SIC-XE-Assembler Design

Garv Sethi
22115057

April 2024

1 Introduction

The Logic of the assembler is decomposed into 3 main functions which we will discuss in the further sections , these functions are MAIN(), PASS1(), PASS2(), The Steps for running the program are as follows:

1. Download and extract the zip file containing the codes
2. Run the command :
   ```
   # g++ -std=c++20 pass2.cpp -o assembler.out
   ```
3. execute the file assembler.out
4. Enter the name of the file 'input_sample.txt' provided along with the code
5. The listing file and object file have been generated

2 Main

The int main() function present in the pass2.cpp file does the following:

1. **Input Handling:** The user is prompted to enter the name of the input file.
2. **Initialization:** Opcode Table (OPTAB) is loaded.
3. **First Pass (PASS1):** Generates an intermediate representation of the source program, identifies symbols and literals, assigns addresses, and detects errors. Writes intermediate and error files.
4. **Writing Tables:** Writes Symbol, Literal, EXTREF, and EXTDEF tables to a file.

5. **Second Pass (PASS2):** Generates actual object code, resolves addresses, and produces the final object and listing files.

6. **Output Notification:** Informs the user about the generated files.

3 Pass1

1. **File Handling**: Opens input, intermediate, and error files. If any file opening fails, it prints an error message and exits the program.

2. **Initializations**: Initializes variables and data structures used in the function.

3. **Reading Source File**: Reads the source file line by line, skipping comment lines.

4. **Parsing**: Parses each line to extract label, opcode, operand, and comment.

5. **Processing Instructions**: Handles various types of instructions including START, END, CSECT, OPCODEs, BYTE, WORD, RESW, RESB, EXTDEF, EXTREF, EQU, ORG, USE, and LTORG.

6. **Updating Location Counter (LOCCTR)**: Adjusts the LOCCTR according to the instruction format and operand.

7. **Handling Symbol Table (SYMTAB)**: Manages the Symbol Table by adding new symbols, updating existing ones, and checking for duplicates.

8. **Handling Literal Table (LITTAB)**: Manages the Literal Table by adding new literals and updating their addresses.

9. **Handling Control Sections (CSECT_TAB)**: Tracks Control Sections, their lengths, and symbol addresses within each section.

10. **Handling Block Changes**: Updates the current block and its location counter when encountering USE directives.

11. **Handling Equates (EQU)**: Resolves expressions for equated symbols and updates their values in the SYMTAB.

12. **Handling Errors**: Detects and reports errors such as duplicate symbols, invalid opcodes, and undefined symbols.

13. **Updating Intermediate File**: Writes the parsed information along with updated LOCCTR to the intermediate file.

14. **Handling END Directive**: Finalizes processing and computes the program length.

4 Pass 2

1. **File Handling**: Opens the intermediate, object, listing, and error files for writing. If any file opening fails, it prints an error message and exits the program.
2. **Initialization**: Initializes various variables and data structures used in the function.
3. **Reading Intermediate File**: Reads the intermediate file line by line, discarding the heading line and handling any comment lines.
4. **Handling START Directive**: If the first opcode encountered is START, it sets the start address and writes the corresponding line to the listing file.
5. **Writing Header Record (H record)**: Writes the header record containing the program name, start address, and program length to the object file.
6. **Processing Instructions**: Iterates through the intermediate file, processing each instruction or directive.
    - For machine instructions (found in OPTAB), it generates the corresponding object code.
    - For assembler directives (e.g., BYTE, WORD), it translates the operand to object code.
    - For control directives (e.g., BASE, NOBASE), it handles base addressing.
7. **Writing Text Records (T records)**: Writes the generated object code to text records in the object file. It handles the maximum record length and splits records if necessary.
8. **Writing End Record (E record)**: Writes the end record to the object file.
9. **Handling Control Section (CSECT)**: If the opcode encountered is CSECT, it writes the end record, processes any remaining lines, and writes the header record for the next section.
10. **Handling End Directive (END)**: Finalizes processing and writes the end record to the object file.
11. **Handling Comments**: Writes any comment lines encountered to the listing file.
12. **Handling Errors**: Detects and reports errors such as invalid register names and missing symbols.

5 Output for code from Lelan L Beck

```
<3>WSL (10) ERROR: UtilTranslatePathList:2866: Failed to translate D:\VMware\bin\
cruelkratos@Ultron:/mnt/c/Users/mailg/OneDrive/Desktop/hi/fonal/SIC-XE-Assembler$ cat input_sample.txt
SUM     START   0
FIRST   LDX     #0
        LDA     #0
        +LDB    #TABLE2
        BASE    TABLE2
LOOP    ADD     TABLE, X
        ADD     TABLE2, X
        TIX     COUNT
        JLT     LOOP
        +STA    TOTAL
        RSUB
COUNT   RESW    1
TABLE   RESW    2000
TABLE2  RESW    2000
TOTAL   RESW    1
        END     FIRSTcruelkratos@Ultron:/mnt/c/Users/mailg/OneDrive/Desktop/hi/fonal/SIC-XE-Assembler$
```

```
cruelkratos@Ultron: /mnt/c/L
cruelkratos@Ultron:/mnt/c/Users/mailg/OneDrive/Desktop/hi/fonal/SIC-XE-Assembler$ g++ -std=c++20 pass2.cpp -o assembler.out
pass2.cpp: In function 'std::string createObjectCodeFormat34()':
pass2.cpp:499:1: warning: control reaches end of non-void function [-Wreturn-type]
  499 | }
      | ^
cruelkratos@Ultron:/mnt/c/Users/mailg/OneDrive/Desktop/hi/fonal/SIC-XE-Assembler$ ./assembler.out
****Input file and executable(assembler.out) should be in same folder****

Enter name of input file:input_sample.txt

Loading OPTAB

Performing PASS1
Writing intermediate file to 'intermediate_input_sample.txt'
Writing error file to 'error_input_sample.txt'
Writing SYMBOL TABLE
Writing LITERAL TABLE
Writing EXTREF TABLE
Writing EXTDEF TABLE

Performing PASS2
Writing object file to 'object_input_sample.txt'
Writing listing file to 'listing_input_sample.txt'
cruelkratos@Ultron:/mnt/c/Users/mailg/OneDrive/Desktop/hi/fonal/SIC-XE-Assembler$
```

```
cruelkratos@Ultron: /mnt/c/L
cruelkratos@Ultron:/mnt/c/Users/mailg/OneDrive/Desktop/hi/fonal/SIC-XE-Assembler$ cat object_input_sample.txt
H^SUM   ^000000^002F03
T^000000^1D^05000001000069101790 1BA0131BC0002F200A3B2FF40F102F004F0000
M^000007^05
M^000017^05
E^000000

cruelkratos@Ultron:/mnt/c/Users/mailg/OneDrive/Desktop/hi/fonal/SIC-XE-Assembler$ cat listing_input_sample.txt
Line    Address Label   OPCODE  OPERAND ObjectCode      Comment
5       00000   0       SUM     START   0
10      00000   0       FIRST   LDX     #0      050000
15      00003   0               LDA     #0      010000
20      00006   0               +LDB    #TABLE2 69101790
25      0000A   0               BASE    TABLE2
30      0000A   0       LOOP    ADD     TABLE,X 1BA013
35      0000D   0               ADD     TABLE2,X        1BC000
40      00010   0               TIX     COUNT   2F200A
45      00013   0               JLT     LOOP    3B2FF4
50      00016   0               +STA    TOTAL   0F102F00
55      0001A   0               RSUB            4F0000
60      0001D   0       COUNT   RESW    1
65      00020   0       TABLE   RESW    2000
70      01790   0       TABLE2  RESW    2000
75      02F00   0       TOTAL   RESW    1
80      02F03           END     FIRST
cruelkratos@Ultron:/mnt/c/Users/mailg/OneDrive/Desktop/hi/fonal/SIC-XE-Assembler$
```