

## Lab 08: LC-3 Simulator and Programs (Due in class, Mon Oct 19)

### A. Why?

The LC-3 Simulator enables us to run LC-3 programs. Some basic operations, like bit-shifting and integer division, are not built in to the LC-3, so it's good to program them.

### B. Outcomes

By the end of the activity you should

- Have some practice using the LC-3 simulator.
- Know how to shift bits left and right.
- Know how to do simple integer division using repeated subtraction.

### C. In-Lab Practice

- If you have a laptop with you and want to install the LC-3 editor and simulator today, go to “Downloading the LC-3 Simulator” below. Otherwise, in lab, log in and see if the LC-3 editor and simulator are already installed; if so, go to “Editing Files And Running The Simulator” below.

#### (C.1) Downloading the LC-3 Simulator

- Go to the website for the textbook (*Introduction to Computing Systems: From Bits and Gates to C and Beyond*, 2nd ed., by Yale N. Patt and Sanjay J. Patel) <http://highered.mcgraw-hill.com/sites/0072467509/>
- Go to Student Resources → LC-3 Simulator and download the simulator.
- Go back to Student Resources and download a manual too.
- For Windows, execute the downloaded `LC301.exe` file. You'll find it wants to unzip itself; let it do that into an appropriate directory (`C:` or some accessible directory; it's up to you). For Unix, unzip the `lc3tools.zip` file; it will create an `lc3tools` directory.

#### (C.2) Editing files and running the simulator

- On Windows, you'll find that the newly-created `lc3` directory contains two executables, `LC3Edit.exe` and `Simulate.exe`. The `LC3Edit` program lets you edit `*.bin` files and create `*.obj` files from them. Run `LC3Edit`. You'll find that it creates an editor window.

- On Unix: The Unix version of the LC-3 system does not come with its own text editor. You can use the emacs editor mentioned in the Unix lab manual, or any text editor you're familiar with.
- Follow the directions in the LC-3 lab manual up through “*Creating the .obj file for your program*”. (We haven't gotten to assembly language yet, which starts in the middle of page 4.) The instructions involve creating an `addnums.bin` file, from which you create an `addnums.obj` file. On Windows you use the LC3Edit Translate → Convert Base 2 menu item; on Unix, you use the `lc3convert` program.
- Go to Chapter 2 and run the simulator. Work through as much of Chapters 2 and 3 as you can. This will involve executing the `addnums.obj` file and using breakpoints and step-by-step execution, then, go through the multiplication example that begins Chapter 4. (The example ends on page 20.)
- Complete whatever you don't finish at home during the week.
- There's nothing to hand in for this part.

## D. Questions to Hand In on Monday

For this assignment, write out your code (and the answer to Question 4) and hand it in. You'll want to do sample runs in the simulator to make sure your code works, but you don't have to hand in sample runs, just the code.

1. The activity TRAP; LC-3 Simulator for Class 14 included questions about looking at the leftmost bit of a register and at left-shifting a register 1 bit. Let's define an operation “Left-shift R2 one bit into R1:” (1) Left-shift R1 one bit, (2) Find the leftmost bit of R2 and copy it into **the rightmost bit of R1**, and (3) Left-shift R2 one bit. Write an LC-3 code fragment to do this. (Write the code in binary and add comments, as we did in class.) Hint: The shortest answer I know is **5 -4** lines long, but take more if you need.

**Example:** Say R1 is *abcd* and R2 is *wxyz*, where each of *a-d* and *w-z* are either 0 or 1. Then left-shifting R1 makes it *bcd0*, copying the leftmost bit of R2 makes R1 *bcdw*, and left-shifting R2 makes it *xyz0*.

2. The easiest way I know to get the left byte of a register is to left-shift it 8 times into another register. Write a code fragment that clears R1 and then left-shifts R2 eight times into R1. You don't have to repeat your code from Question 1, just write “Left-shift R2 one bit into R1” or some such as your loop body. Put a comment at the top of your code fragment to specify any temporary registers you use. (This

makes it easier for anyone who reads your code to know what registers to avoid using.)

3. The LC-3 doesn't include a divide instruction, so we have to simulate it using repeated subtraction. Here's some pseudocode that calculates a quotient  $q$  and remainder  $r$  given the divisor  $x$  and dividend  $y$ ; the goal is to establish  $x = q*y+r$  where  $0 \leq r < y$ . (We'll assume  $x, y, q, r$  are integers with  $x \geq 0$  and  $y > 0$ .)
 

```

 $q \leftarrow 0$ ;  $r \leftarrow x$ ;      ; Establish  $x = q*y+r$  where  $0 \leq r$ 
while  $r \geq y$ 
     $q++$ ;  $r \leftarrow r-y$ ;      ; Re-establish  $x = q*y+r$  where  $0 \leq r$ 
; After loop,  $x = q*y+r$  where  $0 \leq r < y$ 

```

Write LC-3 code that assumes  $x$  and  $y$  are in R3 and R4 and calculates  $q$  and  $r$ , storing  $q$  in R5 and (to make it more fun) storing  $r$  at the *location pointed to* by R6.

4. Study the pseudocode in Question 3. (a) Why did we assume  $y \neq 0$ ? (I.e., what happens if you run the program when  $y = 0$ ?) (b) What happens if  $x < 0$  and  $y > 0$ ? (Does it calculate  $q$  and  $r$  such that  $x = q*y+r$ ? How large is  $r$ ? What is it compared to 0?) (c) Repeat part (b) assuming  $x \geq 0$  and  $y < 0$ . (d) Repeat part (b) assuming  $x < 0$  and  $y < 0$ .