**ASSESSED INDIVIDUAL WORK**

# COMP1006
# LAB EXERCISE #2

Steven R. Bagley

Submission Deadline:
~~**25/10/2022 1200**~~
Extended to: **01/11/2022 1200**

Version: 1.10

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/2022-COMP1006/2022-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 prints out multiple verses of the mowing song and another which gets you to print out the fibonacci sequence. Full detail of the exercises can be found overleaf.

## Marking Pipeline

Marking pipelines for this exercise are available, in the same fashion as for *COMP1005 Programming and Algorithms*. Once you have submitted your answers using `git push`, you can find the marking report in the Continuous Integration section of the projects.cs.nott.ac.uk.

## 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 `01-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 notes). 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, 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 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 from the lecture engagement.
        - 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 example program), and then when the loop is working get it to count down instead by changing the conditional (formed using `CMP` and `B`*`cc`* — 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.

**Assessment Criteria**

There is a gitlab pipeline available for this exercise which will mark your program and give you feedback on how well it works. Specifically, marks are awarded as follows:

- 1 mark is available for the verse structure being correct and the number in each verse counts down correctly.
- 1 mark is available for the number of men counting down in the third line of each verse '4 men, 3 men, …'
- 1 mark is available if the program prints 'men' or 'man' as appropriate.
- 2 marks are available for programs that operate correctly when the value of the variable `verses` is changed (causing the number of verses output to alter).

## Fibonacci sequence

This program should print out the first *n* numbers in the Fibonacci sequence (starting from 1), up to a value specified in the program by the label `fibbEnd` . The Fibonacci sequence is defined as follows:

fibb *n* = fibb *n*-1 + fibb *n*-2

where:

fibb 2 = 1
fibb 1 = 1
fibb 0 = 0

so the fibb 3 would be 2, and so on. In other words, adding together the two previous fibonacci numbers reveals the next number in the sequence. Therefore, if `fibbEnd` was 15, your program should print out:

```
Fibonacci number 1 is 1.
Fibonacci number 2 is 1.
Fibonacci number 3 is 2.
Fibonacci number 4 is 3.
Fibonacci number 5 is 5.
Fibonacci number 6 is 8.
Fibonacci number 7 is 13.
Fibonacci number 8 is 21.
Fibonacci number 9 is 34.
Fibonacci number 10 is 55.
Fibonacci number 11 is 89.
Fibonacci number 12 is 144.
Fibonacci number 13 is 233.
```

```
Fibonacci number 14 is 377.
Fibonacci number 15 is 610.
```

You'll need to use an iterative solution to calculate the fibonacci numbers — adding up the previous values in a loop until you get to the correct point in the sequence. Your program will need to do the following:

- The number of Fibonacci numbers to print out should be read from the memory location defined in the skeleton labelled `fibbEnd`. You will need to read this value from memory using `LDR`.
- This program will need to count up from one to the value read from `fibbEnd`.
- Inside your loop you will need to:
    - For each value, iteratively calculate the fibonacci number at that position in the sequence by `ADD`ing together the two previous Fibonacci numbers.
    - Print out the line of text as illustrated above
- You will probably need to think about how you are going to make use of the various registers. `R0` will be needed to print the strings and the numbers (using `SWI 4`), but you will probably also need to decide on which register you are going to use to keep track of the previous two Fibonacci numbers, and you will also need to use a register as a loop counter.
- Don't try to write the program all in one go, rather break the development down into a series of chunks. If I was writing this I'd probably break it into the following chunks
    - Start by writing a program that loops between 1 and `fibbEnd` printing out a string on each iteration of the loop. As mentioned earlier, this will be similar to the example demonstrated in the online lecture engagement, but you will need to add some extra instructions to make it print the string.
    - Then get it to print out the loop iteration (1, 2, 3, etc.) as part of the string using `SWI 4`.
    - Finally, get it to calculate and print out the Fibonacci number.

You should test your program with various input values to check it works correctly, by changing the number value stored at `fibbEnd` using `DEFW`.

**Assessment Criteria**

There is a gitlab pipeline available for this exercise which will mark your program and give you feedback on how well it works. Specifically, marks are awarded as follows:

- 2 marks are available for a program that produced a correct Fibonacci series when run
- 2 marks are available for printing the string in the correct form as shown above
- 4 marks are available if the program continues to correctly generate Fibonacci sequences for different values of `fibbEnd`.

- 2 marks are available for using a sensible number of branch operations as part of your program.

# 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
```