Name _____

Student Number _____

Instructor: S. Qiao

COMP SCI 3MH3/6MH3

Day Class
Duration of examination: 50 minutes
McMaster University Midterm Examination October 2007

This examination paper includes **5** pages and **7** questions. You are responsible for ensuring that your copy of the paper is complete. Bring any discrepancy to the attention of your invigilator.

SPECIAL INSTRUCTIONS: This paper must be returned with your answers. Use of McMaster standard (Casio-FX991) calculator only is allowed.

1. (3 marks) Protecting the kernel is crucial to a correctly operating computer system. Providing this protection is the reason behind dual mode operation, memory protection, and the timer. To allow maximum flexibility, however, we would also like to place minimal constraints upon the user. The following is a list of operations which are normally protected. What is a *minimal* set of instructions which must be protected?

1. Change to user mode.
2. Change to kernel mode.
3. Read from kernel memory.
4. Write into kernel memory.
5. Instruction fetch from kernel memory.
6. Turn on timer interrupt.
7. Turn off timer interrupt.

Answer 2, 4, 7 (3, 5, 6 disputable).

2. (3 marks) In a system with an asynchronous disk, when a process makes a disk read request, which state is the process going to? Briefly explain your answer.

Answer When a process makes a disk read request (running), it goes to waiting state waiting for a disk interrupt scheduled by the asynchronous disk.

next page ...

3. (2 marks) Which of the following statements is *true*? If you think all the choices are false, you may answer none.
When `SWITCH` (context switch) is called, two threads switch. The first thing the new thread, the one switched in, does is to

(a) return from `SWITCH`
(b) move the old thread, the one switched out, to the ready queue
(c) wait until the old thread returns from `SWITCH`

Answer (a)

4. (3 marks) Pending interrupts, including timer interrupts, are checked by `CheckIfDue()`, which is called by `OneTick()`. List the three circumstances under which `CheckIfDue()` is called.

Answer
Execution of a user instruction
Interrupts are turned on
Machine is idle

5. (3 marks) The test-and-set instruction

```
TSTSET   7, v, c
```

sets register 7 to the value of variable `v` and assigns the constant `c` to the variable `v`. Use this instruction to implement the binary semaphore operation `P()`, where `v` is the binary semaphore value initialized to 1. Assume the following instruction is available:

```
JMPZERO  7, LABEL
```

which jumps to the instruction labeled `LABEL` when register 7 is 0.

Answer

```
LOOP  TSTSET  7, v, 0
      JMPZERO 7, LOOP
```

next page ...

6. (4 marks) The following are the Nachos implementations of the semaphore operations `P()` and `V()`:

```
void
Semaphore::P()
{
    Interrupt *interrupt = kernel->interrupt;
    Thread *currentThread = kernel->currentThread;

    // disable interrupts
    IntStatus oldLevel = interrupt->SetLevel(IntOff);

    if (value <= 0) {                   // semaphore not available
        list->Append(currentThread);
        currentThread->Sleep(FALSE);
    } else {
        value--;
    }
    // restore interrupt level
    (void) interrupt->SetLevel(oldLevel);
}


void
Semaphore::V()
{
    Interrupt *interrupt = kernel->interrupt;
    Thread *currentThread = kernel->currentThread;

    // disable interrupts
    IntStatus oldLevel = interrupt->SetLevel(IntOff);

    if (!queue->IsEmpty()) {
        // a thread waiting on the semaphore, make it ready
        // and give it the possession of the semaphore
        kernel->scheduler->ReadyToRun(queue->RemoveFront());
    } else {
        value++;
    }

    // restore interrupt level
    (void) interrupt->SetLevel(oldLevel);
}
```

next page ...

If the implementation of `V()` is changed to:

```
void
Semaphore::V()
{
    Interrupt *interrupt = kernel->interrupt;
    Thread *currentThread = kernel->currentThread;

    // disable interrupts
    IntStatus oldLevel = interrupt->SetLevel(IntOff);

    if (!queue->IsEmpty()) {
        // a thread waiting on the semaphore, make it ready
        kernel->scheduler->ReadyToRun(queue->RemoveFront());
    }
    // set semaphore free
    value++;

    // restore interrupt level
    (void) interrupt->SetLevel(oldLevel);
}
```

change the implementation of `P()` accordingly. Explain your changes.

Answer

```
    ...

    while (value <= 0) {                    // semaphore not available
        list->Append(currentThread);
        currentThread->Sleep(FALSE);
    }
    value--;

    ...
```

next page ...

7. (2 marks) One of the four necessary conditions for deaklock is *circular wait*. How would you break the condition to prevent deadlocks?

Answer Assign a unique number to each resource, and request resources in strictly increasing/decreasing order.

END!