**ASSIGNMENT A2**

Aim: Implementation of pass2 assembler

Pass2.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package pass2;

import java.io.BufferedReader;

import java.io.FileInputStream;

import java.io.FileWriter;

import java.io.InputStreamReader;

import java.io.PrintWriter;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Map;

import java.util.StringTokenizer;

class Tuple {

//m\_class specifies class of the mnemonic such as IS, DL, or AD

String mnemonic, m\_class, opcode;

int length;

Tuple() {}

Tuple(String s1, String s2, String s3, String s4) {

mnemonic = s1;

m\_class = s2;

opcode = s3;

length = Integer.parseInt(s4);

}

}

class SymTuple {

String symbol, address,length;

SymTuple(String s1, String s2, String i1) {

symbol = s1;

address = s2;

length = i1;

}

}

class LitTuple {

String literal, address,length;

LitTuple() {}

LitTuple(String s1, String s2, String i1) {

literal = s1;

address = s2;

length = i1;

}

}

public class pass2 {

static int lc,iSymTabPtr=0, iLitTabPtr=0, iPoolTabPtr=0;

static int poolTable[] = new int[10];

static Map<String,Tuple> MOT;

static ArrayList<SymTuple> symtable;

static ArrayList<LitTuple> littable;

static Map<String, String> regAddressTable;

static PrintWriter out\_pass2;

static void initializeTables() throws Exception{

symtable=new ArrayList<>();

littable=new ArrayList<>();

regAddressTable =new HashMap<>();

String s;

BufferedReader br;

br=new BufferedReader(new InputStreamReader(new FileInputStream("symtable.txt")));

while ((s=br.readLine())!=null) {

StringTokenizer st=new StringTokenizer(s,"\t",false);

symtable.add(new SymTuple(st.nextToken(), st.nextToken(), ""));

}

br.close();

br=new BufferedReader(new InputStreamReader(new FileInputStream("littable.txt")));

while((s=br.readLine())!=null)

{

StringTokenizer st=new StringTokenizer(s,"\t",false);

littable.add(new LitTuple(st.nextToken(),st.nextToken(),""));

}

br.close();

regAddressTable.put("AREG","1");

regAddressTable.put("BREG","2");

regAddressTable.put("CREG","3");

regAddressTable.put("DREG","4");

}

static void pass2() throws Exception

{

BufferedReader input=new BufferedReader(new InputStreamReader(new FileInputStream("output\_pass1.txt")));

out\_pass2=new PrintWriter(new FileWriter("output\_pass2.txt"),true);

String s;

while((s=input.readLine())!=null)

{

s=s.replaceAll("(\\()"," ");

s=s.replaceAll("(\\))"," ");

String ic\_tokens[]=tokenizeString(s," ");

if(ic\_tokens==null || ic\_tokens.length==0)

{

continue;

}

String output\_str="";

String mnemonic\_class=ic\_tokens[1];

String m\_tokens[]=tokenizeString(mnemonic\_class,",");

if(m\_tokens[0].equalsIgnoreCase("IS"))

{

output\_str +=ic\_tokens[0] + " ";

output\_str +=m\_tokens[1]+" ";

String opr\_tokens[];

for(int i=2;i<ic\_tokens.length;i++)

{

opr\_tokens=tokenizeString(ic\_tokens[i],",");

if(opr\_tokens[0].equalsIgnoreCase("RG"))

{

output\_str +=opr\_tokens[1]+" ";

}

else if(opr\_tokens[0].equalsIgnoreCase("S"))

{

int index=Integer.parseInt(opr\_tokens[1]);

output\_str+= symtable.get(index).address+" ";

}

else if(opr\_tokens[0].equalsIgnoreCase("L"))

{

int index=Integer.parseInt(opr\_tokens[1]);

output\_str+=littable.get(index).address+" ";

}

}

}

else if(m\_tokens[0].equalsIgnoreCase("DL")){

output\_str+=ic\_tokens[0]+"";

if(m\_tokens[1].equalsIgnoreCase("02"))

{

String opr\_tokens[]=tokenizeString(ic\_tokens[2],",");

output\_str += "00 00" +opr\_tokens[1] + " ";

}

}

System.out.println(output\_str);

out\_pass2.println(output\_str);

}

}

static String[] tokenizeString(String str,String separator){

StringTokenizer st = new StringTokenizer(str,separator,false);

String s\_arr[]=new String[st.countTokens()];

for(int i=0;i<s\_arr.length;i++){

s\_arr[i]=st.nextToken();

}

return s\_arr;

}

public static void main(String[] args) throws Exception{

initializeTables();

pass2();

}

}

**INPUTS:**

**PASS1 IC:**

0 (AD,01) (C,100)

100 (IS,04) (RG,1) (S,0)

101 (IS,01) (RG,2) (L,0)

102 (IS,05) (RG,1) (S,1)

103 (IS,02) (RG,3) (L,1)

104 (DL,02) (C,6)

105 (DL,02) (C,1)

106 (IS,01) (RG,4) (L,2)

107 (DL,01) (C,10)

117 (DL,02) (C,5)

118 (IS,02) (RG,1) (L,3)

119 (DL,02) (C,1)

120 (DL,02) (C,1)

121 (AD,02)

121 (DL,02) (C,1)

**SYMTABLE:**

B 119

A 107

C 120

**LIT TABLE:**

6 104

1 105

5 117

1 121

OUTPUT:

100 04 1 119

101 01 2 104

102 05 1 107

103 02 3 105

104 00 006

105 00 001

106 01 4 117

107

117 00 005

118 02 1 121

119 00 001

120 00 001

121 00 001