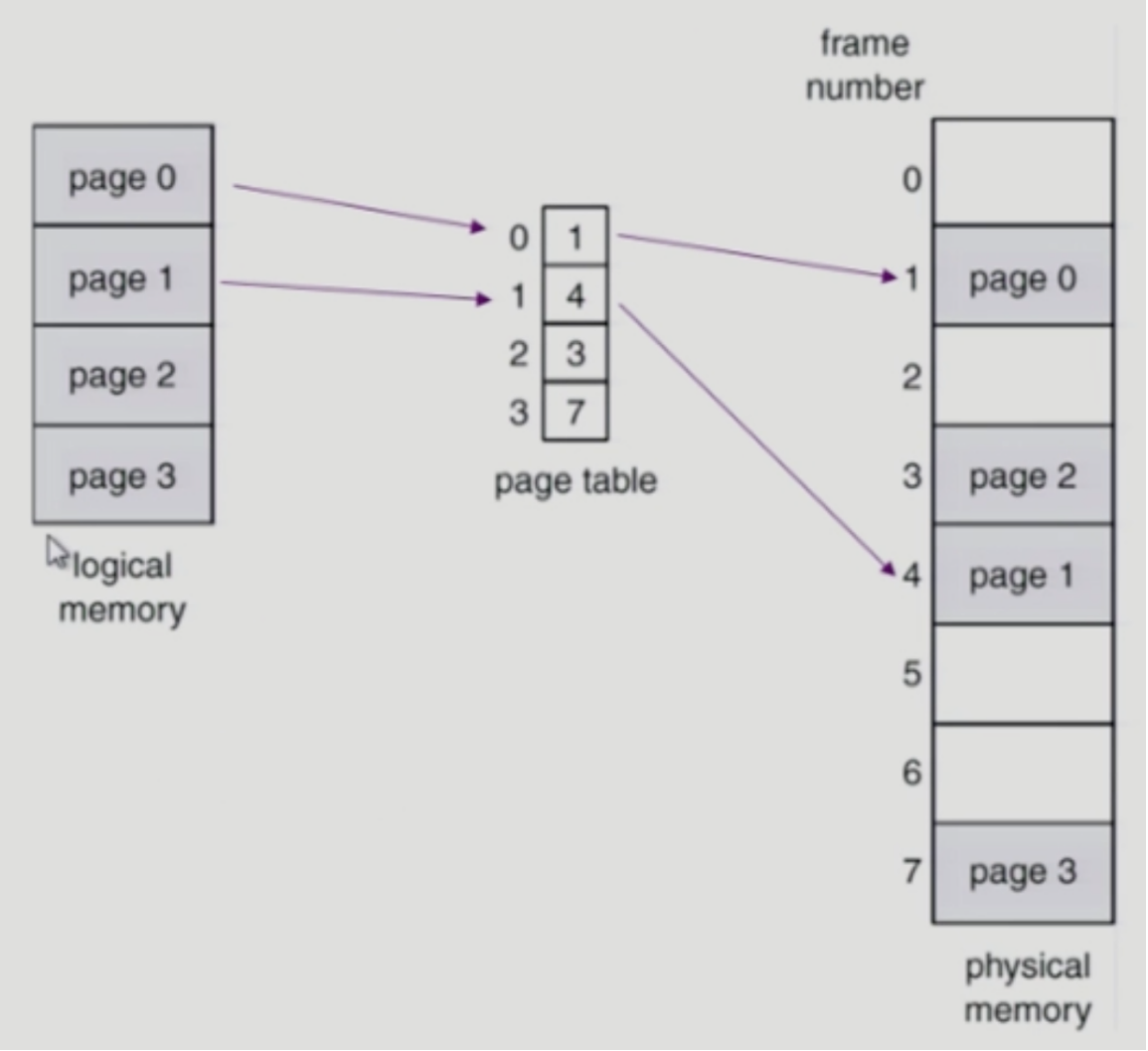
3주차_메모리 관리2

상태	진행 중
강의	CS 스터디
작성일	@2024년 2월 16일

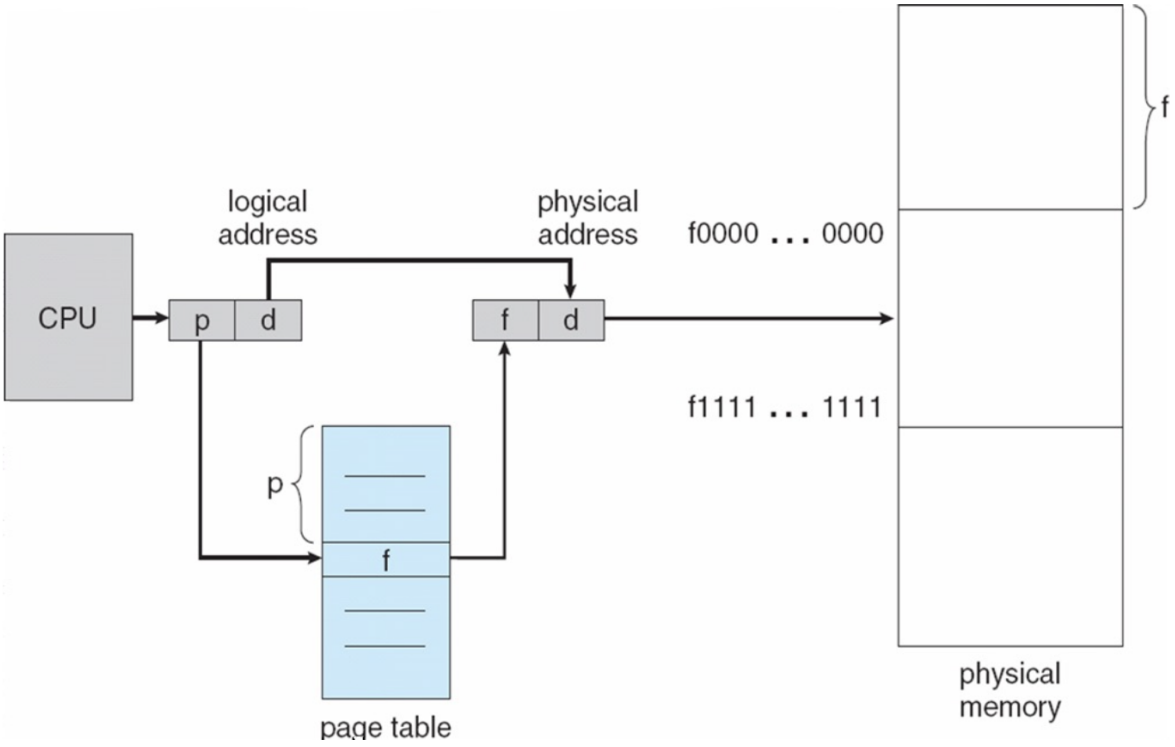
Paging

- Paging
  - Process의 virtual memory를 동일한 사이즈의 page 단위로 나눔
  - Virtual memory의 내용이 page 단위로 noncontiguous하게 저장됨
  - 일부는 backing storage에, 일부는 physical memory에 저장
- Basic Method
  - physical memory를 동일한 크기의 frame으로 나눔
  - logical memory를 동알한 크기의 page로 나눔 (frame과 같은 크기)
  - 모든 가용 frame들을 관리
  - page table을 사용하여 logical address를 physical address로 변환
  - External fragmentation 발생 안함
  - Internal frgmentation 발생 가능

Paging Example



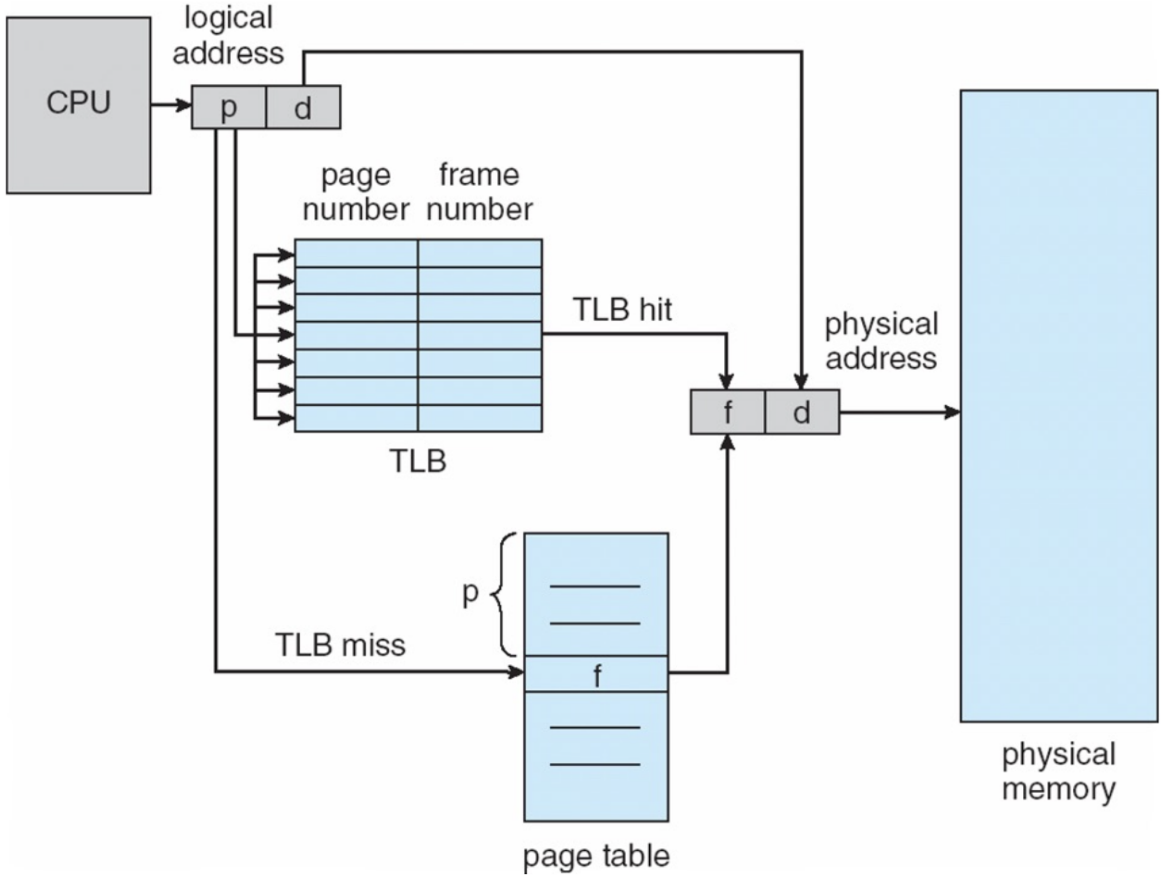
Address Translation Architecture



Implementation of Page Table

- Page table은 main mamory에 상주
- Page-table base register(PTBR)가 page table을 가리킴
- Page-table length register(PTLR)가 테이블 크기를 보관
- 모든 메모리의 접근 연산에는 2번의 memory access 필요
- page table 접근 1번, 실제 data/instruction 접근 1번
- 속도 향상을 위해
  associative register 혹은 translation look-aside buffer (TLB)
  라 불리는 고속의 lookup hardware cache 사용
  (순차적 탐색이 아닌 병렬적으로 한번에 모든 열을 탐색하는 것)

Paging Hardware with TLB



→ 프로세스 마다 page table이 있는거니까 프로세스가 여러개면 page table 도 여러개

Associative Register

- Associative registers(TLB) : parallel search가 가능
  - TLB에는 page table 중 일부만 존재
- Address translation
  - page table 중 일부가 associative register에 보관되어 있음
  - 만약 해당 page #가 associative register에 있는 경우 곧바로 frame #를 얻음
  - TLB는 context switch 때 flush (remove old entries)

Effective Access Time

- Associative register lookup time = β
- memory cycle time = α

- Hit ratio = ε
  - associative register에서 찾아지는 비율
- Effective Access Time (EAT)

$$\underset{\text{<hit>}}{} \qquad \underset{\text{<miss>}}{}$$

$$EAT = (\alpha + \beta)\,\varepsilon + (\alpha + 2\,\beta)(1 - \varepsilon)$$

$$= \alpha + (2 - \varepsilon)\,\beta$$

- Associative register lookup time = ϵ
- memory cycle time = 1
- Hit ratio = α

$$\underset{\text{<hit>}}{} \qquad \underset{\text{<miss>}}{}$$

$$EAT = (1 + \varepsilon)\,\alpha + (2 + \varepsilon)(1 - \alpha)$$

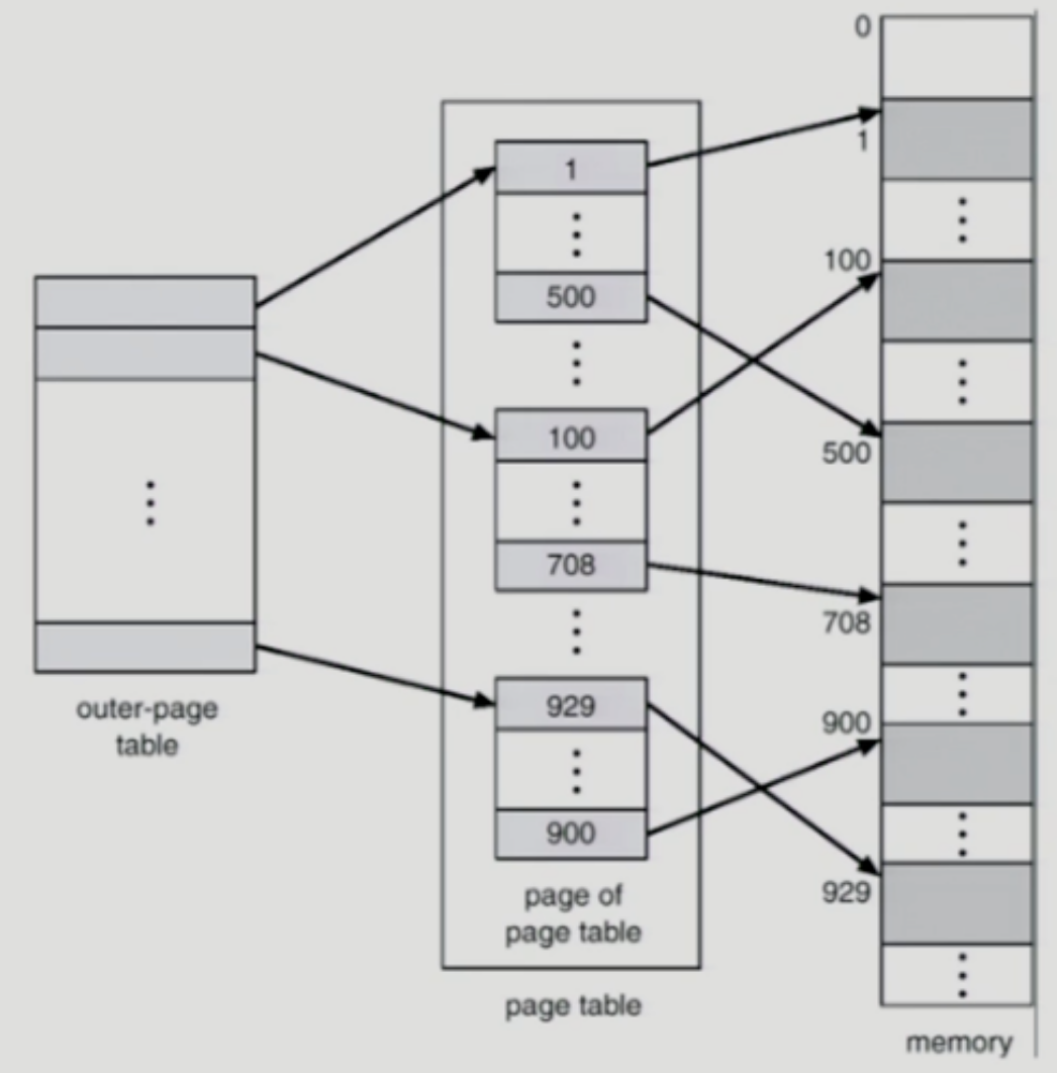
$$= 2 + \varepsilon - \alpha$$

Two-Level Page Table

- 현대의 컴퓨터는 addres space가 매우 큰 프로그램 지원
  - 32bit address 사용시 : 2^{23}(4G)의 주소 공간
    - page size가 4B시 프로세스당 4M의 page table 필요
    - 그러나, 대부분의 프로그램은 4G의 주소 공간 중 지극히 일부분만 사용하므로 page table 공간이 심하게 낭비됨

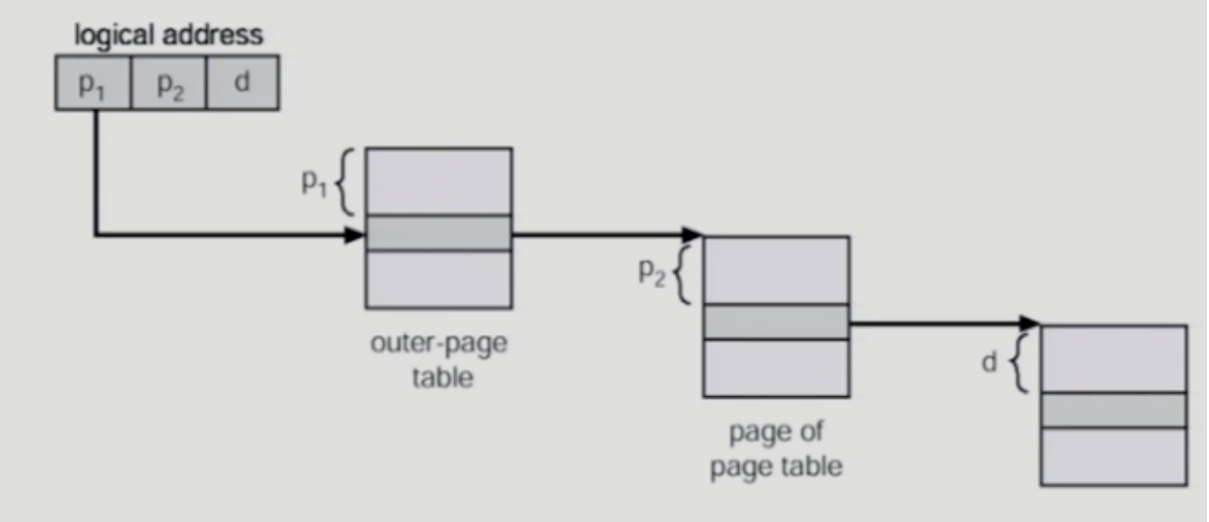
  → page table 자체를 page로 구성

  → 사용되지 않는 주소 공간에 대한 outer page table의 엔트리 값은 NULL (대응하는 inner page table이 없음)



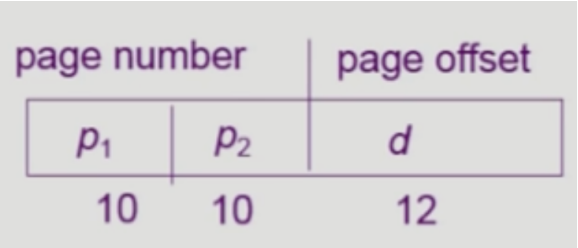
Address-Translation Scheme

- 2단계 페이징에서의 Address-translation scheme



Low-level paging Example

- local address (on 32-bit machine with 4K page size)의 구성
  - 20bit의 page number
  - 12bit의 page offset
- page table 자체가 page로 구성되기 때문에 page number는 다음과 같이 나뉜다. (각 page table entry가 4B)
  - 10-bit의 page number
  - 10-bit의 page offset
- 따라서, logical address는 다음과 같다.



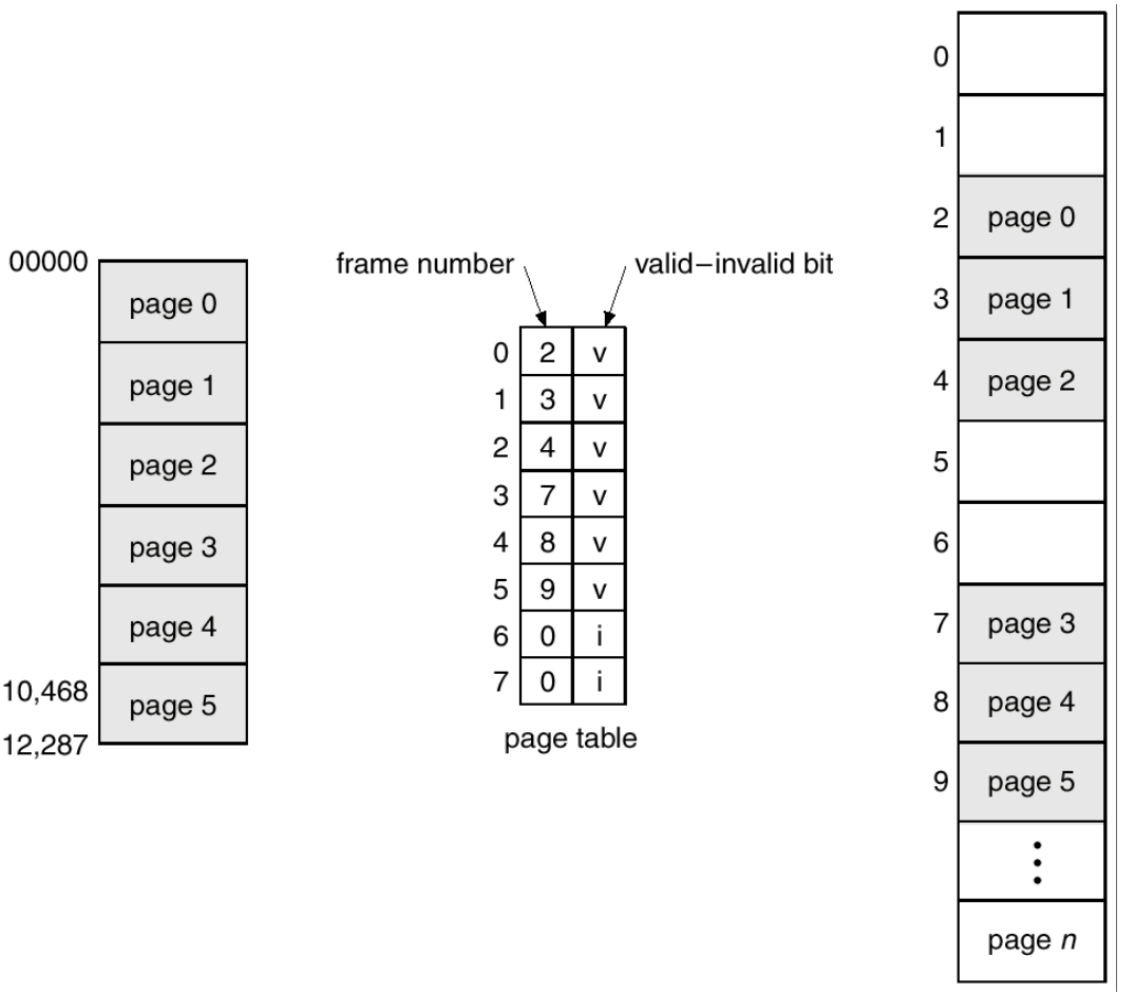
- P_1은 outer page table의 index이고
- P_2는 outer page table의 page에서의 변위(displacement)

Multilevel Paging and Performance

- Address space가 더 커지면 다단계 페이지 테이블 필요

- 각 단계 페이지 테이블이 메모리에 존재하므로 logical address의 physical address 변환에 더 많은 메모리 접근 필요
- TLB를 통해 메모리 접근 시간을 줄일 수 있음
- 4단계 체이지 테이블을 사용하는 경우
  - 메모리 접근 시간이 100ns, TLB 접근 시간이 20ns이고
  - TLB hit ratio가 98%인 경우
    effective memory access time = 0.98 × 120 + 0.02 × 520
    = 128 nanoseconds
    결과적으로 주소변환을 위해 28ns만 소요

Valid (v) / Invalid (i) Bit in a Page Table



Memory protection

- Page table의 각 entry마다 아래의 bit를 둔다
  - Protection bit
    - page에 대한 접근 권한 (read/write/read-only)
  - Valid-invalid bit
    - “valid”는 해당 주소의 frame에 그 프로세스를 구성하는 유효한 내용이 있음을 뜻함 (접근 허용)
    - “invalid”는 해당 주소의 frame에 유효한 내용이 없음*을 뜻함(접근 불허)

  * i) 프로세스가 그 주소 부분을 사용하지 않는 경우
    ii) 해당 페이지가 메모리에 올라와 있지 않고 swap area에 있는 경우