가상 메모리 구성

11th Week

Kim, Eui-Jik

Contents

- 소개
- 가상 메모리: 기본 개념
- 블록 맵핑
- 페이징
- 세그먼테이션

소개

- 메모리 관리 기술은 궁극적으로 제한된 메모리 공간과 경쟁
- 가상 메모리
  - 기본 아이디어: 메모리가 더 많이 존재하는 것처럼 보이게 하는 것
  - 주소 변환
  - 불연속 할당: 페이징과 세그먼테이션

[그림 10-1] 메모리 구성의 진화

가상 메모리: 기본 개념

- 가상 메모리 시스템에서의 두 가지 주소 유형
  - 가상 주소 (virtual address)
    - 프로세스에서 참조하는 주소
  - 물리(실제) 주소 (physical(real) address)
    - 메인 메모리에 있는 사용 가능한 주소

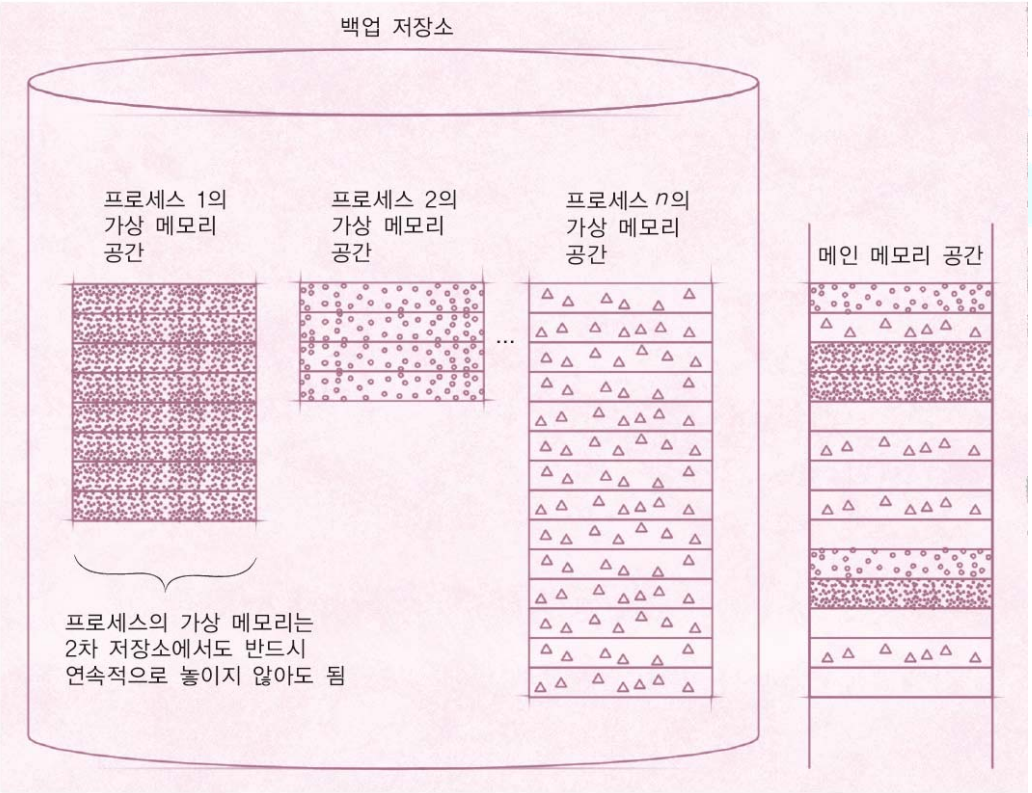
- 메모리 관리 장치 (MMU, Memory Management Unit)
  - 가상주소를 실제 주소로 빠르게 맵핑

가상 메모리: 기본 개념

- 가상 주소 공간 V
  - 프로세스가 참조할 수 있는 가상 주소의 범위
  - |V| = V에 있는 주소의 수
- 실제 주소 공간 R
  - 특정 컴퓨터의 실제 주소 중 사용 가능한 범위
  - |R| = R에 있는 주소의 수
- |V| >> |R|
  - 가상 주소 공간이 실제 주소 공간보다 훨씬 큰 것이 일반적인 현상

가상 메모리: 기본 개념

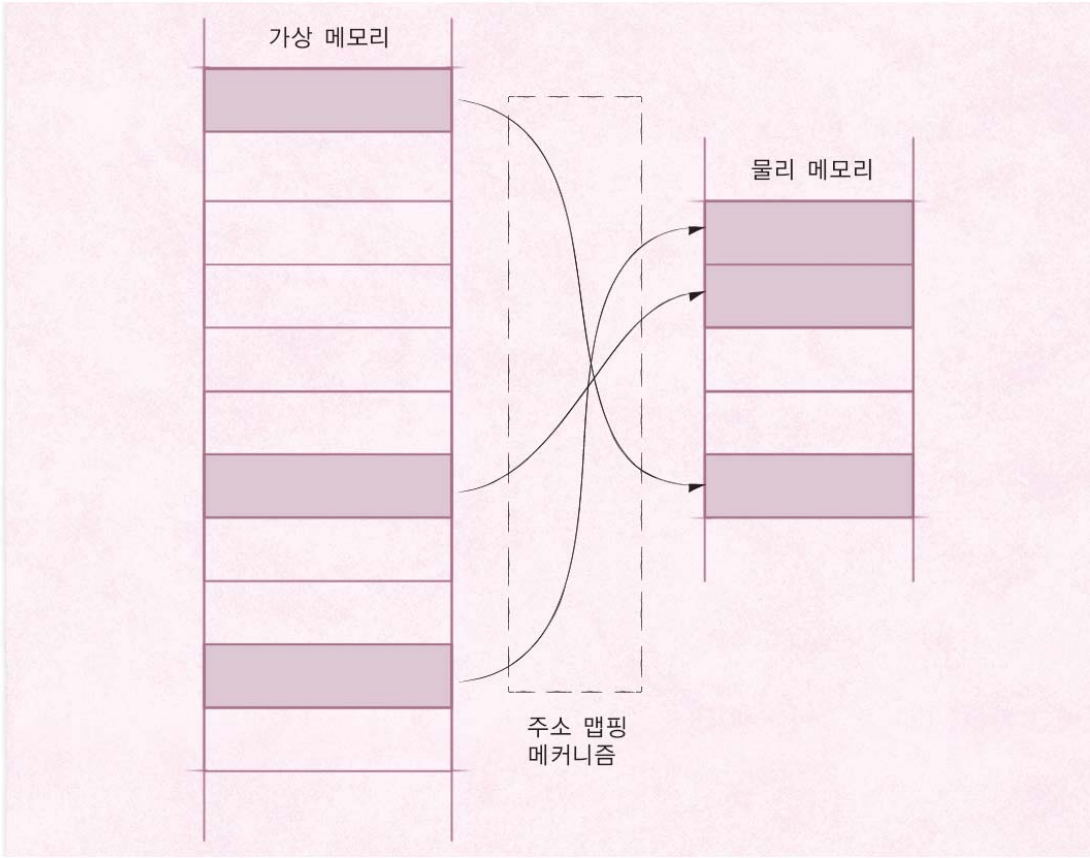
- 프로세스의 실행
  - 프로세스의 코드와 데이터를 2차 저장소에서 메인 메모리로 로드
  - 실제로 이 중 작은 부분만 메인 메모리에 있으면 됨



[그림 10-3] 메모리와 2차 저장소에 존재하는 주소 공간 부분

가상 메모리: 기본 개념

- 가상 주소의 물리 주소 맵핑
  - 프로세스는 가상 주소에만 접근하지만 메인 메모리에서 실행해야 함



[그림 10-4] 가상 주소를 실제 주소에 맵핑

블록 맵핑

- 주소 변환 맵
  - 프로세스의 가상 주소 공간 V에서 어느 영역이 현재 메인 메모리에 있고, 어디에 위치하는지 가리킴
  - 맵핑 정보는 메인 메모리의 작은 부분을 차지하는 것이 필요
    - 해결책: 블록으로 묶는 방법

- 페이지(page)
  - 블록이 고정 크기
  - 페이징(paging): 페이지 관련 가상 메모리 구성

- 세그먼트(segment)
  - 블록이 다른 크기
  - 세그먼테이션(segmentation): 세그먼트 관련 가상 메모리 구성

블록 맵핑

- 블록 사상
  - 블록 단위로 처리
    - 사상이 Byte, Word 단위로 이루어질 경우 주소 사상 테이블(Address Mapping Table) 유지를 위한 정보량이 커짐
    - 사상 정보의 양(테이블)을 줄이기 위해 주소를 블록 단위로 처리

      - 블록은 가상 메모리의 분할 단위로, 블록의 크기가 일정하면 블록을 페이지라 하며 블록화 방법을 페이징 기법이라 함

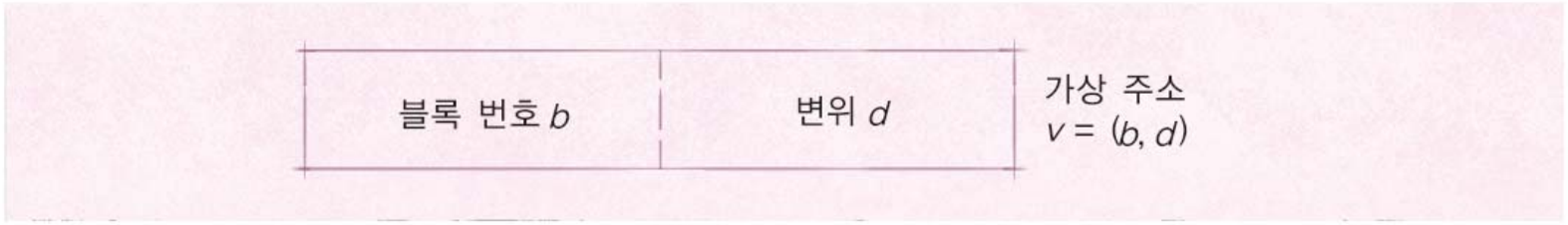
      - 블록의 크기가 다를 경우 세그먼트라 하며 세그먼테이션 기법이라 함

      - 시스템이 블록의 위치만을 유지하고 추적하므로 효율적인 관리 가능

      - 블록의 평균 크기가 클수록 주소사상의 정보의 양(테이블 크기)은 적어지나, 내부 단편화로 인한 2차 기억장치와 메인 메모리 주소사상 시간이 더 요구될 수 있음

블록 맵핑

- 블록 맵핑
  - 주소를 순서 쌍으로 표현
  - 참조하기 위한 항목이 있는 블록과 블록의 시작 위치에서부터 항목의 변위(오프셋)를 명시



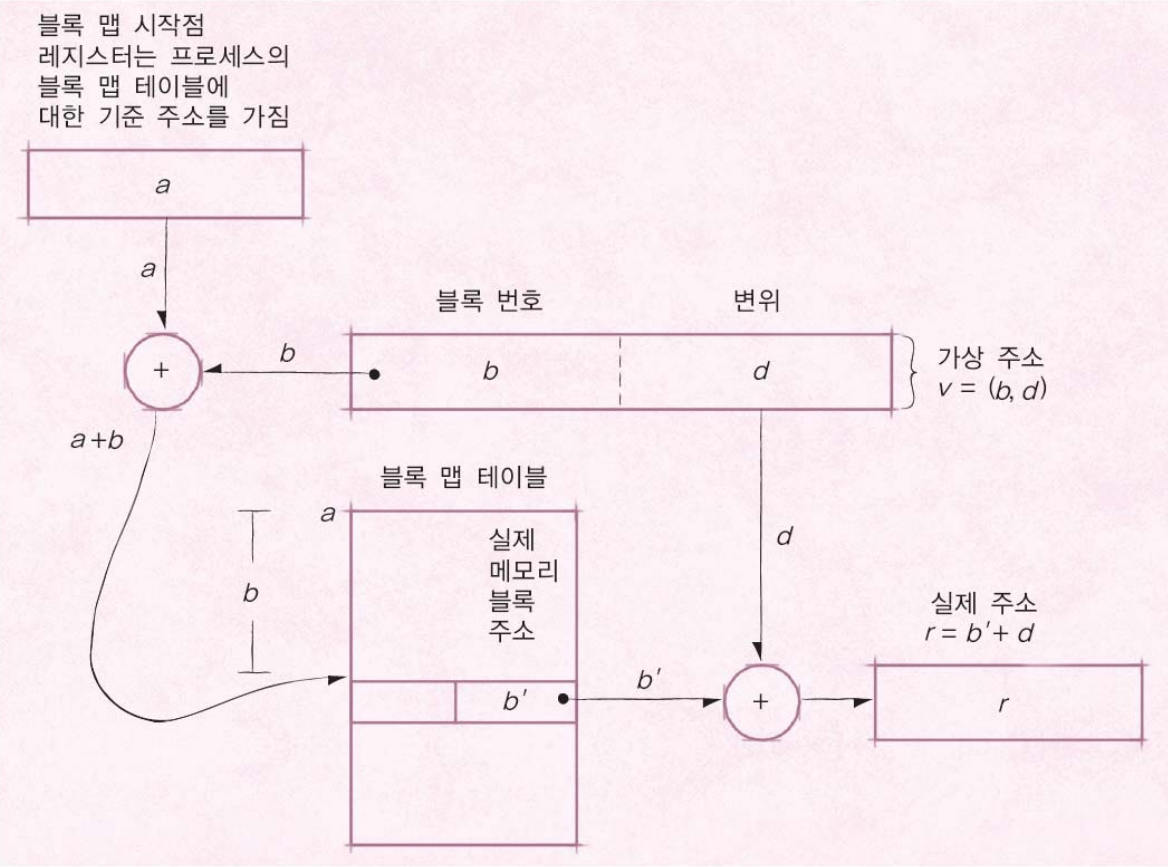
[그림 10-6] 블록 맵핑 시스템에서 가상 주소 형식

블록 맵핑

- 블록 맵핑을 사용한 가상 주소 변환(가상 메모리 주소 v = (b, d))
  - 블록 맵 시작점 레지스터
    - 새 프로세스의 블록 맵 테이블이 위치하는 메인 메모리 주소에 대응하는 실제 주소 a를 저장하는 고속 특수 목적 레지스터
  - 블록 번호 b를 프로세스의 블록 맵 테이블의 기준 주소 a에 더해서 블록 맵 테이블에 있는 블록 b의 엔트리에 대한 실제 주소를 얻어냄
    - b′ : 블록 b의 시작점인 주소
  - 변위 d를 블록 시작 주소 b′에 더해 원하는 실제 주소 r을 얻어냄
    - r = b′ + d

블록 맵핑

- 블록 맵핑을 사용한 가상 주소 변환(가상 메모리 주소 v = (b, d))



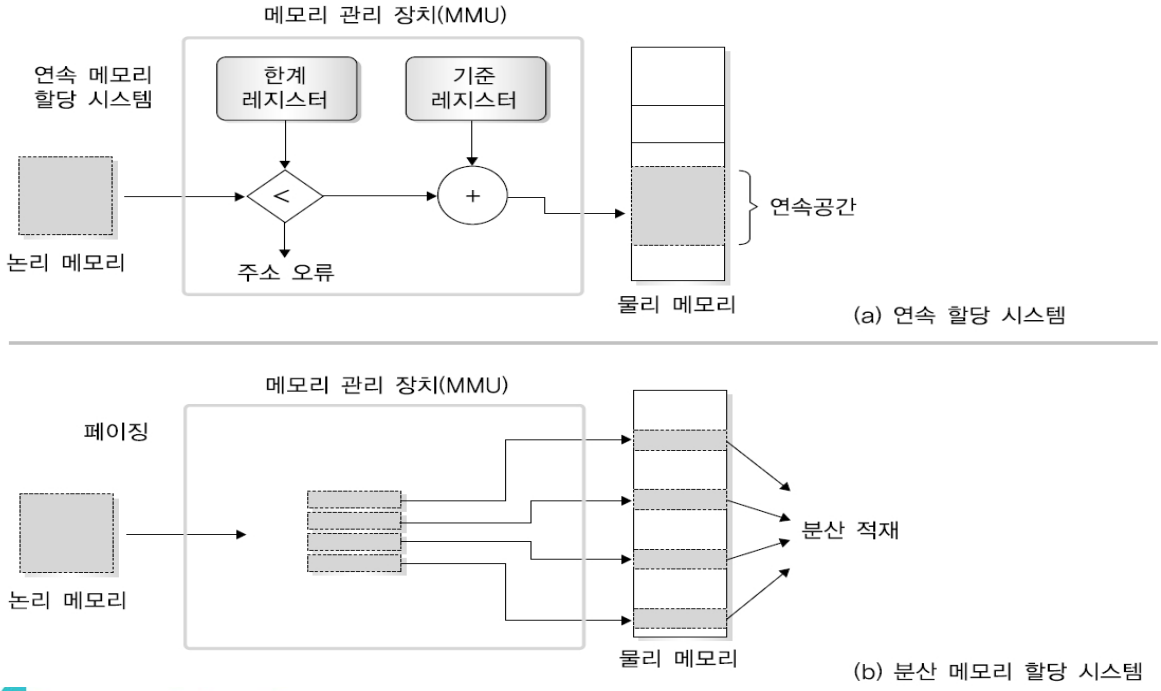
[그림 10-7] 블록 맵핑을 사용한 가상 주소 변환

[참고] 페이징

- 페이징 기법
  - 처리할 작업을 동일한 크기의 페이지로 나누어 처리함
    - 실제 메모리를 프레임(페이지 프레임)이라 불리는 고정 크기 블록으로 나누고, 각 프로세스도 페이지라 불리는 동일한 크기의 작고 고정된 크기의 영역으로 분할
    - 연속 할당과 분산 메모리 할당 시스템의 비교

      - 연속 메모리 할당 시스템은 메모리 관리 장치(MMU)를 이용, 프로세스의 논리 메모리를 연속된 물리적 공간(메모리)에 저장

      - 페이징 시스템은 페이지 4개로 나누어져 물리적 공간(메모리)의 임의의 장소에 각각 저장됨



연속 할당과 분산 메모리 할당 시스템

페이징

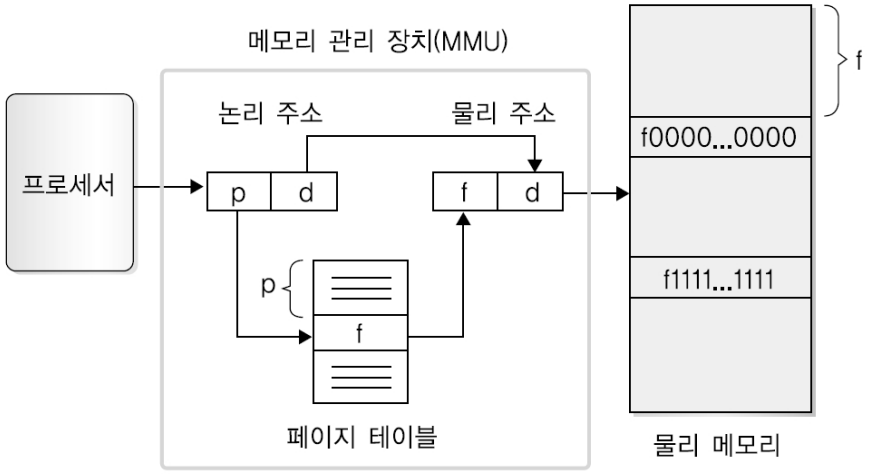
- 메모리 시스템에서 작업 수행을 위해 다음이 준비되어야 함
  - 프로그램에 소요되는 페이지를 결정하여 페이지 번호를 부여
  - 프로그램을 적재하도록 메모리의 빈 프레임을 조사하여 위치 파악
  - 프로그램의 페이지를 빈 페이지 프레임에 적재하도록 준비함
- 장•단점
  - 빈 페이지 프레임이 어떤 작업, 즉 프로세스의 어떤 페이지에도 사용될 수 있음
    - 효율적인 메모리 사용, 페이지 프레임 간 외부 단편화가 발생하지 않음
  - 한 작업의 페이지가 메모리의 여러 위치에 분산 적재됨
    - 페이지들의 위치 정보 등 페이지 관리가 복잡해지므로 운영체제의 부담이 커짐
  - 프레임 단위로 할당되므로 내부 단편화는 발생됨
    - 어떤 프로세스의 메모리 요구가 페이지 범위 내에 맞지 않으면, 할당된 마지막 프레임은 완전히 가득 차지 않을 수 있음

※ 단순한 내부 단편화 현상만 생각하면 크기가 작은 페이지가 바람직함

※ 페이지 테이블 유지가 부담이 될 수 있으나, 페이지 크기를 증가시킴으로써 감소시킬 수 있음

[참고] 페이징

- 페이징 시스템 하드웨어
  - 하드웨어 구조
    - 논리 페이지의 물리 메모리(페이지)프레임 적재과정을 수행하는 페이지 시스템 하드웨어 구조

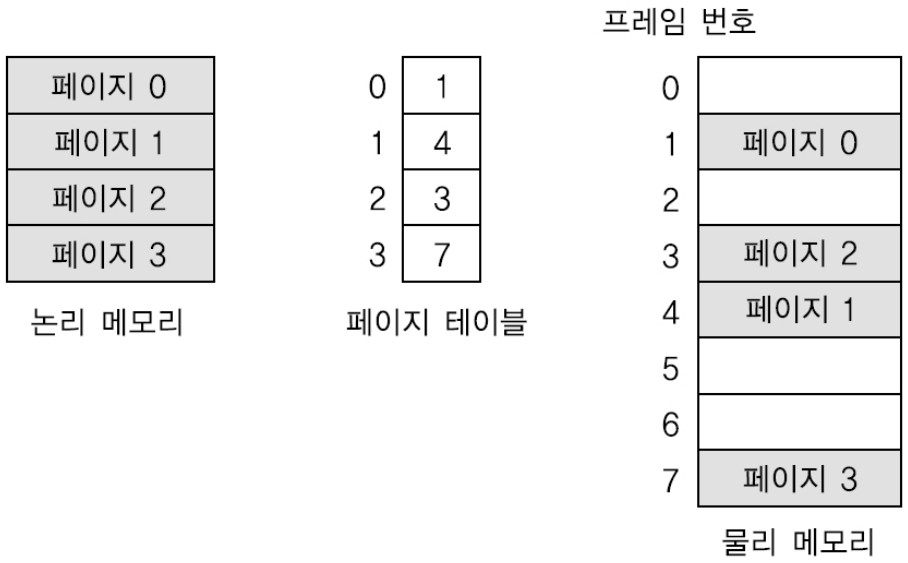


페이징 시스템 하드웨어

- 프로세서에 의해 생성되는 논리 주소는 페이지 번호(p)와 변위(d) 두 부분으로 나뉨
  - 페이지 번호는 페이지 테이블에 대한 색인(p)으로 사용되어 페이지 테이블의 p위치(항목)에 저장된 f값을 얻음
  - f는 실제 메모리의 페이지 기준 주소로 메인 메모리의 프레임 번호가 됨
  - 기준 주소(f)에 페이지 변위(d)가 더해져 실제 메모리 주소(물리 주소)가 결정됨
  - 변위는 페이지 시작위치로부터 얼마나 떨어져 있는가를 나타내는 상대 주소로 프레임 내 위치를 표시하며 페이지 프레임 크기보다 작음

페이징

- 활동 중인 각 작업은 페이지 맵 테이블(PMT, Page Map Table)을 가짐
  - 페이지 맵 테이블은 레지스터로 구성되거나 메인 메모리의 일부로 배정됨
  - 논리 페이지는 페이지 테이블에서 해당 페이지의 프레임 번호를 확인, 실제 물리 메모리의 페이지 프레임을 찾음



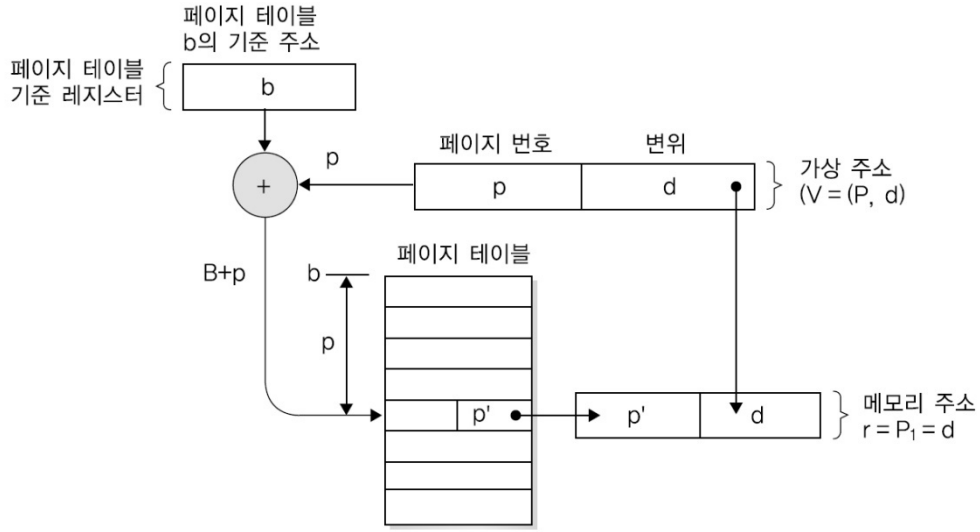
페이지 테이블과 페이징 모델

페이징

- 페이지 테이블의 구현
  - 전용 레지스터를 사용하여 구현
    - 페이징 주소변환을 매우 효율적으로 하기 위해 초고속 논리회로로 설계됨
    - 효율성이 주요 고려 대상임
      - 메모리의 모든 액세스는 페이징 테이블 정보에 의해 수행됨
    - 페이지 테이블 항목이 적을 수록 관리(하드웨어 비용 감소)가 쉬움
  - 페이지 테이블 기준 레지스터(PTBR, Page Table Base Register)
    - 기존 컴퓨터는 페이지 테이블이 매우 크므로, 레지스터로 구현이 적합하지 않음
    - 페이지 테이블을 메모리에 유지, 페이지 테이블 기준 레지스터가 페이지 테이블을 지시함
  - 사상 방법에 따른 주소 변환
    - 직접 사상(Direct Mapping)에 의한 페이지 주소변환
    - 연관 사상(Associative Mapping)에 의한 페이지 주소변환
    - 연관/직접 사상을 결합한 페이지 주소변환

[참고] 페이징

- (1) 직접 사상에 의한 페이지 주소변환
  - 완전한 페이지 사상표가 메인 메모리 또는 고속의 캐시 메모리에 유지됨
  - 가상주소 V=(p, d)를 참조하는 과정
    - 프로세스의 페이지 테이블에 있는 메인 메모리 주소를 페이지 테이블 기준 레지스터에 로드함
    - 가상 주소에 대응하는 메모리 주소를 얻기 위해 페이지 테이블 시작 주소 b를 참조 페이지 번호 p에 더하여 페이지 테이블(b) 내의 색인 p에 관한 메모리 주소 p′를 얻음
    - 페이지 프레임 p′ 가 가상 페이지(페이지 번호) p에 대응하는 것을 가리킴
    - p′ 은 변위 d와 접속, 메모리 주소 r(= p′ + d)를 얻음



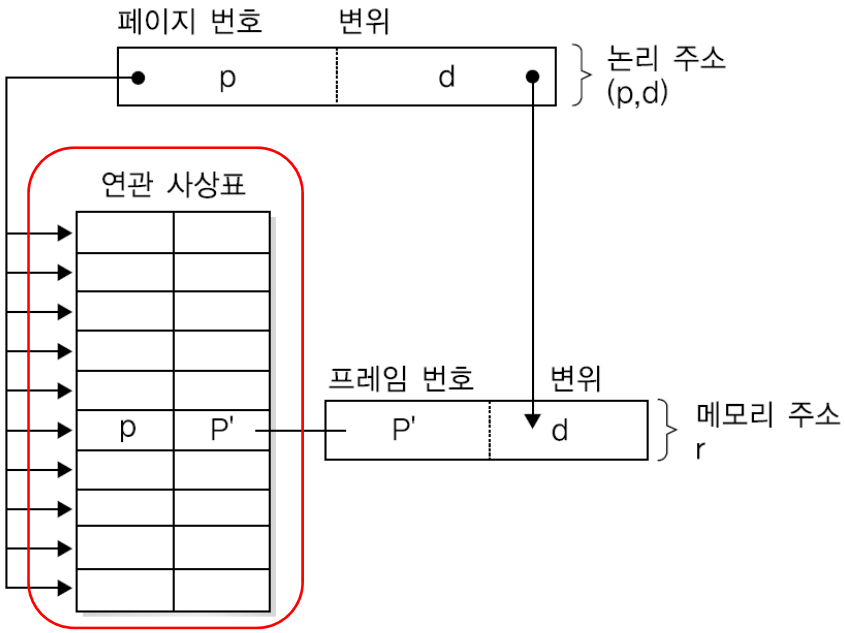
직접 사상에 위치한 페이지 주소변환

[참고] 페이징

- 가상 주소 V가 n개의 페이지를 포함할 경우
  - 사상되는 페이지테이블은 페이지 1, 페이지 2, … , 페이지 n-1에 해당하는 항목을 가짐
- 페이지 테이블 변경은 레지스터 변화로 가능함
  - 메인 메모리에 작성하여 페이지 테이블 기준 레지스터가 페이지 테이블을 지정하는 방법이 바람직함
- 문제점
  - 사용자 메모리 위치에 액세스하는 데 소요되는 시간
  - 주소 p에 액세스하기를 원할 경우, 먼저 p에 대한 페이지 번호의 페이지 테이블 기준 레지스터 변위 값을 사용, 페이지 테이블로 색인
  - 하나의 워드에 액세스하려면, 페이지 테이블을 위해 그리고 워드를 위한 두 번의 메모리 액세스가 필요함
  - 메모리 액세스는 두 배로 느려짐

[참고] 페이징

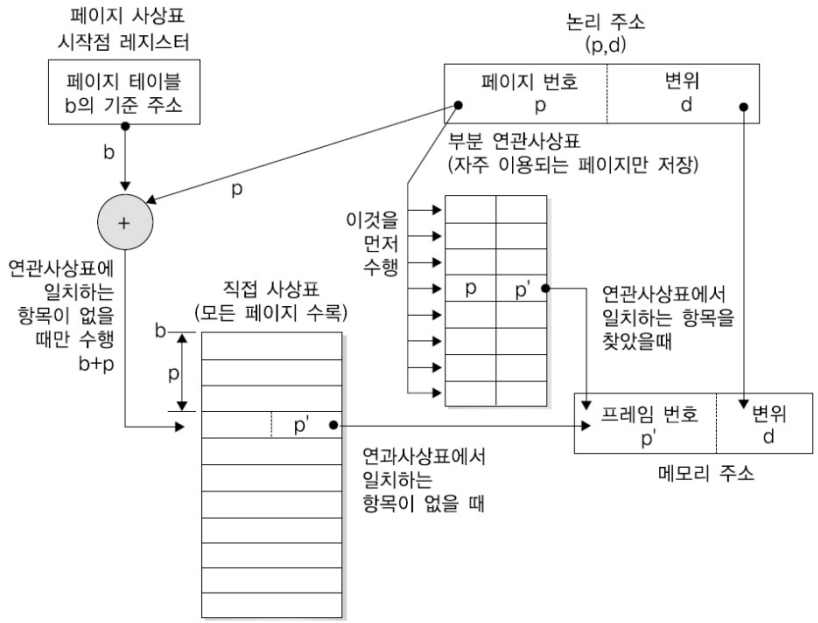
- (2) 연관 사상에 의한 페이지 주소변환
  - 프로세서에 의해 생성된 논리 주소는 연관 레지스터의 집합으로 표현됨
  - 각 레지스터는 키와 값으로 구성
  - 연관 레지스터에 항목 표현과 동시에 모든 키와 비교, 항목이 발견되면 대응하는 값 부분 출력
  - 탐색은 매우 빠르나 하드웨어가 비쌈
  - 페이지 테이블 기준 레지스터(PTBR)은 필요하지 않음
  - 순수 연관 사상을 통한 페이지 주소변환
    - 페이지 p를 찾기 위해 연관 메모리의 모든 항목을 동시에 조사
    - 해당 페이지 프레임 p' 가 발견되면 d와 접속, 메모리 주소 형성
    - 연관 사상표로 향하는 화살표들이 실제로 사상표의 모든 단위 항목으로 표시됨
    - p와 일치하는 것을 찾기 위해 연관 메모리의 모든 단위항목을 동시에 조사함을 의미함
    - 이로 인해 연관 메모리가 비싸짐



순수 연관 사상을 통한 페이지 주소변환

[참고] 페이징

- (3) 연관/직접 사상을 통한 페이지 주소변환
  - 최근 페이지만을 연관 메모리에 유지, 연관 메모리에 그 페이지가 없을 때 직접 사상 방법 제안
  - 지역성을 이용, 최근 참조된 페이지는 곧 다시 사용된다는 것을 적절하게 이용함
  - 연관/직접 사상을 통한 페이지 주소변환
    - 현재 수행 중인 프로그램이 페이지 p를 찾기 위해 연관 메모리의 모든 항목을 동시에 조사함
    - 페이지 p에 해당되는 페이지 프레임 p′ 이 발견되면 d와 접속, 메모리 주소 r= p′ + d를 얻음
    - p와 일치하는 항목이 연관 사상표에 없는 경우 논리 주소의 페이지 번호(p)를 페이지 테이블 기준 레지스터의 b와 합하여 p+b의 직접 페이지 테이블의 p에 대응하는 페이지 프레임 p′ 를 얻음
    - 이어서 p′ +d의 메모리 주소 r을 생성함



연관/직접 사상을 통한 페이지 주소변환

페이징

- 적중률
  - 페이지 번호가 연관 레지스터에서 발견될 비율
    - 적중률 80%일 경우
    - 연관 레지스터 탐색 시 50ns, 메모리에 접근 시 750ns가 걸리면 사상된 메모리 액세스는 페이지 번호가 연관 레지스터에 있는 경우 800ns가 걸림
    - 연관 레지스터에서 페이지 번호를 찾지 못한 경우(50ns), 우선 페이지 테이블과 프레임 번호를 액세스한 후 메모리에서 원하는 워드에 액세스함(750ns). 총 1550ns가 소요됨
    - 메모리 액세스 시간에서 26.6% 늦어짐(750ns에서 950ns)

유효접근시간: (0.8x800)+(0.2x1550)=950ns

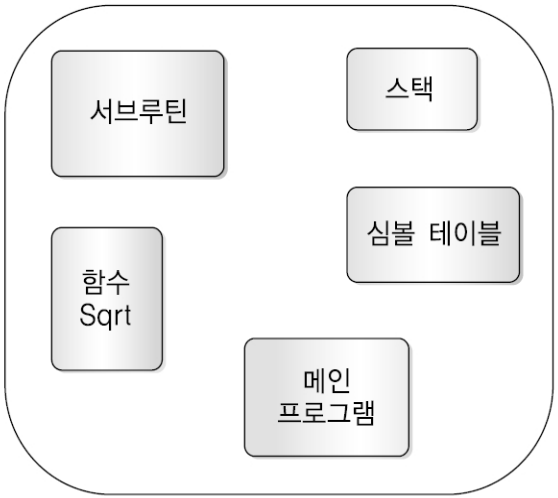
  - 적중률 90%인 경우
    - 증가된 적중률은 메모리 액세스 시 23%로 줄어 듬

유효접근시간: (0.9x800)+(0.1x1550)=875ns

  - 적중률은 연관 레지스터의 수와 관계됨
    - Intel 80486 프로세스는 레지스터 32개, 적중률 98%임

세그먼테이션

- 세그먼트 메모리 관리 기법
  - 메모리의 사용자 관점 지원
    - 메모리는 크기를 변경 가능한 세그먼트 단위 모음
    - 고정 크기를 갖는 페이징과는 다름
    - 메모리를 일반적으로 프로그램을 구성하는 서브루틴(Sub-Routine), 프로시저(Procedure), 함수(Function) 또는 모듈(Module) 등의 각각 다른 크기를 갖는 세그먼트로 나눔
    - 각 세그먼트는 연관된 기능을 수행하는 하나의 모듈 프로그램
    - 각 세그먼트는 연속된 위치에 구성되나, 메모리에서 서로 인접할 필요는 없음



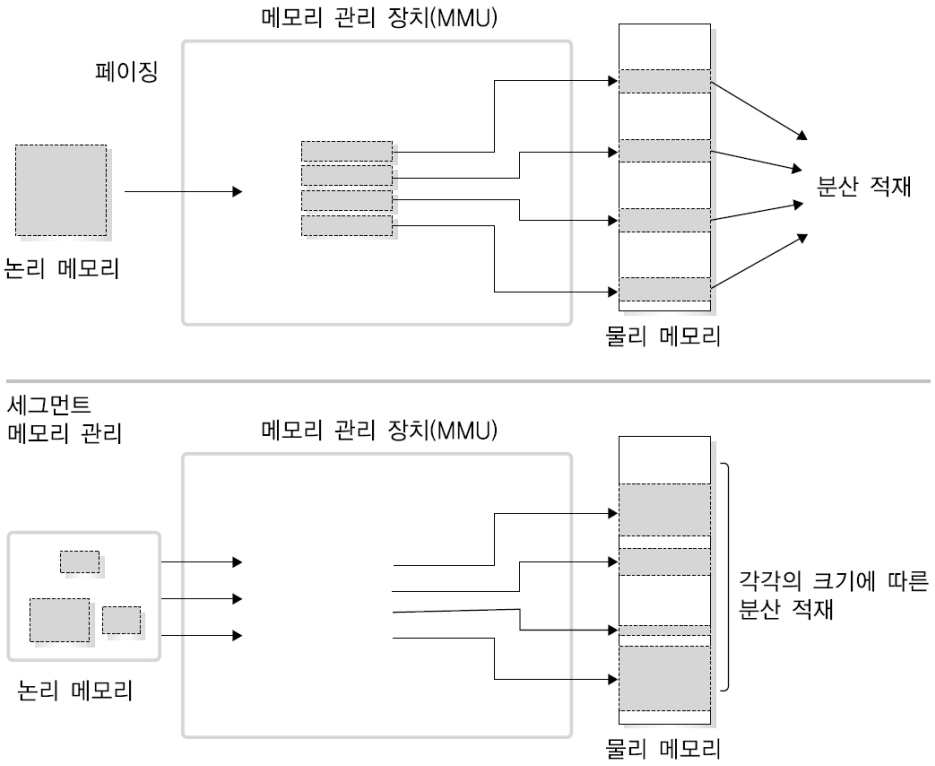
논리 주소 공간

프로그램의 사용자 관점

세그먼테이션

- 세그먼트 메모리 할당
  - 논리 구조 공간을 세그먼트의 모임으로 인식
    - 세그먼트는 일반적으로 컴파일러(어셈블러)가 자동적으로 입력 프로그램(원시 프로그램)을 이용하여 작성함
    - 하드웨어 보호 등 관리에 필요한 세부 사항은 페이징과 비슷하거나 동일함
    - 동적 분할(가변 분할) 기법으로 메모리를 할당

      - 프로세스에 따라 세그먼트 크기가 다르므로 메모리가 일정한 크기의 페이지 프레임으로 나누지 않음



페이징과 세그먼트 메모리 할당 비교

세그먼테이션

- 세그먼트 번지와 세그먼트 테이블 구현
  - 각 프로세스는 자신의 세그먼트 테이블을 가지며, 메인 메모리에 적재됨
  - 각 세그먼트 테이블 항목은 해당 세그먼트가 적재될 메모리의 시작 주소를 알려줌

    - 잘못된 주소들을 사용할 수 없도록 세그먼트의 길이도 함께 알려줌
  - 세그먼트 테이블은 메인 메모리에 유지함

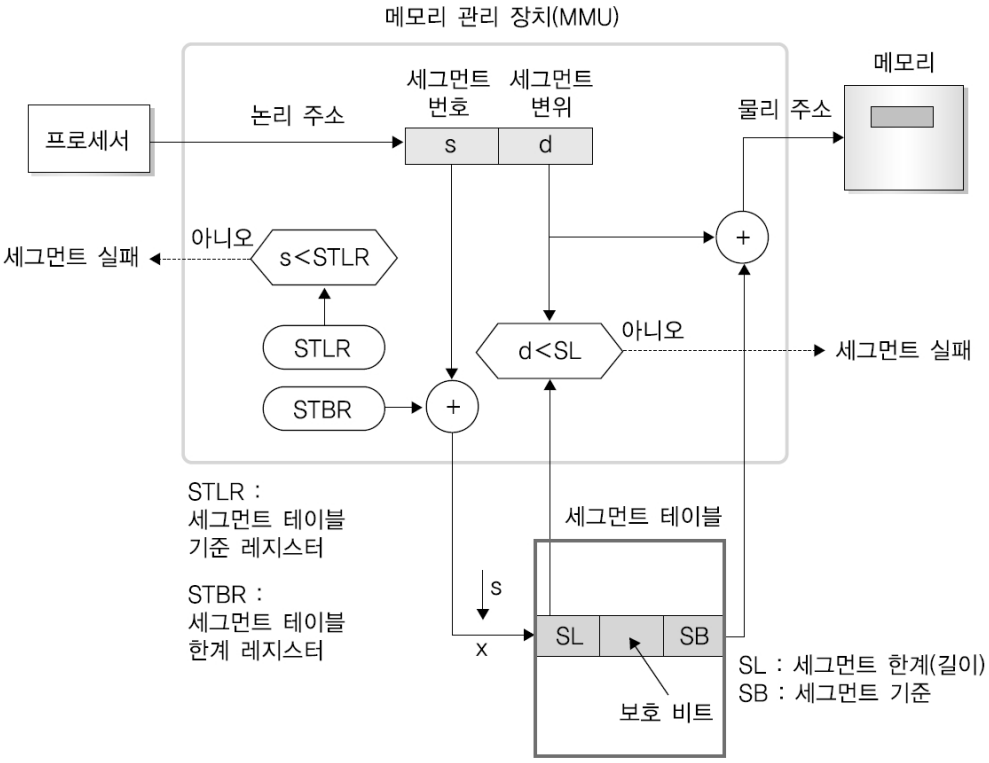
    - 프로세스의 크기에 따라 길이가 가변적이므로 레지스터에 정보를 유지하기 어려움
  - STBR (Segment-Table Base Register)는 메모리에 있는 세그먼트 테이블의 주소(위치)를 나타냄
  - STLR (Segment-Table Length Register)는 프로그램에 의해 사용되는 세그먼트 수

세그먼테이션

- 하드웨어 구조
  - 세그먼트의 논리 주소는 세그먼트 번호 s와 세그먼트 내의 변위 d로 구성됨.
    - 세그먼트 테이블 기준 레지스터(STBR)와 세그먼트 테이블 길이 레지스터(STLR)는 프로세스가 디스패치될 때 로드되어 논리 주소를 검사함

- 세그먼트 하드웨어 구조
  - 세그먼트 논리 주소(s, d)에 대한 세그먼트 번호의 유효성(s < STLR)을 점검
  - 세그먼트 번호(s)가 STLR 값보다 크면 세그먼트 실패로 오류
  - 세그먼트 번호와 STBR을 이용, 실제 메모리의 주소(STBR + s)를 산출함
  - 세그먼트 번호 s는 세그먼트 테이블에 대한 색인으로 사용됨
  - 세그먼트 테이블의 각 항목은 세그먼트의 기준(SB)와 세그먼트의 한계(SL), 즉 크기(길이)를 가짐
  - 세그먼트 주소의 변위 d는 0과 SL 사이의 값이어야 함
  - 변위가 합당한 경우(d < SL) 세그먼트 기준(SB)과 변위(d)가 더해져 메모리 주소를 만듬

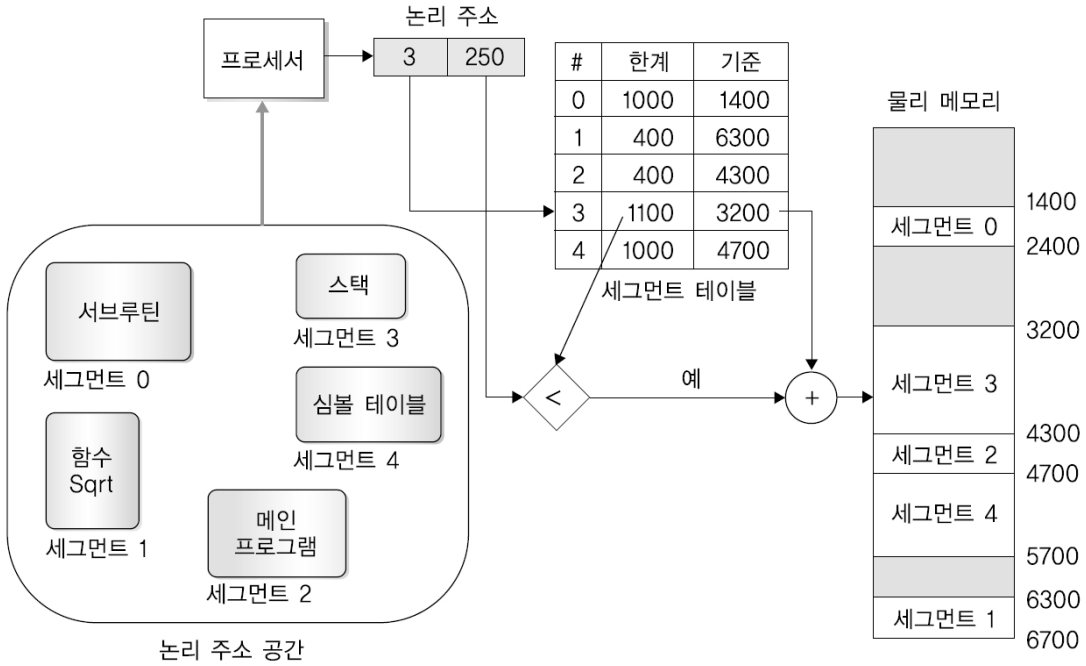
※ 세그먼트 테이블은 각 세그먼트에 대한 한계와 기준 항목을 가지고 메모리 내의 메모리 주소의 시작과 세그먼트의 끝을 나타내는 기준/한계 레지스터의 쌍들로 이루어진 배열임



세그먼트 하드웨어

세그먼테이션

- 세그먼테이션의 예
  - 0에서 4까지 번호가 정해진 5개의 세그먼트가 있다고 가정함
  - 세그먼트 2는 400byte 길이로 4300번지에서 시작
  - 세그먼트 2에서 워드 53에의 액세스는 4353(=4300 + 53)번지로 사상됨
  - 세그먼트 3, 워드 250에 대한 액세스는 3450(=3200+250)번지로 사상됨
  - 세그먼트 0의 워드 1222에 대한 참조인 경우 세그먼트가 1000byte의 길이를 가지므로 (초과) 세그먼트 실패로 운영체제로 넘어감



세그먼테이션의 예

세그먼테이션

- 단편화
  - 모든 세그먼트에 대한 적당한 기억 장소를 찾아 할당
    - 페이징 시스템의 일정한 크기(페이지)에 비해 세그먼트 기법은 크기가 변함
    - 외부 단편화 발생 가능

※ 외부 단편화 문제는 평균 세그먼트 크기에 의존하므로, 평균 세그먼트 크기가 작으면 외부 단편화 또한 작음