CS 354 (Park) **Midterm** **March 3 (Fri.), 2023**

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 (40 pts)

(a) What is the difference between blocking and non-blocking system calls? If performance is of primary importance and messages may be long, why is shared memory based IPC preferred over message passing IPC? In general, what additional kernel support is necessary to achieve orderly IPC via shared memory?

(b) How do we determine whether a process behaves in a CPU-bound or I/O-bound manner with constant overhead (i.e., constant time)? What is the rationale behind dynamically changing priority and time slice values when a process is deemed CPU- or I/O-bound? Would assigning a large time slice to a process deemed CPU-bound cause problems for I/O-bound processes? Would assigning a large time slice to an I/O-bound process cause potential issues for CPU-bound processes? Explain your reasoning.

PROBLEM 2 (40 pts)

(a) What are the pathways that cause a process running in operating systems supporting isolation/protection to switch from user mode to kernel mode? Explain if the same also applies to a XINU process. You implemented a software clock counter, msclkcounter1, in lab1. Suppose a process of high priority runs main() whose code calls disable() before entering a lengthy for-loop. Does this influence the accuracy of msclkcounter1? Explain your reasoning.

(b) Explain the logic behind how XINU handles context-switching in a newly created (ready) process using the same ctxsw() kernel function used to context-switch in a process that executed before and was context-switched out. What happens in both cases when ctxsw() executes the ret instruction? Is restoring EFLAGS before EBP necessary if the process being context-switched in is not a newly created process? Explain.

PROBLEM 3 (20 pts)

How is asynchronous IPC with callback function different from synchronous IPC? In operating systems such as Linux and Windows why is isolation/protection a key consideration when implementing asynchronous IPC?

BONUS PROBLEM (10 pts)

What was the method used by UNIX Solaris to detect and prevent starvation when performing process scheduling? How would we modify XINU to implement the starvation detection (not prevention) method?