Stanford University
Computer Science Department
CS 240 Midterm Spring 2013

June 12, 2013

!!!!!! 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
1-5 (25 points)	
6-10 (25 points)	
11-13 (45 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. Eraser: Describe an error that the state machine in Figure 4 will detect, but the text that discusses the state machine (which disagrees with the figure) will not. State your intuition and reasoning!

2. You run:

```
eraser ./a.out
```

   If eraser emits no error messages, does this mean `a.out` has no errors? If it emits an error, does `a.out` have errors? If you then rerun `a.out`, will eraser emit the same errors? Concisely state why or why not, especially for the last question.

3. Boehm: as suggested in class, you define the semantics of a `volatile` variable `v` as giving two guarantees: (1) no additional loads or stores can be done to `v` other than what appear in the program text and (2) an access to `v` cannot be reordered with any other volatile access or lock call. Which problems (if any) in Section 4 would this fix?

4. Boehm: In what way does Figure 3 undercut the entire point of section 5.1? (Please state what this point is.)

5. Livelock: Will screend be more or less susceptible to livelock when running on a guest OS on VMware? Use any relevant data from the "Comparison of Virtualization..." paper to support your argument.

6. Livelock: In *at most 40 words* and ignoring any speed hacks, give the **complete** livelock solution that they implemented in their system (ignore quotas and grammar). Please write the word count by each line of your description.

7. An implicit but overriding principle of the superpage paper is *primum non nocere* ("first, do no harm") in that they try to never be worse than the base system. Give two examples of choices they made that satisfy this principle and one example that does not.

8. Rectangle A states that the superpage guys should have measured the increase in memory footprint from using superpages. Rectangle B states any difference should be negligible. Who is more correct and why?

9. ESX invisibly takes MPN1 from guest OS1; later, a user process running on OS1 writes to a virtual address that maps to MPN1. OS2 takes PPN2 from a user process, which later writes to a virtual address that maps to PPN2. Will OS1's throughput degrade more than OS2's, or vice versa? Please explain your reasoning.

10. ESX, Figure 8: around the 68 minute mark: explain the causal connection between alloc, balloon, and active in (c) and (d). (I.e., which one is driving the others, and the order in which the others influence each in turn.)

Problem 11: Mesa (15 points)
Consider the memory allocation code in the Mesa paper.

1. The paper states this code has a bug. What is it and what is the fix?

2. Intuitively, how would you change this code to work with Hoare wakeup semantics?

3. What happens if we make `Expand` an `ENTRY` routine?

4. Give an example monitor invariant for this code.

5. Give a quick intuition for where priority inversion could come up in this code.

Problem 12: A Comparison of Software and Hardware Techniques for x86 Virtualization (15 points)

Consider the `isPrime` code in Section 3.2:

1. What does the code for `isPrime` look like when run on their hardware VMM?

2. Judging by the resulting translation did `isPrime` run as unprivileged user code or as privileged kernel code?

3. Give one instruction in the rewritten version of `isPrime` that was produced by self-modifying code.

4. Give concrete inputs to eliminate the jumps to `takenAddr` and `fallthrAddr3`.

5. Assume `isPrime` is only called from a single location l and, thus, they replace the return instructions in `isPrime` with a hard-wired jump to the instruction after l. Would this optimization (1) always work, (2) sometimes work, (3) never work?

Problem 13: Native Client (15 points) Consider the code in figure 3:

1. Give two instructions `inst_is_disallowed` would check for.

2. From the code: do direct jumps have to be aligned to 32-bytes? Do they end the 32-byte blocks?

3. What is the attack if you delete the check `Block(StartAddr[icount-2] != Block(IP))`?

4. What happens if you delete the four characters: `else`

5. If you compute `StartAddr - JumpTargets` what do you get?