Homework#3 for ECE 152
Instruction Sets (Chapter 2)

Hardcopy is due in class on Wednesday, February 16
(Assembly code must be submitted electronically by 10:00AM on Feb 16!)

All homework must be done individually!

1) [10 points] Patterson & Hennessy 2.29
2) [10] P&H 2.38
3) [10] For a stack based ISA, write (on paper) an assembly program that computes
   \( X = A + (B \times (C+D)+E) \). Assume that you have the following instructions: push, pop, add, multiply,
   where tos="top of stack" and:
   \[
   \text{push } Z \rightarrow \text{tos}++; \quad \text{Stack[tos]} = \text{Memory}[Z]
   \]
   \[
   \text{pop } Z \rightarrow \text{Memory}[Z] = \text{Stack[tos]}; \quad \text{tos}--
   \]
   \[
   \text{add } \rightarrow \text{Stack[tos]} = \text{Stack[tos]} + \text{Stack[tos-1]}; \quad \text{tos}--
   \]
   \[
   \text{multiply } \rightarrow \text{Stack[tos]} = \text{Stack[tos]} \times \text{Stack[tos-1]}; \quad \text{tos}--
   \]
4) [10] What MIPS assembly instruction is represented by 0x00644022?
5) [5] Create two “mechanical” questions that I could use on a test of the material covered
   in Chapter 2 and in the Instruction Set lecture notes. Do not make both questions nearly
   identical to each other. By “mechanical”, I mean a question that is concrete and possibly
   quantitative, but not something that just requires memorization. The purpose of this ques-
   tion is to get you to think about what kinds of questions you will need to be able to answer
   to consider yourself proficient with the details of the material. I will give full credit for all
   reasonable questions. You may NOT use questions from another textbook or another class
   or any other source - if you have any questions about this requirement, please ask.
6) [10] Create two “conceptual” questions that I could use on a test of the material cov-
   ered in Chapter 2 and in the Instruction Set lecture notes. By “conceptual”, I mean a ques-
   tion that requires a deep understanding of the material and the ability to synthesize this
   material and explain it in a new way. The purpose of this question is to get you to think
   about the important concepts that we have covered. I will give full credit for all reasonable
   questions. You may NOT use a question from another textbook or another class or any
   other source.
7) [20] Write a MIPS assembly program that produces the first 20 Fibonacci numbers.
   Use the sim program simulator (available on the dsil computers) to run and test your assembly
   program. Spim (and xspim) is a program that simulates the behavior of MIPS32 comput-
   ers and can run MIPS32 assembly language programs. Documentation for spim is avail-
   able in Appendix A of your textbook and at: \url{http://www.cs.wisc.edu/~larus/spim.html}. This spim
   website also contains a link for downloading a PC version of spim, if you’d rather run it on
   your PC than on a dsil workstation. A helpful reference is a simple program that I’ve pro-
   vided for you at: \url{http://www.ee.duke.edu/~sorin/ece152/resources/simple.s}. This simple program
   sums the entries in a list of 9 integers.
To submit your code for this question and the next question, create a directory in your
ECE account. You can name this directory whatever you want, but for this explanation I’ll assume it’s named ~yourlogin/homework3. Put your program files in this directory and name them question7.s and question8.s. Then, following the same instructions as for homework #2, create a .tar.gz file called homework3.tar.gz that includes question7.s and question8.s, and then use the electronic submission website to upload homework3.tar.gz. **You may re-submit as often as you like, but a re-submission will overwrite whatever you’ve previously submitted for this assignment. I will grade whatever has been submitted before 10:00AM on Wednesday, February 16.**

8) [30] Write a MIPS assembly program that can search an array of 100 characters and report how many times the sequence “Duke” appears in it. Once again, use spim to run and test your code. Your program will be tested by the grader with several different arrays of characters, to make sure that your program does indeed work correctly.