

Lab 11: Quiz 3 Review

(Nothing for grading)

Quiz 3 will be Wed Nov 11

Selected Outcomes for Lectures 14–17 and Labs 8–10

- **LC-3 Simulator**

- [Not on the quiz; kind of hard to test for it]

- **LC-3 Assembler Language**

- Recognize the different LC-3 assembler instructions.
- Be able to write instructions in LC-3 assembler.*
 - Know the LC-3 assembler version of the data calculation, movement, and control instructions.
 - Know how the LC-3 TRAP instruction is used to read or write a character or halt the program.
- Know what the .ORIG, .FILL, .BLKW, and .END pseudo-instructions do and how to use them.
 - Understand the difference between HALT and END in LC-3 assembler.
 - Understand the difference between FILL and BLKW in LC-3 assembler.
- Understand how the LC-3 assembler produces a symbol table.

- **Programming Structures**

- Know how to translate low-level pseudocode into binary.*
- Know how to implement simple if N/ZP/while N/ZP structures.*
- Know how to implement logical and and or (C/Java's && and || operators).*
- Know how to implement a jump table.
- Be able to traverse an array that ends with a sentinel value.*
- Be able to read a sequence of values that ends with a sentinel value.

- **LC-3 Programs**

- Know how to detect the leftmost bit of a word.
- Have thought about how to look at the leftmost byte of an LC-3 word.
- Know how to shift bits left and right.
- Know how to do simple integer division using repeated subtraction.
- Be able to read a sequence of digits as ASCII characters and convert them to the equivalent nonnegative binary number.
- Be able to convert a nonnegative binary number to a string of ASCII characters, for printing.

You Write Some Questions

- Write two questions that tests the quiz material. Please hand them in and discuss them

Sample Programming Questions

The following questions ask you to write code fragments (you don't have to declare the program's origin, halt the program, or end it).

1. Sketch assembler code to copy R1 to R2.
2. Sketch assembler code that declares a variable P and constants Q, R, and S, initialized to Q=5, R=10, S=8.
3. Sketch assembler code that calculates $P = Q+R-S$. Assume P, Q, R, and S are labels already declared for you.
4. Sketch assembler code that reads an ASCII character "0"–"9" and converts it to the equivalent binary number 0–9. Save the result in R3. You don't have to check to make sure that the character is in the range "0"–"9".
5. Sketch assembler code that tests R6; if $R6 < 0$, set R6 to 1 (else do nothing).
6. Sketch assembler code that checks to see if the value in R1 is 1–10 inclusive; if it is, print the character "Y"; if not, print "N".
7. Sketch assembler code that tests values starting at label X and stops when it finds a value ≤ 0 . (Assume X is already declared somewhere.) Store the location of the ≤ 0 value into R1.

Final Project

- For Mon Nov 9, go through the ideas we discussed in class Wed Nov 4. Pick one or more ideas you think you might want to do for your final project. Expand on the idea; write it out in a form suitable for someone to read.