CSE 410 - Spring 2004

Homework 2

due on Wednesday, April 14 at 9:30 AM, at the beginning of class

47 points

Name Solution DWJ

Student # _____

This homework is intended to help you understand the calling conventions for transferring control to procedures and returning results from procedures.

There are two source files provided to you. There are some questions in the written section of the homework that refer to the code as supplied, and then there are also procedures that you will write and add to the existing skeleton code.

You will turn in this paperwork with your written answers in class on Wednesday. Also, before class, you will turn in your source files and the log files that show how the programs operate.

Consider the program v2.s. As provided to you, this program is the beginning of a small library of procedures for manipulating two-dimensional vectors (ie, 2-element arrays with an integer x coordinate and an integer y component). The main procedure and the v2Set procedure are provided; you will write v2Print, v2Add, and v2Equal as described later.

The following discussion and questions may be easier to follow if you have the file open in Context and SPIM while looking at the sources.

The purpose of the main procedure in v2.s is to utilize the library procedures as they might be used by any application program that was using this library. Main prints out a little header information, and then sets up for and executes a loop over all the 2-element vectors stored in the coords array, printing a vector, adding delta, then printing the result. Following that, it sets up for and executes another loop over the array, this time printing out all the elements that are not equal to the origin (0,0).

1. (2 pts) Take a close look at the main procedure. Is it a leaf procedure or a non-leaf procedure? non-leaf

2. (2 pts) The main procedure uses a stack frame. How many words are reserved for the frame? 32 bytes = 8 words

3. This program prints out several null-terminated strings using the syscall instruction with op code 4, print_string. Notice that the syscall instruction does not change any registers during its execution, and so all the registers, even the temporaries like $t0 and $t1, are preserved across system calls.

a. (2 pts) How many null-terminated strings are defined in the data section of this program? 6

b. (2 pts) Not including the op code in $v0, how many arguments does the print_string syscall instruction take? 1

4. (2 pts) Read through the first loop in the main program. The top of the loop is at label mainLoopA. The bottom of the loop is the bne instruction at line 95. Consider the condition that the bne instruction is checking and notice how the values involved are calculated. Given the data in the coords array, how many times will this loop execute? 4

5. (2 pts) In v2.s, there is a line that names the author of the program. Change it so that it has your name instead of mine.

6. (5 pts) As provided to you, the code runs but does not produce any output because several of the library procedures are not implemented. The first method that you should implement is v2Print. This procedure takes one argument in $a0, the address of one 2-element vector, and prints the vector to the console as two comma-separated numbers. In order to do this, it uses the print_integer syscall (code 1) and the print_string syscall (code 4), and strComma, a null-terminated string. Implement the v2Print procedure by adding the appropriate code to v2.s.

7. (5 pts) The second procedure to implement is v2Add. This procedure adds the elements of two vectors and places the result in a third vector. The procedure takes three arguments in $a0, $a1, and $a2. The first two arguments are the addresses of the source vectors to add, and the third argument is the address of the vector in which to store the result of the vector addition. Implement the v2Add procedure by adding the appropriate code to v2.s.

8. (5 pts) The last procedure to implement is v2Equal. This procedure takes two vector arguments and compares them. If the contents are equal, then the procedure returns 1 in $v0, if they are not equal then the procedure returns 0 in $v0. Implement the v2Equal procedure by adding the appropriate code to v2.s.

9. After you have completed and checked out the changes described above, run the program. The console output should look like the following (with your name instead of mine):

```
Author: Doug Johnson

Vector Addition

0,0 -> 3,7
10,0 -> 13,7
10,10 -> 13,17
0,10 -> 3,17

Vector Equality

10,0
10,10
0,10
```

Save the SPIM log file. Using the SPIM menu bar, select File -> Save Log File. Save the log file under a meaningful name like v2-PCSpim.log so that you can turn it in along with the v2.s source file later.

Now consider the program bar.s. As provided to you, the main procedure of this program loops over a list of null-terminated strings and prints out the non-empty strings. However, in many analysis tasks we would like to see a more general representation of the data and not the specific characters. So your task is to add two new procedures to this program that will print a row of hyphens "-" for each string, with as many hyphens as there are characters in the string. A line is printed for every string, including the empty string.

There are two new aspects to this program. First of all, the data structure that it uses is an array of string addresses. In other words, the content of array `strings` is a list of pointers to null-terminated strings. Contrast this with the array `coords` in v2.s where the content of the array was the actual 2-element vectors. This is a little more complex to use, but it is a very common and useful way to manipulate strings or other large data objects.

Secondly, one of the procedures you will write is a non-leaf procedure. In contrast, all of the procedures in v2.s were leaf procedures. Thus you will need to manage the stack pointer and stack frame when you write printStringBars.

10. (2 pts) The primary data structure that this program uses is array `strings`. How many address entries are there in the `strings` array (including the 0 address that terminates the list)? 6

11. (2 pts) Each individual entry in the `strings` array is an address. How many bytes are dedicated to a single entry in this array? 4

12. (2 pts) Each individual entry in a null-terminated string is a single ASCII character. How many bytes are dedicated to a single character in one of these strings? 1

13. (2 pts) The stack frame that is established by the `main` procedure is 8 words long. In lecture, I said that there were three reasons for using space in the stack frame: saved registers, local variables, and argument build area. The 8 words in this stack frame are needed for two of these three reasons. What are the two reasons that we need this stack frame?
saved registers and argument build.

14. (2 pts) In bar.s, there is a line that names the author of the program. Change it so that it has your name instead of mine.

15. (5 pts) As provided to you, the program prints the strings, but it does not print the rows of hyphens. Your task is to implement procedures printStringBars and printBar.

    The printStringBars method is given an array of string addresses and loops over the array, calling printBar for each string until it encounters a 0 address.

    Since printStringBars calls printBar, it is a non-leaf procedure and it must have a stack frame to hold the saved return address, an argument build area, and any other registers that must be saved and restored. You can copy and paste much of the stack management code from the main method, modifying it as needed if you use a different set of registers.

printStringBars loops over all of the strings in the array in a fashion very similar to the main method. However, it does not skip empty strings. You can copy and paste much of the looping code from the main method, with appropriate changes to account for the use of $a0 to pass the array address and to eliminate skipping over empty strings. And of course modifying it to call printBar instead of printing the actual string.

If you implement printStringBars before printBar, you could do something very simple in printBar just to show that it has been called and thus checkout printStringBars before moving on to printBar.

16. (5 pts) The printBar method is given a null-terminated string and prints one row of output: a colon character ":", followed by as many hyphens "-" as there are characters in the string, followed by a newline character "\n". The string may be zero length (ie, the first character of the string may be 0).

    printBar prints a colon ":" character each time it is called. Then it loops over all the individual characters of the given string, printing a hyphen "-" for each character, until it reaches a 0 byte. Then it prints a newline character "\n" and returns to the caller.

    Use the Load Byte Unsigned instruction lbu to load a byte from memory into a register in order to check if it is a valid character or the 0 string terminator.

    Since printBar is a leaf procedure you don't need a stack frame to save and restore $ra nor to be an argument build area. If you decide that you need to save and restore registers, you can use a stack frame.

17. After you have completed and checked out the changes described above, run the program. The console output should look like the following (with your name instead of mine):

```
Author: Doug Johnson

All non-empty strings.

This is a string.
This is another string.
This is another string.
This is a string.

String bars.

:-----------------
:-----------------------
:
:-----------------------
:-----------------
```

Save the SPIM log file. Using the SPIM menu bar, select File -> Save Log File. Save the log file under a meaningful name like bar-PCSpim.log so that you can turn it in along with the bar.s source file.