Name: _____

Operating Systems
V22.0202 Spring 2011

Midterm Exam

1. **True/False**. Circle the appropriate choice (there are no trick questions).

   (a) **T F** The CPU's kernel mode provides operations that are not available in user mode.

   (b) **T F** The disk driver is the microprocessor within the hard drive that controls the movement of the disk arm.

   (c) **T F** A trap is an interrupt caused by an external event such as a mouse click.

   (d) **T F** A context switch is initiated by an interrupt, such as clock interrupt or a trap.

   (e) **T F** In a batch system, every process runs to completion before the next process runs.

   (f) **T F** All the processes on a modern computer share a single virtual address space.

   (g) **T F** In lottery process scheduling, the total starvation of lower-priority processes is avoided as long as they are each given at least 1 "lottery ticket".

   (h) **T F** In round robin process scheduling, the longer the quantum the more efficient the use of the CPU (i.e. fewer wasted cycles).

   (i) **T F** On a multiprogramming system supporting swapping (but not virtual memory), base and limit registers can be used to point to the beginning and end of the memory space occupied by the current running process.

   (j) **T F** There is no need to have virtual memory on a computer whose physical memory is bigger than the size of the address space.

2. **Short Answers: Answer this question on this sheet**

   (a) In a batch system using shortest-job-first scheduling, what is the average turnaround time for a set of four jobs whose running times are 13, 3, 10 and 15 minutes? Show your work.

   (b) In a multiprogramming system, how would you implement priority scheduling without having the scheduler search through all the processes in the ready queue looking for the highest priority process to run next?

Please turn this page over.

Put the rest of your answers in the blue book

3. (a) Describe briefly what happens on a CPU when an interrupt occurs. That is, what actions does the hardware perform (not the OS)?

   (b) In your programming assignment, it was assumed that when a keyboard interrupt occurred there was a proces that was blocked, waiting for input from a keyboard. In the real world, of course, a user can type on the keyboard even if no process is waiting for the input. Describe how you think an operating system actually handles keyboard interrupts in the real world. Be specific on the actions that an OS would take when a keyboard interrupt occurs.

4. Suppose you were programming on a system that didn't support semaphores but did support a test-and-set instruction. Suppose further that there was a C library procedure

   ```
   test_and_set(int *val, int *lock);
   ```

   that atomically writes the contents of the location that `lock` points to into the location that `val` points to and writes a 1 into the location that `lock` points to.

   Write some (short!) C code for two processes that have a shared data structure that should only be accessed by one process at a time (i.e. requires mutual exclusion) and that use the `test_and_set()` procedure to ensure mutual exclusion. You can use any other shared variable that you want - assume that any global variable used in both processes is shared. Be sure to write as correct C code as possible. It doesn't have to perform a meaningful computation, it just has to show that you know how to write code that requires mutual exclusion and how to implement mutual exclusion using `test_and_set()`.

5. Assume you have a 16-bit computer (i.e. addresses are 16 bits) supporting virtual memory where the page size is 4KB.

   (a) How many elements in a page table would there have to be?

   (b) How would a virtual address be partitioned into page number and offset (i.e. how many bits in each part)?

   (c) Suppose the MMU translated the virtual addresses issued by a program into physical addresses as follows (all address shown in hexidecimal):

   ```
   3E1F → 7E1F
   5454 → 9454
   2A8D → 6A8D
   E127 → 2127
   C8BA → 08BA
   9762 → D762
   72CF → B2CF
   ```

   Draw the entire page table, leaving blank those elements that cannot be determined from the information given. Ignore the present/absent bit, dirty bit, etc.

   (d) How would the code that your wrote for the first programming assignment be different if one of the traps you had to handle was a page fault? Describe what your trap handler would have done in that case.