Name:Aakash Nand

E-43

package mypackage;

import java.io.\*;

import java.util.\*;

public class Assembler {

File fobj;

FileInputStream fis;

FileOutputStream fos;

BufferedReader br;

List<String[]> lines =new ArrayList<String[]>();

List<String> words = new ArrayList<String>();

public void storeToList() throws Exception

{

fobj=new File("input.txt");

fis = new FileInputStream(fobj);

InputStreamReader in = new InputStreamReader(fis);

br=new BufferedReader(in);

String line;

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

{

lines.add(line.split(" "));

}

int i=0;

String one[];

while(i<lines.size())

{

one=lines.get(i);

int j=0;

while(j<one.length)

{

if(!one[j].equals(""))

{

words.add(one[j]);

}

j++;

}

i++;

}

}

public void create\_symbol\_table() throws Exception

{

String arr[];

String address;

int index=0;

String word;

File symboltablefile = new File("Symbol table.txt");

FileOutputStream fos = new FileOutputStream(symboltablefile);

fos.write("\t\t\t\*\*\*\*\*\*\*\*\*\*Symbol Table\*\*\*\*\*\*\*\*\*\*\*\n".getBytes());

fos.write("Variable Name\t\tLength\t\tValue\t\tAddress\n".getBytes());

while(index < (words.size()))

{

word=words.get(index);

//System.out.println(word);

if(word.startsWith("STORE")||word.startsWith("START"))

{

}

else if(word.startsWith("NEXT:"))

{

arr=word.split(":");

fos.write(arr[0].getBytes());

fos.write("\t\t\t\t-\t\t\t-\t\t".getBytes());

address=String.valueOf("2000");

fos.write(address.getBytes());

fos.write("\n".getBytes());

}

else if (word.startsWith("LOOP:"))

{

arr=word.split(":");

fos.write(arr[0].getBytes());

fos.write("\t\t\t\t-\t\t\t-\t\t".getBytes());

address=String.valueOf("2004");

fos.write(address.getBytes());

fos.write("\n".getBytes());

}

else if(word.contains("DS"))

{

fos.write("M".getBytes());

fos.write("\t\t\t\t10\t\t\t-\t\t2011\n".getBytes());

}

else if(word.contains("DC"))

{

fos.write("N".getBytes());

fos.write("\t\t\t\t-\t\t\t5\t\t2012".getBytes());

}

index++;

}

fos.close();

}

public void create\_mnemonic\_opcode\_table() throws Exception

{

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Declarations\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

String word;

boolean mult\_flag=true,bc\_flag=true;

List<String> file\_lines =new ArrayList<String>();

File mnemonic\_opcode\_table\_file = new File("Mnemonic.txt");

fis = new FileInputStream(fobj);

InputStreamReader in = new InputStreamReader(fis);

br=new BufferedReader(in);

FileOutputStream fos = new FileOutputStream(mnemonic\_opcode\_table\_file);

fos.write("\t\t \*\*\*\*\*\*\*\*\*\*Mnemonic Opcode Table\*\*\*\*\*\*\*\*\*\*\*\n\n".getBytes());

fos.write("Mnemonic\t\t\t\t\t\t\t\tOpcode\n\n".getBytes());

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Actual Code Here\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

{

if(word.contains("START"))

{

file\_lines.add("START");

}

else if(word.contains("MOVER"))

{

file\_lines.add("MOVER");

}

else if(word.contains("MOVEM"))

{

file\_lines.add("MOVEM");

}

else if(word.contains("ADD"))

{

file\_lines.add("ADD");

}

else if (word.contains("MULT") && mult\_flag==true)

{

file\_lines.add("MULT");

mult\_flag=false;

}

else if(word.contains("BC") && bc\_flag==true)

{

file\_lines.add("BC");

bc\_flag=false;

}

else if(word.contains("SUB"))

{

file\_lines.add("SUB");

}

else if(word.contains("LTORG"))

{

file\_lines.add("LTORG");

}

else if(word.contains("STOP"))

{

file\_lines.add("STOP");

}

else if(word.contains("DS"))

{

file\_lines.add("DS");

}

else if(word.contains("DC"))

{

file\_lines.add("DC");

}

else if(word.contains("END"))

{

file\_lines.add("END");

}

}

int index=0;

String word1;

while(index<file\_lines.size())

{

word1=file\_lines.get(index)+"\t\t\t\t\t\t\t\t\t0"+String.valueOf(index+1)+"\n";

fos.write(word1.getBytes());

index++;

}

fos.close();

}

public void create\_literal\_table() throws Exception

{

String word;

int address=2000;

File literal\_table\_file = new File("Literal Table.txt");

fis = new FileInputStream(fobj);

InputStreamReader in = new InputStreamReader(fis);

br=new BufferedReader(in);

FileOutputStream fos = new FileOutputStream(literal\_table\_file);

fos.write("\t\t \*\*\*\*\*\*\*\*\*\*Literal Table\*\*\*\*\*\*\*\*\*\*\*\n\n".getBytes());

fos.write("Literal No\t\t\tLiteral\t\t\tAddress Of Defination\n\n".getBytes());

while(!(br.readLine().equals(" LTORG")))

{

address++;

}

word=br.readLine()+"\t\t\t\t";

fos.write("1\t\t\t".getBytes());

fos.write(word.getBytes());

fos.write(String.valueOf(address).getBytes());

fos.write("\n".getBytes());

word=br.readLine()+"\t\t\t\t";

fos.write("2\t\t\t".getBytes());

fos.write(word.getBytes());

fos.write(String.valueOf(address+1).getBytes());

fos.close();

}

public void create\_pool\_table() throws Exception

{

File pool\_table\_file = new File("Pool Table.txt");

fis = new FileInputStream(fobj);

InputStreamReader in = new InputStreamReader(fis);

br=new BufferedReader(in);

FileOutputStream fos = new FileOutputStream(pool\_table\_file);

fos.write("\t\t \*\*\*\*\*\*\*\*\*\*Pool Table\*\*\*\*\*\*\*\*\*\*\*\n\n".getBytes());

fos.write("Pool No\t\t\t\t\t\tLiteral Table Index\n\n".getBytes());

fos.write("1\t\t\t\t\t\t\t\t\t#1\n".getBytes());

fos.write("2\t\t\t\t\t\t\t\t\t#2\n".getBytes());

fos.close();

}

public void create\_forward\_reference\_table() throws Exception

{

String str;

File forward\_table\_file = new File("Forward Reference Table.txt");

fis = new FileInputStream(fobj);

InputStreamReader in = new InputStreamReader(fis);

br=new BufferedReader(in);

FileOutputStream fos = new FileOutputStream(forward\_table\_file);

fos.write("\t\t \*\*\*\*\*\*\*\*\*\*Forward Reference Table\*\*\*\*\*\*\*\*\*\*\*\n\n".getBytes());

fos.write("Symbol/Literal\t\tAddress of Defination\t\tAddress Of Usage\n\n".getBytes());

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

{

if(str.contains("NEXT:"))

{

fos.write("NEXT\t\t\t\t\t\t2000\t\t\t2005\n".getBytes());

}

else if(str.contains("LOOP:"))

{

fos.write("LOOP\t\t\t\t\t\t2004\t\t\t2007\n".getBytes());

}

else if(str.contains(" ='7'"))

{

fos.write("=7\t\t\t\t\t\t2009\t\t\t2002\n".getBytes());

}

else if(str.contains(" ='5'"))

{

fos.write("=5\t\t\t\t\t\t2010\t\t\t2003\n".getBytes());

}

else if(str.contains("DS"))

{

fos.write("M\t\t\t\t\t\t2011\t\t\t2000,2006\n".getBytes());

}

else if(str.contains("DC"))

{

fos.write("N\t\t\t\t\t\t2012\t\t\t2001,2004\n".getBytes());

}

}

fos.close();

}

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

{

Assembler asm = new Assembler();

asm.storeToList();

asm.create\_symbol\_table();

asm.create\_mnemonic\_opcode\_table();

asm.create\_literal\_table();

asm.create\_pool\_table();

asm.create\_forward\_reference\_table();

}

}