Assignment #1
Simple Statistics Calculator

1 November 2016

This is an <u>individual</u> assignment and will contribute 10% of your final mark for CS1021.

You will be asked to demonstrate your solution during the labs on Friday 18th November 2016. Marks will be awarded for this demonstration.

You must submit your solutions using Blackboard no later than 23:59 on Monday 21st November 2016. Late submissions without a satisfactory explanation will receive zero marks.

Submit your **commented** .s assembly language source files and your report **in PDF format** as attachments to "Assignment #1" in Blackboard.

Solutions will be checked for plagiarism.

Introduction

In this assignment, you will develop a program to compute a number of statistical measures (count, sum, min, max, mean) for a sequence of sample values. You will develop the program in three stages. In Stage 1, you will write a program that can read a single unsigned value entered by a user. In Stage 2, you will extend your program to read a sequence of values and compute the statistical measures. Finally, in Stage 3, you will extend your program further to display the mean of the sequence of values on the screen.

Important Note: You must submit your solution to each stage of the assignment as a separate .s ARM Assembly Language source file. Stages 2 and 3 build on Stages 1 and 2 respectively, so you should begin these stages using your solution from the preceding stage as a starting point.

1 Console Input

The aim of this stage of the assignment is to design, write and test an ARM Assembly Language program that will read an unsigned value entered by a user in decimal form and store the value in register R4. The value will be entered by the user as a string of ASCII characters. For example, if a user enters "2", followed by "1", followed by "3", your program should store 213_{10} (i.e. 11010101_2 or $D5_{16}$) in R4.

The **ConsoleInput** μVision project contains a simple program that demonstrates how to read ASCII characters from the console one character at a time, as they are entered by a user. A single character is read by executing the `BL getkey` instruction. This instruction only completes execution when the user presses a key. When the instruction completes, the ASCII character code corresponding to the key pressed will be stored in R0.

The key pressed by the user is not automatically displayed in the console window so we need to do this ourselves with the `BL sendchar` instruction. This instruction displays the character corresponding to the ASCII code contained in `R0`.

Example: If we execute `BL getkey` and the user types '7' on the keyboard, then 0x37 will be stored in `R0`. If we then execute `BL sendchar` (with 0x37 in `R0`) then '7' will be displayed on the console.)

Very Important Note: Executing the `BL getkey` or `BL sendchar` instructions *may* overwrite the contents of R0–R3 so you should avoid using these registers to store any values that you will need after executing either of these instructions. You must also avoid using R13–R15, as usual.

The program should stop reading ASCII characters when the ASCII `Carriage Return` character is read. This should occur when the user presses the **RETURN** key. The `Carriage Return` character has ASCII code **0x0D**.

Verify that your program stores the value entered in R4.

Using the µVision Console

The program above requires you to enter text using a "console". To simplify this, you should only test your program in Simulation mode, which simulates the ARM hardware. (Using a "console" with the real ARM hardware is slightly more complicated. By using the simulator, you will also be able to work on the assignment on your own computer.)

To open the console window in Debug mode, select the **View – Serial Windows – UART #1** menu option.

When testing your program, you can either step through the program one instruction at a time or use the run icon to run the program without stopping. If you are stepping through one instruction at a time, after executing `BL getkey` you will need to mouse-click in the "UART #1" console window and press a single key. Your program will then continue to the next instruction.

As always, it is highly recommended that you use uVision version 4 and not version 5 for this project as it greatly simplifies the process of stepping through your programs one instruction at a time.

Alternatively, if you choose to run the whole program, you can mouse-click inside the console window and enter the full expression. To see the result of your program in R4, you will need to halt your program with the **Halt** icon (beside **Run**).

2 Sample Sequence and Statistic Computation

In this Stage of development, you will extend your program to read a sequence of values and compute the following statistical measures:

- Count

- Sum

- Min

- Max

- Mean

Your program should store each result in a separate register.

Your solution should be an extension of the program that you developed for Stage 1. You should use the **StatEval** µVision project to develop and thoroughly test your solution.

You may assume when writing your program that the user will be "well behaved" and will only enter valid valid unsigned values. Be sure to state any additional simplifying assumptions that you make.

3   Displaying the Mean

In this final Stage of the assignment, you will extend your program to display the mean computed in Stage 2. The result must be displayed in decimal form.

You can use the `BL sendchar` instruction to display an ASCII symbol on the console. This instruction displays the symbol corresponding to the ASCII code contained in `R0`.

Use the **DisplayResult** µVision project to develop your solution.

Suggested Approach

You can use the **Divide** program developed in a recent lab exercise to convert a value stored in a register to the ASCII characters representing the value. One approach is to divide the mean by decreasing powers of 10, using the whole part of the result as the next ASCII digit and the remainder as the next value to be divided by the next lower power of 10.

For example, given the initial value 4128,

- Dividing 4128 by 1000 will give you a quotient of 4 and a remainder of 128. Therefore $0x04 + 0x30 = 0x34$ (character '4') will be the first ASCII character to be displayed.

- Dividing 128 by 100 will give you a quotient of 1 and a remainder of 28. Therefore $0x01 + 0x30 = 0x31$ (character '1') will be the second ASCII character to be displayed.

- etc.

You will need to choose an appropriate power of 10 to begin with. You can use the **Power** program from the lecture slides to compute powers of 10.

Documentation

You are required to document your approach to the development of the Simple Statistics Calculator program. Your documentation should be in the form of a typed document **in PDF format**. This document must be submitted with your three assembly language `.s` source files. Documents in a format other than PDF will not be graded.

Your report must contain the following:

A description of your solution to each of the three stages in the assignment. Your description should make use of examples, diagrams and pseudo-code where appropriate. **Your description must not take the form of a "narrative" for your assembly language program ("I moved the value from R5 into R4. Then I added R4 to R3 and stored the result in R0"). Such narratives will receive very low marks.** Instead, try to describe how your program works at a higher, conceptual level.

A detailed description of your methodology for testing each of the three stages from above. This must include an account of the inputs used to test your program. Your report should demonstrate that you have thoroughly tested each part of your program. You should give careful consideration to choosing inputs that test different parts of your program (e.g. both positive and zero inputs, zero-length sequences).

Explain why you are using each test input, document the outputs produced by your program (all of the statistical measures, not just the mean displayed on the console). State whether the output is correct and, if not, state why you think it is incorrect. Presenting your tests and results in the form of a table is highly recommended.

Evaluation

The following broad marking scheme will be used for the assignment:

- Stage 1 – Console Input – 20%

- Stage 2 – Sample Sequence and Statistic Computation – 25%

- Stage 3 – Displaying the Result (Mean) – 30%

- Documentation – 25%

Note that marks will be awarded for both the content and presentation of your document. Solutions that are merely working will not automatically attract 100% of the marks available. Marks will also be awarded for the quality of the solution.

Extra Mile

For each of the three stages in this assignment, 20% of the marks available will be awarded for going the "extra mile". How you do this is up to you. The following are some examples of features that may be worthy of these marks:

- Allowing the user to enter either positive or negative values.

- Computing additional statistical measures.

- Formatting the mean in some way (e.g. preventing leading zeros from being displayed), displaying results other than the mean.

- Correctly displaying negative results.

© **UNIVERSITY OF DUBLIN 2016**