TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – GJ (박현우)
uc820@naver.com

목차

Chapter 6 인터럽트와 트랩 / 시스템 호출

Chapter 6 인터럽트와 트랩 / 시스템 호출

I 인터럽트 처리 과정

1) 인터럽트가 대체 뭘까?

⇒ 주변장치나 CPU가 자신에게 발생한 'A건'을 리눅스 커널에게 알리는 매커니즘이다.

2) 이 '사건'을 처리하는 녀석은 누군가?

⇒ 인터럽트 핸들러라는 함수가 작업을 처리한다.

3) 인터럽트가 발생하는 상황은?

⇒ ① 외부 인터럽트 : 태스크와 관계없는 주변장치에서 발생된 비동기적 하드웨어 A건

② 트랩 : 태스크와 관계있는 소프트웨어적 사건 (ex. 0으로 나누기, 세그먼테이션, 페이지, 시스템콜)

※ 인터럽트는 결국, CPU 내부냐 외부냐 어디서 감지 되는지에 따라 나뉜다.

4) 인터럽트 발생시, 운영체제의 인터럽트 핸들러가 수행되는 것일까?

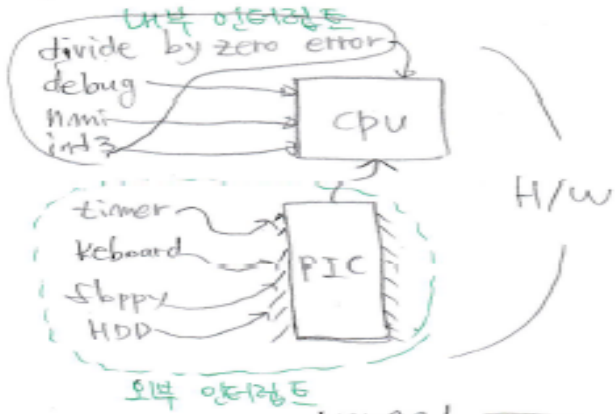
⇒ CPU는 인터럽트가 발생하면 (ARM) program counter (또는 (intel) instruction pointer) 레지스터의 값을 정해진 번지로 변경한다.

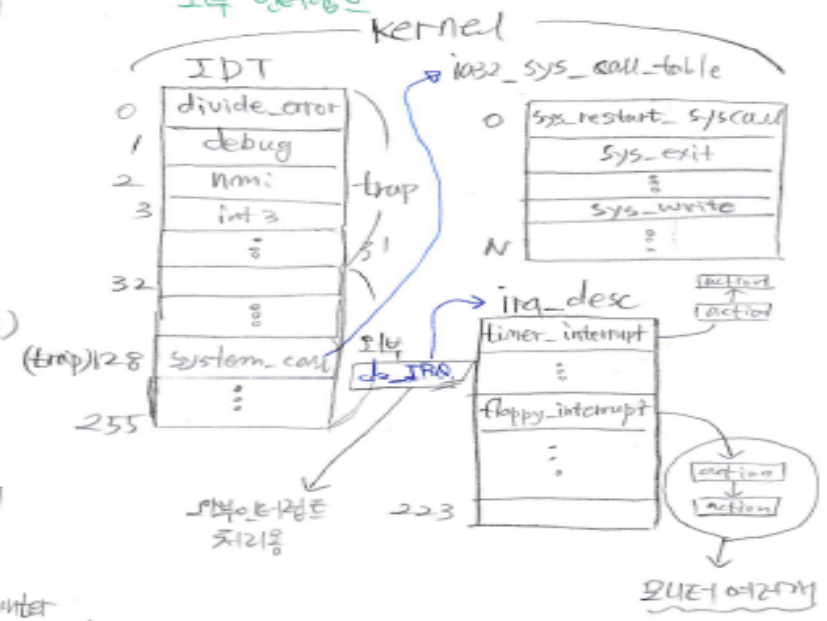
ex) 0x 00 00 0000 + offset

```
__vectors_start:
    W(b)    vector_rst
    W(b)    vector_und
    W(ldr)  pc, __vectors_start + 0x1000
    W(b)    vector_pabt
    W(b)    vector_dabt
    W(b)    vector_addrexcptn
    W(b)    vector_irq
    W(b)    vector_fiq
```

0 = reset
4 = undefined instruction
8 = software interrupt

5) 인터럽트와 트랩의 처리





6) open() 함수가 호출 된다면?

⇒ ① open() 호출 → 레지스터 AX = 5 가 셋팅

② int 0x80 으로 넘어가 kernel 쪽 제어권이 바뀌고 IDT의 128인 system_call이 호출

③ sys_call_table에서 5번에 해당하는 함수가 최종적으로 동작한다.

7) 커널이 인터럽트를 받으면 즉시 인터럽트 핸들러를 왜 호출하지 못하나?

⇒ 바로 'Context switching' 때문이다.

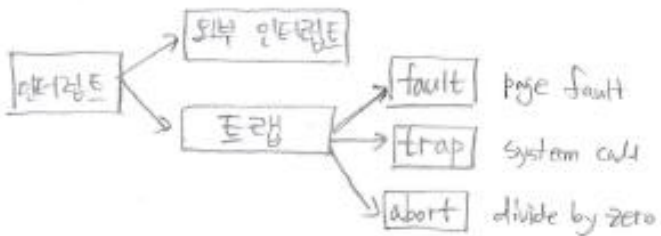
__ switch_to에서 실행

인터럽트도 Context switching을 하니 spinlock, semaphore를 사용한다. 인터럽트를 만드려면 CPU 아키텍처를 이해해야함.

↓

어셈 공부

8) 트랩은 어떻게 구분되나?



① fault를 일으킨 명령이 주소를 eip에 넣어 핸들러 동작후 eip저장된 주소부터 다시 시작

② trap을 일으킨 명령어의 다음주소를 eip에 넣고 그 다음부터 수행

③ 심각한 에러 강제 종료 abort