University of California at Berkeley
College of Engineering
Department of Electrical Engineering and Computer Science

CS 162 I. Stoica
Spring 2010

FIRST MIDTERM EXAMINATION
Tuesday, March 9, 2010

INSTRUCTIONS—READ THEM NOW! This examination is CLOSED BOOK/CLOSED NOTES. There is no need for calculations, and so you will not require a calculator, Palm Pilot, laptop computer, or other calculation aid. Please put them away. You MAY use one 8.5" by 11" double-sided crib sheet, as densely packed with notes, formulas, and diagrams as you wish. The examination has been designed for 100 minutes/100 points (1 point = 1 minute, so pace yourself accordingly). All work should be done on the attached pages.

In general, if something is unclear, write down your assumptions as part of your answer. If your assumptions are reasonable, we will endeavor to grade the question based on them. If necessary, of course, you may raise your hand, and a TA or the instructor will come to you. Please try not to disturb the students taking the examination around you.

We will post solutions to the examination as soon as possible, and will grade the examination as soon as practical, usually within a week. Requests for regrades should be submitted IN WRITING, explaining why you believe your answer was incorrectly graded, within ONE WEEK of the return of the examination in class. We try to be fair, and do realize that mistakes can be made during the regarding process. However, we are not sympathetic to arguments of the form "I got half the problem right, why did I get a quarter of the points?"

_____ SID: _____
(Signature)

_____ Discussion Section (Day/Time): _____
(Name—Please Print!)

QUESTION	POINTS ASSIGNED	POINTS OBTAINED
1	20	
2	20	
3	20	
4	20	
5	20	
6 (bonus question)	5	
TOTAL	100 (+5)	

Question 1. (20 points)

True or False *(12 points). Instructions: write "true" or "false" after each of the following statements (Please give one statement justification for each of your answers).*

a. (2 points) Each physical page belongs to only one process.

b. (2 points) Every interrupt changes the CPU from user mode to kernel mode.

c. (2 points) All Page Table entries will be invalid directly after a context switch between processes.

d. (2 points) The CPU scheduling policy that minimizes average completion time can lead to starvation.

e. (2 points) If there is only one CPU in the computer, then at most one process can be in the "running" state at any time.

f. (2 points) The "Shortest-Job-First" policy always results in the lowest average completion time in a batch system.

Multiple Choice, fill in the blank, short answer *(8 points). Instructions: Circle the correct option(s), or fill in the blanks below.*

a. (2 points) Page Tables can be stored (circle all answers that apply):
    a. In physical memory
    b. In the L2 Cache
    c. On the hard drive

b. (2 points) Condition variables support the following 3 operations:
    a. _____
    b. _____
    c. _____

c. (2 points) Of the following items, circle those that are stored in the *thread* control block.
    a. CPU registers
    b. page table pointer
    c. stack pointer
    d. ready list
    e. segment table
    f. thread priority
    g. program counter

d. (2 points) Processes (or threads) can be in one of three states: **Running**, **Ready**, or **Waiting**. For each of the following examples, write down which state the process (or thread) is in:
    a. Spin-waiting for a variable y to become zero. _____
    b. Having just completed an I/O, waiting to get scheduled again on the CPU. _____
    c. Blocking on a condition variable for some other thread to signal it. _____

Question 2. *Threads and Synchronization (20 points)*

(a) (4 points) Give two reasons of why disabling interrupts can be a bad way to implement critical sections.

(b) (6 points) Assume you are given the CompareAndSwap atomic primitive defined (in C) as follows:

```
bool CompareAndSwap(int* value, int from, int to) {
  if (*value == from) {
    *value = to;
    return true;
  }
  return false;
}
```

Consider the following implementation of lock acquire using CompareAndSwap:

```
void acquire() {
  while(!CompareAndSwap(&value, UNLOCKED, LOCKED));
}
```

i) Give one reason why this implementation may not be efficient.

ii) Without preemption, is this still a valid implementation of acquire()? Why? (Use no more than two sentences for your explanation.)

(c) (8 points) Complete the following implementation of the P() and V() functions for a semaphore using condition variables using Mesa semantics.

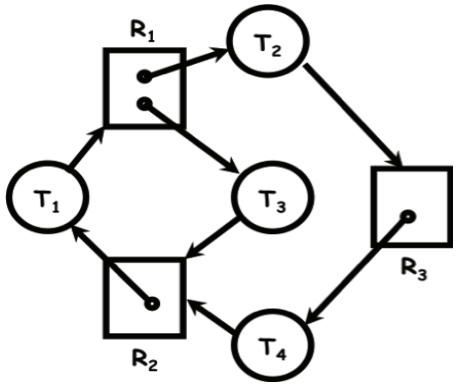
```
void P()  {
    lock.acquire();



    value -= 1;



}

void V()  {



    cond.wake();



    lock.release();
}
```

(d) (2 points) How could you simplify your implementation if you were given condition variables that follow Hoare semantics instead of Mesa semantics? Use no more than two sentences to explain your answer.

Question 3. *Deadlock (20 points)*

(a) (4 points) Name the four conditions under which deadlock occurs.

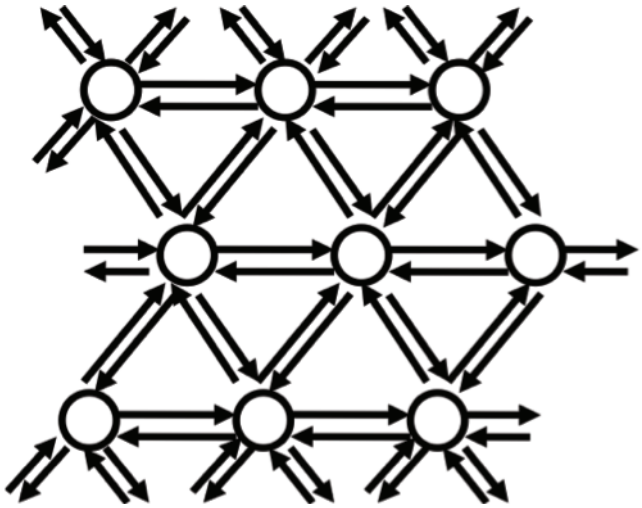
(b) Consider the resource allocation graph below consisting of four tasks (T1, T2, T3, and T4) and three resources (R1, R2, and R3). R1 has two instances while the other two resources have only one instance each. An arrow from a task to a resource denotes the fact that the task requests one instance of the resource, while an arrow from a resource to a task denotes the fact that the task holds an instance of the resource. Assume that resources cannot be preempted and tasks do not voluntary release resources before terminating.



i) (4 points) Are the tasks shown in the above resource graph deadlocked? If yes, explain how each of the four deadlock conditions is satisfied. (Use one sentence per condition.) If not, give a possible sequence in which the tasks execute.

ii) (4 points) Assume you are allowed to add an extra instance of either resource R1 or R2. Are the tasks deadlocked in this case? If yes, explain how each of the four deadlock conditions is satisfied. (Use one sentence per condition.) If not, give a possible sequence in each the tasks execute.

(c) Consider the network below implementing wormhole routing between any two nodes. Each node is connected to six of its neighbors by two network links, one in each direction.
Messages are routed from a source node to a destination node and can stretch through the network (i.e. consume links along the route from source to destination). Messages can cross inside nodes. Assume that no network link can service more than one message at a time, and that each message must consume a continuous set of channels (like a snake).
Hint: You can think about a message as a train that spans multiple links (possible all the way from source to destination), and of each link as a one-way rail.



i) (4 points) Show a situation (with a drawing) in which messages are deadlocked and can make no further progress. Explain how each of the four conditions of deadlock is satisfied by your example.

ii) (4 points) Define a routing policy that avoids deadlocks in the network of (i). Explain using at most two sentences why your routing policy avoids deadlock.

Question 4. *CPU Scheduling (20 points)*

Your boss asks you to pick a scheduling algorithm to run the following processes:

- P1, which requires 1 hour of CPU time to complete;
- P2, which requires 2 hours of CPU time to complete;
- P3, which requires 1 min of CPU time to complete;

Assume all processes are ready at the same time.

a) (5 points) Assume you use First Come First Serve (FCFS) to schedule these processes. Which is the schedule with the largest average completion time? Which is the schedule with the smallest average completion times? Compute the average completion time in each case.

b) (5 points) Assume you use Round-Robin with a time quanta (slice) of 100ms. What is the completion time of each job in this case? Assume that there is no context switch overhead.

c) (5 points) Now assume that P1 arrives at t=0, P2 arrives at time t=10min, and P3 at time t=20min. What will be the completion time of each process when using the Shortest Remaining Time First (SRTF)?

d) (5 points) Assume now that P3 is performing an I/O operation every 10ms and that the I/O operation takes 40ms, but would still take 1 minute to run if it was the only process running. Assume that P1, P2, and P3 are submitted at the same time and you use Round-Robin scheduling (like in part (b)).

i) How long does it take P3 to complete if the time slice is 100ms?

ii) What time slice length would minimize the completion time of P3?

Question 5. *Page Tables & TLBs (20 points)*

Orange Inc hires you to design the virtual memory system for a new cell phone with 32-bit virtual and physical addresses, in which memory is allocated in 2 KB pages. Suppose that you decide to use a single-level page table, in which you also store three metadata bits for each page: Writable, Executable and Valid.

(a) (4 points) Answer the following questions, briefly explaining your solution:

i) How long, in bits, is a virtual page number?

ii) How long, in bits, is a physical page number?

iii) How long, in bits, is an offset within a page?

iv) How much memory is needed to store the page table of each process?

(b) (4 points) Your manager asks you to consider using a multi-level page table in your design. Explain one advantage and one disadvantage of multi-level page tables over single-level page tables. (Use no more than four sentences in total.)

(c) (12 points) In this question, you are asked to predict the results of a sequence of memory accesses on a machine with a single-level page table and a Translation Lookaside Buffer (TLB). Suppose that the machine has 20-bit virtual and physical addresses and 256-byte pages, and that the TLB is fully associative and holds 3 entries. The initial contents of the TLB and page table are shown on the next page. (For the page table, we show only a subset of the entries; assume that the ones not shown are not valid.)

Initial TLB:

Virtual Page #	Physical Page #	Writable?	Valid?
0x100	0x200	0	1
0x101	0x300	0	0
0x200	0x320	1	1

Page Table:

Virtual Page #	Physical Page #	Writable?	Valid?
0x100	0x200	0	1
0x101	0x100	0	1
0x200	0x320	1	1
0x201	0x321	1	0
0xFFF	0x100	1	1

For each memory access in the sequence on the *next page*, write whether the TLB is hit, whether the access succeeds, and, if so, which physical address is accessed. Also show the state of the TLB after each access, assuming that TLB entries are replaced using a Least Recently Used (LRU) policy. Assume that the TLB is not updated if an invalid page is accessed.

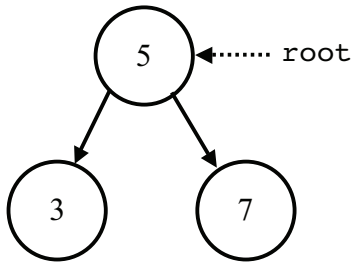
Student Name: _____ SID: _____

Instruction	**TLB Hit?**	**Success?**	**Physical Address**	**New TLB State**
LOAD 0x20012				Virtual Page # | Physical Page # | Writable? | Valid? | | | | | | | | | | | |
STORE 0x10001				Virtual Page # | Physical Page # | Writable? | Valid? | | | | | | | | | | | |
LOAD 0x10101				Virtual Page # | Physical Page # | Writable? | Valid? | | | | | | | | | | | |
STORE 0xFFFFF				Virtual Page # | Physical Page # | Writable? | Valid? | | | | | | | | | | | |
STORE 0x20009				Virtual Page # | Physical Page # | Writable? | Valid? | | | | | | | | | | | |
STORE 0x32000				Virtual Page # | Physical Page # | Writable? | Valid? | | | | | | | | | | | |

Question 6 (BONUS). *Thread Interleavings (5 points)*

The following Java code implements insertion into a binary tree:

```
void insert(Node node, int value) {
  if (value <= node.value) {
    if (node.left == null) {
      node.left = new Node(value);
    } else {
      insert(node.left, value);
    }
  } else {
    if (node.right == null) {
      node.right = new Node(value);
    } else {
      insert(node.right, value);
    }
  }
}
```



Unfortunately, this implementation is not thread-safe. Suppose that we start with a binary tree containing the values 3, 5, and 7, organized as shown above, and that we have a reference `root` to the root node (the one containing 5). We then launch the two threads below, each of which tries to insert two values into the binary tree. What are the possible outcomes after both threads finish? Draw each possible resulting tree in a format like the starting tree above. (You may continue on the back of this page if necessary.)

Thread 1	Thread 2
`insert(root, 0);` `insert(root, 2);`	`insert(root, 1);` `insert(root, 6);`