Lab 5 Preparation Guide

Lab 5 requires that you make use of the LC-3 tools installed on the lab machines. This document will give you some guidance as to how to make use of those tools. Nothing in this document is graded, nor need you turn anything in. You are welcome to learn the tools in any way that you find suitable. However, if you do not learn the tools, you may find it difficult to complete Lab 5.

In this guide, you will learn how to write a program in the LC-3 in machine language and run it in the LC-3 simulator. You will run an extremely simple program which has been provided for you. You will then make a small modification to the program and run it again.

A Sample Program
Update your Subversion repository to obtain the **lab5prep** subdirectory, which contains the file **prep5.bin** with the following contents:

```
0011 0000 0000 0000  ; specifies where the program will be stored (x3000)

0101 010 010 1 00000 ; clear R2 by ANDing it with x00000
0001 010 010 1 01100 ; add the decimal number 12 to R2, storing result in R2

1111 0000 0010 0101  ; halt
```

All text that appears after a semicolon is called **comments** and is ignored by the computer. Their purpose is to make the code readable. Without them, it would be difficult to know what is going on in the above program, but with them it is much easier. **All of the code you write should be well-commented.**

Let's read through this sample program:
- The first line of the sample program specifies where in the LC-3's memory space our program will be stored. In this case, the program will be stored starting at 0x3000.
- Don't mistake the first line of an LC-3 program for an instruction. This line specifies where the program is stored in memory.
- The second line is an **AND** instruction. **Refer to Appendix A in the textbook for a complete description of the instruction set of the LC-3.**
- The third line is an **ADD**. (Note that it is ADD and not AND—the two look similar and can be easily confused with one another)
- The final line is a **TRAP** instruction which is being used here to halt the program.

Before moving on, you should have done the following:
- Read through the sample program
- Look up AND, ADD, and TRAP in Appendix A
- Understand each line of the program

LC-3 Convert
Before we can run the program, it needs to be converted from its text format into an executable **object file**. To do this, we need to use a program called `lc3convert`. `lc3convert` converts a text file containing a program in binary (or hexadecimal) into an object file which can be run by the simulator.

Open a terminal and use 'cd' to navigate to the directory where the sample program is stored. Use 'ls' to display the contents of the directory (you should use this often)—you should notice a file named `prep5.bin`. Run the following command:

<p align="center"><code>lc3convert prep5.bin</code></p>

Now use 'ls' to look at the contents of the directory. You should notice a new file `prep5.obj` has appeared. This is our **object file** which can be run in the simulator.

Before moving on, you should have done the following: use `lc3convert` to convert `prep5.bin` into an object file.

LC-3 Simulator
Since we don't have an actual LC-3 computer, we will need to run our LC-3 programs in a **simulator**. There are two simulator programs available to us: `lc3sim` and `lc3sim-tk`.

First, we will use `lc3sim`. Run the following command, taking note that we are giving `lc3sim` the **object file** (`prep5.obj`) and NOT the human-readable text file (`prep5.bin`).

<p align="center"><code>lc3sim prep5.obj</code></p>

A bunch of text will appear on the screen. Only the last few lines are interesting to us—they show us the contents of the LC-3's registers <u>before</u> our program has been run.

You will also notice that instead of our usual shell prompt we see " `(lc3sim)` " at the bottom of the terminal. **This means we are not at the shell anymore, so our usual commands such as 'cd', 'ls', 'pwd', etc. will not work!** We can only use `lc3sim` commands here. **To see a list of the `lc3sim` commands, use 'help'.** If we wanted to quit the `lc3sim` program and return to the shell, we would use 'quit'—but don't do this yet! We still want to run our program.

To run our program, we will use the 'continue' command:
<p align="center"><code>continue</code></p>
Cool tip: `lc3sim` accepts **abbreviated commands**, meaning that you can actually type '`cont`' or '`c`' instead of '`continue`'.

After our program has finished running (a fraction of a second), the contents of the registers are displayed again. **Compare the contents of the registers before you ran the program to the contents after you ran the program.** (If they are far apart on your screen because you had typed 'help' to display the list of the commands, you can use 'reset' and 'continue' again to display

them close together.) The difference between the registers' contents will be subtle. **What is the difference? Is it what you expected?**

Before we exit `lc3sim`, we will explore one other `lc3sim` command—'step'. Reset your LC-3 simulator by using the 'reset' command. Now, instead of running the entire program using 'continue', **use 'step' to run a single instruction.** The contents of the registers will be displayed after only running the first step. Notice that the PC (program counter) has incremented by 1. `lc3sim` also displays the next instruction to be run, in this case an ADD.

Use 'step' again to run the ADD command, then look at the contents of the registers again. You should notice the same change that you noticed when you ran the whole program before.

You can **exit `lc3sim` now using the 'quit' command**. (If you choose to keep stepping through the program, you will notice that it steps "into" the halt instruction, and numerous other instructions are encountered which aren't part of our program. Don't worry about this for now.)

The ability to 'step' through a program is a useful tool for **debugging** a program. Think about why this is.

There are many other useful commands which allow you to display and manipulate the contents of registers and memory. Learning how to use them will be useful for future labs and MPs.

Before moving on, you should have done the following:
- Use `lc3sim` to run the program using 'continue'
- Observe the change in the contents of the registers
- Run the program again using 'step'
- Use 'quit' to return to the shell
- Use 'help' to get a list of valid commands.

Graphical Simulator (`lc3sim-tk`)
Another way we can run our program is to use `lc3sim-tk`. Run the following command in your terminal:

<p align="center"><code>lc3sim-tk prep5.obj</code></p>

Ignore the warning message which says no symbols are available. You will see two windows, one of which looks like the image on the next page. We can focus on this one for now.

Using what you learned in the previous section about running your program using `lc3sim`, it shouldn't be difficult for you to figure out how to use the graphical version.

The simulator interface displays the contents of the registers at the top of the screen. It also displays the full contents of the LC-3's memory, including our program. **Notice that our program is stored starting at memory address x3000, just as we specified in the first line of `prep5.bin`.** The program is displayed in hexadecimal here, not in binary like in `prep5.bin`, but if you look closely you can see that the values are the same.
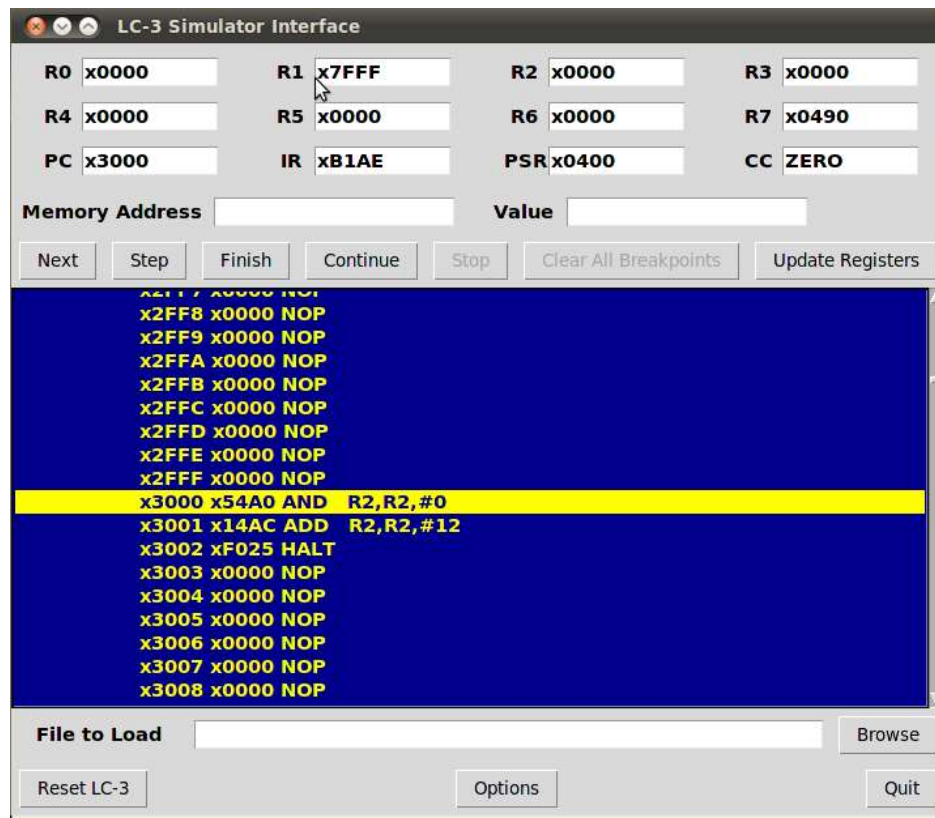
Use the 'Continue' button to run the program. (Don't worry about how the blue window jumps to x0494, which is outside of our program's memory space, after the program has finished running.) **Look at the register contents at the top of the window, particularly R2, and notice what has changed.** You are probably getting used to this by now.

At the bottom-left of the window, click "Reset LC-3" to reset the simulator. Now run your program again using the 'step' button, which you will notice works quite like using the 'step' command in the command-line `lc3sim`.

Before moving on, you should have done the following:
- Load our program in `lc3sim-tk`
- Run the program using 'Continue'
- Reset the simulator
- Run the program step-by-step using 'Step'
- Observe the change to the registers

Modifying the Code

In this section, you will make a small modification to the example program to change its functionality. Originally, the program clears register R2 and adds 12 to this register. Your task is to **make the program add an additional 24 to R2, so that the final stored result is 36. You must do this by adding two instructions** to `prep5.bin`. **Normally you can just double click on the file to open it. If that doesn't work, you can open it with the file editor `gedit` (right click on the file and choose open it with other application).** You may add your instructions as indicated below:

```
0011 0000 0000 0000  ; specifies where the program will be stored (x3000)

0101 010 010 1 00000 ; clear R2 by ANDing it with x00000
0001 010 010 1 01100 ; add the decimal number 12 to R2, storing result in R2
;;;PUT YOUR INSTRUCTIONS AFTER THIS LINE;;;

1111 0000 0010 0101  ; halt
```

A possibly obvious hint is that **the instructions you should add will look exactly like another instruction already in the program.**

Refer to Patt and Patel's Appendix A for a description of the ADD instruction. This should give you everything you need to know.

If you think this sounds difficult, you're probably over-thinking it. It's easy. Ask for help if you don't understand.

After you've modified `prep5.bin` in your text editor, save the file. Use `lc3convert` again to convert `prep5.bin` into an object file – it will overwrite the object file we made earlier.
If `lc3convert` gives you errors, **stop** and go back to `prep5.bin` in your text editor. Find the formatting mistake in your file (for example, wrong number of binary digits on a line, or forgot to use semicolon before your comment) and fix it. Save the file and try `lc3convert` again. If you can't get it to convert successfully, ask for help.

Now you can run your object file in the simulator – you may use either `lc3sim` or `lc3sim-tk`. Run your program step-by-step, watching the value contained in R2. At the end of your program, does R2 contain what you expected it to contain?
Before moving on, you should have done the following:
- Add two instructions to `prep5.bin` to add 24 to R2
- Write a comment on each instruction you added explaining what it does
- Save `prep5.bin` and use `lc3convert` to make an object file
- Run your object file in the simulator
- Use 'step' to watch changes in R2 and notice when they occur

You're done! Feel free to play around with adding other kinds of instructions and watch their effects in the simulator. **You will learn a lot by following your curiosity and experimenting!**