# **CSN-252 // Tutorial-8**

Name: Vishwa Prakash

Enrol: 20114105

CSE, 2nd year

This is an implementation of a SIC/XE assembler in C++. This is a two pass assembler and it supports all four formats of (1, 2, 3, 4) addressing modes.

The following are the machine independent features implemented in this assembler:

1. Literals

2. Symbol Defining Statements

3. Expressions

4. Program Blocks

**Main working of the assembler:**

The main functions which execute the assembler are: load\_tables(), pass1(), pass2() and main().

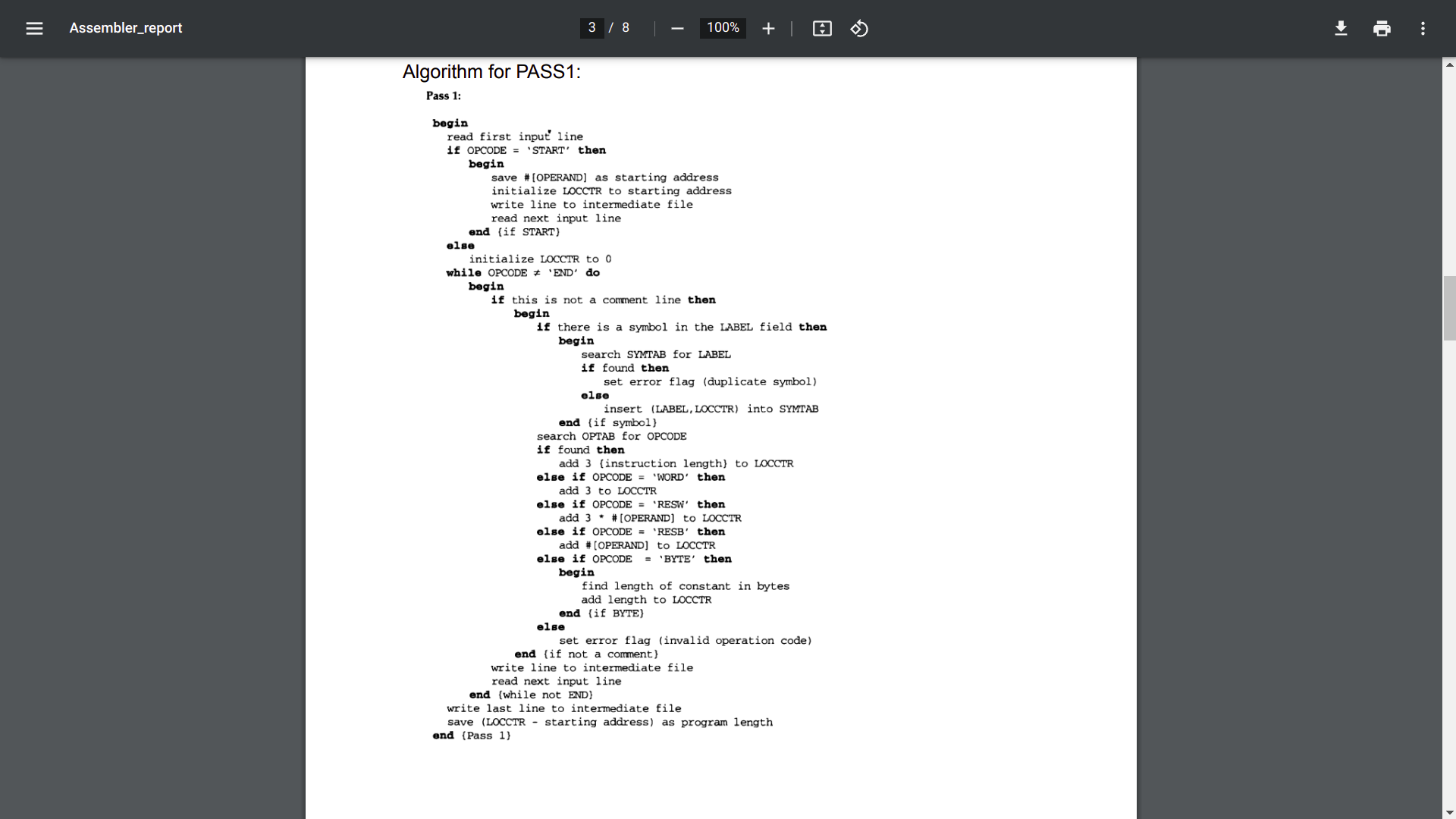
main() runs all passes and runs the assembler.

load\_tables() loads all the data like regtables and optables etc.

pass1():

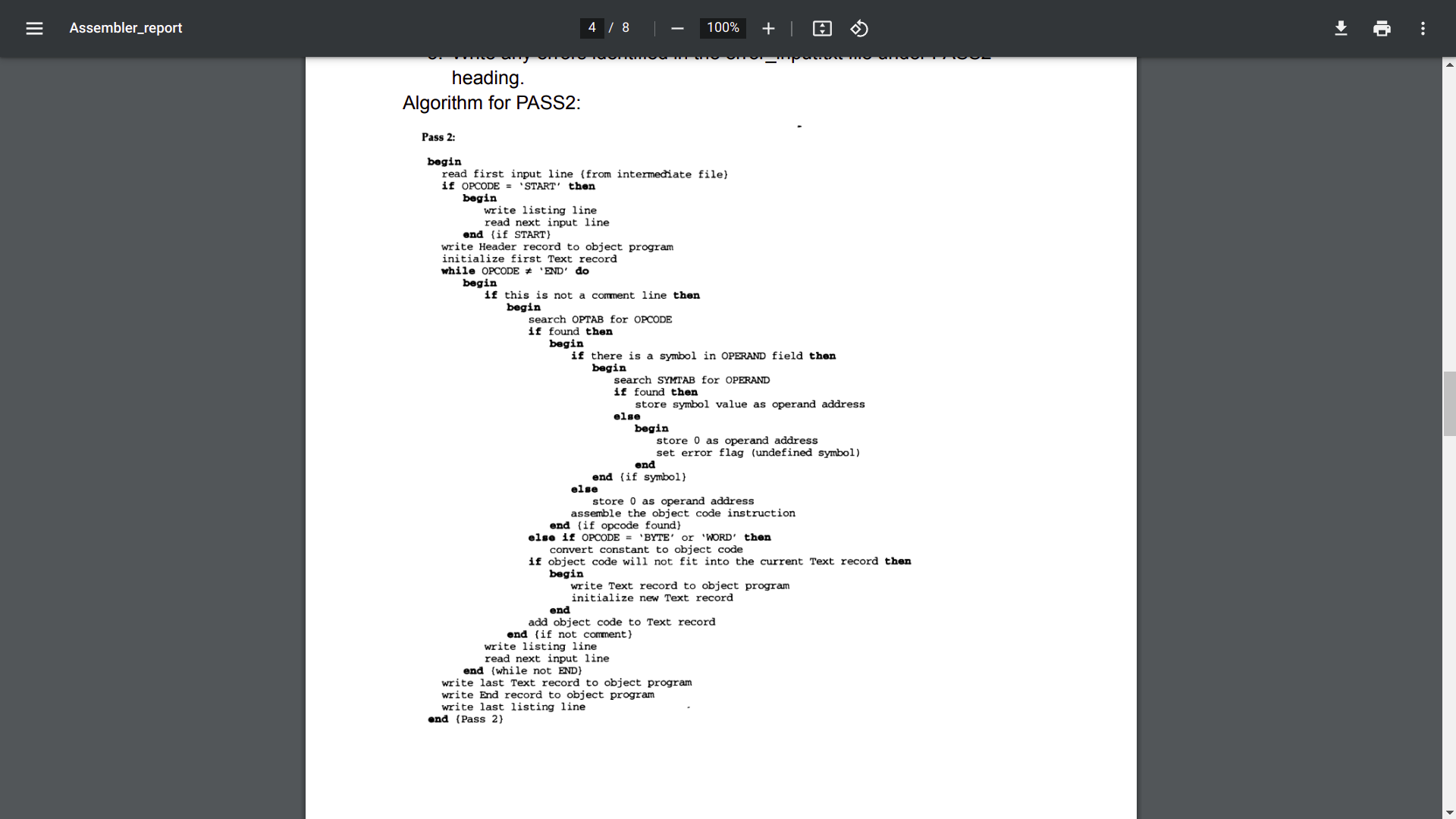
1. PASS 1(defines symbol)
2. Assigns address to all statements in the program.
3. Saves the values(addresses) assigned to labels for use in PASS2.
4. Perform some processing of the assembler directives(This includes processing that affects address assignment, such as determining the length of data areas defined by BYTE, RESW, etc).
5. Write any errors identified in the error\_input.txt file under PASS1 heading

Algorithm for pass1 is as follows:



1. Assemble instructions (translating operation codes and looking up addresses).
2. Generate data values defined by BYTE, WORD, etc.
3. Perform processing of assembler directives not done during PASS1.
4. Write the object program in the object\_input.txt and the assembly listing.
5. Write any errors identified in the error\_input.txt file under PASS2 heading.

Algorithm for Pass2 is as follows:



**So summary of execution of the Assembler:**

1. Pass1 generates a symbol table and an intermediate file for Pass2. 2. Pass2 generates a listing file containing the input assembly code and address, block number, object code of each instruction.

3. Pass 2 also generates an object program including the following type of record: H, D, R, T, M and E types.

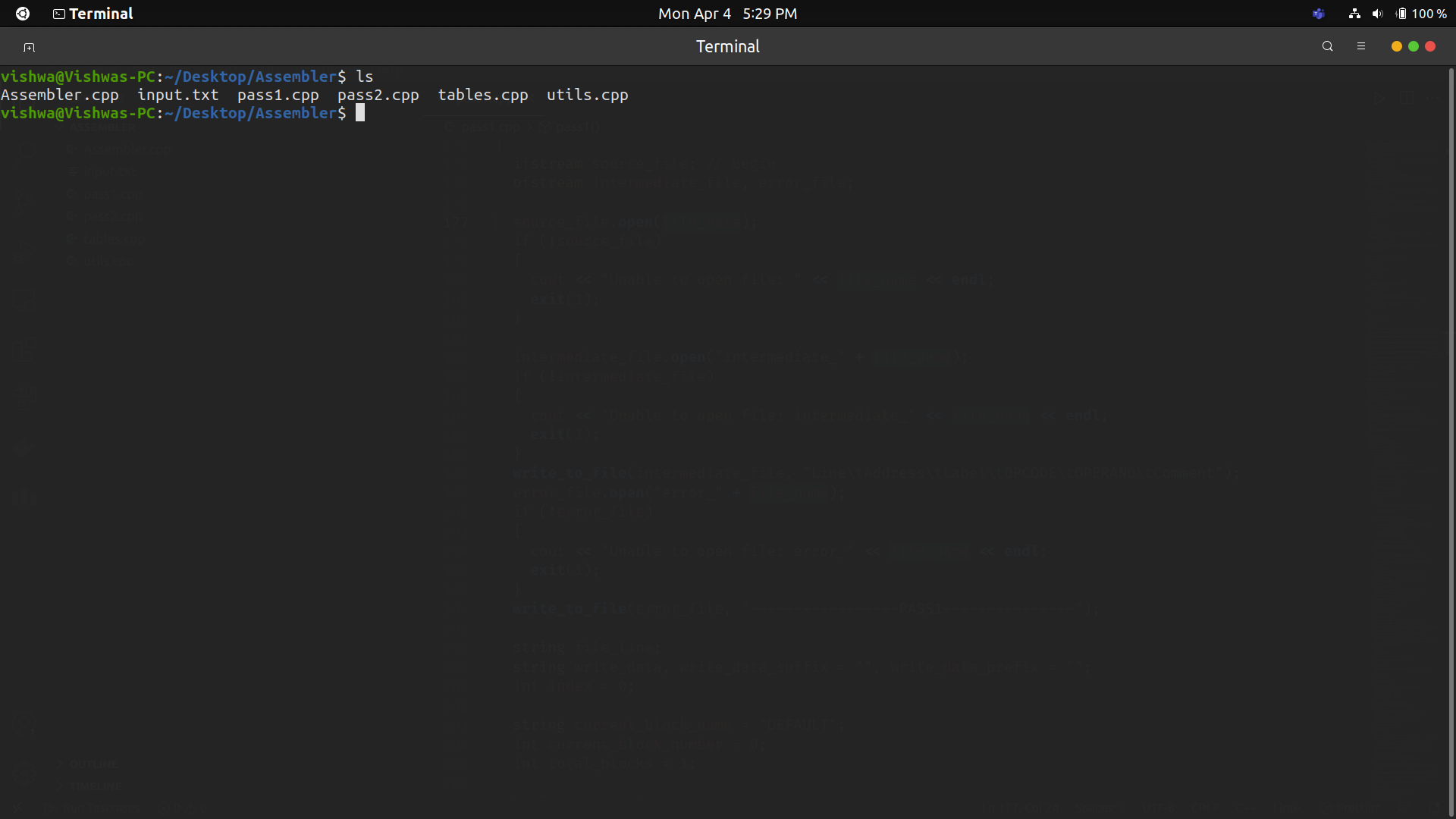
4. An error file is also generated to identify any errors in the assembly program.

**Instructions to run this assembler:**

The assembler consists of of 5 files:

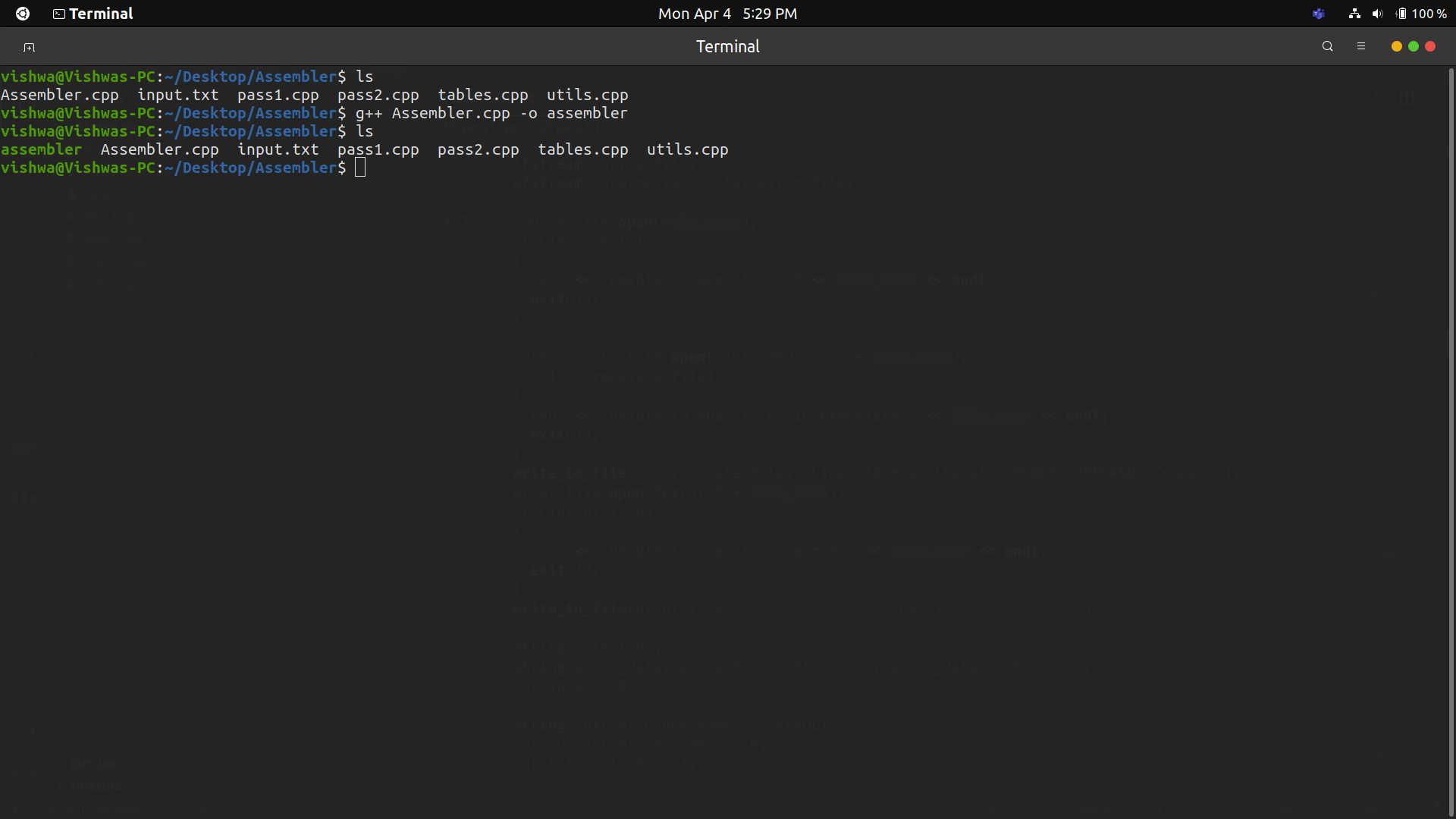
1. Assembler.cpp(main)
2. pass1.cpp
3. pass2.cpp
4. tables.cpp
5. Utils.cpp

Now, first check all files are present by ‘ls’ command. There must be a C++ compiler(for example GNU g++) installed on the machine to compile the files.

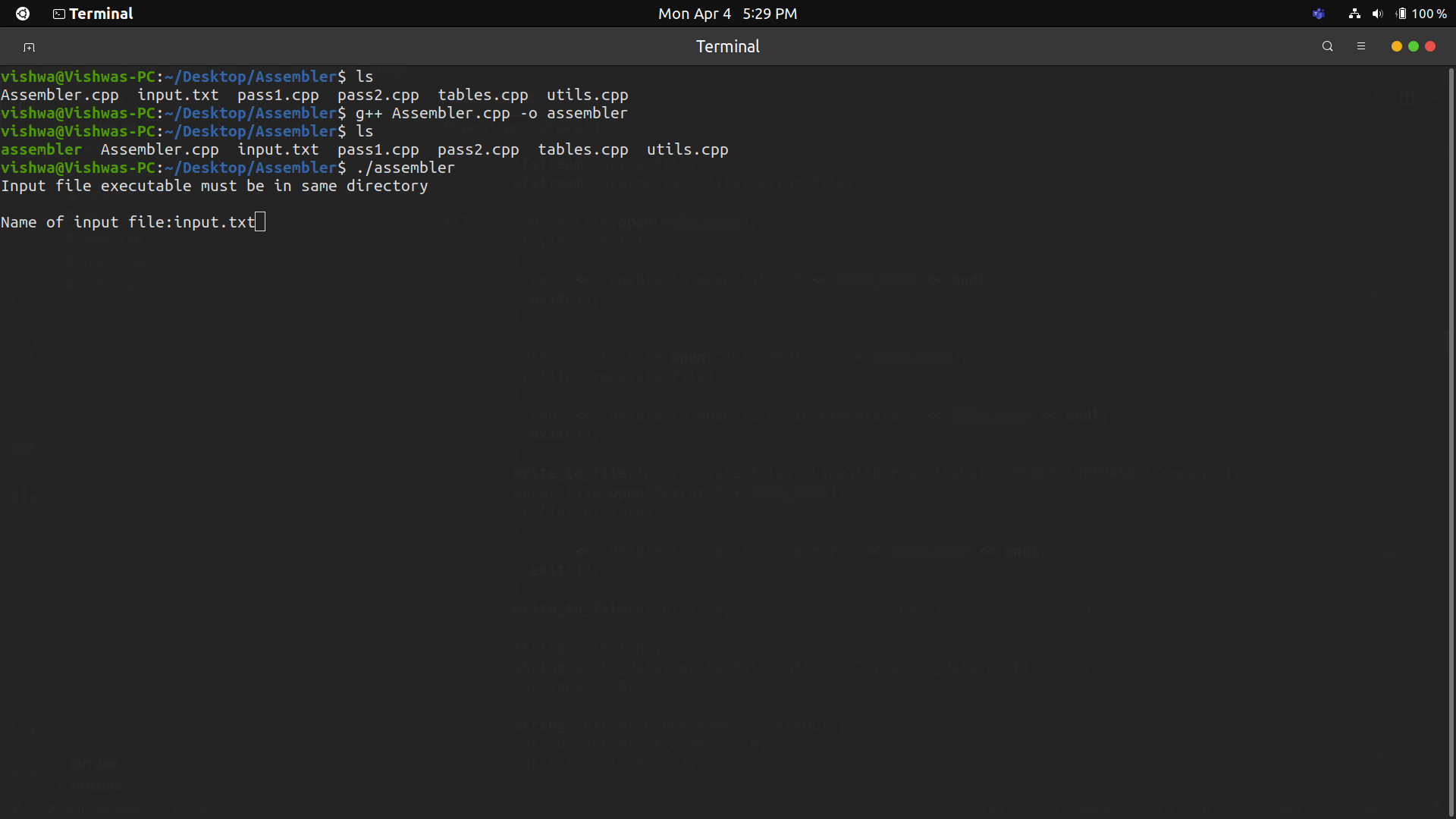


Now, compile the main file - ‘Assembler.cpp’ by using command:

g++ Assembler.cpp -o <name\_of\_executable>



Now, we see there is a ‘assembler’ file which is executable. Run it by using ./<name\_of\_executable> and enter the name of file which contains the input instructions.



Now, we see the few files are generated, namely:

1. Error file: contains all the errors in both passes.
2. Intermediate file: contains the immediate input.
3. Object file: contains the object code.
4. Listing file: contains listing file.

