ECE 206, Fall 2001: Lab 2

Familiarizing Yourself with the LC-2 Simulator

Learning Objectives

This lab will introduce you to the LC-2 simulator and the environment you will use to write programs for the remaining labs of the semester. It will also introduce you to commands that will be useful in debugging your own programs. This lab will only introduce you to the LC-2 simulator used in the Windows® environment of the ECE206 laboratory. Operation of the UNIX version of the simulator is similar; information can be found on the ECE206L web site.

Prerequisites

You should be familiar with the following concepts:

- Converting between binary, decimal, and hexadecimal number systems
- Identifying basic processor components: registers, PC, memory, etc.
- Multiplying two integers by repeated additions

Preparation

Before you come to lab, read Chapters 1-3 of the *Guide to Using the Windows version of the LC-2 Simulator and LC2Edit*, henceforth called the User's Guide. This document can be found on the ECE 206L home page, and on the Labs page. In Prelab Exercise 1, you will be asked questions that can be answered directly from the User's Guide.

In the text, read Sections 5.1 (all of it), 5.2, 5.4.1, and 5.4.2. You will need to understand how to use the operate instructions (ADD, AND, and NOT) and the branch instructions. Prelab Exercises 2 and 3 cover this material.

Complete the Prelab Exercises and read the Lab Exercises, so you'll know what to do when you get to the lab.

Print Chapter 4 of the User's Guide, and bring it with you to your lab meeting.

Bring a floppy disk to your lab meeting. You will use this to save your work, to be submitted later through the Wolfware system.

Name:_____

Prelab Exercises (30 pts)

If you want to turn this in at the beginning of the lab period, use the sheets below. If you prefer to submit electronically, prepare a *plain text* file, called "`prelab2.txt`", containing the answers (not the questions) and submit it to the "prelab2" assignment for your lab section. Do not submit a Word file, or a PDF file, or a scanned copy of the prelab sheets, etc. This assignment will be open until the beginning of your lab period. Late prelab submissions *will not be allowed*, except for university-excused absences.

1. (10 pts) Based on Chapters 1-3 of the User's Guide, answer the following questions.

   (a) In the upper part of the simulator window, what do the following fields represent?
   - R2:
   - PC:
   - IR:

   (b) What is the Console Window used for?

   (c) What is LC2Edit used for?

   (d) When running the simulator, how can I change the contents of memory location x3200?

2. (10 pts) For each of the following operations, write the corresponding LC-2 instruction (in binary).

   (a) Add the contents of R3 to the contents of R1, and store the result in R5.

   (b) Add 17 to the contents of R4.

Name:_____

(c) Set R0 to R7 AND R0.

(d) Add –2 to the contents of R6, and store the result in R1.

(e) Invert all the bits in R2.

3. (10 pts) Fill in the missing parts of the branch instructions and/or interpretations below. "Address" is the location in memory that contains that instruction. "Target address" is the address of next instruction, if the branch is taken. "Offset" is the offset field specified in the instruction. "Conditions" means the state of the condition bits that will cause the branch to be taken.

   (If you are submitting the prelab electronically, put all the contents in each row, not just the missing parts, leaving spaces between each field.)

Address	*Instruction*	*Conditions*	*Offset*	*Target Address*
x3300	____001010101111	_____	_____	_____
_____	0000_____	N, P	x3C	x483C
x3210	0000100101111001	_____	_____	_____
x4503	0000_____	P	_____	x45B2

Laboratory Exercises (70 pts)

The solution to the lab exercises must be submitted electronically through Wolfware. See the ECE 206L Labs page for instructions, if you don't know how to submit. Each exercise will give details about what to submit.

1. *Running and Tracing a Program (30 pts)*

   In this exercise, we will create an program, load it into the LC-2 simulator, and run it. The program is already written, with instructions expressed in binary form. We will convert this binary machine language program to an *object file* suitable for simulation.

   There is space to answer the questions below, but you must submit your answers as a ***plain text*** *file (not a Word file or other formatted document), named "lab2-1.txt".*

   First, open the LC2Edit program, which is used to create programs and convert them into object files. Using the "File" menu, open the file named "lab2-1.bin". (The lab instructor will tell you where to find it.

   In the edit window, you should see a column of binary numbers. The first number is *not* an instruction; it is the address of the location into which the first instruction of the program will be loaded.

   (a) What is the starting address of this program? (Show your answer in hexadecimal.)

   Each of the other numbers is an LC-2 machine instruction.

   (b) What is the first binary instruction of the program? What will this instruction do when executed? (See Figure 5.1 in the text for a list of all instructions and their formats.)

   (c) What is the last binary instruction of the program? What will this instruction do when executed?

   In order to run the program, we must first convert this text file into an object file. Do this by pressing the "B" button (with an arrow) on the toolbar, or by using the

"Convert Base 2" command in the "Translate" menu. This will create a file named "lab2-1.obj" in the same directory as the binary file.

Now run the "Simulate" program. On the "File" menu, use the "Load Program" command to load the "lab2-1.obj" file.

Set a breakpoint at address x3009. Double-click on the grey dot to the left of the address; it should change into a red "stop sign" icon. Press the "Run Program" button (the green arrow). The program runs until the PC reaches the breakpoint.

(d) List the contents of all general purpose registers (R0 through R7), in hex and in decimal, when the breakpoint is reached.

Use the "Step Over" button (blue curved arrow on the left) a few times, watching the results of executing one instruction at a time.

Remove the breakpoint by double-clicking on the stop sign icon at x3009. Resume running the program (green arrow); let it run for a couple of seconds, then press the "Stop Execution" button (with the red X) to stop it.

(e) List the contents of R0 through R7, in hex and in decimal.

Use the "Breakpoints" button (the red stop sign on the toolbar) to bring up the breakpoints menu. Set a breakpoint to catch when the value of R7 is zero. (See pages 18-19 of the User's Guide.)

(f) Resume execution, and list the contents of R0 through R7, in hex and in decimal, when this breakpoint is reached.

2. *Debugging a Program (25 pts)*

   Work through Example 1 of Chapter 4 in the User's Guide. Submit the binary file (not the object file) of the working multiply program; call it "lab2-2.bin".

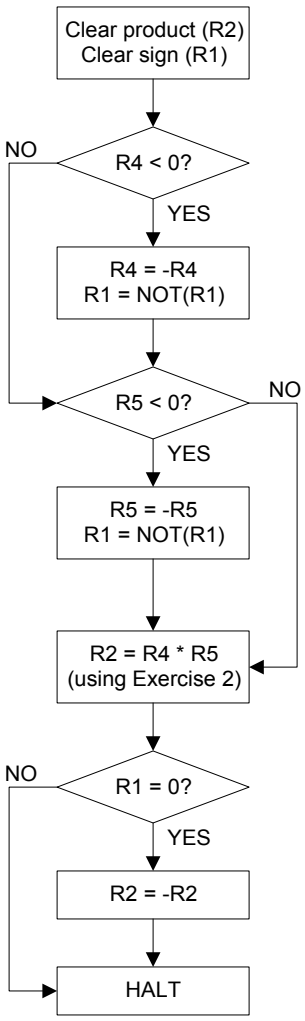
   Include a header in your program file – a set of comments that give your name, your lab section, the date, and a short description of the program.

   Include comments in your program file. Useful comments do not just restate the instruction; they provide an interpretation of the instruction and insight into its role in the program.

   You must provide both a header and program comments to get full credit for this exercise.

3. *Improving the Program (15 pts)*

   The multiply program from Exercise 2 only works for unsigned integers. Create a program that works for both positive and negative integers. On the next page is a flowchart that describes the algorithm. (If you don't know how to read a flowchart, see Chapter 6.)



Here are some tips:

- Clear a register by ANDing it with zero.
- Adding zero to a register will set the condition code (N, Z, P) without changing the register's value.
- To negate a value, flip all its bits and add one.
- The last instruction of the program is Exercise 2 halts the simulation. Do not include this instruction in the "multiply" box shown in the flowchart. Do put it at the end of the program (as shown by the "halt" box).

Include a header and comments in your program. Submit the binary file (not the object file), named "lab2-3.bin".

You must include the header and useful comments to get full credit for this exercise.