## Department of Electrical and Computer Engineering

### The University of Texas at Austin

EE 460N, Fall 2013

Lab Assignment 2

Due: Sunday, Septemper 22, 11:59 pm

Aater Suleman, Instructor

Stephen Pruett, Abhishek Agarwal, Chirag Sakhuja TAs

## Introduction

For this assignment, you will write an instruction-level simulator for the LC-3b. The simulator will take one input file entitled isaprogram, which is an assembled LC-3b program.

The simulator will execute the input LC-3b program, one instruction at a time, modifying the architectural state of the LC-3b after each instruction.

Note: The file isaprogram is the output file from Lab Assignment 1. This file should consist of 4 hex characters per line. Each line of 4 hex characters should be prefixed with '0x'. For example, the instruction NOT R1, R6 is assembled to 1001001110111111. This instruction would be represented in the isaprogram file as 0x93BF.

The simulator is partitioned into two main sections: the shell and the simulation routines. We are providing you with the shell. Your job is to write the simulation routines.

## The Shell

The purpose of the shell is to provide the user with commands to control the execution of the simulator. The shell accepts one or more ISA programs as arguments and loads them into the memory image. In order to extract information from the simulator, a file named dumpsim will be created to hold information requested from the simulator. The shell supports the following commands:

1. go – simulate the program until a HALT instruction is executed.
2. run <n> – simulate the execution of the machine for n instructions
3. mdump <low> <high> – dump the contents of memory, from location **low** to location **high** to the screen and the dump file
4. rdump – dump the current instruction count, the contents of R0–R7, PC, and condition codes to the screen and the dump file.
5. ? – print out a list of all shell commands.
6. quit – quit the shell

## The Simulation Routines

The simulation routines carry out the instruction-level simulation of the input LC-3b program. During the execution of an instruction, the simulator should take the current architectural state and modify it according to the ISA description of the instruction in [Appendix A](http://users.ece.utexas.edu/%7Epatt/13s.460N/13f.460N/handouts/new_byte.pdf). The architectural state includes the PC, the general purpose registers, the condition codes and the memory image. The state is modeled by the following C code:

#define WORDS\_IN\_MEM 0x08000   
#define LC\_3b\_REGS 8  
  
typedef struct System\_Latches\_Struct{  
  
 int PC, /\* program counter \*/  
 N, /\* n condition bit \*/  
 Z, /\* z condition bit \*/  
 P; /\* p condition bit \*/  
 int REGS[LC\_3b\_REGS]; /\* register file. \*/  
} System\_Latches;  
  
int MEMORY[WORDS\_IN\_MEM][2];

The shell code we provide includes the skeleton of a function named process\_instruction, which is called by the shell to simulate the next instruction. You have to write the code for process\_instruction to simulate the execution of instructions. You can also write additional functions to make the simulation modular.

## What To Do

The shell has been written for you. From your ECE LRC account, copy the following file to your work directory:

[lc3bsim2.c](http://users.ece.utexas.edu/%7Epatt/13s.460N/13f.460N/labs/lab2/lc3bsim2.c)

At present, the shell reads in the input program and initializes the machine state. It is your responsibility to complete the simulation routines that simulate the instruction execution of the LC-3b.

Add your code to the end of the shell code. **Do not modify the shell code.**

The accuracy of your simulator is your main priority. Specifically, make sure the architectural state is correctly updated after the execution of each instruction.

It is your responsibility to verify that your simulator is working correctly. You should write one or more programs using all of the LC-3b instructions and execute them one instruction at a time (run 1). You can use the rdump command to verify that the state of the machine is updated correctly after the execution of each instruction.

Since we will be evaluating your code on linux, you must be sure that your code compiles on one of the ECE linux machines using gcc with the -ansi flag. This means that you need to write your code in C such that it conforms to the ANSI C standard. You should also make sure that your code runs correctly on one of the ECE linux machines.

If you need to copy any text files from Windows to Linux (for example, your C program or your test cases), use the dos2unixprogram to convert them (please see [instructions](http://users.ece.utexas.edu/%7Epatt/13s.460N/13f.460N/labs/port.html)). This program will strip away the extra ‘\r’ end-of-line characters which are not used in Linux.

## What To Turn In

Please submit your code electronically following the posted [instructions](http://users.ece.utexas.edu/%7Epatt/13s.460N/13f.460N/labs/lab2/Lab2Submit.html). You will submit only the lc3bsim2.c file with adequately documented source code of your simulation routines.

## Important

1. In Appendix A, please correct the operation of the JSR/JSRR instruction to read:
2. TEMP = PC†  
   if (bit(11)==0)  
    PC = BaseR;  
   else  
    PC = PC† + LSHF(SEXT(PCoffset11), 1);  
   R7 = TEMP;  
     
   \* PC†: incremented PC
3. Please note that LEA **does NOT** set condition codes.
4. LC-3b registers are 16 bits wide. However, when you perform arithmetic or bitwise operations in C on int data types on the Linux x86 machines you are using 32 bits. Therefore, you must be careful about not keeping the higher 16 bits of the results in the architectural state. The shell code includes a macro called Low16bits that you can use to avoid this problem.