

Operating Systems Quiz 1 ECE344, Winter 2021

Duration: 1 hour

Examiner: D. Yuan

Instructions

Examination Aids: This is an open book exam.

All questions have been provided in this exam booklet. You need to provide your answers in Quercus.

If any of the questions appear unclear or ambiguous to you, then make any assumptions you need, state them and answer the question that way. If you believe there is an error, state what the error is, fix it, and respond as if fixed.

Please be brief and specific as possible. Clear, concise answers will be given higher marks than vague, wordy answers. Marks will be deducted for incorrect statements in an answer.

Work independently.

MARKING GUIDE

Q1: (2)

Q2: (4)

Q3: (3)

Q4: (6)

TOTAL: (15)

Question 1. Please give 1 example of protected instruction, and 1 example of an instruction that is not a protected instruction

Question 2: Describe every step taken by the computer system **from** when ECE344 instructor Prof. PermanentHeadDamage presses the "Enter" key during a PowerPoint presentation **to the point** when the OS starts to handle the keyboard event. For each step, specify whether it's performed by the hardware (e.g., CPU, keyboard) or software (OS or process). (You only need to describe the steps we covered in the lecture.)

Question 3: Consider this C program:

```
int myval[3];
int main(int argc, char *argv[])
{
    myval[0] = atoi(argv[1]);
    myval[1] = atoi(argv[2]);
    myval[2] = atoi(argv[3]);

    while (1) {
        printf("myval[0] is %d, loc 0x%lx\n", myval[0], (long) &myval[0]);
        printf("myval[1] is %d, loc 0x%lx\n", myval[1], (long) &myval[1]);
        printf("myval[2] is %d, loc 0x%lx\n", myval[2], (long) &myval[2]);
    }
}
```

When executing this program concurrently in two terminals, one with the command `./a.out 1 2 3` and the other with `./a.out 4 5 6`, what are the outputs? Assume:

- `a.out` is the executable compiled from the above program,
- The hexadecimal *memory address* of the first element of the array, i.e., `myval[0]`, in the process running in the first terminal is the same as the last 4 digits of your student ID. For example, if the last 4 digits of your student ID is 1234, then the memory address of `myval[0]` in the first process is `0x1234`.
- Address Space Layout Randomization is disabled

Question 4: After you've taken ECE344, you decided to start your own company X to manufacture a computer system with a MIPS-like CPU and `os161` as the operating system, in

other words, X builds its own CPU and operating system. Very soon you get your first customer: company Y. However, Y complains that the performance of interrupt handling on your system is slower than your competitors (i.e., Linux running on Intel processors). After investigation, you realize that the problem is that different types of interrupts on your system share the same event number.

Specifically, the following is the code snippet in os161 that defines (part of) the Interrupt Vector Table (also known as Interrupt Descriptor Table):

```
/* Names for trap codes */
#define NTRAPCODES 13
static const char *const trapcodenames[NTRAPCODES] = {
    "Interrupt",
    "TLB modify trap",
    "TLB miss on load",
    "TLB miss on store",
    "Address error on load",
    "Address error on store",
    "Bus error on code",
    "Bus error on data",
    "System call",
    "Break instruction",
    "Illegal instruction",
    "Coprocessor unusable",
    "Arithmetic overflow",
};
```

There are a total of 13 events, and event 0 is for "Interrupt". You conclude that you can optimize the performance by splitting different interrupts as different events. For example, event 13 is now "Keyboard interrupt", event 14 is "Hard drive interrupt", etc.

- (a) Why would this change improve the performance of interrupting handling?
- (b) Describe the modifications you need to make to your computer system to support this change. In particular, specify all the components that need to be changed; specifically, do you need to change the CPU, the OS, or both?

(c) After making this change, an engineer at X came up with a follow-up idea to further improve the performance: assign a different event number to each system call. For example, `read()` system call can have event number 15, `write()` system call can have event number 16, and so on. What are the pros and cons of this idea?