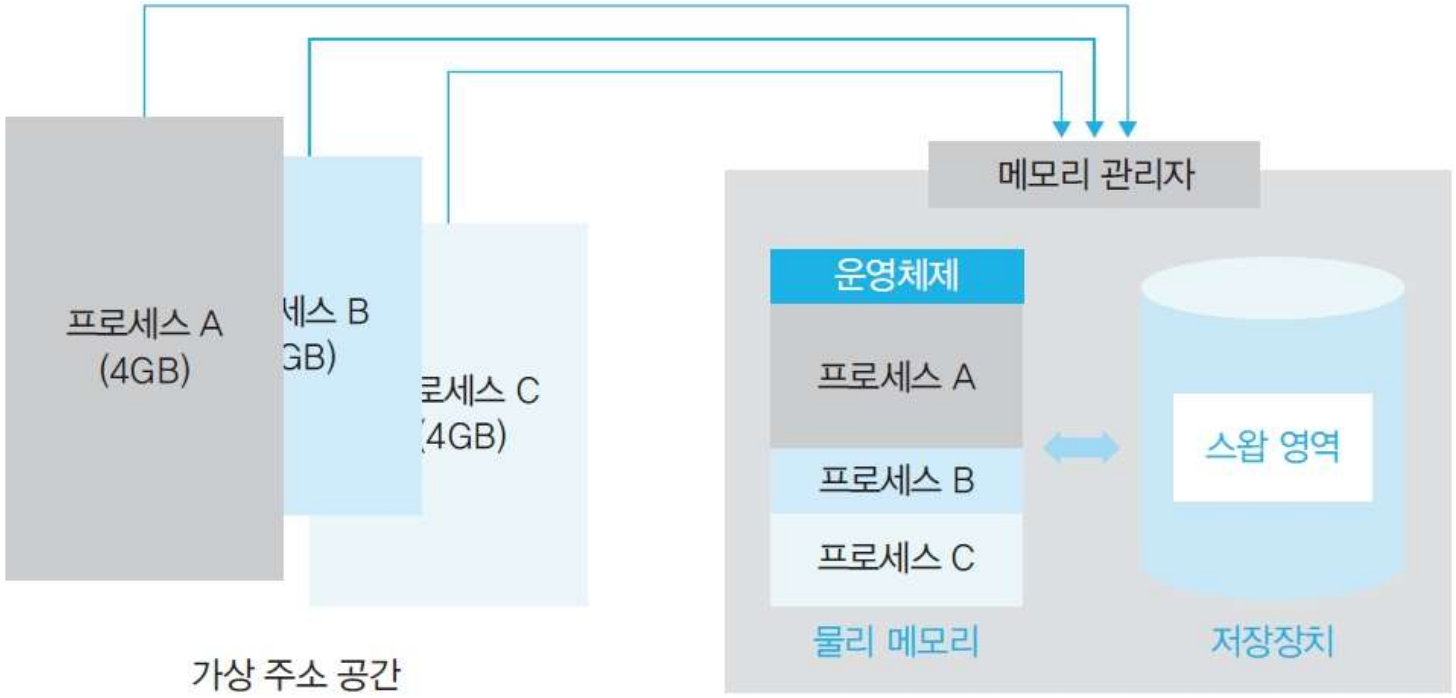


그림 8-2 가상 메모리의 구성

1-1 가상 메모리 시스템

■ 가상 메모리의 크기

정의 8-1 가상 메모리의 크기

가상 메모리에서 메모리 관리자가 사용할 수 있는 메모리의 전체 크기는 물리 메모리(실제 메모리)와 스왑 영역을 합한 크기다.

■ 동적 주소 변환(DAT; Dynamic Address Translation)

- 가상 주소를 실제 메모리의 물리 주소로 변환
- 동적 주소 변환을 거치면 프로세스가 아무 제약 없이 사용자의 데이터를 물리 메모리에 배치할 수 있음

1-2 매핑 테이블의 필요성과 역할

■ 매핑 테이블의 필요성(식당 예)

매핑 테이블

번호	의자	인원	주문
1번	1–4	4	코스1 ×4개
2번	대기 1	2	코스 2, 3
3번	5–7	3	코스 2×3개
4번	대기 2	3	코스 3, 4
⋮	⋮	⋮	⋮

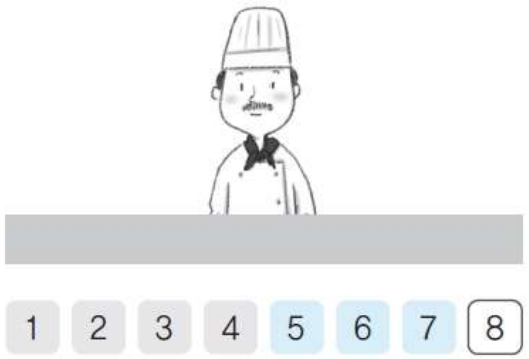


그림 8-3 손님-좌석 매핑 테이블

1-2 매핑 테이블의 필요성과 역할

■ 메모리 매핑 테이블

- 가상 메모리 시스템에서 메모리 관리자는 가상 주소와 물리 주소를 일대일 매핑 테이블로 관리

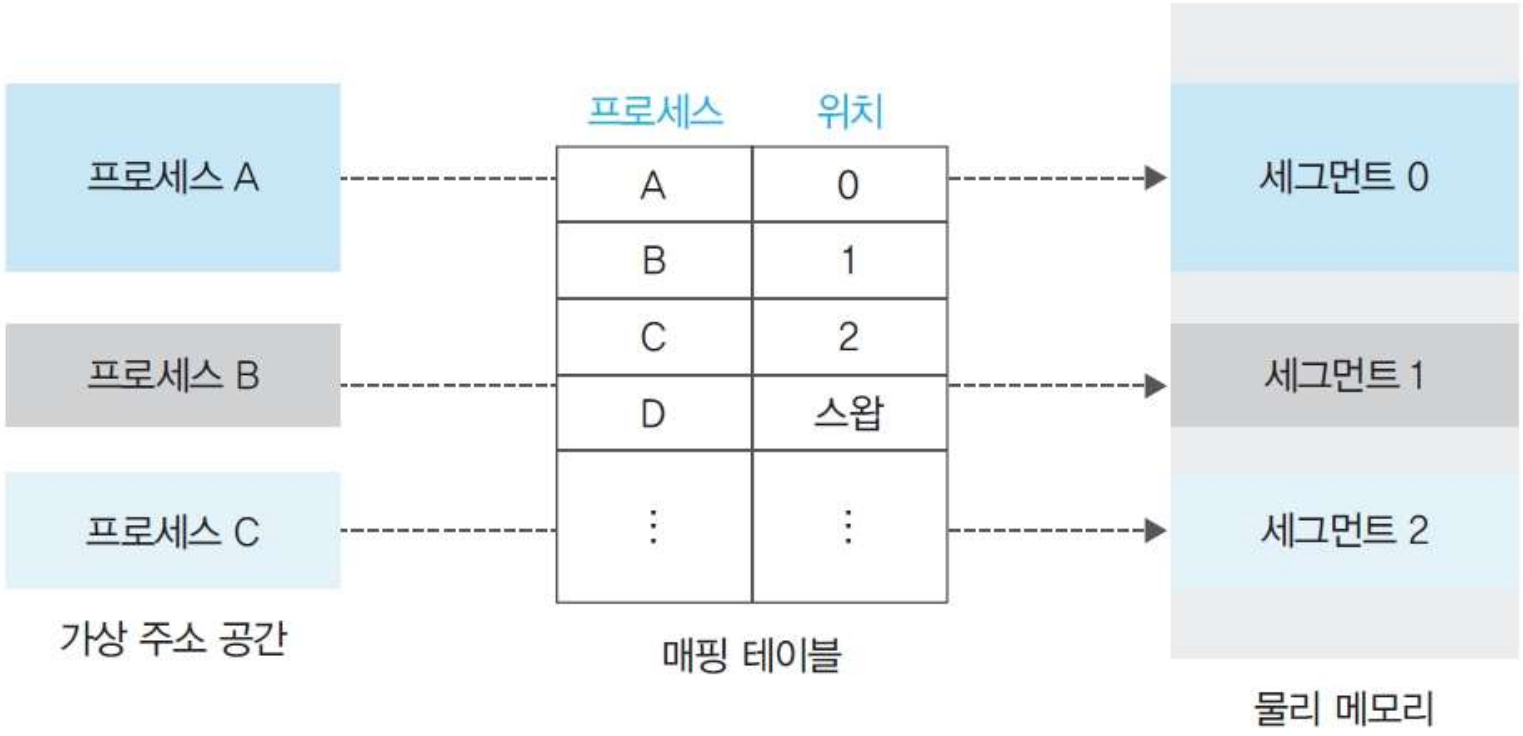


그림 8-4 가상 메모리와 매핑 테이블

1-2 매핑 테이블의 필요성과 역할

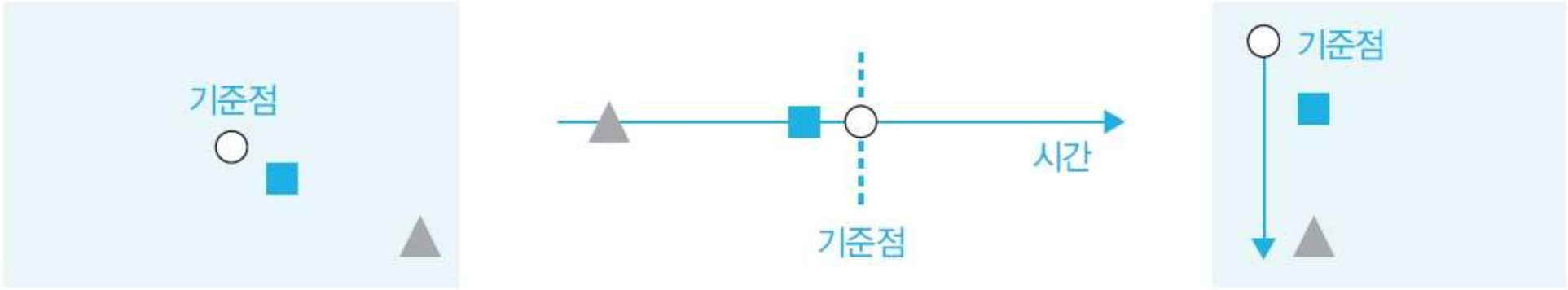
■ 메모리 매핑 테이블

- 가상 메모리 시스템에서 가변 분할 방식을 이용한 메모리 관리 기법은 세그먼테이션, 고정 분할 방식을 이용한 메모리 관리 기법은 페이징
- 페이징 기법에서 사용하는 매핑 테이블을 페이지 매핑 테이블(page mapping table) 또는 페이지 테이블
- 세그먼테이션 기법에서 사용하는 매핑 테이블을 세그먼테이션 매핑 테이블(segmentation mapping table) 또는 세그먼테
- 이션 테이블

1-2 매핑 테이블의 필요성과 역할

■ 지역성

- 지역성(locality)은 기억장치에 접근하는 패턴이 메모리의 특정 영역에 집중되는 성질
- 국부성 또는 집약성
- 공간의 지역성, 시간의 지역성, 순차적 지역성으로 나뉨



(a) 공간의 지역성 (b) 시간의 지역성 (c) 순차적 지역성

그림 8-5 지역성의 유형

1-2 매핑 테이블의 필요성과 역할

■ 지역성

- 공간의 지역성(spatial locality): 현재 위치에서 가까운 데이터에 접근할 확률이 먼 거리에 있는 데이터에 접근할 확률보다 높음을 의미
- 시간의 지역성(temporal locality): 현재를 기준으로 가장 가까운 시간에 접근한 데이터가 더 먼 시간에 접근한 데이터보다 사용될 확률이 높음을 의미
- 순차적 지역성(sequential locality): 작업이 순서대로 진행되는 것을 의미

■ 지역성의 예

- 캐시: 지역성 이론을 사용하는 대표적인 장치
- 페이지 교체 알고리즘(page replacement algorithm)

2-1 페이징 기법의 구현

■ 페이징 기법

- 고정 분할 방식을 이용한 가상 메모리 관리 기법
- 물리 주소 공간을 같은 크기로 나누어 사용
- 가상 주소는 프로세스 입장에서 바라본 메모리 공간으로 항상 0번지부터 시작
- 페이지(page)와 프레임(frame)
  - 가상 주소의 분할된 각 영역을 페이지라고 하며 번호를 매겨 관리
  - 물리 메모리의 각 영역은 가상 주소의 페이지와 구분하기 위해 프레임이라고 함(프레임도 페이지처럼 번호를 매겨 관리)
  - 페이지와 프레임의 크기는 같기 때문에 페이지는 어떤 프레임에도 배치될 수 있음
  - 어떤 페이지가 어떤 프레임에 있는지에 대한 연결(매핑) 정보는 페이지 테이블에 담겨 있음
  - 페이지 테이블에 invalid는 해당 페이지가 스왑 영역에 있다는 의미

2-1 페이징 기법의 구현

■ 페이징 기법

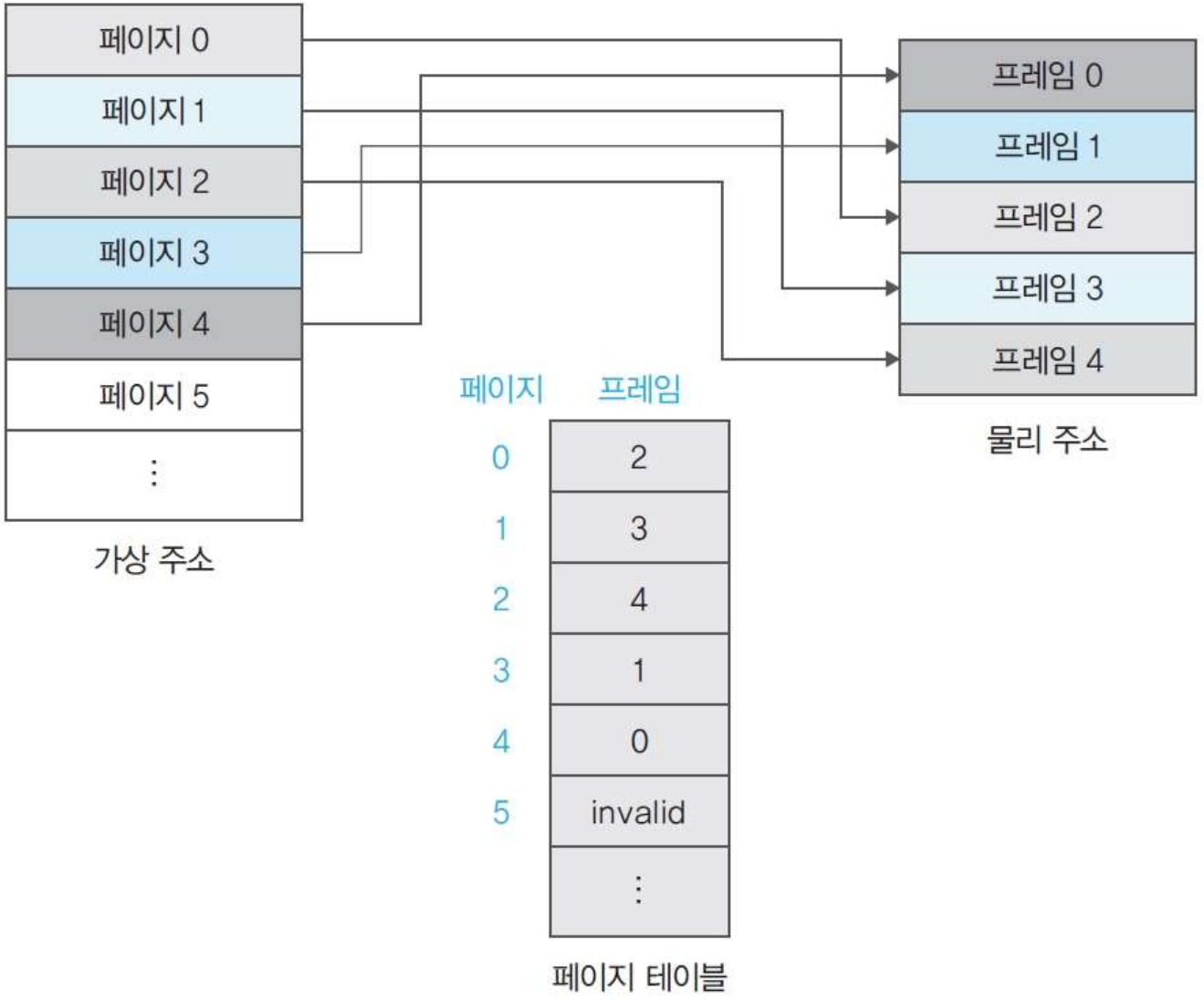


그림 8-6 페이징 기법의 구현

2-2 페이징 기법의 주소 변환

■ 주소 변환 과정

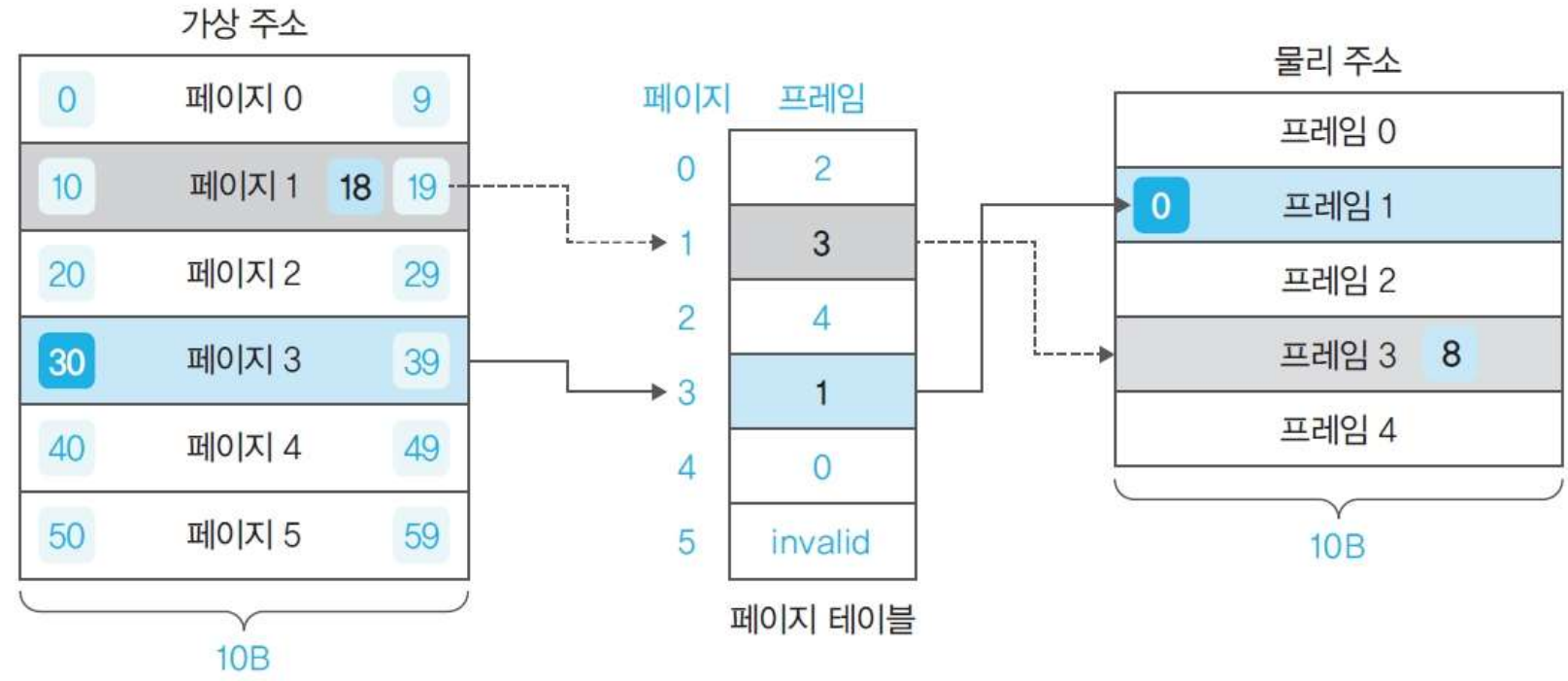


그림 8-7 페이징 기법의 주소 변환

2-2 페이징 기법의 주소 변환

■ 정형화된 주소 변환

- 페이징 기법에서는 가상 주소를 VA=<P, D>로 표현
  - VA : 가상 주소(virtual address)
  - P : 페이지(page)
  - D : 페이지의 처음 위치에서 해당 주소까지의 거리(distance)

- 페이징 기법에서의 주소 변환은 가상 주소 VA=<P, D>를 물리 주소 PA=<F, D>로 변환하는 것
  - PA : 물리 메모리의 주소를 가리키는 용어로 물리 주소 또는 실제 주소
  - F : 프레임(frame)
  - D : 프레임의 처음 위치에서 해당 주소까지의 거리(distance)

2-2 페이징 기법의 주소 변환

■ 페이징 기법의 주소 변환 과정

- VA=<P, D> → PA=<F, D>
- 페이지 테이블을 사용하여 P는 F로 바꾸고 D는 변경 없이 그대로 씀
- D를 변경하지 않는 이유는 페이지와 프레임의 크기를 똑같이 나누었기 때문

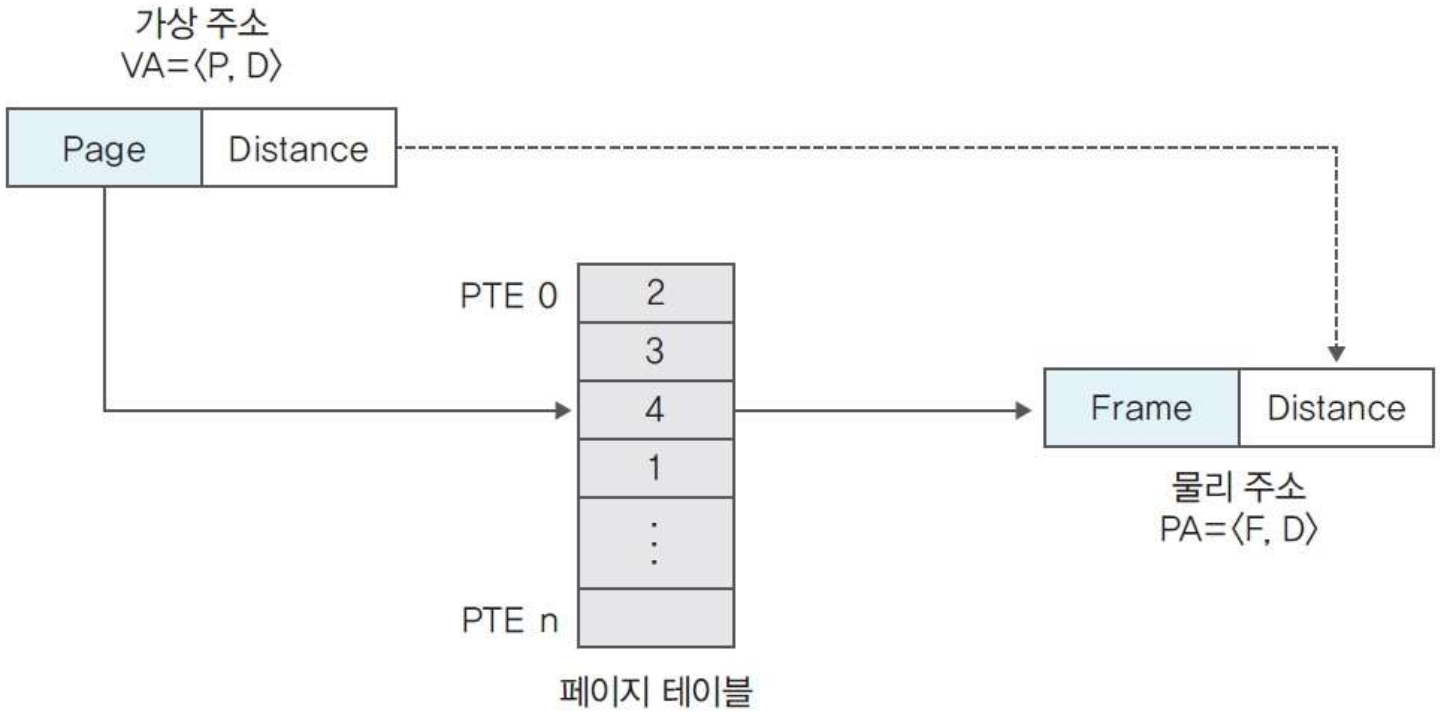


그림 8-8 페이징 기법의 정형화된 주소 변환

2-2 페이징 기법의 주소 변환

■ 페이지의 크기가 다양할 경우 가상 주소를 <P, D>로 변환하는 공식

정의 8-3 가상 주소를 〈P, D〉로 표현하는 공식 (단위: 바이트)

P=나눗셈(가상 주소/한 페이지의 크기)의 몫

D=나눗셈(가상 주소/한 페이지의 크기)의 나머지

- 한 페이지의 크기가 10B인 가상 메모리 시스템에서 가상 주소 32번지
  - P=3(32/10의 몫)
  - D=2(32/10의 나머지)
- 한 페이지의 크기가 512B인 시스템에서 가상 주소 2049번지
  - P=4(2049/512의 몫)
  - D=1(2049/512의 나머지)

2-2 페이징 기법의 주소 변환

■ 16bit CPU의 컴퓨터에서 한 페이지의 크기가 2^{10}B일 때 페이징 시스템

- 한 프로세스가 사용할 수 있는 가상 메모리의 크기는 2^{16}(65,536)B
- 사용자는 0번지부터 65535(2^{16}-1)번까지 가상 주소 공간을 사용 가능
- 가상 주소로 사용할 수 있는 16bit 중 6bit는 페이지 번호로, 10bit는 페이지의 처음 위치에서 해당 주소까지의 거리

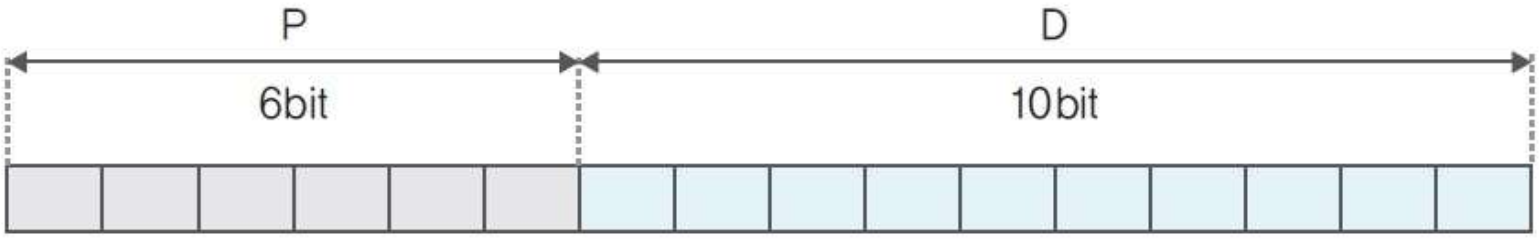


그림 8-9 16bit 가상 주소의 예

2-2 페이징 기법의 주소 변환

■ 16bit CPU의 컴퓨터에서 한 페이지의 크기가 2^{10}B일 때 페이징 시스템

- 전체 페이지의 수는 2^6, 즉 64개이고 페이지 0번부터 63번까지 존재
- 물리 주소도 가상 주소와 같이 1,024B로 나뉨. 프레임은 0부터 31까지만 있음
- 페이지 테이블 엔트리가 0~63으로 총 64개(페이지 테이블의 크기는 물리 주소의 크기가 아니라 프로세스의 크기에 비례)

2-2 페이징 기법의 주소 변환

■ 프로세스가 980번지에 저장된 데이터를 요청했을 때 동적 주소 변환

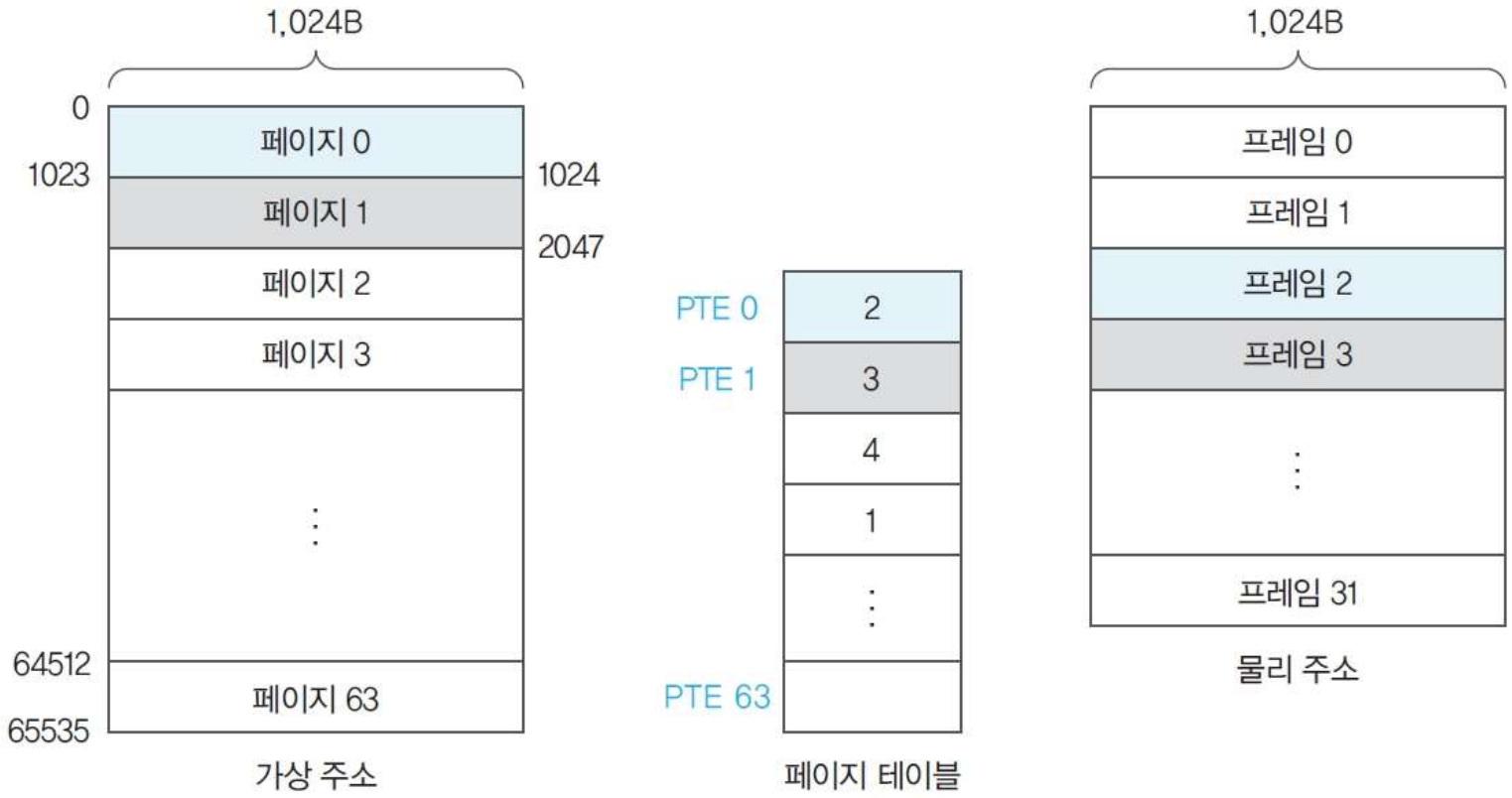


그림 8-10 16bit CPU의 페이징 시스템

2-2 페이징 기법의 주소 변환

■ 프로세스가 980번지에 저장된 데이터를 요청했을 때 동적 주소 변환

- 가상 주소 980번지의 페이지 P, 거리 D를 구함
  - P=0(980/1024의 몫), D=980(980/1024의 나머지) → VA=<0, 980>
- 페이지 테이블로 가서 페이지 0이 프레임 2에 저장되어 있다는 것을 확인
- 물리 메모리의 프레임 2 시작 지점으로부터 980번지 떨어진 곳에 접근하여 데이터를 가져옴

2-3 페이지 테이블 관리

■ 메모리 공유

- 프로세스마다 페이지 테이블 존재함. 프로세스 수가 많아지면 페이지 테이블 크기가 커지고, 프로세스가 실제 사용할 수 있는 메모리 영역이 줄어듬
- 페이지 테이블 크기를 적정하게 유지하는 것이 페이지 테이블 관리의 핵심

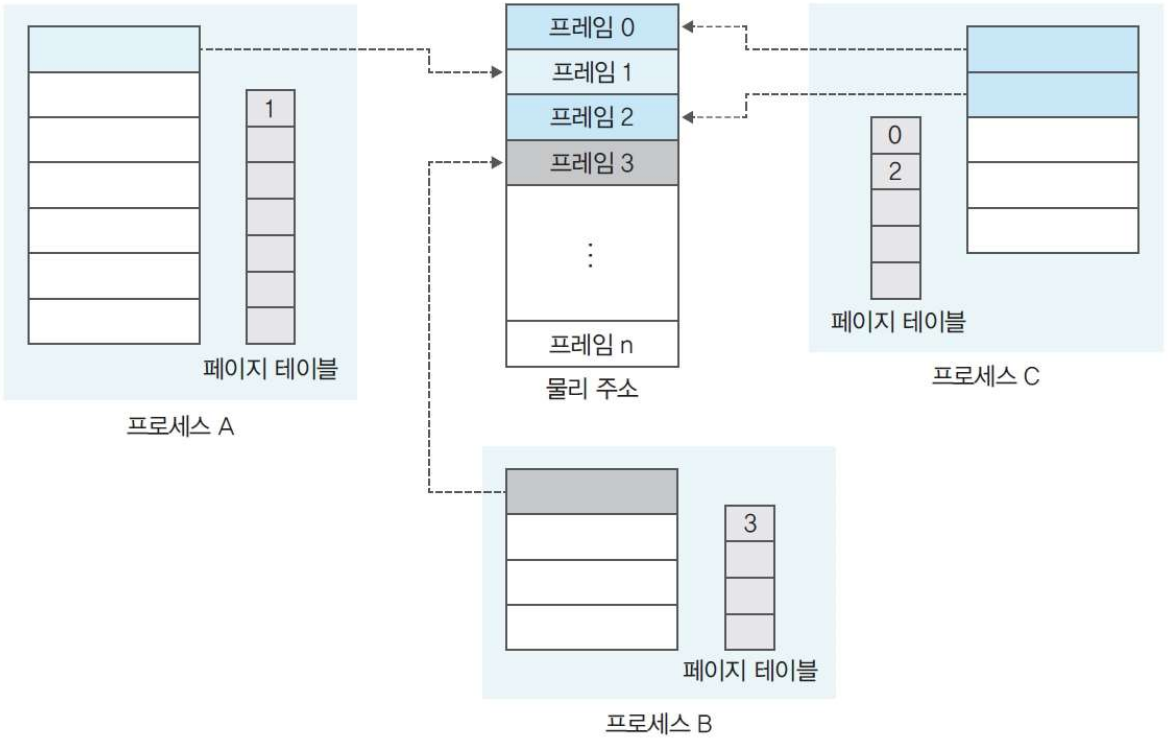


그림 8-11 다수의 프로세스가 있는 페이징 시스템

2-3 페이지 테이블 관리

■ 물리 메모리 내 페이지 테이블의 구조

- 각 페이지 테이블의 시작 주소는 페이지 테이블 기준 레지스터(PTBR; Page Table Base Register)에 보관

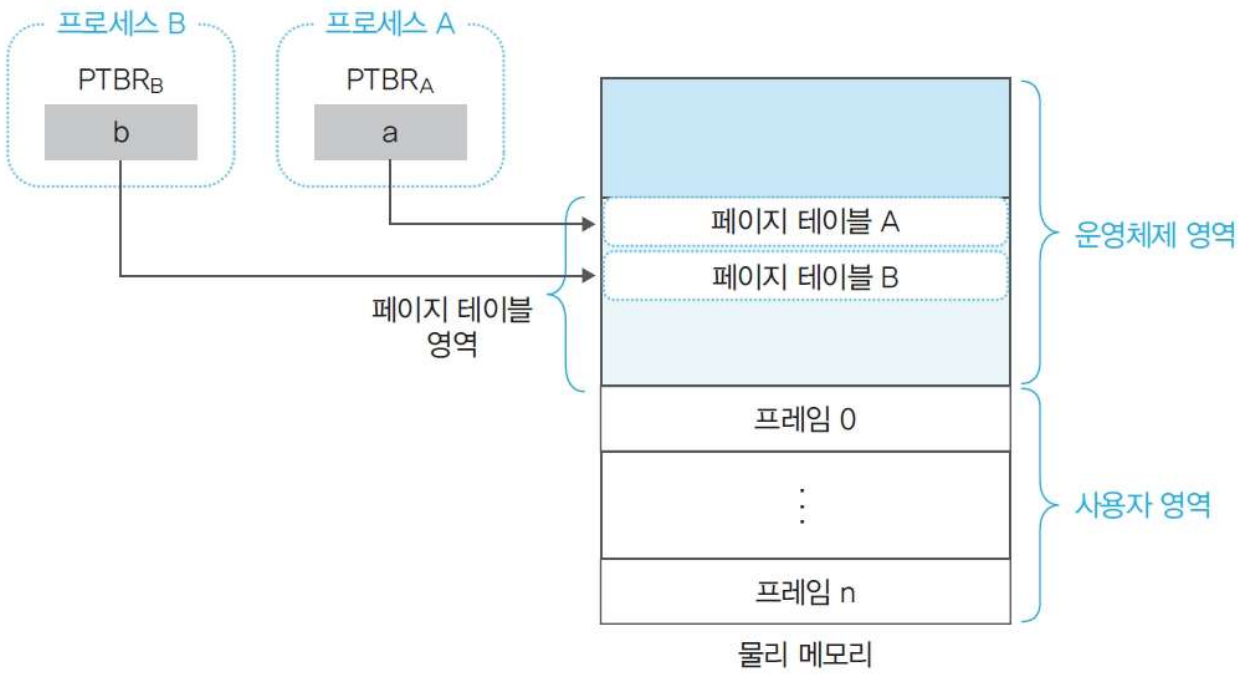


그림 8-12 물리 메모리의 페이지 테이블의 구조

- 물리 메모리의 크기가 작을 때는 페이지 테이블의 일부도 스왑 영역으로 옮겨짐

2-3 페이지 테이블 관리

■ 쓰기 시점 복사

- 프로세스들이 메모리에 있는 페이지를 공유하는 방식: 프레임을 공유해 메모리 절약

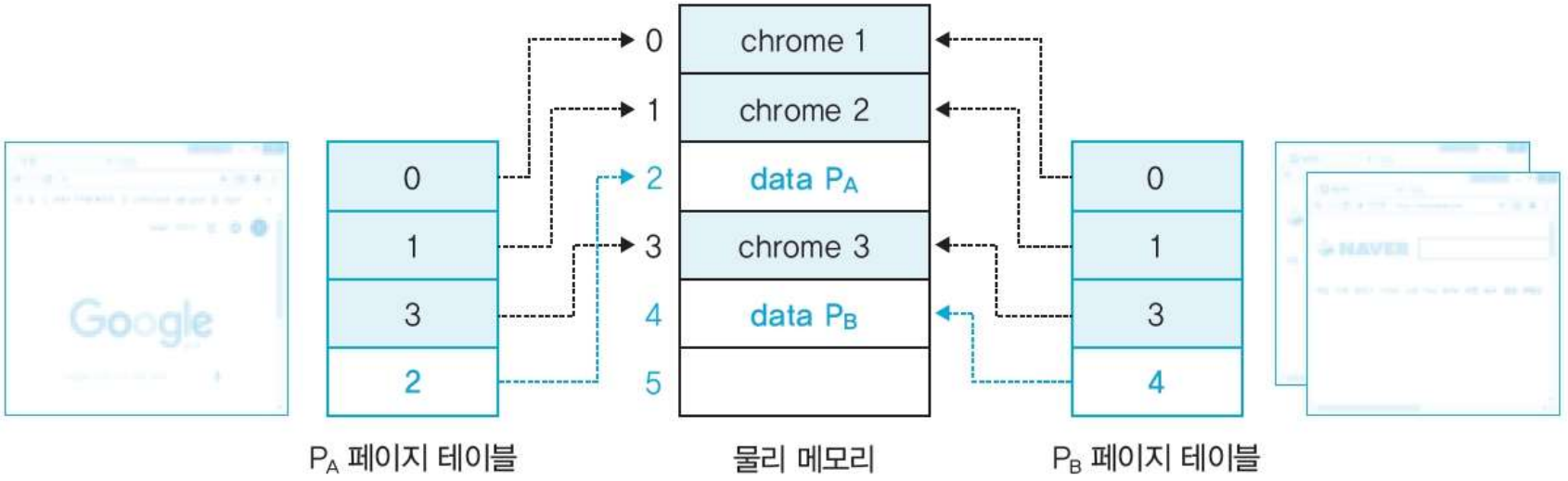


그림 8-13 프레임 공유

2-3 페이지 테이블 관리

■ 쓰기 시점 복사

- 데이터 변화가 있을 때까지 복사를 미루는 방식을 쓰기 시점 복사(copy on write)

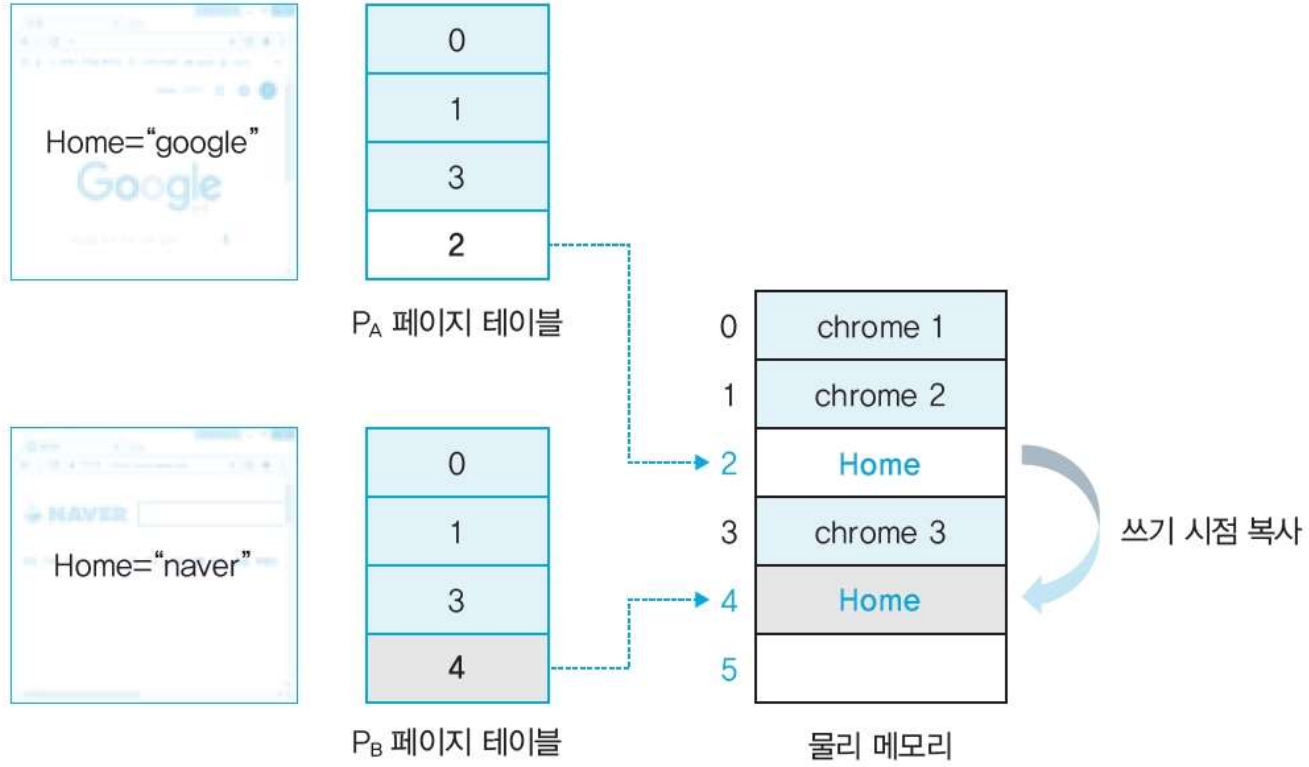


그림 8-15 쓰기 시점 복사

2-3 페이지 테이블 관리

■ 변환 색인 버퍼

- 가상 주소를 물리 주소로 변환하려면 메모리에 두 번 접근해야 함

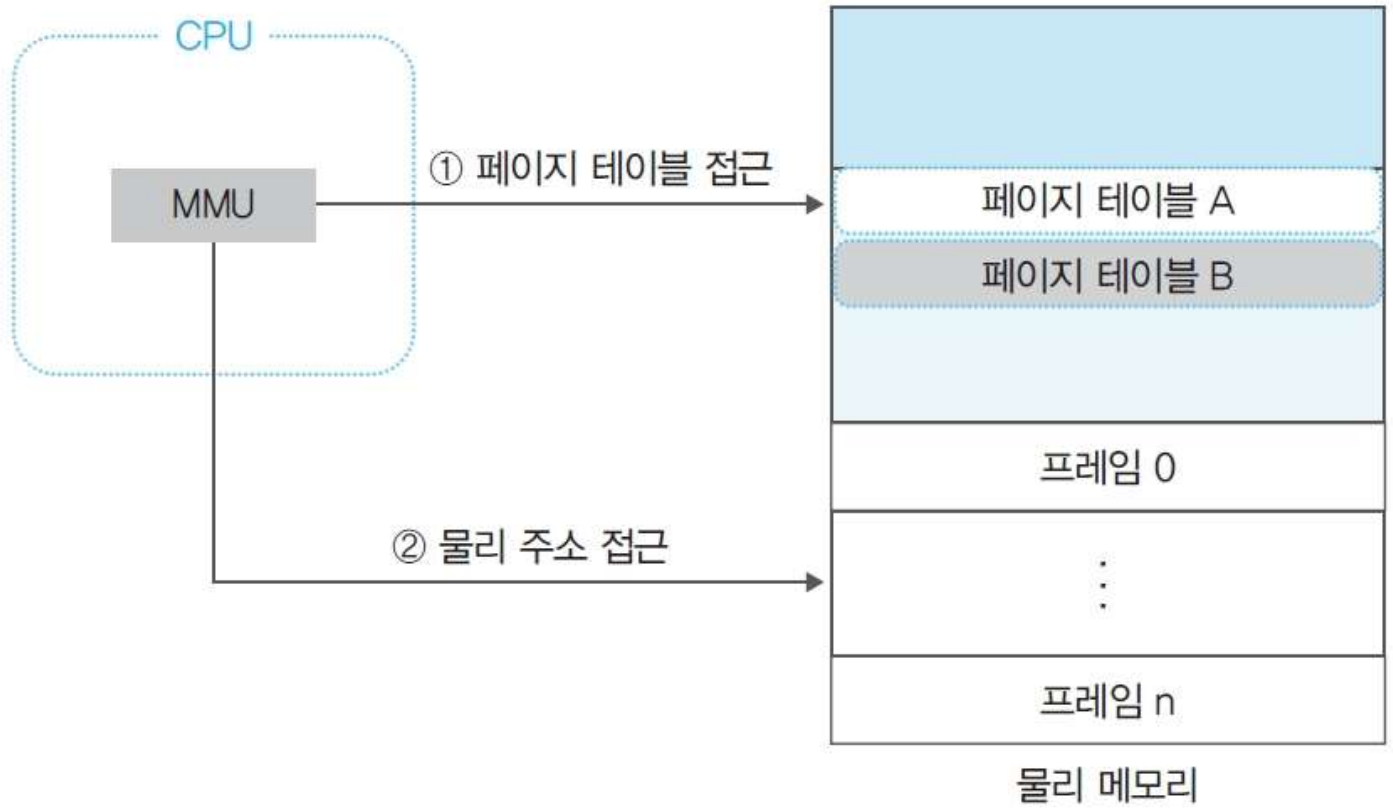


그림 8-16 주소 변환을 위한 두 번의 물리 메모리 접근

2-3 페이지 테이블 관리

■ 변환 색인 버퍼를 이용한 물리 메모리 접근 방식

- 원하는 페이지 번호가 변환 색인 버퍼에 있으면 TLB 히트(TLB hit), 곧바로 물리 주소로 변환
- 원하는 페이지 번호가 없으면 TLB 미스(TLB miss), 메모리에 있는 페이지 테이블을 사용해 프레임 번호로 변환

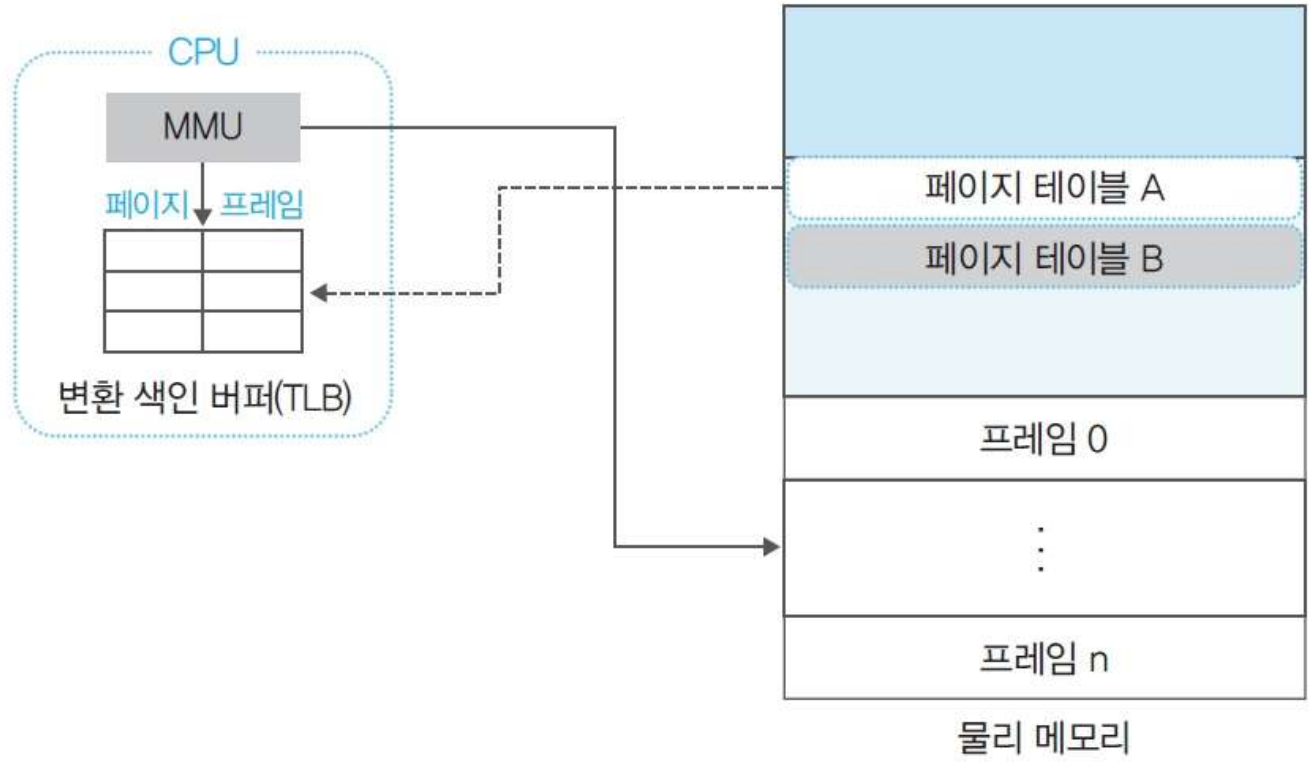


그림 8-17 변환 색인 버퍼를 이용한 물리 메모리 접근

2-3 페이지 테이블 관리

■ 변환 색인 버퍼를 이용한 정형화된 주소 변환

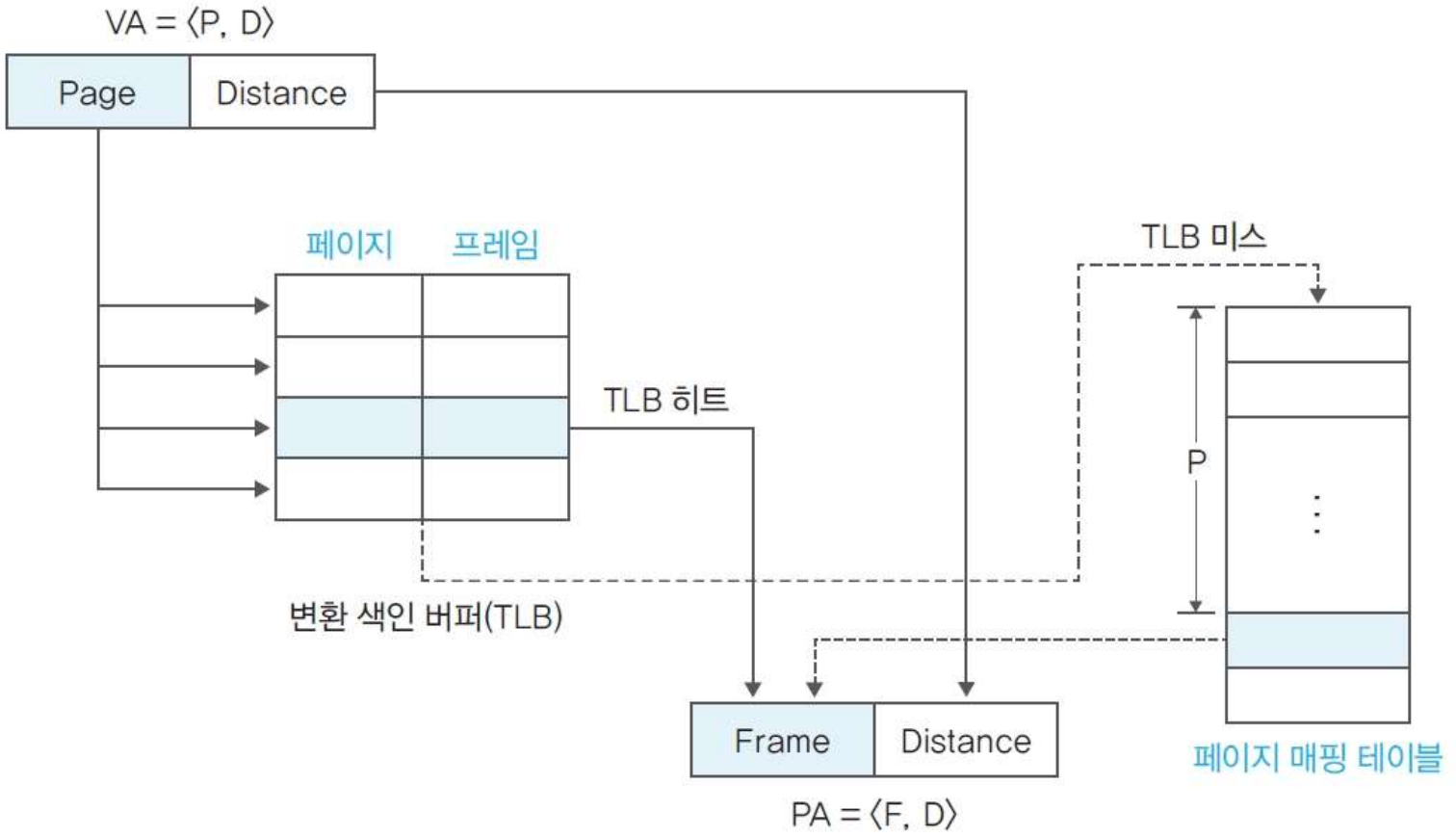


그림 8-18 변환 색인 버퍼를 이용한 정형화된 주소 변환

2-3 페이지 테이블 관리

■ 역 페이지 테이블

- 역 페이지 테이블(invert page table) 방식은 기존 페이징 방식과 반대로 페이지 테이블 구성
- 물리 메모리의 프레임 번호를 기준으로 테이블 구성
- 물리 메모리의 프레임에 어떤 프로세스의 어떤 페이지가 올라와 있는지 표시
- 프로세스 수와 상관없이 테이블이 하나만 존재하는 것이 특징
- 테이블 크기가 작은 것이 장점

2-3 페이지 테이블 관리

■ 역 페이지 테이블의 구성

그림 8-19 역 페이지 테이블의 구성

2-3 페이지 테이블 관리

■ 역 페이지 테이블에서 정형화된 주소 변환

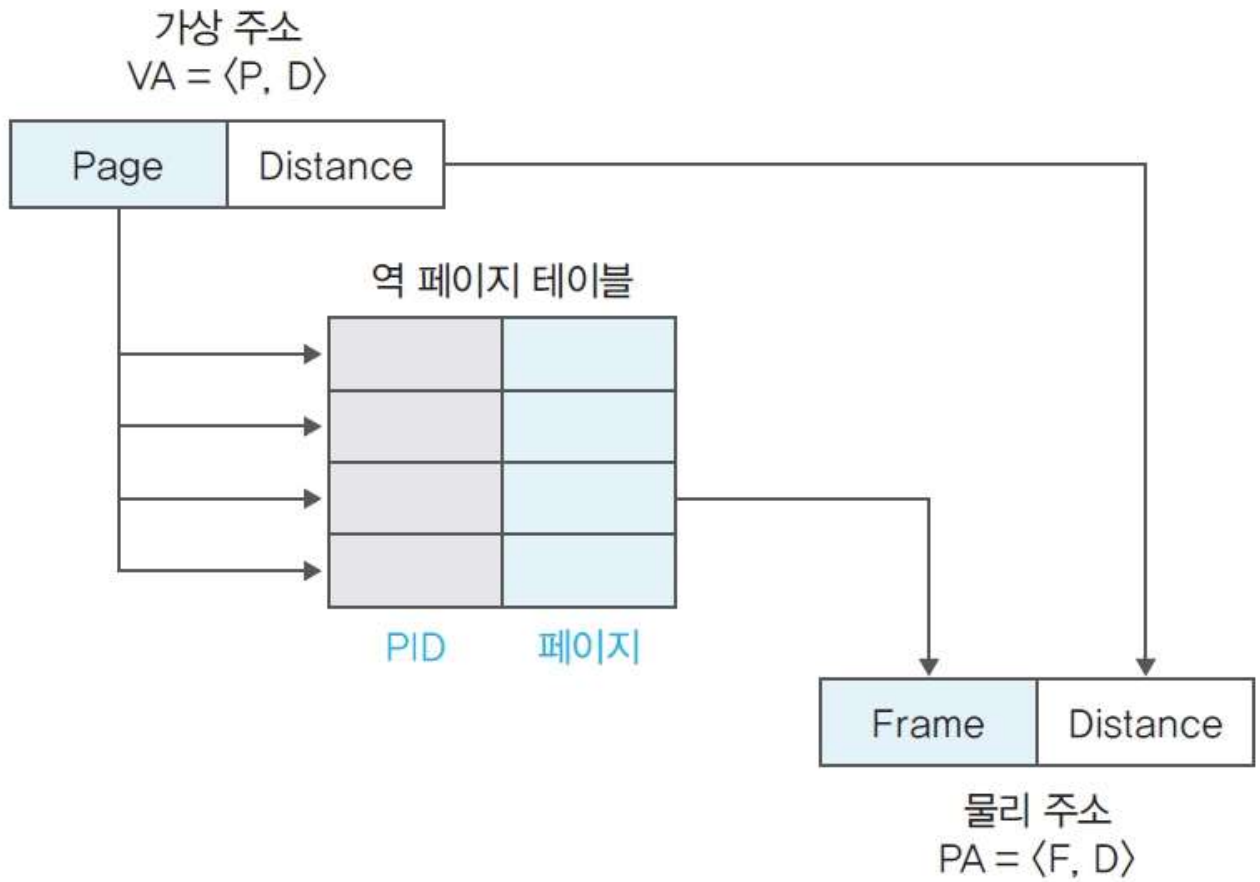


그림 8-20 역 페이지 테이블

2-3 페이지 테이블 관리

■ 다단계 페이지 테이블: 2단계 페이지 테이블

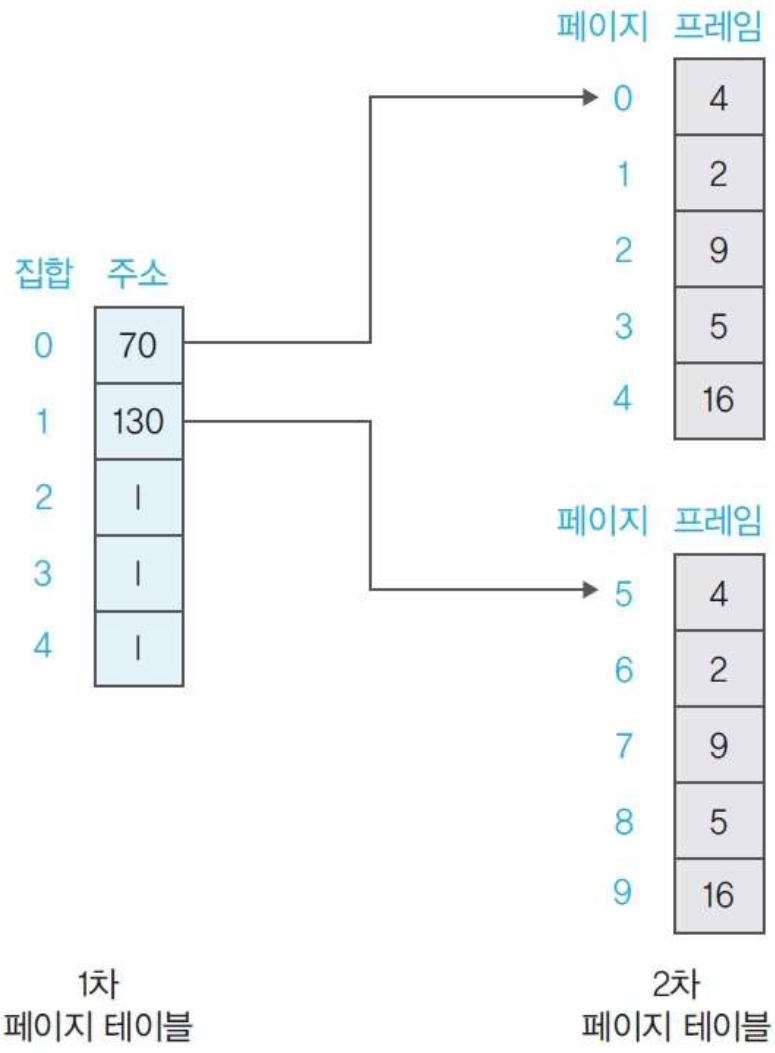


그림 8-21 2단계 페이지 테이블

2-3 페이지 테이블 관리

■ 다단계 페이지 테이블: 2단계 페이지 테이블 방식의 주소 변환

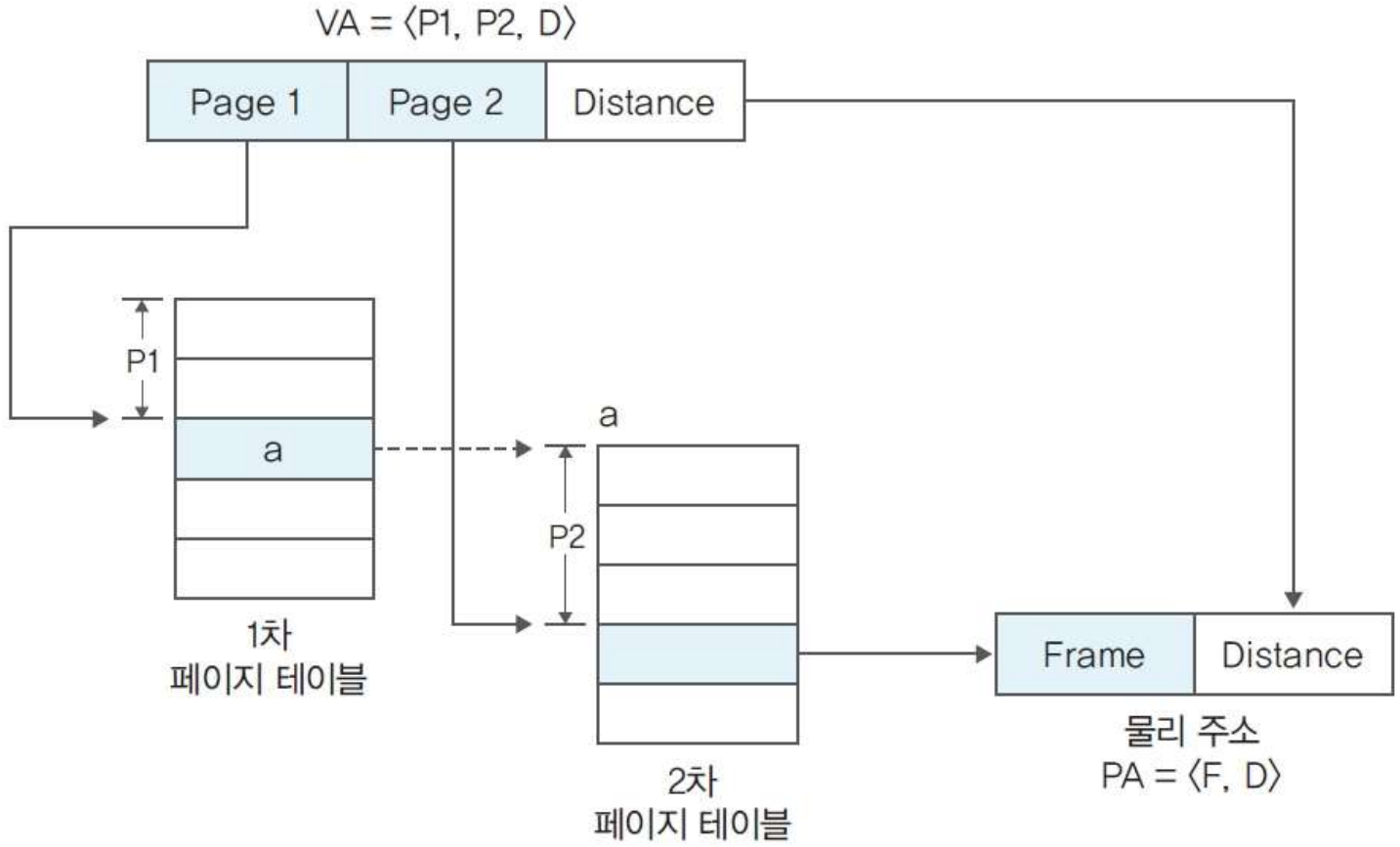


그림 8-22 2단계 페이지 테이블의 주소 변환

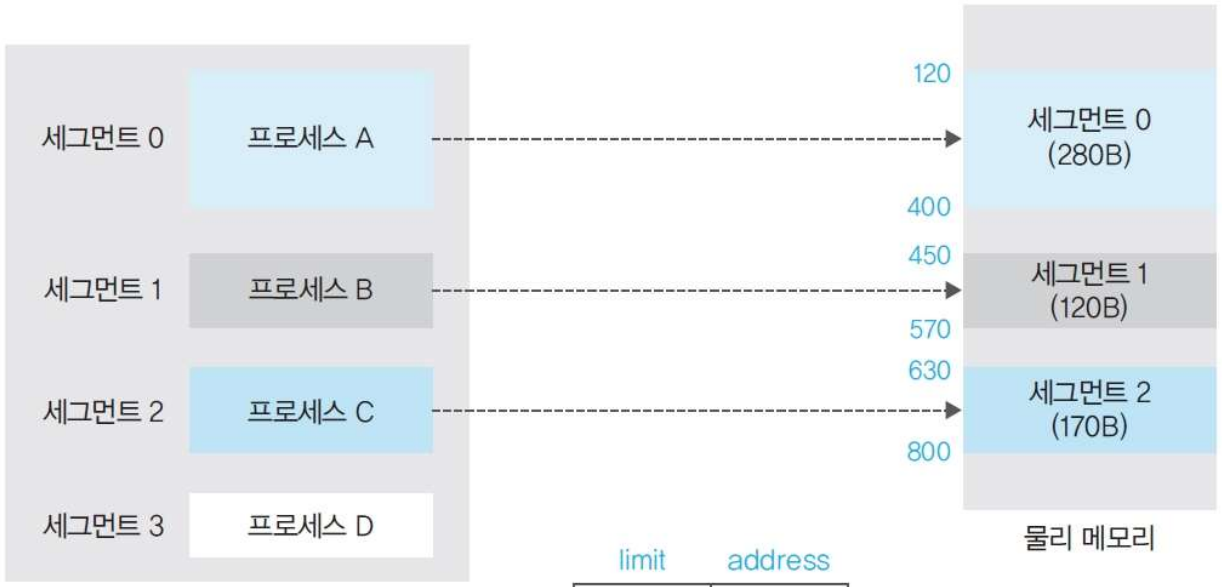
3-1 세그먼테이션 기법의 구현

■ 세그먼테이션 테이블(segmentation table)

- 세그먼트의 크기를 나타내는 limit와 물리 메모리상의 시작 주소를 나타내는 address가 있음
- 각 세그먼트가 자신에게 주어진 메모리 영역을 넘어가면 안 되기 때문에 세그먼트의 크기 정보에는 크기를 뜻하는 size 대신 제한을 뜻하는 limit를 사용
- 세그먼테이션 기법에서도 물리 메모리가 부족할 때 스왑 영역을 사용
- 크기가 100B인 프로세스 D(세그먼트 3)가 스왑 영역에 있고, 세그먼테이션 테이블의 address에 I(invalid)라고 표시되어 있음

3-1 세그먼테이션 기법의 구현

■ 세그먼테이션 테이블(segmentation table)



	limit	address
0	280	120
1	120	450
2	170	630
3	100	I
	⋮	⋮

세그먼테이션 테이블

그림 8-23 세그먼테이션 기법

3-2 세그먼테이션 기법의 주소 변환

■ 프로세스 A의 32번에 접근할 때 주소 변환 과정

① 가상 주소를 구함(VA =<0, 32>)

② 세그먼테이션 테이블에서 세그먼트 0의 시작 주소를 알아낸 후 시작 주소 120에 거리 32를 더하여 물리 주소 152번지를 구함(이때 메모리 관리자는 거리가 세그먼트의 크기보다 큰지 점검하고 만약 크다면 메모리 오류를 출력하고 해당 프로세스를 강제종료하며, 크지 않다면 물리 주소를 구함)

③ 물리 주소 152번에 접근하여 원하는 데이터를 읽거나 씀

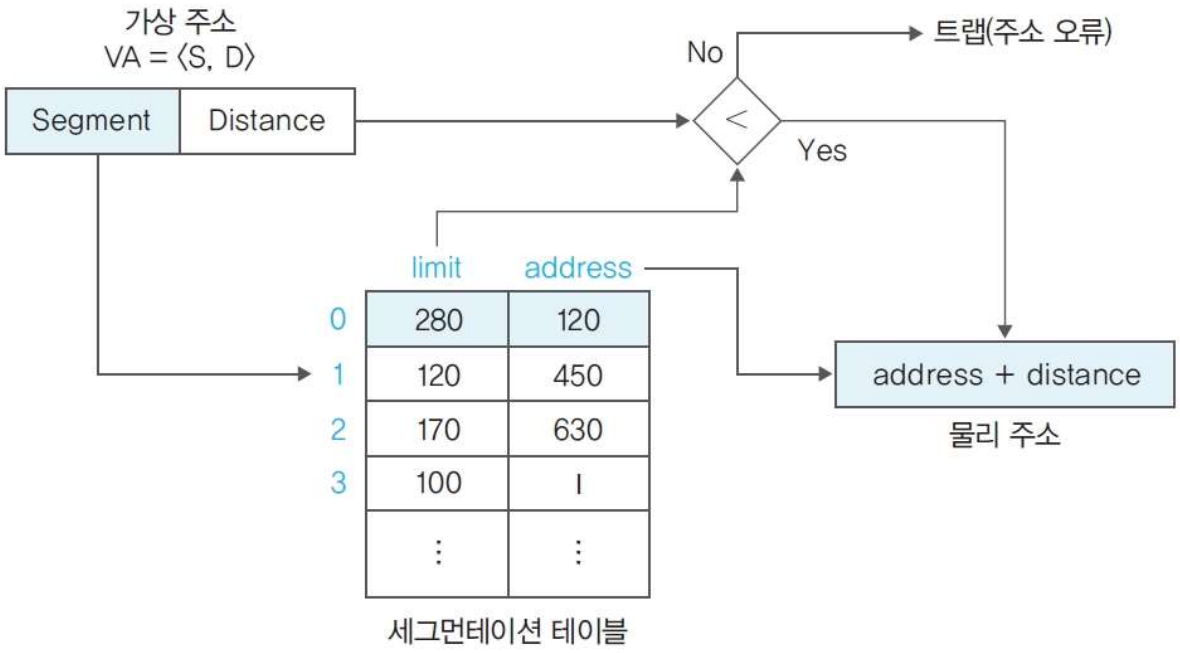


그림 8-24 세그먼테이션 기법의 주소 변환

4-1 캐시 직접 매핑

■ 캐시 직접 매핑

- 메모리의 블록이 캐시로 올라올 때 항상 같은 위치에 올라옴
- 캐시는 메모리의 어떤 블록에서 올라온 페이지인지만 확인
- CPU가 메모리에 접근하려는 주소 <P, D>는 <tag, bd, D>로 바꿀 수 있고, 원하는 데이터를 캐시에 얻기 위해 <tag, D>를 사용
  - 태그 : 캐시에 블록 번호를 명시하는 것
  - bd(block distance): 블록에서의 거리를 의미

4-1 캐시 직접 매핑

■ 캐시 직접 매핑의 예

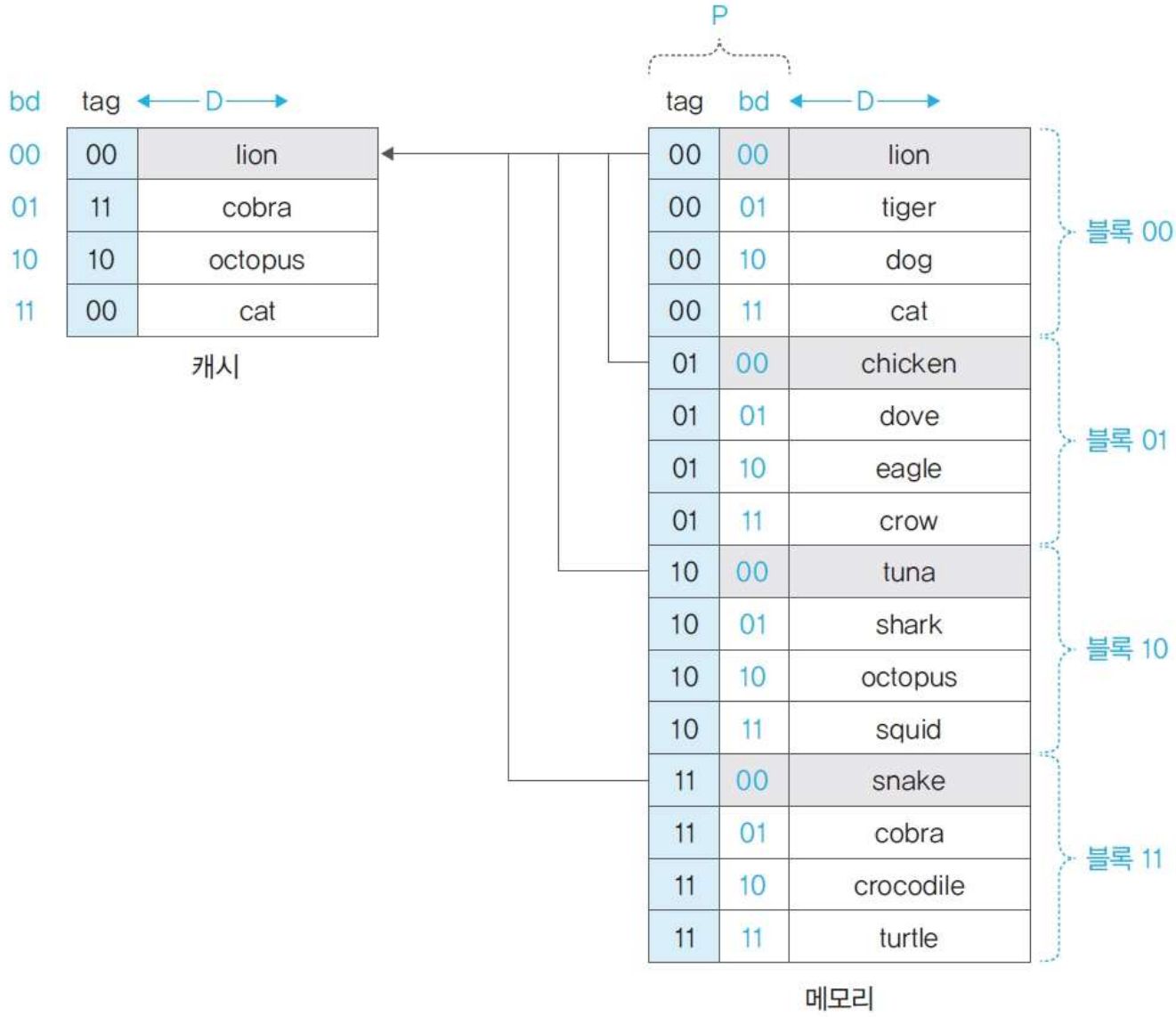


그림 8-25 캐시 직접 매핑

4-1 캐시 직접 매핑

■ 캐시 직접 매핑의 예

- 현재 캐시의 00 위치에 lion, chicken, tuna, snake 중 어떤 데이터가 올라왔는지 확인하기 위해 캐시에서는 태그를 유지
- 태그는 메모리 주소의 앞 2bit에 해당하는 값으로, 어떤 블록에서 올라온 데이터인지를 나타냄
- CPU가 메모리 1101의 cobra가 필요로 하는 경우 CPU는 캐시의 01 위치(주소의 뒷자리)에 가서 태그가 11인지 확인하는데 태그가 11이므로 캐시 히트
- 0111의 crow가 필요하다면 먼저 캐시의 11 위치로 가서 태그가 01인지 확인하는데 캐시의 11 위치 태그가 현재 00이므로 캐시 미스 발생, CPU는 메모리의 0111로 가서 crow를 가져옴

4-2 캐시 연관 매핑

■ 캐시 연관 매핑

- 메모리가 캐시의 어느 위치에도 자유롭게 올라갈 수 있으므로 캐시가 메모리의 주소를 전부 가지고 있음
- CPU가 특정 주소를 필요로 할 때 캐시에서 검색하여 찾는 경우는 캐시 히트, 찾지 못하면 캐시 미스가 발생하여 메모리에서 원하는 데이터를 가져옴
- 장점: 캐시 메모리를 자유롭게 사용할 수 있음
- 단점: 캐시 히트인지, 캐시 미스인지 확인하기 위해 캐시의 모든 주소를 검색해야 함

4-2 캐시 연관 매핑

■ 캐시 연관 매핑

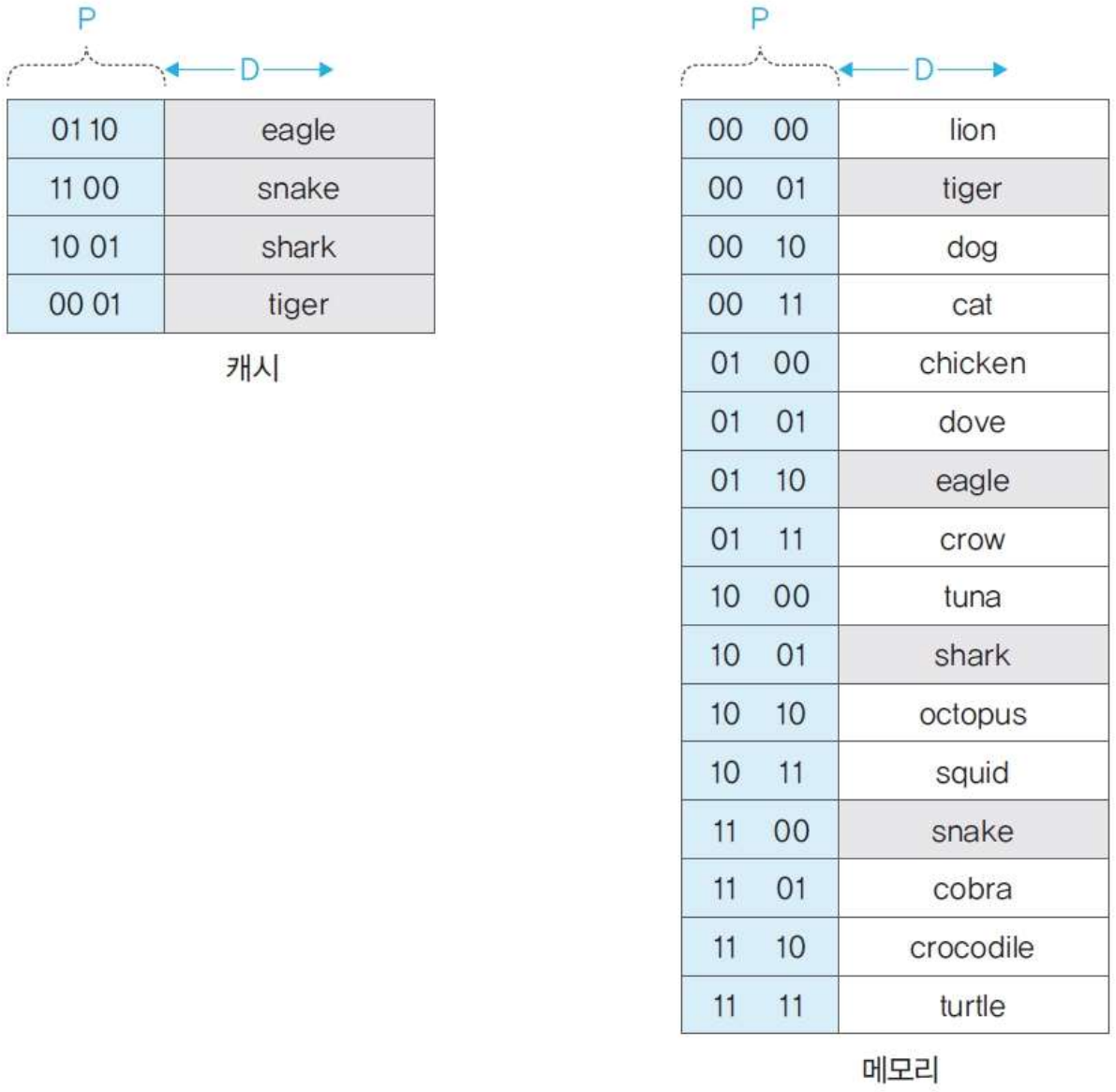


그림 8-26 캐시 연관 매핑

4-3 캐시 집합-연관 매핑

■ 캐시 집합-연관 매핑

- 캐시를 K개의 집합으로 나누고 각 집합에 캐시 직접 매핑 사용
- 직접 매핑을 하는 캐시 메모리를 K개로 나눔으로써 같은 끝자리를 가진 캐시 메모리도 K개가 되기 때문에 직접 매핑의 자리다툼 문제가 완화됨
- 집합 내에서 직접 매핑을 사용하기 때문에 바로 캐시 히트 여부를 알 수 있음

■ 캐시 집합-연관 매핑의 예

- 캐시를 2개의 집합(K=2)으로 나누고 하나의 집합에 2개의 페이지를 사용
- 주소 P의 4bit 중 마지막 1bit만 bd를 나타내고 태그는 앞의 3bit
- 주소의 끝이 0인 페이지는 2개의 집합 중 비어 있는 첫 번째 위치에 들어가고, 끝이 1인 페이지는 2개의 집합 중 비어 있는 마지막 위치에 들어감

4-3 캐시 집합-연관 매핑

■ 캐시 집합-연관 매핑의 예

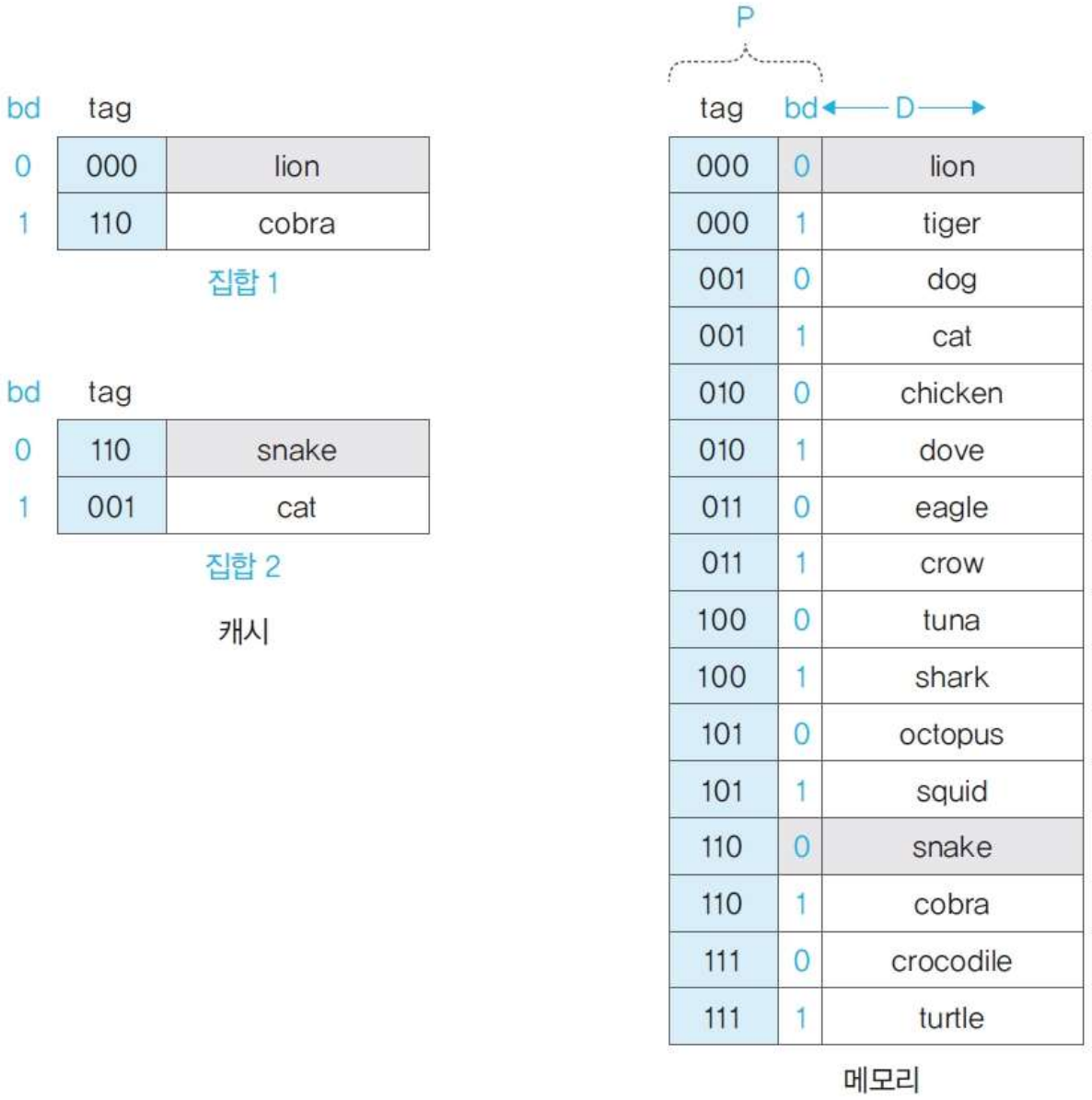


그림 8-27 캐시 집합-연관 매핑