ASSESSED INDIVIDUAL WORK

COMP1006
LAB EXERCISE #2

Steven R. Bagley

Submission Deadline:
26/10/2021 1200

Version: 1.00

These exercises are designed to be straightforward and are primarily to get you used to using Komodo on the school's Linux system and writing ARM assembly language. These exercises are all build on the example from the online lecture engagement on the 18th October, and expects you to be familiar with both loading and 'compiling' (actually, the **Compile** button should read assemble) source code in Komodo. If you are not yet familiar with Komodo I suggest you refer to the video on Moodle outlining how to use Komodo.

Getting Started

You will need to fork and clone the relevant project via `git` in the usual fashion (i.e. as you have previously done for COMP1005). In this case, the project to fork can be found at:

`https://projects.cs.nott.ac.uk/2021-COMP1006/2021-COMP1006-LabExercise02`

Once cloned, you will find skeleton `.s` files for each exercise. It is **strongly recommended** that you start from these skeletons otherwise your programs may not work against the marking test scripts.

Note: For all these programs you may well find it useful to start by generating a C version (or any other programming language you are familiar with) first to ensure the logic of the program is sorted before you start transliterating it into ARM assembler.

There are two exercises this week, one which gets you to print out times tables and another which prints out multiple verses of the mowing song. Full detail of the exercises can be found overleaf.

Exercise 1 — Times Tables

For this exercise you are to write a simple program to print out a 'Times Table'. Your program should read the times table to produce from the variable labelled `table` in the supplied `01-times.s` skeleton file.

When run, your program should load the value from the memory location labelled `table` (using `LDR`) into a register and then print out the 'Times Table' for that number, e.g. if `table` contains the number 13, you should print out the thirteen times table, like this:

```
0x13 is 0
1x13 is 13
2x13 is 26
3x13 is 39
4x13 is 52
5x13 is 65
6x13 is 78
7x13 is 91
8x13 is 104
9x13 is 117
10x13 is 130
11x13 is 143
12x13 is 156
13x13 is 169
```

You should print out all the lines up to the square, so if `table` contains the value 400 you should print the 400 entries up to 400x400…

Some hints for completing this exercise:

- You will need to count up from zero to the value you read from `table`. This is similar to the example we considered in the online lecture engagement, although you will need to adjust which registers are used (since you'll need to print things out using `R0` and `SWI 3/4`). You might want to use that example as a starting point by modifying it to print out each value as it counts up, then modify it so it prints out each number starting from zero, and so on…
- All the strings you should need are defined in the skeleton file. Remember you can move to a newline by printing the new line character (which has ASCII code 10) using `SWI 0`, thusly:
  ```
  MOV R0, #10
  SWI 0
  ```
- Likewise, we can print any character out using `SWI 0`, by putting the character in single quote marks after the `#`, so to print an '`x`', you'd:
  ```
  MOV R0, #'x'
  SWI 0
  ```
- Remember the multiplication is just repeated addition, and so there's no need to do any multiplication for this exercise. You can build up the times table values by repeatedly adding the value read from `table` as you loop around printing each line of the table.

- You'll need to build each line up in parts, printing each out either as a number (with `SWI 4`), a character (with `SWI 0`), or a string (`SWI 3`).
- You should check that your program works correctly for different values of table, the pipeline will be testing different values. You should also ensure that your program produces no output if

Exercise 2 — The Mowing Song (again)

This exercise combines 'The Mowing Song' program you wrote last week, with the loops and data processing instructions we've been looking at this week. This week, you are to extend your mowing song program to print out all verses of the song using a loop.

Your program should read (using `LDR`) the number of verses to be displayed from a value defined (using `DEFW`) in memory — this is labelled as `verses` in the skeleton `02-meadow.s`. You should then printout the required number of verses. Each verse output will have the 'number of men' decremented by one in each verse as shown in the example output in the Appendix. The final verse should always be '*1 man went to mow …*'.

The tricky part of this exercise (compared to last week) is the middle line of the verse:

```
4 men, 3 men, 2 men, 1 man and his dog, Spot
```

This line can no longer just be printed out since the number of men printed depends on which verse you are in. You will need to implement a second loop counting down from the number of men in each verse. Note, that the final verse (where the number of men is one), only prints the '*1 man and his dog*' part.

Some hints for completing this exercise:

- You will need to define the strings you want to print out using `SWI 3` using `DEFB` and you must include the `,0` part after the string has finished (see the example in the lecture slides). I strongly suggest you place these were suggested in the source file.
- Remember that `SWI 3` and `SWI 4` expect the address of the string to print or the number to print in `R0` — you will almost certainly have to move values from other registers into `R0` to get using `MOV`.
- I suggest breaking the problem down into different stages and tackle each individually…
    - Firstly, I'd make your program print out the required number of verses, by putting your code from last week into a loop. The code for the loop will be very similar to the code you wrote for the times table (and the code example from the lecture engagement).
        - Start by making the program print out the correct number of verses, with a blank line between them using a similar loop to the one you created for the times table.
        - Next, I suggest changing your code to print the first line '4 men went to mow' with the correct number of men (based on the verse).
        - It may well be easier to start by getting your loop to count up (as in the times table program above), and then when the loop is working get it to count down

instead by changing the conditional (formed using `CMP` and `Bcc` — where `cc` represents the required condition code) and using `SUB` instead of `ADD`.

- Then, I'd work on the middle line of the verse ('4 men, 3 men,…' etc.). There are a lot of similarities between the code need here and code need to print out each verse.

- Don't forget a `SWI 2` to stop your program at the end…
- Your program should work for any positive integer value of verses

Example output can be found in the Appendix.

APPENDIX

The Mowing Song (four verses):

```
4 men went to mow
Went to mow a meadow
4 men, 3 men, 2 men, 1 man and his dog, Spot
Went to mow a meadow

3 men went to mow
Went to mow a meadow
3 men, 2 men, 1 man and his dog, Spot
Went to mow a meadow

2 men went to mow
Went to mow a meadow
2 men, 1 man and his dog, Spot
went to mow a meadow

1 man went to mow
Went to mow a meadow
1 man and his dog, Spot
Went to mow a meadow
```