Problem Set 1  
Computer Science 240  
Fall 2014  
Due: Friday, September 12

**Reading.** Patterson & Hennessy Chapter 1

**Exercise 1.1.** The first chapter of Patterson and Hennessy lays the foundation for the rest of the book. It introduces the basic ideas and definitions, places the major components of software and hardware in perspective, and introduces integrated circuits. As the authors point out in their introduction, there is a lot of special terminology and an alphabet soup of acronyms. Patterson and Hennessy Exercises 1.1 will help you learn this new terminology so that we can use it throughout the semester. All terms are defined in the first chapter of the text.

**Exercise 1.2.** Suppose you wish to run a program with $7.5 \times 10^9$ instructions on a 5 GHz machine with a CPI of 0.8.

a. What is the expected CPU times?
b. When you run P, it takes 3 seconds of wall clock time to complete. What is the percentage of the CPU time P received?

**Exercise 1.3.** If benchmark suites are designed to provide a real-world metric for a specific computing task, explain why benchmark suits need to be updated.

**Exercise 1.4.** Consider the following hypothetical news release:

“The company will unveil the industry's first 5 GHz version of the chip, which offers a 25% performance boost over the company’s former speed champ, which runs at 4GHz. The new chip can be plugged into system boards for the older original chip (which ran at 1 GHz) to provide a 70% performance boost”

Comment on the definition (or definitions) of performance that you believe the company used. Do you think the news release is misleading?

**Exercise 1.5.** Write a MIPS assembly language program that prompts the user to enter her name, then prints a personalized hello message. For example, if the user entered “Jean” then the program would print “Hello Jean”. A sample output is shown below (note that the MIPS interpreter added the text **** user input : Jean):

Please enter your name: **** user input : Jean
Hello Jean

-- program is finished running --
You may pattern your code after the *helloWorld.asm* program from the second lecture (see that day’s notes for a copy of this code). The *read_string* syscall uses mode number 8.

You will first need to set aside buffer space in the `.data` section in which to store the string entered by the user. This may be done by using the `.space` assembler directive:

```assembly
str_buffer:
    .space 16
```

This sets aside a buffer of sixteen bytes in memory starting at location `str_buffer`.

However, the `read_string` syscall will only allow you to use 15 of these. A zero value is appended to you’re the end of the input string. The syscall will also append a `LF` character (ASCII code 10, “line feed”) to the end of the string if there is room.

Now to read a string from the user, load the proper sys code into $v0, put the address of your buffer into $a0, the length of your buffer (16 in this example) into $a1, and execute a syscall.

Please use good programming style (comments, indentation, meaningful variable names, etc…). At the beginning of your program include a comment stating whether or not your program works.