**CMPE-250 Laboratory Exercise Two**

**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; however, other than code provided by the instructor for this exercise, all code was developed by me.

Joel Yuhas

Performed February 4, 2016

Submitted February 11, 2016

Lab Section L2

Instructor: Muhammad Shaaban

TA: Peter Muller

Lecture Section: 01

Professor: Alessandro Sarra

**Abstract**

Arithmetic operations were used and tested in ARM Cortex-M0+ program. An expression was created in decimal format and then converted and calculated in hex by hand. A program was then written that performed this operation using the ARM Cortex-M0+ program while applying to the programs specific syntaxes. Arithmetic such as bit shift and conversion were covered and were necessary for achieving correct results. The final program produced expected results, and the exercise was successful.

**Procedure**

First, an equation was presented that contained the arithmetic equation which needed to be coded. The equation is shown in equation (1).

-5 + 62 – (9 / 4) – (7 x 9) + 58 + 17 (1)

The expression was not allowed to be simplified and order of operations needed to be taken into account at all times. The operation was solved one step at a time, with each value converted to hex until an answer had been reached. The final answer was a value of 0x00000043 or 67, and its derivation is shown below.

-5 + 62 = 57 = 0x00000039

1

(9 / 4) = 2.25 (floored) = 2 = 0x00000002

2

0x00000039 – 0x00000002 = 0x00000037 = 55

3

(7 x 9) = 63 = 0x0000003F

4

0x00000037 - 0x0000003F = 0xFFFFFFF8 = -8

5

58 + 17 = 75 = 0x0000004B

6

0x0000004B + 0xFFFFFFF8 = 0x00000043 = 67

7

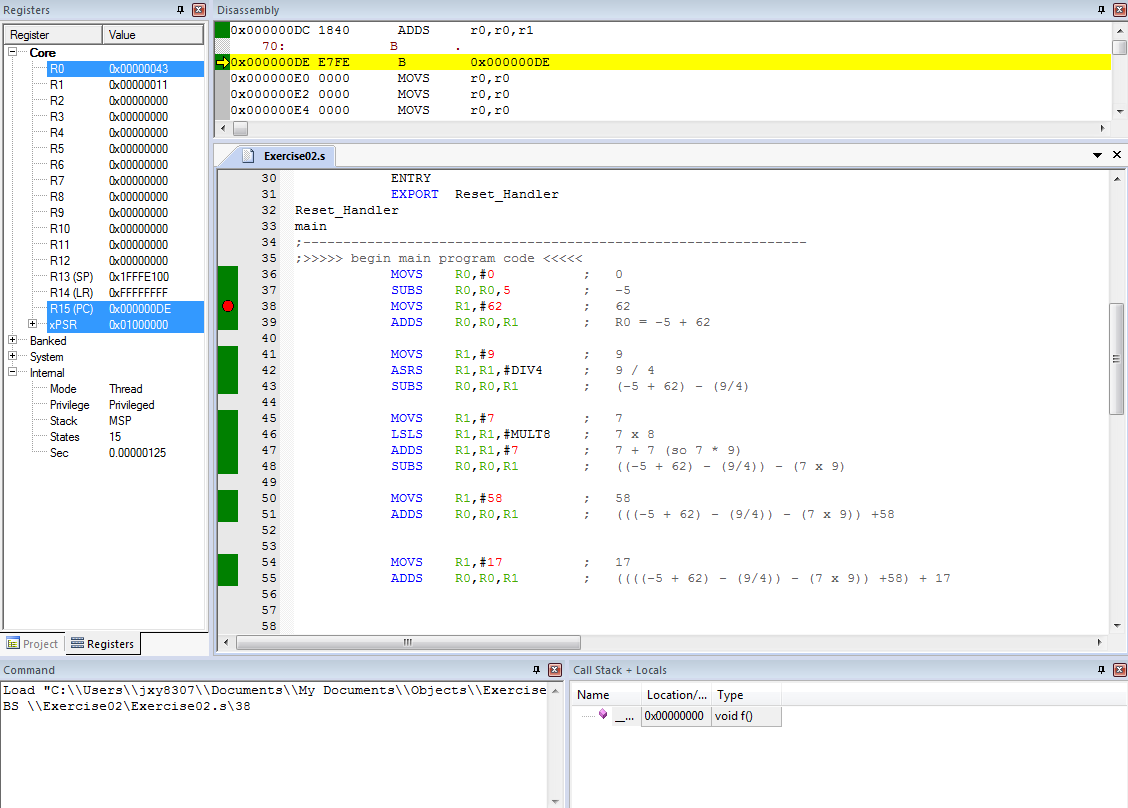
Next, the program itself was coded. In order to obtain a value of -5, register 0 (R0) needed to be loaded with a 0, and then subtracted by 5. In order to do the multiply and divide commands, EQUates were created at the top of the code, which stored the arithmetic for performing the multiplication and division. This was done by utilizing bit shift methods, shifting right to divide by 2n, n being the number of shifts, and shifting to the left to multiply by 2n. In the event that the number needed to be divided by something other than a base of 2, other techniques needed to be employed. Such was the case when 7 was multiplied by 9, 7 had to be multiplied by 8 and then another seven had to be added separately to achieve the desired results.

Once the arithmetic was coded correctly, the program was built and assembled in ARM Cortex-M0+. A listing file and a map file were also created and used for later submission.

Finally, to ensure correctness, the program was ran through a debugger in order to observe the calculations step by step. When the program produced satisfactory results and was deemed complete, a screen capture was taken and the file was saved.

**Results**

Figure 1. shows the screen capture of the final results and code of the program. Notice the comments to the right of the program commands that dictate the arithmetic that is being evaluated. The EQUates are handled near the top, off of the screen but their call symbol (DIV4 and MULT8) are visible in the body of the code. Also notice how 7 is multiplied by 8 and then added by 7 in order to achieve 7 x 9, as mentioned in the procedure. Finally, the left hand side shows the final result stored in register R0 after the debugger was ran, which was the correct predicted value of 0x00000043.



**Figure 1. Finalized Arithmetic Program**

**Conclusion**

Arithmetic operations, as well as implementing the operations into the ARM Cortex-M0+ code, were the main focus of the exercise. The expression given used addition, subtraction, multiplication, and division in such a way that it required that all basic arithmetic operations needed to be executed in the code in order to achieve correctness. Basic concepts like order of operations and bit shifting had to be used as well in order to fully demonstrate how the ARM Cortex-M0+ program handles multiplying and dividing. Unit conversion was handled as well (from decimal to hex) during the hand calculations. Finally, the debugger and its functions were also demonstrated and tested while displaying how the program used hexadecimal numbers by loading said numbers into the registers and performing mathematical operations. This also served as an outlet to practice use with the debugger and further understandings with the program. Ultimately, the exercise served as a way to get familiar with the ARM Cortex-M0+ commands, layout, input and output of data, while also creating a better understanding of performing arithmetic operations correctly in assembly.