CSE 2304 Lab 3

2/22/2016

Due: 11:59PM, Mon 3/7/2016

The objective of this lab is to teach you how to use subroutines and the stack.

Problem 1 – Using Subroutines: In Class

Subroutine implementation in MIPS is as simple as labeling the starting point of the chunk of code where your subroutine begins and call *jr* at the end of your subroutine to return. Use the instruction *jal* to invoke your subroutine. When you are using subroutines, keep in mind that registers $a0…$a3 should be used for arguments (if you need to use more than 4 arguments, use the stack); registers $v0 and $v1 should be used for return values; if you plan to use any saved registers ($s0… $s8) within your subroutine, you would have to save their data in memory (preferably in the stack) and restore their values before returning your subroutine. The following program illustrates the implementation of a simple subroutine (greater) which takes two arguments (argument values are in $a0 and $a1). If $a1 is greater than $a0, 1 is returned ($v0 =1). If $a1 is equal to $a0, 0 is returned, otherwise -1 is returned. Try to run this sample code in class.

```
# CSE 2304 Lab 3
# Problem 1
      .data # Data declaration section
str1:       .asciiz "Please Enter an Integer: "
str2:       .asciiz "Output: "

      .text

main:       # Start of code section

      li $v0, 4          # system call code for printing string = 4
      la $a0, str1       # load address of string to be printed into $a0
      syscall            # call operating system to perform print operation

      li $v0, 5          # read in integers
      syscall
      add $a1, $v0, $zero

      li $v0, 4
      la $a0, str1
      syscall
      li $v0, 5          # read in integers
      syscall
      move $a0, $v0

      jal greater        # Subroutine call
      move $t0, $v0            # Save result here.

      li $v0, 4
      la $a0, str2
```

```
        syscall
        li $v0, 1    # system call code for print_int
        move $a0, $t0
        syscall

        li $v0, 10          # exits program
        syscall

#Implermentation of greater subroutine
greater:
        bgt $a1, $a0, gt
        beq $a1, $a0, eq
        #Here we know that $a0 < $a1
        li $v0, -1
        jr $ra

gt:     li $v0, 1
        jr $ra

eq:     li $v0,0
        jr $ra

# END OF PROGRAM
```

In Class: You may work in group. ***(Should be completed in lab.)***

Problem 2 – Work alone: Let's create the problem where only two registers are available for you to make a computation. Using integer inputs: a and b from the user, solve the following problem using only the stack and registers $t0 and $t1.

$a^2 - ab + 8a - 10b + 19$

Hints:

To Push
subu $sp,$sp,4 # Move stack pointer
sw $t0,($sp) # store contents of $t0 at ($sp)

To Pop
lw $t0,($sp) # pop a char off the stack
addu $sp,$sp,4 #Move stack pointer

You may also want to set the first value in the stack to 0 to have a value to check for empty stack.

Problem 3 – Work alone: In oder to better manage stacks, it is wise to provide at least 2 function interfaces to add (Push) and remove (Pop) stuff from the stack. In this assignment you will write the subroutines PUSH and POP to implement a Stack. When you invoke the PUSH subroutine, the value to be added/pushed to the stack should be in

register $a0. Your POP subroutine should remove/pop a value from the top of the stack and return this value in $v0, if the stack is not empty. You should also implement a subroutine ISEMPTY that returns a 1 in $v0 if it is empty, a 0 in $v0 if not empty.

Problem 4 – Work alone: Use your stack to write a subroutine ISPAL to determine if a string (alphanumeric, including space) is a palindrome. It should work as follows:

1. Read a string from the console. (Reserve a memory buffer that can take at least 80 characters)
2. PUSH each character of the string onto the stack (Remember a character only uses 8 bits of memory space)
3. Go through the string again character-by-character. For each one, see if it is the same as the character you get when you pop the stack. If all are the same it is a palindrome.
4. If it is a palindrome, return 1 in $v0, otherwise return 0 in $v0.

Deliverables: You are expected to complete **Problem 2, 3, and 4** alone and submit both a softcopy of your MIPS source code and a one-page report discussing your design and implementation, any difficulties you have encountered to complete this lab, and what you think you have learned. In your MIPS code, write your name, class and section number in the first lines as comments. All of your source code files should be saved either as a .s file or .asm file. All files should be zipped into one. The zipped file should be named in the format: Firstname_Lastname_Lab#.zip.

Note: You should be ready to demo your program in lab. No late submission will be accepted.