Stanford University
Computer Science Department
CS 240 Quiz 1 Spring 2009

April 29, 2010

!!!!!! SKIP 20 POINTS WORTH OF QUESTIONS. !!!!!

This is an open-book exam. You have 75 minutes. Cross out the questions you skip. 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
1-9 (45 points)	
10-13 (50 points)	
total (max: 75 points):	

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:

Short answer questions: in a sentence or two, *say why your answer holds.* (5 points each).

1. You run your code on a machine that does not guarantee that a load will see the value of a previous store unless an explicit memory synchronization operation was performed (e.g., a memory barrier, a lock or unlock). If you check your code using an unmodified version of Eraser what parts the Eraser state machine will cause it to miss errors of this type? When will it catch them?

2. You change MESA to use "recursive locks" where the same lock can be acquired multiple times by the same thread. Why would this be useful as a system grows in size? How would Lampson say `wait` should work in this brave new world and why?

3. Consider the following MESA monitor:

```
condition c;

entry foo() {
        wait(c);
        return;
}
entry bar() {
        signal(c);
        return;
}
```

Point out four places where a thread could get switched out. NOTE: the switches cannot occur for completely identical reasons.

4. Assume we try to use a variant of the double-check lock idiom in a not particularly useful routine:

```
int div(unsigned x) {
      static int *p;

      if(!p) {
              lock(l);
              if(!p) {
                      int *t = malloc(sizeof *t);
                      *t = 4096;
                      *p = t;
              }
              unlock(l);
      }
      return x / *p;
```

Assume your compiler obeys the POSIX requirements laid out in Boehm and that on your machine that a load always returns the value of the last store. Give two Boehm-style compiler optimizations that could break this code. Does acquiring lock `l` on the read of `p` fix the problem?

5. You see a lock implementation that looks like:

```
void lock(int *l) {
        while(*l != 0)
                yield(*l);
        *l = thread_id();
}
void unlock(int *l) {
        *l = 0;
}
```

   Using the "Cooperative Task Management" paper's classification system, what type of threading system would use such a lock implementation? What is the point of assigning the thread ID to the lock? Let's say you took a program that worked fine, put such lock calls in, and now it has race conditions. What is going on?

6. The "Why Events are a bad idea" paper states: "our thread package ... translates blocking I/O requests into asynchronous requests internally." How does this work?

7. Say concretely what will happen if we compiled the following code with Rinard's failure oblivious compiler and ran it. Does this code satisfy or violate the requirements that they state are needed for failure oblivious to work well?

```
char src[4];
strcpy(src, "hello world");
printf("%s\n", src);
```

8. Hardware VMM vs Software VMM: Give two calls to `isPrime` that would remove the two continuations left in its translated code.

9. Hardware VMM vs Software VMM: A guest OS evicts a page. How will VMware see this and what should it do? Does it matter if the page contained code or data?

Problem 10: Binary Fun (10 points) Explain how to rewrite Eraser to be a binary-level failure oblivious system rather than a race detector. Make sure you say how you would reuse shadow memory so that you could *always* catch errors like this:

```
p = malloc(1024);
p += 10000;
*p = 10;
```

Problem 11: More Binary Fun (10 points) The VMware guys, nervous about the rise of hardware VMMs, want to make their software approach more valuable. One idea: build a variant of Eraser by using their binary translation engine to instrument guest code. Sketch how to do this. What parts of Eraser would be easy for them? What could be a significant obstacle? Give two example of where a VMware-level Erawer could be more powerful than Eraser.

Problem 12: Virtual Machines (15 points)

1. (5 points) ESX forcibly takes a page from a guest OS using paging (i.e., so the guest does not know it no longer has the page). An application on the guest OS references this page. What problem does this cause that similarly comes up with cooperative and/or user-level threading?

2. (5 points) Assume you can define a new hardware exception that the guest OS would receive: explain what this would look like and how would you use it to remove the problem?

3. (5 points) Assume the VMM knows that the guest OS has a `sleep` system call that will put the current application to sleep for a given amount of time. How could VMware use `yield` to potentially fix the problem in a spirit similar to its balloon driver? What are the tradeoffs as compared to defining a new exception type?

Problem 13: Superpages (15 points) Three points each:

1. (3 points) Give two programs in Table 2 that illustrate the problem they ran into with the Alpha's TLB miss cycle counter.

2. (3 points) Table 2: What is weird about FFTW? What type of access pattern must it be doing?

3. (3 points) If you could pick a single superpage size, what would it be and why? How much difference would you expect in their results in Table 1?

4. (3 points) You compute the speedups in Table 1 as an average of the four runs they did rather than discarding the first one: what do you expect to happen?

5. (3 points) Assume you changed the Navarro promotion scheme to allocate an entire reservation's superpage as soon as *any* page in the reservation was allocated. If you reran the experiments in Table 1, what result do you expect?