Nikit Gokhe

Class : TY-Comp D1

Roll No. 324022

GR No. 21810522

# Assignment 02

# Aim:

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

# Theory:

**Macro Processor:**

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.

A macro is a fragment of code which has been given a name. Whenever the name is used, it is replaced by the contents of the macro. There are two kinds of macros. They differ mostly in what they look like when they are used. Object-like macros resemble data objects when used, function- like macros resemble function calls.

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.

# Source Code:

MDT = []  #Macro definition table array

MNT = {}  #Macro name table dictionary

ARG = {}  # argument dictionary

iARG = 1

output = []  #lines of output

i=0  #index output

iMDT=1

ctr=0

f = open("/content/input2.txt", "r")

Lines = f.readlines()  #reading line by line

f.close

for Line in Lines:

    Line = Line.replace(","," ")

    substrings = Line.split()

    if ctr==1:

        for x in range(0,len(substrings)):

            if substrings[x] in MNT:

                for l in range(x+1,len(substrings)):

                    if substrings[l].isnumeric() is False:

# to avoid taking numeric argument in my argument dict

                        ARG[substrings[l]]="#" + str(iARG)

                        iARG+=1

                    else:

                        ARG["#" + str(iARG)] = substrings[l]

# now as this call is inside macro definition which means the macro called is already with #1 and #2 argument so we will replace it with the actual numeric argument, this will only happen when the argument is numeric

                        iARG+=1

                for y in range(MNT[substrings[x]]-1,iMDT):

                    MDT.append(MDT[y])

                    strings= MDT[iMDT-1].split()

                    for string in strings:

                        if string in ARG:

                            MDT[iMDT-1]= MDT[iMDT-1].replace(string,ARG[string])

                    iMDT+=1

                    if "MEND" in MDT[y+1]:

                        break

            elif substrings[x] in ARG:

               Line = Line.replace(substrings[x],ARG[substrings[x]]) # replacing argument with #1 and #2...

       if substrings[0] not in MNT:

            MDT.append(Line[0:-1])

            iMDT+=1

        if "MEND" in Line:

            ctr=0

    elif "MACRO" in Line:

        MNT[substrings[1]]=iMDT

        for x in range(2,len(substrings)):

            ARG[substrings[x]]="#" + str(iARG)

# creating an argument dict to replace it with #1 and #2...

iARG+=1

        iARG=1

        ctr=1

 # to indicate that macro defintion table should be updated

    else:

        output.append(Line[0:-1])

        i+=1

print("\nMNT Table")

print("SrNo M-Name MDT-Pointer")

x=0

for Mname in MNT:

    x+=1

    print(str(x),"   ",Mname,"  ",MNT[Mname])

print("\nMDT Table:")

for i in range(0,len(MDT)):

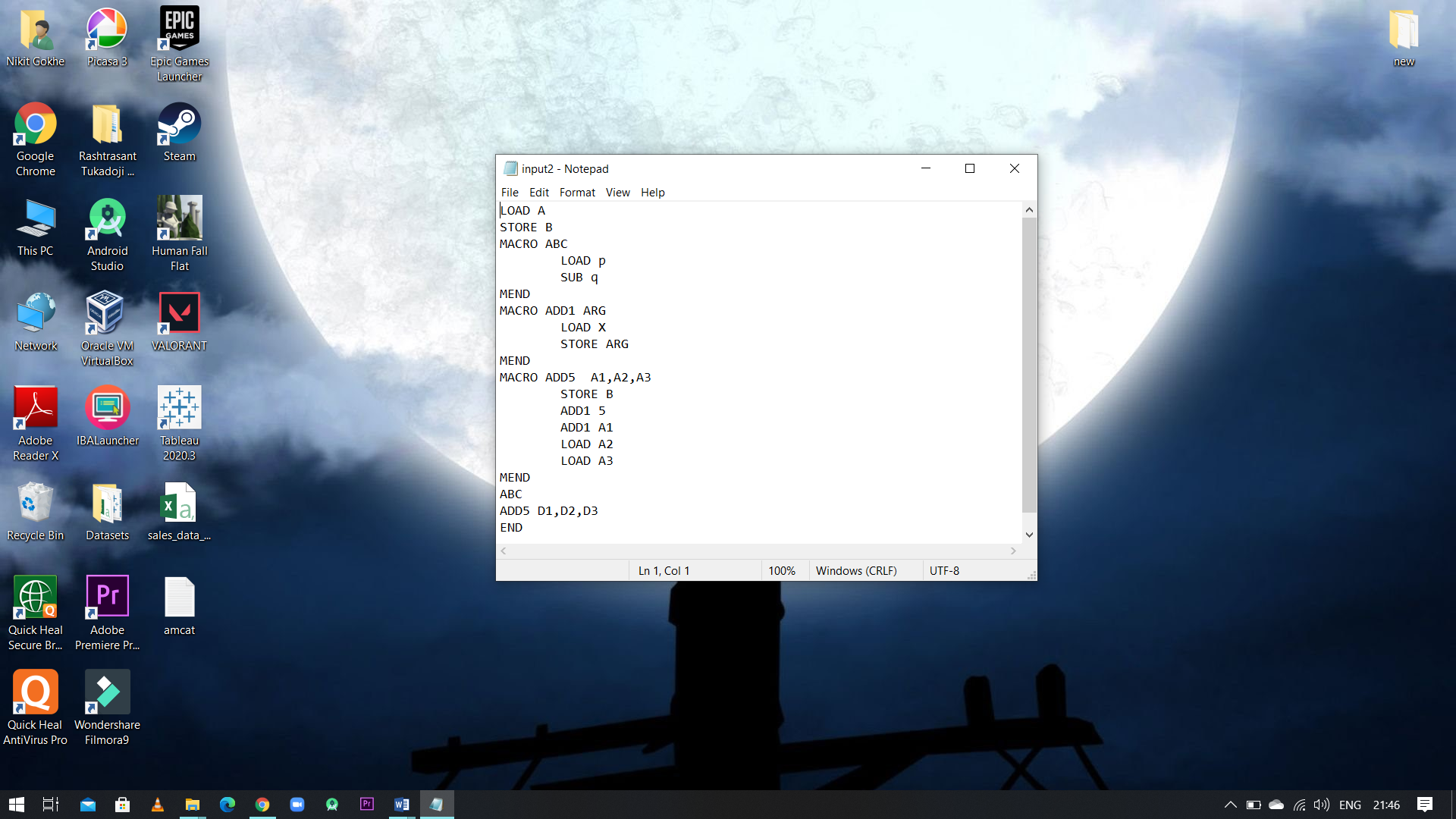
    print(i+1,"  ",MDT[i])

print("\nIntermediate Code:")

for i in range(0,len(output)):

    print(output[i])

# Input:



# Output:

