운영체제💻

이중모드 하드웨어 보호

> 이중모드?

`1` 한 컴퓨터를 여러 사람이 동시에 사용하는 환경이 있을 수 있다. EX) 수강신청

- 서버 컴퓨터는 하나인데 접속하는 사람은 매우많음.
- 또는 한 사람이 여러 개의 프로그램을 동시에 사용하는 상황도 있을 수 있음.
- 이러한 환경에서 누군가의 고의 혹은 실수로 인해 전체 프로그램에 영향을 줄 수 있음.
    - 특히 서버 컴퓨터가 그렇다고 할 수 있는데, 누군가 악의적으로 서버를 다운시키면 다른 사람들도 모두 못 쓰게 되는 경우. 대표적으로 => STOP(cpu 멈춰!) , HALT ,RESET 명령등이 있음. 중지된 cpu를 다시 깨우려면 껐다 키는 방법밖에 없음.
- 그러므로 사용자 프로그램은 STOP 등 `치명적인 명령들을 내릴 수 없게 해야한다`
    - 사용자 모드 vs 관리자 모드가 그래서 존재하는것.
    - 이것이 이중 모드의 모티브!
    - 관리자 모드는 `시스템 모드` = `모니터 모드` = `특권 모드` 라고 하기도 함.
    - Supervisor, system, monitor, priviliged mode
- 특권 명령 (privilieged instructions)
    - STOP, HALT, RESET, SET_TIMER, SET_HW

특권 명령은 관리자 모드에서만 내릴 수 있는 명령들이다. 타이머 같은 경우, 일반 유저가 막 서버 시간을 자기멋대로 바꿔버리면 안되니까?, SET_HW => 하드웨어의 값을 바꾸는 것.

그렇다면, `이중모드 (dual mode)` 는 어떻게 만드는가?

우선, cpu 안에는 `레지스터` , ALU(산술 연산장치), 제어유닛 등이 있는데,

이중 레지스터는 비트들의 모음인데, (32 비트 컴퓨터라면 비트가 32개)
이 비트를 활용해서 cpu의 상태를 나타내는 비트로 쓸 수 있다.

구체적으로, 이를 `FLAG` 라고 하는데? (어떠한 사건이 일어나면 깃발을 들고 경고한다고 생각)

- carry (자리수가 올라갔을때 씀)
- negative (연산 결과가 음수일 경우)
- zero (연산 결과가 0)
- overflow (연산 결과가 자리범위를 넘어버린 경우)

뭐 특정 비트가 1이면 캐리가 발생했다든가, 이런 식으로 활용할 수 있음. 컴퓨터 구조 시간에 이런 4가지 정도의 flag 들을 배우게 되는데,
OS 에서는 여기 예를들면 비트를 하나 더 할애해서 이중 모드를 나타내는 비트를 추가하는것.

가령, monitor 비트가 1이면 관리자 모드다 뭐 이런식으로 이중 모드를 나타 낼 수 있음.

=> 파워를 키면 부팅이 되고 OS를 주기억장치 영역으로 올리는 등 이 과정에서 모니터 비트는 1을 유지하다가, 게임 프로그램등을 실행할 때에는 OS가 유저모드로 전환시켜줌.

`정리`

- 이중모드=> 레지스터에 모드를 나타내는 플래그를 활용해 on/off 한다.
- OS 서비스 실행될 동안은 `관리자 모드`

- 사용자 프로그램이 실행될 때에는 `사용자 모드`
- 하드웨어 / 소프트웨어 인터럽트가 발생하면 `관리자 모드`
- 운영체제 서비스가 끝나면 다시 `사용자 모드`

게임 프로그램을 쓰다가, 게임 스코어를 하드디스크에 저장하고 싶다 (서버에 기록하고싶다) 라고 할 때, 하드디스크에 저장하는건 OS가 하는것. 게임 프로그램은 OS에게 게임 스코어를 저장해 달라! 라고 부탁을 하게 되고 (소프트웨어 인터럽트) 인터럽트가 걸리게 되면 cpu는 현재 하던일을 중지하고 OS 안에 들어있는 ISR로 점프해서 OS 안의 코드가 동작함.

=> 게임 프로그램이 직접 하드디스크에 접근할 수 없다는 뜻과 같은데, 만약 게임 프로그램이 직접 하드디스크에 접근 가능하게끔 해버리면, 하드 디스크 안에 있는 남의 파일도 뒤질 수 있게 된다는 문제점이 생김. (서버 컴퓨터에는 여러 사용자 파일들이 있으니까)

- 하나의 프로그램을 실행 중이더라도 모드 전환이 많이 일어날 수 있음.

만약, 일반 유저가 특권 명령을 내린다고 해보자. Cpu가 이 명령을 읽어와서 보니 monitor 비트가 꺼져있으면, cpu는 이 명령이 그래서 잘못 내려진 것으로 판단함 (내부 인터럽트가 발생했구나) => 그래서 cpu에셔 이 명령을 실행하지 않고 os의 잘못된 명령의 경우 작동하는 ISR로 점프하고 잘못된 명령을 내린 프로그램을 강제 종료해서 메모리에서 날려버림.

결국, `이중 모드는 보호(protection)와 관련이 있음`

(1) 입출력 장치 보호

- 프린터, 하드디스크 등 => 서버 컴퓨터를 경유하여 누군가 프린트 하고 있는데, 이걸 뭐 리셋시켜버린다든가 그러면 안되니까? 하드디스크에서 다른사람 파일 읽고 쓰면 안되니까?
    - 입출력 명령(IN and OUT)을 특권 명령으로 함. (OS만 이런 명령들 내릴 수 있도록)
    - 그럼 입출력을 하고 싶으면, 유저 모드에서 OS한테 소프트웨어 인터럽트 걸고, OS 안에 해당 인터럽트를 처리하는 루틴으로 간 다음 입출력 완료 후, 다시 사용자 모드로 돌아오는 식으로 하는 것.
    - 다른 사람 파일을 읽으려고 하는 상황을 생각해보면, 이걸 OS 한테 부탁해봤자 OS쪽에서 거부할 수 있게 됨. 곧, OS에게 소프트웨어 인터럽트 걸어봤자 OS의 ISR 에서 정당한 요청인가 확인하는 부분이 있기 때문임. (네이버 N드라이브나 이런데 올려도 다른 유저들이 내걸 뒤져볼 수 없는것)

(2) 메모리 보호

OS 가 상주하긴 하는데, 나머지 MM 영역에 다양한 사용자 프로그램이 올라와 있을텐데?

유저 프로그램 1이 다른 유저 프로그램 2를 읽으려고 하거나, 유저 프로그램 1이 OS의 ISR 코드를 바꿀 수 있거나(이게 해킹) 뭐 이러면 안되니까?
유저 1 프로그램이 돌때는 메모리 안에서 가동범위 영역이 유저 1 프로그램으로 한정되어야 한다는 의미.

그럼 이렇게 한정시키는건 어떻게 하나?

cpu 에서 메모리로 address bus가 가는데 (메모리의 어느 주소를 읽어오겠다) => 그 번지수에 해당하는 내용이 data bus로 오게 되는 구조이다. 애초에 address bus를 잘라버리는 방법이 있을 수 있는데, 애초에 address bus를 잘라버리면 자기 영역에도 못들어 간다는 단점이 생김.

그래서 address bus에 문지기 하나 둬서, 문지기가 유저 1에 할당된 주소로만 bus 보내도록 하는것.

`문지기의 정체` => 레지스터(극히 소량의 데이터나 처리중인 중간 결과를 일시적으로 기억해 두는 고속의 전용 영역)를 둬서 base, limit 설정하고 base ~ limit 사에 있는 주소를 요청할 때에만 통과 시켜주도록 함. 자기 범위를 넘어서는 번지를 읽으려고 하면 문지기가 cpu에게 인터럽트 신호 보내줌. 그럼 cpu에서 OS ISR로 가고, 잘못된 번지를 읽으려고 하는 애를 강제 종료시킴.

다른 사용자 또는 운영체제 영역 메모리에 접근 시도하는 것을 `Segment Violation` 이라고 함.

- MMU (Memory Management Unit = 문지기 이름) 을 둬서, 다른 메모리 영역에 대한 침범을 감시하도록 한다.
- MMU 설정(base limit 값을 바꾸거나)은 특권명령이다! => OS만 바꿀 수 있다.

(3) CPU 보호

- CPU 자체에 대한 공격이 들어올 수도 있으니까!