CS530, Fall 2023, Project #2

30 Sep 2023

You shall develop, test, and deliver a two pass assembler for the XE variant of the SIC/XE family of machines.

XE ASSEMBLER REQUIREMENTS:
The XE assembler program shall open SIC/XE assembler source files (such as fig 2.9 and 2.15 in your text) and generate listing files for the XE machine (such as fig 2.10 and 2.16 in your text). The assembler shall implement all XE features and assembler directives up to and including section 2.3.4 of the text (does not implement Control Sections, EXTDEF, and EXTREF).

Input - The user will start the program (the program shall be named "axe") and will provide the XE assembler source file(s) names as arguments on the command-line, each separated by spaces (i.e.):

```
      "[cssc0000@edoras ~a2/axe first.sic second.sic third.sic"
```

Note, if no arguments are provided on the command line, your program shall print a friendly message stating why you are stopping and then terminate the program.

Your axe program will open each file, scan the file and processing each assembler directive and assembling each instruction using the two pass assembler logic discussed in the text and in class.

The specification for the format of the input file is the SIX/XE assembler source file as specified in the text and in *SIC/XE Assembler Source* in our course on Canvas.

Output – SIC/XE Listing file(s) such as those found in figure 2.10 and 2.16 of the text with the same name as the source file, with ".l" as the filename extension (ex: "first.l second.l third.l"). Also, print the SYMTAB you created while assembling the instructions into a separate file with the same filename, with ".st" as the filename extension (ex: "first.st second.st third.st").

The specification for the format of the Listing file is similar to the Source file but amended as seen in fig 2.10 and 2.16 in your text and can be found in *SIC/XE Assembler Listing* in our course on Canvas.

TEAMS:
You shall work in teams of two people on this project. You may choose to use pair programming, dividing up the work, or other methods for work completion; this is up to you although I strongly encourage you to attempt pair programming!

ADDITIONAL REQUIREMENTS:
README file - you shall create a README file; consult the instructions for README file content as posted on Canvas. Also, your source files SHALL CONTAIN sufficient comments for making the source easy to read. Points will be taken off for poorly (or non) commented source or inadequate README file documentation.

Compiler and make (and Makefile) – You shall use C/C++ (gcc/g++) and use make to compile your program for this assignment; you will need to create a Makefile for your project. Consult the example Makefile(s) in this course Canvas Resources module. Name the executable 'asl' .

Test files – You shall prepare and include test files used during the development and test of your project.

Software Design Document - You are required to perform software design of this system. Include a Software Design Document (SDD) and turn it in with your project. Note, you will not be held to formal design specification/formatting or use any of the formal methods. Turn in a file which contains your software design. You may include a kanban (and stories), models, drawings, descriptions, diagrams or similar supporting items to describe your design. You will also include a listing of the tools you used for your system/software design (IDEs, repositories, compilers, debuggers, etc) along with any libraries. This is a significant part of your grade and needs to be adequately captured in your documentation. Include a description of how your team was organized and how effectively you worked together and areas for improvement.

Make sure that all files (README, source files, header files, Makefile) contains each team member's names and RedIDs!

TURNING IN YOUR WORK:
A draft copy of your SDD is due as posted on Canvas and in the syllabus
The full, final project is due as posted on Canvas and in the syllabus

Your project shall include C/C++ source files, an include/header file, a Makefile, and a README file. ONLY ONE MEMBER OF YOUR TEAM TURNS IN THE PROJECT. To turn in your project, each team shall select one person, all files shall be in that person's class account on edoras in a directory named "a2" (~/a2). Leave any test files in this directory as well. BE SURE BOTH TEAM MEMBERS NAMES AND CLASS ACCOUNTS (USERNAMES) are in the README FILE. Finally, the designated person turns in the project by uploading a tarball/zip-file with all project files to Blackboard and entering anycomments in the assignment's turnin.