가상 메모리

가상 메모리의 정의
> 물리적 메모리 크기의 한계를 극복하기 위해 나온 기술
> 프로세스를 실행할 때 실행에 필요한 일부만 메모리에 로드, 나머지는 디스크에 두는 것
(실행은 프로세스 전체가 물리적 메모리에 있는 것처럼 수행됨)

관리
> 가상 주소(logical address) - 가상으로 주어지는 주소
> 실제 주소(phtsical address) - 실제 메모리상에 있는 주소

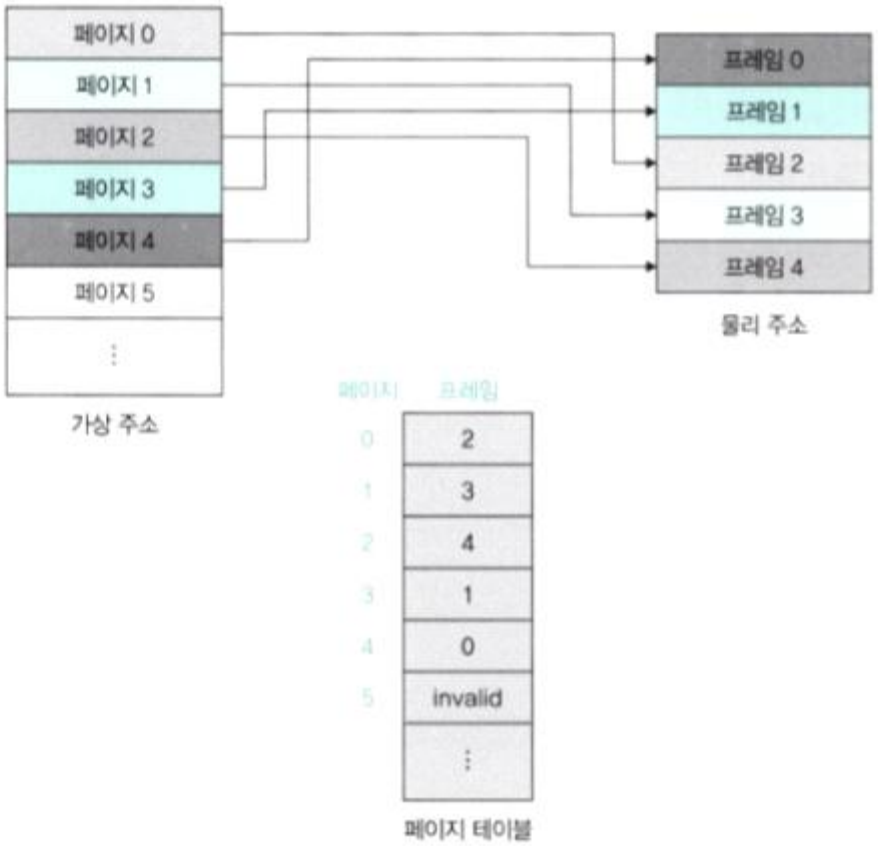
가상 주소는 메모리 관리 장치(MMU)에 의해 실제 주소로 변환
-> 사용자는 실제주소 의식할 필요 없이 프로그램 구축 가능

가상 메모리는 가상 주소와 실제 주소가 매핑 되어 있음,
프로세스의 주소 정보가 들어있는 **페이지 테이블**로 관리된다.
이때 속도 향상을 위해 **TLB**를 사용한다.

TLB?
TLB(Translation Lookaside Buffer, 페이지 정보 캐쉬)
메모리와 CPU 사이에 있는 주소 변환을 위한 캐시이다. 최근에 일어난 가상 메모리와 물리 주소의 변환 테이블을 저장해둔다.
CPU가 가상 주소를 가지고 메모리에 접근하려고 할 때 우선은 TLB에 접근하여 가상 주소에 해당되는 물리 주소를 찾고, 만약 TLB에 매핑이 존재하지 않는다면 MMU가 페이지 테이블에서 해당되는 물리 주소로 변환한 후 메모리에 접근하게 된다.
TLB 사용 이점: 물리주소를 갖고 있으면 메모리(RAM)에 두 번 들릴 필요없이, 바로 해당 물리주소(in 메모리)를 찾아갈 수 있음.
출처 : https://ahnanne.tistory.com/15

#페이징 기법
페이징 기법의 정의 : 고정 분할 방식을 이용한 가상 메모리 관리 기법, 물리 주소 공간을 같은 크기로 나누어 사용한다. 여기서 가상 주소의 분할된 각 영역을 '페이지'라고 부른다.
> 물리 메모리의 각 영역은 가상 주소의 페이지와 구분하기 위해 프레임(frame)이라고 부른다. (페이지와 프레임은 각각 번호를 매겨 관리)
> 페이지와 프레임의 크기는 같다.
> 페이지 테이블은 하나의 열(column)로 표현된다.

프레임(Frame): 물리 메모리를 사용하는 최소 크기 단위.
페이지(Page): 가상 메모리를 사용하는 최소 크기 단위.



페이징 기법의 주소 변환 :
> 페이징 기법에서 페이지는 페이지 테이블이라는 **자료구조**로 관리
> 가상주소는 페이지 번호(P)와 변위(d)로 구성
ex) 가상 주소 v - (p,d) p는 페이지 번호, d는 페이지 p의 시작 위치로부터의 변위
> 프레임 시작 주소(f) + 변위(d)를 통해 물리 주소를 계산, 실제 물리 주소에 접근함
> 요구 페이징(Demand Paging) : 프로세스의 모든 데이터를 물리 메모리에 적재하지 않고, 필요한 시점에만 메모리에 적재. 필요하지 않은 페이지 프레임은 물리 메모리에서 내림
> 페이지 부재(page fault) : 가상 메모리 공간에는 존재하지만 실제 물리 메모리에는 없을 때 일어나는 '인터럽트', 페이지 폴트가 발생하면 os에서 해당 페이지를 물리 메모리에 올림

*각 프로세스는 하나의 페이지 테이블을 가짐!!!
*페이지 테이블은 os영역에 있음!!!(페이지 테이블++ , 프로세스 실제 사용 메모리 영역--)
*스왑 공간은 물리적 메모리(RAM)의 용량이 가득 차게될 경우 사용되는 여유 공간

페이지 매핑 방식 :

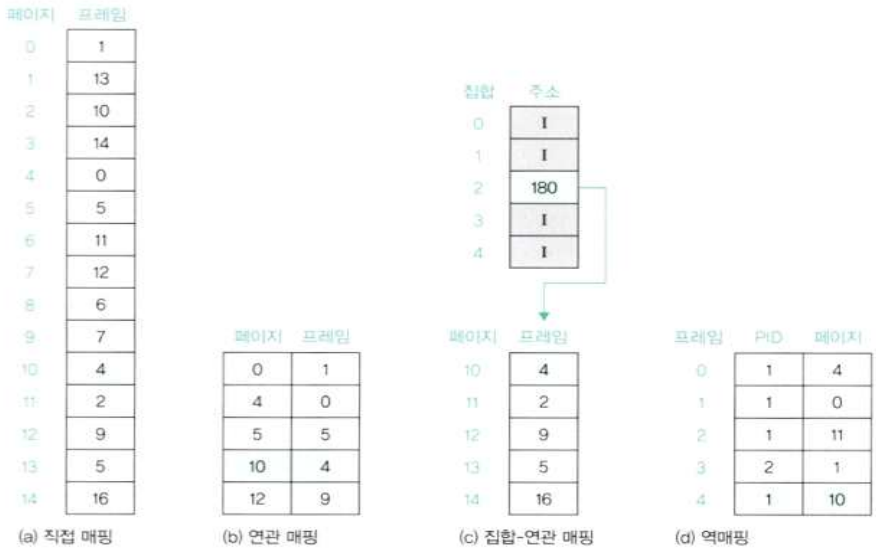


그림 8-12 페이지 테이블 매핑 방식의 특징

1. 직접 매핑 : 페이지 테이블 전체가 물리 메모리의 os에 존재하는 방식, 페이지 테이블을 직접 활용
2. 연관 매핑 : 테이블 전체를 스왑 영역에 관리하는 방식, 물리 메모리 여유가 적을 때 사용. 모든 테이블 페이지를 저장장치의 스왑 영역에 저장, 그 중 일부만 물리 메모리에 가지는 방식
3. 집합-연관 매핑 : 페이지 테이블을 같은 크기의 여러 집합으로 나누고, 각 묶음의 시작 주소를 가진 위치 테이블을 새로 만들어 관리!
4. 역 매핑 : 물리 메모리의 프레임 번호를 기준으로 테이블을 구성. 물리 메모리의 프레임에 어떤 프로세스의 어떤 페이지가 올라와 있는지를 표시 즉, 프로세스 수 상관 없이 테이블 하나만 존재, 테이블 크기가 매우 작다. (모든 페이지 테이블을 다 검색한 후에야, 해당 페이지가 스왑 영역에 있다는 것을 알게 되므로 속도가 느려 질 수 있다.)

페이징 장점 : 물리 메모리를 같은 크기로 나누어 관리하기 때문에 메모리 관리가 수월하다. 외부 단편화 안 생김 (같은 크기로 나누기 때문에)
페이징 단점: 내부 단편화 생길 가능성 있음 (같은 크기로 나누기 때문에)

#세그먼테이션 기법

정의 : 가변 분할 방식을 이용한 가상 메모리 관리 기법(메모리 프로세스 크기에 따라 가변적으로 나누어 사용)

> 세그멘테이션 기법에도 매핑 테이블 사용, 세그멘테이션 테이블이라 함

> 세그먼트 크기인 limit과 물리 메모리상의 시작 주소인 address로 구성

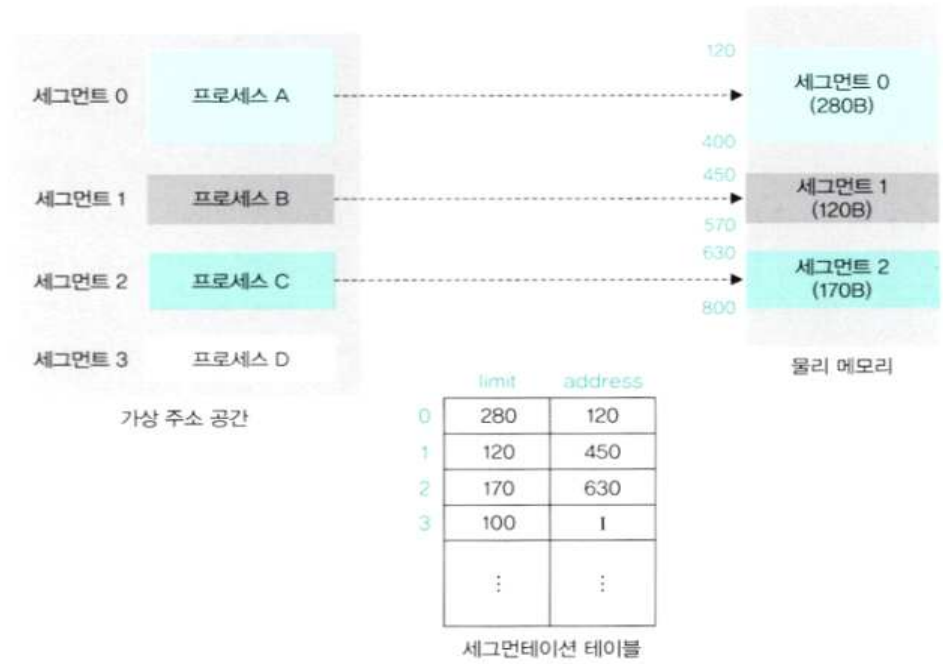
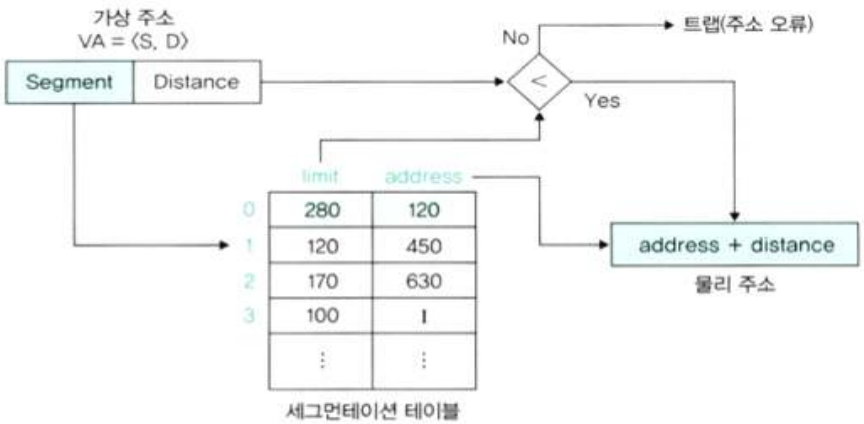


그림 8-18 세그먼테이션 기법

세그먼테이션의 주소 변환:



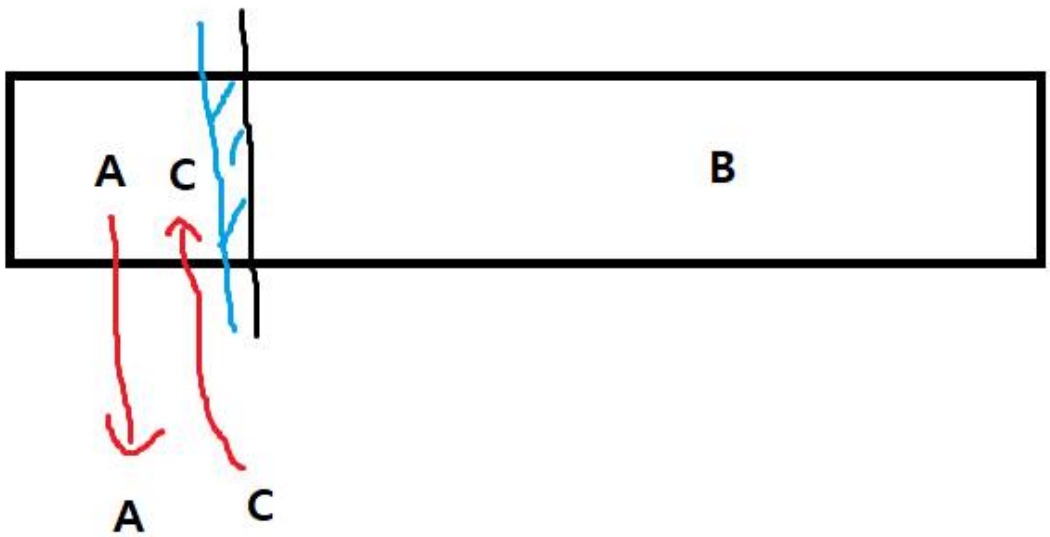
가상 주소를 VA=(S,D)라고 표현
(S- 세그먼테이션 번호, D- 세그먼트 시작 지점에서 해당 주소까지 거리)
장점 : 내부 단편화 문제 해소, 보호와 공유 기능을 수행할 수 있다. 프로그램의 중요성에 따라 분리하여 저장 가능
단점 : 외부 단편화 문제로 인해 추가적 관리가 불가피함

#단편화

정의 : 주기억장치 상에서 빈번하게 기억 장소가 할당되고 반납됨에 따라 기억장소들이 조각들로 나누어지는 현상!!
메모리 공간이 조각조각 나뉘어 실제로 사용가능한 메모리가 충분히 존재하지만, 할당이 불가능한 상태가 발생

내부 단편화 : 프로세스가 요청한 양보다 더 많은 메모리르 할당할 때 발생, 메모리 분할 자유공간과 프로세스가 사용하는 공간의 크기 차이를 의미 (프로세스가 필요한 양보다 더 많은 메모리가 할당된 상태 ㅠㅠ)

외부 단편화 : 메모리 할당 및 해제 작업의 반복으로 작은 메모리가 중간 중간 존재할 수 있다. 이렇게 사용하지 않는 메모리가 존재해서 총 메모리 공간은 충분하지만 실제로 할당할 수 없는 상황. (메모리 중간 중간 사용하지 않는 공간이 생겨서 더 이상 사용할 수 없는 상황)



#세그먼테이션-페이징 혼용 기법

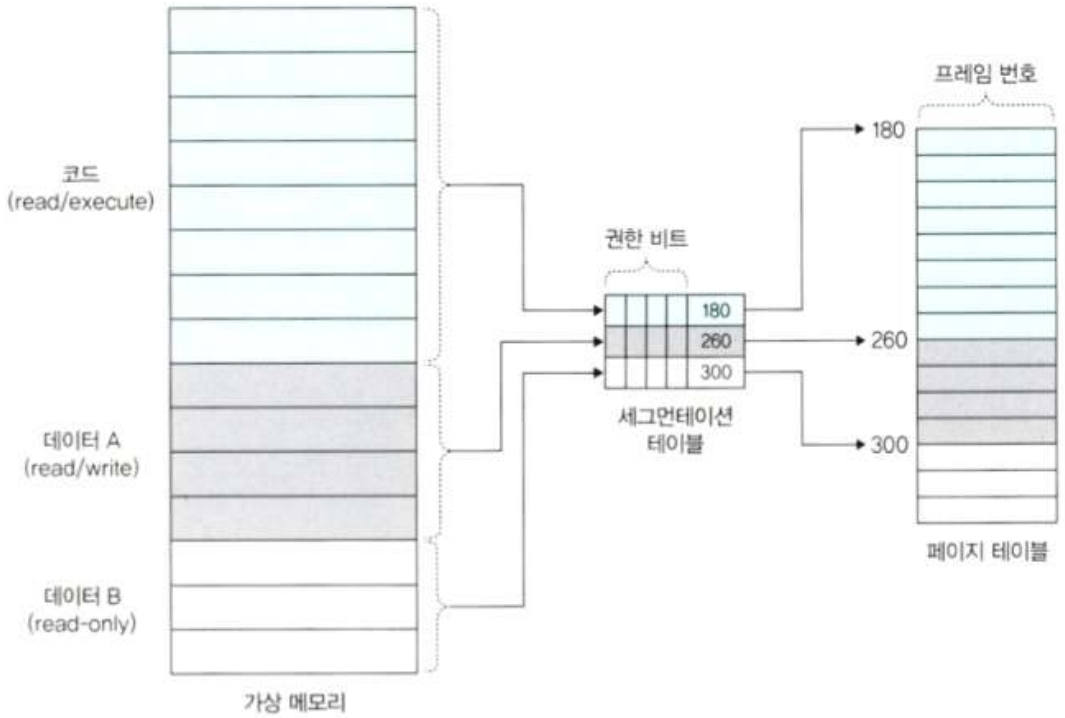


그림 8-22 페이징 테이블과 세그먼테이션 테이블의 혼합

> 세그먼테이션-페이징 혼합 기법은 페이지로 분할된 가상 주소 공간에서 서로 관련 있는 영역을 하나의 세그먼트로 묶어 세그먼테이션 테이블로 관리하고, 각 세그먼트를 구성하는 페이지를 해당 페이지 테이블로 관리하는 방식이다.

> 페이징 기법에 세그먼테이션을 추가하고, 중복되는 데이터를 세그먼테이션 테이블로 옮겨오면 테이블의 크기를 줄일 수 있다.

> 현재의 대부분의 OS에서 이 기법을 활용하고 있다.