CSE 378: 09 WI - Ceze

Homework 1: Comparing Encoded Strings

Due: Tuesday, January 20, 2009 at 5:00 PM.

The objective of this first assignment is to write a program in the MIPS assembly language and run it through SPIM to verify that it works. You will write an assembly version of a function that determines whether one string is an *encoding* of second string, we call this function strencodecmp(s1,s2) . We will use a simple naïve encoding.

Writing Assembly language programs

There are several 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.

Secondly, you should follow proper conventions when writing your program. For this assignment, we will focus on register usage conventions. Each register has a name associated with it and a purpose, and it is good style to follow these conventions. For example, the register a0 is meant to be used to pass arguments to a function, while the register t0 is used for temporary storage of data and could be erased by a called function. A list of register names and uses can be found on the green card included with the textbook.

Last but not least, simplicity is key when writing your 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.

Writing an encode compare function

The purpose of **strencodecmp(s1,s2)** is to compare two strings in memory, character-by-character, and determine whether the second string **s2** is an encoding of the first string **s1**.

The encoding is as follows: replace each letter in the string with the next letter in the alphabet. In other words, add 1 to every letter in the string (an 'a' becomes a 'b', a 'b' becomes a 'c' and so on). Encoding of ‘z’ should be ‘a’, and encoding of ‘Z’ should be ‘A’. Do not worry about numbers, assume the string only contains letters. For example, “abCd” encoded would become “bcDe”.

String representation

Strings are represented by contiguous bytes in memory (each byte is an ASCII character) followed by the NULL character (0x0).

Interface

Your program will be provided with the memory address of the beginning of strings **s1** and **s2** in registers a0 and a1 respectively. If **s2** is an encoding of **s1**, the function should return **0**. Otherwise, it can return any value different than 0.

What you should do:

1. Start by using the template SPIM file on the MIPS resources page: http://www.cs.washington.edu/education/courses/cse378/resources/template.spim

2. Write the encode string comparison function observing the given interface. You should make this function a separate routine from main. That is, main should be able to call your function (using jal).

3. Test your program. You can do this by adding some sample strings to the top of your file and modifying main to set up the registers $a0 and $a1 to point to those strings and then invoke your routine. Load your program into SPIM and execute it. You can verify that your code works based on the value in register $v0, which you can view by either printing it out with a syscall or setting a break point at the end of your routine.

4. When you are satisfied with your solutions, you will turn in the assembler (.s) file for your strencodecmp(s1,s2) function. Remember to comment your code!

5. Submit your assignment via the Catalyst WebTools at: https://catalysttools.washington.edu/collectit/dropbox/iannacci/4564 Please include your name at the top of this file in a comment.