Cpl이랑 dpl비교 맞으면 실행

* Pl과 cpl 비교
  + 맞으면 시행
  + 안맞으면 커널로 바꾸고 실행

IDT에서 불러옴

CPL을 cs에서 불러옴

DPL을 가져오고 CPL <= DPL인지 체크

Cpu 내부 레지스터인 esp와 ss에 PL < CPL 인 경우에만 저장(???)

ss, esp 를 테스크 세그멘트 디스크립터러로부터 로드하기

ss 푸시

esp 푸시

eflags 푸시

cs 푸시

eip 푸시

eflags에서 제어 플레그, 시스템 플레그, 상태 플레그 등이 있음

그런데 왜 이런 단편적인 비트 정보를 가지고 잇는 건지를 모르겠음

제어 플레그 같은 경우 단지 위의 주소로 이동하는 지만 알지 얼마만큼 가는지도 모르면 의미가 없지 않나요?

Trap == 인터럽트?

**Systems calls, exceptions, and interrupts**

이 세가지가 되도록 os가 해야될 일

* 시스템은 프로세서가 다시 잘 돌도록 하기 위해서 레지스터를 저장
* 커널이 실행할 공간을 골라야됨
* 커널은 이벤트에 대해서 찾아와야 한다(예시 시스콜 인자)
* 시스템은 유저와 커널로부터 아이솔레이션을 유지해야 된다.

이걸 알기 위해서는 os는 하드웨어가 어떻게 핸들링하는지 잘 알아야 한다.

이 세가지는 한가지 메커니즘으로 동작함

기본 계획은 다음과 같습니다. 인터럽트는 정상 프로세서 루프를 중지시키고

인터럽트 핸들러라는 새로운 시퀀스를 실행하기 시작합니다. 인터럽트를 시작하기 전에 처리기에서 프로세서는 레지스터를 저장하므로 운영 체제가 해당 레지스터를 복원할 수 있습니다. 그것들이 인터럽트로부터 돌아왔을 때. 에서 전환에 도전 인터럽트 처리기는 프로세서가 사용자 모드에서 커널로 전환해야한다는 것입니다

모드, 그리고 뒤로.

트랩은 전류에 의해 발생함

시스콜과 인터럽트가 같이 오면?

**Code: Assembly trap handlers**

Tvinit 코드에서 IDT의 초기화를 진행

커널은 시스템 호출 게이트 권한을 DPL\_USER로 설정 -> 이러면 무조건 통과잖아?

사용자 프로그램은 멍시적 int 명령어로 트랩을 생성합니다.

예외처리가 발생할 경우 vector13으로 이동하여 protection exception이 발생한다.

%ss, %esp 에 있던 old 유저의 것을 새 스택에 넣는다. (새 스택이 뭐지?)

만약 새 프로세스가 커널 모드로 실행할 경우 이걸 할 필요 없다.

질문:

커널은 시스템 호출 게이트 권한을 DPL\_USER로 설정 -> 이러면 무조건 통과잖아?

개념

Idt(interrupt descriptor Table) :