Programming Lab 1: Rotate Register

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

Due: 10/15/2018 at 12pm

1 Overview

This lab is intended to familiarize you with the LC3 ISA and assembly code. By the end of this lab, you should be able to:

- write about 40 lines of assembly code.
- implement a rotate register using ADDs, ANDs, NOTs and branch instructions (BR).
- use the LC3 simulator.
- use the LC3 ISA to translate instructions into assembly.

This lab is 90% of the Lab 1 grade; hence, it is only worth 90 points. 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 talk to or show anyone other than the TAs or Dr. Cuevas your code.
Any violations of this rule constitutes academic dishonesty.

2 Background

A rotate register is a register that can rotate bits either left or right. A *left-rotate* is defined as a left-shift of the register whose most significant bit becomes the least significant bit (Figure 1). Similarly, a *right-rotate* is defined as a right-shift of the register whose least significant bit becomes the most significant bit (Figure 2).

Figure 1. Left-rotate register.

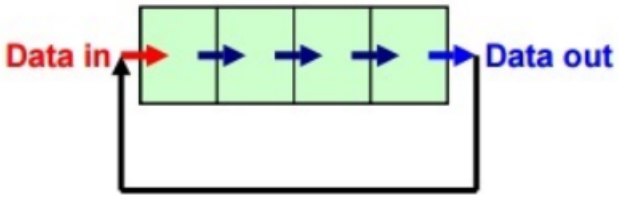

Figure 2. Right-rotate register.

The left-rotate and right-rotate operations are extremely useful in engineering because they preserve the original data in the register, albeit shifted, whereas the left-shift and right-shift operations inherently cause some data loss. Rotate registers are part of a larger class of shift registers that are widely used in communications protocols, delay circuits, and elsewhere.

3 Lab Specifications

In this lab, you will implement a rotate register that rotates in both the left and right directions using LC3 assembly. Both functions of the register must be implemented in the same file.

Inputs:

1. R0 - Bit pattern to rotate.
2. R1 - Number of times to rotate by. If it is positive, rotate left. If it is negative, rotate right. You may assume that you will rotate at most 16 times in either direction.

Outputs:

1. R0 - Rotated bit pattern.

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. R0 = xABCD, R1 = 4 should yield R0 = xBCDA.
2. R0 = x160E, R1 = 7 should yield R0 = x070B.
3. R0 = xA110, R1 = -7 should yield R0 = x2142.
4. R1 = 0 should not change R0, since you are rotating 0 times.
5. R0 = 0 should always output 0. Likewise, R0 = xFFFF should always output xFFFF.

These are not the only test cases we will run your code through...there are a total of 18 different test cases. You should come up with at least 5-6 different test cases to run your code through before submitting.

Tips and tricks:

1. Read the entire lab document CAREFULLY before starting.

2. Attend office hours! This is your first lab, it's not an easy lab, and it only gets harder.

3. Develop an algorithm and have it checked by Dr. Cuevas or the TAs in-person before starting to code. We won't tell you how to code, but we can point you in the right direction.

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

5. 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, unless the autograder fails as a result. (See regrade policy below.)

6. 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 "EIDLab1.asm", replacing "EID" with your EID.

3. Submit the .asm file (and ONLY the .asm file) to the Lab 1 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 1 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.