Homework #4

- Exception Handler를 통해서 paging과 관련된 대표적인 두 가지 exception을 확인해 보고자 한다
  - page fault – page 가 present하지 않을 경우 발생
  - protection fault – page의 protection을 위반할 경우 발생
- 이를 위해 우리는 앞서 완료한 HW#3을 이용하고자 한다.
  - page fault
    - 0x1ff000를 non-present로 한 후, main에서 0을 입력했을 때, access하도록 함
  - protection fault
    - 0x1ff000를 read-only로 한 후, main에서 0을 입력했을 때, write하도록 함

기대하는 테스트 결과

- page fault

```
==========================
        Page Fault Occurs~!
        Address: 0x1ff000
==========================
```

- protection fault

```
==========================
     Protection Fault Occurs~!
        Address: 0x1ff000
==========================
```

Hint #1

- Paging 시, 어디서 exception이 발생했는지는 cr2를 통해서 알 수 있다.
- Paging 시 어떠한 exception이 발생했는지는 Error Code를 통해서 알 수 있다.
  - 그렇다면 error code는 어디에 위치하는가? 책 379p 그림 참조

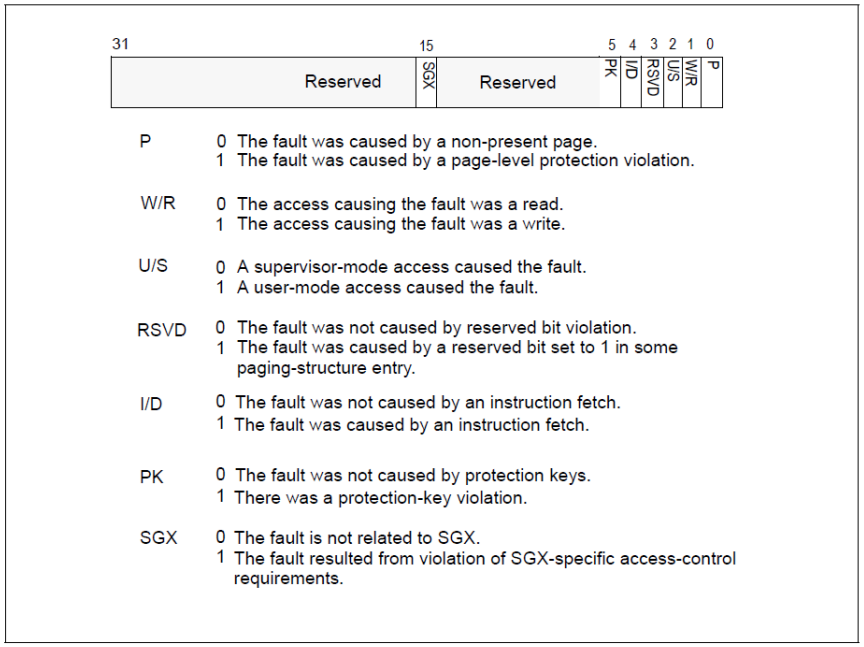

Figure 4-12. Page-Fault Error Code

Homework #4

- exception이 발생했으면, 문제를 해결해야 한다.
- 각각의 경우에 page table entry를 수정하여 문제를 수정함으로써 재부팅 없이 코드가 정상 동작하도록 해보자.
  - example)
    - long *ptr = (*long)0x1ff000;
      *ptr = 0; ← exception 발생
    - 발생 이후, page table entry를 exception이 발생 안하도록 적절히 수정
      - page fault와 protection fault에 따라 분리해서 처리
    - 이후, exception에서 돌아와 다시 위의 코드가 정상 동작하도록 한다.

기대하는 테스트 결과

- page fault
  ==========
  Page Fault Occurs~!
  Address: 0x1ff000
  ==========
  Messesges……
  Messesges……
  Messesges……
  abcde0abcde

Hint #2

- page fault를 해결하기 위해서는 page table entry를 수정해야 한다.
- 본래, page table entry 수정후에는 해당 page와 관련하여 TLB에 cache된 것을 invalidation해야 하고, 이를 위해서는 INVLPG명령어를 사용해야 한다.
  - https://wiki.osdev.org/Inline_Assembly/Examples
- 그런데, QEMU에서는 TLB가 엄격하게 emulation되지 않아서 TLB를 invalidation안해도 동작할 것이다. 그러나, 우리는 넣도록 하자.
  - page table entry 수정;
    invlpg(fault_addr); ← fault_addr은 page fault가 발생한 주소