Systems I
Lab 7
Procedures in assembly

This assignment is due Sunday, Apr-12, 11:59 pm

1 The assembler/simulator

Once again, we will use the *Venus Simulator*, which you can access in your web browser at:

`https://venus.cs61c.org`

Also again, this reference of RISC-V instructions may be useful:

`https://marks.page/riscv/`

2 Three procedures

2.1 Counting instances of a value in an array

We will once again work with arrays. Specifically, you will **write this procedure**:

```
int count (array_base, length, value)
```

The job of the `count()` procedure is to search the array that begins at the address `array_base` and contains `length` elements, counting the number of instances of the given `value` that appear in the array. The return value is that count.

Begin by downloading this code to your computer, saving it as `count.s`:[1]

`https://bit.ly/COSC-175-2526s-count`

Open the *Venus Simulator*. In the *Venus* tab, in the terminal, use the `upload` command, which will allow you to select the downloaded `count.s` file from your computer. Once uploaded, you can edit it:

```
[user@venus] /# upload
[user@venus] /# edit count.s
```

This code contains a call to the `count()` procedure, passing it the array defined in the `.data` segment, and searching for the value `102`. (You could change the value to seek to another value if you like.) The result of the call is then printed to the console before the

[1] Rather than last week's copy-and-paste approach to getting code into the simulator, it turns out that it is possible to download the code to your computer and then upload it into the simulator. So rather than view this code, select it for downloading.

program then exits.

Your task: Write the `count()` procedure to work as described above. Test and debug your code to be sure that it works correctly.

Download your code: When you have written code within the `count.s` file that is worth keeping, go back to the `Venus` tab terminal and download it:

```
[user@venus] /# download count.s
```

2.2 Exponentiation—recursion!

First download, and then upload into the simulator and edit, another code file (`exp.s`):

`https://bit.ly/COSC-175-2526s-exp`

This small program loads values `x` and `y`, passing them as arguments, `exp(x, y)`. Of course, this called function is the integer exponentiation function from class:

```
int exp (a, b)
```

Write the function recursively, debugging and testing it. Remember, as always, to **download your completed work**.

2.3 Fibonacci numbers—double-recursion!!?!

One more starter-code file (`fib.s`) for you to download (to your computer) and then upload-and-edit (on the simulator):

`https://bit.ly/COSC-175-2526s-fib`

This starter code contains a call to `fib()`, passing it whatever value is given at the label `n:` in the `.data` segment. You must **write this procedure**…

```
int fib (n)
```

…which must calculate and return the n^{th} Fiboanacci number, defined as:

$$F_n = \begin{cases} n & \text{if } 0 \leq n \leq 1 \\ F_{n-1} + F_{n-2} & \text{if } n \geq 2 \end{cases}$$

Write the `fib()` procedure recursively. You may not write it *iteratively*—that is, you cannot compute the answer by using a loop. Determine how you will use the stack to keep track of the recursive calls.

As always, test and debug your code. When it's ready, **download it**.

3 How to submit your work

Copy your completed `count.s`, `exp.s`, and `fib.s` source to the `lab-7` folder within your Google Drive folder.