LPCC Assignment 1-C

Name: Digvijay Pawar
Class: T.Y Btech Comp B
GR no.: 21810344
Roll no.: 322043

Aim: Generate Symbol table, Literal table, Pool table & Intermediate code of a two-pass Assembler for the given source code.
1-c: Generate literal table and pool table from given assembly code.

Objective:
1. To generate literal table and pool table
2. To understand the working of two-pass Assembler

Theory:

Literal Table:

- A literal table is created for the literals which are used in the program.
- The literal table contains the literal name, operand value and length.
- The literal table is usually created as a hash table on the literal name.

Pool Table:

- Awareness of different literal pools is maintained using the auxiliary table POOLTAB.
- This table contains the literal number of the starting literal of each literal pool.
- At any stage, the current literal pool is the last pool in LITTAB.
- On encountering an LTORG statement (or the END statement), literals in the current pool are allocated addresses starting with the current value in LC and LC is appropriately incremented.
- Number of entries in pool table is equal to no of LTORG instruction in the program + 1.

Program:

1C.py :

```
import pandas as pd
import re

literal = dict()
var1 = list()
sym = dict()
Loc_Count = 0

re_lit = re.compile(r'=[0-9]')

f_in = open('Task.txt','r')

for line in f_in:
    line.strip()
    words = line.split()

    if line.startswith('START'):
        Loc_Count = int(words[-1])
        continue

    if len(words)>3 :
        sym[str(words[0])] = Loc_Count

    if 'DC' in line:
        sym[str(words[0])] = Loc_Count

    if re_lit.search(line):
        var1.append(str(words[-1]))
        literal[str(words[-1])] = 0

    if line.startswith('END'):
        for w in var1:
            if literal.get(w)==0:
                literal[w] = Loc_Count
                Loc_Count += 1

    if 'DS' in line:
        sym[str(words[0])] = Loc_Count
        Loc_Count += int(words[-1])
        continue
```

```
    if line.startswith('ORIGIN'):
        sub = words[-1].split('+')
        if sub[0] in sym.keys():
            Loc_Count = sym[str(sub[0])] + int(sub[1])
        continue

    if 'EQU' in line:
        if words[0] not in sym.keys():
            symb[str(words[0])] = sym[str(words[-1])]

    if 'LTORG' in line:
        for w in var1:
            literal[w] = Loc_Count
            Loc_Count += 1
        continue

    Loc_Count += 1

lit_table = pd.DataFrame(list(literal.items()),columns=['Literal','Address'])
print(lit_table)

pool = literal.values()

p_table = list()
p_table.append('#1')

counter = list(pool)[0]
cnt = 1
for i in pool:
    if i-counter>1:
        temp='#'+str(cnt)
        p_table.append(temp)
    cnt+=1
    counter = i

p_table = pd.DataFrame(list(p_table),columns=['Pool Table'])
print(p_table)
```

Input File :

Task.txt:

```
START 200
MOVER AREG =6
```

MOVER BREG X
L1 MOVER BREG =2
LTORG
NEXT ADD AREG =3
X DS 1
END

Output: