

Assembler Report

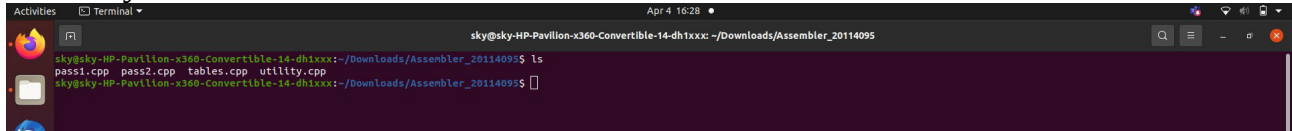
Sunidhi Yadav
20114095

OBJECTIVE:

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

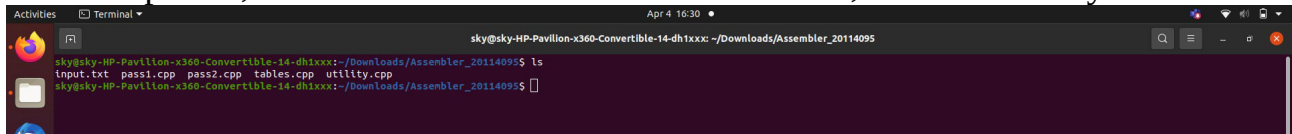
STEPS:

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



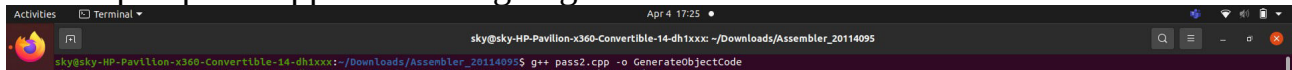
```
sky@sky-HP-Pavilion-x360-Convertible-14-dh1xxx: ~/Downloads/Assembler_20114095$ ls
pass1.cpp  pass2.cpp  tables.cpp  utility.cpp
sky@sky-HP-Pavilion-x360-Convertible-14-dh1xxx: ~/Downloads/Assembler_20114095$
```

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



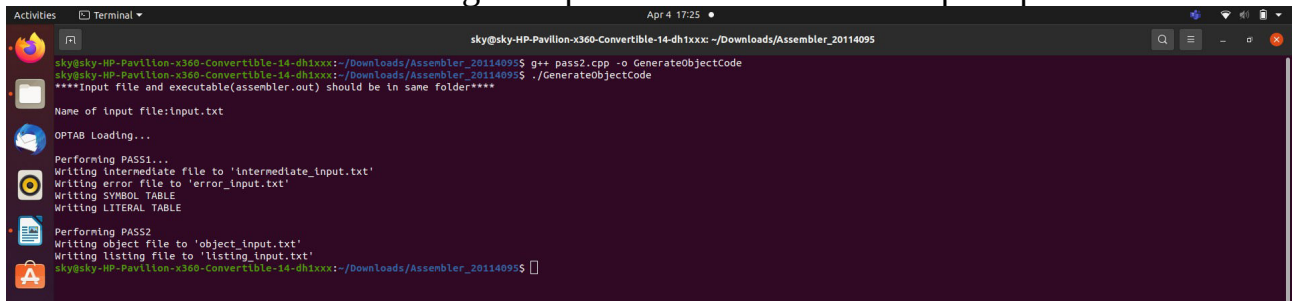
```
sky@sky-HP-Pavilion-x360-Convertible-14-dh1xxx: ~/Downloads/Assembler_20114095$ ls
input.txt  pass1.cpp  pass2.cpp  tables.cpp  utility.cpp
sky@sky-HP-Pavilion-x360-Convertible-14-dh1xxx: ~/Downloads/Assembler_20114095$
```

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



```
sky@sky-HP-Pavilion-x360-Convertible-14-dh1xxx: ~/Downloads/Assembler_20114095$ g++ pass2.cpp -o GenerateObjectCode
sky@sky-HP-Pavilion-x360-Convertible-14-dh1xxx: ~/Downloads/Assembler_20114095$
```

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

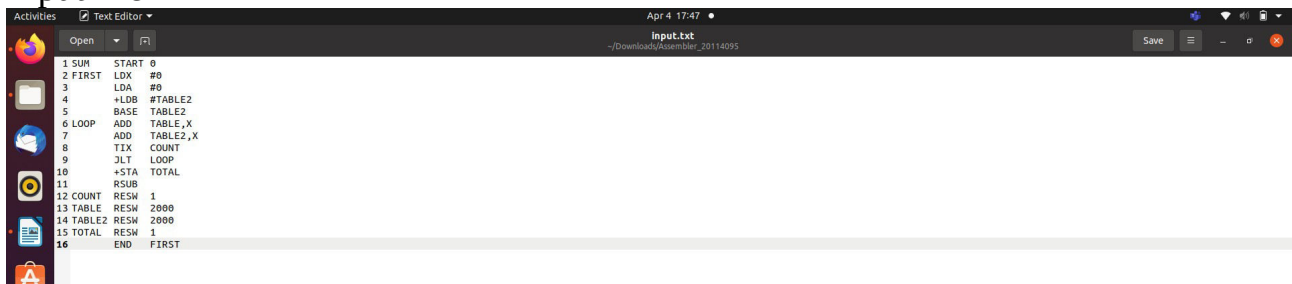


```
sky@sky-HP-Pavilion-x360-Convertible-14-dh1xxx: ~/Downloads/Assembler_20114095$ g++ pass2.cpp -o GenerateObjectCode
sky@sky-HP-Pavilion-x360-Convertible-14-dh1xxx: ~/Downloads/Assembler_20114095$ ./GenerateObjectCode
****Input file and executable(Assembler.out) should be in same folder****
Name of input file:input.txt
OPTAB Loading...
Performing PASS1...
Writing intermediate file to 'intermediate_input.txt'
Writing error file to 'error_input.txt'
Writing SYMBOL TABLE
Writing LITERAL TABLE
Performing PASS2
Writing object file to 'object_input.txt'
Writing listing file to 'listing_input.txt'
sky@sky-HP-Pavilion-x360-Convertible-14-dh1xxx: ~/Downloads/Assembler_20114095$
```

5. The object code is generated and saved in the file object_input.txt.

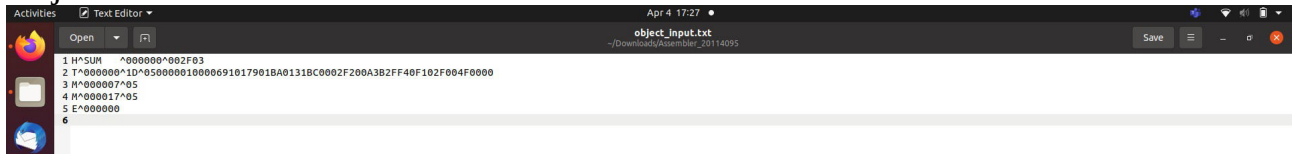
The file error_input.txt is also generated which reports any irregularity which may have been present in the instructions contained in the input file.

Input file-



```
1 SUM      START 0
2 FIRST    LDX   #0
3          LDA   #0
4          +LDB  #TABLE2
5          BASE  TABLE2
6 LOOP     ADD   TABLE,X
7          ADD   TABLE2,X
8          TLX   COUNT
9          JLT   LOOP
10         +STA  TOTAL
11         RSUB
12 COUNT    RESW  1
13 TABLE   RESW 2000
14 TABLE2  RESW 2000
15 TOTAL    RESW 1
16          END   FIRST
```

Object code-



A screenshot of a text editor window titled 'object_input.txt'. The editor shows six lines of hexadecimal object code. The first line is '1 H^SUM ^000000^002F03', the second is '2 T^000000^1D^050000010000691017901BA0131BC0002F200A3B2FF40F102F004F0000', the third is '3 M^000007^05', the fourth is '4 M^000017^05', the fifth is '5 E^000000', and the sixth is '6'.

H^SUM ^000000^002F03

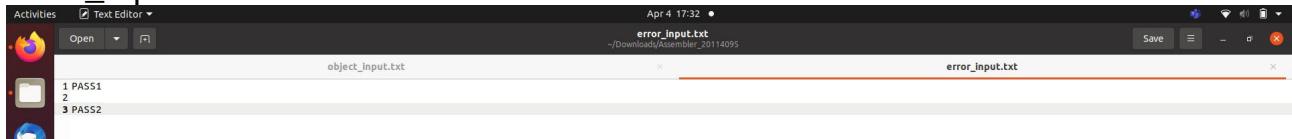
T^000000^1D^050000010000691017901BA0131BC0002F200A3B2FF40F102F004F0000

M^000007^05

M^000017^05

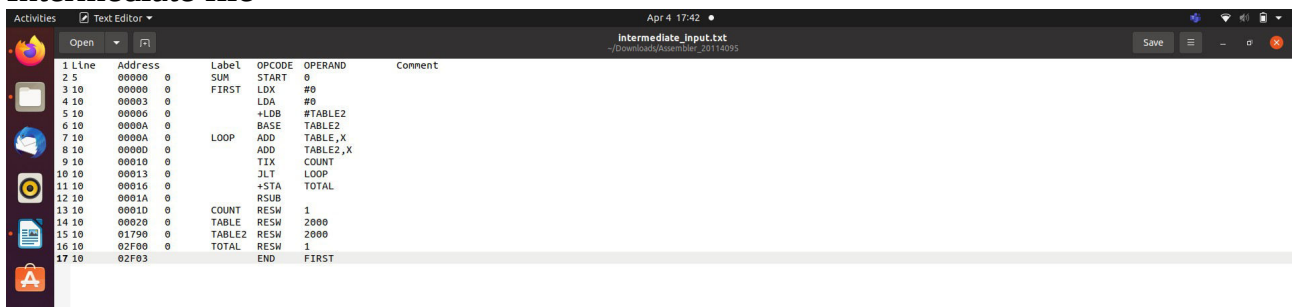
E^000000

Error input-



A screenshot of a text editor window titled 'error_input.txt'. The editor shows three lines of text: '1 PASS1', '2', and '3 PASS2'.

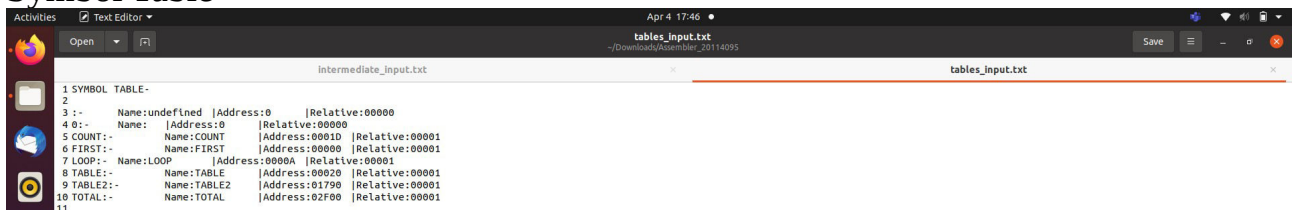
Intermediate file



A screenshot of a text editor window titled 'intermediate_input.txt'. The editor shows a table of assembly instructions. The table has columns for Line, Address, Label, Opcode, Operand, and Comment. The instructions are as follows:

Line	Address	Label	Opcode	Operand	Comment
2 5	00000	0	SUM	START	0
3 10	00000	0	FIRST	LDA	#0
4 10	00003	0	LDA	#0	
5 10	00006	0		+LDB	#TABLE2
6 10	0000A	0		BASE	TABLE2
7 10	0000A	0		ADD	TABLE2,X
8 10	0000D	0		ADD	TABLE2,X
9 10	00010	0		TIX	COUNT
10 10	00013	0		JLT	LOOP
11 10	00016	0		+STA	TOTAL
12 10	0001A	0		RSUB	
13 10	0001D	0	COUNT	RESW	1
14 10	00020	0	TABLE	RESW	2000
15 10	01790	0	TABLE2	RESW	2000
16 10	02F00	0	TOTAL	RESW	1
17 10	02F03	0	TOTAL	END	FIRST

Symbol Table-



A screenshot of a text editor window titled 'tables_input.txt'. The editor shows a symbol table with columns for Name, Address, and Relative. The symbols are as follows:

Line	Name	Address	Relative
1	SYMBOL	TABLE-	
2			
3 :-	Name: undefined	Address:0	Relative:00000
4 0:-	Name:	Address:0	Relative:00000
5 COUNT:-	Name: COUNT	Address:0001D	Relative:00001
6 FIRST:-	Name: FIRST	Address:00000	Relative:00001
7 LOOP:-	Name: LOOP	Address:0000A	Relative:00001
8 TABLE:-	Name: TABLE	Address:00020	Relative:00001
9 TABLE2:-	Name: TABLE2	Address:01790	Relative:00001
10 TOTAL:-	Name: TOTAL	Address:02F00	Relative:00001
11			

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.