Computer Architecture Laboratory

MIPS Assembly Language Programming - II

Goals:

a) Learn assembly language subroutine methodology
b) Understand MIPS procedure call conventions
c) Write a subroutines in MIPS assembler

1. As a group we will examine the way MIPS uses registers, memory and the stack as it relates to subroutine calls. These are sections 3.6, A.5 and A.6 in our text.

2. Write a subroutine, *find,* in MIPS assembly language. The subroutine should have two arguments. The first is a character to be searched for; the second is a pointer to a null-terminated string (use registers $a0 and $a1 respectively). The *find* routine should locate the first instance of the sought-after character in the string and return its address in register $v0. If that character does not exist in the string, then *find* should return a pointer to the null character at the end of the string. For example, if the arguments to *find* are the letter "b" and a pointer to the string "imbibe," then the return value will be a pointer to the third character in the string. Thoroughly test your procedure. (Hint: see problem 3.23) Be sure the subroutine correctly handles the situation where the character is not in the string.

3. Write a subroutine, *count*, in MIPS assembly language. The *count* procedure has two arguments. The first is the character to be counted, and the second is a pointer to a string in register (using registers $a0 and $a1 respectively). The subroutine returns a count of the total number of times the character appears in the string in register $v0. You must use your *find* subroutine from part 2 of this lab to find the next occurrence of the character. (Hint: see problem 3.24) The routine should accept strings of up to 255 characters.

4. Imbed subroutine *count* in a program that interactively gets one character and the string to be searched as data from the console and prints the results to the console. Inputs should have prompts; outputs should have explanations. Mail me the assembly language test program (named *your_last_name-2.asm*). I will execute the program in SPIM to see if it has done the job correctly. The program should be well commented so that I am able to understand your logic. Answers to the following questions will determine the grade for this lab.

1) Did the program execute?
2) Did it return a correct value?
3) Is it readable?
4) Is it written efficiently?

N.B. For an overall structure, this program should have subroutine *find* embedded in subroutine *count* imbedded in test program *your_last_name-2.asm*.

Hints:

1. Modularize this project or it will get totally out of hand
2. Prototype your algorithms in C before writing them in assembler.
3. See the various syscalls for console I/O