Name: Seán Roche
Student Number: 17332021
Date of Submission: 13/11/17
Module: CS1021

Assignment #1 - Simple Calculator

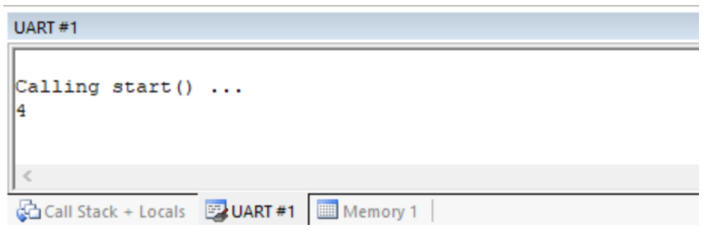
Introduction

For this assignment we were tasked with building a simple calculator capable of evaluating simple arithmetic expressions when entered by a user on a keyboard. We were instructed to do this in three stages.

1. Console Input
2. Expression Evaluation
3. Displaying the Result

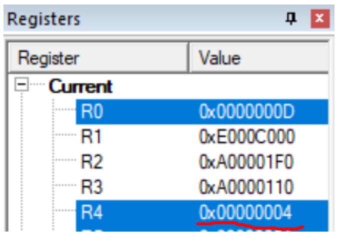
Part 1 – Console Input

For this part we simply had to write a simple program that would allow the user to input a number in decimal form which would be read as an unsigned value and stored in a register.



UART #1

```
Calling start() ...
4
```



Registers

Register	Value
Current	
R0	0x0000000D
R1	0xE000C000
R2	0xA00001F0
R3	0xA0000110
R4	0x00000004

The concept used to achieve this functionality can be summarised as follows:

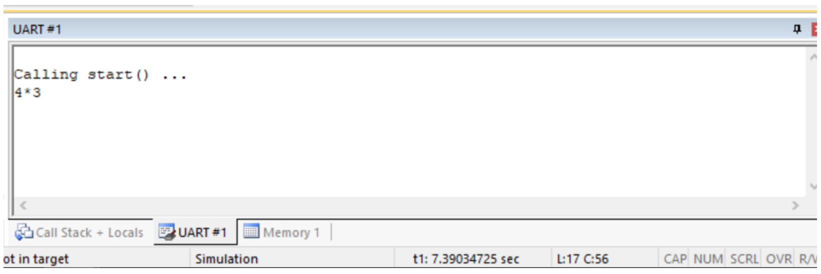
- Starter values were to be set for two registers using a MOV instruction.
- The program would only proceed when the user clicks a digit which was achieved using a branch (BL).
- The program then checks to see if the enter key has been pressed.
- If the enter key has been pressed the input is echoed back to the console and the program stops.

This solution perfectly satisfied our requirements for Part 1. All numbers entered by the user were echoed back to the console in their decimal form and stored in R4 as hexadecimals as required. (See sample test results below)

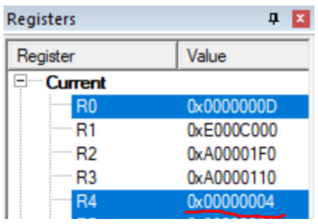
Decimal Input	Decimal shown in console?	Value store in R4
4	Yes	0x00000004
35	Yes	0x00000024
99	Yes	0x00000093

Part 2 – Expression Evaluation

In Part 2 we extended our program to evaluate simple expressions. These expressions involved basic arithmetic with two inputs being added (+), subtracted (-) and multiplied (*).



```
Calling start() ...
4*3
```



Register	Value
Current	
R0	0x0000000D
R1	0xE000C000
R2	0xA00001F0
R3	0xA0000110
R4	0x00000004

- To start, I set about identifying what type of arithmetic or rather what symbol was in question. To do this I simply used the compare instruction to compare the input after the first number (i.e, the symbol) to the ASCII values for the three symbols.
- The corresponding branch could then be used to allow the program to proceed.
- The symbol was then echoed to the console.
- The necessary arithmetic could be carried out using MUL, ADD, SUB commands.
- The program would store the result in R5 and stop.

Again, the program performed as planned in the SUDO code. (See test results.)

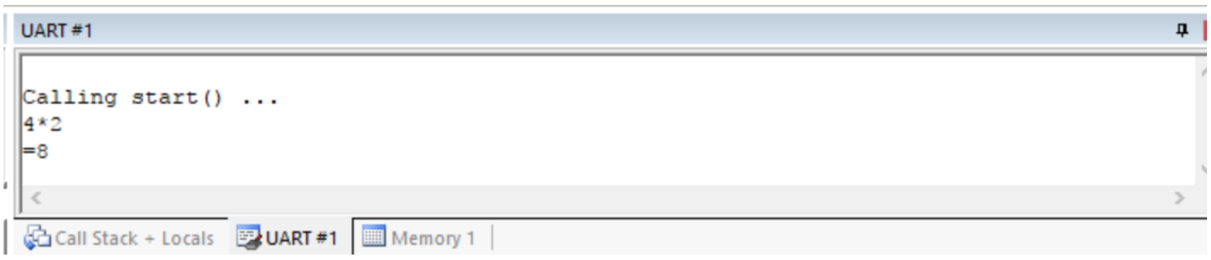
Input	Inputs shown in console?	Value store in R4	Result Correct?
4*3	Yes	0x0000000C	Yes
20 + 20	Yes	0x00000028	Yes
6 - 4	Yes	0x00000002	Yes

SUDO

```
if (input = +) {
....
}
else if
{
else
```

Part 3 – Expression Evaluation

The final part of the program essentially involved displaying the result of arithmetic expressions in the console in their decimal forms.



```
Calling start() ...
4*2
=8
```

- I drafted a concept which aimed to store the each digit in the answer in a register, starting with the least significant figure.
- To achieve this I devised a loop which would incrementally subtract 10 until the remainder was less than 10. This digit would then be taken as the least significant figure.

```
SUDO
while (number<=10) {
(number- 10)
}
```

- Once this figure was stored in a register the loop would be repeated however first the original result number would be decreased by the value of the least significant figure.
- The program would only store two digits at a time in registers and then print them in the right order to display the result.

Unfortunately, this turned out to be a massive failure and in the end I only managed to get the loop to work to display single digit answers. My methodology was flawed as the loop would have to be dynamic to display answers that were a few digits long (as registers would be being used up).

The correct solution may have been to use more branches and comparisons to resolve specific use cases where there were 0's in the middle or at the end of answers. These could then be printed using a less flawed loop.

Input	Inputs shown in console?	Value store in R4	Result Correct?	Result displayed?
4*3	Yes	0x0000000C	Yes	No
20 + 20	Yes	0x00000028	Yes	No
6 - 4	Yes	0x00000002	Yes	Yes