CSN-252
SYSTEM SOFTWARE

Tutorial - 08
Design of SIC-XE Assembler

Vraj Tamakuwala
22114098 - O4

Table of Contents

1 Introduction

The SIC-XE assembler is a two-pass system I developed using C++ programming language. It takes a text file containing the SIC-XE instruction set as input and produces an object code file, tables, an intermediate file, and an error file (if any).
During the first pass, the assembler generates a symbol table and an intermediate file, laying the groundwork for the second pass. The second pass produces a listing file that includes the input assembly code and details such as addresses, block numbers, and object codes for each instruction. Additionally, it generates an object program and an error file (if any), highlighting any errors in the input assembly program.

This assembler supports the following SIC-XE instructions:

- Literals
- Expressions
- Symbol-defining statements
- Program Blocks

2 Architecture and Working

The assembler is divided into various files according to their functionalities.

- **tables.hpp :** Contains all the necessary structures for making symbol table-SYMTAB, literal table-LITTAB, block table-BLOCKTAB, and the Opcode and Register mappings.
- **utility.hpp :** Contains all the basic functions that hardly change, like checking precedence, conversion of Hex string to Int and vice-versa, validating input sequence, checking for comment lines, etc.
- **pass1.hpp :** Covers the actual logic for pass1 of the assembler. Reads the input file and generates a valid intermediate file, namely *intermediate_file.txt*, which comes very handy for pass2. Various functionalities include evaluating expressions, handling literals, and generating symbol and block tables.
- **pass2.hpp :** It reads the intermediate file produced by pass1, evaluates the object code corresponding to each instruction, and generates the listing file *assembly_listing.txt* along with errors(if any) in it. It also writes the object code into a file named *obj_program.txt*.
- **compile.cpp :** It is the file containing the *main* function. The input file needs to be present in the same folder. It uses all the above files to generate the corresponding output files.

3 Steps to compile

1. Extract all the above-mentioned files. Place your input file in *.txt* format at the same location.

```
PS C:\Users\vrajt\OneDrive\Desktop\SIC-XE> ls


    Directory: C:\Users\vrajt\OneDrive\Desktop\SIC-XE


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a---l        11-04-2024     18:57            638 compile.cpp
-a---l        11-04-2024     18:56            437 inputPB_code.txt
-a---l        11-04-2024     16:56          15259 pass1.hpp
-a---l        11-04-2024     16:57          17114 pass2.hpp
-a---l        11-04-2024     16:58           7382 tables.hpp
-a---l        11-04-2024     16:46           3548 utility.hpp


PS C:\Users\vrajt\OneDrive\Desktop\SIC-XE>
```

2. Compile the *compile.cpp* file using command *g++ compile.cpp -o compile*.
   After successful compilation, type in *./compile* to run your file.
   Enter the name of your input file (here, *inputPB_code.txt*) and see the magic!

```
PS C:\Users\vrajt\OneDrive\Desktop\SIC-XE> g++ compile.cpp -o compile
PS C:\Users\vrajt\OneDrive\Desktop\SIC-XE> ./compile
Enter the input file name: inputPB_code.txt
PS C:\Users\vrajt\OneDrive\Desktop\SIC-XE>
```

3. This will generate three files, namely:
   - intermediate_file.txt: File generated after pass1.
   - assembly_listing.txt: Listing file generated after pass2. Also contains the error (if any).
   - obj_program.txt: Contains the object program of given SIC-XE code.

4 Sample Program

```
1 copy    start   0
2 first   stl     retadr
3 cloop   jsub    rdrec
```

```
        lda     length
        comp    #0
        jeq     endfil
        j       cloop
endfil  J       @retadr
        use     cdata
retadr  resw    1
length  resw    1
        use     cblks
buffer  resb    4096
bufend  equ     *
maxlen  equ     4096
        use
rdrec   clear   x
        clear   a
        clear   s
        +ldt    #maxlen
loop    td      input
        jeq     loop
 .gkjbhlnl
        rd      input
com     pr      a, s
        jeq     exit

        stch    buffer,x
        tixr    t
        jlt     loop
exit    stx     length
        rsub
        use     cdata
input   byte    x'f3'
        end     first
```

Listing 1: Sample Program 1 - Input

```
00000   0 COPY   START  0
00000   0 FIRST  STL    RETADR
00003   0 CLOOP  JSUB   RDREC
00006   0 -      LDA    LENGTH
00009   0 -      COMP   #0
0000C   0 -      JEQ    ENDFIL
0000F   0 -      J      CLOOP
00012   0 ENDFIL J      @RETADR
-       0 -      USE    CDATA
00000   1 RETADR RESW   1
00003   1 LENGTH RESW   1
-       1 -      USE    CBLKS
00000   2 BUFFER RESB   4096
-       2 BUFEND EQU    *
-       2 MAXLEN EQU    4096
-       2 -      USE    -
00015   0 RDREC  CLEAR  X
```

```
00017   0 -       CLEAR   A
00019   0 -       CLEAR   S
0001B   0 -       +LDT    #MAXLEN
0001F   0 LOOP    TD      INPUT
00022   0 -       JEQ     LOOP
00025   0 -       RD      INPUT
00028   0 -       COMPR   A, S
0002A   0 -       JEQ     EXIT
0002D   0 -       STCH    BUFFER,X
00030   0 -       TIXR    T
00032   0 -       JLT     LOOP
00035   0 EXIT    STX     LENGTH
00038   0 -       RSUB    -
-       0 -       USE     CDATA
00006   1 INPUT   BYTE    X'F3'
00007   1 -       END     FIRST
```

Listing 2: Sample Program 1 - intermediate_file

```
00000    0 COPY     START   0
00000    0 FIRST    STL     RETADR      172038
00003    0 CLOOP    JSUB    RDREC       4B200F
00006    0          LDA     LENGTH      032035
00009    0          COMP    #0          290000
0000C    0          JEQ     ENDFIL      332003
0000F    0          J       CLOOP       3F2FF1
00012    0 ENDFIL   J       @RETADR     3E2026
         0          USE     CDATA
00000    1 RETADR   RESW    1
00003    1 LENGTH   RESW    1
         1          USE     CBLKS
00000    2 BUFFER   RESB    4096
         2 MAXLEN   EQU     4096
         2          USE
00015    0 RDREC    CLEAR   X           B410
00017    0          CLEAR   A           B400
00019    0          CLEAR   S           B440
0001B    0          +LDT    #MAXLEN     75101000
0001F    0 LOOP     TD      INPUT       E3201F
00022    0          JEQ     LOOP        332FFA
00025    0          RD      INPUT       DB2019
00028    0          COMPR   A,S         A004
0002A    0          JEQ     EXIT        332008
0002D    0          STCH    BUFFER,X    57A012
00030    0          TIXR    T           B850
00032    0          JLT     LOOP        3B2FEA
00035    0 EXIT     STX     LENGTH      132006
00038    0          RSUB                4F0000
         0          USE     CDATA
00006    1 INPUT    BYTE    X'F3'       F3
         1          END     FIRST
```

```
/*************************************************/
ERRORS in the SIC/XE Program:

PASS1 Errors:

PASS2 Errors:
```

Listing 3: Sample Program 1 - assembly_listing

```
H^COPY__^000000^001042
T^000000^15^172038^4B200F^032035^290000^332003^3F2FF1^3E2026
T^000015^1D^B410^B400^B440^75101000^E3201F^332FFA^DB2019^A004^332008^57
   A012^B850
T^000032^09^3B2FEA^132006^4F0000
T^000041^01^F3
E^000000
```

Listing 4: Sample Program 1 - obj_program

Some other sample programs to try on...

```
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
```

Listing 5: Sample Program 2 - Input

```
test     start   1000
first    stl     retadr
cloop    jsub    rdrec
         lda     length
         comp    zero
         jeq     endfil
         j       cloop
endfil   ldl     retadr
         rsub
zero     word    0
retadr   resw    1
length   resw    1
```

```
buffer  resb    4096
rdrec   ldx     zero
        lda     zero
loop    td      input
        jeq     loop
        rd      input
        comp    zero
        jeq     exit
        stch    buffer,x
        tix     maxlen
        jlt     loop
exit    stx     length
        rsub
input   byte    x'f3'
maxlen  word    4096
        end     first
```

Listing 6: Sample Program 3 - Input (contains error)