CMPE-250 Assembly and Embedded Programming

Laboratory Exercise 2

Basic Arithmetic Operations

By submitting this report, I attest that its contents are wholly my individual writing about this exercise and that they reflect the submitted code. I further acknowledge that permitted collaboration for this exercise consists only of discussions of concepts with course staff and fellow students. Other than code provided by the instructor for this exercise, all code was developed by me.

Andrei Tumbar
09-01-20

Lab Section:	1
Instructor:	Melton
TA:	Cameron Hudson
	Chris Augsburger
	Anthony Bacchetta
Lecture Section:	1
Lecture Instructor:	Melton

Abstract

In this laboratory exercise, basic arithmetic operations were performed to evaluate students' proficiency with simple assembly instructions. A given arithmetic expression was provided. Intermediate results were calculated before writing the assembly instructions so that the code could be debugged after writing. The assembly instructions included moving constants into registers and performing arithmetic operations such as addition, subtraction, multiplication, and division. Assembly was written for ARM Cortex-M0+ processors.

Procedure

This exercise was built around calculating the value of the following arithmetic expression:

$$18 + ([-11] \div 2) + 13 + (5 \cdot 9) - 42 + 3$$

One of the requirements is that all constants that can be seen in the expression, must be loaded into registers before they are used. Immediate values used in bitshifting operations do not need to be loaded beforehand.

By continuously loading `R0` with the intermediate result of each step, more registers are left free for other operations. To compute the result from the equation above, use `R1` to first store -11 and then divide by 2 (perform with a right-arithmetic shift by one bit). Follow-up by storing 18 into `R0` and adding `R0` and `R1`. This result should be stored back into `R0`.

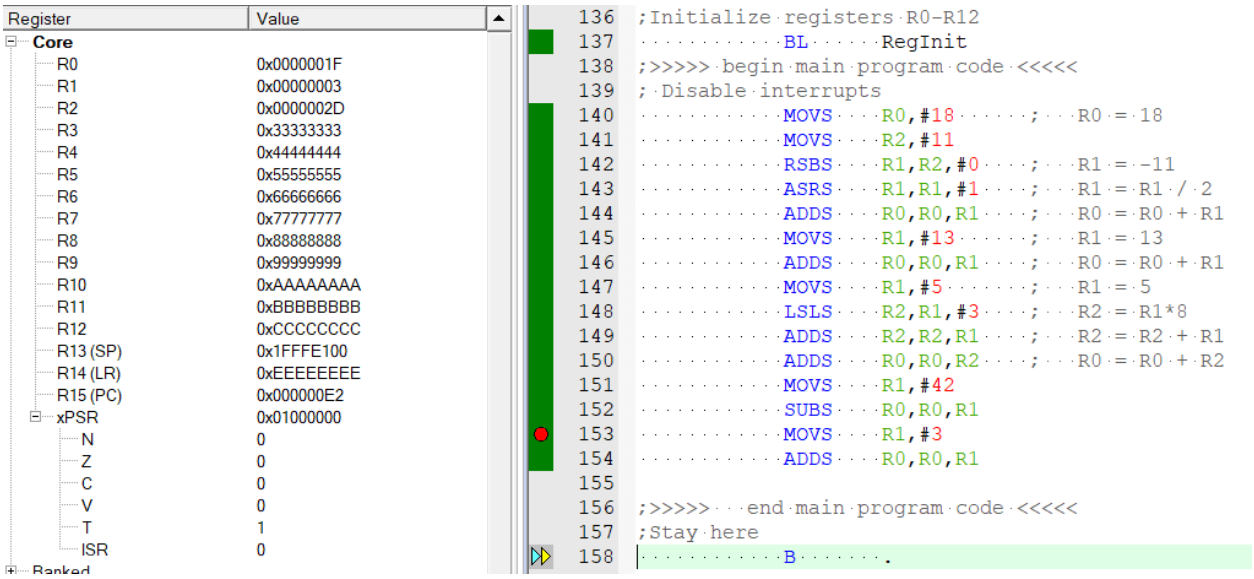
`R0` should now have the result from computing $18 + ([-11] \div 2)$. The rest of the expression can be completed using the same method. The multiplication $5 \cdot 9$ can be performed by calculated $5 \cdot 8 + 5$. Multiplication by a power 2 can also be expressed by the following:

$$\alpha \cdot 2^{\beta} = \alpha << \beta$$

This calculation can be performed using the $R1$ register because the previous result is no loonger needed. The rest of the expression is simply performed using `ADDS` and `SUBS` operations.

Results

The expression should yield the result of 31. This is because $\frac{-11}{2}$ is an integer division and should therefore yield -6 instead of -5.5. 31 in hex is `0x1F`. After the assembly instructions were written, the program was run with the debugger to verify the results. A screen capture was taken after the final instruction was performed.



Register	Value
Core	
R0	0x0000001F
R1	0x00000003
R2	0x0000002D
R3	0x33333333
R4	0x44444444
R5	0x55555555
R6	0x66666666
R7	0x77777777
R8	0x88888888
R9	0x99999999
R10	0xAAAAAAAA
R11	0xBBBBBBBB
R12	0xCCCCCCCC
R13 (SP)	0x1FFFE100
R14 (LR)	0xEEEEEEEE
R15 (PC)	0x000000E2
xPSR	0x01000000
N	0
Z	0
C	0
V	0
T	1
ISR	0
Banked	

```
136 ;Initialize registers R0-R12
137             BL      RegInit
138 ;>>>>> begin main program code <<<<<
139 ; Disable interrupts
140             MOVS    R0,#18      ;   R0 = 18
141             MOVS    R2,#11
142             RSBS    R1,R2,#0    ;   R1 = -11
143             ASRS    R1,R1,#1    ;   R1 = R1 / 2
144             ADDS    R0,R0,R1    ;   R0 = R0 + R1
145             MOVS    R1,#13      ;   R1 = 13
146             ADDS    R0,R0,R1    ;   R0 = R0 + R1
147             MOVS    R1,#5       ;   R1 = 5
148             LSLS    R2,R1,#3    ;   R2 = R1*8
149             ADDS    R2,R2,R1    ;   R2 = R2 + R1
150             ADDS    R0,R0,R2    ;   R0 = R0 + R2
151             MOVS    R1,#42
152             SUBS    R0,R0,R1
153             MOVS    R1,#3
154             ADDS    R0,R0,R1
155
156 ;>>>>>   end main program code <<<<<
157 ;Stay here
158             B       .
```

Figure 1: Debugger results after program execution.

Figure 1 shows the values of every register. Line 137 initialy fills every register with `RN=0xNNNNNNNN`. General purpose registers `R3` and above were never used and therefore still contain the values they were initalized with. `R0` holds the result of the expression in question, `0x1F` or 31 in base 10 which matches the prediction.

Conclusion

This laboratory exercise was important because it taught students basic assembly instructions and program flow as well as how to effectively use the debugger. By calculating intermediate values before the program is written and including pseudo-code along with the assembly instructions, the program was very easy to debug. These basic assembly skills can be applied to more complex programs because they provide the more basic operations and computations.