Nikit Gokhe

Class : TY-Comp D1

Roll No. 324022

GR No. 21810522

# Assignment 01

# Aim:

Generate Symbol table, Literal table, Pool table & Intermediate code of a two-pass Assembler for the given source code.

# Theory:

**Symbol Table:** Each entry in symbol table has two primary fields name and address. The symbol table uses concept of forward reference to achieve address of symbols.

**OPTAB:** Each entry in operation table contains fields like mnemonic, class and mnemonic info. The class field many contain. IS (Imperative Statements), AD (Assembler Directives) or DL (Declarative Statements)

**Literal Table:** Literal table contains all literals and address of all literals. For literals LTORG directives places the all constants at consecutive memory locations.

If we are not using LTORG then all literals are placed after END in memory.

**Pool Table (POOLTAB):** Awareness of different literal pools is maintained using the auxiliary table POOLTAB. At any Stage, the current literal pool is the last pool in LITTAB.

# Source Code:

#1

MOT={

    'STOP':{"class": "1","opcode": "00"},'ADD':{"class": "1","opcode": "01"},'SUB':{"class": "1","opcode": "02"},'MULT':{"class": "1","opcode": "03"},

    'MOVER':{"class": "1","opcode": "04"},'MOVEM':{"class": "1","opcode": "05"},'COMP':{"class": "1","opcode": "06"},'BC':{"class": "1","opcode": "07"},

    'DIV':{"class": "1","opcode": "08"},'READ':{"class": "1","opcode": "09"},'PRINT':{"class": "1","opcode": "10"},'START':{"class": "3","opcode": "00"},

    'END':{"class": "3","opcode": "01"},'ORIGIN':{"class": "3","opcode": "02"},'EQU':{"class": "3","opcode": "03"},'LTORG':{"class": "3","opcode": "04"},

    'DS':{"class": "2","opcode": "01"},'DC':{"class": "2","opcode": "02"},'AREG':{"class": "4","opcode": "01"},'BREG':{"class": "4","opcode": "02"},

    'CREG':{"class": "4","opcode": "03"},'EQ':{"class": "5","opcode": "01"},'LT':{"class": "5","opcode": "02"},'GT':{"class": "5","opcode": "03"},

    'NE':{"class": "5","opcode": "04"},'ANY':{"class": "5","opcode": "05"}

}

#2 to read input file

LC = 0

size = 0

ST = {}

LT =[]

LTadd =[]

iLT=0

iPT=0

PT = {}

code = []

PT[iPT]=iLT

f = open("/content/input.txt", "r")

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

f.close

#3

for Line in Lines:

    output = Line[0:-1]

    for i in range(len(Line),28):

        output = output + " "

    Line = Line.replace(',',' ')

    subStrings = Line.split()

    # print(LC)

    if "START" in Line:

        LC = int(subStrings[1])

        output = output + "(AD,01)  (C," + str(LC) + ")"

    elif "ORIGIN" in Line:

        output = output + str(LC)

        if "+" in subStrings[1]:

            subStrings[1] = subStrings[1].replace("+"," ")

            part = subStrings[1].split()

            LC = ST[part[0]] + int(part[1])

        elif "-" in subStrings[1]:

            subStrings[1] = subStrings[1].replace("+"," ")

            part = subStrings[1].split()

            LC = ST[part[0]] + int(part[1])

        output = output + " (AD,03)  (C," + str(LC) + ")"

    elif "END" in Line:

        for x in range(len(LT)):

            if(LTadd[x]==0):

                output = output + str(LC)

                output = output + " (DL,02)  (C," + str(x) + ")\n"

                LTadd[x]=LC

                LC+=1

                iLT+=1

        break

    else:

        if "LTORG" in Line:

            output = output + "  "

        else:

            output = output + str(LC)

        #if label adding it to ST

        if not subStrings[0] in MOT:

            if "EQU" not in Line:

                ST[subStrings[0]] = LC

            else:

                LC=ST[subStrings[-1]]

                ST[subStrings[0]] = LC

        # declarative sentence

        for i in range(len(subStrings)):

           if subStrings[i] in MOT and MOT[subStrings[i]]['class']== '2':    #if class=2 then declarative

               LC = LC + 1

        # imperative sentence

        for i in range(len(subStrings)):

            if subStrings[i] in MOT and MOT[subStrings[i]]['class']== '1':   #if class=2 then imperative

                LC +=1

                if subStrings[-1].isalpha() is False:

                    LT.append(subStrings[-1])

                    LTadd.append(0)

        #dealing with LTORG for pool table

        for i in range(len(subStrings)):

            if subStrings[i]=="LTORG":

                ctr = 0

                for x in range(len(LT)):

                    if(LTadd[x]==0):

                        output = output + str(LC)

                        output = output + " (DL,02)  (C," + str(x) + ")\n"

                        ctr=1

                        LTadd[x]=LC

                        LC+=1

                        iLT+=1

                if(ctr==1):

                    iPT +=1

                    PT[iPT]=iLT

    for word in subStrings:

        if word in MOT and word!="START" and word!="ORIGIN" and word!="LTORG" :

            if MOT[word]['class'] == "1":

                output = output + " (IS," + str(MOT[word]['opcode']) + ")"

            elif MOT[word]['class'] == "2":

                output = output + " (DL," + str(MOT[word]['opcode']) + ")"

            elif MOT[word]['class'] == "3":

                output = output + " (AD," + str(MOT[word]['opcode']) + ")"

            elif MOT[word]['class'] == "4":

                output = output + " (RG," + str(MOT[word]['opcode']) + ")"

            elif MOT[word]['class'] == "5":

                output = output + " (CC," + str(MOT[word]['opcode']) + ")"

        elif word in ST:

            pos = list(ST.keys()).index(word)

            output = output + " (S," + str(pos) + ")"

        elif word in LT:

            x= LT.index(word)

            if LTadd[x]!=0:

                output = output + " (" + str(LTadd[x]) + ")"

            else:

                output = output + " " + word

        elif word.isnumeric() is True and "START" in Line is False:

            output = output + " (C," + word + ")"

    code.append(output)

for x in range(0,len(code)):

    strings = code[x].split()

    if not strings[-1] in MOT:

        if strings[-1] in ST:

            pos = list(ST.keys()).index(strings[-1])

            l=len(strings[-1])

            code[x] = code[x][0:-(l+1)]

            rep = "  (S," + str(pos) + ")"

            code[x] = code[x] + rep

        elif strings[-1] in LT:

            pos= LT.index(strings[-1])

            l=len(strings[-1])

            code[x] = code[x][0:-(l+1)]

            rep = "  (L," + str(pos) + ")"

            code[x] = code[x] + rep

    print(code[x])

print(" ")

print("Symbol Table:")

print("Sr.No Symbol  Address")

a = 0

for sym in ST:

    a +=1

    print(str(a),"   ",sym,"   ",ST[sym])

print(" ")

print("Literal Table:")

print("Sr.No Literal  Address")

a = 0

for x in range(len(LT)):

    print(str(a),"   ",LT[x],"   ",LTadd[x])

    a +=1

print(" ")

print("Pool Table:")

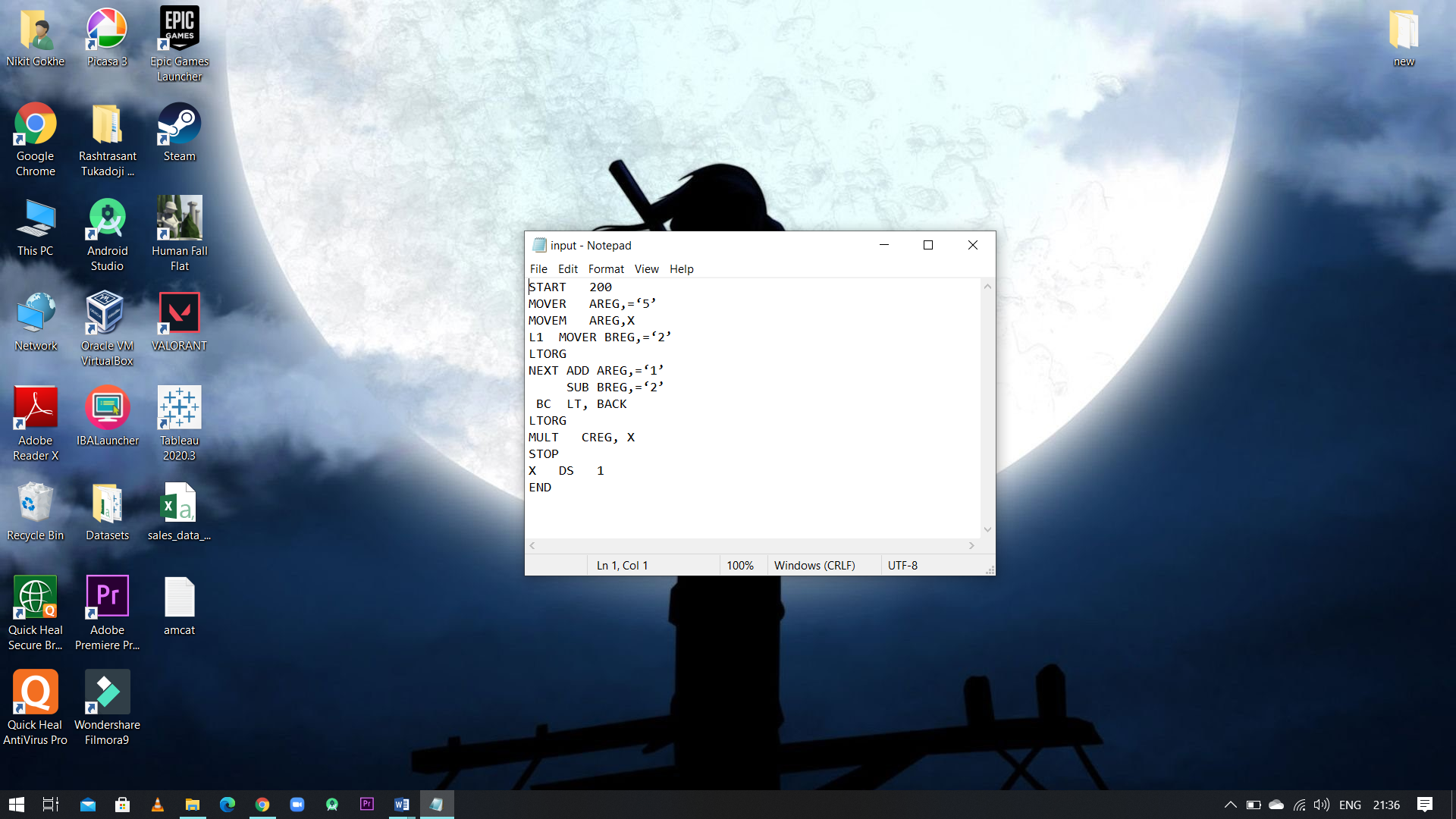
print("SrNo Literal Number")

for i in PT:

    if (PT[i]<len(LT)):

        print(i,"   ",PT[i])

# Input:



# Output:

