CSL373/CSL633 Minor 2 Exam
Operating Systems
Sem II, 2012-13

Answer all 6 questions Max. Marks: 30

1. **True or False**. Give short reasons. No marks for incomplete/wrong reasons. [10] (2 each)

a. Pages that are shared between two or more processes can never be swapped out to disk

b. CLOCK algorithm (1-bit approximation to LRU) always has a poor hit ratio, when compared with LRU

c. With hardware-managed TLBs (as on x86), every write to a virtual address results in a write to the page table entry corresponding to that virtual address (to set the dirty and accessed bits). In other words, every write to a virtual address results in two physical memory writes.

d. Pintos does not need to use spinlocks.

e. A blocking lock is always more efficient than a spinlock, as it relinquishes CPU.

2. Implement an infinite queue with potentially multiple producer and multiple consumer threads. Use locks and sleep/wakeup for synchronization, if needed. [2]

3. Consider the statement ‘release(&ptable.lock)’ at line 2487 in forkret() function. The comment before the line says that this releases the ptable.lock from scheduler. Where is the corresponding call to bracketing acquire(ptable.lock)? Is it true that the parent acquires the lock and the child releases it in forkret()? List the sequence of lock/unlock events (for ptable.lock) by parent/child on a fork. (Also say in which function these events occurred). [6]

4. Consider the kernel stack (kstack) of a switched-out process (a process in RUNNABLE state but not currently running) in xv6. What is the minimum number of code pointers on the kernel stack (i.e., they point to an executable instruction)? List them. How many of these are user pointers (i.e., pointing into user address space)? [6]

5. There are two ways to access files on Pintos:
    a. open/read/write/close system calls
    b. mmap/munmap system calls

    List at least one advantage of each interface (read/write and mmap/munmap) of accessing files. [3]

6. Implement a blocking lock using sleep and wakeup calls (as on xv6). In other words, implement ‘blocking_acquire()’ and ‘blocking_release()’. You are free to use other synchronization primitives like spinlocks. [3]