CS 354 (Park) **Final** **May 3 (Thu.), 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 feature characterizes real-world file sizes? How has it influenced the design of traditional/UNIX file systems? How would the XINU file system fare under real-world file sizes?

(b) What is Belady's optimal page replacement criterion? In what sense can LRU be viewed as approximating Belady's optimal page replacement? What is global clock and how is it related to LRU? Why are page faults handled by software (i.e., kernel) and not hardware? In modern operating systems, why are pages not evicted at the time of page fault interrupts?

(c) Describe deadlock by providing a simple example involving processes and semaphores. How would a kernel go about detecting deadlock in app processes? What is the resultant overhead? Beyond overhead, what is the rationale that justifies operating systems not getting involved in deadlock detection?

(d) What is memory thrashing? What are its characteristic and observable performance symptoms? Would increasing CPU processing power help alleviate the problem? Would increasing main memory help? Explain your reasoning.

(e) In the asynchronous IPC with callback function lab, we considered an implementation in XINU where the principle of preserving isolation/protection—to the extent feasible—was followed even though XINU does not support isolation/protection. How was this done? Suppose a process that registered a callback function for message reception has low priority and a message is transmitted to this process. Under XINU's legacy scheduler, will be callback function be executed in a timely manner? What about the case of the Solaris TS scheduler?

PROBLEM 2 (40 pts)

(a) What motivates the subdivision of the lower half of operating systems into top and bottom halves? Include a real-world example where this is beneficial. What options exist for implementing the bottom half? What are the pros/cons? Why is DMA hardware support essential for video streaming in isochronous mode? What role does the top half play?

(b) We discussed producer/consumer queues in the context of process coordination/synchronization as part of IPC. That is, a process acts as a writer of a FIFO queue and another process acts as its reader. Suppose we are considering device I/O where a user process performs a read() system call, and the upper and lower halves coordinate with the help of a shared FIFO queue to copy data bits from a device buffer (e.g., Ethernet card) through the shared kernel FIFO queue to a user space buffer whose pointer is provided through read(). Explain why the techniques studied in the context of IPC apply for device I/O. Use an example to specify who the writers/readers of the kernel FIFO queue and user space buffer are. For this example, there is no need to subdivide the lower half into top and bottom halves.

BONUS PROBLEM (10 pts)

What differentiates tickless kernels from tickful kernels? Why might a tickless kernel be well-suited for mobile devices if they are not actively used? We know how XINU's lower half is designed to handle clock interrupts as a tickful kernel. Describe how you would modify XINU's lower half so that it operates as a tickless kernel. Consider two tasks only: sleep queue and time slice management.