**Lab 5: Introduction to Branching in ARM Assembly**

Due Monday, August 03 at 11:59 PM

**Goals for This Lab**

By the time you have completed this work, you should be able to:

* Execute instructions conditionally to accomplish larger tasks
* Translate conditionals (if/else) to ARM assembly
* Translate loops to ARM assembly

**Provided files:**

* min\_max.s
* two\_ints.txt
* median.s
* three\_ints.txt
* factorial.s
* one\_int.txt
* collaborators.txt

**Rubric:**

- min\_max.s: 20

- median.s: 50

- factorial.s: 30

**Step-by-Step Instructions**

**Step 1: Download all Required Files**

Download all the files listed under **Provided files** above. These **must** all be placed in the same folder/directory, and it should be someplace that you can easily access later. For example, these could go on either a personal laptop, or a personal USB drive.

If the files are placed in different directories, then none of the .s files will work correctly, as these assume that the .txt files are in the same directory.

**Step 2: Edit min\_max.s**

Open the min\_max.s file, and open it up in a text editor of your choice. Note that word processors (e.g., Microsoft Word, Pages, Google Docs) will probably **not** work for this purpose, as you must save your file as plain text. You must write ARM assembly code which will read the two integers in the provided two\_ints.txt file, and will then print the minimum and maximum values in the file, respectively. Example output of this code is shown below, which results from reading in the provided two\_ints.txt file:

Min: 13

Max: 42

The most straightforward way to do this is to do the following, in order:

1. Open up two\_ints.txt.
2. Read in the two integers into two separate registers.
3. Close the file.
4. Compare the integers to each other (as with the cmp instruction).
5. Put the smaller integer in one register, and the larger integer in another register. Exactly which integer goes where depends on the result of the comparison in the previous step. A conditional mov instruction is likely the most straightforward way to do this (see absolute\_value.s for an example using conditional instructions).
6. Print out the “Min: ” string, followed by the minimal integer, followed by a newline.
7. Print out the “Max: ” string, followed by the maximal integer, followed by a newline.

Be sure to **test your solution!** Try it with different values in two\_ints.txt. You will not submit two\_ints.txt anyway, so you can mess around with this file as much as you want. As a suggestion, the following cases may be of interest for testing:

* The first entry in two\_ints.txt is the smallest value.
* The second entry in two\_ints.txt is the smallest value.
* two\_ints.txt contains only negative values
* two\_ints.txt contains both negative and positive values

**Step 3: Edit** [**median.s**](https://kyledewey.github.io/comp122-fall17/labs/lab_5/median.s)

Open the median.s file, and open it up in a text editor of your choice. Note that word processors (e.g., Microsoft Word, Pages, Google Docs) will probably **not** work for this purpose, as you must save your file as plain text. You must write ARM assembly code which will read the three integers in the provided three\_ints.txt file, and will then print the median value in the file. To be clear, the median value is the one that would appear second in the file, *if* if the file were sorted. Example output of this code is shown below, which results from reading in the provided three\_ints.txt file:

78

As with the previous step, you will need to compare multiple integers to each other, likely using the cmp instruction. However, you will need to perform more comparisons than you did in the previous step, and it likely won't be very straightforward. It is recommended to write code that determines the median value in whatever language you're most familiar with **first**, and **then** try to translate this code into ARM assembly.

As a hint, there are a number of ways in which the median value can be found. Doing it with a very large if...else if...else statement is possible, though it will result in a lot of code. This approach looks similar to the ARM assembly code in big\_if.s, which is a translation of the Java code in BigIf.java. My own solution based on this approach needs 26 lines of ARM assembly just for the if, and it's pretty tricky to understand. Another approach is based on effectively hard-coding a sorting algorithm over three integers, based on swapping values as needed and then taking the second one. My own code based on this approach only needs 12 instructions, and it's surprisingly pretty straightforward relative to the other solution. I have no preference regarding which approach you take, as long as it is correct.

Be sure to **test your solution!** Try it with different values in three\_ints.txt. You will not submit three\_ints.txt anyway, so you can mess around with this file as much as you want. As a suggestion, the following cases may be of interest for testing:

* The median value is the first entry in three\_ints.txt
* The median value is the second entry in three\_ints.txt
* The median value is the third entry in three\_ints.txt
* three\_ints.txt contains two identical values
* three\_ints.txt contains three identical values
* three\_ints.txt contains only negative values
* three\_ints.txt contains both negative and positive values

**Step 4: Edit factorial.s**

Open the factorial.s file, and open it up in a text editor of your choice. Note that word processors (e.g., Microsoft Word, Pages, Google Docs) will probably **not** work for this purpose, as you must save your file as plain text. You must write ARM assembly code that will compute the [factorial](https://en.wikipedia.org/wiki/Factorial) of an input number, along with showing intermediate values along the way. Psuedocode showing what you need to implement is shown below:

n = <<first integer read from "one\_int.txt">>

accum = 1

while (n != 0) {

accum = accum \* n

print accum

print "\n"

n = n - 1

}

print accum

print "\n"

A sample run of the program with the given one\_int.txt file is shown below:

5

20

60

120

120

120

The code you write will likely look similar to the ARM assembly code in while\_loop.s, which is a translation of the Java code in WhileLoop.java.

Your code **must** be able to work with more than just 5. With this in mind, be sure to **test your solution!** At the very least, you should test values between 1 and 10. You may assume that no negative inputs will be given as inputs.

**Step 5: Turn in Your Code Using** [**Canvas**](https://canvas.csun.edu)

Log into [Canvas](https://canvas.csun.edu), and go to the COMP 122L class. Click “Assignments” on the left pane, then click “Lab 5”. From here, you can upload your .s files. Specifically, you must turn in the following three files:

* min\_max.s
* median.s
* factorial.s

In addition, if you collaborated with anyone else, be sure to download collaborators.txt and write the names of the people you collaborated with in the file, one per line. Please submit this file along with the other three files.

You can turn in the assignment multiple times, but only the last version you submitted will be graded.

**IMPORTANT: Your Code Must Run Under** [**ARMSim#**](http://armsim.cs.uvic.ca/)

The code you submit **must** run under [ARMSim#](http://armsim.cs.uvic.ca/) without modification.   
**Code with syntax errors gets an automatic 0**.   
If you can't get your code to do the right thing, it's better to submit code that runs but does the wrong thing. Similarly, your code must read in from input files, as opposed to simply printing out the answers for the given inputs. Code that prints hard-coded answers will receive an automatic 0.