Indian Institute of Technology, Roorkee

CSN-252

System Software

SIC/XE Assembler

Roopam Taneja
22125030
CSE O3 Batch

SIC/XE Assembler

Introduction

The objective of this project is to implement a version of two-pass SIC/XE assembler in C++. It supports Program Blocks.

Features implemented:

- Literals
- Expressions
- Assembler Directives
- Symbols
- Program Blocks

Design and Working

It is a two-pass assembler which assembles the input program in two passes.

1) Pass 1:
    - The intermediate file is created and updated and the error file is also updated if the need arises. The required symbols are declared in the symbol table.
    - The actual processing of the input starts when the START symbol is encountered whereas any comments are ignored.
    - The LOCCTR is set as per the value given in the START directive, otherwise it is by default initialised to zero. Two nested loops are at the heart of the program and keep executing till the directive END is encountered.
    - Errors such as duplicate symbols are mentioned in the error file.
2) Pass 2:
    - The second pass acts on the intermediate file generated by pass 1, the file is processed via the readIntermediateFile() function.
    - We then proceed to generate the listing file and the object program. The error file is updated as and when necessary.
    - The symbol table is used to resolve any issues created when symbols are used as operands.
    - The various assembler directives are also taken into account while creating the object file.

The following source code files are present in `src` folder:

- `src/helper_functions.cpp` contains frequently used utlity functions.
- `src/table_structs.cpp` contains various tables and structs used throughout the code.:
    - SYMTAB: The struct contains information of labels like name, address, block number, a character representing whether the label exists in the symbol table or not, an integer representing whether label is relative or not.
    - OPTAB: The struct contains information of opcode like name, format, a character representing whether the opcode is valid or not.
    - LITTAB: The struct contains information of literals like its value, address, block number, a character representing whether the literal exists in the literal table or not.
    - REGTAB: The struct contains information of registers like its numeric equivalent, a character representing whether the register exists or not.

  - BLOCKS: The struct contains information of blocks like its name, start address, block number, location counter value for end address of block, a character representing whether the block exists or not.

- `src/pass1.cpp` generates Symbol Table, Intermediate File for the pass2 and also generates Error File.
- `src/pass2.cpp` works on the Intermediate File generated by pass1 and generates Listing File and Object Program File

Following files are generated by the assembler :

1) Object Program
2) Listing File
3) Error File
4) Intermediate File
5) A file showing various tables created by the Assembler

Installation

Extract the contents of zip file.

Now in terminal:

```
g++ .\src\pass2.cpp -o .\testing\assembler
```

Now copy the assembly code you wish to assemble in `assembly_code.txt` file of `testing` folder. Some sample assembly codes have been provided in `sample_programs` folder.

```
cd testing
.\assembler.exe
```

All output and intermediate files are generated in the same folder.

Sample Results

1) `sample_code.txt` :

   This is question 3 of section 2.2 in the prescribed textbook. As mentioned, the assembler will be tested on this.

```
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   FIRST
```

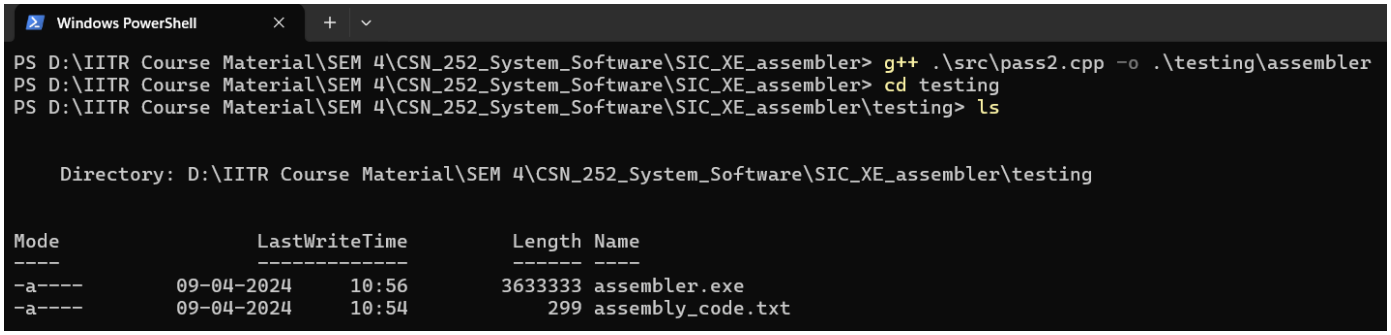

Figure 1: Directory Structure Before

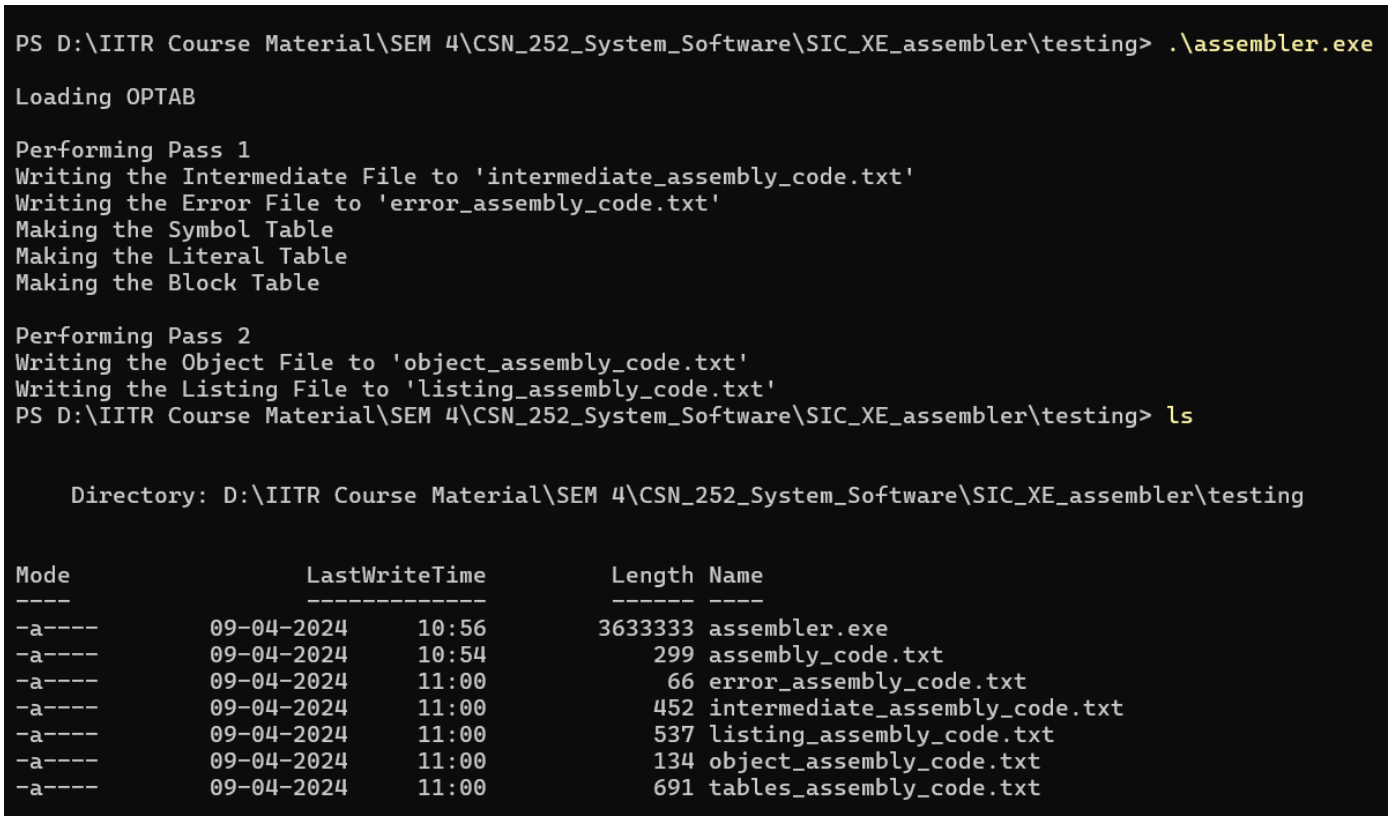

Figure 2: Directory Structure After

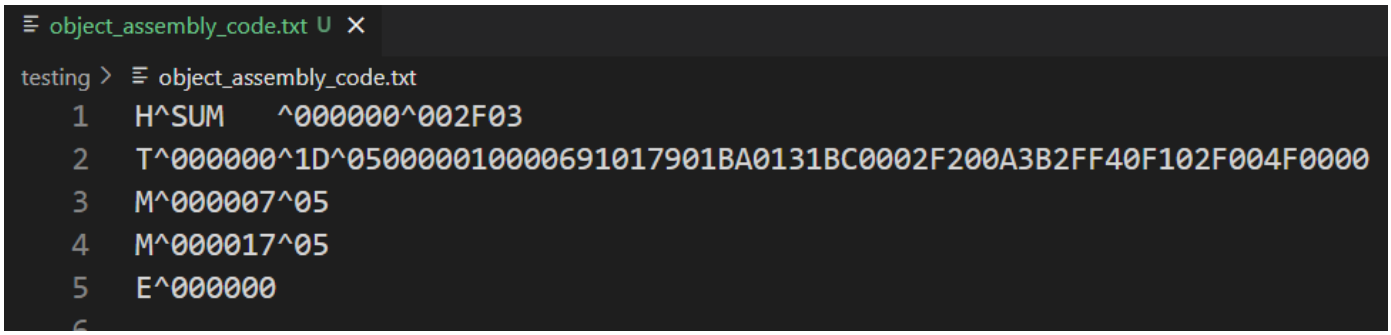

Figure 3: Object File

listing_assembly_code.txt U

testing > listing_assembly_code.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
```

Figure 4: Listing File

intermediate_assembly_code.txt U

testing > intermediate_assembly_code.txt

```
Line    Address Label   OPCODE  OPERAND Comment
5   00000   0   SUM START   0
10  00000   0   FIRST   LDX #0
15  00003   0       LDA #0
20  00006   0       +LDB    #TABLE2
25  0000A   0       BASE    TABLE2
30  0000A   0   LOOP    ADD TABLE,X
35  0000D   0       ADD TABLE2,X
40  00010   0       TIX COUNT
45  00013   0       JLT LOOP
50  00016   0       +STA    TOTAL
55  0001A   0       RSUB
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
```

Figure 5: Intermediate File

2) `program_blocks_code.txt` :

This provides a sample code given in the textbook (Fig 2.11) to demonstrate the usage of program blocks.

```
TEST     START   0
FIRST    STL     RETADR
CLOOP    JSUB    RDREC
         LDA     LENGTH
         COMP    #0
         JEQ     ENDFIL
         JSUB    WRREC
         J       CLOOP
ENDFIL   LDA     =C'EOF'
         STA     BUFFER
         LDA     #3
         STA     LENGTH
         JSUB    WRREC
         J       @RETADR
         USE     CDATA
RETADR   RESW    1
LENGTH   RESW    1
         USE     CBLKS
BUFFER   RESB    4096
BUFEND   EQU     *
MAXLEN   EQU     BUFEND-BUFFER
.
.        SUBROUTINE TO READ RECORD INTO BUFFER
.
         USE
RDREC    CLEAR   X
         CLEAR   A
         CLEAR   S
         +LDT    #MAXLEN
RLOOP    TD      INPUT
         JEQ     RLOOP
         RD      INPUT
         COMPR   A, S
         JEQ     EXIT
         STCH    BUFFER,X
         TIXR    T
         JLT     RLOOP
EXIT     STX     LENGTH
         RSUB
         USE     CDATA
INPUT    BYTE    X'F1'
.
.        SUBROUTINE TO WRITE RECORD FROM BUFFER
.
         USE
WRREC    CLEAR   X
         LDT     LENGTH
WLOOP    TD      =X'05'
         JEQ     WLOOP
         LDCH    BUFFER,X
         WD      =X'05'
         TIXR    T
```

```
JLT     WLOOP
RSUB
USE     CDATA
LTORG
END     FIRST
```

object_assembly_code.txt U

testing > object_assembly_code.txt

```
1  H^TEST   ^000000^001071
2  T^000000^1E^1720634B20210320602900003320064B203B3F2FEE0320550F2056010003
3  T^00001E^09^0F20484B20293E203F
4  T^000027^1D^B410B400B44075101000E32038332FFADB2032A00433200857A02FB850
5  T^000044^09^3B2FEA13201F4F0000
6  T^00006C^01^F1
7  T^00004D^19^B410772017E3201B332FFA53A016DF2012B8503B2FEF4F0000
8  T^00006D^04^454F4605
9  E^000000
10
11
```

Figure 6: Object File

```
Line    Address Label   OPCODE  OPERAND ObjectCode      Comment
5       00000   0       TEST    START   0
10      00000   0       FIRST   STL     RETADR  172063
15      00003   0       CLOOP   JSUB    RDREC   4B2021
20      00006   0               LDA     LENGTH  032060
25      00009   0               COMP    #0      290000
30      0000C   0               JEQ     ENDFIL  332006
35      0000F   0               JSUB    WRREC   4B203B
40      00012   0               J       CLOOP   3F2FEE
45      00015   0       ENDFIL  LDA     =C'EOF' 032055
50      00018   0               STA     BUFFER  0F2056
55      0001B   0               LDA     #3      010003
60      0001E   0               STA     LENGTH  0F2048
65      00021   0               JSUB    WRREC   4B2029
70      00024   0               J       @RETADR 3E203F
75      00000   1               USE     CDATA
80      00000   1       RETADR  RESW    1
85      00003   1       LENGTH  RESW    1
90      00000   2               USE     CBLKS
95      00000   2       BUFFER  RESB    4096
100     01000   2       BUFEND  EQU     *
105     01000           MAXLEN  EQU     BUFEND-BUFFER
110     .
115     .       SUBROUTINE TO READ RECORD INTO BUFFER
120     .
125     00027   0               USE     DEFAULT
130     00027   0       RDREC   CLEAR   X       B410
135     00029   0               CLEAR   A       B400
140     0002B   0               CLEAR   S       B440
145     0002D   0               +LDT    #MAXLEN 75101000
150     00031   0       RLOOP   TD      INPUT   E32038
155     00034   0               JEQ     RLOOP   332FFA
160     00037   0               RD      INPUT   DB2032
165     0003A   0               COMPR   A,S     A004
170     0003C   0               JEQ     EXIT    332008
175     0003F   0               STCH    BUFFER,X        57A02F
180     00042   0               TIXR    T       B850
185     00044   0               JLT     RLOOP   3B2FEA
190     00047   0       EXIT    STX     LENGTH  13201F
195     0004A   0               RSUB            4F0000
200     00006   1               USE     CDATA
205     00006   1       INPUT   BYTE    X'F1'   F1
210     .
215     .       SUBROUTINE TO WRITE RECORD FROM BUFFER
220     .
225     0004D   0               USE     DEFAULT
230     0004D   0       WRREC   CLEAR   X       B410
235     0004F   0               LDT     LENGTH  772017
240     00052   0       WLOOP   TD      =X'05'  E3201B
245     00055   0               JEQ     WLOOP   332FFA
250     00058   0               LDCH    BUFFER,X        53A016
255     0005B   0               WD      =X'05'  DF2012
260     0005E   0               TIXR    T       B850
265     00060   0               JLT     WLOOP   3B2FEF
270     00063   0               RSUB            4F0000
275     00007   1               USE     CDATA
280     00007   1               LTORG
285     00007   1       *       =C'EOF'         454F46
290     0000A   1       *       =X'05'          05
295     00066                   END     FIRST
```

Figure 7: Listing File

```
--------------------------------SYMBOL TABLE--------------------------------

:-      name:undefined  |address:0       |relative:00000
0:-     name:    |address:0       |relative:00000
BUFEND:-        name:BUFEND     |address:001000 |relative:00001
BUFFER:-        name:BUFFER     |address:00000  |relative:00001
CLOOP:- name:CLOOP      |address:00003  |relative:00001
ENDFIL:-        name:ENDFIL     |address:00015  |relative:00001
EXIT:-  name:EXIT       |address:00047  |relative:00001
FIRST:- name:FIRST      |address:00000  |relative:00001
INPUT:- name:INPUT      |address:00006  |relative:00001
LENGTH:-        name:LENGTH     |address:00003  |relative:00001
MAXLEN:-        name:MAXLEN     |address:01000  |relative:00000
RDREC:- name:RDREC      |address:00027  |relative:00001
RETADR:-        name:RETADR     |address:00000  |relative:00001
RLOOP:- name:RLOOP      |address:00031  |relative:00001
WLOOP:- name:WLOOP      |address:00052  |relative:00001
WRREC:- name:WRREC      |address:0004D  |relative:00001

--------------------------------LITERAL TABLE--------------------------------

C'EOF':-        value:C'EOF'    |address:00007
X'05':- value:X'05'     |address:0000A

--------------------------------BLOCK TABLE--------------------------------

CBLKS:- value:CBLKS     |address:00071
CDATA:- value:CDATA     |address:00066
DEFAULT:-       value:DEFAULT   |address:00000
```

Figure 8: Various Tables

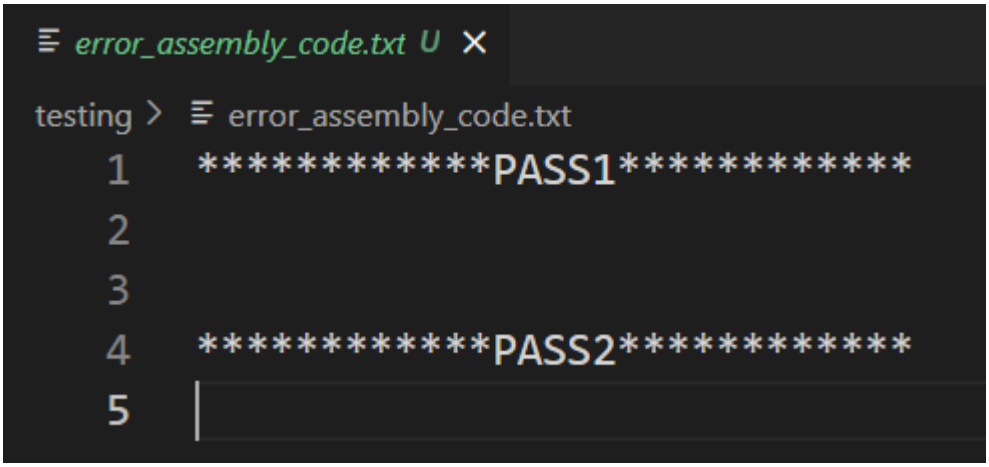

Figure 9: Error File

3) `csect_code.txt` :

This provides a sample code given in the textbook (Fig 2.15) to demonstrate the usage of control sections.

(Tested for showing that it correctly detects errors like wrong opcodes for unsupported features)

```
COPY     START   0
         EXTDEF  BUFFER, BUFEND, LENGTH
         EXTREF  RDREC, WRREC
FIRST    STL     RETADR
CLOOP    +JSUB   RDREC
         LDA     LENGTH
         COMP    #0
         JEQ     ENDFIL
         +JSUB    WRREC
         J       CLOOP
ENDFIL   LDA     =C'EOF'
         STA     BUFFER
         LDA     #3
         STA     LENGTH
         +JSUB   WRREC
         J       @RETADR
RETADR   RESW    1
LENGTH   RESW    1
         LTORG
BUFFER   RESB    4096
BUFEND   EQU     *
MAXLEN   EQU     BUFEND-BUFFER

RDREC    CSECT
.
.       SUBROUTINE TO READ RECORD INTO BUFFER
.
         EXTREF  BUFFER, BUFEND, LENGTH
         CLEAR   X
         CLEAR   A
         CLEAR   S
         LDT     MAXLEN
RLOOP    TD      INPUT
         JEQ     RLOOP
         RD      INPUT
         COMPR   A, S
         JEQ     EXIT
         +STCH   BUFFER,X
         TIXR    T
         JLT     RLOOP
EXIT     +STX    LENGTH
         RSUB
INPUT    BYTE    X'F1'
MAXLEN   WORD    BUFEND-BUFFER

WRREC    CSECT
.
.       SUBROUTINE TO WRITE RECORD FROM BUFFER
.
         EXTREF  LENGTH, BUFFER
         CLEAR   X
```

```
        +LDT    LENGTH
WLOOP   TD      =X'05'
        JEQ     WLOOP
        +LDCH   BUFFER,X
        WD      =X'05'
        TIXR    T
        JLT     WLOOP
        RSUB
        END     FIRST
```

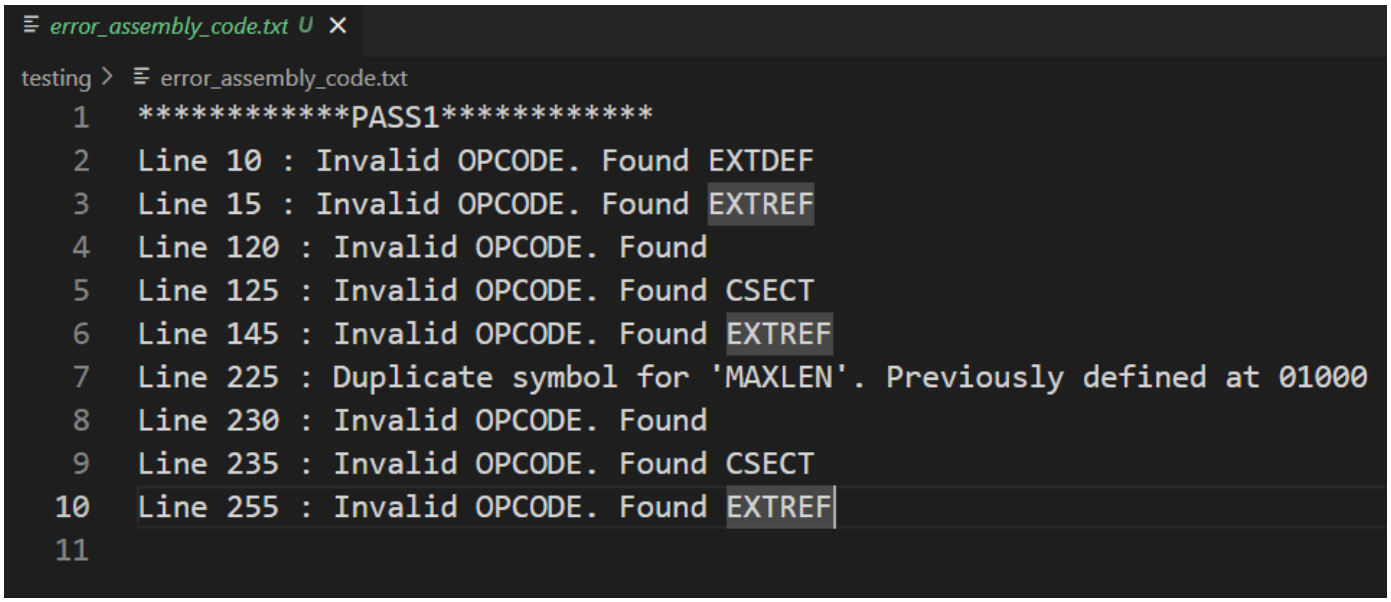

Figure 10: Error File