10. Memory Management 3

Inverted Page Table

지금까지의 page table은 process 마다 page table을 가졌는데

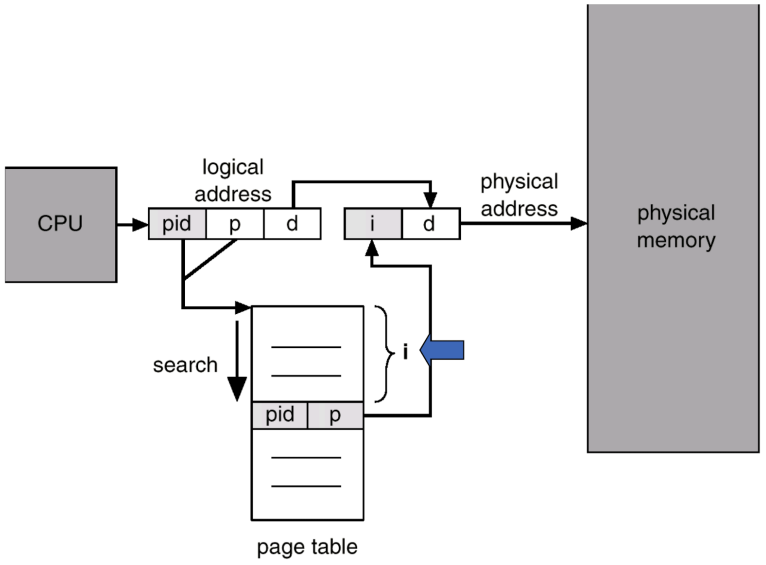
⇒ 아니 이럴 필요가 있냐 이거지 (page table 저장하느라 저장공간 터지겠다)

그래서 해결책으로 내놓은게 physical page를 logical page로 mapping 시키자

이러면 시스템 전반에 단 하나의 page table만 두는 거

왜 physical page(frame)을 역으로 mapping하는게 더 적은 메모리를 잡아먹어?

⇒ physical frame의 개수는 상한이 있으니까



그래서 이제는 physical memory에 올라와있는 logical address를 찾는거

page table이 시스템 전체에 걸쳐 단 하나만 존재하니까 이 주소가 어떤 프로세스의 주소를 말하는지 알려주기 위해서 pid를 지정해줘야 돼

장점: 메모리적으로는 아낄 수 있음

단점: 검색할 때 무조건 선형탐색할 수밖에 없음(물론 이거는 hash table 써서 key값을 pid와 p로 두면 빠르게 찾을 수 있긴 한데, 페이지 찾겠다고 이런 연산을 지속적으로 하겠다고...?)

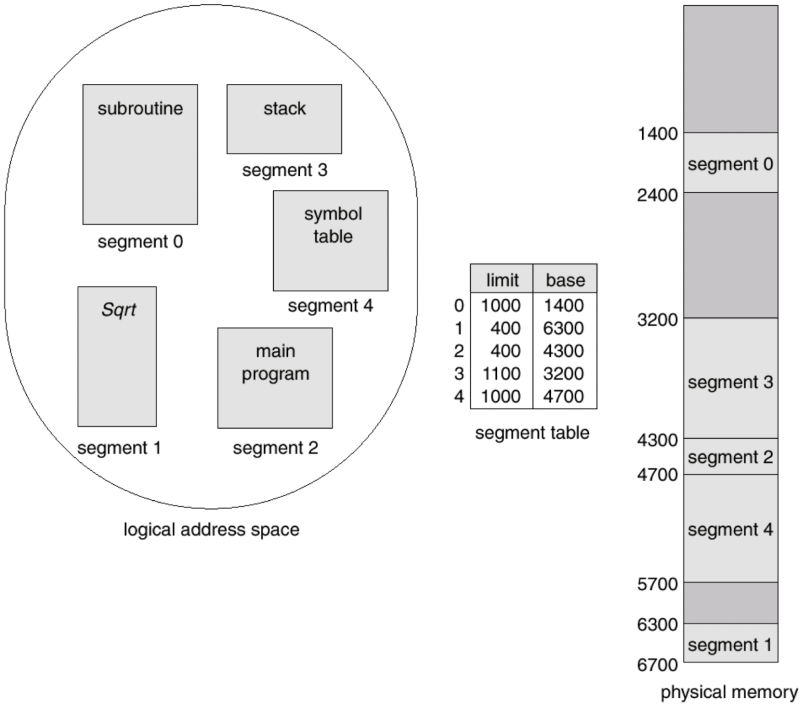
그래서 내놓은 다른 방법이 TLB 사용

(어차피 physical address 크기만큼 page table이 결정될테니까 TLB 쓰면 성능 향상 노려볼만 하지)

Segmentation

지금까지는 process를 물리적인 크기로만 잘랐는데 이제는 의미 단위로 자를 예정

⇒ 이러면 좋은게 확실히 하나의 작업을 할 때 그 작업이 통으로 메모리에 올라오니까 이득이 있지



이런식으로 하나의 프로세스를 필요한 의미단위로 분리하고 paging처럼 physical memory에 할당

Segment table

1. base

   physical address의 시작 주소를 저장

2. limit

   해당 segment의 길이를 저장

동작하는 방식은 MMU와 동일

Segmentation Architecture

1. Protection

   segment table의 entry에서 valid, invalid를 통해 비정상적인 접근을 방지할 수 있음

   의미 단위로 나눴기 때문에 가장 이득을 볼 수 있는 부분이 R,W,X 로 쪼갤 수 있는 것

2. Sharing

   segment가 애초에 read 작업만 이뤄지는 경우, 모든 스레드에 대해 허용해도 문제 없으니까

   이런 점에서 sharing은 큰 이득을 낼 수 있어

3. Allocation

   paging이랑 다르게 external fragmentation이 발생(애초에 딱 맞게 할당해버리니까)

   주로 first-fit 방식 사용

Segmentation with Paging

segment에서 발생한 external fragmentation을 해결하기 위해 paging을 도입

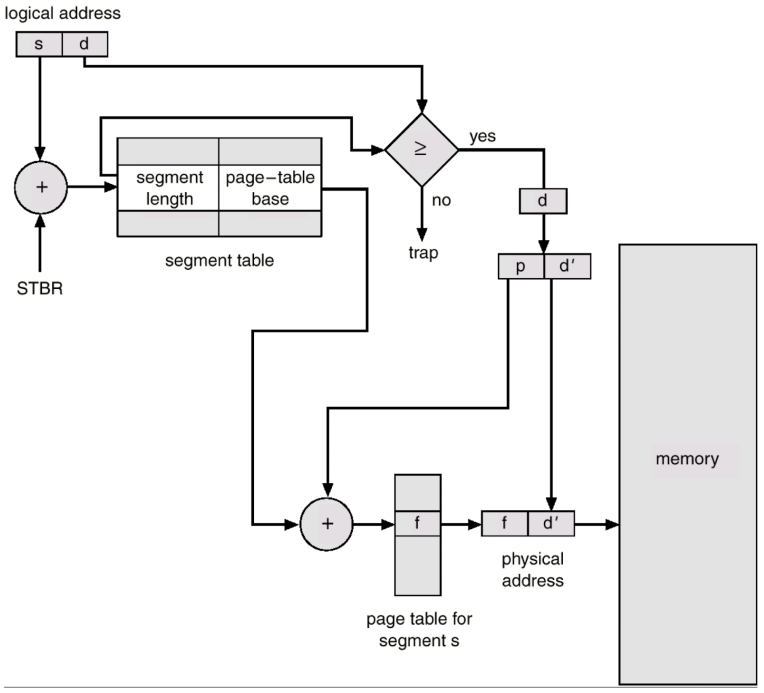
그래서 segment를 page 단위로 찢어 ⇒ 이렇게 하다보니 segment 마다 page table이 필요

Segment number s	Page number p	Displacement d

Virtual address v = (s, p, d)

(Virtual address format in a paged and segmented system)

virtual address를 나타내면 이런식으로 나옴



1. segment 번호랑 offset을 들고 어디에 있는지 확인
2. STBR(base register)에 있는 base 주소값을 더해서 segment주소를 segment table에서 찾아
3. segment table에서 segment 발견하고, 해당 segment의 page table 주소를 받아
4. segment length 가 d 이상인지 확인 (d가 offset이니까)
    a. 조건을 충족하면 5번
    b. 충족하지 않으면 trap
5. 이제 d에 해당하는 page 번호와 page offset을 찾아
6. page 번호 찾아서 page-table base 주소를 이용해서 page frame을 찾아
7. 실제 physical frame 번호와 page offset을 이용해서 physical address 발견