LPCC Assignment 2-a

Name: Digvijay Pawar

Class: TY.Btech Comp B2

Gr.No: 21810344

Roll No: 322043

**Aim:** Design suitable data structures & implement pass-I for a macro without parameter.

**Objective:** To understand concepts of Macro.

**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. identifiers, strings, numbers, operators and punctuations symbols can be considered as tokens.

**Code:**

**2A.py :**

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

output = []

code = {}

para = {}

curr\_mac = "NULL"

for line in fhand:

line.strip()

dup\_line = line

words=line.split()

if words[0] == "MACRO":

curr\_mac = words[1]

param = []

for y in words[2:]:

param.append(y)

code[words[1]] = []

para[words[1]] = param

elif words[0]!="MACRO" and curr\_mac=="NULL":

output.append(dup\_line)

elif words[0] == "MEND":

code[curr\_mac].append(words)

curr\_mac = "NULL"

elif words[0] != "MACRO" and curr\_mac != "NULL":

code[curr\_mac].append(words)

mdt = []

start = {}

i = 1

for key in code.keys():

values = {}

start[key] = i

for x in code[key]:

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

n = 0

st1 = x[:]

for element in st1:

if element in para[key]:

st1[n] = values[element]

n = n + 1

temp = [i,st1]

mdt.append(temp)

i = i + 1

print("First Pass: ")

print()

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

print()

for x in output:

print(x, end=" ")

print()

print("Macro 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 para | Starting Index")

for x in para.keys():

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

fhand.close()

**task.txt:**

START

READ A

READ B

MACRO SUB1

MOVER AREG N1

SUB AREG N2

MOVEM AREG N1

MEND

MACRO ADD1

MOVER AREG X

ADD AREG Y

MOVEM AREG X

MEND

ADD1 A B

SUB1 A B

A DS 1

B DS 1

**END**

**Output:**

