CS 354 (Park) **Midterm** **March 2 (Fri.), 2018**

Remarks: Keep the answers compact, yet precise and to-the-point. Long-winded answers that do not address the key points are of limited value. Binary answers that give little indication of understanding are no good either. Time is not meant to be plentiful. Make sure not to get bogged down on a single problem.

PROBLEM 1 (60 pts)

(a) What is the practical motivation for isolation/protection? What are the four hardware support features required to achieve isolation/protection? What software support feature is required? What is XINU's approach to isolation/protection?

(b) What are the two hardware support features for providing mutual exclusion? Why is one considered preferable to the other? Can the techniques for providing mutual exclusion be used to prevent producer/consumer queues from becoming corrupted by multiple readers/writers? Explain your reasoning.

(c) What are the roles of the upper and lower halves of XINU and other operating systems? We noted that the scheduler (resched() in XINU) is located between the two halves. Give an example in XINU where the upper half invokes resched(). Do the same for the lower half.

(d) What is the first action that a XINU system call executes when it is entered? What is its last action? Why are these actions performance by XINU? Why are they considered, in general, detrimental to operating system responsiveness? Use the sleepms() system call which enqueues the calling process into a sleep queue (a priority queue similar to XINU's ready list) to explain. What other operations are performed by system calls at their beginning that increase overhead?

PROBLEM 2 (40 pts)

(a) Describe in words—in sequence and correct order—the logical steps that XINU's context switch function ctxsw() takes to save the state of the current process before it is switched out. You don't need to go into address calculation using offsets and indirection. When a new process is context switched in, how does XINU know where in memory (i.e., RAM) to find its saved state? Why does writing the code of ctxsw() require knowledge of how a C compiler (in our case gcc) manages function calls?

(b) Explain the scheduling overhead (i.e., time complexity) of XINU. What is the corresponding overhead of fair scheduling used by Linux? Why is the scheduling overhead of UNIX Solaris which uses a multilevel feedback queue constant? In your answers to XINU, Linux, and Solaris scheduling, explain enqueue and dequeue operations and their overhead separately.

BONUS PROBLEM (10 pts)

We noted in class that modern kernels are reactive systems. What does that mean? Is XINU (the same goes for other kernels such as Linux/UNIX and Windows), as an operating system, a collection of dedicated system processes performing kernel chores? Explain.