

CS167 Homework Assignment 1

Due October 7, 2009

1. The text, on page 3-10, describes how to switch from one thread to another. The implicit assumption is that the thread being switched to in the switch routine has sometime earlier yielded the processor by calling switch itself. Suppose, however, that this thread is newly created and is being run for the first time. Thus when its creator calls switch to enter its context, it should start execution as if its first routine had just been called. Show what the initial contents of its stack should be to make this happen. Assume an x86-like assembler language, as used in Chapter 3 of the text.
2. 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. (You don't need to supply the assembler code.)
3. Slides VI-6 and VI-7 (from pages 3-23 and 3-24 of the text) show the assembler code of main.s and subr.s. Describe what changes are made to the corresponding machine code (in main.o and subr.o) when these routines are processed by ld, forming prog. Be specific. I.e., don't say "the address of X goes in location Y," but say, "the value 11346 goes in the four-byte field starting at offset 21 in xyz.o." Note that for the x86 instructions used in this example, an address used in an instruction appears in the four-byte field starting with the second byte of the instruction.

4. Many C and C++ compilers allow programmers to declare thread-specific data (Section 2.2.4 of the text) as follows.

```
__thread int X=6;
```

This declares X to be a thread-specific integer, initialized to 6. Somehow we must arrange so that each thread has its own private copy of it, initialized to 6. Doing so requires the cooperation of the compiler, the linker, and the threads library. Describe what must be done by all for this to work. Note that thread-specific data (TSD) items must be either global or static local variables. It makes no sense for them to be non-static local variables. You may assume that TSD items may be initialized only to values known at compile time (in particular, they may not be initialized to the results of function calls). Assume that only static linking is done — do not discuss the further work that must be done to handle dynamic linking. Note that a program might consist of a number of separately compiled modules.