CS270 Programming Assignment 9
"LC-3 Assembler Project"

Goals

For this assignment you will rename and extend your LC-3 parser program from PA7 to make it into an LC-3 assembler. The goals of this programming assignment are:

1. To extend your C skills by modifying an existing C program.
2. To solidify your understanding of bit manipulation in the C language.
3. To write a functional LC-3 assembler that produces code for the LC-3 simulator.

The Assignment

Write an LC-3 assembler program that inputs LC-3 assembly code and outputs LC-3 machine code in the .hex format expected by the *lc3convert* program. The same set of directives, instructions, operands and comments that you processed in PA7 must be handled by the PA9 assembler. You must use the code you wrote for PA7 as a starting point for the assignment. The program should produce a file in the .hex format, which is described in the specification. The goal of the assignment is produce code that is functionally identical to the original assembler. In other words, running the .hex file through *lc3convert* should produce a .obj file that is identical to the one made by running *lc3as* on the original assembly code. Here are some of the steps you may need to perform:

1. Rename lc3parse.c and lc3parse.h to lc3assemble.c and lc3assemble.h and fix the Makefile for the new file names.
2. Add and initialize a field for machine code to the instruction data structure. This field should be an unsigned integer that exactly matches the LC-3 instruction size.
3. Compute the PC offset for instructions that have a reference to a label. You will need to use the current instruction address and the address of the referenced label for the calculation.
4. Write a new function that iterates the instruction array to output the machine code, with something like the following prototype:

void writeMachine(char *outputFile, Instruction instructions[]);

Note: Comment out or remove the call to writeAssembly() in the main program and call writeMachine() instead. This keeps you from having to change the number of command-line parameters.

Program Specification

This section is a specification of the program for the assignment, you will be graded on how closely you follow this specification:

PROGRAM NAME

The name of your program should be *lc3assemble*, with the following command line arguments:

usage: lc3assemble <input file> <output file> <symbol file>

The input file is an LC-3 assembly program. We have provided an example assembly file to get you started. The output file is LC-3 machine code in hexadecimal format.

INPUT SPECIFICATION

Your program must parse all legal LC-3 assembly code, with the following exceptions:

- .ORIG, .END, .BLKW, and .FILL are supported, .EXTERNAL and .STRINGZ are not
- Labels cannot exceed 10 characters.
- The line length in the assembly file cannot exceed 127 characters.

OUTPUT SPECIFICATION

The first line is the starting code address, the remainder are instructions or data. All values are 4-digit hexadecimal numbers without a preceding 'x' or '0x'. Please do not include comments in your output, they are shown in the .hex example below in order to clarify the format.

3000 ; start program at location x3000
2206 ; load location 3007 into R1
2406 ; load location 3008 into R2
56E0 ; AND R3,R3,0
16C1 ; ADD R3,R3,R1
16C2 ; ADD R3,R3,R2
3603 ; store R3 into location 3009
F025 ; halt
1111 ; location 3007 = 0x1111 (constant)
0006 ; location 3008 = 0x0006 (constant)
0000 ; location 3009 = result (variable)

ERROR HANDLING

The program must handle the same error conditions as PA7 by reporting the exact error message specified below in quotes, then exiting the program:

- "Duplicate Symbol", caused by more than one label of the same name.
- "Invalid Operation", caused by an invalid operation or directive.
- "Invalid Operands", caused by incorrectly specified or the wrong number of operands.
- "Out of Range", caused by an out of range immediate value or offset.
- "Invalid Reference", caused by a reference to a nonexistent label.

The program must also free all allocated memory, and compilation must be without errors or warnings.

SOURCE CONTROL

You must develop your program under source control, as described in the R13 specification. The name and location of the repository are up to you, but it must meet the following requirements:

- All of the source files and the Makefile should be stored in the repository.
- At least five revisions of lc3assemble.c should be stored, spanning a period of at least a week.

To verify usage of source control, you will run the following commands and include the output in a README file which must be included in the pa9.tar file you submit:

linux prompt> svn log lc3assemble.c
linux prompt> svn log lc3assemble.h
linux prompt> svn log Makefile
linux prompt> svn info
linux prompt> svnlook tree <repository path>

Note: The *svnlook* command must be run on a department system.

Submission Instructions

1. Make sure that your tar file is named pa9.tar, all lowercase.
2. Include a Makefile to build your program, an example is provided.
3. Make sure that your tar file unpacks to a directory named PA9.
4. Test your tar file in another directory using $ tar xvf pa9.tar
5. Comment headers are required for each file and function.

Grading Criteria

To grade the assignment, we will examine and run the program on the example.asm file (20 points), our own test files (40 points), making sure that a correct .hex file is generated in both cases. 10 points will be given for following the instructions in the assignment. The remainder of the grade will be given for style considerations: commenting, indentation, naming (30 points). For the first time this semester, points will be deducted for excessive code size (-5 points), compiler warnings (-2 points per, up to -10 points) and compiler errors (-10 points per, up to -30 points).