ECE199JL: Introduction to Computer Engineering **Fall 2012**
Lab 3 **due: 10 p.m. on Tuesday 27 November**

Computing a Boolean Function

In this laboratory assignment, you will write a short program in LC-3 machine code to evaluate a three-input Boolean function. Although almost no one writes such code by hand today, all software executed directly on a computer takes this form, and thus it is useful for you to write a small amount in order to understand the interface between the software and the hardware.

Please read the entire document, including the grading rubric, before you begin programming.

Description:: A three-input Boolean function has been encoded into one byte (half of an LC-3 word) and placed into a memory location. Given the three inputs to the function and the address of the memory location that holds the encoded function, write a program to calculate the function's value.

The inputs to the function are provided to you in registers R1, R2, and R3. Each holds either a 0 (false) or a 1 (true). The function is encoded using the representation in the table below, and the coded function is stored in memory at the address specified by R4. Your program must evaluate the function to either true (1) or false (0) and place the resulting value (0 or 1) into R0.

R3	R2	R1	function defined by
0	0	0	bit 0 of M[R4]
0	0	1	bit 1 of M[R4]
0	1	0	bit 2 of M[R4]
0	1	1	bit 3 of M[R4]
1	0	0	bit 4 of M[R4]
1	0	1	bit 5 of M[R4]
1	1	0	bit 6 of M[R4]
1	1	1	bit 7 of M[R4]

Example:

R1 x0000
R2 x0001
R3 x0001
R4 x4123

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M[x4123]	0	0	0	0	0	0	0	0	1	0	1	1	0	1	1	0

Your program should examine bit 6 of M[x4123] and return R0=0.

Flow Chart: As the first step in writing your program, you must systematically decompose the problem to the level of LC-3 instructions as part of your homework for Friday 16 November. Using the approach described in class and in the textbook, draw the resulting flow chart, labeling the elements with LC-3 assembly or RTL, and turn in a copy.

Specifics:

- Your program must be called `lab3.bin` — we will NOT grade files with any other name.
- Your code must begin at memory location x3000. Remember that the first line in your `.bin` file specifies the starting address of your code (x3000) in binary.
- The last instruction executed by your program must be a HALT (TRAP x25).
- Your program must use an iteration construct to calculate a bit mask that allows you to extract the appropriate bit from the function word using an AND instruction.
- Your program must use a conditional construct to change the result of the AND instruction that extracts the appropriate bit from the function word into a 0 or a 1.
- You may assume that R1, R2, and R3 hold either 0 or 1.
- You may assume that R4 points to a valid memory location.
- You may not make assumptions about the initial contents of R0, R5, R6, nor R7.
- You may use any registers, but we recommend that you avoid using R7.
- Your program must not modify the contents of any memory location.

Tools: Use a text editor on a Linux machine (`vi`, `emacs`, or `pico`, for example) to write your program for this lab. Use the LC-3 simulator in order to execute and test the program. Your code must work on the EWS lab machines to receive credit, so make sure to test it on one of those machines before handing it in.

Testing: You should test your program thoroughly before handing in your solution. *Hint: use the simulator's disassembly ("list") capability to double-check your encodings before running your program.* Remember, when testing your program, *you* need to set the relevant register and memory contents appropriately in the simulator. When we grade your lab, we will initialize the registers and memory for you. Developing a good testing methodology is essential for being a good programmer. For this assignment, you should run your program multiple times for different functions and inputs and double check the output by hand.

Handin: We will tell you how to turn in your program electronically using Subversion (`svn`).

Grading Rubric:

Functionality (40%)
- 25% - program obtains correct bit from coded function
- 15% - program returns answer as 0 or 1

Style (30%)
- 10% - uses a single iteration to calculate a mask bit
- 10% - uses a single conditional to convert AND result to 0 or 1
- 10% - uses spaces correctly to separate fields of instructions

Comments, clarity, and write-up (30%)
- 5% - introductory paragraph clearly explaining program's purpose and approach used
- 10% - code includes table of registers that explains their meaning and contents as used by the code
- 15% - code is clear and well-commented (every line)