DAYANANDA SAGAR COLLEGE OF ENGINEERING
COMPUTER SCIENCE & ENGINEERING

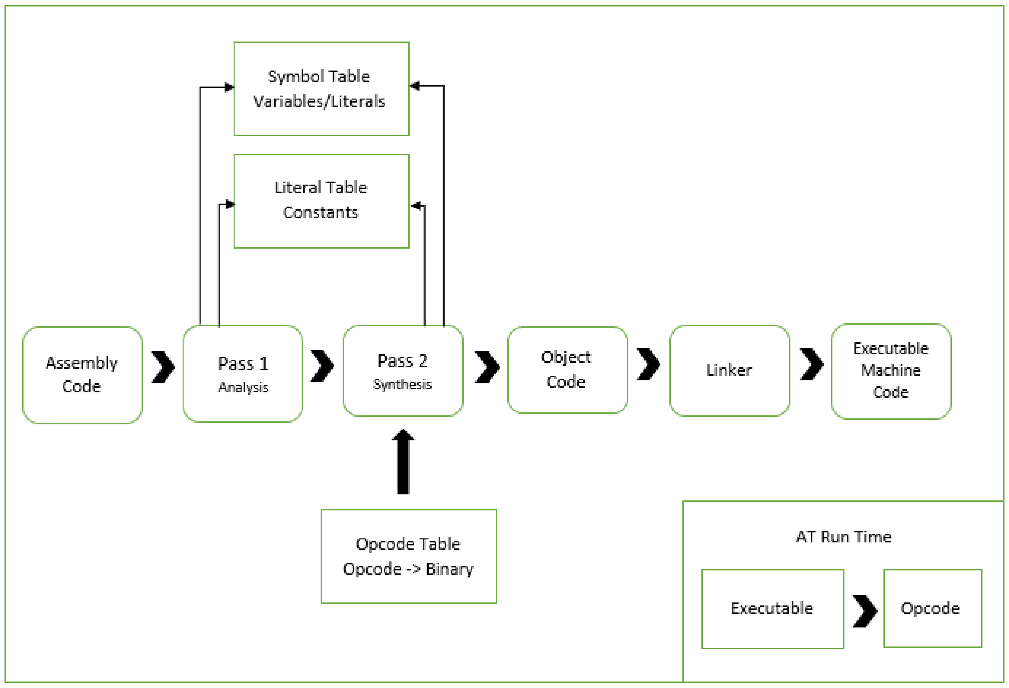
Minor Project- Report

Apr 2023-Jul 2023

Course Faculty:

Course Name & Code: SYSTEM SOFTWARE LAB WITH MINI-PROJECT (19CS6DCSSW)
Semester: VI Date: 14/06/2023

TITLE OF THE PROJECT	**TWO PASS ASSEMBLER**			
STUDENT NAME	PRUTHA VASISHT	PUNITH KUMAR P R	SUHAS REDDY	RACHANA K
USN	1DS20CS157	1DS20CS158	1DS20CS159	1DS20CS160
INDIVIDUAL CONTRIBUTION	Pass 2 Code in the Assembler	Pass 1 Code in the Assembler	Pass 1 Code in the Assembler	Pass 2 Code in the Assembler
GUIDE	Prof. Aparna			
PROJECT ABSTRACT:	A two-pass assembler works by performing the following steps in two passes: 1. In Pass 1, addresses are assigned to all the statements in the program. Then the values assigned to the labels and symbols are saved for use in Pass 2 using SYMTAB and OPTAB. It also processes pseudo-operations. An intermediate file is generated. 2. In Pass 2, instructions are assembled using values in SYMTAB and OPTAB. Data values defined by BYTE and WORD are generated and the assembler directives which were not processed in Pass 1 are processed. The object program and assembly listing are written. We will be focusing on generating machine codes from a set of assembly language codes.			
PLATFORM USED (H/W & S/W TOOLS TO BE USED)	Development Environment: Visual Studio Code, Windows OS Programming Language: C++			

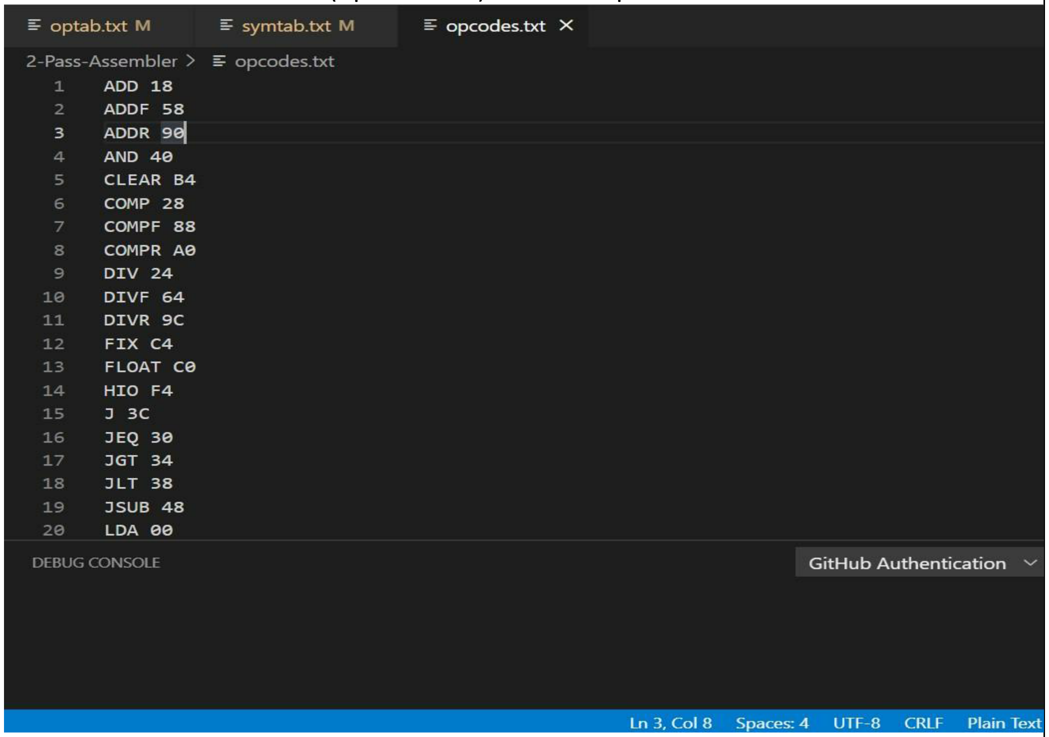
INTRODUCTION	An assembler is a program for converting instructions written in low-level assembly code into relocatable machine code and generating along information for the loader. If the assembler does all this work in one pass, then it is called a single pass assembler and otherwise if it does it in multiple passes then it is called a multiple pass assembler. When it does it in two passes, it is called a 2 Pass Assembler. The work done in both passes can be described as follows - Pass - 1: - Define symbols and literals and remember them in symbol table (SYMTAB) and literal table (OPTAB) respectively. - Keep track of the location counter. - Process pseudo-operations like macros and directives. Pass - 2: - Generate object code by converting symbolic opcode into respective numeric opcode. - Generate data for literals and look for values of symbols in OPTAB and SYMTAB.
DESIGN	The architectural design of a two-pass assembler can be shown as follows -  Symbol Table Variables/Literals Literal Table Constants Assembly Code Pass 1 Analysis Pass 2 Synthesis Object Code Linker Executable Machine Code Opcode Table Opcode -> Binary AT Run Time Executable Opcode

PROJECT SOURCE CODE LINK (GITHUB/ GOOGLE DRIVE)	https://github.com/punith-kumar-pr/2-pass-assembler
CONCLUSION /FUTURE ENHANCEMENT	The program implements the working of a 2 Pass Assembler for SIC machine. Further, the program could be written for SIC/XE machine where the instructions are written in 4 different formats.

UI SCREENSHOTS

INPUT:

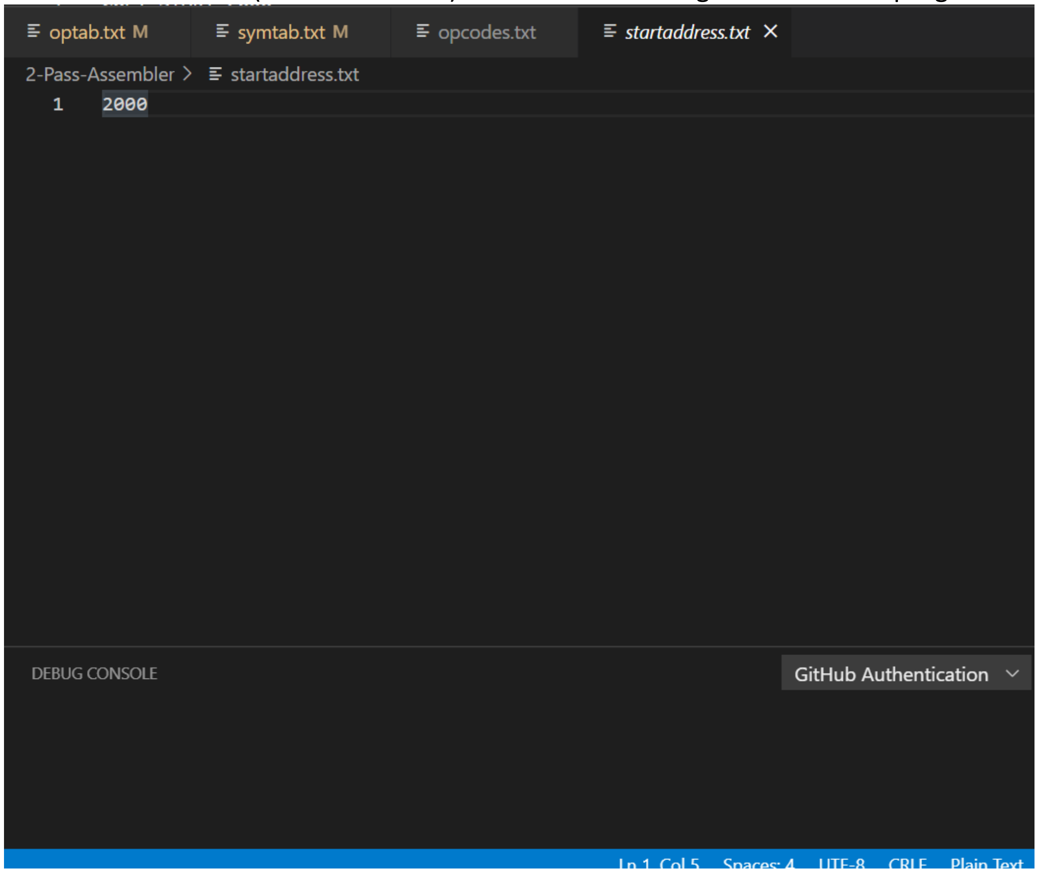
1. The below text file (opcodes.txt) contains opcodes for all mnemonics.



optab.txt M
symtab.txt M
opcodes.txt
2-Pass-Assembler > opcodes.txt
DEBUG CONSOLE
GitHub Authentication
Ln 3, Col 8 Spaces: 4 UTF-8 CRLF Plain Text

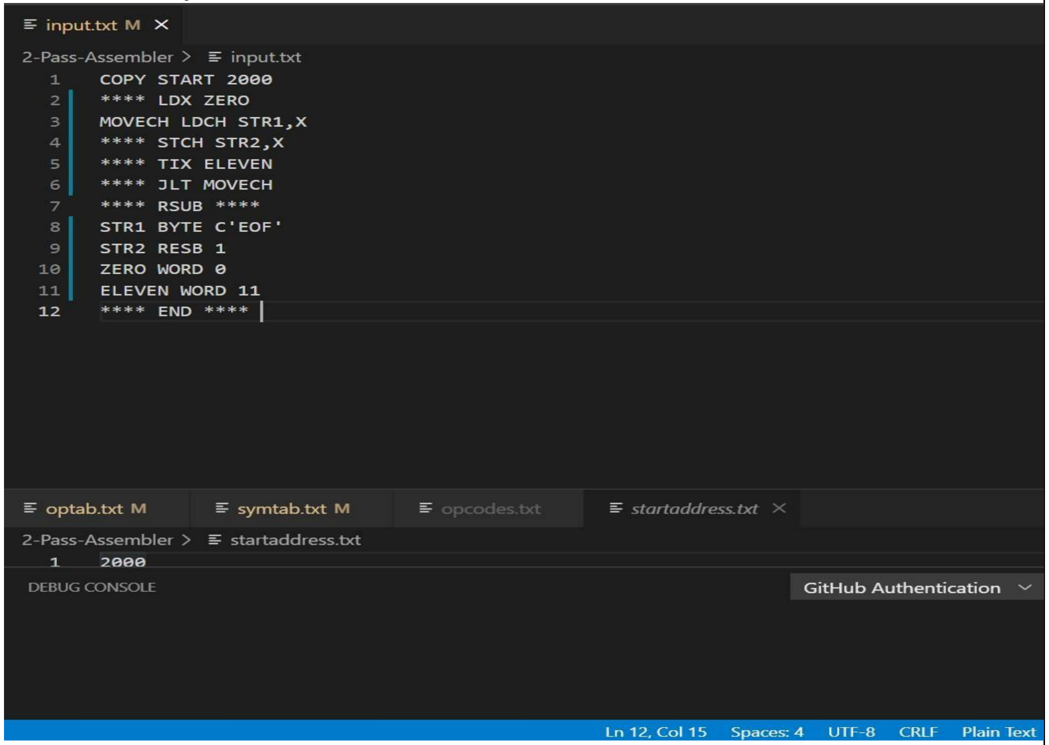
```
ADD 18
ADDF 58
ADDR 90
AND 40
CLEAR B4
COMP 28
COMPF 88
COMPR A0
DIV 24
DIVF 64
DIVR 9C
FIX C4
FLOAT C0
HIO F4
J 3C
JEQ 30
JGT 34
JLT 38
JSUB 48
LDA 00
```

2. This text file (startaddress.txt) contains the starting address of the program.



```
2000
```

3. This text file (input.txt) contains the source code that needs to be converted into object code.

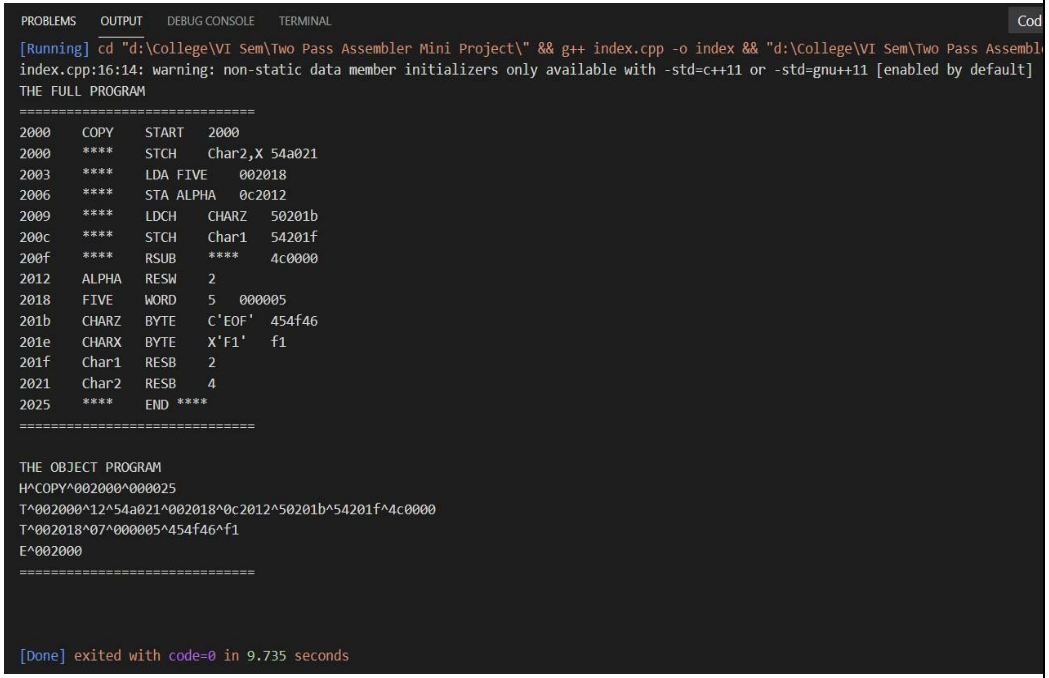


```
COPY START 2000
**** LDX ZERO
MOVECH LDCH STR1,X
**** STCH STR2,X
**** TIX ELEVEN
**** JLT MOVECH
**** RSUB ****
STR1 BYTE C'EOF'
STR2 RESB 1
ZERO WORD 0
ELEVEN WORD 11
**** END ****
```

```
2000
```

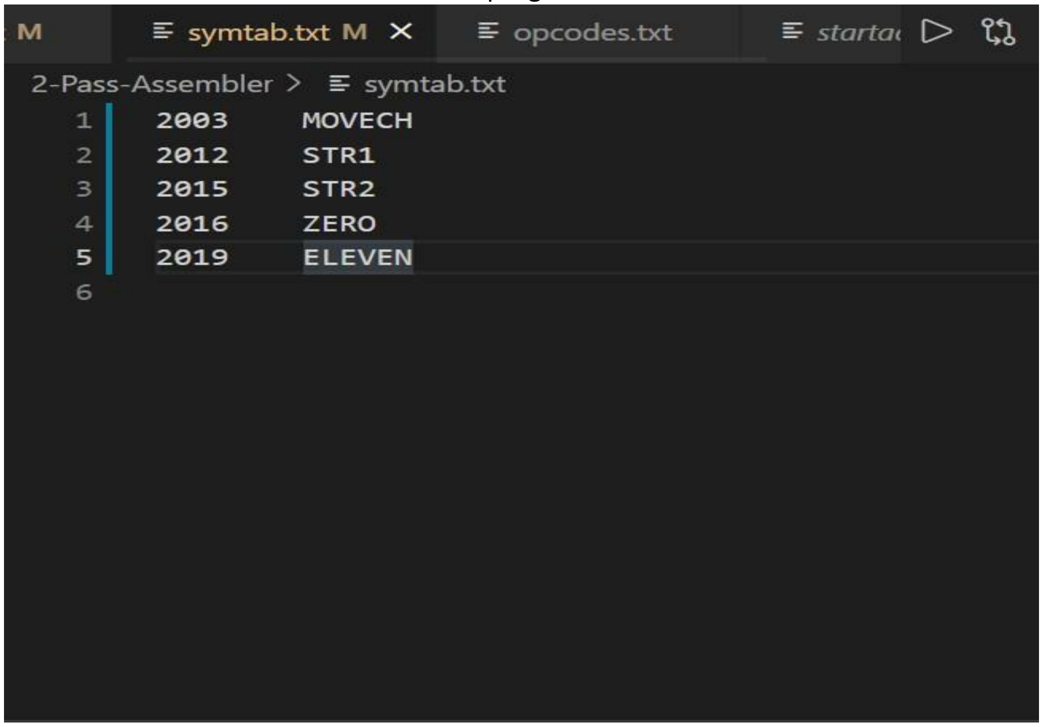
OUTPUT:

The final output program and object program is displayed.



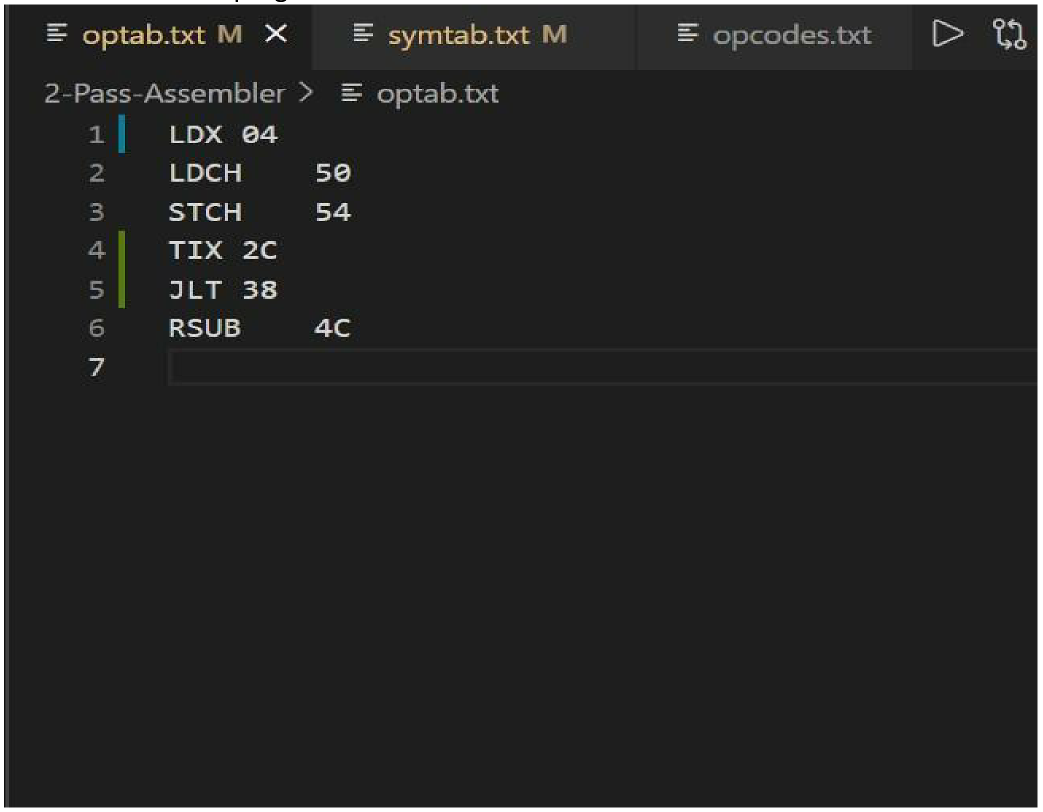
```
[Running] cd "d:\College\VI Sem\Two Pass Assembler Mini Project\" && g++ index.cpp -o index && "d:\College\VI Sem\Two Pass Assembl
index.cpp:16:14: warning: non-static data member initializers only available with -std=c++11 or -std=gnu++11 [enabled by default]
THE FULL PROGRAM
==============================
2000    COPY    START   2000
2000    ****    STCH    Char2,X 54a021
2003    ****    LDA FIVE    002018
2006    ****    STA ALPHA   0c2012
2009    ****    LDCH    CHARZ   50201b
200c    ****    STCH    Char1   54201f
200f    ****    RSUB    ****    4c0000
2012    ALPHA   RESW    2
2018    FIVE    WORD    5   000005
201b    CHARZ   BYTE    C'EOF'  454f46
201e    CHARX   BYTE    X'F1'   f1
201f    Char1   RESB    2
2021    Char2   RESB    4
2025    ****    END ****
==============================

THE OBJECT PROGRAM
H^COPY^002000^000025
T^002000^12^54a021^002018^0c2012^50201b^54201f^4c0000
T^002018^07^000005^454f46^f1
E^002000
==============================



[Done] exited with code=0 in 9.735 seconds
```

1. The following text file (symtab.txt) will contain the symbols and their addresses from the source program.



```
2003     MOVECH
2012     STR1
2015     STR2
2016     ZERO
2019     ELEVEN
```

2. This text file (optab.txt) contains the opcodes and mnemonics from the source program.



```
LDX 04
LDCH    50
STCH    54
TIX 2C
JLT 38
RSUB    4C
```

References	[1] "Two Pass Assemblers," www.entcengg.com. [Online]. Available: https://www.entcengg.com/two-pass-assemblers/. [Accessed: June 12, 2023]. [2] Prithi Mishra, System Software. Bengaluru: Subhas Publications, 2015.