ECE153A Midterm
Autumn Quarter
November 2, 2000
UC Santa Barbara
Total 70 pts

I. (25 pts) For each question, choose the letter (a), (b), (c), (d), or (e) as your answer. Score = (5× number right) - number wrong, so random guessing nets you nothing on the average. If you write down the steps that arrive at your solution to a problem, you may receive partial credit.

1: Who do you predict will win the U.S. presidential election next week? (a) Bush (b) Gore (c) Buchanan (d) Nader.

Suppose the initial values of registers R_1, R_2 and R_3 are all 0. Suppose memory addresses 1000, 1004 and 1008 contain the following initial values.

```
Memory Address Value
----------------------------
1000: 1008
1004: 1000
1008: 1004
----------------------------
```

Answer the following *two* questions.

2: What is the value of R_2 after executing the following sequence of instructions?

1. LOAD R_1,#1000
2. LOAD R_2,(R_1)
3. ADDL R_2,@(R_2)

(a) $R_2 = 2016$ (b) $R_2 = 2008$ (c) $R_2 = 2000$ (d) $R_2 = 2004$ (e) $R_2 = 2012$.

3: What is the value of R_2 after executing the following sequence of instructions?

1. LOAD R_2,#1008
2. ADDW R_2,(R_2)+

(a) $R_2 = 2014$ (b) $R_2 = 2018$ (c) $R_2 = 2020$ (d) $R_2 = 2012$ (e) $R_2 = 2016$.

4: You are given the following three periodic tasks. Identify a feasible real-time schedule that maximizes the total importance factor.

Task i	E_i (ms)	T_i (ms)	*Importance Factor*
1	20	40	20
2	160	200	40
3	200	400	30

(a) Task 2 (b) Tasks 1 and 2 (c) Tasks 2 and 3 (d) Tasks 1 and 3 (e) Tasks 1, 2 and 3.

5:Which of the following statement(s) is (are) false?

1) Interrupt driven IO is more efficient than pooled IO because the CPU does not have to constantly check the status of the IO devices.

2) The most critical performance objective of a real-time scheduler is to maximize the throughput of the system.

3) Shortest Job First (SJF) is a CPU scheduling policy that can achieve the theoretical minimum average waiting time for the processes.

(a) 1, (b) 2, (c) 3, (d) 1 and 2, (e) 2 and 3.

II. (24 pts) **True or False questions**. Indicate whether each of the following statement is *TRUE* or *FALSE*. If you choose *FALSE*, you must explain the reason(s) or no credit is given.

1: Using a faster CPU can always make a set of non-schedulable periodic tasks schedulable.

2: No process context switch happens when a process is moved from the *Blocked* state to the *Ready* state.

3: Once a non-preemptive scheduler gives the processor to a program that is in an "while (TRUE);" loop, the non-preemptive scheduler will never regain control of the CPU.

4: Suppose a system employs a time-sliced (quantum) and round-robin scheduling policy. Having a small time-slice (quantum) improves the average waiting time of new processes.

5: The following two processes, P_1 and P_2, cannot run into deadlock since semaphores are used. (Assume that both S and Q are initialized to be one.)

P_1	P_2
$wait(S)$	$wait(Q)$
$wait(Q)$	$wait(S)$
...	...
$signal(Q)$	$signal(S)$
$signal(S)$	$signal(Q)$

6: The system call mechanism (i.e., trap) is in place to improve the throughput of the system.

III. (21 pts) Short Answers.

1 (Real-time Systems (9 pt): You are given the following four periodic tasks. Identify a real-time schedule that maximizes the CPU utilization.

Task i	E_i (ms)	T_i (ms)
1	40	200
2	20	100
3	30	60
4	45	300

2 (Multi-level Feedback Scheduling) (12 pts):
Suppose a system employs a Multi-Level Feedback policy with the following three scheduling rules:

1. There are three service queues in the system, each uses a FIFO scheduling policy. The first queue (the highest priority queue) has a quantum of 30 ms. The second queue has a quantum of 40 ms. The third queue (the lowest priority queue) has a quantum of 100 ms. A higher priority process always preempts a lower priority process. A preempted process is always given its full quantum back. A context switch takes 5 ms.

2. A newly arrived process is placed in the second queue.

3. A process' priority is demoted after a quantum-end and is promoted after an IO completion.

Answer the following two questions:
(a) How do you change the scheduling rules to minimize the response time of a newly arrived process? (5 pts)

(b) How can you write a trick program to fool the scheduler to always gain the highest scheduling priority? (4 pts)

(c) As a system designer, how can you fix the scheduling rules to avoid the subterfuge attack described in part (b)? (3 pts)