**Name : Aditi Nikam Roll No : 323045**

**GR No : 21910513 Class : TY Comp C2**

**WEEK 4**

**Code:**

import pandas as pd

import os

IC\_output = open('IC\_Code.txt','w+')

emot = pd.read\_csv('emot.csv')

cls\_field = pd.read\_csv('class\_field.csv')

st = pd.DataFrame()

lt = pd.DataFrame()

file = open("file.txt","r")

file.seek(0)

emot\_val = emot['Mnemonic'].tolist()

emot\_cls\_val = emot['Class'].tolist()

emot\_op = emot['Opcode'].tolist()

value\_cls\_field = cls\_field['VAL. OF CLASS FIELD'].tolist()

sym\_cls\_field = cls\_field['SYMBOL'].tolist()

LC = 0

literal\_values = []

LC\_literal = []

symbol\_values = []

symbol\_size = []

LC\_symbol = []

pool\_table = [0]

lit\_ltorg = []

for line in file:

    line = line.replace('\n',' ')

    line = line.replace(',',' ')

    words = line.split(' ')

    IC\_line = ''

    if words[0] == 'START':

        LC = *int*(words[1])-1

    if words[0] == 'ORIGIN':

        data = words[1].split('+')

        if data[0] in symbol\_values:

            ind = symbol\_values.index(data[0])

            LC = LC\_symbol[ind] + *int*(data[1])

        IC\_line = "(AD,3)(C," + *str*(LC) + ")"

        IC\_output.write(IC\_line+"\n")

        continue

    if len(words) > 1 and words[1] == "EQU":

        if words[0] not in symbol\_values:

            symbol\_values.append(words[0])

            LC\_symbol.append(LC)

        LC\_symbol[symbol\_values.index(words[0])] = LC\_symbol[symbol\_values.index(words[2])]

        IC\_line = "(S," + *str*(symbol\_values.index(words[0])) + ")(AD,4)(C," + *str*(LC\_symbol[symbol\_values.index(words[2])]) + ")"

        IC\_output.write(IC\_line+"\n")

        continue

    if len(words) > 1 and words[1] == "DC":

        if words[0] in symbol\_values:

            LC\_symbol[symbol\_values.index(words[0])] = LC

            symbol\_size[symbol\_values.index(words[0])] = words[2]

        LC = LC + *int*(words[2])

        continue

    for word in words:

        if word.startswith('='):

            literal = *int*(word[2:*int*(len(word)-1)])

            if literal not in lit\_ltorg:

                lit\_ltorg.append(literal)

                literal\_values.append(literal)

            rev\_lit = literal\_values[::-1]

            IC\_line += "(L," + *str*(len(literal\_values) - rev\_lit.index(literal)-1) + ")"

            continue

        if word == 'LTORG':

            pool\_table.append(len(literal\_values))

            for k in range(pool\_table[-2],len(literal\_values)):

                LC\_literal.append(LC)

                LC += 1

                if k < len(literal\_values)-1:

                    IC\_line += "(DL,2)(C," + *str*(literal\_values[k]) + ")\n"

                else:

                    IC\_line += "(DL,2)(C," + *str*(literal\_values[k]) + ")"

            lit\_ltorg = []

            # IC\_line += "(DL,02) (C," + literal\_values[k] + ")"

            LC -= 1

            continue

        if word.isdigit():

            IC\_line += "(C," + word + ")"

        if word not in emot\_val and word != '' and not word.isdigit():

            if word not in symbol\_values:

                symbol\_values.append(word)

                LC\_symbol.append(LC)

                symbol\_size.append(1)

            else:

                LC\_symbol[symbol\_values.index(word)] = LC

            IC\_line += "(S," + *str*(symbol\_values.index(word)) + ")"

        # if word in symbol\_values:

        #     IC\_line += "(S," + str(symbol\_values.index(word)) + ")"

        if word in emot\_val:

            class\_val = emot\_cls\_val[emot\_val.index(word)]

            symbol = sym\_cls\_field[value\_cls\_field.index(class\_val)]

            op\_code = *str*(emot\_op[emot\_val.index(word)])

            IC\_line += "(" + symbol + "," + op\_code + ")"

    LC += 1

    IC\_output.write(IC\_line+"\n")

for i in range(pool\_table[-1],len(literal\_values)):

    LC\_literal.append(LC)

    LC += 1

    IC\_line = "(DL,2)(C," + *str*(literal\_values[i]) + ")"

    IC\_output.write(IC\_line+"\n")

df\_st = pd.DataFrame()

df\_lt = pd.DataFrame()

df\_pt = pd.DataFrame()

st\_values = pd.Series(symbol\_values)

lc\_symbol = pd.Series(LC\_symbol)

sizes = pd.Series(symbol\_size)

df\_st.insert(*loc*=0, *column*='Symbol', *value* = st\_values)

df\_st.insert(*loc*=1, *column*='Size', *value* = sizes)

df\_st.insert(*loc*=1, *column* = 'Address', *value* = lc\_symbol)

df\_st.to\_csv('Symbol Table.csv')

print("\nSymbol Table\n")

print(df\_st)

lit\_vals = pd.Series(literal\_values)

lc\_literal = pd.Series(LC\_literal)

df\_lt.insert(*loc*=0, *column*='Literals', *value* = lit\_vals)

df\_lt.insert(*loc* = 1, *column* = 'Address', *value* = lc\_literal)

df\_lt.to\_csv('Literal Table.csv')

print('\nLiteral Table\n')

print(df\_lt)

pool = pd.Series(pool\_table)

df\_pt.insert(*loc* = 0, *column* = 'Pool', *value* = pool)

df\_pt.to\_csv('Pool Table.csv')

print('\nPool Table\n')

print(df\_pt)

**emot.csv:**

Mnemonic,Class,Opcode

STOP,1,0

ADD,1,1

SUB,1,2

MULT,1,3

MOVER,1,4

MOVEM,1,5

COMP,1,6

BC,1,7

DIV,1,8

READ,1,9

PRINT,1,10

START,3,1

END,3,2

ORIGIN,3,3

EQU,3,4

LTORG,3,5

DS,2,1

DC,2,2

AREG,4,1

BREG,4,2

CREG,4,3

EQ,5,1

LT,5,2

GT,5,3

NE,5,4

LE,5,5

GT,5,6

ANY,5,7

**class\_field.csv:**

Type,SYMBOL,VAL. OF CLASS FIELD

Imperative Statements,IS,1

Declarative Statements,DL,2

Assembler Directive,AD,3

CPU Register,RG,4

Conditional codes,CC,5

**file.txt:**

START 200

MOVER AREG,='5'

MOVEM AREG,X

L1 MOVER BREG,='2'

ORIGIN L1+3

LTORG

NEXT ADD AREG,='1'

SUB BREG,='2'

BC LT, BACK

LTORG

BACK EQU L1

ORIGIN NEXT+5

MULT CREG,='4'

STOP

X DS 1

END

**Output:**



**Symbol Table.csv:**

,Symbol,Address,Size

0,X,214,1

1,L1,202,1

2,NEXT,207,1

3,BACK,202,1

**Literal Table.csv:**

,Literals,Address

0,5,205

1,2,206

2,1,210

3,2,211

4,4,216

**Pool Table.csv:**

,Pool

0,0

1,2

2,4

**IC\_Code.txt:**

(AD,1)(C,200)

(IS,4)(RG,1)(L,0)

(IS,5)(RG,1)(S,0)

(S,1)(IS,4)(RG,2)(L,1)

(AD,3)(C,205)

(DL,2)(C,5)

(DL,2)(C,2)

(S,2)(IS,1)(RG,1)(L,2)

(IS,2)(RG,2)(L,3)

(IS,7)(CC,2)(S,3)

(DL,2)(C,1)

(DL,2)(C,2)

(S,3)(AD,4)(C,202)

(AD,3)(C,212)

(IS,3)(RG,3)(L,4)

(IS,0)

(S,0)(DL,1)(C,1)

(AD,2)

(DL,2)(C,4)