Lab 6: Memory Operations and Arrays in ARM Assembly

Due **Monday, August 10 at 11:59 PM**

Goals for This Lab

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

- Access values in memory with the `ldr` instruction
- Store values in memory with the `str` instruction
- Iterate over arrays in memory

Provided files:

- `find_min_array.s`
- `add_amount_array.s`
- `collaborators.txt`

Step 1: Edit `find_min_array.s`

Open the `find_min_array.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 find and print out the smallest element of the array, where the array is specified with the `array` label and the array length is specified with the `array_length` label. Example output of this code is shown below, based on the provided array and array length in `find_min_array.s`:

```
-5
```

Multiple different implementation approaches are possible. One such implementation approach is shown below, implemented in pseudocode:

```
min = array[0];
for each element of the array:
  if element < min:
    min = element;
  endif
endfor
```

You should test your code with different values for the elements of the array, and different array lengths. On my end, I will test your code by subbing out different values for `array` and `array_length`. Code that simply prints out the minimum value for the given array will receive no credit.

Step 2: Edit add_amount_array.s

Open the add_amount_array.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. This program will read values from a source array, add a specified amount to each value, and put the result in another (sink) array. A number of definitions are provided in the file, summarized below:

- `array_source`: The source array to read from
- `array_sink`: The destination array to write results to
- `array_length`: The length of the arrays. It is assumed that the `array_source` and `array_sink` arrays have the same length.
- `add_amount`: The amount to add to each element

Multiple different implementation approaches are possible. One such implementation approach is shown below, implemented in pseudocode:

```
counter = 0;
while counter < array_length:
  array_sink[counter] = array_source[counter] + add_amount;
  counter++;
endwhile
```

The bottom portion of the code will iterate over each element of the `array_sink` and print out its value. **Do not modify this portion of the code in any way.** Any modifications will result in a 0.

You should test your code with different values for the elements of the array, and different array lengths. On my end, I will test your code by subbing out different values for `array` and `array_length`. Code that simply prints out the minimum value for the given array will receive no credit.

Step 3: 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 6”. From here, you can upload your `.s` files. Specifically, you must turn in the following two files:

- `find_min_array.s`
- `add_amount_array.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 two 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.