README

Assignment #2 – SIC/XE Assembler
CS-530, Spring 2024

Team:
Carlos Lopez, cssc4002, RED ID: 827117558
David Granda-Ventura, csscxxxx, RED ID: 824371438

Files:

- asxe.cpp - This file is the main source file for our two pass assembler. It reads in a SIC/XE source file, performs the first and second pass, then output a symbol and library table.
- Directives.cpp - this file contains a dictionary of the SIC/XE's machines directives and their corresponding memory sizes.
- Directives.h - header file for the directives source file
- Opcode.cpp - this file contains a dictionary of the SIC/XE's machines opcodes as binary strings, corresponding memory sizes, and format lengths. We duplicated some entries to include format 4 instructions as they will all have a '+' at the beginning of the opcode mnemonic.This file also contains getter functions for the dictionary.
- opcode.h - header file for opcode source file.
- Print.cpp - file responsible for printing out a formatted SIC/XE's symbol and library file. This function is called upon by asxe.cpp
- Print.h - header file for the print source file
- Makefile - makefile for the entire two pass assembler program
- sampleXE.txt - sample SIC/XE source file to test assembler functionality
- START.txt - sample SIC/XE source file to test assembler functionality

Compile Instructions:

This project was compiled using edoras and the Makefile. I wrote the makefile in different ways and one issue is that it doesn't always make and it believes that the executable is already up to date.

Operating Instructions:

To compile without the Makefile you can enter 'g++ -std=c++11 asxe.cpp -o asxe opcode.cpp directives.cpp print.cpp' followed by ./asxe sampleXE.txt OR ./asxe START.txt to execute the SIC/XE source file(s).

This program can also be executed without any input arguments at all with simply just ./asxe entered and will output a message stating that there was insufficient input given and that the file will terminate. However, it was typically always tested with one input file given. It loops through each argument found in argv following ./asxe.

Design Decisions/Lessons Learned:

We decided to use regex for matching patterns to literals in the source code because it was fast and easy to implement. But, we learned too late that the g++ version on edoras is too outdated for regex and other functions like bitset conversions that we had already implemented and were fundamental to our already complete program. We forgot about system limitations and unfortunately didn't realize it until it was too late so we had to overhaul a lot of literal parsing which is the biggest issue with our program. We also learned about checking up on each other's progress which we didn't do all that much at the beginning of the project in mid-March.

Deficiencies/Bugs:

Sometimes, the object code in the listing file has lowercase hexadecimal digits. Literal functionality from SIC/XE source code is not all there. Makefile is inconsistent.