CSN-252 TUTORIAL-08 SIC-XE ASSEMBLER

Uppala Vivek Narayan,

21114108,

O4 SUB BATCH.

ABOUT THE ASSEMBLER:

Here we are trying to implement the assembler (a 2 pass assembler specifically). This assembler designed has the following features as mentioned below:

- Converts the given assembly program into object program.

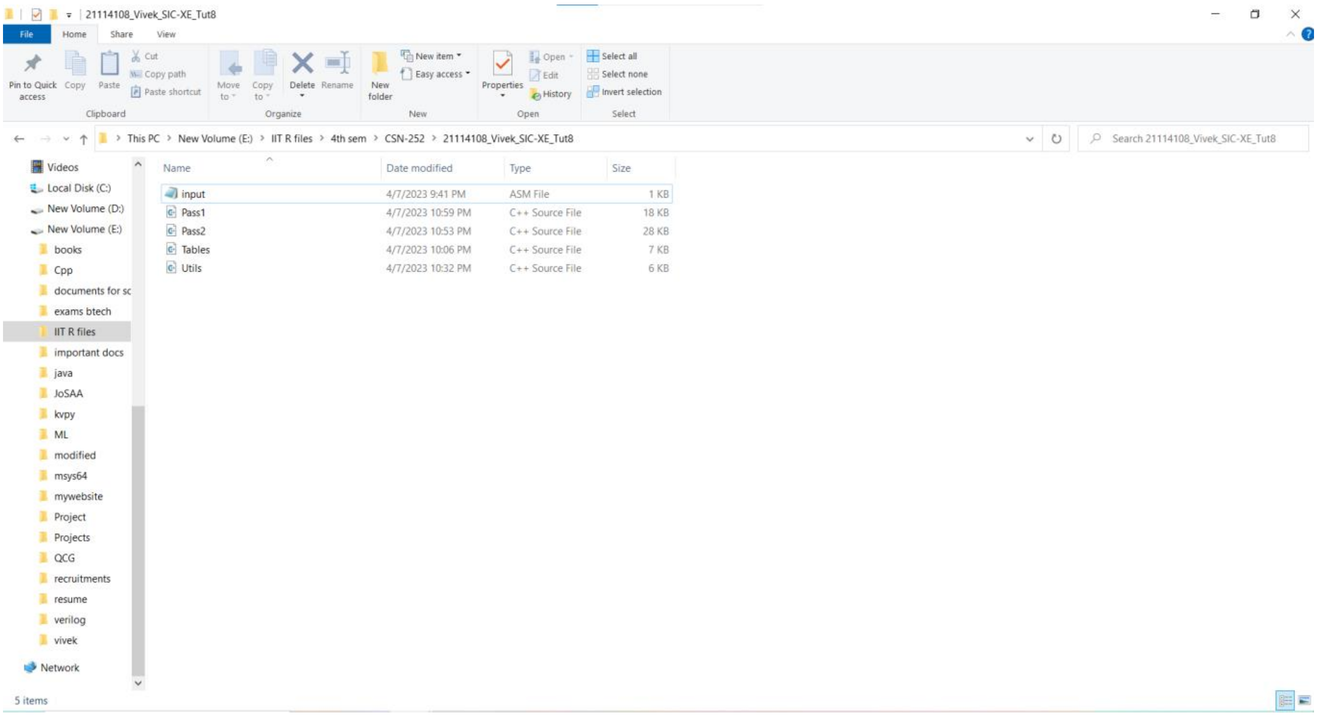
- It also produces given files as output:
  - Object program file : It contains the object program for the input assembly program given. The header record, text records, end record and modification records .
  - Intermediate file : The file has some useful listing like the addresses of all instuctions and variables that is produced as an output of pass1 and which is used by pass2.
  - Listing file : This file contains the listings of object code, addresses of all instructions and is an output of pass2.
  - Error file : This file contains the errors occurred in the assembly code while pass1 and pass2 run.
  - Tables file : This table contains all the tables which is necessary during the object code production like symbol and literal tables.
- **Assembler Input**- Assembler source program using the instruction set of SIC/XE.
- **Assembler Output-** Assembler will generate the following files as output-
  o Pass 1 will generate a Symbol Table, Intermediate file for pass2 .
- My assembler supports the following machine independent features-

  1. Literals

2. Symbol Defining Statements
3. Expressions
4. Program Blocks
   - Pass 2 will generate a listing file containing the input assembly code and address, block number, object code of each instruction, an object program file, and an error file.
   - Note: My assembler only works for program blocks and not for control sections since I am an even enrollment number person.

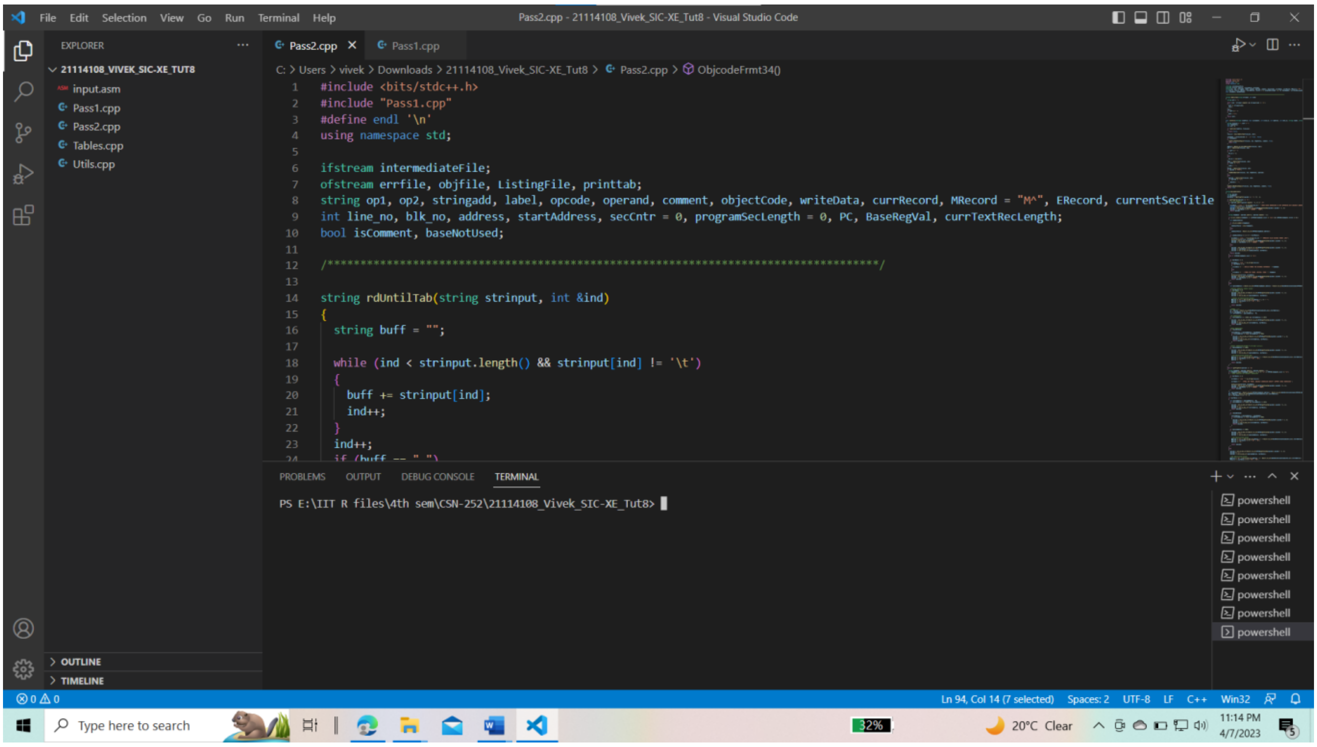
Procedure to use the assembler in your system:

- Download the zip file named as SIC-XE(or any other name given).
- Extract it and open that folder which contains 5 files. Four being Pass1, Pass2, Utils, and Tables along with a input program file**.**



Name	Date modified	Type	Size
input	4/7/2023 9:41 PM	ASM File	1 KB
Pass1	4/7/2023 10:59 PM	C++ Source File	18 KB
Pass2	4/7/2023 10:53 PM	C++ Source File	28 KB
Tables	4/7/2023 10:06 PM	C++ Source File	7 KB
Utils	4/7/2023 10:32 PM	C++ Source File	6 KB

- Now open this folder in your desired editors like I used VScode, or you can directly run-in command prompt.

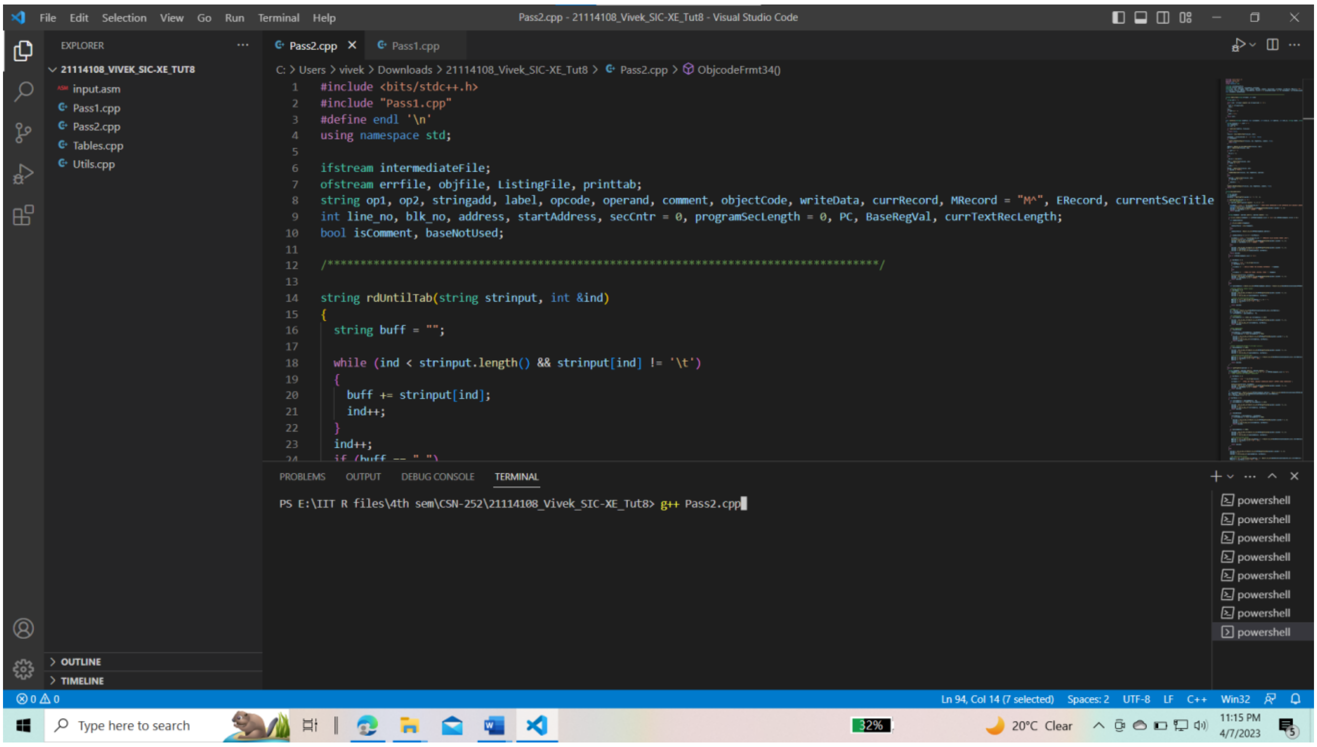


Pass2.cpp - 21114108_Vivek_SIC-XE_Tut8 - Visual Studio Code
21114108_VIVEK_SIC-XE_TUT8
input.asm
Pass1.cpp
Pass2.cpp
Tables.cpp
Utils.cpp
C: > Users > vivek > Downloads > 21114108_Vivek_SIC-XE_Tut8 > Pass2.cpp > ObjcodeFrmt34()

```
#include <bits/stdc++.h>
#include "Pass1.cpp"
#define endl '\n'
using namespace std;

ifstream intermediateFile;
ofstream errfile, objfile, ListingFile, printtab;
string op1, op2, stringadd, label, opcode, operand, comment, objectCode, writeData, currRecord, MRecord = "M^", ERecord, currentSecTitle
int line_no, blk_no, address, startAddress, secCntr = 0, programSecLength = 0, PC, BaseRegVal, currTextRecLength;
bool isComment, baseNotUsed;

/*****************************************************************************/

string rdUntilTab(string strinput, int &ind)
{
  string buff = "";

  while (ind < strinput.length() && strinput[ind] != '\t')
  {
    buff += strinput[ind];
    ind++;
  }
  ind++;
  if (buff == " ")
```

```
PS E:\IIT R files\4th sem\CSN-252\21114108_Vivek_SIC-XE_Tut8>
```

- Go to this directory. Now we run the command “g++ Pass2.cpp” for compilation. You get a new file “a.exe” which is the file to be run now.

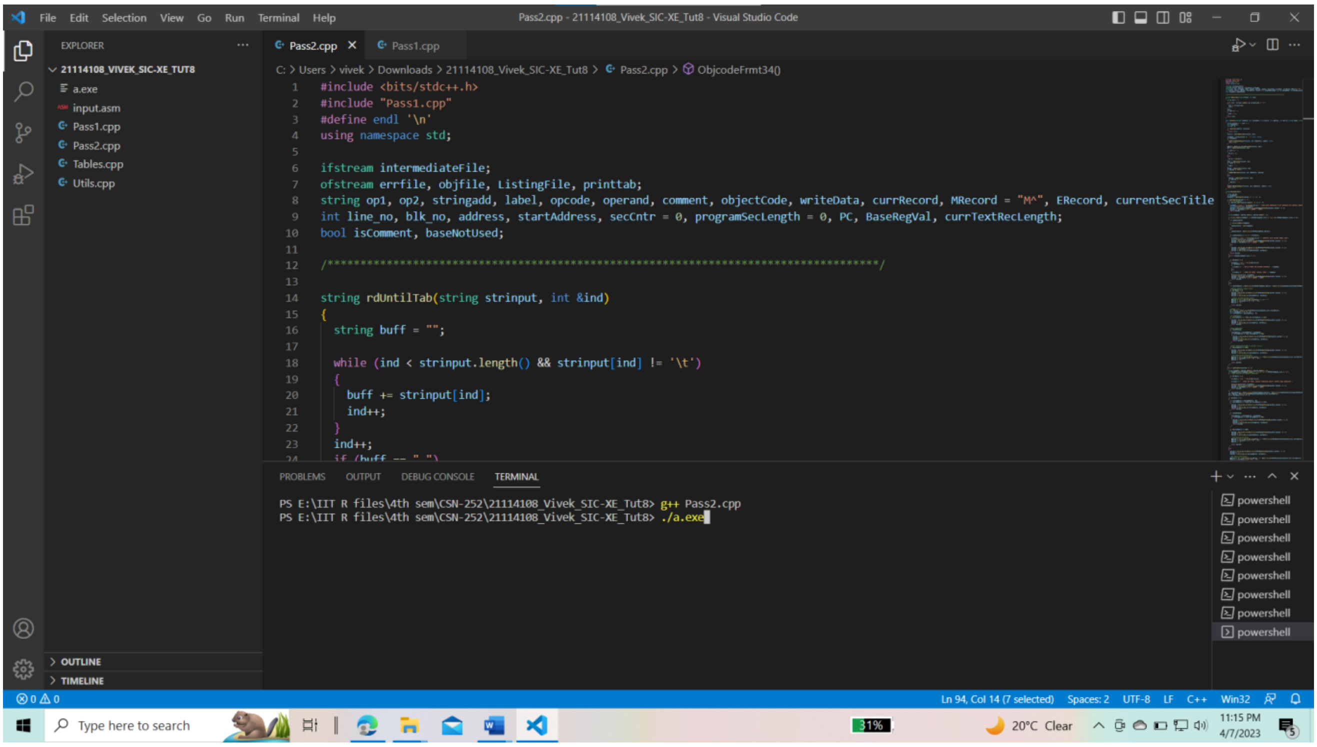


Pass2.cpp - 21114108_Vivek_SIC-XE_Tut8 - Visual Studio Code
21114108_VIVEK_SIC-XE_TUT8
input.asm
Pass1.cpp
Pass2.cpp
Tables.cpp
Utils.cpp
C: > Users > vivek > Downloads > 21114108_Vivek_SIC-XE_Tut8 > Pass2.cpp > ObjcodeFrmt34()

```
#include <bits/stdc++.h>
#include "Pass1.cpp"
#define endl '\n'
using namespace std;

ifstream intermediateFile;
ofstream errfile, objfile, ListingFile, printtab;
string op1, op2, stringadd, label, opcode, operand, comment, objectCode, writeData, currRecord, MRecord = "M^", ERecord, currentSecTitle
int line_no, blk_no, address, startAddress, secCntr = 0, programSecLength = 0, PC, BaseRegVal, currTextRecLength;
bool isComment, baseNotUsed;

/*****************************************************************************/

string rdUntilTab(string strinput, int &ind)
{
  string buff = "";

  while (ind < strinput.length() && strinput[ind] != '\t')
  {
    buff += strinput[ind];
    ind++;
  }
  ind++;
  if (buff == " ")
```

```
PS E:\IIT R files\4th sem\CSN-252\21114108_Vivek_SIC-XE_Tut8> g++ Pass2.cpp
```

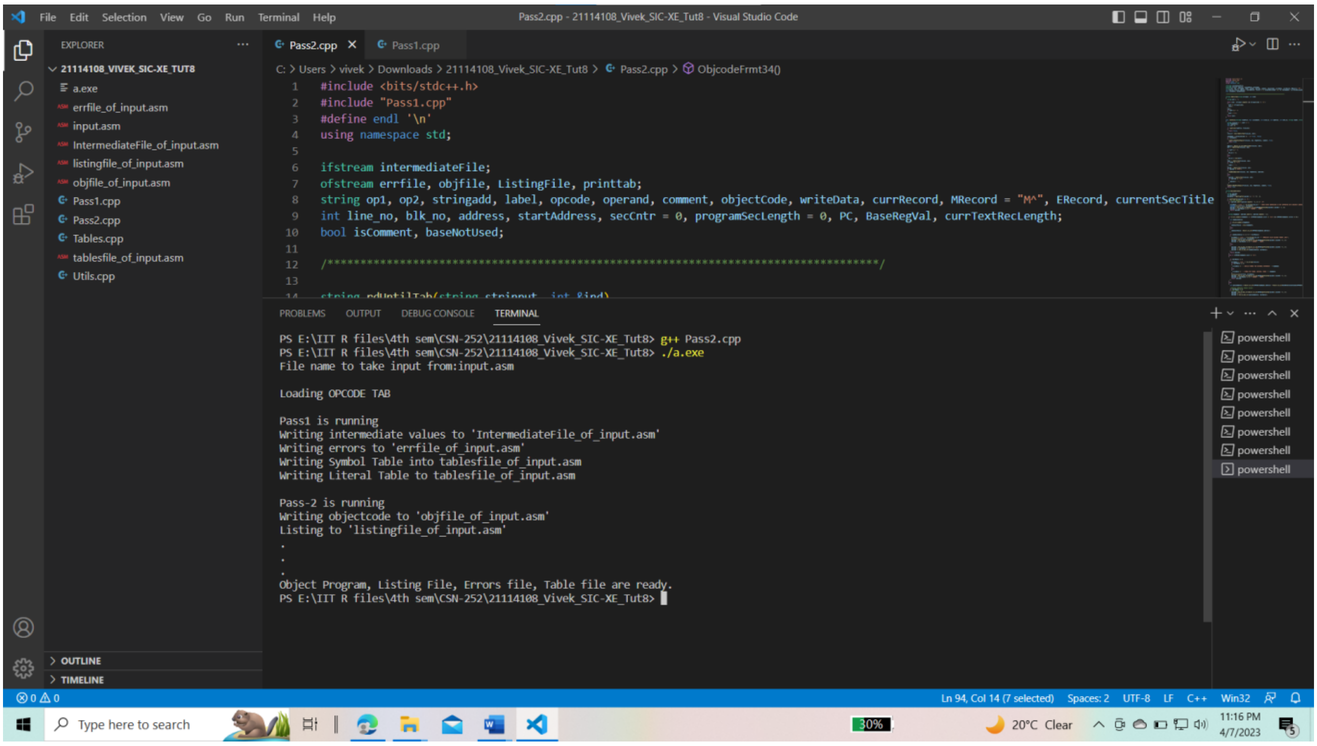
- Now run the command “.\a.exe” which runs the code, and we are asked to give input file name.


Pass2.cpp - 21114108_Vivek_SIC-XE_Tut8 - Visual Studio Code
EXPLORER
21114108_VIVEK_SIC-XE_TUT8
a.exe
input.asm
Pass1.cpp
Pass2.cpp
Tables.cpp
Utils.cpp
Pass2.cpp
Pass1.cpp
C: > Users > vivek > Downloads > 21114108_Vivek_SIC-XE_Tut8 > Pass2.cpp > ObjcodeFrmt34()
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
OUTLINE
TIMELINE

```cpp
#include <bits/stdc++.h>
#include "Pass1.cpp"
#define endl '\n'
using namespace std;

ifstream intermediateFile;
ofstream errfile, objfile, ListingFile, printtab;
string op1, op2, stringadd, label, opcode, operand, comment, objectCode, writeData, currRecord, MRecord = "M^", ERecord, currentSecTitle
int line_no, blk_no, address, startAddress, secCntr = 0, programSecLength = 0, PC, BaseRegVal, currTextRecLength;
bool isComment, baseNotUsed;

/*********************************************************************************/

string rdUntilTab(string strinput, int &ind)
{
  string buff = "";

  while (ind < strinput.length() && strinput[ind] != '\t')
  {
    buff += strinput[ind];
    ind++;
  }
  ind++;
```

```
PS E:\IIT R files\4th sem\CSN-252\21114108_Vivek_SIC-XE_Tut8> g++ Pass2.cpp
PS E:\IIT R files\4th sem\CSN-252\21114108_Vivek_SIC-XE_Tut8> ./a.exe
```

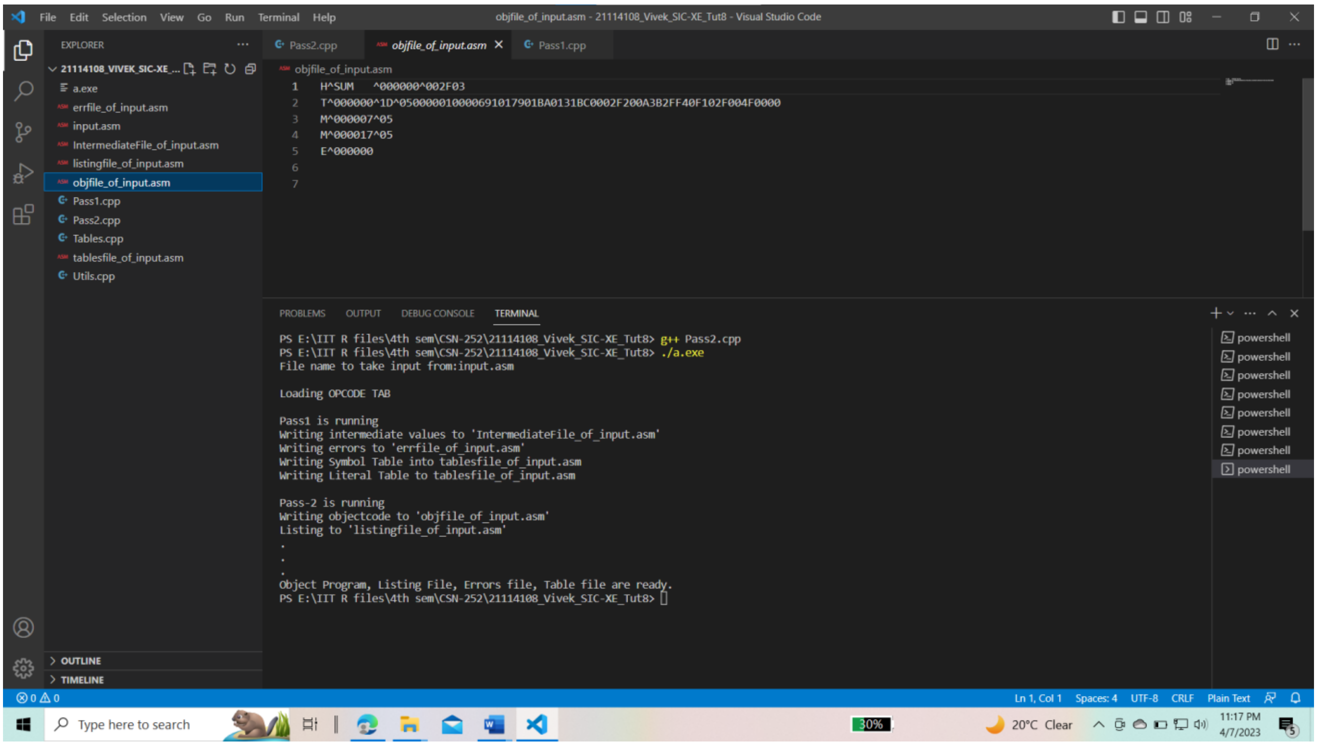
- As we have the input program in input.asm already(we can use some other input file with different program) we enter it.



```
PS E:\IIT R files\4th sem\CSN-252\21114108_Vivek_SIC-XE_Tut8> g++ Pass2.cpp
PS E:\IIT R files\4th sem\CSN-252\21114108_Vivek_SIC-XE_Tut8> ./a.exe
File name to take input from:input.asm

Loading OPCODE TAB

Pass1 is running
Writing intermediate values to 'IntermediateFile_of_input.asm'
Writing errors to 'errfile_of_input.asm'
Writing Symbol Table into tablesfile_of_input.asm
Writing Literal Table to tablesfile_of_input.asm

Pass-2 is running
Writing objectcode to 'objfile_of_input.asm'
Listing to 'listingfile_of_input.asm'
.
.
.
Object Program, Listing File, Errors file, Table file are ready.
PS E:\IIT R files\4th sem\CSN-252\21114108_Vivek_SIC-XE_Tut8>
```

- The code is run successfully, and the terminal consists of the output dialogues as above. ○ You can also see 5 additional files. They are mentioned below.

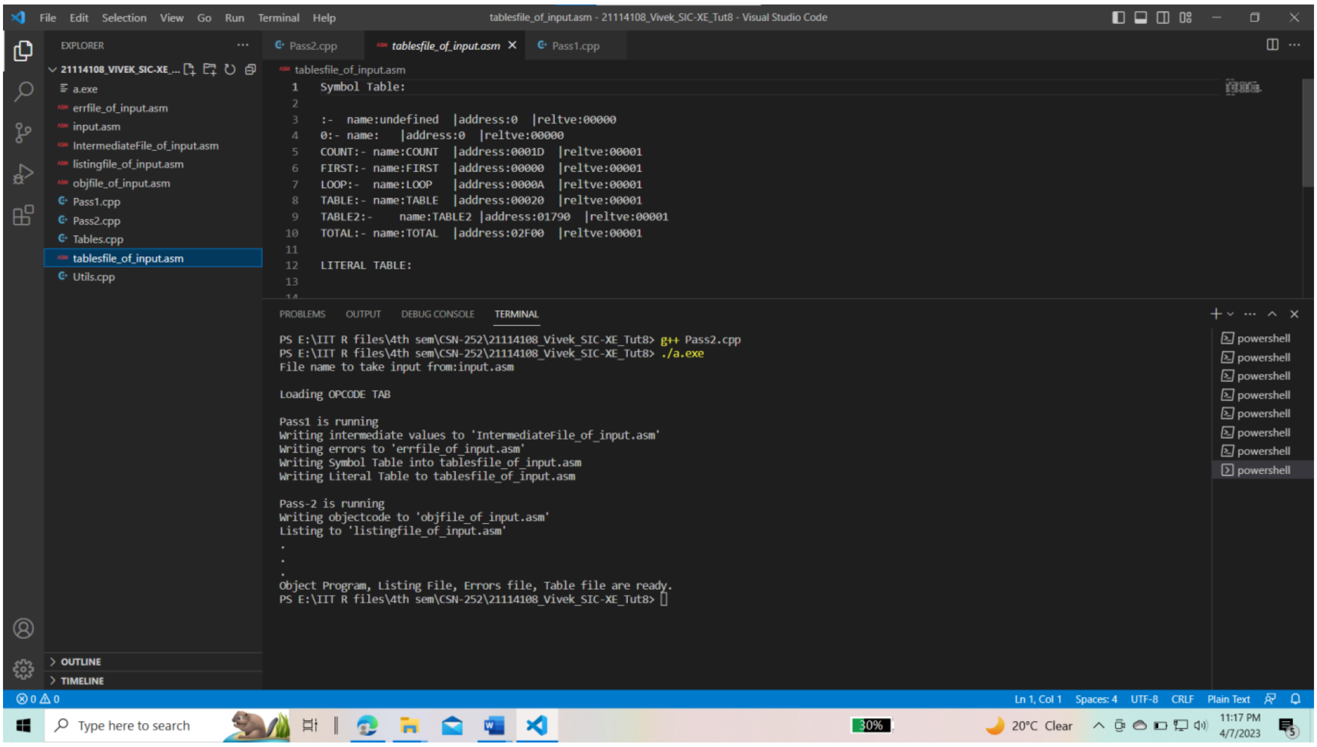
- objfile_of_input.asm



```
H^SUM   ^000000^002F03
T^000000^1D^050000010000691017901BA0131BC0002F200A3B2FF40F102F004F0000
M^000007^05
M^000017^05
E^000000
```

```
PS E:\IIT R files\4th sem\CSN-252\21114108_Vivek_SIC-XE_Tut8> g++ Pass2.cpp
PS E:\IIT R files\4th sem\CSN-252\21114108_Vivek_SIC-XE_Tut8> ./a.exe
File name to take input from:input.asm

Loading OPCODE TAB

Pass1 is running
Writing intermediate values to 'IntermediateFile_of_input.asm'
Writing errors to 'errfile_of_input.asm'
Writing Symbol Table into tablesfile_of_input.asm
Writing Literal Table to tablesfile_of_input.asm

Pass-2 is running
Writing objectcode to 'objfile_of_input.asm'
Listing to 'listingfile_of_input.asm'
.
.
.
Object Program, Listing File, Errors file, Table file are ready.
PS E:\IIT R files\4th sem\CSN-252\21114108_Vivek_SIC-XE_Tut8> 
```

- tablesfile_of_input.asm

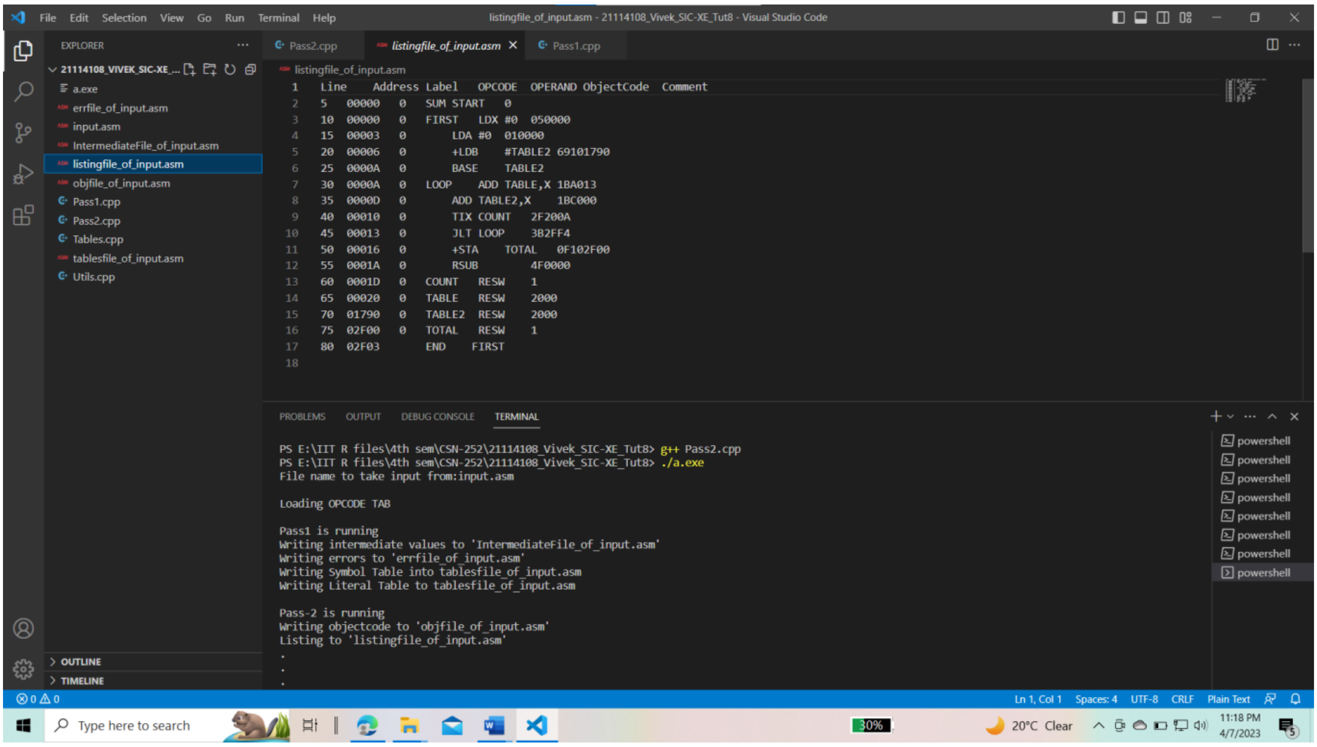


tablesfile_of_input.asm - 21114108_Vivek_SIC-XE_Tut8 - Visual Studio Code
Pass2.cpp
tablesfile_of_input.asm
Pass1.cpp

```
Symbol Table:

:-  name:undefined  |address:0  |reltve:00000
0:- name:   |address:0  |reltve:00000
COUNT:- name:COUNT  |address:0001D  |reltve:00001
FIRST:- name:FIRST  |address:00000  |reltve:00001
LOOP:-  name:LOOP   |address:0000A  |reltve:00001
TABLE:- name:TABLE  |address:00020  |reltve:00001
TABLE2:-    name:TABLE2 |address:01790  |reltve:00001
TOTAL:- name:TOTAL  |address:02F00  |reltve:00001

LITERAL TABLE:

```

```
PS E:\IIT R files\4th sem\CSN-252\21114108_Vivek_SIC-XE_Tut8> g++ Pass2.cpp
PS E:\IIT R files\4th sem\CSN-252\21114108_Vivek_SIC-XE_Tut8> ./a.exe
File name to take input from:input.asm

Loading OPCODE TAB

Pass1 is running
Writing intermediate values to 'IntermediateFile_of_input.asm'
Writing errors to 'errfile_of_input.asm'
Writing Symbol Table into tablesfile_of_input.asm
Writing Literal Table to tablesfile_of_input.asm

Pass-2 is running
Writing objectcode to 'objfile_of_input.asm'
Listing to 'listingfile_of_input.asm'
.
.
.
Object Program, Listing File, Errors file, Table file are ready.
PS E:\IIT R files\4th sem\CSN-252\21114108_Vivek_SIC-XE_Tut8> 
```

- listingfile_of_input.asm

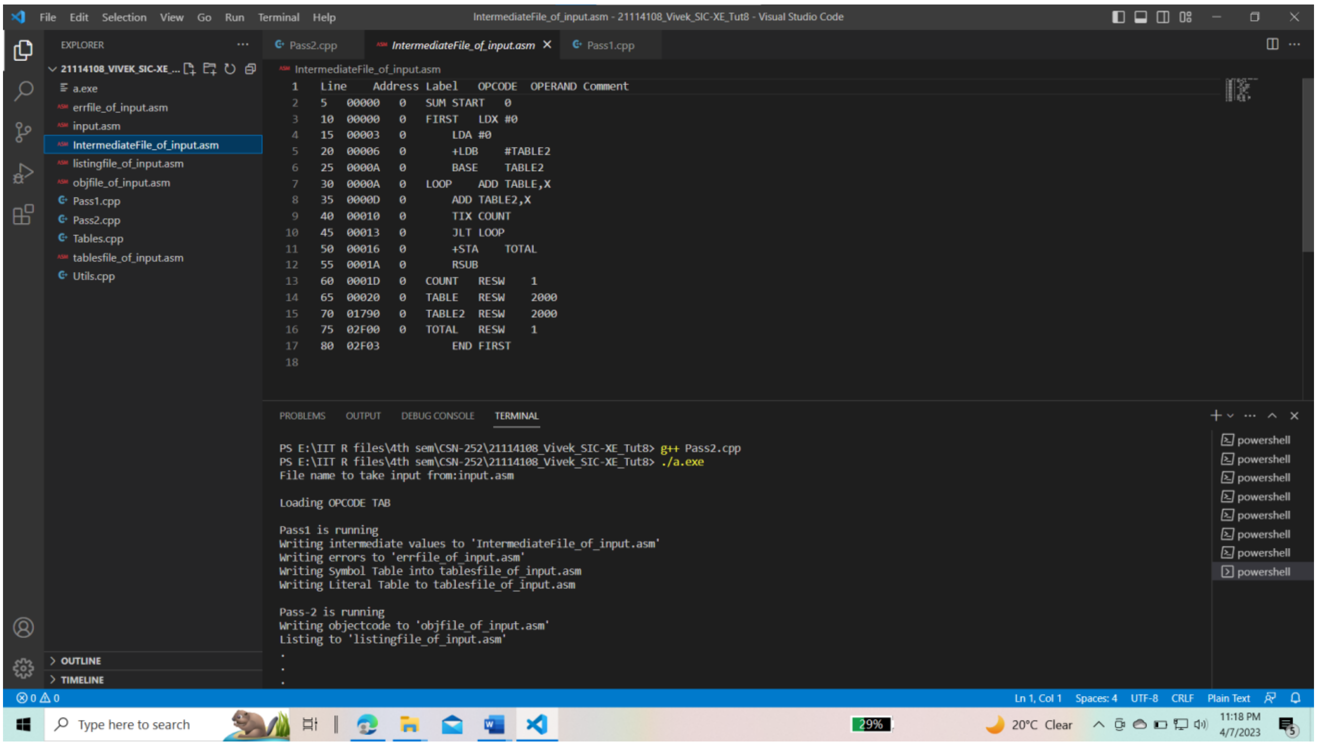


listingfile_of_input.asm - 21114108_Vivek_SIC-XE_Tut8 - Visual Studio Code
Pass2.cpp
listingfile_of_input.asm
Pass1.cpp

```
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

```

```
PS E:\IIT R files\4th sem\CSN-252\21114108_Vivek_SIC-XE_Tut8> g++ Pass2.cpp
PS E:\IIT R files\4th sem\CSN-252\21114108_Vivek_SIC-XE_Tut8> ./a.exe
File name to take input from:input.asm

Loading OPCODE TAB

Pass1 is running
Writing intermediate values to 'IntermediateFile_of_input.asm'
Writing errors to 'errfile_of_input.asm'
Writing Symbol Table into tablesfile_of_input.asm
Writing Literal Table to tablesfile_of_input.asm

Pass-2 is running
Writing objectcode to 'objfile_of_input.asm'
Listing to 'listingfile_of_input.asm'
.
.
```

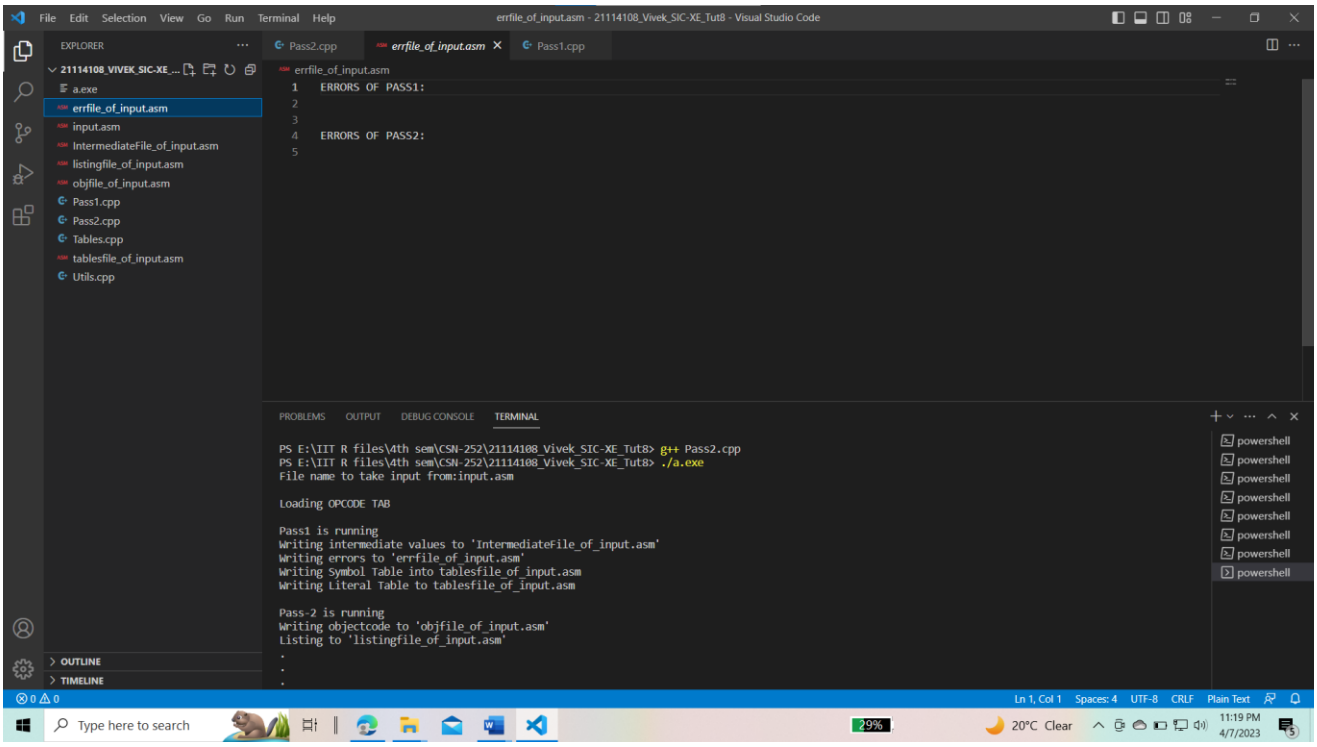
- IntermediateFile_of_input.asm



```
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
```

```
PS E:\IIT R files\4th sem\CSN-252\21114108_Vivek_SIC-XE_Tut8> g++ Pass2.cpp
PS E:\IIT R files\4th sem\CSN-252\21114108_Vivek_SIC-XE_Tut8> ./a.exe
File name to take input from:input.asm

Loading OPCODE TAB

Pass1 is running
Writing intermediate values to 'IntermediateFile_of_input.asm'
Writing errors to 'errfile_of_input.asm'
Writing Symbol Table into tablesfile_of_input.asm
Writing Literal Table to tablesfile_of_input.asm

Pass-2 is running
Writing objectcode to 'objfile_of_input.asm'
Listing to 'listingfile_of_input.asm'
.
.
.
```

- errfile_of_input.asm



```
ERRORS OF PASS1:


ERRORS OF PASS2:
```

```
PS E:\IIT R files\4th sem\CSN-252\21114108_Vivek_SIC-XE_Tut8> g++ Pass2.cpp
PS E:\IIT R files\4th sem\CSN-252\21114108_Vivek_SIC-XE_Tut8> ./a.exe
File name to take input from:input.asm

Loading OPCODE TAB

Pass1 is running
Writing intermediate values to 'IntermediateFile_of_input.asm'
Writing errors to 'errfile_of_input.asm'
Writing Symbol Table into tablesfile_of_input.asm
Writing Literal Table to tablesfile_of_input.asm

Pass-2 is running
Writing objectcode to 'objfile_of_input.asm'
Listing to 'listingfile_of_input.asm'
.
.
.
```

INPUT AND OUTPUT:

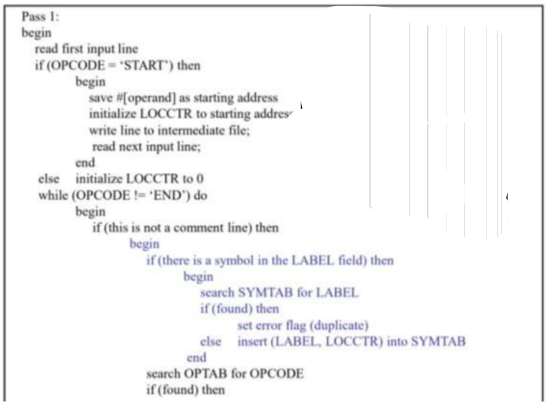
- Sample input code:

```
SUM    START  0
FIRST  LDX   #0
       LDA   #0
       +LDB  #0
       +LDB  #TABLE2
       LDT   =X'05'
       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
```

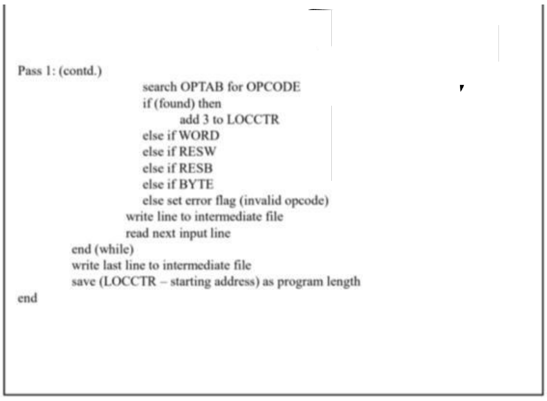
- Input output object code:

```
H^SUM   ^000000^002F03
T^000000^1D^050000010000691017901BA0131BC0002F200A3B2FF40F102F004F0000
M^000007^05
M^000017^05
E^000000
```

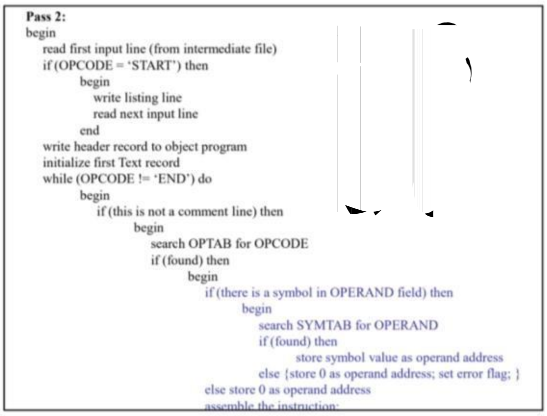
The design of the assembler is built upon the principles described by L.L.Beck in the book suggested by sir. I am attaching a few pseudocodes I followed to get to know the right implementation of Pass1 and Pass2 and built upon it.



```
Pass 1:
begin
   read first input line
   if (OPCODE = 'START') then
            begin
               save #[operand] as starting address
               initialize LOCCTR to starting addres
               write line to intermediate file;
                read next input line;
            end
    else    initialize LOCCTR to 0
    while (OPCODE != 'END') do
            begin
                if (this is not a comment line) then
                        begin
                           if (there is a symbol in the LABEL field) then
                                   begin
                                      search SYMTAB for LABEL
                                      if (found) then
                                              set error flag (duplicate)
                                      else    insert (LABEL, LOCCTR) into SYMTAB
                                    end
                           search OPTAB for OPCODE
                           if (found) then
```



```
Pass 1: (contd.)
                           search OPTAB for OPCODE
                           if (found) then
                                   add 3 to LOCCTR
                           else if WORD
                           else if RESW
                           else if RESB
                           else if BYTE
                           else set error flag (invalid opcode)
                        write line to intermediate file
                        read next input line
            end (while)
            write last line to intermediate file
            save (LOCCTR – starting address) as program length
end
```



```
Pass 2:
begin
    read first input line (from intermediate file)
    if (OPCODE = 'START') then
            begin
               write listing line
               read next input line
            end
    write header record to object program
    initialize first Text record
    while (OPCODE != 'END') do
            begin
                if (this is not a comment line) then
                        begin
                           search OPTAB for OPCODE
                           if (found) then
                                   begin
                                      if (there is a symbol in OPERAND field) then
                                              begin
                                                 search SYMTAB for OPERAND
                                                 if (found) then
                                                         store symbol value as operand address
                                                 else {store 0 as operand address; set error flag; }
                                      else store 0 as operand address
                                      assemble the instruction;
```

The above figures show us the algorithm implemented. We also implemented literals, expressions, program blocks in addition to the above algorithm.

Tables:

All the data structures required for the assembler to run is kept in this file. It contains the structs for labels, opcode, literal, blocks. After the execution of the pass1.cpp, the Tables like SYMTAB, LITTAB, etc., are printed in a separate file and then pass2.cpp is executed.

Data Structures used:

1. Map: Maps are associative containers that store elements formed by a combination of a key value and a mapped value, following a specific order.

Here map is used to store the SYMBOL TABLE, OPCODE TABLE, REGISTER TABLE, LITERAL TABLE, BLOCK TABLE. Each map of these tables contains a key in the form of string(data type) which represent an element of the table and the mapped value is a struct which stores the information of that element.

2. Structure: **Structures** are user defined data types which are used to store group of items of non-similar data types. Structures of each map are as below.

OPTAB

The struct contains information of opcodes: name, format type, and a character representing whether the opcode is valid or not.

REGTAB

The struct contains information of registers : its numeric equivalent or say register number, and a character representing whether the registers exits or not.

SYMTAB

The struct contains information of labels: name, address, block number, a character representing whether the label exits in the symbol table or not, an integer representing whether label is relative or not.

LITTAB

The struct contains information of literals: its value, address, block number, a character representing whether the literal exits in the literal table or not.

BLOCKS

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

CONCLUSION:

I have learnt the working of the SIC-XE assembler and the functionalities. Over the course of working on the project I realized quite a bit of new C++ functions and became better at understanding the pseudocodes explained in the book and understood the two-pass assembler completely. I thank Sir for giving us this as a homework since it cleared most of my doubts with regards to assemblers.