Due date: Monday, Mar 4 (11:59 PM)

CS 271, Winter 2013 — Programming Assignment 3

Submission instructions

- Submit a single MIPS assembly file via TEACH: `http://engr.oregonstate.edu/teach`
- Your file must be named `PA3`[your-username]`.asm`
- For example, my submission would be named `PA3walkiner.asm`

Learning objectives

The goal of this assignment is to implement a more complex program using MIPS assembly language. By the end of this assignment, you should be able to:

1. effectively use global arrays (read, write, and enumerate)
2. design and implement an algorithm based around procedure calls
3. adhere to procedure calling conventions
4. manipulate the stack

Description and Requirements

Calculating the frequency of letters in text is an important task in cryptography, computational linguistics, and coding theory. In this assignment, you will write and test a MIPS assembly program that does this.

Your program will be organized into a number of procedures. You must define and use at least the following procedures in your program. (More is fine.)

void main()

- Call three procedures in order: **setup**, **analyze**, and **results**. Each of these is described below.
- Use the exit system call to end your program.

void setup()

- Print an introduction that includes: your name, a title, and a notification of any extra credit you have attempted.
- Prompt the user to enter a string and read the input into a global variable named *text* in memory. (You should reserve 1024 bytes in the data segment for this string. Remember that a string is equivalent to an array of characters.)

void analyze()

- Iterate over the 26 letters of the alphabet. (It is up to you how you do this.)
- For each letter `l`, call the procedure **count(l)**. This will return an integer `n`, the number of times `l` appears in the text.
- Store `n` in a global integer array named *freq*. (You should reserve space for a 26 integer array in the data segment.)
- When this procedure is done, *freq* should contain the frequency of each letter in the user-provided text.

int count(char l)

- The argument to this procedure, l, is one of the letters of the alphabet. (You may pass this as a character, or as an integer representing a character, e.g. A = 1, Z = 26.)
- The return value of this procedure, n, is the number of times l occurs in the user-provided text.
- Therefore, you should iterate over the text, counting the occurrences of l.
- Your count should be *case-insensitive*. So, if l = A, count both A and a.

void results()

- Iterate over the *freq* array, printing the frequency of each letter, as shown in the example output below.

Helpful information:

- Both of the arrays *text* and *freq* are global arrays in the data segment, so you should not need to pass any arguments related to them.
- You can convert from a capital ASCII character to lowercase character by adding 0x20.
- You can convert an integer in the range [1,26] to a character in the range [A,Z] by adding 0x40.
- The procedures **setup** and **results** are leaves that take no arguments, so you can omit the subroutine linkage.
- However, you should respect calling conventions when implementing and using **analyze** and **count**. For example, you should use the stack to save the value of $ra and any other registers that are needed when calling a procedure from within another procedure.

Important: To get full credit, your code must be fully documented! Specifically, the following are required:

- Your program should have a *header block* comment containing your name, the date, and a brief description of your program.
- Each procedure should have a *procedure block* comment containing the pseudocode implementation of the algorithm the procedure implements. You should do this before writing your assembly code—it will help!
- You must provide a *mapping* between register names and pseudocode variable names. You will probably have to do this for each procedure.
- Your in-line comments after assembly instructions should refer to the pseudocode implementation.

Extra credit: (Small amount. Standard requirements must be fulfilled first!)

1. (Easy) Print out a text-based histogram illustrating the frequency of the characters. For example, if A appeared four times and B appeared twice, part of your output might appear as follows.

   ```
   A: 4 xxxx
   B: 2 xx
   ...
   ```

2. (Easy) Implement the above with the help of a procedure. Specifically, you should define a procedure **void printXs(int n)**, and call this from within the **results** procedure.
3. (Hard) Order your output from the most frequently occurring letters to the least.
4. Do something astoundingly creative! If you do, note it in the description you print out in the introduction, and make it clear that it might be extra-credit worthy. :)

Example execution

Below is an example execution that illustrates what your program should do. Your output need not match this output exactly. The important thing is that it must be clear and easy to tell that your program is computing and printing the correct results. (Also remember that printing the histogram x's is optional extra credit.)

Program output is in `typewriter font` while user input is in ***red, bold, italic font***.

```
Letter Frequency Analysis by, Eric Walkingshaw

String to analyze: Time flies like an arrow. Fruit flies like a banana.

A: 6 xxxxxx
B: 1 x
C: 0
D: 0
E: 5 xxxxx
F: 3 xxx
G: 0
H: 0
I: 6 xxxxxx
J: 0
K: 2 xx
L: 4 xxxx
M: 1 x
N: 3 xxx
O: 1 x
P: 0
Q: 0
R: 3 xxx
S: 2 xx
T: 2 xx
U: 1 x
V: 0
W: 1 x
X: 0
Y: 0
Z: 0
```