Name: Dhroov Makwana
Gr no.: 22010538
Roll no.: 321042
Batch: A2

Assignment 2

Aim: Design a suitable Data Structure to implement the First Pass of a 2-pass macro processor

Expected Output:

1. MNT (Macro Name Table)
2. MDT (Macro Definition Table)
3. Formal Vs Positional Parameter List
4. Actual Vs Positional Parameter List
5. Intermediate Code

Theory:

A Macro instruction is the notational convenience for the programmer. For every occurrence of macro the whole macro body or macro block of statements gets expanded in the main source code. Thus, Macro instructions make writing code more convenient.

• Macro represents a group of commonly used statements in the source programming language.
• Macro Processor replaces each macro instruction with the corresponding group of source language statements. This is known as the expansion of macros.
• Using Macro instructions programmer can leave the mechanical details to be handled by the macro processor.
• Macro Processor designs are not directly related to the computer architecture on which it runs.
• Macro Processor involves definition, invocation, and expansion.

Macro processor takes a source program containing macro definitions and macro calls. It converts it into an assembly language program without macro definition or calls.

Algorithm:

Pass 1:
1) Initialize MNTC (Macro Name Table) Counter = 0 and MDTC (Macro Definition Table Counter) = 0

2) Scanning of all macro definitions one by one . If MACRO found in program then for each macro perform:
3) MNTC = MNTC + 1 and enter name of MACRO in MNTC (Macro Name Table).
4) For every model statement MDTC = MDTC + 1 in definition of macro
5) Generate argument list array.

Pass 2:

1) Scan main program for macro call. For each macro call perform.
2) Scan MNT to detect Macro Name and its address in MDT (Macro Definition Table).
3) Replacement of all formal parameters by actual parameters.
4) Replace macro call by model statements from MDT.

Code:

```
# Reading file and storing contents
f = open("sample2.asm", 'r')
file = (f.read()).split("\n")
f.close()
SC = []
for i in file:
    a = i.split("\t")
    SC.append(a)
del file
# displaying file
f = open("sample2.asm", 'r')
print("Source Code : \n")
print(f.read())


# Macro function
def macro(a):
    # macro name
    mname = SC[a][1]
    MNT[0].append(mname)
    # macro parameters
    if SC[a][2] != '':
        para = list(SC[a][2].split(","))
```

```
        Parameter[mname] = {}
        for i in range(0, len(para)):
            Parameter[mname][para[i]] = "#" + str(i + 1)
    else:
        para = []
    num_para = len(para)
    MNT[1].append(num_para)
    # starting index for macro
    MNT[2].append(len(MDT) + 1)
    # macro definition Table
    a += 1
    while SC[a][0] != "MEND":
        while SC[a][0] == '':
            del (SC[a][0])
        if SC[a][0] in MNT[0]:
            i = MNT[0].index(SC[a][0])
            i = MNT[2][i] - 1
            t_para = list(SC[a][1].split(","))
            if SC[a][0] not in actual_para.keys():
                actual_para[SC[a][0]] = {}
            while MDT[i] != "MEND":
                b = MDT[i]
                for j in range(0, len(t_para)):
                    temp = str("#" + str(j + 1))
                    if b.find(temp) != -1:
                        b = b.replace(temp, t_para[j])
                        actual_para[SC[a][0]][t_para[j]] = temp
                    # if "#"+str(j) is present in b:
                    #   replace with para(j)
                MDT.append(b)
                i += 1
        else:
            b = SC[a]
            if b[1] in para:
                b[1] = Parameter[mname][b[1]]
            MDT.append(" ".join(b))
        a += 1
    MDT.append("MEND")


MDT = []
MNT = [[], [], []]
Parameter = {}
actual_para = {}
# Intermediate Code
print("----------------------")
print("Intermediate Code : ")
print("----------------------\n")
i = 0
while i < len(SC):
    if SC[i][0] == 'MACRO':
```

```
            macro(i)
            while SC[i][0] != "MEND":
                i += 1
        else:
            if SC[i][0] == "":
                del (SC[i][0])
            print(" ".join(SC[i]))
        i += 1
# MNT
print("\n\n---------------------")
print("MNT : ")
print("---------------------\n")
a = len(MNT[0])
print("Name\tNo. Para\tStart index")
for i in range(0, a):
    print(MNT[0][i], "\t", MNT[1][i], "\t\t", MNT[2][i])
# MDT
print("\n\n---------------------")
print("MDT : ")
print("---------------------\n")
for i in MDT:
    print(i)
# Formal Parameters
print("\n\n-------------------------------")
print("Formal Vs Positional Parameters : ")
print("-------------------------------\n")
for i in Parameter:
    print(i)
    print("Formal\tPositional Parameter")
    for j in Parameter[i]:
        print(j, "\t", Parameter[i][j])
    print("")
# Actual Parameters
print("\n\n-------------------------------")
print("Actual Vs Positional Parameters : ")
print("-------------------------------\n")
for i in actual_para:
    print(i)
    print("Actual\tPositional Parameter")
    for j in actual_para[i]:
        print(j, "\t", actual_para[i][j])
    print("")
```

Input file:

```
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    D1,D2,D3
END
```

Output:

```
---------------------
Intermediate Code :
---------------------

LOAD A
STORE B
ABC
ADD5 D1,D2,D3
END
```

```
---------------------
MNT :
---------------------

Name    No. Para    Start index
ABC      0       1
ADD1     1       4
ADD5     3       7
```

```
--------------------
MDT :
--------------------

LOAD p
SUB q
MEND
LOAD X
STORE #1
MEND
STORE #2
LOAD X
STORE 5
LOAD X
STORE 10
LOAD #1
LOAD #3
MEND
```

```
------------------------------
Formal Vs Positional Parameters :
------------------------------

ADD1
Formal  Positional Parameter
ARG      #1

ADD5
Formal  Positional Parameter
A1   #1
A2   #2
A3   #3
```

```
------------------------------
Actual Vs Positional Parameters :
------------------------------

ADD1
Actual  Positional Parameter
5    #1
10   #1


Process finished with exit code 0
```

Conclusion: Implemented the first pass of two pass macro-processor using python and displayed the contents of MDT, MNT, intermediate code, also displayed the formal vs positional parameters list and actual vs positional parameters list.