CSE 378 Spring 2005

Machine Organization and Assembly Language Programming

Problem Set #2

Due: Wednesday April 13th

By the due date, you should have read Chapter 1, Chapter 3 (Sections 3.1 to 3.3), Chapter 2 (Sections 2.1 to 2.9). It would not be a bad idea to read also Sections 2.10, 2.13, 2.15 through 2.18. You should also read Appendix A1, A2, A5, A6, A9, A10 (not every line of the latter; this is SPIM's "reference manual").

1. Chapter 2. Ex 2.30 How many times will the line of code at label "outer" be executed? How many times will the line of code at label "inner" be executed? Give the result as a function of n, the number of elements in the array (in the exercise $n = 2500$). If X and Y are the arrays, what are the maximum and minimum number of times that the instruction "add $v0,$v0,1" (the one before the instruction labelled "skip") will be executed if all elements of X are different and all elements of Y are different. If we allow values to be repeated in X and in Y, what is the maximum number of times this instruction can be executed?
2. Chapter 2 Ex.2. 34 You are encouraged to use SPIM to find the errors (both "logical" and "syntactical"). In fact, SPIM might "self-correct" some of the errors.
3. For your own enlightenment do Exercise 2.37 using SPIM but do not turn it in.
4. Programming in SPIM. Chapter 2 Ex 2.21 (on the CD). Be sure to test your program with strings that generate an error. The resulting value, either a positive integer or the value "-1" in case of an error should be displayed on the SPIM console.
5. Programming in SPIM. Write a program in MIPS assembly language that finds the number of elements strictly greater than the first element, the number of positive elements and the number of negative elements in an array of integers (each element is a 32-bit positive, negative, or null integer). Your input, the array and its size, should be in the ".data" section of your program.

   As output, your program should display the 3 values asked for above on the console with appropriate messages.
6. After reading the "Compact Code and Stack Architecture" on the CD 2.19.3-4 and "In More Depth" CD-IMD-2.20-7-8 relative to stack architectures, repeat "on paper" the last exercise if it had to be programmed on a stack machine. You do not have to write the part relative to the output on the console.

   You can assume that there is a local area of storage that contains the array, its size, and locations to store the 3 values you are asked for etc. as well as any temporaries/constants that you might need. You can also assume that if you need constants, they have been initialized in that local storage.

   You can be extremely vague in your addressing, i.e., use instructions like:

```
Push            array[first]  #put first element of array on top of stack
Pop             negs          #i.e, store top of stack in ``negs'' and pop
Pop                           #remove top of stack
Ifeq            target        #if top of stack = 0 branch target. Pop (in either case)
```

Useful instructions will be Push, Pop, all arithmetic-logical operations that you need, all relational operators associated with transfer of control (ifeq, ifne etc.). Another operation that might be useful is “dup” (duplicate the top of stack).

A future programming assignment will be based on the stack architecture concept so this exercise is important.

You should design your own test files for the programming assignments but you won’t have to turn them in. We’ll have our own!

A good template for SPIM programs can be found under the “Software” section in the CS378 homepage or at:
http://www.cs.washington.edu/education/courses/378/05sp/handouts/template.spim

Instructions for **Turnin** for the SPIM programming will be given to you shortly.