# ANAY DOMBE

#TY\_IT\_16

#GR NO:151666

# ASSEMBLER

MOT={"Mnemonic":["Add","Sub","Mult","Jmp","Jneg","Jpos","Jz","Load","Store","Read","Write","Stop"],"Opcode":[1,2,3,4,5,6,7,8,9,10,11,12],"No\_operands":[1,1,1,1,1,1,1,1,1,1,1,0],"Length":[2,2,2,2,2,2,2,2,2,2,2,1]}

POT={"Pseudo":["Db","Dw","Org","ENDP","Const","End"],"No\_operands":[2,2,1,1,1,0],"Length":[1,1,1,1,1,1]}

Symbol\_Table={"Symbol":[],"Address":[]}

file=open("assembler\_input","r")

loc\_counter=0

temp=0

line=file.readline()

while line:

strings=line.split()

for word in strings:

#\*\*\*\*\*\*\* CHECK IF WORD IS PRESENT IN MOT \*\*\*\*\*\*\*

for var in MOT["Mnemonic"]:

if var==word:

index=MOT["Mnemonic"].index(word) # finding the index of word in Mnemonic

loc\_counter+=MOT["Length"][index]

flag=0

break

if flag==0: # word is found in Mnemonic

flag=1

continue

elif flag==1 and word=="Loop:" or flag==1 and word=="Outer:":

temp2=1

for var in Symbol\_Table["Symbol"]:

if var==word: # checking if word is already present in symbol table

temp2=0

break

if temp2==1:

Symbol\_Table["Symbol"].append(word)

idx=Symbol\_Table["Symbol"].index(word)

Symbol\_Table["Address"].append(loc\_counter)

elif flag==1 and word!="Endp" and word!="Outer" and word!="Loop": # to distinguish from labels

temp1=1

for var in Symbol\_Table["Symbol"]:

if var==word: # checking if word is already present in symbol table

temp1=0

break

if temp1==1:

Symbol\_Table["Symbol"].append(word)

Symbol\_Table["Address"].append(0)

elif word=="Endp":

temp=1 # to indicate end of code

address=loc\_counter

print "\*\*\*\*\*\*\*\*\* AFTER FIRST PASS STRUCTURE OF SYMBOL TABLE \*\*\*\*\*\*\*\*\*\n"

print Symbol\_Table

break

if temp==1:

line=file.readline()

while line:

strings=line.split()

for word in strings:

for var in Symbol\_Table["Symbol"]:

if word==var:

temp3=0

index=Symbol\_Table["Symbol"].index(word)

break

else:

temp3=1

if temp3==0:

continue # word has been found in Mnemonic and we want to find length of next word in POT

else:

for var in POT["Pseudo"]:

if word==var:

idx=POT["Pseudo"].index(word)

Symbol\_Table["Address"][index]=address # first address is stored,then loc\_counter++

address+=POT["Length"][idx]

line=file.readline()

print "\*\*\*\*\*\*\*\*\* AFTER SECOND PASS STRUCTURE OF SYMBOL TABLE \*\*\*\*\*\*\*\*\*\n"

print Symbol\_Table

break

else:

line=file.readline()

file.close()

print "\n"

print "\*\*\*\*\*\*\*\*\* TARGET PROGRAM \*\*\*\*\*\*\*\*\*\n"

file=open("assembler\_input","r")

line=file.readline()

temp1=0

while line:

strings=line.split()

for word in strings:

if word=="Endp":

temp1=1

break

for var in MOT["Mnemonic"]: # check if word is present in Mnemonic

if var==word:

flag=1

index=MOT["Mnemonic"].index(word)

print MOT["Opcode"][index]

break

if flag==1:

flag=0 # word is found in Mnemonic and read next word

continue

elif flag==0:

for var in Symbol\_Table["Symbol"]:

if word==var or word=="Outer" or word=="Loop":

if word=="Outer":

word=word.replace("Outer","Outer:")

elif word=="Loop":

word=word.replace("Loop","Loop:")

elif word=="Outer:" or word=="Loop:":

continue

idx=Symbol\_Table["Symbol"].index(word)

print Symbol\_Table["Address"][idx]

break

if temp1==1:

break

print "\n"

line=file.readline()

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

anay@ubuntu:~/Desktop$ python assembler.py

\*\*\*\*\*\*\*\*\* AFTER FIRST PASS STRUCTURE OF SYMBOL TABLE \*\*\*\*\*\*\*\*\*

{'Symbol': ['N', 'zero', 'count', 'sum', 'Loop:', 'X', 'new', 'new1', 'new3', 'one', 'Outer:'], 'Address': [0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 34]}

\*\*\*\*\*\*\*\*\* AFTER SECOND PASS STRUCTURE OF SYMBOL TABLE \*\*\*\*\*\*\*\*\*

{'Symbol': ['N', 'zero', 'count', 'sum', 'Loop:', 'X', 'new', 'new1', 'new3', 'one', 'Outer:'], 'Address': [41, 37, 40, 39, 8, 42, 43, 44, 45, 38, 34]}

\*\*\*\*\*\*\*\*\* TARGET PROGRAM \*\*\*\*\*\*\*\*\*

10

41

8

37

9

40

9

39

10

42

8

42

1

39

1

43

1

44

1

45

9

39

8

40

1

38

9

40

2

41

7

34

4

8

11

39

12

INPUT

Read N

Load zero

Store count

Store sum

Loop: Read X

Load X

Add sum

Add new

Add new1

Add new3

Store sum

Load count

Add one

Store count

Sub N

Jz Outer

Jmp Loop

Outer: Write sum

Stop

Endp

zero Const

one Const

sum Db

count Db

N Db

X Db

new Db

new1 Db

new3 Db

END