CSE 378 10wi Homework 1: Getting Started with MIPS

Due: Friday, January 22, 2010 at 5pm

The objective of this first assignment is to write 2 basic programs in the MIPS assembly language and run them through SPIM to verify that they work. For this purpose, you will write an assembly version of **(a) the strcmp() function** and **(b) the atoi() function** and simulate execution in SPIM.

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.

a. Writing a string comparison function

The purpose of strcmp() is to compare two strings in memory, character-by-character, to see which comes first in the standard lexicographic order.

String representation

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

Interface

Your function will be provided with the memory address of the beginning character of the first string in register a0, and the memory address of the beginning character of the second string in register a1. If the first string is greater, return a positive number. If the second string is greater, return a negative number. If the strings are equal, return 0.

Lexicographic ordering

The normal strcmp() compares two strings character-by-character, according to the raw ASCII values of the characters. This ordering is fairly intuitive; here are some examples:
"*a*" < "*b*"
"*abc*" < "*abcd*"
"*A*" < "*a*"

b. Writing a string to integer conversion function

The atoi() function parses a string *str*, interpretes its content as an signed integral number, and returns the result as a 32-bit 2's complement binary integer. The function takes an optional initial *plus* or *minus* sign followed by as many numerical digits as possible, and interprets them as a numerical value. The string may contain additional non-digit characters after those that form a integral number. The conversion should be stopped immediately, and only the integral number should be returned. If *str* is not a valid integral number, no conversion is performed. You can ignore overflow.

Interface

Your function will be provided with the memory address of the beginning character of *str* in register a0. Return the int value of *str*. Return 0xdeadbeef if *str* does not represent a valid

integral number.

Testing

Here are a few possible test cases you could consider:
1234
-423
34.77
$hu231n$
You should come up with more test cases so that you fully test the function.

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/CurrentQtr/template.spim
2. Write the string comparison function strcmp() and the string to integer conversion function atoi() observing the given interfaces. Try to write the C version first if you have trouble with the assembly code.
3. Load your program into SPIM and execute it. Write a main function that tests your strcmp() and atoi() with several different inputs. Some credit will be awarded for the quality of your tests. Do not make it excessively redundant!
4. When you are satisfied with your solution, you will turn in the assembler (.s) file. This file should include the main function, the strcmp function and the atoi function. Please include your name at the top of the .s file in a comment.
5. Submit your assignment via the Catalyst WebTools at:
https://catalysttools.washington.edu/collectit/dropbox/perkins/8464