System Architecture 2008/09
Assignment 7

Question 7.1: Synchronization Primitives

1. Recall the requirements for a valid solution to the problem of critical sections.
2. Distinguish the various types of synchronization objects and summarize their respective operations’ semantics:
   - signals, counting semaphores, binary semaphores, mutex objects, barriers, condition variables, monitors, locks
3. What are strong semaphores as opposed to weak semaphores?
4. Which of the above objects are suitable to protect critical sections?

Question 7.2: Emulating Atomic Test-And-Set

Consider a computer that does not have a test-and-set instruction, but does have an instruction to swap the contents of a register and a memory word in a single indivisible action. Can that be used to write a routine to enter a critical section, like acquire(SMP) on slide 51 of lecture 9 (Mutual Exclusion)?

Question 7.3: Environmental Influences on Mutual Exclusion

1. Explain why spinlocks are not appropriate for single-processor systems, yet are often used in multi-processor systems.
2. Explain why disabling interrupts is not an appropriate means for implementing synchronization primitives in multi-processor systems.
3. Show how to implement the P() and V() semaphore operations in multi-processor environments using the testAndSet() instruction. The solution should exhibit minimal busy waiting.

Question 7.4: Generalizing Peterson’s Algorithm

Peterson’s solution to the mutual exclusion problem for two threads (i.e., Algorithm 3 of the lecture slides, see below) can be generalized to provide mutual exclusion among $n > 1$ threads. Design this solution and prove that it meets the following three requirements:

- mutual exclusion
- deadlock freedom
- no starvation
Question 7.5: IPC Basics

Explain the following design parameters for an IPC mechanism. Discuss pros and cons of each possible parameter value.

- connection-oriented vs. connectionless
- asynchronous send vs. synchronous send
- asynchronous receive vs. synchronous receive
- buffered vs. unbuffered
- direct addressing vs. indirect addressing