87/100

Name: ______Danley Nemorin____

University of Massachusetts - Boston
Intro to Operating Systems

Dr. Ronald Cheung
CS 444 - Fall 2020

In-Class, Open Book Mid-term Examination
November 3, 2020

The work on this examination is to be your own and you are expected to adhere to the UMass-Boston honor system. All questions can be answered by either circling the correct answer(s) or by one or two short sentences. In the latter case, brevity counts. Do not try to make up for a lack of understanding by providing a rambling answer.

1. (3 points) To reset the receiver data ready condition in the IIR register of the serial port, what do you have to do?

I don't know how to really do this, as I didn't do this on the homework nor remember…

-3: No answer

(3 points) How about resetting the transmitter ready condition in the IIR?

outpt(baseport+UART_IER, UART_IER_RDI);
outpt(baseport+UART_IER, UART_IER_RDI|
UART_IER_THRI);

-3: changing the IER does not reset the condition

2. (6 points) Name 2 things a processor has to do to acknowledge a pending hardware interrupt:

ack

i) ___Send interrupt signal to the PIC

OK
____

ii) ______It needs to move all registers AND the EFLAGS

into the stack, before papering for the ISR?

-3: question is on cpu ack interrupt
____

3. (6 points) A user mode program uses a ____user_________ stack to handle function calls and a ___kernel__ stack to handle system calls.

OK

4. (3 points) What is the reason why a multi-threaded program runs faster than a multi-process one?

A multi-threaded program only needs to manage a few things compare to a multi-threaded program. One thread handles a few things (such as PC, PSW, Stack, etc) compare to what a Stack holds. So when it comes to saving Process information on the process table, it's very slow compare to saving information of a thread in a thread table. OK

5. (3 points) What is the advantage of using a POSIX compliant OS?

Purpose of POSIX compliant OS, from what I remember is portability. Incase the OS fails, or stops improving we users/developers can move code to another OS that has similar System calls since it's POSIX compliant. Thus doesn't force developers to spend time writing things from scratch in their code base. OK

6. (6 points) What information do we need to program in order for the processor to know where the service routine begins?

i) for a hardware interrupt:

______ In order to create the service route to begin, we need to create an ioinit, in which we set up an vector number in the IDT , there the computer knows the interrupt and is able to execute the interrupt service routine whenever the hardware sends it. In hw1/hw2 for IO, the interrupt gets called nonstop.
OK

ii) for a software trap:

___You have to setup a trap gate, where you also can create a vector table to be used upon, the service routine can be called from user and in code.______________________________
OK

____

7. (30 points) Circle True (T) or False (F) for each question. (You may add in an explanation if the question is unclear).

T or F F	Interrupts can occur in the middle of a system call because the IF flag =0
T or F T	There is only 1 trap service routine(single entry point) for all syscall services
T or F F	For a process with multiple threads, we set the thread quantum time longer than a process quantum time.
T or F F	ret restores the EFLAGS register
T or F F	After fork(), the parent process always runs first.
T or F T	A running process services its own interrupts as well as interrupts from other blocked processes
T or F F	A zombie process occupies resources.
T or F F	Implementing a thread scheduler in user mode requires kernel mode thread system call services
T or F F	debug_log() in hw2 can store debug data in locations starting at 0x100100

T or F F	In the system call trap service routine, we send EOI to the PIC.

All OK

8. The following C function is provided:

```
#define LENGTH 10

char arr[LENGTH];
int index=0;

int funct(char c)
{
    int i=0;
    if(index >= LENGTH)
    {
      printf("\n No storage\n");
      return -1;
    }
    arr[index] =c;
    index++;
    return index;
}
```

i) (5 points) Is the function thread safe? Why or why not?

It's not thread safe because it's accessing global memory. It should be passed through an parameter.

OK

ii) (15 points) If it is not, write a thread safe version of funct(…).

int funct(char c, char * arr, int * index , int * length,) {

```
   int i=0;
   if(*index >= *length)
   {
     printf("\n No storage\n");
     return -1;
   }
   arr[*index] =c;
   index++;  <-- *index = *index +1
   return *index;
```

}

-1: minor logic error

9. (20 points) If the following syscall function is initialized with eax =0x40; ebx=0x30; ecx=0x20; edx =0x10

```
.globl _syscallc, _syscall

_syscll:   pushl %edx
           pushl %ecx
           pushl %ebx
           pushl %eax
           call _syscallc
           addl $4, %esp

label:     popl %ebx
           popl %ecx
           popl %edx
           iret
```

i) With the listed program, value of %eax at label: =

________0x44___X_______

ii) If this program is modified and this instruction is replaced by: popl %eax, value of %eax at label:

__________0x40___________ OK

and the syscallc program and function in C is given as follows:

```
void syscallc(int arg1,int arg2, int arg3
, int arg4)
{
         arg1 = add3( arg2, arg3, arg4);
}

int add3(int a1, int a2, int a3)
{
         int out;
         out = a1 + a2 - a3;
         return out;
}
```

iii) arg1 =__0x40_____

arg2 = __0x30______

arg3 = _0x20______

arg4 = __0x10______

OK

-3: 1 wrong