Lab 3: Looping in MIPS

Due Friday 9 November 2018, 11:59 PM

Minimum Submission Requirements

- Create a Lab3 folder (note the capitalization convention, include no extra characters in the directory name) that contains the following files:
  - Lab3.asm
  - README.txt
- Commit and Push your repository
- Tag the commit that you would like to be graded
  - The tag must be in the form Lab3_submission_# (note the capitalization convention)

Lab Objective

This lab will introduce you to the MIPS ISA using MARS. You will write a program that iterates through a set of numbers and prints either “Flux,” “Bunny,” or “Flux Bunny.”

Lab Preparation

Read chapters 2, 3, and 7 from Introduction To MIPS Assembly Language Programming. Download and walk through Part 1 of the MARS tutorial. You will need to download Fibonacci.asm.

Specification

You will write a simple program in the MIPS32 language using the MARS integrated development environment. This program will prompt the user for a number. Next, the program will iterate through a set of integers (starting at 0, ending at the number input by the user) and print to the console one of four outputs depending on the number. If the number is evenly divisible by 5 (with no remainder), then the output is “Flux.” If the number is evenly divisible by 7, then the output is “Bunny.” If the number is divisible by both 5 and 7, then the output should be “Flux Bunny”. Lastly, if the output is not divisible by either 5 or 7, then the number itself should be printed.

An example of the expected output is given. The output should match this format exactly (note the capitalization convention). There should be a new line character after each output:

```
Please input a positive integer: 10
Flux Bunny
1
2
3
4
Flux
6
Bunny
8
9
Flux

-- program is finished running --
```

Note that part of our grading script is automated, so it is imperative that your program's output matches the specification. The prompt should be "Please input a positive integer: " NOT "Please *enter* a positive integer: ". There should be one space after the ":" in the prompt. The prompt should be the first text printed to the console. "Flux" and "Bunny" should be capitalized. There should be no space printed after the numbers. A new line character should be printed after the last number.

Your code should end cleanly without error. Make sure to use the exit syscall. You may assume that the user input is a positive integer i.e. no error handling is required.

Files

Lab3.asm

This file contains your code.

Header Comment

Your code should include a header comment with your name, CruzID, date, lab number, course number, quarter, school, program description and notes. Every program you write should include information like this. This is a good opportunity to start developing effective code documentation skills. An example header comment is shown below.

```
###############################################################################
# Created by:  Last Name, First Name
#              CruzID
#              7 August 2018
#
# Assignment:  Lab 64: Hello World
#              CMPE 012, Computer Systems and Assembly Language
#              UC Santa Cruz, Fall 2018
#
# Description: This program prints 'Hello world.' to the screen.
#
# Notes:       This program is intended to be run from the MARS IDE.
###############################################################################
```

Every block or section of code should have a comment describing what that block of code is for. In-line comments should be lined up (using spaces) for ease of readability.

Register Usage
You are permitted to use $v0, $a0 and the temporary ($t) registers for this lab. Registers $v0 and $a0 should only be used for the syscalls. At the beginning of your code, and optionally at the beginning of each block of code, indicate the functionality of the registers used. For instance, if you are using $t0 and $t1 for the user input and and loop counter, respectively, your comments should include something like the following:

```
# REGISTER USAGE
# $t0: user input
# $t1: loop counter
```

White Space
Line up instructions, operands, and comments to increase readability. Code should be indented from labels.

Bad Example
```
LOOP:
LI $t1 2 #initialize $t1
ADDI $t0 $t0 1 #increment $t0
BLT $t0 $t1 LOOP #determine if code should re-enter loop
```

Good Example
```
LOOP:
      LI   $t1 2          # initialize $t1
      ADDI $t0 $t0 1      # increment $t0
      BLT  $t0 $t1 LOOP   # determine if code should re-enter loop
```

A Note About Tabs
It is preferable to line up comments using spaces as opposed to tabs. Text editors can have different standards for the width of one tab character. For this reason, it is preferable to line up comments using spaces, not tabs, so that the code appears the same regardless of text editor.

README.txt
This file must be a plain text (.txt) file. It should contain your first and last name (as it appears on Canvas) and your CruzID. Your CruzID is your email address before @ucsc.edu. Your answers to the questions should total at least 8 sentences with complete thoughts.

Your README should adhere to the following template:

```
------------------------
Lab 3: Looping in MIPS
CMPE 012 Summer 2018

Last Name, First Name
CruzID
-------------------------
```

The text of your prompt (“Please input a positive integer: ”) and output strings

(“Flux” and “Bunny”) are stored in the processor’s memory. After assembling your program, what is the range of addresses in which these strings are stored?
Write the answer here.

What were the learning objectives of this lab?
Write the answer here.

Did you encounter any issues? Were there parts of this lab you found enjoyable?
Write the answer here.

How would you redesign this lab to make it better?
Write the answer here.

Grading Rubric

6 pts assembles without errors
13 pts output matches the specification
- 1 pt format of prompt
- 1 pt format of “Flux”
- 1 pt format of “Bunny”
- 1 pt format of “Flux Bunny”
- 1 pt format of integer
- 2 pts conditions for printing “Flux”
- 2 pts conditions for printing “Bunny”
- 2 pts conditions for printing “Flux Bunny”
- 2 pts conditions for printing integer

1 pt complete header comments in code and README
1 pt useful & sufficient comments
1 pt comment on register usage
1 pt clean visual structure / use of white space
Note: line up instructions, operands, and comments using spaces
indent code from labels
2 pts readme file complete (should at least 8 sentences total with complete thoughts)