CS120 Fall 2015 Lab – Week of October 11th
For lab, we’ll build and run the LC3 simulator for Linux.
1) Download the LC3 source from the course web page.
   http://highered.mheducation.com/sites/dl/free/0072467509/104652/lc3tools_v12.zip
2) Open a terminal window, change to the directory where the source code is, and then do the following:
   % ./configure
   % make
This should set up and compile the LC3 tools for you. MANY open source software packages build this way. If this is your first time doing this, congratulations! You’ll be doing this often over the next few years.

To start the LC3 simulator (do this from a terminal window)
   % ./lc3sim-tk

To run the assembler
   % ./lc3as name-of-the-file-to-assemble.asm

LAB WORK
First up, we’ll convert some assembly code into machine code instructions – this is the stuff we’ve done in class for the past week or so. Yes, I know, it’s not any fun.

Code to convert:
AND R0, R0, #0
ADD R0, R0, #7
AND R1, R1, #0
ADD R1, R1, #5

AND R2, R0, R1
NOT R3, R2

Convert each line to binary (using the charts below), and then convert that into hex.

Enter the hex into the memory of the LC3 simulator (put it at memory location x3000, x3001, ...). Change the PC to x3000.

Single step through the instructions – make sure you know what will be in registers 0 through 3 after you execute the NOT instruction. This is EXACTLY the sort of question that would show up on the next exam....
Next, we'll use the LC3 assembler, to make life easier. Enter the following into a text file (add the extension ".asm").

```
.ORIG x3000
  LD R0, VALUE1
  LD R1, VALUE2
  AND R2, R2, #0
LOOP  ADD R2, R2, R1
      ADD R0, R0, #.-1
      BRP LOOP
  VALUE1 .FILL 8
  VALUE2 .FILL 7
.END
```

Run this through the assembler, and then load the file into the LC3 simulator. Step through the code, one instruction at a time. This code will multiply the numbers (in a very stupid way). You can change the numbers in VALUE1 and VALUE2, and try it again. Note that VALUE1 has to be positive for this to work right....

Three important ideas to make sure you catch – if you get this, then lots of other things will be easier...

1) Having the assembler do the translation is a lot better than doing yourself.
2) LOOP, VALUE1, and VALUE2 are “labels,” and indicate locations in memory. Many of the machine instructions use offsets of different kinds, and the assembler does all of the unpleasant math for you.
3) The machine treated the memory locations for VALUE1 and VALUE2 as “instructions.” The machine DOES NOT KNOW THE DIFFERENCE between instructions and data. They're just 16-bit numbers – how you use them is up to you. Note also that they show up as “NOP” – which is short for NO OPERATION. Nothing happens when you execute them... If you look at the BR instruction, you can probably guess why....