CSE 378 Spring 2010 – Homework 1: Getting Started with MIPS

Due: Wednesday April 14th, 2010 at 5pm
You may use up to 2 late days on this assignment.

The objective of this first assignment is to write a simple program which implements 2 simple functions in the MIPS assembly language and run it through SPIM to verify their workings. To satisfy this purpose you will write an assembly version of **i.) the rot13() string encryption function** and **ii.) the itoa() integer to string function**, insert them into a provided SPIM template, and execute the program in SPIM.

Writing Assembly language programs

There are a few things that you should keep in mind when writing an assembly program. The first is to write good comments as you write your code, rather than inserting them when you are done with the program. This will make it much easier for you to track down problems as you are writing, and will save you a lot of time when debugging. Leave the comments in the program when you turn it in, as this will make it much easier for the TAs to figure out what you were trying to do with your code, and will be essential for getting partial credit if necessary.

Simplicity is key when writing assembly programs. There will often be ways to write programs that are more compact or faster than the straightforward solution, but you should focus on getting a working program rather than an optimized program. Always make sure that you have a working version to start from when you begin trying to optimize your program. Remember, an optimized program that fails to perform the required tasks is going to get a lower score than an unoptimized program that succeeds. If you do choose to work on optimizing your programs, be sure to clearly comment and explain your code where necessary, as the purpose of some optimizations is not immediately apparent.

i. Writing a string manipulating function

The purpose of the rot13() function is to rotate each character of an ASCII-encoded string forward by 13 places in the alphabet, while preserving case. Hence 'A' becomes 'N', 'z' becomes 'm', and 'R' becomes 'E'.

String representation

Strings are represented as contiguous bytes of memory, where each byte is an ASCII character, followed by the NUL byte/character (0x00).

Interface

Your function will be provided with the memory address of the first character of the string to perform the rot13 substitution cipher on via the a0 register. Your function will modify the strings characters and return a pointer to the first character of the modified string. You may assume that we will only give you strings containing alphabetic characters.

Handling Case

You will need to ensure that alphabetic case is handled appropriately. That is, applying rot13 to the string “O” should yield “B”, not “\”. You should leverage the numerical representation of ASCII characters which is provided to you on the back side of your MIPS green sheet for help with handling the case of characters.

ii. Writing an integer to string comparison function

The itoa() function converts a signed, 32-bit two's-complement integer to a null-terminated string. Negative integers are indicated by a leading minus sign. For example, the number 1234 becomes “1234\0” (where '\0' is the terminating NUL character).

Interface

Your function will be provided the integer to convert in register a0 and a pointer to an allocated block of memory big enough to handle the longest string representation of a signed, 32-bit integer (“-2147483648”) in register a1. Your function will convert the integer to a string and return the pointer to the first character of the string representation.

Division

Division in the MIPS assembly language is a more complicated operation than the standard add and sub arithmetic instructions. The div instruction only takes rs and rt fields, making it of the format 'div $rs, $rt'. As specified in the core arithmetic instruction section on the green sheet, the instruction places the result of the division into the lo register and the remainder into the hi register. To get the values from these two registers, you will need to use the Move From Low (mflo) and Move From Hi (mfhi) instructions, which only take a destination register.

What you need to do

1. Download the SPIM template from the MIPS resources page: http://www.cs.washington.edu/education/courses/cse378/CurrentQtr/hw1-template.s
2. Write the string encryption function rot13() and the integer to string function itoa(), observing the given interfaces. The template code gives a strong indication of where to place your implementations.
3. Load your program into SPIM and execute it. When your program is finished executing you should see the result of applying rot13() to a string, the result of applying rot13() to the encrypted string, and the result of your itoa() function on the SPIM output console.
4. When you are satisfied with your solution, turn in the assembler (.s) file. This file should include the provided code and your implementations of rot13() and itoa(). Be sure to place your name into the comment area at the top of the .s file.
5. Submit your assignment to the Catalyst CollectIt tool for this assignment linked to the 378 course web page (http://www.cs.washington.edu/378).