Assignment 2-c

Name: Pranav Wagh

Class: T.Y B

Roll no.: 322061

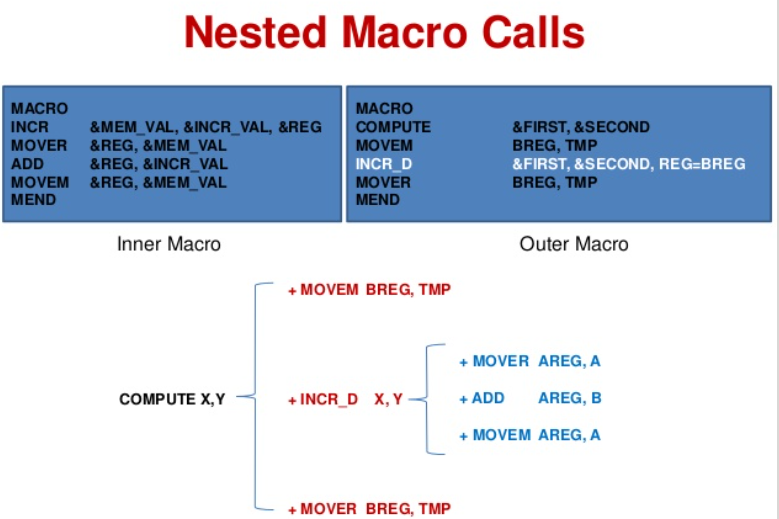
GR no.: 21810340

**Aim:** Design suitable data structures & implement pass-I for a nested macro.

**Objective:** To understand concepts of Nested macro and Nested macro call.

**Theory:**

Writing a macro is another way of ensuring modular programming in assembly language. A

macro is a sequence of instructions, assigned by a name and could be used anywhere in the program. In NASM, macros are defined with %macro and %endmacro directives.

**Program:**

fhand = open('input3.txt', 'r')

output\_code = []

code = {}

parameters = {}

current\_macro = "NULL"

for line in fhand:

line.strip()

dup\_line = line

words=line.split()

if words[0] == "MACRO":

current\_macro = words[1]

param = []

for y in words[2:]:

param.append(y)

code[words[1]] = []

parameters[words[1]] = param

elif words[0]!="MACRO" and current\_macro=="NULL":

output\_code.append(dup\_line)

elif words[0] == "MEND":

code[current\_macro].append(words)

current\_macro = "NULL"

elif words[0] != "MACRO" and current\_macro != "NULL":

code[current\_macro].append(words)

mdt = []

start = {}

i = 1

actual\_pram = {}

def MACRO\_expansion(key,lst):

global i,actual\_pram

values = {}

k = 0

for y in parameters[key]:

values[y] = lst[k]

k = k + 1

for x in code[key]:

if x[0] not in code.keys() and x[0] != "MEND":

n = 0

st1 = x[:]

for element in st1:

if element in parameters[key]:

st1[n] = values[element]

n = n + 1

temp = [i,st1]

mdt.append(temp)

i = i+1

elif x[0] in code.keys():

temp = []

for y in x[1:]:

temp.append(y)

if x[0] not in actual\_pram.keys():

actual\_pram[x[0]] = []

actual\_pram[x[0]].append(temp)

MACRO\_expansion(x[0],temp)

for key in code.keys():

loop = 1

values = {}

for x in parameters[key]:

values[x] = "#" + str(loop)

loop = loop+1

start[key] = i

for x in code[key]:

if x[0] not in code.keys():

n = 0

stmt = x[:]

for element in stmt:

if element in parameters[key]:

stmt[n] = values[element]

n = n + 1

temp = [i,stmt]

mdt.append(temp)

i = i + 1

elif x[0] in code.keys():

temp = []

for y in x[1:]:

temp.append(y)

if x[0] not in actual\_pram.keys():

actual\_pram[x[0]] = []

actual\_pram[x[0]].append(temp)

MACRO\_expansion(x[0],temp)

for line in output\_code:

line = line.replace(","," ")

words = line.split()

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

temp = []

for y in words[1:]:

temp.append(y)

if words[0] not in actual\_pram.keys():

actual\_pram[words[0]] = []

actual\_pram[words[0]].append(temp)

print("\n\*\*\*\*\*\*\*First Pass of Macroprocessor\*\*\*\*\*\*\*")

print()

print("Intemediate Code : ") #Display Intermediate Code

for x in output\_code:

print(x, end=" ")

print()

print("\nMacro Defination Table (MDT) : ") #Display MDT

for x in mdt:

print(x[0],end = " ")

for y in x[1]:

print(y,end = " ")

print()

print()

print("Macro Name Table(MNT) : ") #Display MNT

print("Name of Macro | No. of Parameters | Starting Index")

for x in parameters.keys():

print(x,"\t|",len(parameters[x]),"\t\t\t|",start[x])

print("\nFormal vs Positional Parameters list: \n")

for key in parameters.keys():

if len(parameters[key]) > 0:

print("MACRO = ",key)

print("Formal Parameter| Positional Parameter")

k = 1

for x in parameters[key]:

print(x,"\t\t| ","#"+str(k))

k = k + 1

print()

print("\nActual vs Positional Parameters list: \n")

for key in actual\_pram.keys():

if len(parameters[key]) > 0:

print("MACRO = ",key)

for x in actual\_pram[key]:

k = 1

print("Actual Parameter| Positional Parameter")

for element in x:

print(element,"\t\t| ","#"+str(k))

k = k + 1

print()

fhand.close()

**Input File :**

START

MACRO COMPUTE &ARG

MOVER AREG,&ARG

ADD ARG,1

MOVEM AREG,&ARG

MEND

MACRO COMPUTE1 &ARG1,&ARG2,&ARG3

COMPUTE &ARG1

COMPUTE &ARG2

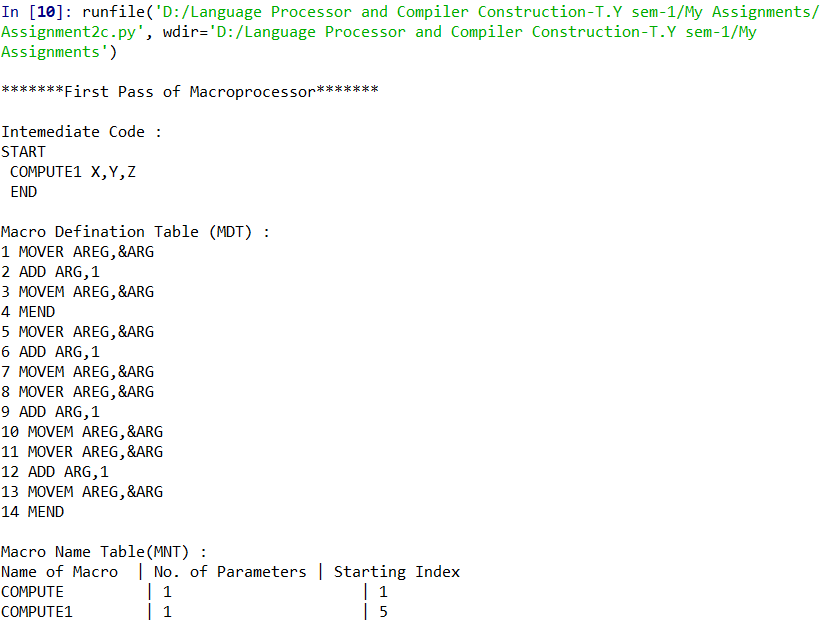
COMPUTE &ARG3

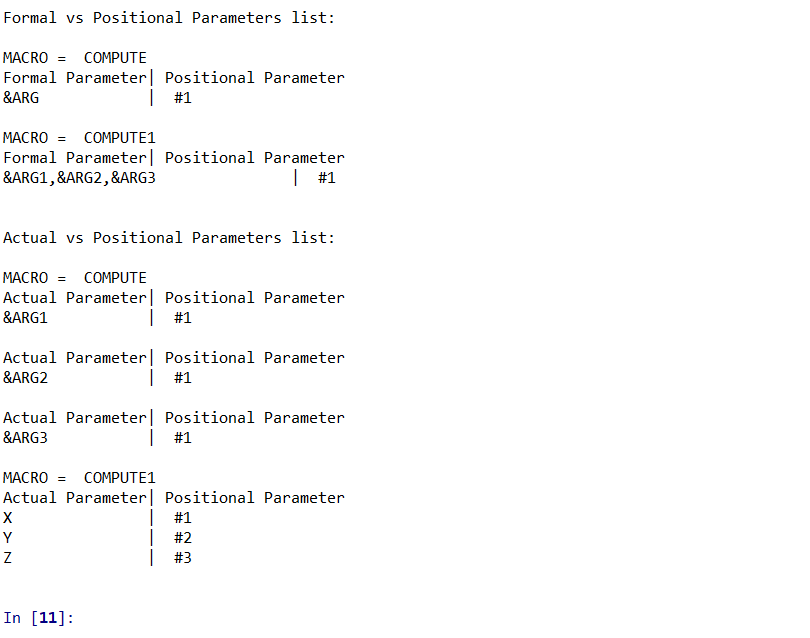
MEND

COMPUTE1 X,Y,Z

END

**Output:**

****

****

**Conclusion:**

Thus code for nested macro with call is implemented.