LPCC Assignment 1-D

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-d: Generate Symbol Table (ST), Literal Table (LT) and Intermediate code(IC) for the assembly language program.

**Objective:**

1. To generate symbol table, literal table, pool table and Intermediate code.

2. To understand the working of two-pass Assembler.

**Theory:**

**Intermediate Code:**

Intermediate code consists of a set of IC units, each unit consisting of the following three fields:

1. Address

2. Representation of mnemonics opcode

3. Representation of operands

Intermediate code can be in variant I or variant II form the literal table is usually created as a hash table on the literal name.

**Program:**

import pandas as pd

import re

literal = dict()

var1 = list()

symbol = dict()

LocCount = 0

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

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

for line in tfile:

line.strip()

words = line.split()

if line.startswith('START'):

LocCount = int(words[-1])

continue

if len(words)>3 :

symbol[str(words[0])] = LocCount

if 'DC' in line:

symbol[str(words[0])] = LocCount

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] = LocCount

LocCount += 1

if 'DS' in line:

symbol[str(words[0])] = LocCount

LocCount += int(words[-1])

continue

if line.startswith('ORIGIN'):

sub = words[-1].split('+')

if sub[0] in symbol.keys():

LocCount = symbol[str(sub[0])] + int(sub[1])

continue

if 'EQU' in line:

if words[0] not in symbol.keys():

symbol[str(words[0])] = symbol[str(words[-1])]

if 'LTORG' in line:

for w in var1:

literal[w] = LocCount

LocCount += 1

continue

LocCount += 1

symbol\_tb = pd.DataFrame(list(symbol.items()),columns=['Symbol','Address'])

print(symbol\_tb)

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

print(literal\_tb)

pool = literal.values()

pool\_table = list()

pool\_table.append('#1')

counter = list(pool)[0]

cnt = 1

for i in pool:

if i-counter>1:

temp='#'+str(cnt)

pool\_table.append(temp)

cnt+=1

counter = i

pool\_table = pd.DataFrame(list(pool\_table),columns=['Pool Table'])

print(pool\_table)

optable = {

'START':"('AD',01)",

'END':"('AD',02)",

'LTORG':"('AD',05)",

'ORIGIN':"('AD',03)",

'EQU':"('AD',04)",

'DC':"('DL', 01)",

'DS':"('DL', 02)",

'ADD':"('IS', 01)",

'SUB':"('IS', 02)",

'MOVER':"('IS', 04)",

'MOVEM':"('IS', 05)",

'READ':"('IS', 09)",

'PRINT':"('IS', 10)"

}

register = ['AREG','BREG']

re\_constant = re.compile('[0-9]+')

interLine = list()

count = 1

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

for line in fhand:

line.strip()

words = line.split()

flag=0

if words[0] in optable.keys():

if words[0]=='ORIGIN':

tup = list()

tup.append(words[0])

tup.append(words[1])

interLine.append(tup)

flag=1

tup = list()

tup = optable[words[0]]

interLine.append(tup)

elif words[0] in symbol.keys() and flag!=1:

if words[1] in optable.keys():

tup = list()

tup = optable[words[1]]

interLine.append(tup)

if "AREG" in words and flag!=1:

tup = list()

tup.append("AREG")

interLine.append(tup)

if "BREG" in words and flag!=1:

tup = list()

tup.append("BREG")

interLine.append(tup)

if words[-1] in symbol.keys() and flag!=1:

tup = list()

tup.append(words[-1])

interLine.append(tup)

if words[-1] in literal.keys() and flag!=1:

tup = list()

tup.append("L,")

tup.append("0"+str(count))

count+=1

interLine.append(tup)

if words[-1][0] != "=" and flag!=1:

if re\_constant.search(words[-1][0]):

tup = list()

tup.append("C,")

tup.append(words[-1])

interLine.append(tup)

interLine.append("NEWLINE")

print("Intermediate Code : ")

for i in range(len(interLine)):

if interLine[i]=="NEWLINE":

print("")

continue

print(interLine[i]),

**Input File:**

START 200

MOVER AREG =6

MOVER BREG X

L1 MOVER BREG =2

LTORG

NEXT ADD AREG =3

X DS 1

END

**Output:**

