**LPCC ASSIGNMENT-2(WEEK-2)**

****

**SUBMITTED TO:**

**PROF. PALLAVI REGE**

**VISHWAKARMA INSTITUTE OF INFORMATION TECHNOLOGY, PUNE**

**COMPUTER ENGINEERING DEPARTMENT**

**BY:**

**NAME: ABHISHEK MORE**

**G.R No.: 21810033**

**ROLL NO.: 323036**

**CLASS: T.Y COMP**

**BATCH: COMP C2**

**LANGUAGE USED: PYTHON**

**ASSIGNMENT-2 (WEEK-2)**

**AIM:**

Design suitable data structures & implement FIRST pass of a two-pass Macro processor.

**INPUT CODE:**

LOAD A

STORE B

MACRO ABC

LOAD p

SUB q

MEND

MACRO ADD1 ARG

LOAD X

STORE ARG

MEND

MACRO ADD5 A1, A2, A3

STORE A2

ADD1 5

ADD1 10

LOAD A1

LOAD A3

MEND

ABC

ADD5 6, 7, 8

END

**SOURCE CODE:**

fr = open("Input.txt", "r") #Reading Input ALP code

intermediate\_code = [] #Intermediate Code code = {} #Stores

Code Body of each Macro parameters = {} #Stores Formal

Parameters of each macro current\_macro = "NULL" #Hold

Current macro in which reading

for x in fr: #Read input line by line y = x #For

Intermediate Code x = x.replace(","," ")

#Replace comma by space

lst = x.split()

if lst[0] != "MACRO" and current\_macro == "NULL": #If Intermediate Code

intermediate\_code.append(y) elif lst[0] == "MACRO": #If macro

current\_macro = lst[1] #Store current macro param = [] #Formal

Parameters of current macro

for y in lst[2:]:

param.append(y) code[lst[1]] = []

#code of current macro

parameters[lst[1]] = param #

parameters[current\_macro] = param elif

lst[0] == "MEND": # If MEND end of Macro

code[current\_macro].append(lst) #Add

MEND in the code of Current Macro

current\_macro = "NULL" elif lst[0] !=

"MACRO" and current\_macro != "NULL":

#Code inside Macro

code[current\_macro].append(lst) #Add

into code of current macro

mdt = [] #Macro Defination Table

start = {} #Starting Index i = 1

#Default Starting Index = 1 ala =

{} # Actual Parameter List

def expand(key,lst1): #Recursive Function to expand nested macro calls

global i

global ala

values = {}

k = 0

for y in parameters[key]: #Replace Formal Parameters by Actual parameters in Nested call

values[y] = lst1[k]

k = k + 1

for x in code[key]: # Go through the code of the current macro

if x[0] not in code.keys() and x[0] != "MEND": #If not nested macro and neglect 'MEND' for

nested macro

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():#If nested macro

temp = []

for y in x[1:]: # Store all the actual parameters into list

temp.append(y) if x[0] not in ala.keys():#Add actual

parameter into ala

ala[x[0]] = []

ala[x[0]].append(temp) expand(x[0],temp) #

Expand Nested macro Recursively

for key in code.keys(): #Add each macro into MDT by expanding

t = 1

values = {}

for x in parameters[key]: # replace Formal parameters by positional parameters

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

t = t+1

start[key] = i # Start index of the current a]macro for x in

code[key]: # Go through the code of the current macro if

x[0] not in code.keys(): # If not nested macro

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(): #If nested macro

temp = []

for y in x[1:]: # Store all the actual parameters into list

temp.append(y) if x[0] not in ala.keys(): #Add actual

parameter into ala

ala[x[0]] = []

ala[x[0]].append(temp) expand(x[0],temp) #

Expand Nested macro Recursively

for x in intermediate\_code: #Look for Actual parameters in Intermediate code

y = x.replace(","," ")

lst = y.split()

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

temp = []

for y in lst[1:]:

temp.append(y)

if lst[0] not in ala.keys():

ala[lst[0]] = []

ala[lst[0]].append(temp)

print(" First Pass of Macroprocessor ") print("------------------------------------------

-----------------------------------------")

print("\nIntemediate Code :-") #Display Intermediate Code

print("--------------------")

for x in intermediate\_code:

#str1 = " " #print(str1.join(x)) print(x) print("----------------------------------

-------------------------------------------------") print("\n Macro Defination

Table(MDT) \n") #Display MDT for x in mdt:

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

for y in x[1]:

print(y,end = " ") print() print("--------------------------------------------

---------------------------------------")

print("\n Macro Name Table(MNT) \n") #Display MNT

print("Name of Macro No. of Parameters Starting Index") for x in

parameters.keys():

print(x," ",len(parameters[x])," ",start[x]) print("------------

-----------------------------------------------------------------------") print("\n

Formal v/s Positional parameter list \n")

for key in parameters.keys():

if len(parameters[key]) > 0:

print(key,':-')

print("Formal Parameter Positional Parameter")

k = 1

for x in parameters[key]:

print(x," ","#"+str(k)) k = k + 1 print() print("-----

------------------------------------------------------------------------------") print("\n

Actual v/s positional parameter list (ALA) \n") for key in ala.keys():

if len(parameters[key]) > 0:

print(key,':-')

for x in ala[key]:

k = 1

print("Actual Parameter Positional Parameter")

for element in x:

print(element," ","#"+str(k))

k = k + 1

print() print("----------------------------------------------------------------------

-------------") fr.close()

**OUTPUT:**

First Pass of Macroprocessor

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

Intemediate Code :-

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

LOAD A

STORE B

ABC

ADD5 6, 7, 8

END

----------------------------------------------------------------------------------- Macro Defination

Table(MDT)

1 LOAD p

2 SUB q

3 MEND

4 LOAD X

5 STORE #1

6 MEND

7 STORE #2

8 LOAD X

9 STORE 5

10 LOAD X

11 STORE 10

12 LOAD #1

13 LOAD #3

14 MEND

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

Macro Name Table(MNT)

Name of Macro No. of Parameters Starting Index

ABC 0 1

ADD1 1 4

ADD5 3 7

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

Formal v/s Positional parameter list

ADD1 :-

Formal Parameter Positional Parameter

ARG #1

ADD5 :-

Formal Parameter Positional Parameter

A1 #1

A2 #2

A3 #3

----------------------------------------------------------------------------------- Actual v/s positional

parameter list (ALA)

ADD1 :-

Actual Parameter Positional Parameter

5 #1

Actual Parameter Positional Parameter

10 #1

ADD5 :-

Actual Parameter Positional Parameter

6 #1

7 #2

8 #3

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

**CONCLUSION:**

Successfully generated Intermediate code of a two-pass Assembler for the given source code also displayed the MNT, MDT table, Fromal vs Positional parameter list and Actual vs Positional parameter list using PYTHON programming language.