1 Overview

In this project, you’ll be getting some basic experience with how to write assembly language. You will write a main() function. In main(), you will read four different variables, which will tell you what sorts of operations you must perform. For each of these four operations, you will read a set of four other variables and use the values in your calculations.

We have written a set of testcases; we will provide some of them to you to test your code. Each testcase will define all 8 of the variables that you need to read. For instance, one testcase might be:

```
data

print: .word 1
sum: .word 0
multiply: .word 0
minimum: .word 1

foo: .byte 0
bar: .word 42
baz: .word 17
fred: .half -123
```

Your program will read the values of print, sum, multiply, and minimum. For each that is nonzero, your program will perform that operation; you may have to perform zero, one, or multiple operations.

1.1 Required Filename

Name your file proj01.s

2 A Note About Grading

Your code will be tested automatically. We will include all of the testcases that we have provided to you - and also some number of additional testcases, which we will not give to you. Since your code is being tested automatically, you must match the required output exactly! Any extra spaces, blank lines (or missing lines), misspelled words, etc. will cause your testcase to fail.
Thus, be careful to test your code thoroughly, and to read the spec carefully. Test your code with every testcase that we provide! While it’s not a perfect guarantee, if your code passes all of the testcases we provide, there’s a good chance that it will also pass the ones that we have not given to you.

3 The Actions

3.1 print
If the memory location print is nonzero, the program will print the numbers, one on each line. Include the message shown below, and also a blank line after the last number.

```
Printing the four values:
0
42
17
-123
```

3.2 sum
If the memory location sum is nonzero, the program will add all of numbers, and print out the sum. Sum up the values in the following order: foo, bar, baz, fred; after each step (including when you have only read foo), print the current total. Print each total on a separate line. Also print the message shown below, and include a blank line at the end.

```
Running totals:
0
42
69
-54
```

3.3 multiply
If the memory location multiply is nonzero, the program will “multiply” each number by 7, and print the result. The word “multiply” is in quotes because you are not allowed to use the MIPS multiply instruction; instead, you must do this using only the add and/or sub instructions.

Print out each “multiplied” value, but not on separate lines! Instead, print them out all on the same line, with exactly two spaces between each one. Include a newline at the end, and then a blank line - but do not have any leading or trailing spaces. (That is, only put spaces between numbers.)
"Multiplying" each value by 7:
0  294  119  -861

3.4 minimum
If the memory location minimum is nonzero, the program will find the minimum of all of the numbers, and print it out. Print exactly the message shown below, and a blank line afterwards. There is exactly one space between the colon and the minimum value.

Minimum: -123

4 Testing
You can find a set of testcases for this project at http://www.cs.arizona.edu/classes/cs252/fall16/projects/proj01/
Each testcase has two parts: the .s file that contains the code we provide, and an .out text file which shows exactly what the output should be.

Here is an example of one of the testcases:

.data

print:   .word  1
sum:     .word  0
multiply: .word  0
minimum: .word  0
foo:     .half  1
bar:     .word  2
baz:     .word  17
fred:    .half  -123

(Information about how to run these testcases on the next page.)
4.1 Automatic Testing

We have also provided a testing script (in the same directory), named grade_proj01. Place this script, all of the testcase files, and your program (proj01.s) in the same directory. (I recommend that you do this on Lectura, or a similar department machine. It might also work on your Mac, but no promises!)

Run the testing script, and it will produce a report about your program. This is an example of a run which passed all testcases (we’ll have more than 5 testcases by the time you run this - but there were only 5 when I created the project spec):

****************************
* Testcase 'test01_printOnly' passed
****************************

****************************
* Testcase 'test02_sumOnly' passed
****************************

****************************
* Testcase 'test03_multOnly' passed
****************************

****************************
* Testcase 'test04_minimumOnly' passed
****************************

****************************
* Testcase 'test05_all4' passed
****************************

*******************************************
* OVERALL REPORT
* attempts: 5
* passed: 5
*******************************************

(More examples of testcases on the next page.)
This is an example of a run where a testcase failed:

**************************************************
* Testcase 'test01_printOnly' passed
**************************************************

**************************************************
* Testcase 'test02_sumOnly' passed
**************************************************

**************************************************
* Testcase 'test03_multOnly' passed
**************************************************

**************************************************
* TESTCASE 'test04_minimumOnly' FAILED
**************************************************

----- CORRECT OUTPUT ----
Minimum: 0

----- ACTUAL OUTPUT ----
Minimum: -123

----- diff OUTPUT ----
6c6
< Minimum: 0
---
> Minimum: -123
----- END diff ----

**************************************************
* Testcase 'test05_all4' passed
**************************************************

**************************************************
* OVERALL REPORT
* attempts: 5
* passed: 4
* failed:
  * test04_minimumOnly
*
4.2 Running spim By Hand

If the testing script reports errors (or if you just hate scripts), you’re welcome to run your code by hand. To do this, you need to join the two file together (your code, and the testcase), and then run them through SPIM.

To run on the command line (that is, without a GUI), try this command:

```bash
spim -f <(proj01.s test01_printOnly.s)
```

(Make everything exactly exactly as it is above...except, you’ll have to replace the testcase name with the name of the testcase you want to run.)

**COOL TRICK:** If you don’t like the 5 lines of text that SPIM prints out at the beginning of the program, you can add `tail` at the end to remove them:

```bash
spim -f <(proj01.s test01_printOnly.s) | tail -n+6
```

4.3 Running Inside QtSpim

If you prefer to run inside QtSpim (which is a good idea if you’re still debugging!) perform the following steps to load the program and testcase:

- Select “File...Reinitialize and Load File”; load the testcase file first.
- Select “File...Load File”; load your program.

Then you can hit Run (F5) or Single-Step (F10) to run your program.

When you change your file, you must re-load both files (the testcase, and your program); make sure that you also re-initialize, as I showed you above.

5 Turning in Your Solution

You must turn in your program using turnin on lectura. Use the assignment name cs252_f16_proj01. Turn in only your program (for example: proj01.s); do not turn in any testcases, grading scripts, or other files.

If you need to use a late day, simply add _late to the end of the assignment name. Remember, both turnin folders automatically close - the first one at the due date, and the second one 24 hours later!