Experiment No.	: **02**
Experiment Title	**For any Assembly input for a hypothetical machine, implement pass 1 by displaying SYMTAB, LITAB and POOLTAB and intermediate code.**
Student Name	**Ronak Surve**
Roll No.	**64**
Objectives :	**1) Students will be able to learn and identify the mnemonics, Pseudo opcodes and symbols in an assembly language program.** **2) Students will be able to implement the working of Pass 1 of 2 pass Assembler**
Theory /Algorithm :	Assembler is a program for converting instructions written in low-level assembly code into relocatable machine code and generating along information for the loader. It generates instructions by evaluating the mnemonics (symbols) in operation field and find the value of symbol and literals to produce machine code. Now, if assembler do all this work in one scan then it is called single pass assembler, otherwise if it does in multiple scans then called multiple pass assembler. Here assembler divide these tasks in two passes: • **Pass-1:** 1. Define symbols and literals and remember them in symbol table and literal table respectively. 2. Keep track of location counter 3. Process pseudo-operations • **Pass-2:** 1. Generate object code by converting symbolic op-code into respective numeric op-code 2. Generate data for literals and look for values of symbols
Program Code:	see below

```
#import null as null

IS = ["MOVER", "MOVEM", "ADD", "SUB", "MULT", "DIV", "BC",
"COMP", "READ", "PRINT"]
POT = ["START", "END", "EQU", "ORIGIN", "LTORG"]
DL = ["DS", "DC"]
LC = 00
REGISTERS = ["AREG", "BREG", "CREG", "DREG"]
SYMBOLS = ['A', 'B', 'C', 'D', 'NUM', 'LOOP']

f1 = open("input.txt", 'r')
f1 = f1.readlines()
```

```python
f2 = open("output.txt", 'w')
x = 0
for i in range(len(f1)):
    f1[i] = f1[i].split()
    print(f1[i])
f2.write("LC\t\tOPCODE\t\tOP1\t\tOP2\n")
for i in range(len(f1)):
    if (len(f1[i]) == 2):
        if (f1[i][0]) in POT:
            LC = str(f1[i][1])
            f2.write(LC)
            LC = int(LC) + 1
            x = LC

    if (len(f1[i]) == 3):
        if (f1[i][0]) in SYMBOLS:
            f2.write("\n")
            f2.write(str(x))
            x = x + 1
            f2.write("\t\t")
            f2.write("( " + "DL," +
str(SYMBOLS.index(f1[i][0]) + 1) + " )")
            f2.write("\t")
            if (f1[i][1]) in DL:
                f2.write(str(DL.index(f1[i][1]) + 1))
            else:
                f2.write(str(REGISTERS.index(f1[i][1]) +
1))
            f2.write("\t\t")
            f2.write(str(f1[i][2]))
            f2.write("\n")

    if (len(f1[i]) == 3):
        if (f1[i][0]) in IS:
            f2.write("\n")
            f2.write(str(x))
            x = x + 1
            f2.write("\t\t")
            f2.write("( " + "IS," + str(IS.index(f1[i][0])
+ 1) + " )")
            f2.write("\t")
            # #f2.write(f1[i][1])
            # if (f1[i][1]) in REGISTERS:
            #     f2.write(str(DL.index(f1[i][1]) + 1))
            # else:
            #     f2.write(str(REGISTERS.index(f1[i][1]) +
1))

            f2.write("\t\t")
            f2.write(str(f1[i][2]))
            f2.write("\n")
```

Input to the Program:	**Input.txt** START 100 A DC 10

```
   MOVER AREG, B
   MOVEM BREG, ='1'
   ADD AREG, ='2'
   SUB BREG, ='1'
B  DC 20
   ORIGIN 300
   LTORG
   MOVER  AREG,  NUM
   MOVER CREG, LOOP
   ADD BREG, ='1'
NUM DS 5
LOOP  DC  10
END
```

Output of the program:

Output.txt

```
 LC            OPCODE              OP1           OP2
100
101            ( DL,1 )            2             10

102            ( IS,1 )                           B

103            ( IS,2 )                          ='1'

104            ( IS,3 )                          ='2'

105            ( IS,4 )                          ='1'

106            ( DL,2 )            2             20
300
301            ( IS,1 )                         NUM

302            ( IS,1 )                         LOOP

303            ( IS,3 )                         ='1'

304            ( DL,5 )           1               5

305            ( DL,6 )           2               1
```

Outcome of the Experiment:	Assembler is a program for converting instructions written in low-level assembly code into relocatable machine code and generating along information for the loader.
References:	https://www.geeksforgeeks.org/introduction-of-assembler/