주변장치와 입출력 장치는 CPU나 메모리와 달리 인터럽트라는 메커니즘을 통해 관리된다.

그래서 인터럽트, 왜 하는거요?

그 이유는 입출력 연산이 CPU 명령 수행속도보다 현저히 느리기 때문이다.

운영체제를 악덕 사장님, CPU를 비싼 월급 주고 데려온 고오급 인력이라고 생각해보자. 악덕 사장 입장에서는 비싼돈 들여온 만큼 고오급 인력이 쉬지않고 일해서 돈값을 했으면 좋겠다고 생각할 것이다.

> 내 피같은 돈! 뽕을 뽑아먹겠어! *(고용노동부 국번없이 1350)*

그런데 아주 오래걸리는 입출력 연산을 CPU가 매번 기다린다면(월급루팡 한다면)...? 비싼 돈 주고 모셔온 CPU를 백분 활용하지 못해 운영체제 사장님은 환장할 지경.

CPU가 입출력 처리를 기다리며 쉬는 꼴을 못보는 사장님은 연산 결과가 나올 때 까지 다른 일을 시킨다. 우리 직원 뽕을 뽑아야하니까!

그리고 입출력 직원에게 자신의 업무가 완료되면 그때 CPU선배님에게 작업 완료를 알리라고 일러둔다. CPU가 다시 해당 작업도 이어서 할 수 있도록 한다.

여기서 입출력 직원이 CPU선배님에게 작업 완료를 알려주는 것이 **인터럽트** 이다!

인터럽트란?

CPU가 프로그램을 실행하고 있을 때, 입출력 하드웨어 등의 장치나 예외상황이 발생하여 처리가 필요할 경우에 "마이크로프로세서"에게 알려 처리할 수 있도록 하는 것을 말한다.

↳ cpu

인터럽트는 크게 ① 하드웨어 인터럽트와 ② 소프트웨어 인터럽트로 나뉜다.

① 하드웨어 인터럽트

하드웨어가 발생시키는 인터럽트로, (CPU가 아닌) "다른 하드웨어 장치"가 "cpu에 어떤 사실을 알려주거나 cpu 서비스를 요청해야 할 경우" 발생시킨다.

(=주변장치)

↳ 주변장치가 "내가 처리해야할 작업은 다 끝냈어요!"라는 메시지를 'IRQ Line'으로 날림
→ 운영체제가 해당 주변장치에 접근하여 주변장치의 결과물을 가져옴.

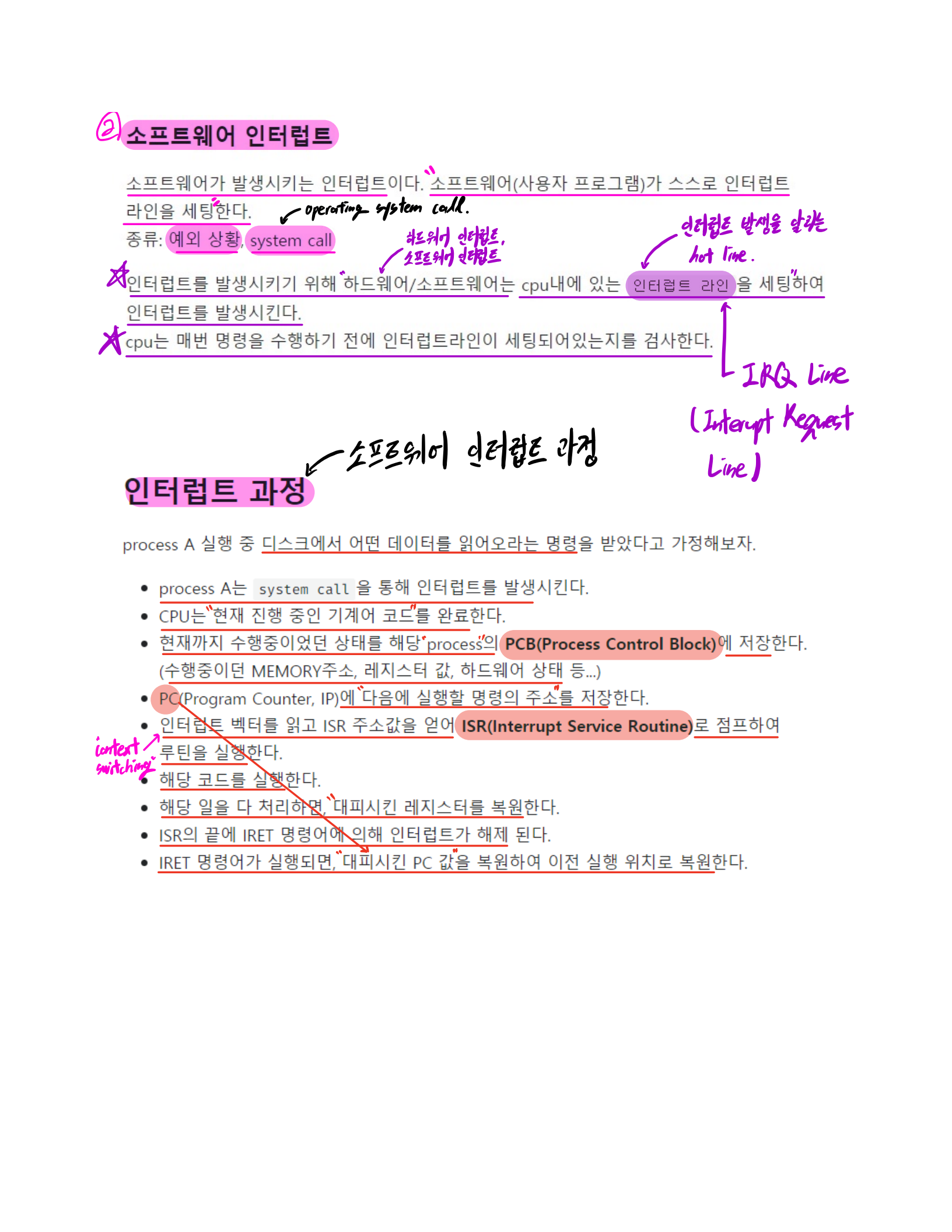
② 소프트웨어 인터럽트

소프트웨어가 발생시키는 인터럽트이다. 소프트웨어(사용자 프로그램)가 스스로 인터럽트 라인을 세팅한다.

종류: 예외 상황, system call

인터럽트를 발생시키기 위해 하드웨어/소프트웨어는 cpu내에 있는 `인터럽트 라인`을 세팅하여 인터럽트를 발생시킨다.

cpu는 매번 명령을 수행하기 전에 인터럽트라인이 세팅되어있는지를 검사한다.



인터럽트 과정

process A 실행 중 디스크에서 어떤 데이터를 읽어오라는 명령을 받았다고 가정해보자.

- process A는 `system call`을 통해 인터럽트를 발생시킨다.
- CPU는 현재 진행 중인 기계어 코드를 완료한다.
- 현재까지 수행중이었던 상태를 해당 process의 **PCB(Process Control Block)**에 저장한다. (수행중이던 MEMORY주소, 레지스터 값, 하드웨어 상태 등...)
- PC(Program Counter, IP)에 다음에 실행할 명령의 주소를 저장한다.
- 인터럽트 벡터를 읽고 ISR 주소값을 얻어 **ISR(Interrupt Service Routine)**로 점프하여 루틴을 실행한다.
- 해당 코드를 실행한다.
- 해당 일을 다 처리하면, 대피시킨 레지스터를 복원한다.
- ISR의 끝에 IRET 명령어에 의해 인터럽트가 해제 된다.
- IRET 명령어가 실행되면, 대피시킨 PC 값을 복원하여 이전 실행 위치로 복원한다.

인터럽트 발생 시 처리과정

▶ 인터럽트 동작 원리

1. 프로그램 실행을 중단합니다.
2. 현재의 프로그램 상태를 보존합니다.
3. 인터럽트 처리 루틴을 실행합니다.
4. 인터럽트 서비스 루틴을 실행합니다.
5. 인터럽트 요청 신호가 발생했을 때 보관한 PC의 값을 다시 PC에 저장합니다.
6. PC의 값을 이용하여 인터럽트 발생 이전에 수행중이던 프로그램을 계속 실행합니다.

인터럽트와 특권 명령

명령어의 종류

CPU가 수행하는 명령에는 `일반 명령`과 `특권 명령`이 있다.

일반 명령은 메모리에서 자료를 읽어오고, CPU에서 계산을 하는 등의 명령이고 모든 프로그램이 수행할 수 있는 명령이다.

특권 명령은 보안이 필요한 명령으로 입출력 장치, 타이머 등의 장치를 접근하는 명령이다. 특권 명령은 항상 `운영체제` 만이 수행할 수 있다.

kernel mode vs user mode

운영체제는 하드웨어적인 보안을 유지하기 위해 기본적으로 두가지 operation을 지원한다.

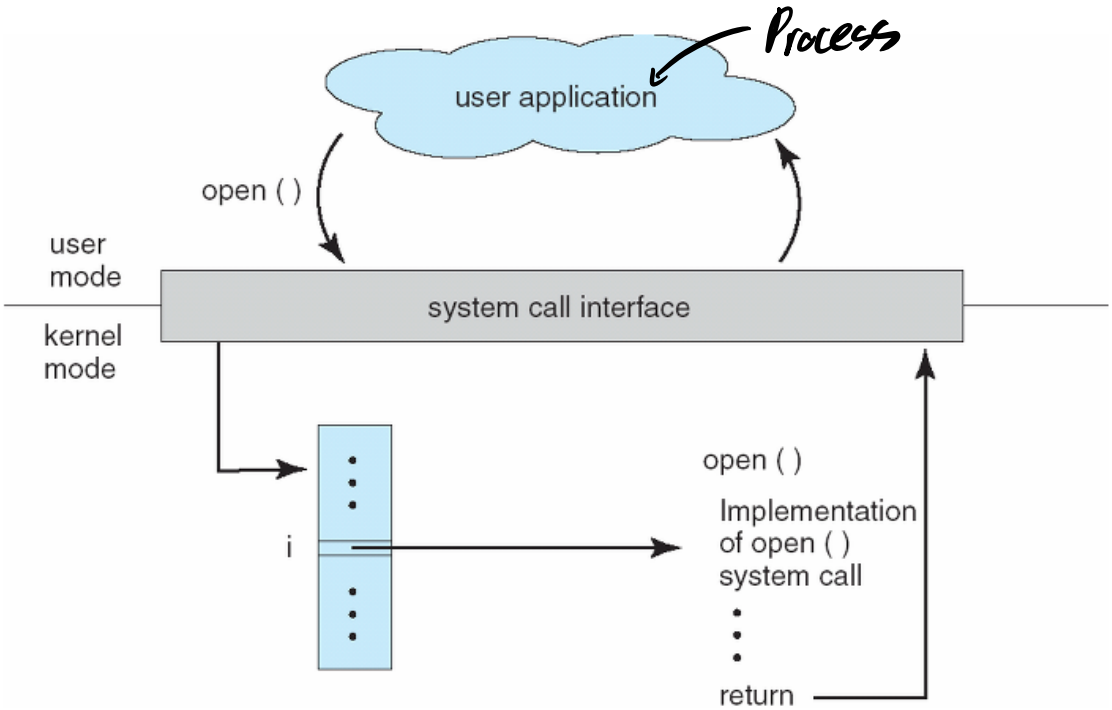
kernel mode는 운영체제가 CPU의 제어권을 가지고 명령을 수행하는 모드로 `일반 명령` 과 `특권 명령` 모두 수행할 수 있다.

하지만 **user mode**는 일반 사용자 프로그램이 CPU제어권을 가지고 명령을 수행하는 모드이기 때문에 `일반 명령` 만을 수행할 수 있다.

과정을 살펴보자 ← 소프트웨어 인터럽트 과정

위의 process A가 프로그램 명령 수행중에 디스크 입출력 명령을 읽은 경우를 생각해 보자. 사용자 프로그램은 입출력 장치에 접근하는 명령을 수행할 수 없다. user mode에서 특권 명령을 수행할 수 없기 때문이다.
이런 경우에 사용자 프로그램은 운영체제에게 `시스템 콜`을 통해 특권명령을 대신 수행해달라고 요청한다. 시스템 콜은 주소 공간 자체가 다른 곳(커널의 code영역)으로 이동해야 하므로 **프로그램이 인터럽트 라인에 인터럽트를 세팅**하는 명령을 통해 이루어진다.

CPU가 인터럽트 라인을 검사하고 인터럽트가 발생한 것을 감지하게 된다. 현재 수행중인 사용자 프로그램을 잠시 멈추고 CPU의 제어권을 운영체제에게 양도한다. (kernel mode) 그리고 이 때 하드웨어적으로 `모드 비트`가 1에서 0으로 자동으로 세팅되어 특권 명령을 수행할 수 있게 된다.



관련 용어

인터럽트 핸들러

실제 인터럽트를 처리하기 위한 루틴으로 인터럽트 서비스 루틴이라고도 한다.
운영체제의 코드 영역에는 인터럽트별로 처리해야할 내용이 이미 프로그램되어 있다.

인터럽트 벡터

인터럽드 발생시 처리해야 할 "인터럽트 핸들러의 주소"를 인터럽트 별로 보관하고 있는 테이블이다.

PCB(Process Control Block)

커널의 데이터 영역에 존재하며 각각의 프로세스마다 고유의 PCB가 있다.
인터럽트 발생 시 프로세스의 어느 부분이 수행중이었는지를 저장한다.
(수행중이던 memory 주소, 레지스터값, 하드웨어 상태 ...)

< 인터럽트 핸들러>

인터럽트를 처리하기 위해 커널이 실행하는 "함수"
aka 인터럽트 서비스 루틴
인터럽트별 핸들러가 존재, 장치를 관리하는 커널 코드인 장치드라이버안에 있다.

특징>

'linux-2.6.25.10/drivers'

인터럽트가 발생했을 때 "커널"이 OS 호출
인터럽트 컨텍스트라는 특수한 상황에서 실행(이 상황에서는 실행이 중지될 수 없어 단위 컨텍스트라고도 부른다.)
인터럽트는 비동기적, 언제든 발생가능, 핸들러도 언제든 실행가능, 중단된 프로세스를 다시 빨리 재개하기 위하여 핸들러는 보통 짧고 간단하게 짜여져야한다.

**핸들러의 최소요구조건은 인터럽트를 받았다는 사실을 인터럽트를 발생시킨 하드웨어에 알려줘야 한다는 점

IRQ (Interrupt Request)가 무엇입니까?