Lab 4: Introduction to ARM Assembly with ARMSim#

Due **Tuesday, July 28** at 11:59 PM

Goals for This Lab

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

- Use common arithmetic and bitwise operations in ARM assembly
- Use ARMSim to print strings in ARM assembly
- Use ARMSim to open and read integers from files in ARM assembly

Provided files:

- `print_hardcoded_strings.s`
- `two_ints.txt`
- `print_two_not.s`
- `binops.s`
- `collaborators.txt`

Step-by-Step Instructions

Step 1: Familiarize Yourself with a ARMSim# Tutorial

Read and familiarize yourself with a short ARMSim# tutorial. This tutorial discusses how to open and run ARM assembly code from within ARMSim#, which you will be doing frequently over the next several weeks. It would also be beneficial to take a glance at the ARMSim# manual, in particular the portion that covers the `SWI` I/O operations (starting on page 21).

Step 2: 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 `print_two_not.s` and `binops.s` will not correctly, as these programs assume that `two_ints.txt` is in the same directory.

Step 3: Edit `print_hardcoded_strings.s`

Open the `print_hardcoded_strings.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, when run under ARMSim#, will print the following:

```
first
second
```

As a hint, your code will look very similar to `hello.s`.

Step 4: Edit `print_two_not.s`

Open the `print_two_not.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 (in order):

1. The first integer on its own line
2. The bitwise negation (NOT) of the first integer on its own line
3. The second integer on its own line
4. The bitwise negation (NOT) of the second integer on its own line

Example output of this code is shown below, which results from reading in the provided `two_ints.txt` file:

```
63
-64
5
-6
```

`print_two_not.s` contains comments which further describe exactly how you can go about this. As a hint, your code will look similar to `read_and_print_int.s`.

Step 5: Edit `binops.s`

Open the `binops.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 (hereafter referred to as `A` and `B`, respectively) in the provided `two_ints.txt` file, and will then print (in order):

1. `A + B` on its own line
2. `A - B` on its own line
3. `A * B` on its own line
4. `A & B` (bitwise AND) on its own line
5. `A | B` (bitwise OR) on its own line
6. `A ^ B` (bitwise XOR) on its own line
7. `A << B` (shift left) on its own line
8. `A >>> B` (logical shift right) on its own line
9. `A >> B` (arithmetic shift right) on its own line

Example output of this code is shown below, which results from reading in the provided two_ints.txt file:

```
68
58
315
5
63
58
2016
1
1
```

binops.s contains comments which further describe exactly how you can go about this. As a hint, your code will look similar to arithmetic_ops.s, as well as read_and_print_int.s.

Step 6: Turn in Your Code Using Canvas

Log into Canvas, and go to the COMP 122L class. Click "Assignments" on the left pane, then click "Lab 4". From here, you can upload your .s files. Specifically, you must turn in the following three files:

- print_hardcoded_strings.s
- print_two_not.s
- binops.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#

The code you submit **must** run under ARMSim# 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,print_two_not.s and binops.s must actually read in the integers from the two_ints.txt file. On my end, I will be changing the contents of two_ints.txt, and will check to see that your code still does the right thing. If you're struggling to read from the file, it's better to submit code that does the operations with hardcoded constants. Code that simply prints out the answers for the provided two_ints.txt will receive an automatic 0.