CSN 252 System Software
SIC/XE ASSEMBLER

(CONTROL SECTIONS)

NAME: BALAGA PAVAN SAI

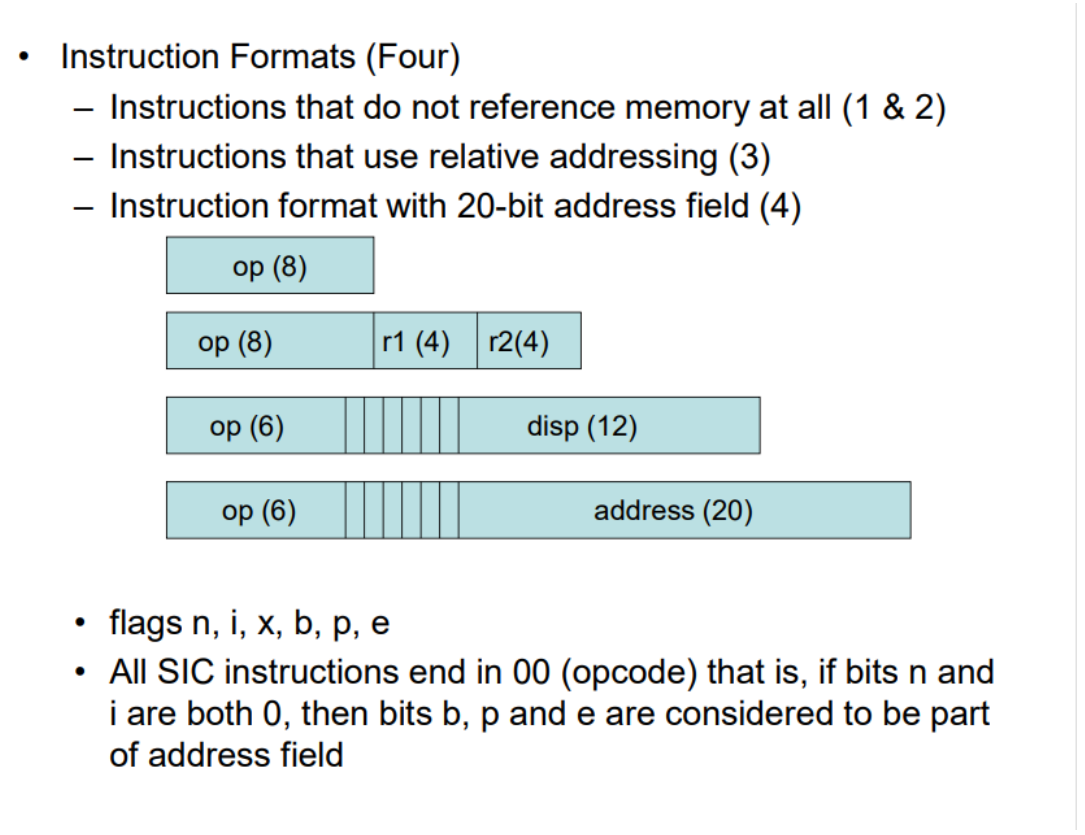
ENROLL: 21114025

Contents

Project Introduction

This project implements an assembler that supports SIC/XE instructions. The assembler includes all the instructions. It supports Control Sections.

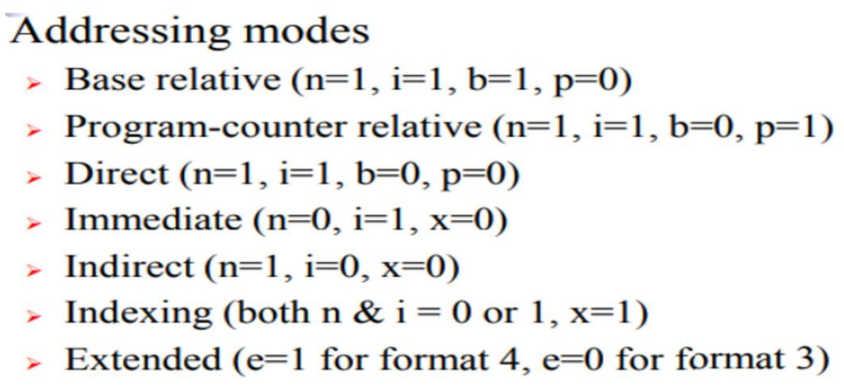
The instruction formats supported by SIC/XE :



- Instruction Formats (Four)
  - Instructions that do not reference memory at all (1 & 2)
  - Instructions that use relative addressing (3)
  - Instruction format with 20-bit address field (4)

- flags n, i, x, b, p, e
- All SIC instructions end in 00 (opcode) that is, if bits n and i are both 0, then bits b, p and e are considered to be part of address field

The Addressing modes supported by the Assembler are:

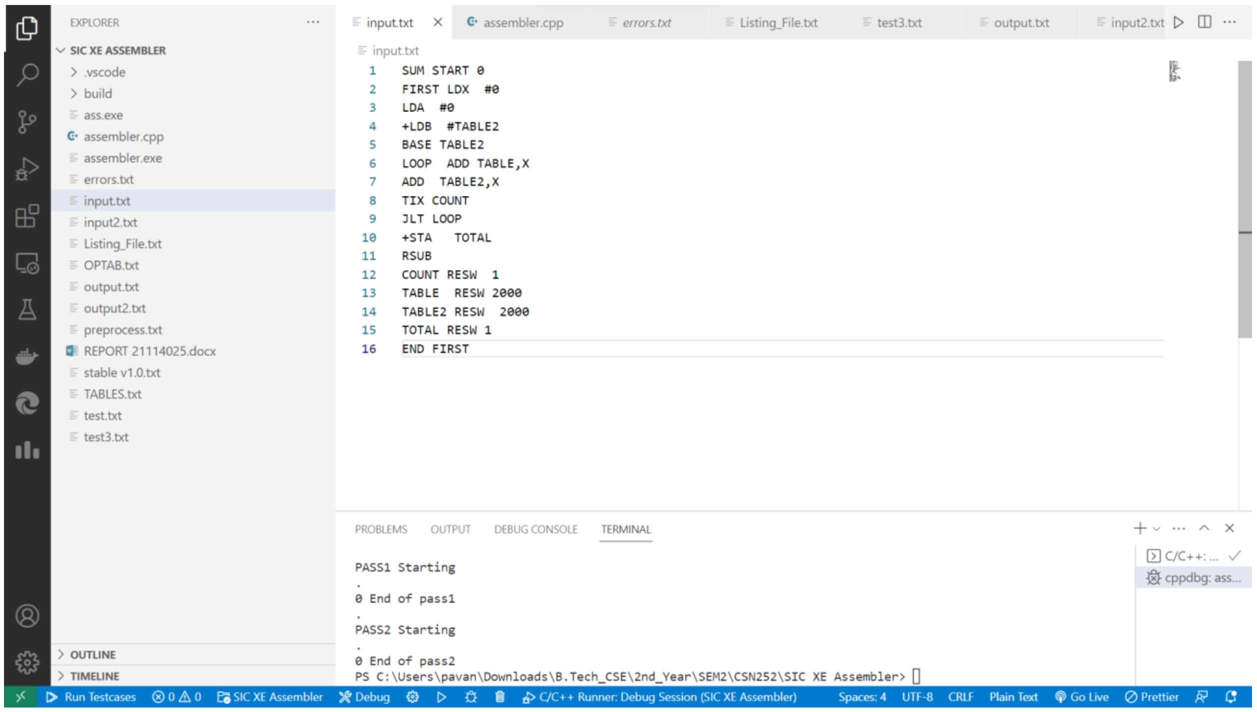


Addressing modes

- Base relative (n=1, i=1, b=1, p=0)
- Program-counter relative (n=1, i=1, b=0, p=1)
- Direct (n=1, i=1, b=0, p=0)
- Immediate (n=0, i=1, x=0)
- Indirect (n=1, i=0, x=0)
- Indexing (both n & i = 0 or 1, x=1)
- Extended (e=1 for format 4, e=0 for format 3)

Steps To Run

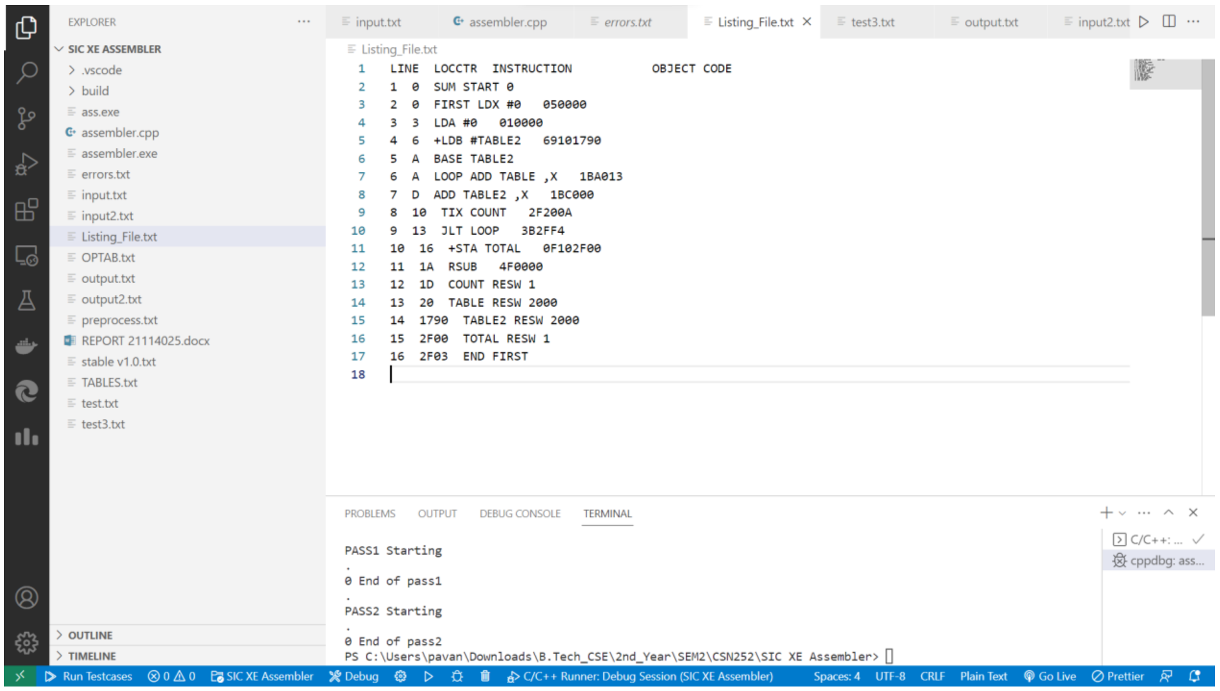
1. Download the “assembler.cpp” attached with this document.
2. Create a folder named “input.txt”. add your input to this folder.



```
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
```

```
PASS1 Starting
.
0 End of pass1
.
PASS2 Starting
.
0 End of pass2
PS C:\Users\pavan\Downloads\B.Tech_CSE\2nd_Year\SEM2\CSN252\SIC XE Assembler>
```

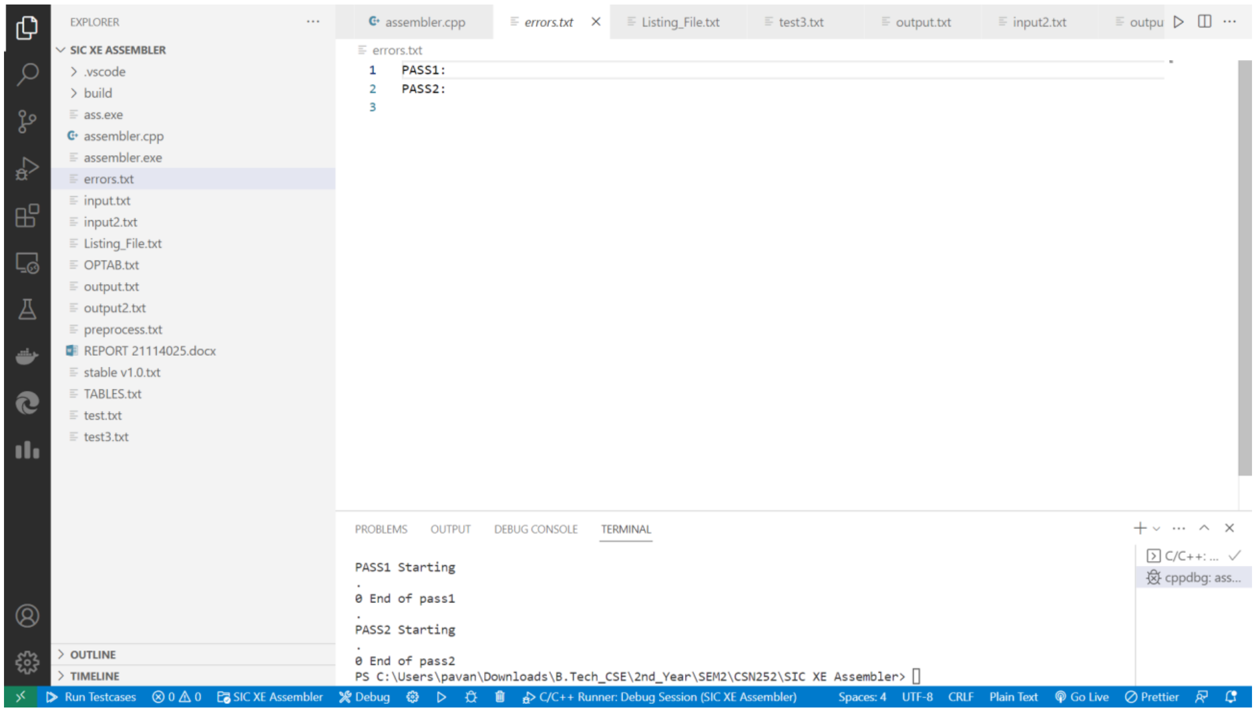
3. Compile assembler.cpp (any c++ compiler) and run the .exe file.
4. The program produces:

a) 

```
LINE  LOCCTR  INSTRUCTION          OBJECT CODE
1  0  SUM START 0
2  0  FIRST LDX #0   050000
3  3  LDA #0   010000
4  6  +LDB #TABLE2   69101790
5  A  BASE TABLE2
6  A  LOOP ADD TABLE ,X   1BA013
7  D  ADD TABLE2 ,X   1BC000
8  10  TIX COUNT   2F200A
9  13  JLT LOOP   3B2FF4
10  16  +STA TOTAL   0F102F00
11  1A  RSUB   4F0000
12  1D  COUNT RESW 1
13  20  TABLE RESW 2000
14  1790  TABLE2 RESW 2000
15  2F00  TOTAL RESW 1
16  2F03  END FIRST
```

```
PASS1 Starting
.
0 End of pass1
.
PASS2 Starting
.
0 End of pass2
PS C:\Users\pavan\Downloads\B.Tech_CSE\2nd_Year\SEM2\CSN252\SIC XE Assembler>
```

Listing_File.txt containing the object codes.

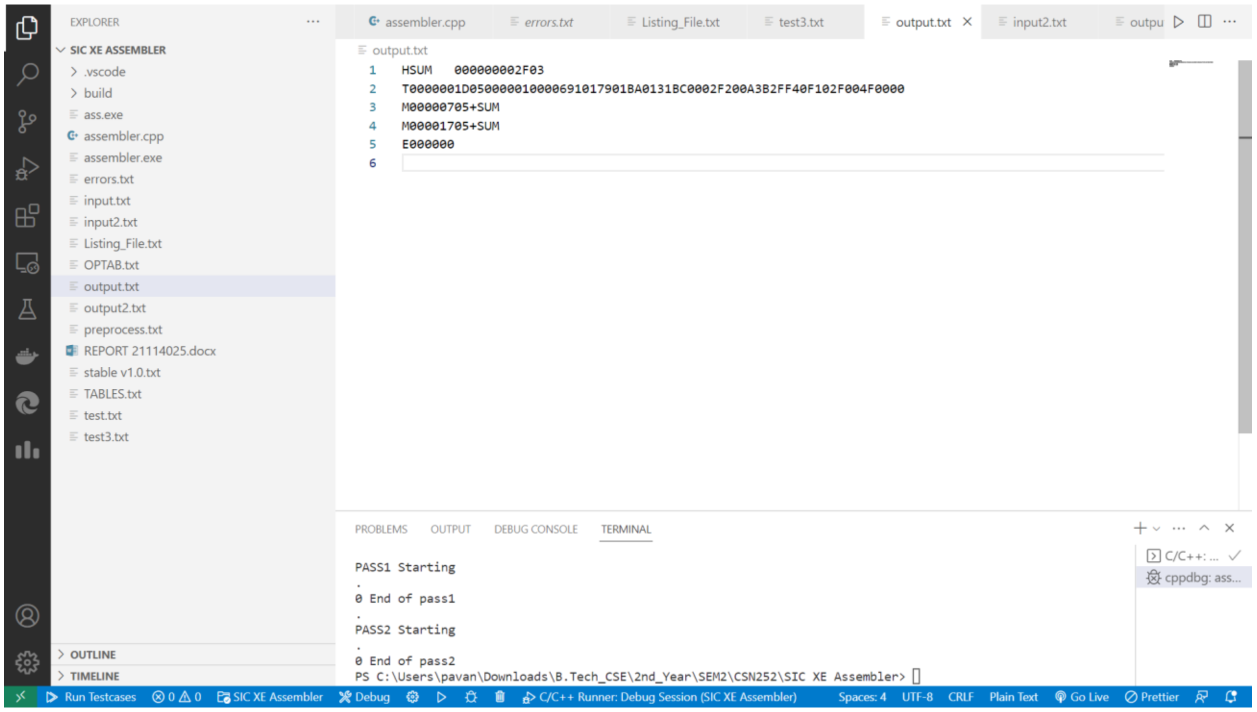


```
PASS1:
PASS2:

```

```
PASS1 Starting
.
0 End of pass1
.
PASS2 Starting
.
0 End of pass2
PS C:\Users\pavan\Downloads\B.Tech_CSE\2nd_Year\SEM2\CSN252\SIC XE Assembler>
```

b) errors.txt contains the errors In pass1 and pass2 respectively.



```
HSUM   000000002F03
T0000001D050000010000691017901BA0131BC0002F200A3B2FF40F102F004F0000
M00000705+SUM
M00001705+SUM
E000000

```

```
PASS1 Starting
.
0 End of pass1
.
PASS2 Starting
.
0 End of pass2
PS C:\Users\pavan\Downloads\B.Tech_CSE\2nd_Year\SEM2\CSN252\SIC XE Assembler>
```

c) ”output.txt” contains the object program of the SIC/XE.

Working of Assembler

- The Input is preprocessed by preprocess.txt which removes comments, white spaces, and converts the input.txt to preprocess.txt.
- Pass1 assigns address to all statements in the program and saves the values(addresses) assigned to all labels. Some assembler directives are processed.
- Pass2 assembles instructions by generating Opcodes using tables created in pass1. The data values defined by BYTE, WORD, etc. are assigned values. The assembler directives that were not processed during pass1 are processed. The object program and the assembly listing are written into output.txt and Listing_File.txt.

CONCLUSION

This SIC/XE assembler supports control sections. The outputs for sample inputs are given in the folder (sample io) attached to this document.

The GitHub link of the project is:

pavansai444/21114025-Assembler-SIc-Xe: This assembler is cpp implementation of SIC XE assembler that implements only control sections. (github.com)