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 (36 pts)
(a) Based on the material covered thus far in class, what are key simplifying features of XINU’s kernel architecture that distinguish it from modern operating systems such as UNIX/Linux and Windows?
(b) Deadlocks can be a significant problem for multi-threaded applications that use semaphore system calls to effect mutual exclusion of shared resources. What is the approach followed by modern kernels and why may the approach be construed reasonable?
(c) Why is it meaningful to say that process schedulers based on multi-level feedback queues incur $O(1)$ overhead with respect to time complexity? The revised Linux scheduler, CFS (completely fair scheduler), departs from the traditional CPU- vs. I/O-bound process classification approach by implementing a form of fair scheduling. What is the overhead of CFS discussed in class?

PROBLEM 2 (36 pts)
(a) There are hardware and software methods for implementing process synchronization. Binary semaphores are considered software primitives for achieving mutual exclusion, in contrast to a test instruction which is a hardware primitive. Does this mean that binary semaphores are pure software primitives that do not require any hardware support? Explain using XINU’s wait() and signal() system calls.
(b) Suppose XINU were to be extended to provide multi-processor/multi-core support. Based on the material covered so far, what parts of the XINU kernel would need to be modified, and for what reason? Do existing system calls need to be changed?
(c) What is asynchronous input with callback function, and how must it be implemented in modern kernels to achieve isolation/protection? Why is zero-copy an instance of asynchronous output, where is it used in real-world applications, and what benefits does it bring?

PROBLEM 3 (28 pts)
(a) Suppose RMS or EDF are implemented as part of a kernel aimed at supporting hard real-time guarantees whose primary target is real-time compressed video apps. What technical issues—both kernel design issues and application related assumptions—make achieving real-time guarantees challenging?
(b) Explain how full virtualization works to achieve low overhead support of virtualization in modern computing systems. Use an example of two guest kernels (e.g., Linux and Windows) and hypervisor running over hardware. What hardware features does full virtualization depend upon, and what complication is introduced in x86 machines? Describe one approach to deal with this complication.

BONUS PROBLEM (10 pts)
Operating systems may be viewed as reactive systems comprised of passive library functions that are invoked by system calls from user processes and interrupts from hardware devices. Suppose a kernel design is proposed where its services are implemented as a set of kernel processes. What would be the drawbacks of such an approach?