

# CS 167 Midterm Exam

Closed Book  
October 22, 2008

Do all of questions 1 through 3.

1. Recursion, in the context of programming languages, refers to a function that calls itself. For example, the following is a simple example (assume that it's called only with appropriate argument values):

```
int factorial(int n) {
    if (n == 1)
        return n;
    else
        return n*factorial(n-1);
}
```

Tail recursion is a restriction of recursion in which the result of a recursive call is simply returned — nothing else may be done with the result. For example, here's a tail-recursive version of the factorial function:

```
int factorial(int n) {
    return f2(n, 1);
}

int f2(int a1, int a2) {
    if (a1 == 1)
        return a2;
    else
        return f2(a1-1, a1*a2);
}
```

- a. Why is tail recursion a useful concept? (Hint: consider memory use.)
  - b. Explain how tail recursion might be implemented so as to obtain the advantages mentioned in part a.
2. We have a new architecture for interrupt handling. There are  $n$  possible sources of interrupts. A bit vector is used to mask them: if bit  $i$  is 1, then interrupt source  $i$  is masked. The operating system employs  $n$  threads to handle interrupts, one per interrupt source. When interrupt  $i$  occurs, thread  $i$  handles it and interrupt source  $i$  is automatically masked. When the thread completes handling of the interrupt, interrupt source  $i$  is unmasked. Thus if interrupt source  $i$  attempts to send an interrupt while a previous interrupt from  $i$  is being handled, the new interrupt is masked until the handling of the previous one is completed. In other words, each interrupt thread handles one interrupt at a time.  
  
Threads are scheduled using a simple priority-based scheduler. It maintains a list of runnable threads (the exact data structure is not important for this problem). There's a global variable *CurrentThread* that refers to the currently running thread.
    - a. When an interrupt occurs, on which stack should the registers of the interrupted thread be saved? Explain. (Hint: there are two possibilities: the stack of the interrupted thread and the stack of the interrupt-handling thread.)

- b. After the registers are saved, what further actions are necessary so that the interrupt-handling thread and the interrupted thread can be handled by the scheduler? (Hint: consider the scheduler's data structures.)
  - c. Recall that Windows employs DPCs (deferred procedure calls) so that interrupt handlers may have work done when there is no other interrupt handling to be done. How could this be done in the new architecture? (Hint: it's easily handled in the new architecture.)
  - d. If there are multiple threads at the same priority, we'd like their execution to be time-sliced — each runs for a certain period of time, then yields to the next. In Windows, this is done by the clock interrupt handler's requesting a DPC, which forces the current thread to yield the processor. Explain how such time-slicing can be done on the new architecture.
3. Linux and other Unix operating systems provide the *madvise* system call with which a thread can tell the operating system how it will be referencing memory. For example, a thread can specify that its references to a particular region of memory will be random, meaning that there's no predicting whether any particular page will be referenced or not at any given moment. It can also specify that its references to a particular region of memory will be sequential, meaning that pages are going to be accessed in sequential order.

Explain how the operating system might use this information in both of the above cases with both of the above flags. In particular, how might it affect page-ins and page-outs?

**If you do all of the following correctly, you'll get an A regardless of how well you do on the first three problems. If you miss any of the following, your grade will be based solely on how well you do on the first three problems.**

- 4. The origin of the term *foobar* is disputed. However, one possibility is that it comes from *fubar*, an acronym used by members of the U.S. military in World War II.
  - a. What does *fubar* stand for? (Hint: there's the correct version and the not-so-correct but more polite version. Either is acceptable as an answer.)
  - b. In what was clearly an attempt at humor by the designers of the Digital VAX-11 computer, *fubar* was the acronym for the name of a diagnostic register. What did it stand for here?
- 5. The CIT building was completed in 1988 and CS moved into the fourth and fifth floors. Renovations to the third floor were completed in 2005 and CS expanded into it. Who was the prior occupant of the third floor?
- 6. Who was the computer-industry pioneer noted for having the motto "think"?
- 7. Which Brown alumnus succeeded Steve Jobs as CEO of Apple? (He had absolutely no CS background.)
- 8. Which Brown alumnus was the ninth employee of Microsoft? (He definitely had a CS background.)
- 9. Which Brown CS faculty member has been on the Brown faculty the longest? In which department did he or she start?
- 10. Early CRT-based (CRT stands for cathode-ray tube) computer terminals displayed strictly text and provided a single window. What were the dimensions of the window in terms of number of lines of text and characters per line?