CSCI 2321 (Principles of Computer Design), Spring 2013

Homework X

Credit: Up to 30 extra-credit points.

1 Overview

This is a very open-ended list of problems, some involving programming and some not. You can receive at most 30 extra-credit points, but aside from that the amount of credit will depend on the difficulty of the problem. (For example, you might get 5 or 10 points for a simple program addressing one of programming problems, or more for a more complicated program; for one of the non-programming problems you might get 5 points for a page or so of prose, more if you write at greater length or in more depth.) I'm also open to other ideas you might have for extra credit. (One somewhat obvious option is problems in the textbook's end-of-chapter problem set that are more challenging than the ones assigned in class, or that address material we didn't cover.) With regard to turning things in:

- As with previous assignments, anything you send me by e-mail should have a subject line that includes the name or number of the course ("csci 2321", "Computer Design", etc.)
- Include with your prose/program an estimate of how long you spent on it, to help me assess the amount of extra credit to give. I trust you to be honest!

2 Problems

Do as many of the following as you like. You can turn in hardcopy (put it in my mailbox in the department office) or send me by e-mail me something I can print (PDF preferred, but anything I can reasonably print from Linux is okay).

1. In this course we focused on the MIPS architecture and its assembly language because it's simple and regular, and in theory once you have this background you should be well-prepared to learn about other architectures and their assembly languages. Choose some other architecture (x86 comes to mind, but there are others) and write a one-page-or-so executive-level summary of how it compares to the MIPS architecture (e.g., does it also have a notion of general-purpose registers, what if any special-purposes registers does it have, how do (some of) the instructions compare to those used in MIPS, etc.). Include a list of the sources you consulted (parts of the textbook, Web sites, etc.) You can even do this more than once for several different architectures.
2. For testing MIPS assembler programs we used a simple emulator (SPIM). Based on a very quick Google search it appears that there are other tools that provide similar or greater functionality (cross-compilers that generate MIPS assembler or object code from C code. full-fledged virtual machines that implement the MIPS architecture.) Find one or more that seem to you likely to be useful for this course and explain why you think it would be useful and what would be involved in installing it.

3 Programming Problems

Do as many of the following as you like; submit your program(s) by e-mail, with each source-code file as an attachment.

4 Programming Problems

Do as many of the following as you like; submit your program(s) by e-mail, with each source-code file as an attachment.

1. Write a complete MIPS program to do something you think is (at least a bit) interesting and doable. Your program should consist of a single `.s` file, e.g., `mypgm.s`, and should be runnable using SPIM with the command `spim -f mypgm.s`. How much credit you get depends on the difficulty of the problem. Some ideas that occur to me (though I have not tried them so can't be sure how doable they are!):

   - A program to get an integer from the user and prints its two's complement representation.
   - A program to get a sequence of integers from the user (perhaps ending when the entered value is 0) and compute their sum and optionally their average, rounded to the nearest integer.

2. Some of the homeworks and exams had you do things that (should?) seem very automatable. Examples that come to mind:

   - Converting a number in decimal form to a text form of its IEEE-754 representation, or vice versa.
   - Converting a line or lines of MIPS assembler to a text form of its binary representation, or vice versa. (This might take a while if you want to support all instructions, but you could choose to accept a subset, though obviously(?) the more you do the more credit you can get.)

   Write a program in a high-level language to perform one of these tasks (or some other task you had to do as part of a homework assignment or quiz and that you think is similarly automatable). You can use any high-level language I can easily test on one of our classroom/lab Linux systems. Whatever language you used in CSCI 1320 (Scala or Python for most of you?) might be a good choice.