Introduction to Systems Software – COP3404

Semester Project – Fall 2017

DUE: November 30, 2017

Team:

Emily Czarnecki, Brennan Ledbetter, Christopher Kile

**OBJECTIVE:**

Build a two pass assembler that reads in SIC/XE source code and generates .lst and .obj files.

**NOT INCLUDED:**

The program cannot process literals, floating-point instructions, indirect instructions i.e. the use of ‘@’ within any operand, the EQU, USE, CSECT directives, the HIO, LPS, SKK, STI, STSW, SVC, SIO, TIO instructions, macros, and program blocks.

**TO NOTE:**

The program can handle comments as long as they start with a ‘.’ as well as blank lines. The source code provide **does not** need to be case sensitive. For example, the opcode ‘CLEAR’ can be written in all caps, lowercase, or a combination of both.

**ERRORS:**

**Errors will occur if:**

1. No file is provided as an argument.
2. The file cannot be found or read.
3. The use of any directives or instructions not supported by the assembler, or anything noted in the NOT INCLUDED section.
4. Duplicate symbols
5. Mistakes in the source code, such as misspelling of instructions or symbol names.

**HOW IT WORKS:**

The main class “Assembler” contains the key methods “passOne” and “passTwo”. Each of which are called through the main method. The main method obtains the source file through the command line argument which is passes to passOne. The main method also establishes the hash tables “SYMTAB” and “OPTAB” to be used in both passes. passOne of the assembler, reads in each line of the source file, processes it and writes each line to the intermediate file – the .lst file – along with the address of each instruction, stored in the variable “LOCCTR”. It uses the OPTAB to confirm the OPCODE. Upon encountering a symbol, if not a duplicate, the symbol gets stored in SYMTAB, along with the current value of LOCCTR. The building of the SYMTAB is key in generating the object code in passTwo. passOne completes at either the end of the file or if the directive “END” is encountered. The next primary method is passTwo. Reading from the .lst file, passTwo processes each line to generate the appropriate object code. As each line is processes, passTwo determines the format of the instruction, if it needs to use PC relative or base relative, determines the OPCODE from the OPTAB, and sets the n, i, x, b, p, e bits accordingly. If a symbol is encountered, SYMTAB is search and the address location is returned to be used to build the object code. All this information is passed to helper functions called by passTwo to build the object code. After the object code is built it is written to the .obj file. The completed object file contains the header record, text records, modification records, and end record.

**HELPER FUNCTIONS/CLASSES:**

While passOne and passTwo are the primary methods of the program, a number of helper functions and classes were implemented to completed the program. Classes were created to assist in parsing the source code and the lines from the intermediate file. An object called “line” was created to correctly parse the pieces into appropriate variables for use in each pass. The input is split into a string array, ignoring all blank spaces, and each part of the array is stored into the correct variables. These variables are used to build the object line. The same approach is used to create an intermediate line which is written to the .lst file. The class “HexUtil” contains a number of functions that help with processing strings and integers and converting them into hexadecimal or back, adding and subtracting hexadecimal, and formatting any hexadecimal numbers, such LOCCTR to the desired length. Also in this class are the methods that build the object code to be written in the .obj file. Each instruction format has its own method taking in the necessary information to build the correct machine code. The last two classes build the OPTAB and SYMTAB hash tables.