Stanford University
Computer Science Department
CS 240 Quiz 1
Spring 2003

May 15, 2003

This is an open-book exam. You have 50 minutes to answer eight out of ten questions. Write all of your answers directly on the paper. Make your answers as concise as possible. Sentence fragments ok.

NOTE: We will take off points if a correct answer also includes incorrect or irrelevant information. (I.e., don't put in everything you know in hopes of saying the correct buzzword.)

Question	Score
1 - 2	
3 - 5	
6 - 8	
total	

Stanford University Honor Code

In accordance with both the letter and the spirit of the Honor Code, I did not cheat on this exam nor will I assist someone else cheating.

Name and Stanford ID:

Signature:

Answer all 8 of the following 8 questions and, in a sentence or two, say *why* your answer holds. (5 points each).

1. The Therac paper is a study in severed feedback loops, ranging from AECL not informing customers of problems in the Therac-25 to the video camera being off when a patient was zapped. From the paper, give three (other) examples of such missing feedback loops.

   There were a bunch of different possible answers:

   (a) *lack of mechanism to follow up reports at aecl*
   (b) *assertion that no other accidents had happened*
   (c) *printout feature disabled, so no hard copy of the data*
   (d) *reporting regulations only applied to manufacturers rather than users*
   (e) *error messages*
   (f) *aecl could not reproduce accidents*
   (g) *letter to users did not warn that patient injury was involved*
   (h) *overdoses not acknowleged.*
   (i) *audio/video off*
   (j) *slow letters*
   (k) *lack of information propogation for complaints by users*

2. Assume we have two threads, T1 and T2, that execute the following code in the following order:

```
int *q = NULL; /* shared global variable */

        T1                               T2
 int *p = malloc(sizeof *p);
 *p = 0;
 q = p;
                                  while(q == NULL)
                                       ;
                                  *q = 1;
                                  q = 0;
while(q != NULL)
    ;
printf("p = %d\n", *p);
exit();
                                   exit();
```

   Assuming naive memory semantics (if you don't know what this means, you are just fine): What will Eraser do for this code? Is its behavior correct?

   Eraser will notice the access to the shared variable without lock and will report an error.

   There is no race condition in the code because the threads implicitly synchronize using the pointer value stored in `q`.

3. For the following snippet of Mesa monitor code:

```
void foo() {
    signal(c);
}
```

If the code runs on a system with different priority processes, what is a potential way that it will generate useless overhead? Is there an automatic way to fix this problem? *If a thread T2 with a higher priority is waiting on c, then when c is signaled by T1, the system will switch from the current thread T1 to T2, which will immediately block when attempting to acquire the monitor lock, be put on the lock queue, and then the system will eventually switch back to T1.*

The compiler can release the monitor lock before the signal if signal is the last instruction in the monitor function.

4. What would Hoare-wakeup semantics correspond to in a message passing system? *. Hoare-wakeup semantic corresponds to adding the port to the list of ports you are waiting on and immediately processes a message on that port if one is available, before processing any other message.*

5. Identify several potential weaknesses in the VMS memory system and describe experiments to measure them.
Something about a mix of applications; heavy paging, light paging, varying memory sizes, varying the parameters used to control the lists.

6. When the reference bit of a base page within a superpage is reset, the superpage management system demote the superpage speculatively, recursively performing the demotions with a probability $p = 1$. Suppose we set $p = 0.5$ instead. Give one reason why the system might perform better because of that, and one reason why the system might perform worse because of that.
Worse: if you didn't demote and only one base page out of the superpage was being used, you waste memory. Better: if you didn't demote and all pages were being heavily used, you save useless faults and repromotions.

7. When measuring active memory (for use in idle memory tax calculations), ESX Server keeps three statistical averages: a "slow" average, a "fast" average, and a "very fast" average. Explain the problem that could result if ESX stopped using the "slow" average.
Without the "slow" average, the system will react too quickly to changes. E.g., assume the guest OS runs two applications, the first memory intensive, the second CPU bound. Each app gets 10ms to run (the scheduling quanta). Then when the CPU bound one is scheduled, it will use little memory, most of the guest's memory will be judged as idle and the entire guest system will be penalized.

8. Assume you can mark pages as non-executable. What SFI restrictions does this lift and how would you exploit this flexibility?

SFI can mark pages as non-executable. Therefore it does not have to verify that jumps do not jump to the data segment. Thus, it does not have to differentiate between code and data and so will not need two different segment registers, and the code segment does not need to be as large as the data segment.