ELEN 685: Lab 1

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(R) environment of the ECE206 laboratory. Information about the UNIX version of the LC-2 simulator can be found on the ECE206 web site.

Prerequisites

You should be familiar with the following concepts:

- Converting between binary, decimal, and hexadecimal number systems
- Performing basic logical operations on numbers (x AND y, x OR y, NOT z, etc.)

Prelab Assignment

There is no prelab assignment for this lab.

Laboratory Assignment

This lab is separated into two sections. The first describes the LC-2 simulator and how to create a program and run it using the simulator. The second section walks you through a short debugging exercise by teaching you how to trace a program execution to determine what it is doing, and often, what it's doing wrong.

Section 1: The LC-2 Simulator
For this part of the assignment you will write a simple program that implements a few arithmetic operations.

Step 1: Understanding the Program
The example program below has six different instructions. These instructions are listed below in assembly format.

```
AND R0, R0, x0          ; Clear register R0 to zero

AND  R1, R1, x0         ; _____

AND  R2, R2, x0         ; _____

AND  R3, R3, x0         ; _____

AND  R4, R3, x0         ; _____

AND  R5, R5, x0         ; _____

AND  R6, R6, x0         ; _____

AND  R7, R7, x0         ; _____

ADD  R2, R1, xE         ; Set R2 to _____

ADD  R1, R1, x4         ; _____

AND  R5, R1, R2         ; _____

NOT  R4, R1             ; Set R4 to _____

TRAP x25                ; Execute TRAP routine x25 (halt)
```

Recall that a semicolon indicates a comment in the LC-2 simulator. Add comments on the lines above that describe what each of the instructions do. What is stored in each of the registers at the end of the program execution? Fill in the list below with each registers contents at the end of program execution.

R0 _____
R1 _____
R2 _____

R3 _____
R4 _____
R5 _____
R6 _____
R7 _____

Step 2: Writing the Program
Open the LC-2 editor by double clicking on the LC2Edit icon located on the desktop. You will type your programs in the top window of the editor. The bottom window will report information as you assemble and convert your programs. For this lab, you will enter the example program in binary. Fill in the first line of the program as

0011000000000000

This tells the simulator to load your program into memory at location x3000. In this class we require all programs to start at address x3000. The remaining lines of your program are the machine-language instructions to be executed. The opcodes and bit fields for each instruction can be found in the text or the user's manual. For instance, the first instruction is:

opcode for AND: 0101
destination register R1: 001
source register R1: 001
immediate addressing mode: 1
immediate value zero: 00000

Here is the rest of the program listing, along with the assembly language version of each instruction. Type each machine instruction (just the binary number) on a separate line.

```
0101001001100000   AND  R1, R1, x0
0001010001101110   ADD  R2, R1, xE
0001001001100100   ADD  R1, R1, x4
0101010001000011   AND  R2, R1, R3
1001100001111111   NOT  R4, R1
1111000000100101   TRAP x25
```

After typing in the remaining instructions, save the file by selecting "File => Save." Since you have written the program as binary code, you should save the file with a ".bin" extension. For example, "lab1.bin" might be an appropriate filename. Save the file on the floppy disk that you brought to lab - insert the disk and save to drive "A:".

Step 3: Converting the Program to True Binary
In order for the simulator to read your program, you must convert the file you've typed in, containing ASCII 0's and 1's, to a format acceptable to the LC-2 simulator. The conversion program transforms your input text consisting of 0's and 1's into true binary (machine code) format. To

convert the program, select the "Translate => Convert Base 2" and select your saved binary file. If you saved your file as "lab1.bin", the true binary format will be written to the file "lab1.obj".

Step 4: Running the LC-2 Simulator
To run the LC-2 simulator, double click the Simulate icon on the desktop. This will open a simulator window and console window. It is now necessary to load the true binary version of your program into the simulator. You want to load "lab1.obj" into the simulator. Select "File => Load Program." This will bring up a dialog box that will ask you to select the filename of your program. Select "lab1.obj" and click "Open". The display of memory will be updated to show its new contents. To run the program select "Simulate => Run Program." Verify that the registers contain the correct data after program execution. Demonstrate your working program to the lab instructor.

Section 2: Tracing a Program
Your task for the second part of this lab is to explain what the program "sample.bin" does by tracing it and examining the values it produces in registers and certain memory locations during its execution. You will use the LC-2 simulation tools to do this. The following steps are provided for you to accomplish the task.

Step 1: Initial Set Up
You will need to open the file "sample.bin" using the LC-2 editor. The file is located in c:\lc2\examples\lab1. Once the file is opened, convert the binary representation to true binary following the instructions in Section 1: The LC-2 Simulator Step 3: Converting the Program to True Binary.

Step 2: Setting Up the Program
Open the Simulator and load the converted "sample.obj" file into the simulator' s memory as specified in Section 1: The LC-2 Simulator Step 4: Running the LC-2 Simulator. With the program loaded into the simulator memory you can now enter input data to the program. To enter the input data select "Simulate => Set Value." This will bring up a dialog box that will ask you for the register number or memory location and the value you wish to set it to. Type the following in the corresponding boxes:

Location: x3100
Value: xA21C

This will put the hexadecimal value xA21C into the memory location x3100. Use the scroll bar next to the memory display to find memory location x3100. Notice xA21C is stored there. You can also fill in a value in the "Jump to:" box and press Enter to select the region of memory you want to view. Once you have verified xA21C is in x3100, scroll back to location x3000.

Step 3: Tracing the program
Breakpoints allow a program to execute until the Program Counter reaches a particular address, and then stop execution. This is useful for examining the contents of locations at various points in the execution of a program. To set a breakpoint at a particular point in the program, double click the gray square next to the memory location at which you want to stop. Set a breakpoint at the following locations:

x3002
x3004
x300D

You should notice the gray squares change to red circles when the breakpoint is set. You can delete breakpoints by double clicking the red circle.

You can step through the program one instruction at a time by selecting "Execute => Step Over" or by clicking the  button. The simulator will execute one instruction. You should notice the program counter advance one memory location. You can run the entire program as described above. The simulator will stop each time a breakpoint is encountered.

Step 4: What to Report
Examine and list the values in register 1 and register 2 every time the simulator stops at breakpoint x3002.

R1: _____ R2: _____

R1: _____ R2: _____

R1: _____ R2: _____

R1: _____ R2: _____

R1: _____ R2: _____

R1: _____ R2: _____

R1: _____ R2: _____

R1: _____ R2: _____

R1: _____ R2: _____

Explain what the program does, in 15 words or less.

_____
_____