프로세스가 실행될때 디스크에 있는 실행파일이 메인 메모리로 올라간다.

이때 프로세스의 모든 정보가 메인메모리에 올려지는것이 아니고

일부분만 메모리에 올라가는데 나머지는 디스크에 존재하는 swap 영역에 존재한다.

이 swap영역에 존재하는게 가상메모리이다.

프로세스당 보유한 메모리 영역은

code 영역,data영역,stack영역,heap영역,커널영역 이렇게 있다.

이러한 주소 공간을 가상 메모리 라고 하고 disk에 존재한다.

cpu에서 가상메모리 주소를 호출하면 mmu에서 번역을 돌리는데

이때 참조하는 테이블이 page tbl 이다

이 pagetbl에는 가상페이지 넘버와 물리페이지 넘버가 매핑되어있는데

만약 찾는 가상메모리주소가 이 page tbl에 없다면 valid bit=0 으로

disk의 가상메모리로가서 있는지 찾아보고 page tbl에 등록하고

main memory 에 자주 안쓰는 정보와 swap 한다.

swap된 데이터는 가상테이블로 내려간다.

만약 page hit 성공적으로 매핑된 주소를 찾았다면

메인메모리에서 해당 데이터를 읽어온다.

그런데 hit 했을 때 메모리를 두번 접근하게 되는데

이는 page table base register를 이용해서 물리 주소를 얻어오고

그 물리주소로 memory에서 값을 읽어 cpu에 반환하게 된다.

이를 해결하기 위해 translation 용 cache를 다는데 TLB라고 한다.

가상페이지와 물리 페이지 간의 주소 번역을 해준다.

이걸 사용하면 tlb에서 찾고자 하는 vpn이 존재하면

memory access 하지 않고 tlb가 pte를 반환.

이 값을 토대로 memory access 하여 값을 cpu에 전달할 수 있다.

요약

1.cpu가 virtual addrass -> mmu에 전송

2.mmu 는 va와 매핑되는 Page Table Entry Address 를 보낸다.

3.main memory는 page table entry 정보를 리턴해준다

4.mmu 는 mainmemory에 물리메모리 주소를 전송

5.접근한 데이터를 cpu에 전송한다.

mmu 입장에서 mainmemory 접근이 두번됨.

tlb cache 사용 mainmemory 접근 1번으로 줄임.

\* page table 은 main memory에 있다.

mmu에서는 가상 주소로 매핑된 엔트리 주소를 가지고

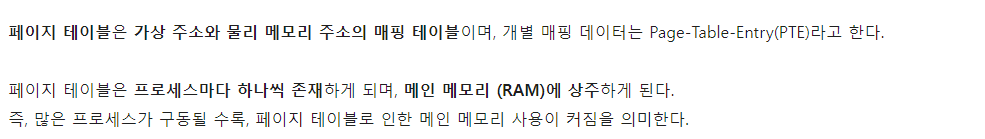
mainmemory의 pagetbl에 해당 entery를 반환받아서

hit인지 fault인지 판단한다.

메모리 관련 공부 참고 블로그

https://talkingaboutme.tistory.com/entry/Memory-Address-Translation?category=608888

메모리 할당시 object 크기가 page 단위보다 작기 때문에 page size 단위로 heap에 할당, 해제한다.



http://egloos.zum.com/sweeper/v/2988646