**Name:** Nick Petro **CoE 0147**

# Lab 5, Spring 2015

Using Stacks and Recursive Functions

**Important:**

- While collaborations are allowed, names of all the collaborators must be listed in the assignment. Each student should submit individual assignments for grading, despite collaboration.
- Solutions (***.asm files and text files if any**) for this assignment should be submitted at `Courseweb/CourseDocuments/Recitations/Lab5` by Feb-23 midnight.
- Each program of the assignment must be in a separate file. You must write your name in the first line of every .asm file in the following format.
  #Name: First_name Last_name
  #rest of the file...

In this lab, you will first write a few (non-recursive) functions that use the stack. In the second part, you will have to write recursive functions. Functions must comply with MIPS calling conventions, such as:

1. Pass parameters via the `$a*` registers.
2. Put the return value in `$v*` registers.
3. Save registers on the stack as necessary (i.e., if your function uses any `$s*` registers, it should save and restore `$s*` registers using a prologue/epilogue).
4. Save `$ra` on the stack, if the function is a non-leaf function.

## 1 Using Stacks

In this part, we will write a string manipulation function. Recall that strings are ASCII-encoded bytes that are terminated with a NULL character (0x0). You are required to write a program that randomly replaces 4 characters with asterisks—'*' (0x2A). The input string is atleast 10 characters long and atmost 63 characters long (excluding the NULL character). Additionally, it will not contain any '*' when it is entered by the user. You can declare a buffer as follows to store the input string from the user:

```
.data
input_str: .space 64
```

Once you receive the string, you will have to find it's length, $N$. To randomly replace a character of this string with '*', you will have to generate a random number $x$ in the range 0 to $(N-1)$. This $x$ will be used as the index for the next character to be replaced. The procedure of generating a random number and replacing a character with '*' should be done 4 times. *Note:* If the generated random number points to a character that is already '*', then you need to regenerate a random number unless it points to a character that is not '*'. Make sure there are four random '*'s in your output. For the same input string, the output string can be different each time you run the program because of the randomness. The output format is shown in the sample output:

```
   Enter your string?
University of Pittsburgh ← input from user (no need to print)
Here is your output:
Un*versity*of Pi*tsbu*gh
```

Because of the random indexes, another output of the same program could be:

```
   Enter your string?
University of Pittsburgh ← input from user (no need to print)
Here is your output:
Unive**ity of Pi*tsbur*h
```

For submission, name your solution file as lab5part1.asm. Please do not forget the '.asm' extension and do not use any upper-case letter in the filename.

## 2 Recursive Functions (Fibonacci Sequence)

The Fibonacci sequence is a special sequence of numbers discovered by mathematician Leonardo Fibonacci. In the Fibonacci sequence, each number is the sum of the two previous numbers. Its function can be defined as following:
$Fibonacci(n) = Fibonacci(n-1) + Fibonacci(n-2), if\ (n > 2)$
$Fibonacci(n) = 1, if\ (n <= 2)$

Example:

| | |
|---|---|
| $Fibonacci(1)$ | 1 |
| $Fibonacci(2)$ | 1 |
| $Fibonacci(3)$ | 2 |
| $Fibonacci(4)$ | 3 |
| $Fibonacci(5)$ | 5 |
| $Fibonacci(6)$ | 8 |

The Fibonacci sequence function can expressed recursively, as shown in the following pseudo-code:

```
int Fibonacci(int n) {
if (n <= 2) {
     return 1; //The base case stops the recursion
}
else {
     return Fibonacci(n-1)+Fibonacci(n-2); }
}
```

Base your solution on the pseudo-codes above, translate the Fibonacci function into MIPS assembly. **Your code must call itself recursively (TAs will explicitly check for 'jal' instructions inside the function).**

```
   Enter the sequence index?
8 ← input from user (no need to print)
The Fibonacci value is:
21
```

Your function must find $Fibonacci(n-1)$ and $Fibonacci(n-2)$. Call a function that adds the two results together and return the sum. The Fibonacci function will save/restore any `$s` registers that it defines.

For submission, name your solution file as lab5part2.asm. Please do not forget to put the '.asm' extension and do not use any upper-case letter in the filename.