THE UNIVERSITY OF AUCKLAND
FACULTY OF SCIENCE

COMPSCI 210 S1 C 2016
Computer Systems 1
Assignment 1 – Data Representation and Assembly Language Programming
Due Friday 8th of April 11:59 pm
Worth 10%

Hardly anyone programs in assembly language anymore. This assignment should help you understand why :).

Part A

Go to www.coderunner.auckland.ac.nz and complete the 3 Python programs which deal with data representation.

[12 marks]

For the following assembly language programs you may assume the input buffer can accept up to 10 characters. It will not be tested beyond this.

Part B1

Write an LC-3 assembly language program called B1.asm which **repeatedly** asks the user for a decimal digit and waits for the user to type a digit ("0" to "9") followed by the return/enter key. The program then prints the binary form of the digit as a 4 bit **unsigned** value. Any value apart from a digit causes the program to finish.

e.g. The program should work like this:

```
Enter a digit: 3

0011

Enter a digit: abc
```

[4 marks]

Part B2

Write an LC-3 assembly language program called B2.asm which prints out a list of Fibonacci numbers. The program asks the user to enter a number between 3 and 23, and repeats this until the user types a value which is between 3 and 23. If the value is *n* the program then prints out the first *n* elements of the Fibonacci sequence and then stops.

e.g. The program should work like this:

```
Enter a number from 3 to 23: abc

Enter a number from 3 to 23: 1

Enter a number from 3 to 23: 50

Enter a number from 3 to 23: 10

1 1 2 3 5 8 13 21 34 55
```

This program **must** calculate the Fibonacci numbers iteratively.

Each Fibonacci number is produced by adding the two previous numbers in the sequence.

[4 marks]

Part B3

This program called B3.asm should work exactly like Part B2 but it should be implemented differently. The calculation of each Fibonacci number must be performed by a recursive function modelled on the following Python code:

```
def fib(n):
    if n < 3:
        return 1
    else:
        return fib(n-2) + fib(n-1)
```

[4 marks]

Hints

Be careful when calling a function/subroutine which includes `TRAP` instructions or `JSR` instructions. R7 is used by both of these to store the return address. If you don't save the return address somewhere else you won't be able to get back to where you were coming from.

For Part B3 you will need to have enough stack space for each of the function calls.

Questions (3 marks)

Submit the answers to these as either a text or pdf document called A1Q.txt or A1Q.pdf (please don't use a proprietary format). Include the declaration of originality as specified below.

Q1. Why does the specification for B2 only accept up to the 23rd Fibonacci number?

[1 mark]

Q2. You were told that the input buffer has a maximum size of 10 characters. Your program does not have to check for more than 10 characters. If a program did not limit the user to only 10 characters what would happen if the user typed more than 10 characters? Explain how this could be a problem.

[2 marks]

Marking Guide for Part B

Part B1 (4 marks)

Prints correct prompt and accepts input from the user until the user presses return/enter. (1 mark)

Displays the 4 bit version of the input digit and loops back for more input if the input is valid. (2 marks)

Finishes if the input is not valid. (1 mark)

Part B2 (4 marks)

Program repeatedly loops asking for input until a valid number is entered. (1 mark)

Prints the required section of the Fibonacci sequence when a valid number is entered. (2 marks)

Source code clearly indicates the section of code which generates the Fibonacci sequence and shows it to be iterative. (1 mark)

Part B3 (4 marks)

Source code clearly indicates the recursive function calls and makes it understandable how the parameters and return addresses are saved. (2 marks)

Fibonacci function is implemented recursively and works as specified (2 marks)

Style (3 marks)

All files submitted using Canvas must include the student's name and declaration of originality in the starting comment, as shown here. By including the declaration you are stating that it is true. You do not get **any** marks for style or the questions if the declaration is not included in **all** files.

```
; login name - astu001
; This program is my own unaided work, and was not copied,
; nor written in collaboration with any other person.
; Alex Stuart <- this should be your name
```

Good labels are used for branch destinations and data values. (1 mark)

Sections of code are clearly commented and laid out consistently. (1 mark)

Contents of registers are shown in such a way that the marker can identify what each register is being used for in each section of code. (1 mark)

Total (30 marks worth 10%)

Submitting the assignment

Submit Part A directly into CodeRunner at www.coderunner.auckland.ac.nz.

Submit the three Part B programs and the answers to the questions in Canvas.