

# ECE 190: The LC-3 Tools Tutorial

Spring 2007 Edition

Last modified February 15<sup>th</sup>, 2007

---

## 1. Introduction

This tutorial will familiarize you with the tools available for building, testing, and running your LC-3 MPs. MP1 will be a machine code MP; you will translate your work into a series of 1's and 0's that will be converted into an object file that can be run by our simulator. MP2 will be written in LC-3 assembly, which will let you take advantage of a number of features peculiar to the LC-3 assembler.

You will need to have completed the vim tutorial and have an updated version of *test.asm* before you can complete this tutorial.

---

## 2. Conversion and Assembly

Begin by entering your ECE 190 work directory (if you aren't there already) with the *ece190* command. You should have a modified copy of *test.asm* in this directory; if there is a line at the bottom that says "I don't assemble! Replace me!", you need to complete Lesson 5 of this tutorial first.

Before you can test your code in the LC-3 simulator, you must convert it into an object file that the simulator can interpret. The program you use to do this depends on the type of file you are trying to convert. If your file is in machine code (a series of 0's and 1's, as it will be for MP1), use the command

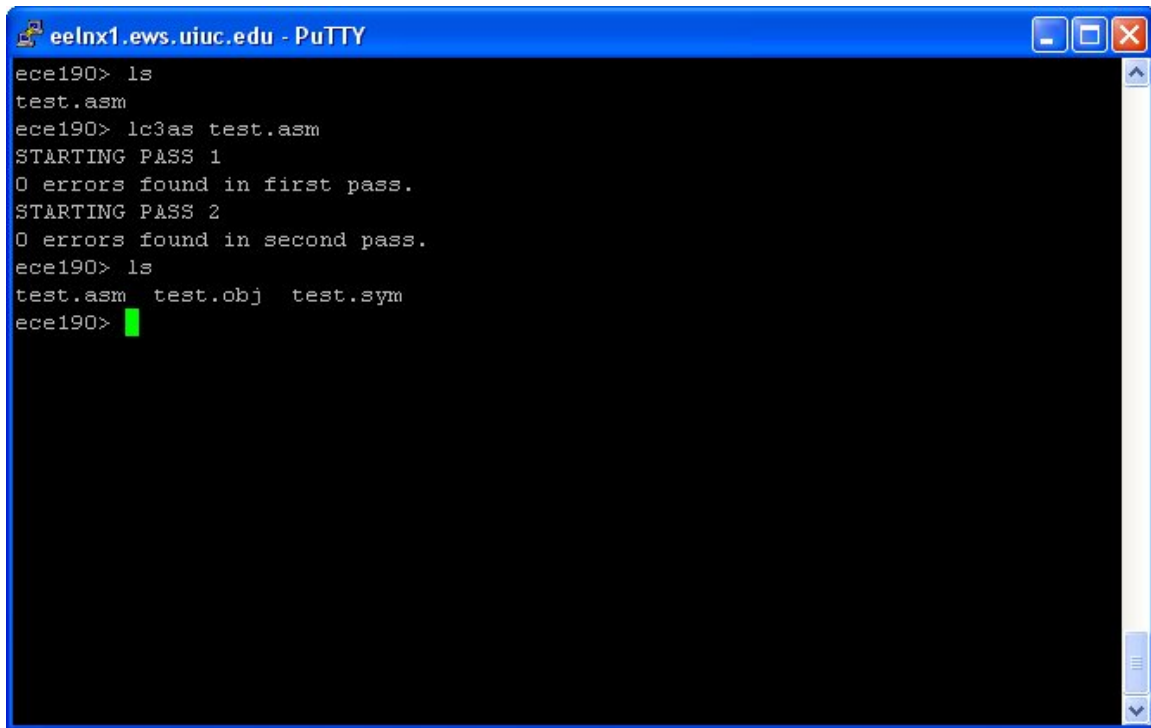
```
lc3convert file.bin
```

to generate an object file. If your file is an assembly file (as it will be in MP2 and as it is in this tutorial), you use

```
lc3as file.asm
```

to create an object file. Converting or assembling *file.bin* or *file.asm* will create a file in the same directory named *file.obj*, which is the actual object file that the simulator can run. You may also see a file named *file.sym*, which is the symbol table generated by the assembler; we'll discuss symbol tables in more detail later on in the course.

You should assemble the *test.asm* file you have so that you can open it in the simulator. A screenshot of this process is at the top of the next page.

A screenshot of a PuTTY terminal window titled "eelnx1.ews.uiuc.edu - PuTTY". The terminal shows the following commands and output:

```
ece190> ls
test.asm
ece190> lc3as test.asm
STARTING PASS 1
0 errors found in first pass.
STARTING PASS 2
0 errors found in second pass.
ece190> ls
test.asm  test.obj  test.sym
ece190>
```

A green cursor is visible on the line "ece190>".

Figure 1: After assembling test.asm

Notice the presence of the *test.obj* file. Next we will open this file in the LC-3 simulator.

---

### 3. The LC-3 Simulator

To start the LC-3 simulator, type *lc3sim test.obj* and press enter. This will open the object file in the simulator and display a command prompt. The LC-3 starts off halted; at the bottom of the screen, you can see the state of the PC, IR, PSR, the register file, and the instruction contained at the current PC. Before continuing, let's confirm that the object file contains our actual code. Type *list* and hit enter; you will see the screenshot at the top of the next page.

The *list* command prints out the contents of memory near the current value of the PC. This allows you to quickly examine what instructions you are about to execute.

Our next task is to actually try the program out. The *continue* command tells the simulator to execute commands until it hits a breakpoint, a HALT trap, or encounters some sort of error. The results of the *continue* command are also on the next page.

```
eelnx1.ews.uiuc.edu - PuTTY
x0494 x0FF9 BRNZP TRAP_HALT
Loaded "test.obj" and set PC to x3000
(lc3sim) list
x2FF6 x0000 NOP
x2FF7 x0000 NOP
x2FF8 x0000 NOP
x2FF9 x0000 NOP
x2FFA x0000 NOP
x2FFB x0000 NOP
x2FFC x0000 NOP
x2FFD x0000 NOP
x2FFE x0000 NOP
x2FFF x0000 NOP
x3000 xE205 LEA R1,MESSAGE
x3001 x6040 LDR R0,R1,#0
x3002 x0402 BRZ DONE
x3003 xF021 OUT
x3004 x1261 ADD R1,R1,#1
DONE x3005 xF025 HALT
MESSAGE x3006 x0048 .FILL 'H'
x3007 x0065 .FILL 'e'
x3008 x006C .FILL 'l'
x3009 x006C .FILL 'l'
(lc3sim)
```

Figure 2: 'list'ing the instructions near x3000

```
eelnx1.ews.uiuc.edu - PuTTY
x2FFB x0000 NOP
x2FFC x0000 NOP
x2FFD x0000 NOP
x2FFE x0000 NOP
x2FFF x0000 NOP
x3000 xE205 LEA R1,MESSAGE
x3001 x6040 LDR R0,R1,#0
x3002 x0402 BRZ DONE
x3003 xF021 OUT
x3004 x1261 ADD R1,R1,#1
DONE x3005 xF025 HALT
MESSAGE x3006 x0048 .FILL 'H'
x3007 x0065 .FILL 'e'
x3008 x006C .FILL 'l'
x3009 x006C .FILL 'l'
(lc3sim) continue
H
--- halting the LC-3 ---
PC=x0494 IR=xB1AE PSR=x0400 (ZERO)
R0=x0000 R1=x7FFF R2=x0000 R3=x0000 R4=x0000 R5=x0000 R6=x0000 R7=x0490
x0494 x0FF9 BRNZP TRAP_HALT
(lc3sim)
```

Figure 3: The results of running our program

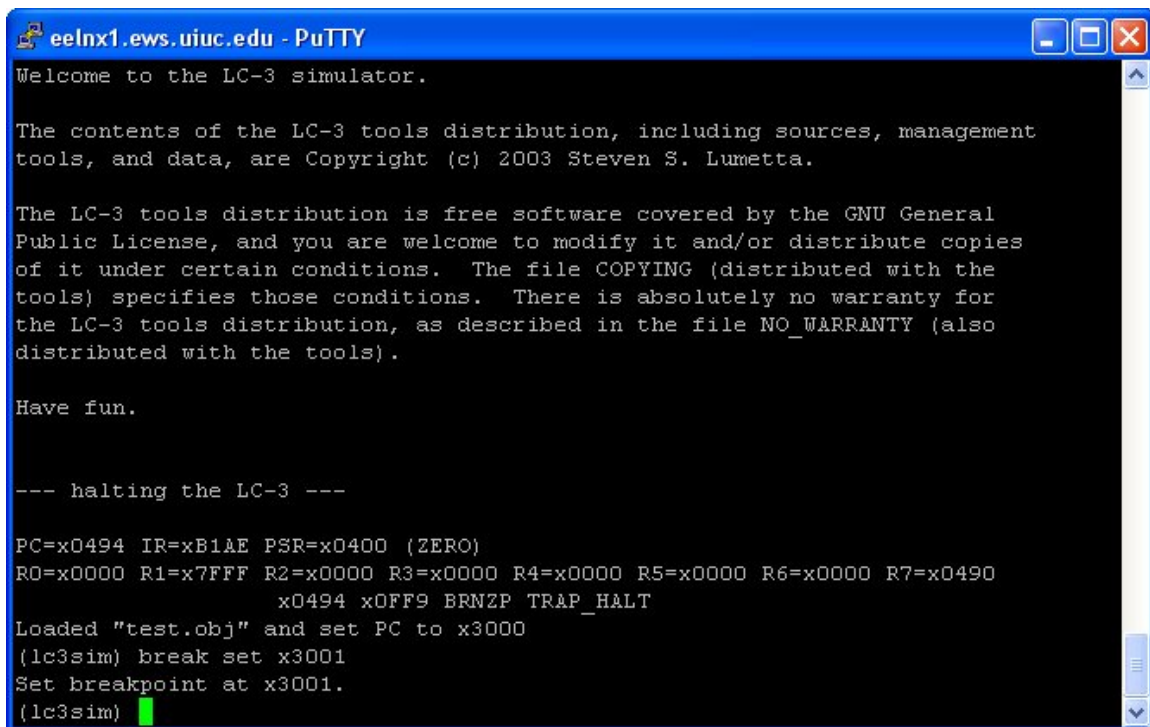
The program was supposed to print “Hello world!” but only got through the ‘H’ before giving up the ghost. Something must be wrong with the code. Let’s take a quick look at how to control execution of code in the simulator.

---

#### 4. Execution Control in the LC-3 Simulator

First, let’s reset our program. Issue the *reset* command to the simulator. It will reload the object file from scratch and reset the PC and register file.

We’ll debug the code by setting a *breakpoint*. A breakpoint is a stop sign that you insert in the code. When the simulator is about to execute a line with a breakpoint, it stops and returns control to you so that you may examine the state of the processor. Set a breakpoint at memory address x3001 by typing *break set x3001*. You will see a picture like the one below.

The image shows a terminal window titled "eelnx1.ews.uiuc.edu - PuTTY". The text inside the terminal is as follows:

```
Welcome to the LC-3 simulator.

The contents of the LC-3 tools distribution, including sources, management
tools, and data, are Copyright (c) 2003 Steven S. Lumetta.

The LC-3 tools distribution is free software covered by the GNU General
Public License, and you are welcome to modify it and/or distribute copies
of it under certain conditions. The file COPYING (distributed with the
tools) specifies those conditions. There is absolutely no warranty for
the LC-3 tools distribution, as described in the file NO_WARRANTY (also
distributed with the tools).

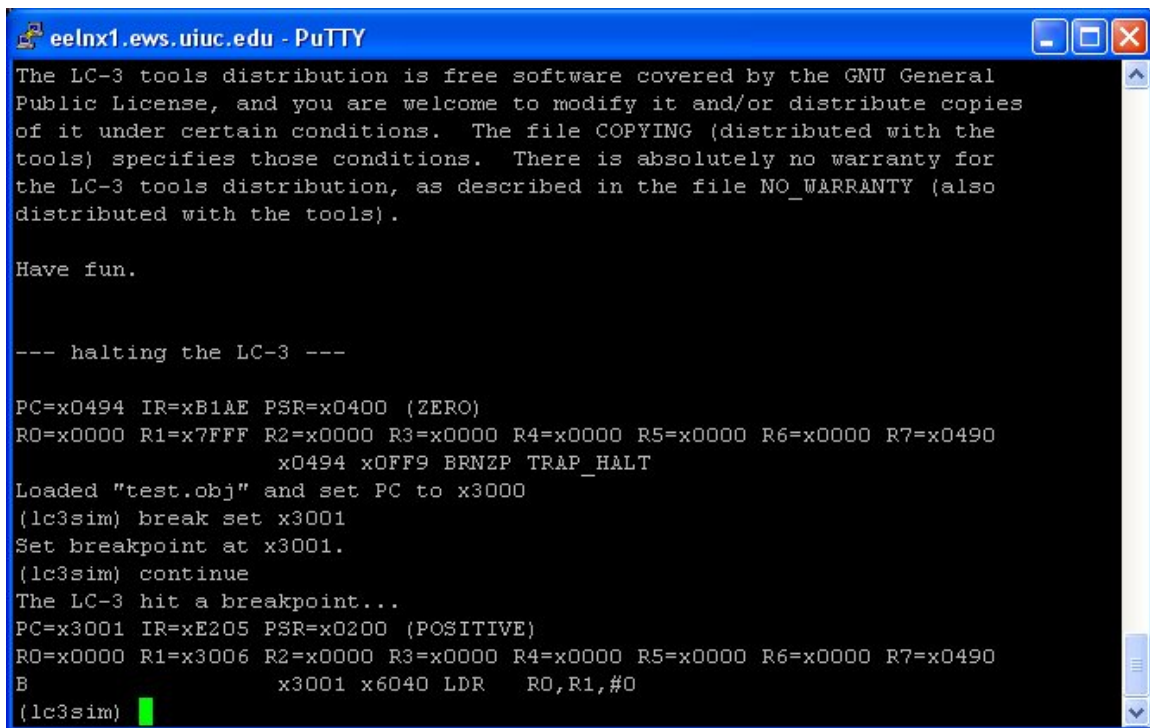
Have fun.

--- halting the LC-3 ---

PC=x0494 IR=xB1AE PSR=x0400 (ZERO)
R0=x0000 R1=x7FFF R2=x0000 R3=x0000 R4=x0000 R5=x0000 R6=x0000 R7=x0490
x0494 xOFF9 BRNZP TRAP_HALT
Loaded "test.obj" and set PC to x3000
(lc3sim) break set x3001
Set breakpoint at x3001.
(lc3sim) █
```

Figure 4: After setting a breakpoint at x3001

Now run the program again with *continue*. The simulator will inform you that it has hit a breakpoint and display the processor state, as shown in the figure on the next page. You may now execute instructions one at a time with the *next* instruction. Do this until you see the message “Halting the LC-3.”



```
eelnx1.ews.uiuc.edu - PuTTY
The LC-3 tools distribution is free software covered by the GNU General
Public License, and you are welcome to modify it and/or distribute copies
of it under certain conditions.  The file COPYING (distributed with the
tools) specifies those conditions.  There is absolutely no warranty for
the LC-3 tools distribution, as described in the file NO_WARRANTY (also
distributed with the tools).

Have fun.

--- halting the LC-3 ---

PC=x0494 IR=x01AE PSR=x0400 (ZERO)
R0=x0000 R1=x7FFF R2=x0000 R3=x0000 R4=x0000 R5=x0000 R6=x0000 R7=x0490
                                x0494 x0FF9 BRNZP TRAP_HALT
Loaded "test.obj" and set PC to x3000
(lc3sim) break set x3001
Set breakpoint at x3001.
(lc3sim) continue
The LC-3 hit a breakpoint...
PC=x3001 IR=x0205 PSR=x0200 (POSITIVE)
R0=x0000 R1=x3006 R2=x0000 R3=x0000 R4=x0000 R5=x0000 R6=x0000 R7=x0490
B                                x3001 x6040 LDR  R0,R1,#0
(lc3sim)
```

Figure 5: After hitting the breakpoint

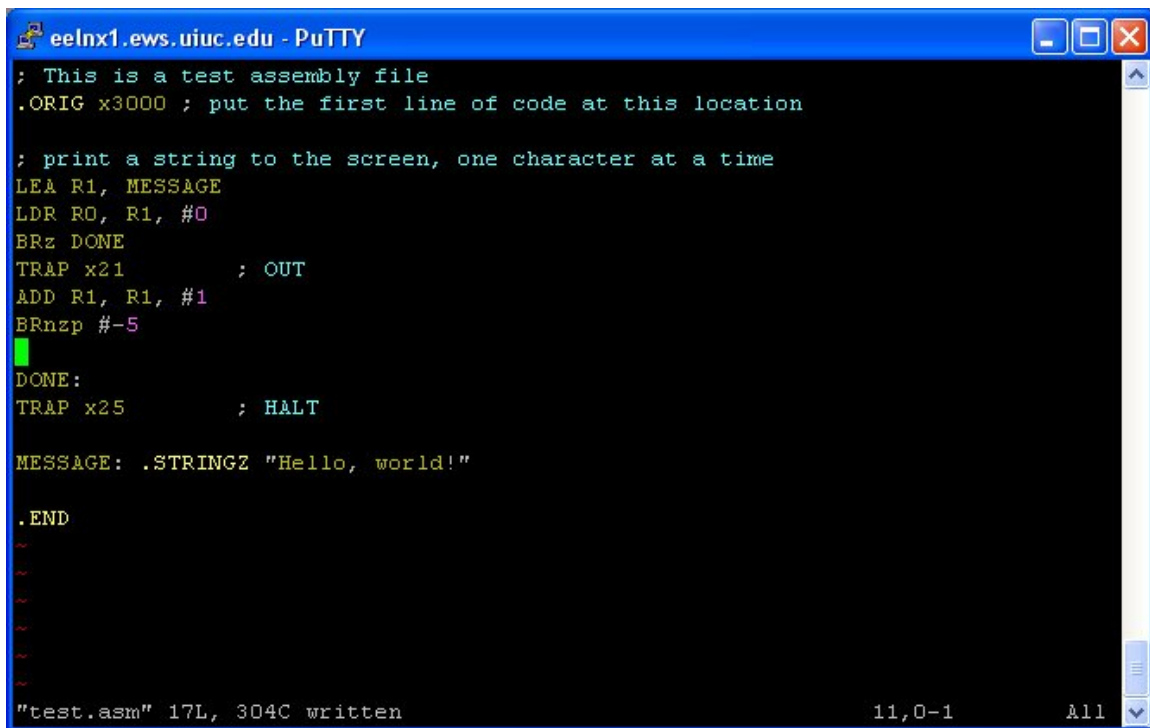
If you examine the instructions executed before reaching the HALT instruction, you'll notice that the OUT instruction, which prints a character to the screen, is only executed once. The code is missing some way to execute this code until it reaches the end of the "Hello, world!" string.

Quit the LC-3 simulator with the *quit* command and open *test.asm* in your text editor. After the line "ADD R1, R1, #1" add the line "BRnzp #-5". This will cause the code to branch back and load another character so that printing may continue, as shown at the top of the next page.

Reassemble with *lc3as*, open it in *lc3sim*, and run the code with *continue*. This time the entire string will be printed.

You now have enough skills to make some use of the LC-3 simulation tools. You should know how to convert or assemble your files, open them in the simulator, start execution with *continue*, set breakpoints with *break set*, and execute one instruction at a time with *next*.

When you are done playing with the simulator, quit and return to the Linux tutorial to complete Lesson 7.



```
eelnx1.ews.uiuc.edu - PuTTY
; This is a test assembly file
.ORIG x3000 ; put the first line of code at this location

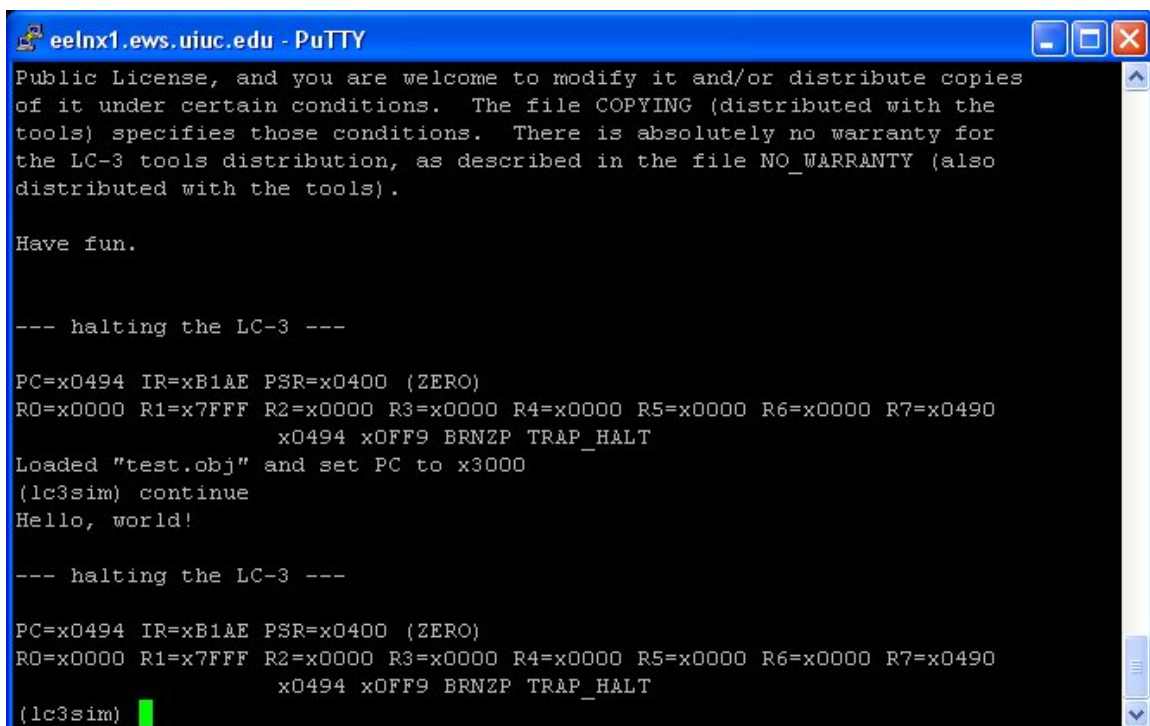
; print a string to the screen, one character at a time
LEA R1, MESSAGE
LDR R0, R1, #0
BRz DONE
TRAP x21      ; OUT
ADD R1, R1, #1
BRnzp #-5
DONE:
TRAP x25      ; HALT

MESSAGE: .STRINGZ "Hello, world!"

.END

"test.asm" 17L, 304C written      11,0-1      All
```

Figure 6: The edited test.asm source



```
eelnx1.ews.uiuc.edu - PuTTY
Public License, and you are welcome to modify it and/or distribute copies
of it under certain conditions.  The file COPYING (distributed with the
tools) specifies those conditions.  There is absolutely no warranty for
the LC-3 tools distribution, as described in the file NO_WARRANTY (also
distributed with the tools).

Have fun.

--- halting the LC-3 ---

PC=x0494 IR=xB1AE PSR=x0400 (ZERO)
R0=x0000 R1=x7FFF R2=x0000 R3=x0000 R4=x0000 R5=x0000 R6=x0000 R7=x0490
x0494 xOFF9 BRNZP TRAP_HALT
Loaded "test.obj" and set PC to x3000
(lc3sim) continue
Hello, world!

--- halting the LC-3 ---

PC=x0494 IR=xB1AE PSR=x0400 (ZERO)
R0=x0000 R1=x7FFF R2=x0000 R3=x0000 R4=x0000 R5=x0000 R6=x0000 R7=x0490
x0494 xOFF9 BRNZP TRAP_HALT
(lc3sim)
```

Figure 7: It works!