CS F342 Computer Architecture

Lab Sheet 6

Semester 1 – 2019-20

Goals for the Lab: We build up on prior labs and explore basics of functions and recursion.

Background:

- Calling a subroutine is between a *caller*, who makes the subroutine call, and the *callee,* which is the subroutine itself.
- The caller passes arguments to the callee by placing the values into the argument registers **$a0-$a3.**
- The caller calls **jal** followed by the label of the subroutine. This saves the return address in **$ra**. The return address is PC + 4, where PC is the address of the jal instruction. If the callee uses a frame pointer, then it usually sets it to the stack pointer. The old frame pointer must be saved on the stack before this happens.
- The callee usually starts by pushing any registers it needs to save on the stack. If the callee calls a helper subroutine, then it must push $ra on the stack. It may need to push temporary **($t0-$t7)** or saved registers **($s0-$s7)** as well.
- Once the subroutine is complete, the return value is placed in **$v0-$v1**.The callee then calls **jr $ra** to return back to the caller.

Exercise 1 – With Sample Code: Study the code given below.

```
.data
   line: .asciiz "\n"
   print_str: .asciiz "value of $s0: "
   .text

   main:

   li $a0,10
   la $a1, print_str
   la $a2, line
   jal increase_the_value #function to increase the value of $s0 by 10 and print

   jal print_value #function to print value stored in $a0

   li $v0,10
   syscall


# cont .. to next page
```

```
increase_the_value:
    addi $sp,$sp,-8 #4 bytes for each value
    sw $a0,($sp) #call by value
    sw $ra,4($sp) #since we are having nested procedure, which will overwrite the current value of $ra

    addi $a0,$a0,10

    jal print_value #print_value is a nested procedure

    lw $a0,($sp) #restore the value of $a0, main function should get old value of $a0
    lw $ra,4($sp) #restore the value of $ra
    addi $sp,$sp,8

    jr $ra

print_value:
    addi $sp,$sp,-4 #Since $a0 will be used to print the string, its original value would be lost
    sw $a0,($sp) #saving the original value of $a0(as received by this procedure)
#since we are not calling any other procedure in this procedure value of $ra wouldnt change, hence no need to store it in stack
    move $a0,$a1
    li $v0,4
    syscall


    lw $a0,($sp)
    li $v0,1
    syscall

    move $a0,$a2
    li $v0,4
    syscall

    lw $a0,($sp)
    addi $sp,$sp,4

    jr $ra
```

Exercise 2: Write a function to count the number of vowels in a given string and also return the string after removing the vowels and print that string in main function. Call the function twice with two different strings.

```
Input          : String (without space)

Output         : Single integer
```

Exercise 3: Write a program that asks if the user wants a triangle or a square. It then asks the user for the size of the object (the number of lines it takes to draw the object). The program then writes a triangle or a square of stars "*" to the console.

```
*******
*******
*******
*******
*******
*******
```

or

```
*
**
***
****
*****
******
```

Write a subroutine for each figure. In them, use a subroutine **print_star_line** that writes a line of a given number of stars. (that number is passed as an argument to **print_star_line** function).

Take home assignment:

Print the pyramid as:

```
      *
     **
    ***
   ****
  *****
 ******
*******
```

Exercise 4: Find Factorial of a given integer recursively**.** Take care of the base case.

Exercise 5: Disassemble the following hex instructions.

- `02002009`
- `03e00008`
- `0c100013`

References:

http://stackoverflow.com/questions/18991655/how-do-you-perform-a-recursive-operation-in-mips-assembly

http://stackoverflow.com/questions/22317560/mips-assembly-removing-vowels-from-an-input-string