Name: Sejal Girish Mahajan

Roll no. 05

PASS 1

```
from os import sep, write


class Mnemonics:
    def __init__(self):
        self.AD = {
            "START": 1,
            "END": 2,
            "ORIGIN": 3,
            "EQU": 4,
            "LTORG": 5
        }
        self.RG = {
            "AREG": 1,
            "BREG": 2,
            "CREG": 3,
            "DREG": 4
        }
        self.DL = {
            "DC": 1,
            "DS": 2
        }
        self.IS = {
            "STOP": 0,
            "ADD": 1,
            "SUB": 2,
            "MULT": 3,
            "MOVER": 4,
```

```
        "MOVEM": 5,
        "COMP": 6,
        "BC": 7,
        "DIV": 8,
        "READ": 9,
        "PRINT": 10
    }
    self.CC = {
        "LT": 1,
        "LE": 2,
        "EQ": 3,
        "GT": 4,
        "GE": 5,
        "ANY": 6
    }

def getClassType(self,string):
    if string in self.AD:
        return "AD"
    elif string in self.CC:
        return "CC"
    elif string in self.DL:
        return "DL"
    elif string in self.IS:
        return "IS"
    elif string in self.RG:
        return "RG"
    else:
        return ""

def getMachineCode(self,string):
```

```
        if string in self.AD:
            return self.AD[string]
        elif string in self.CC:
            return self.CC[string]
        elif string in self.DL:
            return self.DL[string]
        elif string in self.IS:
            return self.IS[string]
        elif string in self.RG:
            return self.RG[string]
        else:
            return -1



class pass1:
    def __init__(self):
        self.lookup = Mnemonics()
        self.symbolTable = {}
        self.literalTable = {}
        self.litTableIndex = 0 #Used to append values in literal Table
        self.poolTable = [0]
        self.IC = [] #IC Intermediate Code
        self.location = 0
        self.litTabPtr = 0 #Literal Table Pointer
        self.InputFile = open("input.txt","r")
        self.literalTableFile = open("literalTable.txt","w")
        self.symbolTableFile = open("symbolTable.txt","w")
        self.poolTableFile = open("poolTable.txt","w")
        self.ICFile = open("intermediateCode.txt","w")
```

```
def calculateLocation(self,string):
    if "+" in string:
        string = string.split("+")
        return self.symbolTable[string[0]] + int(string[1])
    elif "-" in string:
        string = string.split("-")
        return self.symbolTable[string[0]] - int(string[1])
    else:
        return self.symbolTable[string]


def parseFile(self):
    for line in self.InputFile.readlines():
        self.IC.append([])
        line = line.strip("\n")
        line = line.split("\t")



        # For label
        if line[0] != "":
            if line[0] in self.symbolTable:
                self.symbolTable[line[0]] = self.location #Set location
            else:
                self.symbolTable[line[0]] = self.location #add label,loc


        # For Opcode
        if line[1] == "START":
            self.location = int(line[2])
            self.IC[-1].append(('AD',1))
            self.IC[-1].append(("C",int(line[2])))
        elif line[1] == "LTORG":
            # literalKeys = list(self.literalTable.keys())
```

```
        for i in range(self.poolTable[-1],len(self.literalTable)):
            self.literalTable[i][1] = self.location
            self.IC[-1].append(("DL",1))
            self.IC[-1].append(("C",self.literalTable[i][0]))
            self.IC[-1].append(self.location)
            self.location += 1
            self.litTabPtr += 1
            if i < len(self.literalTable) - 1:
                self.IC.append([])
        self.poolTable.append(self.litTabPtr)
    elif line[1] == "ORIGIN":
        self.location = self.calculateLocation(line[2])
        self.IC[-1].append(("AD",3))
        self.IC[-1].append(("C",self.location))
    elif line[1] == "EQU":
        newlocation =  self.calculateLocation(line[2])
        self.symbolTable[line[0]] = newlocation
        self.IC[-1].append(("AD",4))
        self.IC[-1].append(("C",newlocation))
    elif line[1] == "DC":
        self.IC[-1].append(("DL",1))
        self.IC[-1].append(("C",int(line[2])))
        self.IC[-1].append(self.location)
        self.location += 1
    elif line[1] == "DS":
        self.IC[-1].append(("DL",2))
        self.IC[-1].append(("C",int(line[2])))
        self.IC[-1].append(self.location)
        self.location += int(line[2])
    elif line[1] == "STOP":
        self.IC[-1].append(("IS",0))
```

```
            self.IC[-1].append(self.location)
            self.location += 1
        elif line[1] == "END":
            self.IC[-1].append(("AD",2))
            if self.litTabPtr != len(self.literalTable):
                # literalKeys = list(self.literalTable.keys())
                for i in range(self.poolTable[-1],len(self.literalTable)):
                    self.IC.append([])
                    self.literalTable[i][1] = self.location
                    self.IC[-1].append(("DL",1))
                    self.IC[-1].append(("C",self.literalTable[i][0]))
                    self.IC[-1].append(self.location)
                    self.location += 1
                    self.litTabPtr += 1
                self.poolTable.append(self.litTabPtr)
        elif line[1] == "PRINT":
            self.IC[-1].append(("IS",10))
            symTabKeys = list(self.symbolTable.keys())
            self.IC[-1].append(("S",symTabKeys.index(line[2])))
            self.IC[-1].append(self.location)
            self.location += 1
        elif line[1] == "READ":
            self.IC[-1].append(("IS",9))
            self.symbolTable[line[2]] = None
            symTabKeys = list(self.symbolTable.keys())
            self.IC[-1].append(("S",symTabKeys.index(line[2])))
            self.IC[-1].append(self.location)
            self.location += 1
        elif line[1] == "BC":
            self.IC[-1].append(("IS",7))
            classType = self.lookup.getClassType(line[2])
```

```
        machineCode = self.lookup.getMachineCode(line[2])
        self.IC[-1].append((classType,machineCode))
        if line[3]  not in self.symbolTable:
            self.symbolTable[line[3]] = None
        symTabKeys = list(self.symbolTable.keys())
        self.IC[-1].append(("S",symTabKeys.index(line[3])))
        self.IC[-1].append(self.location)
        self.location += 1
    else:
        #For Opcode
        classType = self.lookup.getClassType(line[1])
        machineCode = self.lookup.getMachineCode(line[1])
        self.IC[-1].append((classType,machineCode))

        #For Operand1
        classType = self.lookup.getClassType(line[2])
        machineCode = self.lookup.getMachineCode(line[2])
        self.IC[-1].append((classType,machineCode))

        #For Operand2
        if "=" in line[3]:
            constant = line[3].strip("=")
            constant = int(constant.strip("'"))
            self.literalTable[self.litTableIndex] = [constant,None]
            self.IC[-1].append(("L",self.litTableIndex))
            self.IC[-1].append(self.location)
            self.litTableIndex += 1
        else:
            if line[3] in self.symbolTable:
                symbolTableKeys = list(self.symbolTable.keys())
                self.IC[-1].append(("S",symbolTableKeys.index(line[3])))
```

```
                self.IC[-1].append(self.location)
            else:
                self.symbolTable[line[3]] = None
                symbolTableKeys = list(self.symbolTable.keys())
                self.IC[-1].append(("S",symbolTableKeys.index(line[3])))
                self.IC[-1].append(self.location)

        self.location += 1

    self.printLiteralTable()
    self.printSymbolTable()
    self.printPoolTable()
    self.printIntermdeiateCode()

  def printLiteralTable(self):
    tab = "\t"
    endline = "\n"
    print("\nLITERAL TABLE:")
    for item in range(len(self.literalTable)):
        line = str(item) + tab + str(self.literalTable[item][0]) + tab +str(self.literalTable[item][1]) +
endline;
        print(line,end="")
        self.literalTableFile.write(line)
    self.literalTableFile.close()
    print("\n")


  def printSymbolTable(self):
    tab = "\t"
    endline = "\n"
    print("\nSYMBOL TABLE:")
```

```
    for index,item in enumerate(self.symbolTable):
        line = str(index) + tab + str(item) + tab + str(self.symbolTable[item]) + endline
        print(line,end="")
        self.symbolTableFile.write(line)
    self.symbolTableFile.close()
    print("\n")

def printPoolTable(self):
    tab = "\t"
    endline = "\n"
    print("\nPOOL TABLE:")
    for item in range(len(self.poolTable)):
        print(self.poolTable[item])
        self.poolTableFile.write(str(self.poolTable[item]) + endline)
    self.poolTableFile.close()

def printIntermdeiateCode(self):
    tab = "\t"
    endline = "\n"
    print("\nIntermediate Code:")
    for item in self.IC:
        line = ""
        for i in range(len(item)):
            line += str(item[i])
            if i != len(item):
                line += tab
        line += endline
        print(line,end="")
        self.ICFile.write(line)
    self.ICFile.close()
```

```
obj = pass1()
obj.parseFile()


LITERAL TABLE:
0 5 211
1 1 212
2 1 219
SYMBOL TABLE:
0 A 217
1 LOOP 202
2 B 218
3 NEXT 214
4 BACK 202
5 LAST 216
POOL TABLE:
0
2
3
Intermediate Code:
('AD', 1) ('C', 200)
('IS', 4) ('RG', 1) ('L', 0) 200
('IS', 5) ('RG', 1) ('S', 0) 201
('IS', 4) ('RG', 1) ('S', 0) 202
('IS', 4) ('RG', 3) ('S', 2) 203
('IS', 1) ('RG', 3) ('L', 1) 204
('IS', 4) ('RG', 1) ('S', 0) 205
('IS', 4) ('RG', 3) ('S', 2) 206
('IS', 4) ('RG', 1) ('S', 0) 207
('IS', 4) ('RG', 3) ('S', 2) 208
('IS', 4) ('RG', 1) ('S', 0) 209
```

('IS', 7) ('CC', 6) ('S', 3) 210

('DL', 1) ('C', 5) 211

('DL', 1) ('C', 1) 212

('IS', 4) ('RG', 1) ('S', 0) 213

('IS', 2) ('RG', 1) ('L', 2) 214

('IS', 7) ('CC', 1) ('S', 4) 215

('IS', 0) 216

('AD', 3) ('C', 204)

('IS', 3) ('RG', 3) ('S', 2) 204

('AD', 3) ('C', 217)

('DL', 2) ('C', 1) 217

('AD', 4) ('C', 202)

('DL', 2) ('C', 1) 218

('AD', 2)

('DL', 1) ('C', 1) 219