EXPERIMENT 3

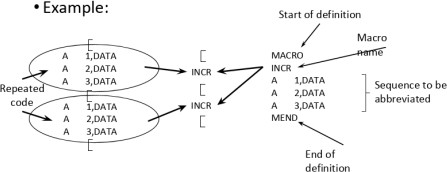
**CLASS: TE CMPN A PID:182027**

**NAME: REBECCA DIAS ROLL NO. : 19**

**AIM: To demonstrate a 2 pass macro processor. THEORY:**

## Macro instructions or Macros are single line abbreviations for groups of

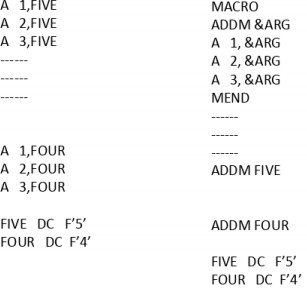
instructions. Single instruction is used to represent a block of code. For every occurrence of this one line macro instruction, the macro processing assembler will substitute the entire block.



Features of Macro Facility:

Macro Instruction Arguments

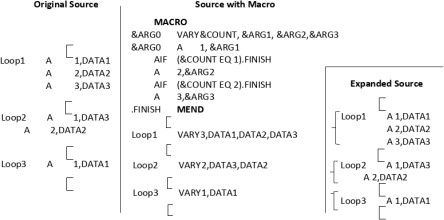
Macro calls replaces the call by a block of code. No ﬂexibility to modify code that replaces the call. Extension for providing arguments or parameters in macro call. Macro instruction argument (dummy arguments) are used in deﬁnition. It is speciﬁed in the macro name line and distinguished by ‘&’ Arguments that are not speciﬁed, are presumed blank by macro processor.



Conditional Macro Expansion

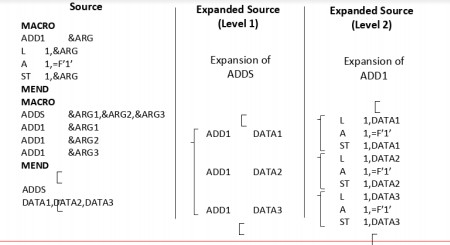
AIF and AGO permit conditional reordering of the sequence of macro expansion. Machine instructions that appear in the expansion of a macro call can be selected based on condition.

AIF is used for conditional branching whereas AGO is used for unconditional branching.



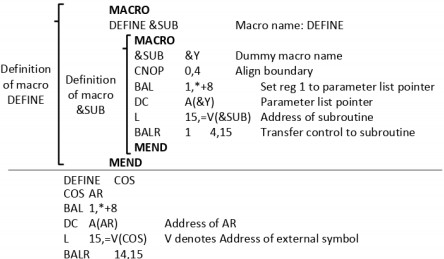
Macro calls within Macros

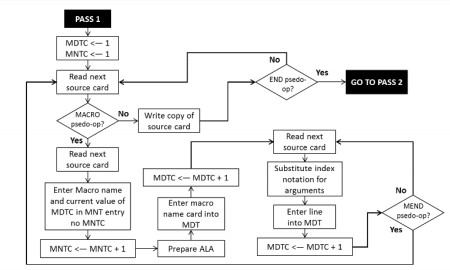
Also known as nested macro calls. A macro can be called within another macro. A macro can call itself (using AIF or AGO) so long as it doesn’t go into an inﬁnite loop. Macro calls within macros can have several levels.



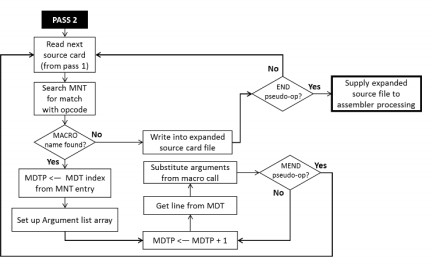
Macro Instruction deﬁning Macros

Macros can be deﬁned within a macro. Inner macro deﬁnition is not deﬁned until after the outer macro has been called. Group of macros can be deﬁned for subroutine calls with some standardized calling sequence.



Flowchart of a 2 pass macro processor: Pass 1 –

Pass 2 –



Example: source.txt

ABC START MACRO

ADD &ARG1 , &ARG2 L 1 , &ARG1

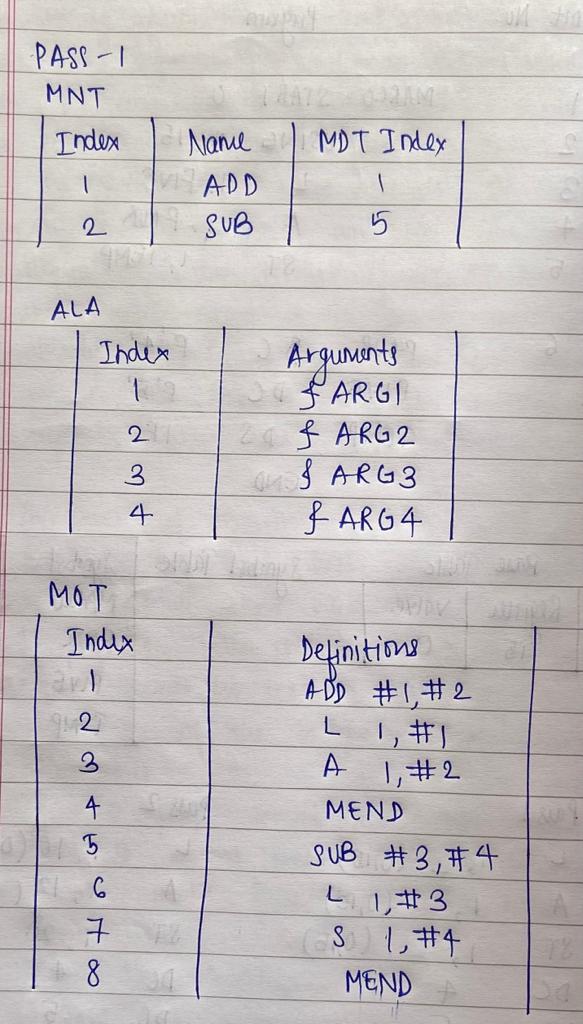
A 1 , &ARG2 MEND

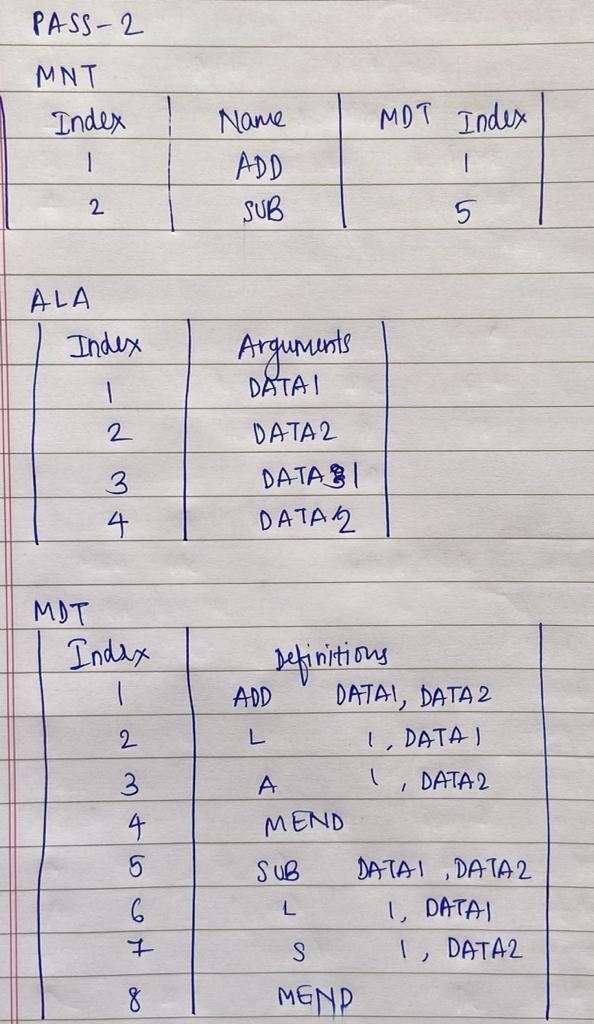
MACRO

SUB &ARG3 , &ARG4 L 1 , &ARG3

S 1 , &ARG4 MEND

ADD DATA1 , DATA2 SUB DATA1 , DATA2 DATA1 DC F'9' DATA2 DC F'5' END





**IMPLEMENTATION:**

with open("source.txt","r") as fi: content=fi.readlines()

words=[]

for line in content:

words.append(line.strip().split(" "))

MNT=[]

ALA={}

MDT=[] MNTC=0 MDTC=0

ALAC=0

global\_tracker={} num=25

for i in range(len(words)):

if (words[i][0]=="MACRO"): j=1

arg\_list=[]

while (words[i+j][0]!="MEND"): if (j==1):

MNTC+=1 MDTC+=1

MNT.append((MNTC,words[i+j][0],MDTC)) formatted\_line=words[i+j][0]+" "

for word in words[i+j]: if ("&" in word):

ALAC+=1

ALA[word]=ALAC

formatted\_line+="#"+str(ALA[word])+" , " arg\_list.append(word)

global\_tracker[words[i+j][0]]=arg\_list

formatted\_line=formatted\_line[:len(formatted\_line)-3] MDT.append([MDTC,formatted\_line])

else:

formatted\_line=""

for word in words[i+j]:

if (word not in arg\_list):

formatted\_line+=word+" " else:

formatted\_line+="#"+str(ALA[word])

MDTC+=1

MDT.append([MDTC,formatted\_line]) j+=1

MDTC+=1

MDT.append([MDTC,"MEND"])

#Pass 1

print("Pass 1") print("\nMNT") print("-"\*num)

print("Index\tName\tMDT Index") for values in MNT:

print(f"{values[0]}\t{values[1]}\t{values[2]}")

print("\nALA") print("-"\*num)

print("Index\tArguments")

for key,value in ALA.items(): print(f"{value}\t{key}")

print("\nMDT") print("-"\*num)

print("Index\tDefinitions") for values in MDT:

print(f"{values[0]}\t{values[1]}")

gen\_strings=[]

for macro\_name,arguments in global\_tracker.items(): formatted\_line=macro\_name+" "

for arg in arguments:

formatted\_line+=arg+" , "

formatted\_line=formatted\_line[:len(formatted\_line)-3] gen\_strings.append(formatted\_line.split(" "))

mapper={}

for x in gen\_strings: for word in words:

if (word[0]==x[0] and word!=x): for i in range(len(word)):

if (word[i]!=x[i]):

mapper[ALA[x[i]]]=word[i]

for i in range(len(MDT)):

\_,m\_content=MDT[i]

for key,value in mapper.items():

if ("#"+str(key) in m\_content):

m\_content=m\_content.replace("#"+str(key),mapper[key])

MDT[i][1]=m\_content

#Pass 2

print("\nPass 2") print("\nMNT")

print("-"\*num)

print("Index\tName\tMDT Index") for values in MNT:

print(f"{values[0]}\t{values[1]}\t{values[2]}")

print("\nALA") print("-"\*num)

print("Index\tArguments")

for key,value in mapper.items(): print(f"{key}\t{value}")

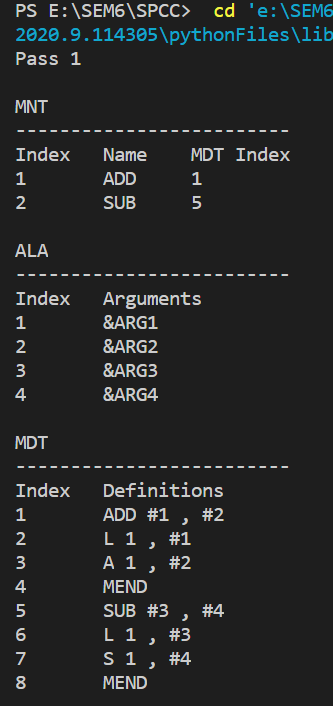
print("\nMDT")

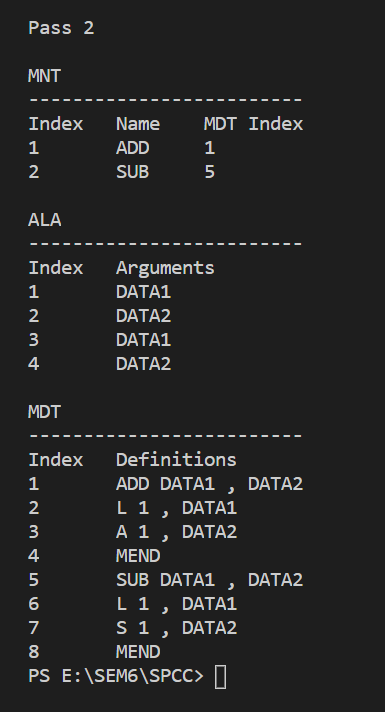
print("-"\*num)

print("Index\tDefinitions") for values in MDT:

print(f"{values[0]}\t{values[1]}")

# OUTPUT:

****

****

**CONCLUSION:**

## The working of a two pass macro processor is demonstrated. The output of the program was cross checked with actual