SP ESE LAB:

------------------------------

Name: Yash Oswal

Div: B Roll no: 38

SRN: 201901226

------------------------------

ALP:

BEGIN 800

NEXT RD Sum

MVR R1, N

MVM R1, Sum

AD R1, R2

ML R1, X

JP NEXT

X DCN 4

Sum DST 5

N DCN 10

STOP

SYMBOL TABLE:

NEXT 800

X 814

Sum 815

N 820

OUTPUT WITH LC:

BEGIN 800

800 NEXT RD Sum

802 MVR R1 N

804 MVM R1, Sum

807 AD R1 R2

809 ML R1 X

811 JP NEXT

814 X DCN 4

815 Sum DST 5

820 N DCN 10

821 STOP

ERROR CHECKING:

Checking line 1 for errors

['BEGIN', '800']

[+] No errors at line 1

Checking line 2 for errors

['NEXT', 'RD', 'Sum']

Checking line 3 for errors

['MVR', 'R1,', 'N']

[+] No errors at line 3

Checking line 4 for errors

['MVM', 'R1,', 'Sum']

Checking line 5 for errors

['AD', 'R1,', 'R2']

[+] No errors at line 5

Checking line 6 for errors

['ML', 'R1,', 'X']

[+] No errors at line 6

Checking line 7 for errors

['JP', 'NEXT']

Checking line 8 for errors

['X', 'DCN', '4']

[+] No errors at line 8

Checking line 9 for errors

['Sum', 'DST', '5']

[+] No errors at line 9

Checking line 10 for errors

['N', 'DCN', '10']

[+] No errors at line 10

Checking line 11 for errors

['STOP']

yashoswal@blackdex:~/Documents/TY-Assignments/SP$ python ese.py

Checking line 1 for errors

['BEGIN', '800']

[+] No errors at line 1

Checking line 2 for errors

['NEXT', 'RD', 'Sum']

Checking line 3 for errors

['MVR', 'R1,', 'N']

[+] No errors at line 3

Checking line 4 for errors

['MVM', 'R1,', 'Sum']

Checking line 5 for errors

['AD', 'R1,', 'R2']

[+] No errors at line 5

Checking line 6 for errors

['ML', 'R1,', 'X']

[+] No errors at line 6

Checking line 7 for errors

['JP', 'NEXT']

Checking line 8 for errors

['X', 'DCN', '4']

[+] No errors at line 8

Checking line 9 for errors

['Sum', 'DST', '5']

[+] No errors at line 9

Checking line 10 for errors

['N', 'DCN', '10']

[+] No errors at line 10

Checking line 11 for errors

['STOP']

SOURCE CODE (python3):-

from io import TextIOWrapper

MOT={

'BEGIN' : ('#R1', 'AD', 1),

'STOP' : ('#R2', 'AD', 0),

'ORIGIN': ('#R3', 'AD', 0),

'MVR' : ('01', 'IS', 2),

'MVM' : ('02', 'IS', 3),

'AD' : ('03', 'IS', 2),

'RD' : ('04', 'IS', 2),

'SB' : ('05', 'IS', 2),

'JP' : ('06', 'IS', 3),

'ML' : ('07', 'IS', 2),

'DCN' : ('#R4', 'DL', 1),

'DST' : ('#R5', 'DL', 1),

}

REG={

'R1':1,

'R2':2,

'R3':3,

'R4':4

}

class vars():

LC=0

opt=open("LC\_Code.txt",mode="a+")

opt.truncate(0)

symtab={}

words=[]

symindex=0

def listToString(s):

str1 = " "

return (str1.join(s))

def STOP():

vars.opt.write(f"\t{listToString(vars.words)}\n")

def ORIGIN(addr):

vars.opt.write(f"\t{listToString(vars.words)}\n")

vars.LC =int(addr)

def DS(size):

vars.opt.write(f"\t{listToString(vars.words)}\n")

vars.LC=vars.LC+int(size)

def DC(value):

vars.opt.write(f"\t{listToString(vars.words)}\n")

vars.LC+=1

def JP():

vars.opt.write(f"\t{listToString(vars.words)}\n")

vars.LC+=3

def RD():

vars.opt.write(f"\t{listToString(vars.words)}\n")

vars.LC+=2

def MVM():

vars.opt.write(f"\t{listToString(vars.words)}\n")

vars.LC+=3

def OTHERS(key,k):

z=MOT[key]

i=0

y=z[-1]

for i in range(1,y+1):

vars.words[k+i]=vars.words[k+i].replace(",","")

if(vars.words[k+i] in REG.keys()):

vars.opt.write(f"\t{listToString(vars.words)}\n")

vars.LC+=z[-1]

return

else:

if(vars.words[k+i] not in vars.symtab.keys()):

vars.symtab[vars.words[k+i]]=("\*\*",vars.symindex)

vars.opt.write(f"\t{listToString(vars.words)}\n")

vars.symindex+=1

vars.LC+=z[-1]

def detect\_mn(k):

if(vars.words[k]=="BEGIN"):

vars.LC = int(vars.words[1])

vars.opt.write(f"\t{listToString(vars.words)}\n")

elif(vars.words[k]=='STOP'):

STOP()

elif(vars.words[k]=="ORIGIN"):

ORIGIN(vars.words[k+1])

elif(vars.words[k]=="DST"):

DS(vars.words[k+1])

elif(vars.words[k]=="DCN"):

DC(vars.words[k+1])

elif(vars.words[k]=="JP"):

JP()

elif(vars.words[k]=="RD"):

RD()

elif(vars.words[k]=="MVM"):

MVM()

else:

OTHERS(vars.words[k],k)

def pass\_one(alp:TextIOWrapper):

lc=1

for line in alp:

error\_handler(line,lc)

lc+=1

vars.words=line.split()

if (vars.LC>0):

vars.opt.write(str(vars.LC))

k=0

if vars.words[0] in MOT.keys():

val = MOT[vars.words[0]]

detect\_mn(k)

else:

if vars.words[k] not in vars.symtab.keys():

vars.symtab[vars.words[k]]=(vars.LC,vars.symindex)

vars.symindex+=1

else:

x = vars.symtab[vars.words[k]]

if x[0] == "\*\*":

vars.symtab[vars.words[k]] = (vars.LC,x[1])

k=1

detect\_mn(k)

vars.opt.close()

sym=open("symbol\_table.txt","a+")

sym.truncate(0)

for x in vars.symtab:

sym.write(x+"\t"+str(vars.symtab[x][0])+"\n")

sym.close()

def error\_handler(line:str,lc:int):

print(f"\nChecking line {lc} for errors")

l=line.split()

print(l)

try:

if l[0] == 'JP' or l[1] == 'RD' or l[0] == 'MVM':

return

except IndexError:

return

if l[0] in MOT.keys():

op = MOT[l[0]]

if (len(l)-1) < op[-1]:

print(f"[-] Error at line {lc}: Less operands than expcted")

exit(-1)

elif (len(l)-1) > op[-1]:

print(f"[-] Error at line {lc}: More operands than expcted")

exit(-1)

else:

print(f"[+] No errors at line {lc}")

elif l[1] in MOT.keys():

op = MOT[l[1]]

if (len(l)-2) < op[-1]:

print(f"[-] Error at line {lc}: Less operands than expcted")

exit(-1)

elif (len(l)-2) > op[-1]:

print(f"[-] Error at line {lc}: More operands than expcted")

exit(-1)

else:

print(f"[+] No errors at line {lc}")

else:

print(f"[-] Invalid Instruction at line {lc}: {line}")

exit(-1)

def getFile():

alp = open('ese.asm','r')

return alp

if \_\_name\_\_=='\_\_main\_\_':

alp=getFile()

pass\_one(alp)