Stanford University
Computer Science Department
CS 240 Quiz 1
Winter 2002

This is an open-book exam. You have 50 minutes to answer as many questions as possible. The number in parenthesis at the beginning of each question indicates the number of points given to the question. 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	Points	Score
1 - 10	40	
11	10	
total	50	

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:

1. Short-attention-span questions (40 points)
Answer **8 of the following 10 questions** (i.e., skip two) and, in a sentence or two, say *why* your answer holds. (5 points each).

1. What would be a difficulty in detecting the Therac races using an Eraser-type approach? *1) Therac didn't use locks. 2) The bugs were triggered by the timing of user input events, thus making testing of erroneous code paths difficult. 3) One bug was a race involving a variable which overflowed, something Eraser doesn't catch. 4) A creative answer was that the Therac had rather limited memory, so a tool like Eraser which doubles the heap wouldn't work on actual machines.*

   Any one of the above answers: 5 points

   Answers that were too vague or didn't mention any difficulties specific to Therac: 3 points. Answers such as "erroneous input was required to trigger the bug," or "Eraser must test all branches" fall into this category.

2. The Eraser paper claims that they have never observed more than 10,000 distinct sets of locks for a given program. Give a data structure and locking policy that would pass this limit.

   1) A linked list (array, matrix, etc.) with lock acquired on every element. 2) database with a lock for each record.

3. Would a Mesa programmer have any use for a Mesa version of Eraser?
   Yes, to help detect potential deadlock, and to detect when you should be using a monitor to access shared variables but aren't (Mesa doesn't force you to use monitors). Also, you can synthesize locks using monitors and use an Eraser-like tool.

   "Yes" but poor/no explanation: 3 points. "No": 0 points.

4. Give two examples where Mesa makes a "New Jersey" style decision.

   1) Mesa signalling semantics make the implementation simpler and faster at the expense of forcing users to follow a coding convention on calls to wait(). Hoare semantics would be the "right thing." 2) Monitor locks are not recursive, so, for example, entry procedures cannot contain recursive calls. 3) The monitored record "Locks with" clause doesn't protect an object from modification. 4) Does not worry much about fairness/priority inversion for locks: "in a properly designed system should not be many processes waiting for locks." 5) Unwind doesn't automatically unlock a monitor unless the programmer installs an exception handler. 6) Naked notify means that monitors receiving interrupts from devices can be executing without the monitor lock and therefore could have race conditions.

5. Explain from the Mesa paper: "...[while] any procedure suitable for forking can be called sequentially, the converse is not true."
   The context of the quote is very specific. Procedures suitable for forking are also orinday procedures and so can be called sequentially. However, procedures which are meant to be called sequentially might not catch exceptions, which would cause system to go to the debugger.

   It is also true that forking off arbitrary procedures may cause race conditions. While correct, it wasn't the reason for the quote in the paper. This answer got 2 points.

6. What is the parallel in the procedure approach to a message sitting on a queue in the message-based approach?
   A thread sitting in a monitor lock queue. Answers only indicating a thread that called "wait" got 3 points.

7. Huck believes that the HPT will perform uniformly better than an inverted page table. What is the intuition for this belief?
   IPT has an additional layer of indirection since it must hash into the HAT and then follow another pointer reference into the page table.

8. Huck talks about having one HPT or IPT for the entire system. Why would you not have one per process as with forward mapped page tables?
   Having one HPT or IPT for the whole system saves the memory space and time costs of maintaining a hash table for each process whose size is proportional to the size of physical memory.

   The key point was avoiding the use of space for multiple hash tables and by having only one. Some points were also awarded for discussing the relation of the tables to physical memory. Points were subtracted for additional incorrect descriptions of the page table architectures.

9. How might you expect the relative performance of the webservers to change if the FLASH experiments were rerun without any modification on a multi-processor system.
   MP and MT should start doing much better since they parallelize. AMPED could improve slightly since the OS may take advantage of concurrency.

   3-5 points were subtracted for answers that said MT would not perform better or that SPED would perform the best. 1 point was deducted for saying that AMPED would see no speed improvement. Points may have also been deducted for extraneous incorrect reasoning.

10. From what you can imagine about the features of their workload: would google.com or CNN.com be more likely to be more satisfied with SPED?

    The main point was that SPED performs better on cached workloads than on diskbound workloads.

    1-3 points were deducted for incorrect reasoning as to the types of workloads of each site or about how SPED functions.

2. OOZE (10 points).

Part A. (5 points) What would you need to do to compact OOZE objects on disk similarly to how the system compacts them in RAM? How is this different than compacting them when they are in RAM?

Two differences. One, object pointer encodes disk location, so if you move object, have to update all pointers to it. Two, objects on disk live in pseudo-classes so do not have the same freedom as moving RAM objects. Can combine pseudo-classes, but not put two different types of objects in same class.

fragmentation

Part B. (5 points) Recall that the object pointer in the OOZE paper has the pseudo-class number in the upper bits and the offset in the lower bits. Would swapping these fields (so the pseudo-class number is in the low bits) have the same effect as swapping the page number and offset fields in the virtual address system?

Should have no real effect in the OOZE system: swapping the two fields has no effect on where the object lives in memory or on disk. In contrast a page-based system will get unhappy since switching will put things spatially close together on different pages, which will consume extra TLB entries, page table entries, and require more pages to be in core.