# Assembler Report

Sunidhi Yadav
20114095

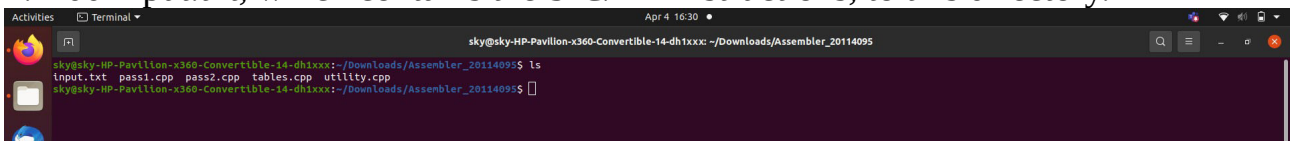
**OBJECTIVE:**

The objective of this project is to implement a two pass assembler for the SIC/XE architecture.

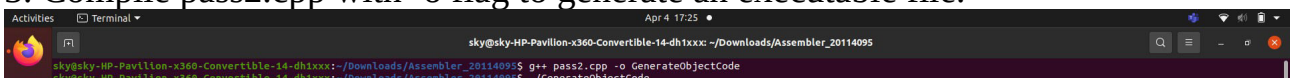
**STEPS:**

1. Save all the four files pass1.cpp, pass2.cpp, tables.cpp, utilities.cpp into one directory.

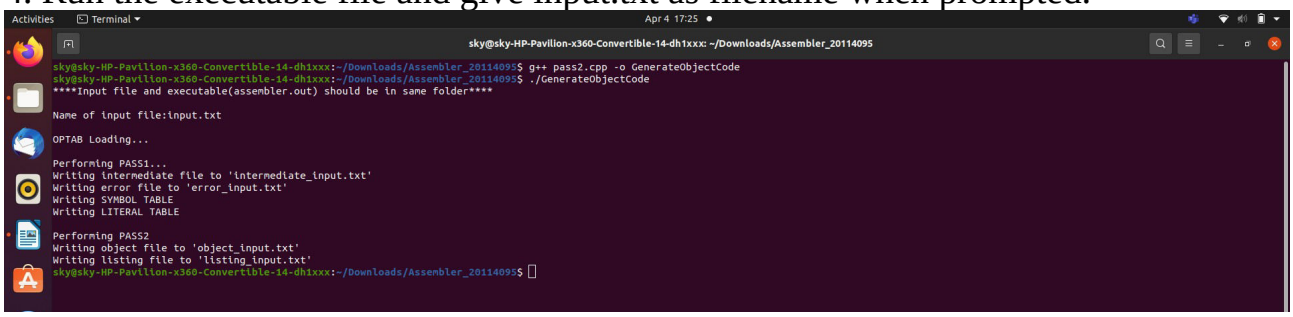

2. Add input.txt, which contains the SIC/XE instructions, to this directory.




3. Compile pass2.cpp with -o flag to generate an executable file.

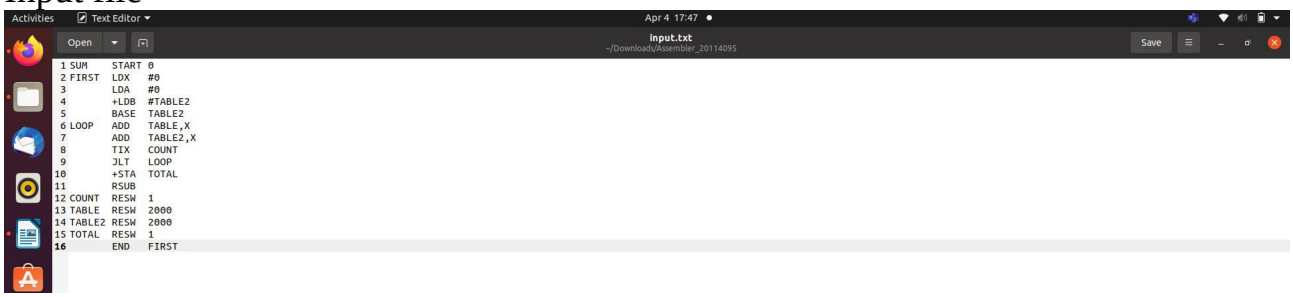



4. Run the executable file and give input.txt as filename when prompted.

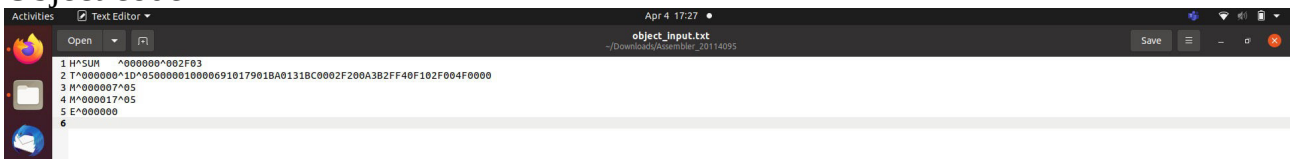



5. The object code is generated and saved in the file object_input.txt.
The file error_input.txt is also generated which reports any irregularity which may have been present in the instructions contained in the input file.

Input file-

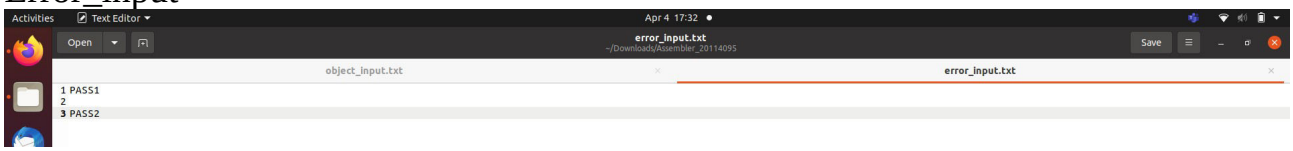


Object code-

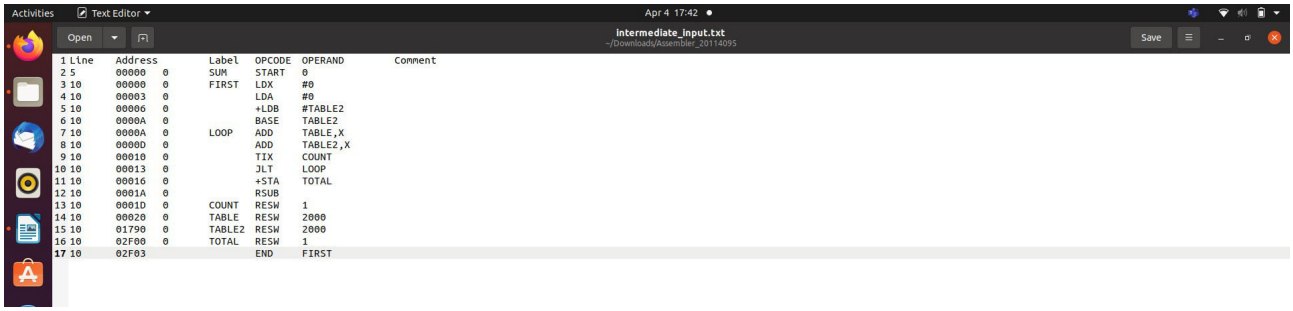



H^SUM ^000000^002F03
T^000000^1D^05000001000069101790 1BA0131BC0002F200A3B2FF40F102F004F0000
M^000007^05
M^000017^05
E^000000

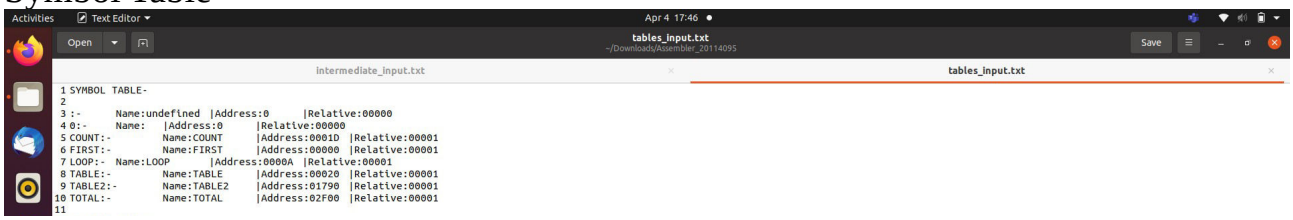
Error_input-




Intermediate file




Symbol Table-




**DETAILS OF THE ASSEMBLER:**

The assembler will support-

**All 4 instruction formats-**

1. Format 1 (1 byte)
   all 8 bits for opcode
2. Format 2 (2 bytes)
   8 bits for opcode, 4 for r1, 4 for r2
3. Format 3 (3 bytes)

6 bits for opcode, 1 bit for n, 1 bit for i, 1 bit for x, 1 bit for b, 1 bit for p, 1 bit for e, 12 bits for displacement

4. Format 4 (4 bytes)

6 bits for opcode, 1 bit for n, 1 bit for i, 1 bit for x, 1 bit for b, 1 bit for p, 1 bit for e, 20 bits for displacement

**All the various addressing modes-**

1. Base Relative:

n=1, i=1, b=1, p=0

2. Program-counter relative:

n=1, i=1, b=0, p=1

3. Direct:

n=1, i=1, b=0, p=0

4. Immediate:

n=0, i=1, x=0

5. Indirect:

n=1, i=0, x=0

6. Indexing:

Both n and i = 0 or 1, x=1

7. Extended:

e=0 for format 3, e=1 for format 4

**FEATURES:**

1. Literals
2. Symbol defining statements
3. Expressions
4. Program Blocks

We give input.txt as the input to the assembler. This file contains the machine instructions which the assembler converts into object code.

**EXECUTION:**

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 and 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.