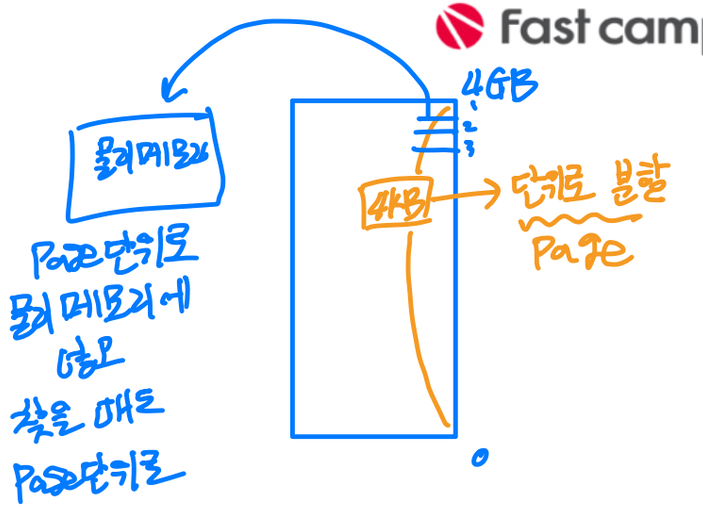
가상메모리 system에서
가장 많이 쓰는 메커니즘

페이징 시스템



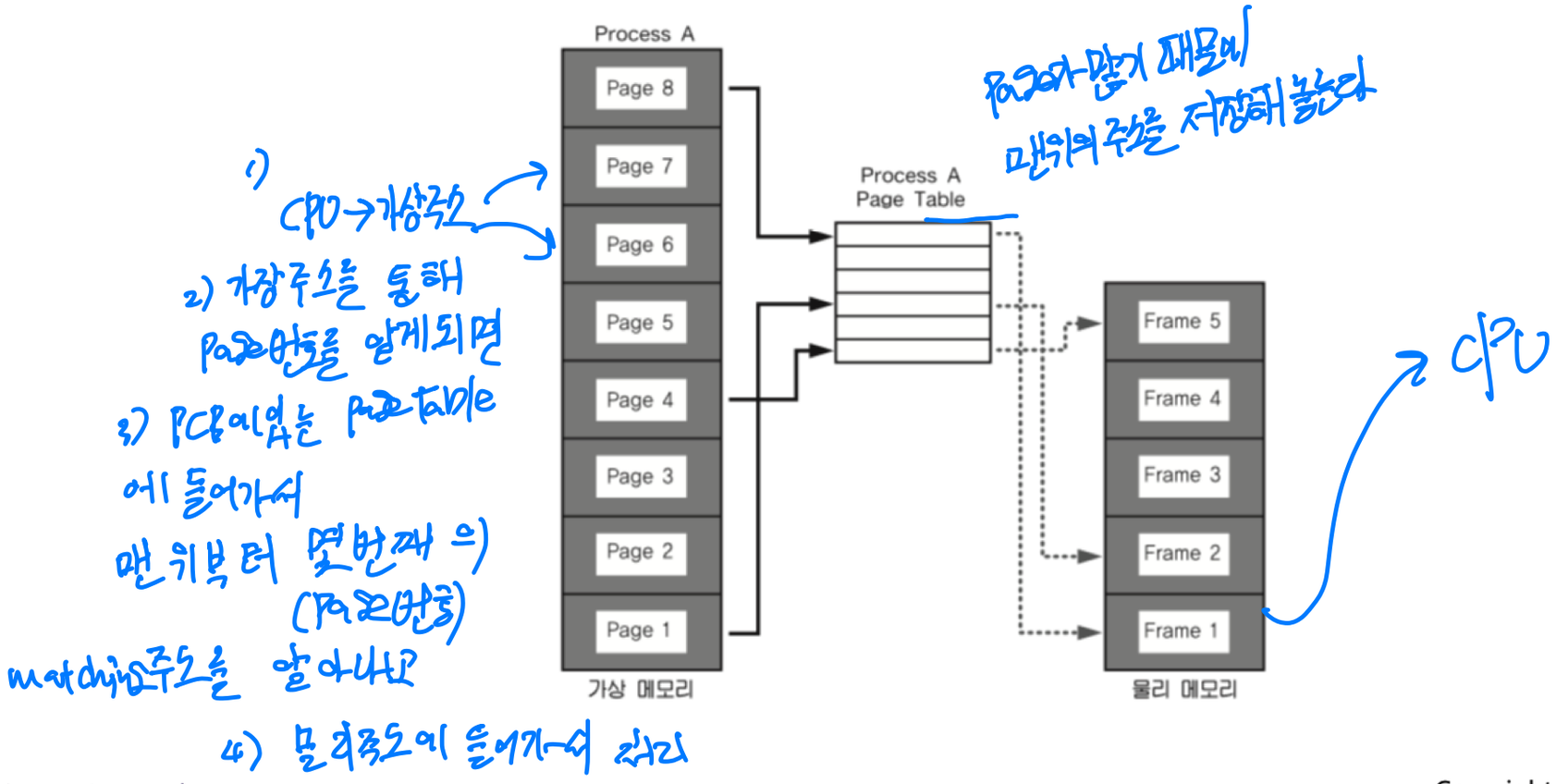
페이징 시스템(paging system)

- 페이징(paging) 개념
  - 크기가 동일한 페이지로 가상 주소 공간과 이에 매칭하는 물리 주소 공간을 관리
  - 하드웨어 지원이 필요
    - 예) Intel x86 시스템(32bit)에서는 4KB, 2MB, 1GB 지원
  - 리눅스에서는 4KB로 paging
  - 페이지 번호를 기반으로 가상 주소/물리 주소 매핑 정보를 기록/사용

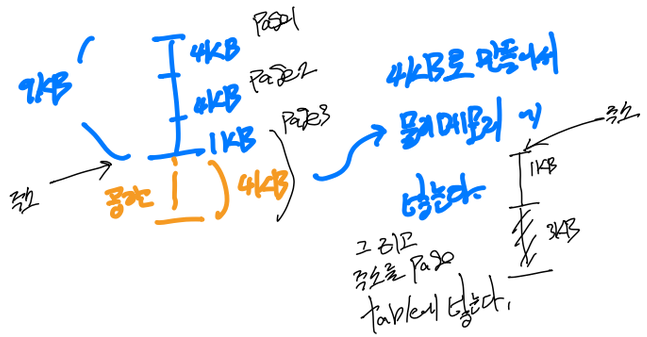
페이징 시스템(paging system)

> 실질적인 예를 기반으로 페이징 시스템에 대해 알아보겠습니다.

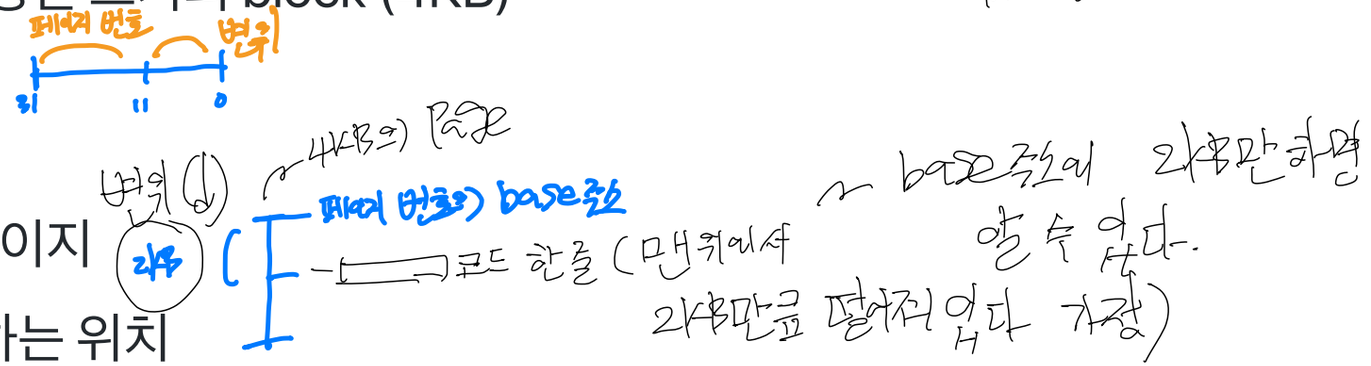
- 프로세스(4GB)의 PCB에 Page Table 구조체를 가리키는 주소가 들어 있음
- Page Table에는 가상 주소와 물리 주소간 매핑 정보가 있음



페이징 시스템 구조

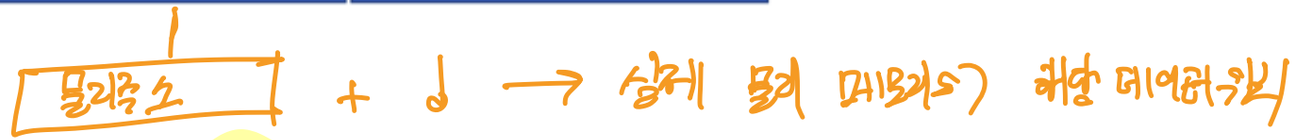


- page 또는 page frame: 고정된 크기의 block (4KB)
- paging system
  - 가상 주소 v = (p, d)
    - p: 가상 메모리 페이지
    - d: p안에서 참조하는 위치 (변위)



가상 주소(Virtual Address) $v = (p, d)$

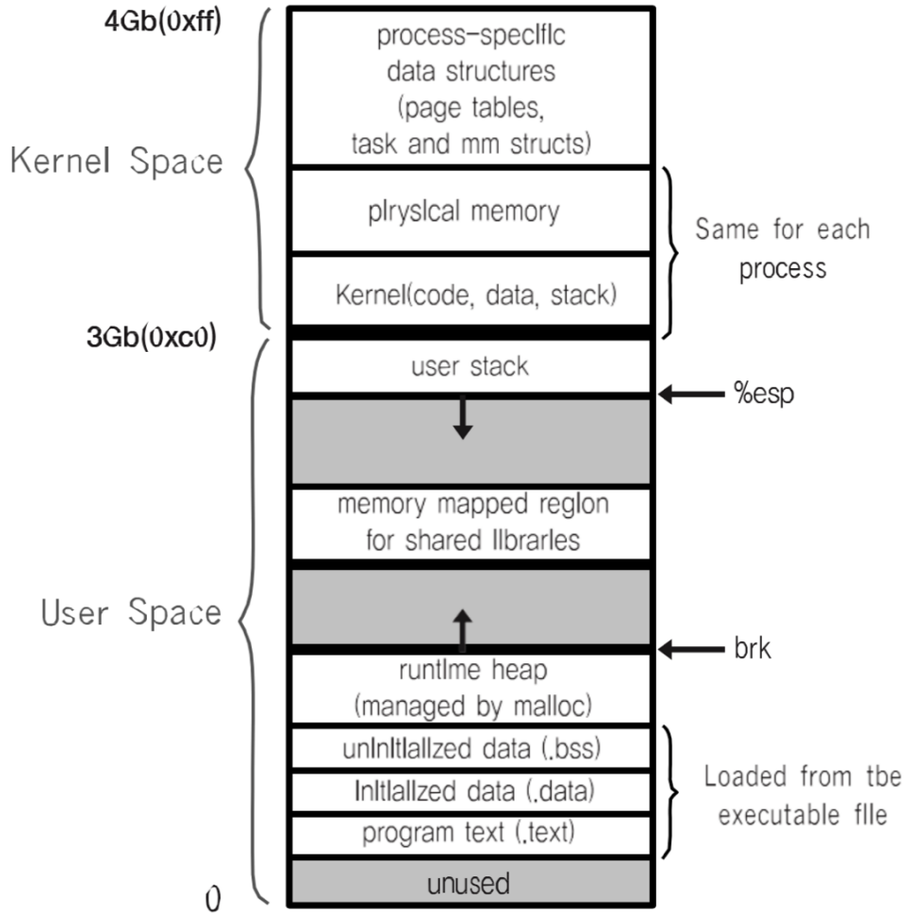
페이지 번호 p	변위(오프셋) d



- 페이지 크기가 4KB 예
  - 가상 주소의 0비트에서 11비트가 변위 (d)를 나타내고,
  - 12비트 이상이 페이지 번호가 될 수 있음

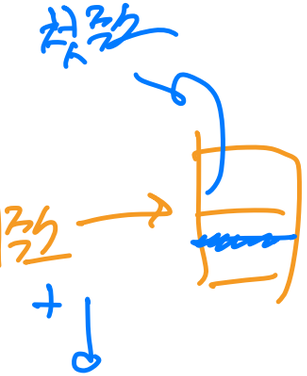
쉬었다 가기 - 모든 것은 결국 bit와 연결

- 프로세스가 4GB를 사용하는 이유 - 32bit 시스템에서 2의 32승이 4GB

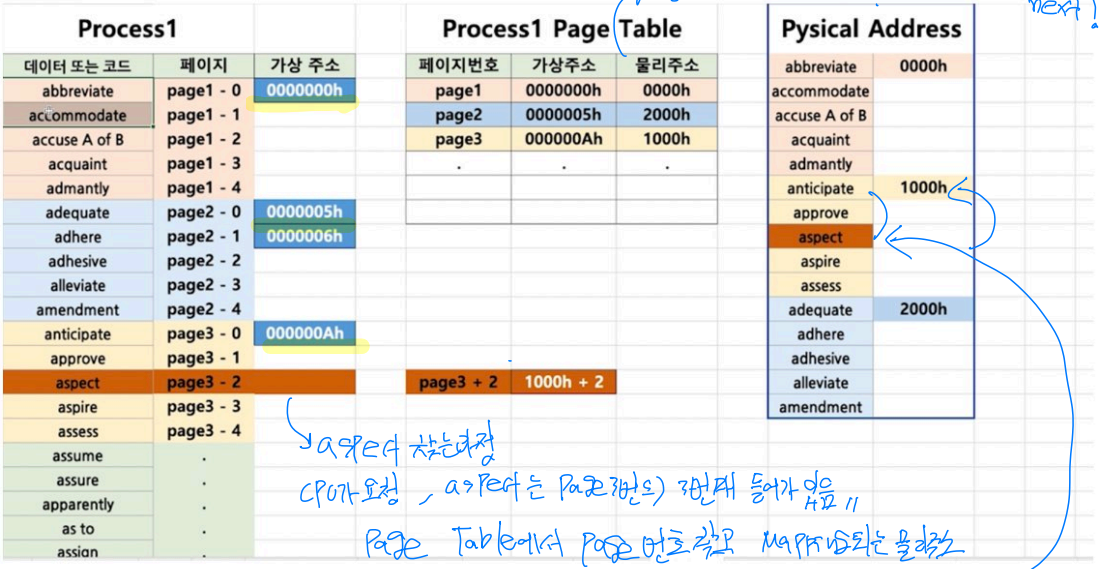


페이지 테이블(page table)

- page table
  - 물리 주소에 있는 페이지 번호와 해당 페이지의 첫 물리 주소 정보를 매핑한 표
  - 가상주소 v = (p, d) 라면
    - p: 페이지 번호
    - d: 페이지 처음부터 얼마 떨어진 위치인지
- paging system 동작
  - 해당 프로세스에서 특정 가상 주소 엑세스를 하려면
    - 해당 프로세스의 page table 에 해당 가상 주소가 포함된 page 번호가 있는지 확
    - page 번호가 있으면 이 page가 매핑된 첫 물리 주소를 알아내고(p')
    - p' + d 가 실제 물리 주소가 됨



> OS.xlsx --> PagingSystem, RealPagingSystem


Process1
데이터 또는 코드
페이지
가상 주소
abbreviate
page1 - 0
0000000h
accommodate
page1 - 1
accuse A of B
page1 - 2
acquaint
page1 - 3
admantly
page1 - 4
adequate
page2 - 0
0000005h
adhere
page2 - 1
0000006h
adhesive
page2 - 2
alleviate
page2 - 3
amendment
page2 - 4
anticipate
page3 - 0
000000Ah
approve
page3 - 1
aspect
page3 - 2
aspire
page3 - 3
assess
page3 - 4
assume
assure
apparently
as to
assign
Process1 Page Table
현재 저장되어있는 !!
페이지번호
가상주소
물리주소
page1
0000000h
0000h
page2
0000005h
2000h
page3
000000Ah
1000h
page3 + 2
1000h + 2
Pysical Address
abbreviate
0000h
accommodate
accuse A of B
acquaint
admantly
anticipate
1000h
approve
aspect
aspire
assess
adequate
2000h
adhere
adhesive
alleviate
amendment
next !
aspect 찾는과정
CPU가요청 , aspect는 Page2번의 3번째 들어가 있음 !!
Page Table에서 Page번호찾고 Mapping되는 물리주소
→ 1000h + 2

메모리에 모든 데이터가 들어갈 필요 없으므로
물리주소가 있는지 없는지를 나타내는 비트 정보가 있다.
v : valid
i : invalied

Process1

데이터 또는 코드	페이지	가상 주소
abbreviate	page1 - 0	0000000h
accommodate	page1 - 1	
accuse A of B	page1 - 2	
acquaint	page1 - 3	
admantly	page1 - 4	
adequate	page2 - 0	0000005h
adhere	page2 - 1	0000006h
adhesive	page2 - 2	
alleviate	page2 - 3	
amendment	page2 - 4	
anticipate	page3 - 0	000000Ah
approve	page3 - 1	
aspect	page3 - 2	
aspire	page3 - 3	
assess	page3 - 4	
assume	.	
assure	.	
apparently	.	
as to	.	
assign	.	

Process1 Page Table

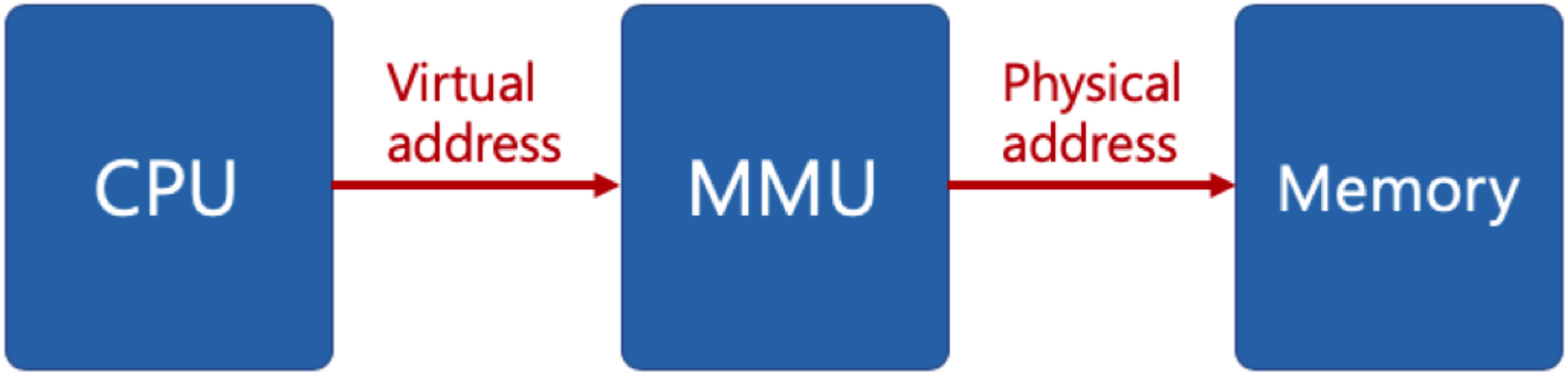
페이지번호	가상주소	물리주소	valid-invalid bit
page1	0000000h	0000h	v
page2	0000005h	~~2000h~~	i
page3	000000Ah	~~1000h~~	i
.	.	.	
page3 + 2	1000h + 2	?	

Pysical Address

abbreviate	0000h
accommodate	
accuse A of B	
acquaint	
admantly	

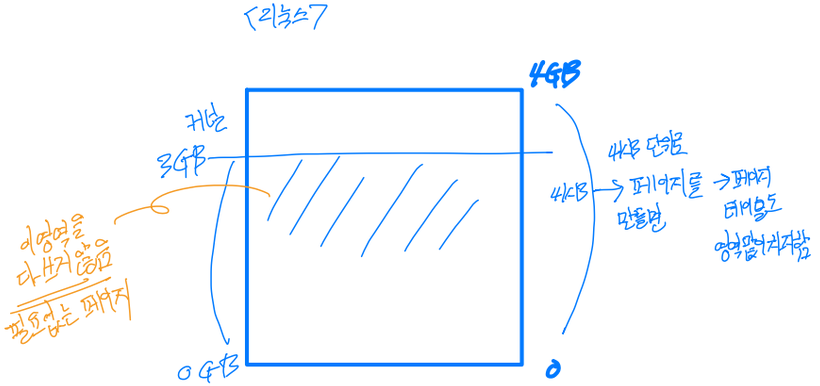
페이징 시스템과 MMU(컴퓨터 구조)

- CPU는 가상 주소 접근시
  - MMU 하드웨어 장치를 통해 물리 메모리 접근 (가상주소 → 물리 메모리로 변환)



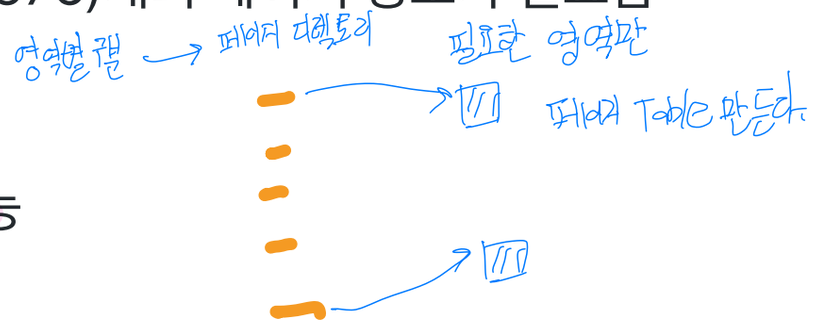
- 프로세스 생성시, 페이지 테이블 정보 생성
  - PCB등에서 해당 페이지 테이블 접근 가능하고, 관련 정보는 물리 메모리에 적재
  - 프로세스 구동시, 해당 페이지 테이블 base 주소가 별도 레지스터에 저장(CR3)
  - CPU가 가상 주소 접근시, MMU가 페이지 테이블 base 주소를 접근해서,물리 주소를 가져옴

Page table은 Default로 물리 메모리에 들어가있다.
↳ Page table을 물리 메모리에 적재함과 동시에,
CR3 레지스터를 가져와서



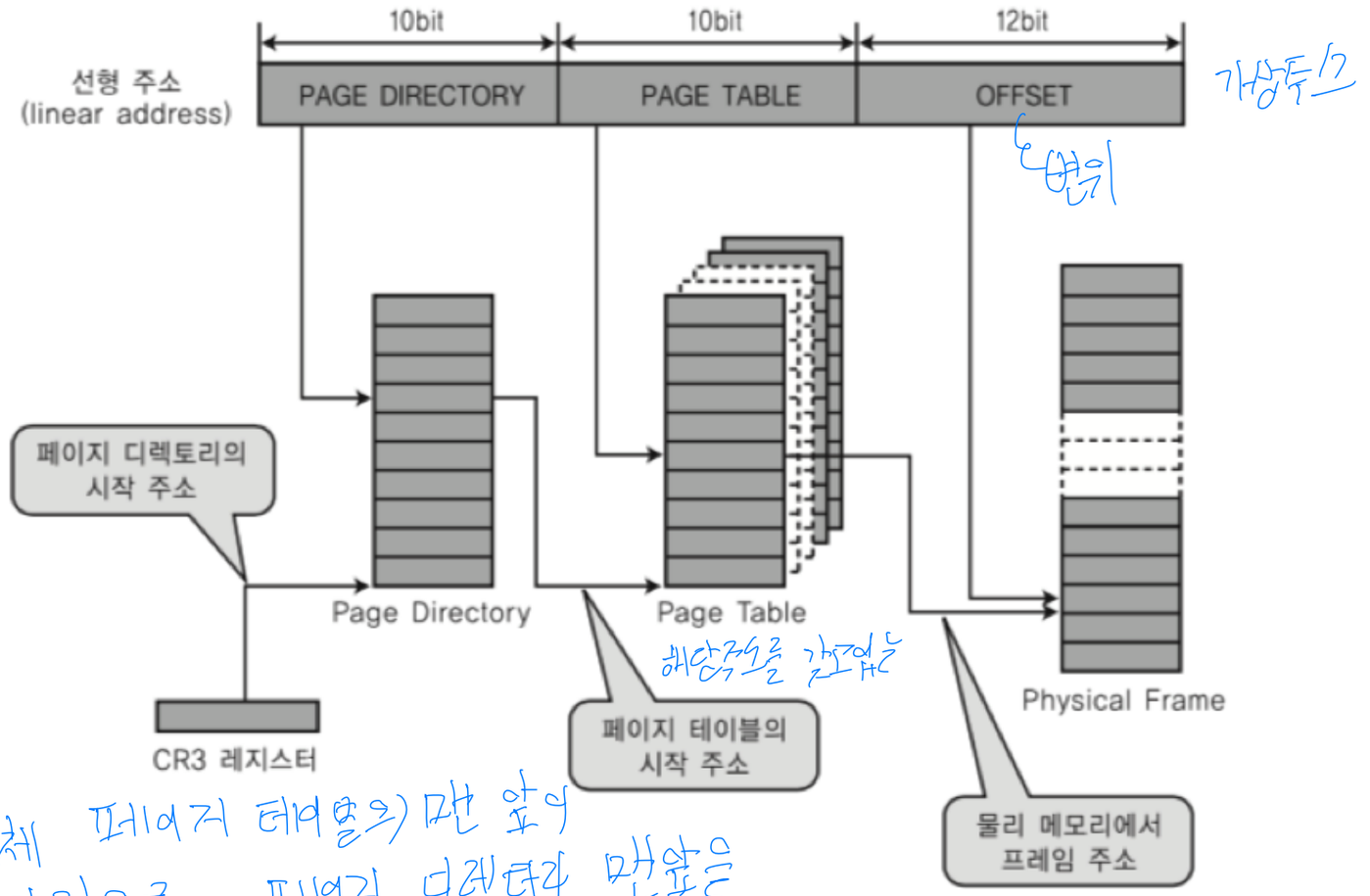
다중 단계 페이징 시스템

- 32bit 시스템에서 4KB 페이지를 위한 페이징 시스템은
  - 하위 12bit는 오프셋
  - 상위 20bit가 페이징 번호이므로, 2의 20승(1048576)개의 페이지 정보가 필요함
- 페이징 정보를 단계를 나누어 생성
  - 필요없는 페이지는 생성하지 않으면, 공간 절약 가능



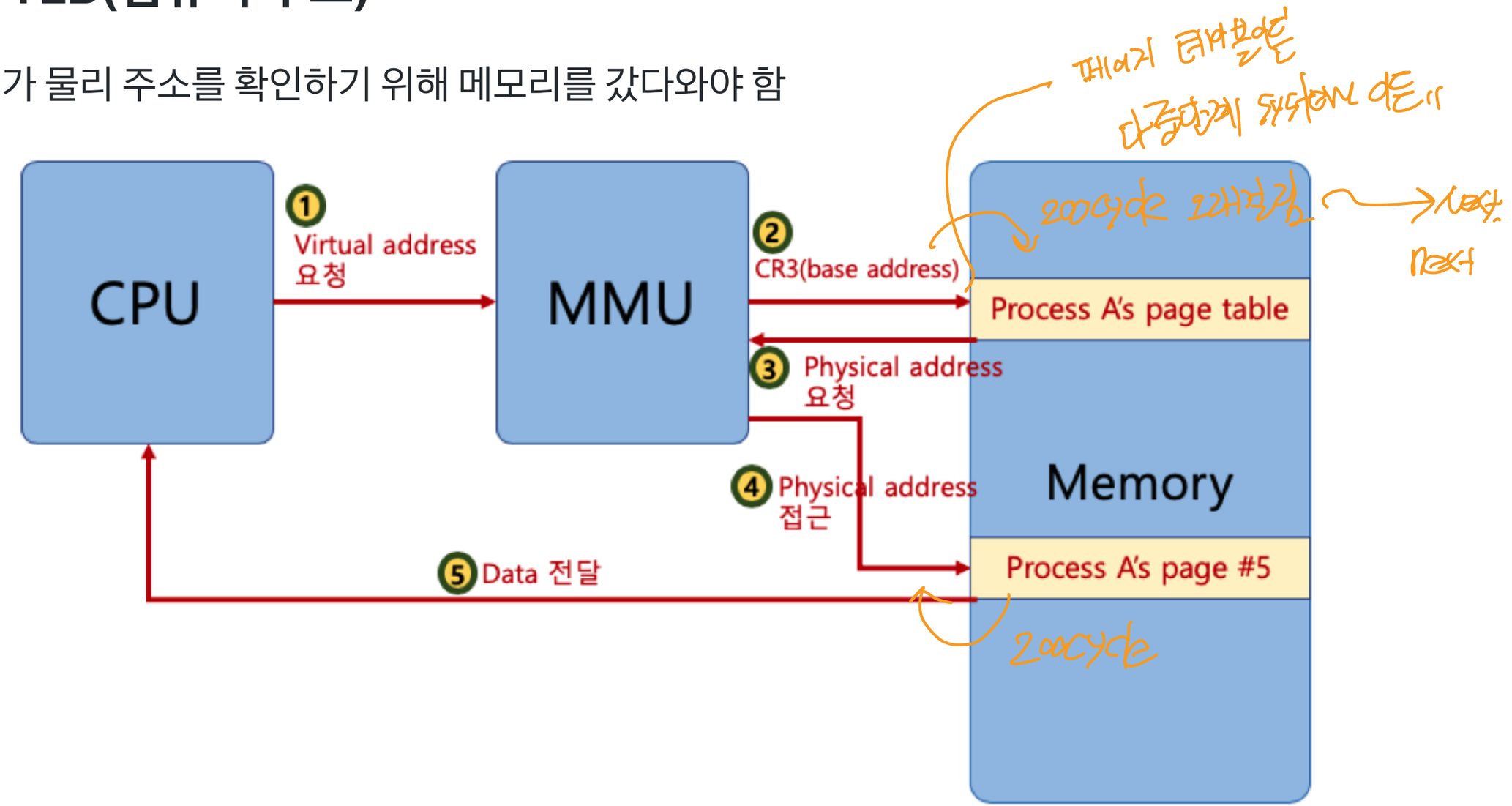
다중 단계 페이징 시스템

- 페이지 번호를 나타내는 bit를 구분해서, 단계를 나눔 (리눅스는 3단계, 최근 4단계)

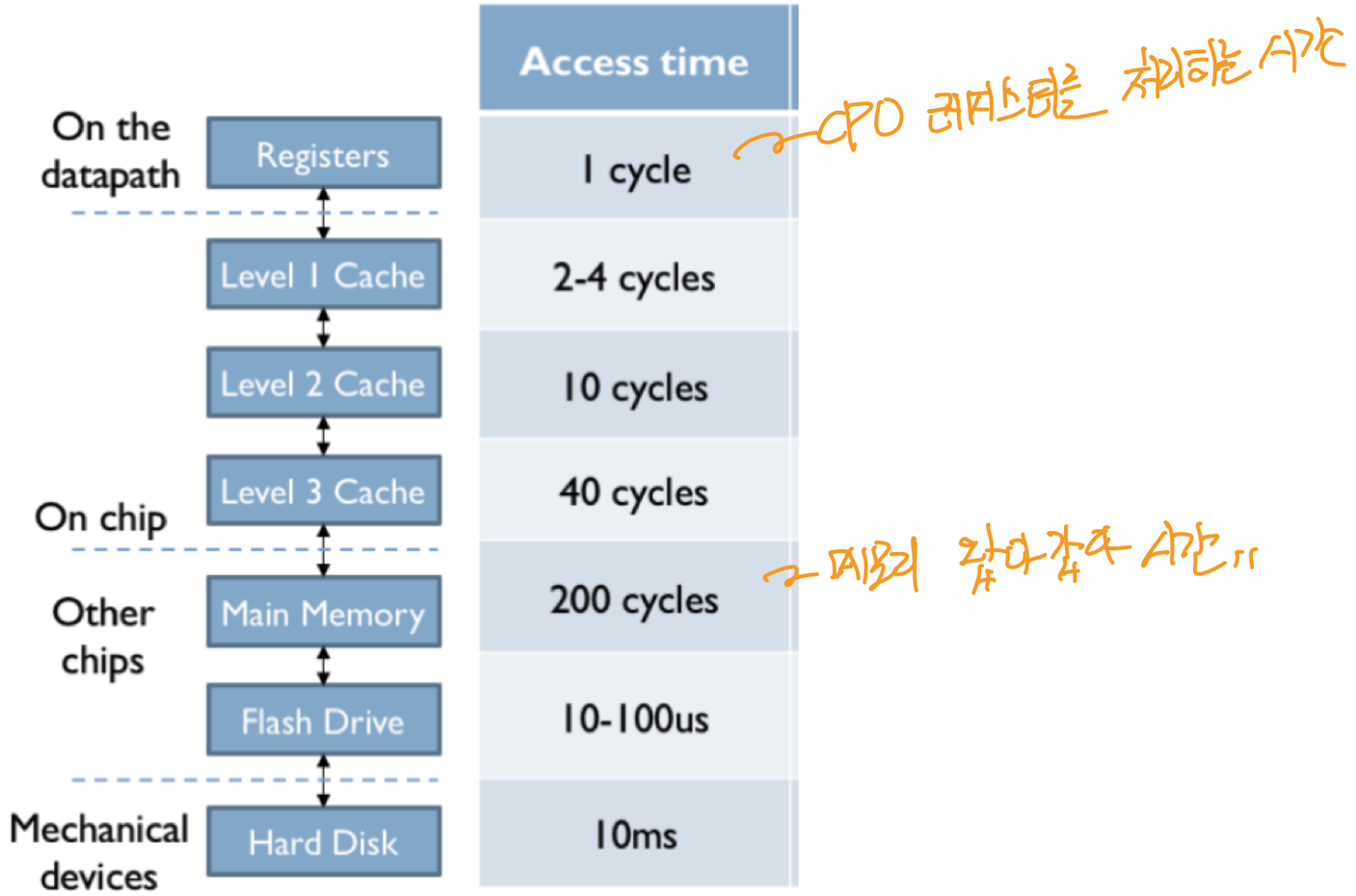


MMU와 TLB(컴퓨터 구조)

- MMU가 물리 주소를 확인하기 위해 메모리를 갔다와야 함



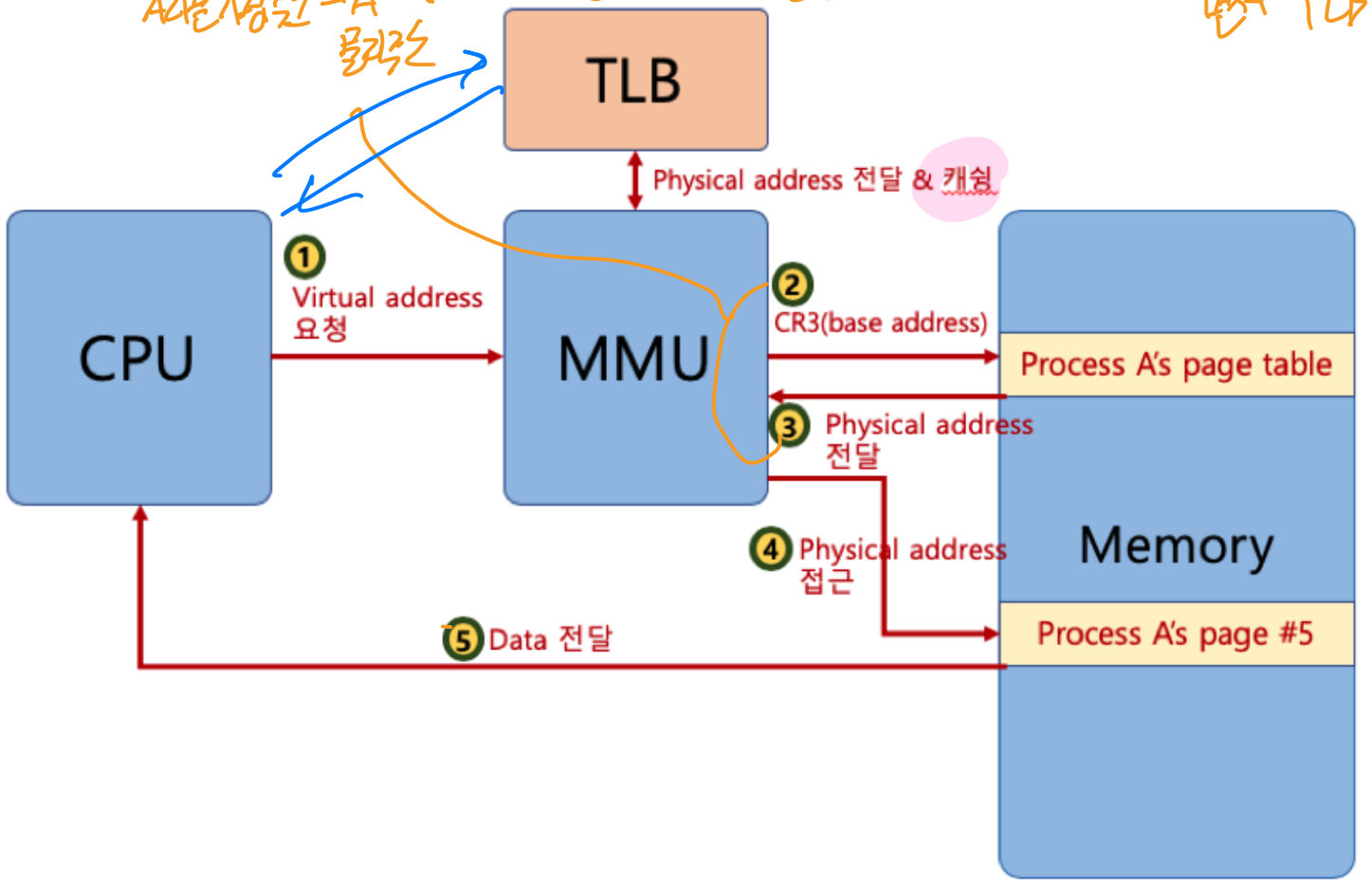
메모리 계층 - 컴퓨터 구조 복습



출처: http://computationstructures.org/lectures/caches/caches.html

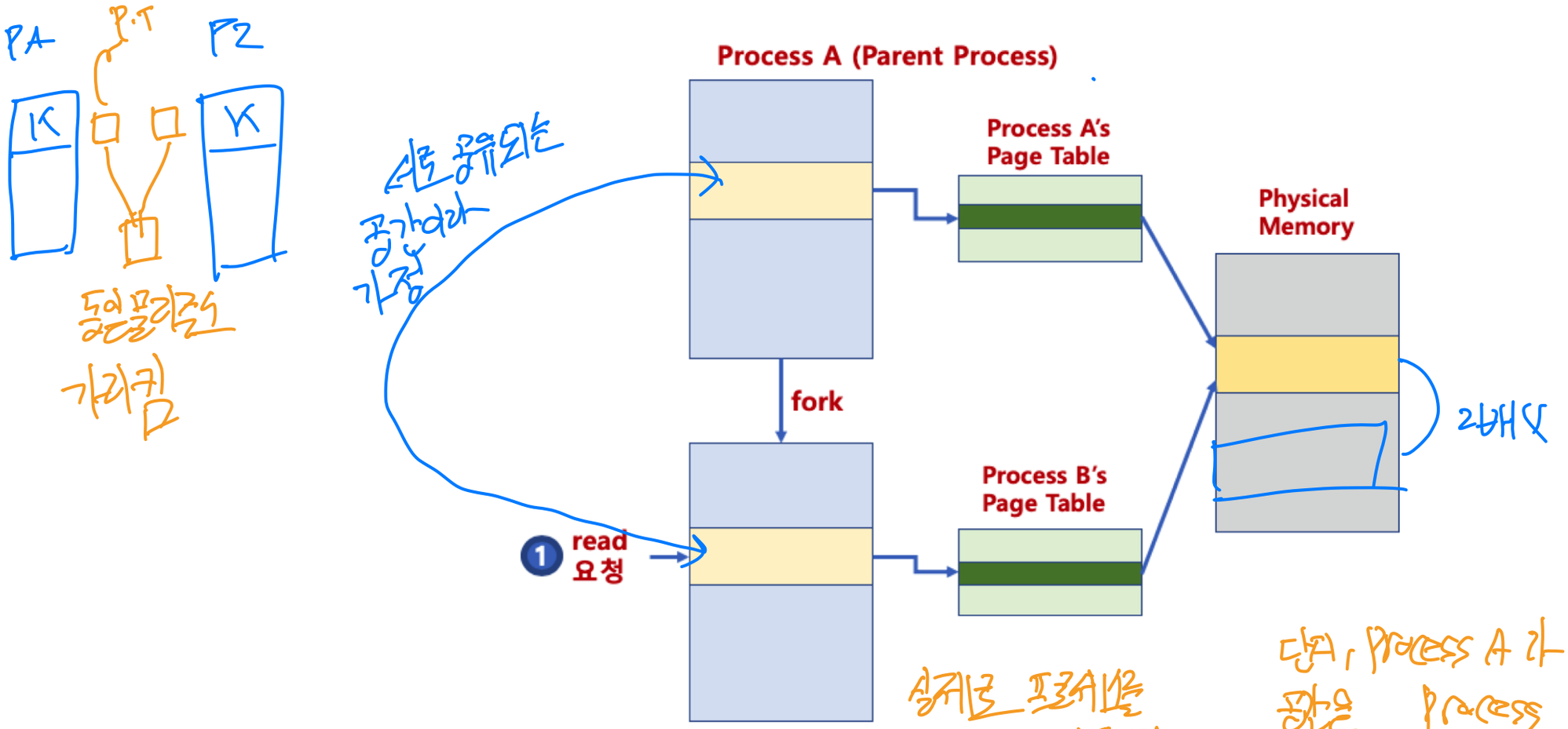
MMU와 TLB(컴퓨터 구조)

- TLB(Translation Lookaside Buffer): 페이지 정보 캐쉬



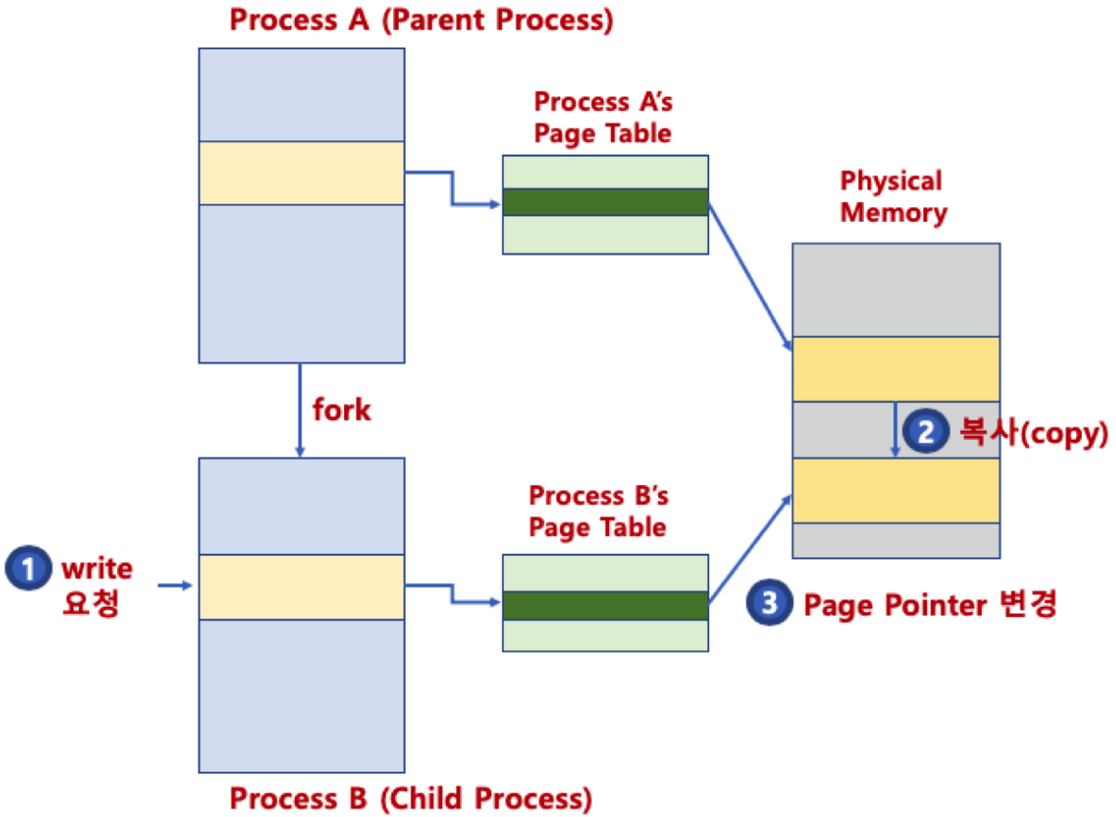
페이징 시스템과 공유 메모리

- 프로세스간 동일한 물리 주소를 가리킬 수 있음 (공간 절약, 메모리 할당 시간 절약)



페이징 시스템과 공유 메모리

- 물리 주소 데이터 변경시
  - 물리 주소에 데이터 수정 시도시, 물리 주소를 복사할 수 있음 (copy-on-write)



*프로세스 생성 시간 ↓
* 커널, 공유메모리는 물리 메모리 공간 공유 가능
반드시 P.T에 해당 Page의 물리 주소만 업데이트 시키면 됨 - 관점에서
- 프로

요구 페이징 (Demand Paging 또는 Demanded Paging)

- 프로세스 모든 데이터를 메모리로 적재하지 않고, 실행 중 필요한 시점에서만 메모리로 적재함
  - 선행 페이징(anticipatory paging 또는 prepaging)의 반대 개념: 미리 프로세스 관련 모든 데이터를 메모리에 올려놓고 실행하는 개념
  - 더 이상 필요하지 않은 페이지 프레임은 다시 저장매체에 저장 (**페이지 교체 알고리즘 필요**)

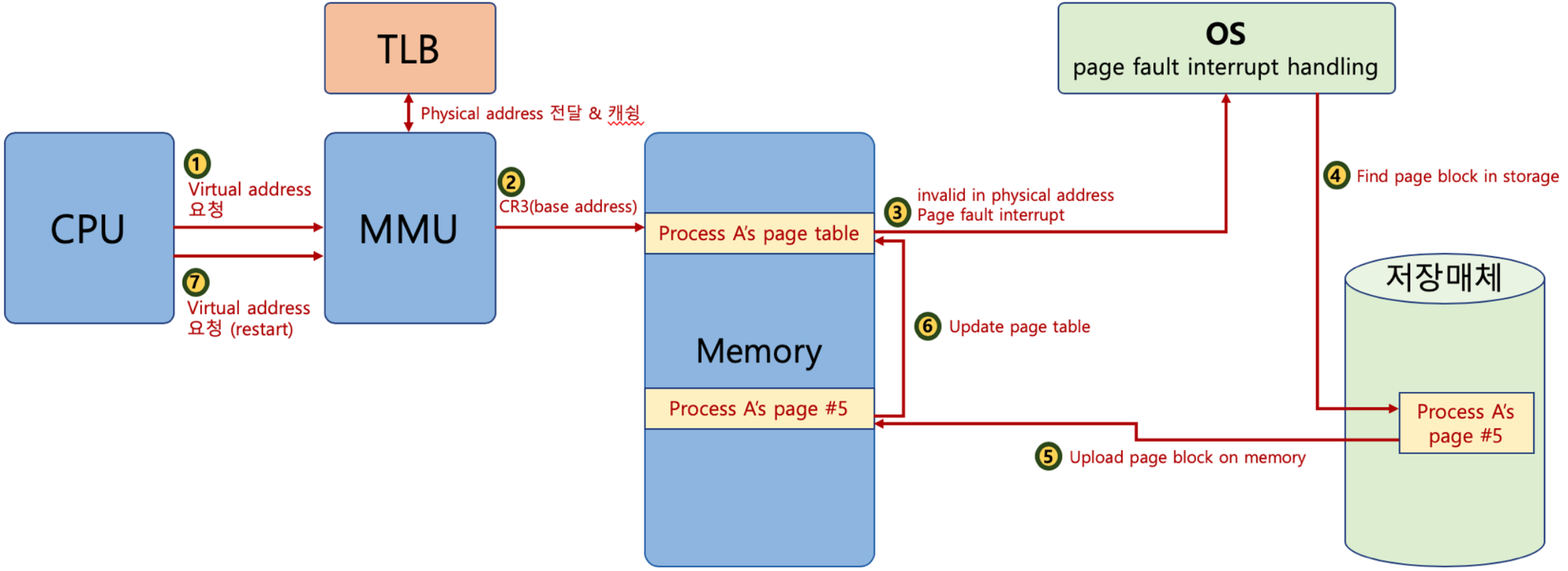
> OS.xlsx --> DemandPaging, RealDemandPaging

페이지 폴트 (page fault)

- 어떤 페이지가 실제 물리 메모리에 없을 때 일어나는 인터럽트
- 운영체제가 page fault가 일어나면, 해당 페이지를 물리 메모리에 올림

Fast campus

페이지 폴트와 인터럽트



생각해보기

- 페이지 폴트가 자주 일어나면?
  - 실행되기 전에, 해당 페이지를 물리 메모리에 올려야 함
    - 시간이 오래 걸림
- 페이지 폴트가 안 일어나게 하려면?
  - 향후 실행/참조될 코드/데이터를 미리 물리 메모리에 올리면 됨
    - 앞으로 있을 일을 예측해야 함 - 신의 영역