Programming Lab 2: Quadratic Equation Solver

EE 306: Introduction to Computing
Professor: Dr. Al Cuevas
TAs: Apurv Narkhede, Nic Key, Ramya Raj, Jerry Yang

Due: 10/29/2018 at 12pm

1 Overview

This lab is intended to familiarize you with the load/store instructions of LC3. By the end of this lab, you should be able to:

1. write and debug about 100-150 lines of assembly code.
2. implement a multiply, a divide, a square and a (brute-force) square-root routine using only ADD, AND, NOT and branch (BR) instructions.
3. use LC3 pseudo-op .FILL to pre-load values in memory.
4. identify addressing modes used by various instructions: immediate, register, PC-offset, register+offset, and indirect.
5. distinguish between and apply the three kinds of load/store instructions in LC3: PC-offset (LEA, LD/ST), register+offset (LDR/STR), and indirect (LDI/STI).

Late submissions will not be accepted; grades will be published within 24hrs of the due date. Regrade requests will only be considered if the autograder is incorrect (see regrade policy below).

Note: You may NOT show anyone other than the TAs or Dr. Cuevas your code.
Any violation of this rule constitutes academic dishonesty.

2 Background

The solutions of the quadratic equation of the form $ax^2 + bx + c = 0$ are given by the equation

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a},$$

where a is non-zero. The term under the radical, $b^2 - 4ac$, is called the discriminant. The discriminant can be used to determine the number of solutions that a quadratic equation has without needing the above equation:

$$\begin{cases} 0 \text{ solutions if } b^2 - 4ac < 0 \\ 1 \text{ solutions if } b^2 - 4ac = 0 \\ 2 \text{ solutions if } b^2 - 4ac > 0 \end{cases}.$$

3 Lab Specifications

In this lab, you will implement a quadratic equation solver routine in LC3 assembly. You may assume that there will be no inputs or calculations that result in a decimal number or overflow your program. You may also assume that $a \neq 0$.

Inputs: The inputs to your program are the coefficients a, b and c of the quadratic equation $ax^2 + bx + c = 0$. They will be stored sequentially at memory location x4000.

1. a will be stored in memory location x4000.
2. b will be stored in memory location x4001.
3. c will be stored in memory location x4002.

Outputs: The outputs of your program will again be stored sequentially at memory location x4000.

1. x4000 - Number of solutions
2. x4001 - First solution (if it exists)
3. x4002 - Second solution (if it exists)

If two solutions exist, the first solution is less than the second solution. Note that if there are less than two solutions to the quadratic equation, you do not need to worry about unused output memory locations (e.g. x4002). You may NOT assume that R2-R7 or any other registers/memory are initialized to any specific value (e.g. zero) before your program starts.

Example test cases:

1. $a = 1, b = 0, c = 0$ should yield 1 solution, $x = 0$.
2. $a = -5, b = -11, c = -10$ should yield no solutions, since the discriminant is less than 0.
3. $a = 6, b = 180, c = 0$ should yield 2 solutions, $x_1 = -30, x_2 = 0$.
4. $a = -1, b = 0, c = 4$ should yield 2 solutions, $x_1 = -2, x_2 = 2$.

These are not the only test cases we will run your code through...there are a total of 20 different test cases. You should come up with at least 5-6 different, interesting test cases to run your code through before submitting. You are welcome to - and should - share your test cases on Piazza.

Hints:

1. Start by writing out the quadratic equation as a series of steps. What's the first step? (Probably to figure out the number of solutions.)
2. You will be reusing the multiply routine several times. Don't rewrite code...we'll teach you how to write subroutines later, but for now, you can copy and paste. You do not - and should not - use the JSR or JSRR instructions.
3. Write and debug your multiply, square, sqrt, and divide routines separately (e.g. in different files) before you put it all together.

4. Since you are guaranteed that there will be no overflow, you can use a "brute-force" method to calculate the square root: start at 1, square it, and see if it matches the discriminant. Do this until you find a match.

5. Use branches wisely. If you only have one solution, do you need to calculate two solutions?

Tips and tricks:

1. Read the entire lab document CAREFULLY before starting.

2. Attend office hours, but don't come without having thought about the problem.

3. Make good comments in your code to save time and frustration when debugging. Use semicolons to make comments in the LC3 editor.

4. Debug as much as possible! The most frustrating thing is when you code fails all of the test cases because of a single mistyped number. If that happens, we won't give credit back. (See regrade policy below.)

5. Manage your time wisely! It will take 2-5x longer to debug your code than it will take to write your code.

4 Submission

1. Make sure you fill out the starter file header with your name, EID and recitation section time.

2. Rename the .asm file "EIDLab2.asm", replacing "EID" with your EID.

3. Submit the .asm file (and ONLY the .asm file) to the Lab 2 Canvas assignment. Do NOT include any other files.

4. If you do not follow these submission instructions exactly, the autograder will spit out a 0!

5 Regrade Policy

We will only award points back if the **autograder** graded your code incorrectly, i.e. your code generated the correct output, but the autograder said you failed the test case. Unfortunately, we will NOT award points back if you fail all the test cases because of a single mistyped character.

If you wish to submit a regrade request, complete the following steps:

1. Review the autograder output, uploaded as a submission comment on your lab submission.

2. Visit Dr. Cuevas or a TA (preferably Jerry) in office hours to comfirm that the autograder output is incorrect **AND** that your code produces the correct output.

3. Email Jerry at jerryyang747@utexas.edu with the following:

   - Subject line: "EE 306: Lab 2 Regrade Request [EID]"
   - Screenshot of the autograder output

- A description of what went wrong
- Who you confirmed the error with
- Your code, attached as an .asm file

After grades are released on Canvas, you have **7 days** to submit a regrade request.