Opcode.java

```
package pract;

import java.util.HashMap;
import java.util.Map;

public class opcodes {
 Map data=new HashMap();
 Map mnemonic=new HashMap();
 Map directive=new HashMap();
 Map condition=new HashMap();
 Map register=new HashMap();

 public opcodes()
 {
  register.put("AREG", "31");
  register.put("BREG", "32");
  register.put("CREG", "33");
  register.put("DREG", "34");

  condition.put("LT", "21");
  condition.put("GT", "22");
  condition.put("LTE", "23");
  condition.put("GTE", "24");
  condition.put("EQU", "25");

  directive.put("START", "1");
  directive.put("END","2");
  directive.put("ORIGIN","3");
  directive.put("LTORG","4");

  mnemonic.put("ADD", "1");
  mnemonic.put("SUB", "2");
  mnemonic.put("MUL", "3");
  mnemonic.put("DIV", "4");
  mnemonic.put("MOVER", "5");
  mnemonic.put("MOVEM", "6");
  mnemonic.put("READ", "7");

  data.put("DS", "1");
  data.put("DC", "2");



 }

}
```

Passone.java

```
package pract;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.Map;

public class passone {
public static void main(String[] args) throws IOException {
  // TODO Auto-generated method stub
  FileReader file;
  //File f=new File("input.asm");
  BufferedReader reader;
  String line;
  int lc=0,sindex=0,lindex=0;
  int lcount=0;

  String op[][]=new String[10][4];
  opcodes OPTAB=new opcodes();
  Map symtab=new HashMap();
  Map littab=new HashMap();
  Map pooltab=new HashMap();
  /*boolean result;
  result=f.createNewFile();
  if(result)
  {
   System.out.print("File created successfully"+f.getCanonicalPath());
  }*/
   file = new FileReader("input.asm");
   reader = new BufferedReader(file);


     while((line = reader.readLine()) != null)
     {
      //System.out.print(line);
      String split_words[] = line.split("\t");
      if(lcount==0)
      {
       lc=Integer.parseInt(split_words[2]);
      // System.out.print(split_words[1]+"\t"+split_words[2]);
       op[lcount][0]=split_words[1];
       op[lcount][1]="(AD,1)";

      }
      else {
       if(split_words[0]!=null)
       {
```

```
 //op[lcount][3]="(S,"+sindex+")";
 if(!symtab.containsKey(split_words[0]))
 {
  sindex++;
  symtab.put(split_words[0], lc);

 }

 //symtab.put(split_words[0],lc);

}
if(OPTAB.mnemonic.containsKey(split_words[1]))
{
 op[lcount][0]=split_words[1];
 op[lcount][1]="(IS,"+OPTAB.mnemonic.get(split_words[1])+ ")";
 op[lcount][2]= "("+OPTAB.register.get(split_words[2])+ ")";
 if(split_words[3].contains("="))
 {
  int pt=0;
  lindex++;
  op[lcount][3]="(C,"+lindex+")";
  pt=Integer.parseInt(split_words[3].substring(1));
  lc++;
  littab.put(split_words[3],lc);
  //lc=lc+1;
 }else if(split_words[3]!=null)
 {
  if(!symtab.containsKey(split_words[3]))
  {
   sindex++;
   symtab.put(split_words[3], lc);
  }

  op[lcount][3]="(S,"+sindex+")";

  //lc++;
 }

}
else if(OPTAB.data.containsKey(split_words[1]))
{
 op[lcount][0]=split_words[1];
       op[lcount][1]="(DL,"+OPTAB.data.get(split_words[1])+")";
 op[lcount][2]= "(C,"+split_words[2]+ ")";
    //LC= LC+ split_words[2]
  lc=lc+Integer.parseInt(split_words[2]);
}
else if (OPTAB.directive.containsKey(split_words[1]))
 {
  op[lcount][0]=split_words[1];
  op[lcount][1]="(AD,"+OPTAB.directive.get(split_words[1])+")";
       //add (AD,2) or (AD,3) or (AD,4) to opcode[line_count][1]
 }
}
```

```
 lcount++;
}
System.out.println("OPCODE TABLE");
System.out.println("_____________________________________________________");
System.out.println("Mnemonic"+" "+"class"+"\t"+"Info");

for(int i=0;i<lcount;i++)
{
 System.out.print(op[i][0]+"\t"+op[i][1]);
 if(op[i][2]!=null)
  System.out.print("\t"+op[i][2]);
 else
  System.out.print("\t"+"     ");
 if(op[i][3]!=null)
  System.out.print("\t"+op[i][3]);
 else
 System.out.print("\t"+"     ");
   System.out.println();

}
System.out.println();
System.out.println("SYMBOL TABLE");
System.out.println("_____________________________________________________");
int v=0;
System.out.println("symbol"+"\t"+"Address"+"\t"+"index");
for (Object name : symtab.keySet())
   {
  v++;
 if(v==1)
  continue;


      // search  for value
      Object url = symtab.get(name);
      System.out.println(name +"\t" + url+"\t"+v);
   }
System.out.println();
System.out.println("Literal TABLE");
System.out.println("_____________________________________________________");
System.out.println("Literal"+"\t"+"Address");
for (Object name : littab.keySet())
   {
      // search  for value
      Object url = littab.get(name);
      System.out.println( name + "\t" + url);
   }
}

}
```

Output :

OPCODE TABLE

Mnemonic class Info
START (AD,1)
DS (DL,1) (C,2)
ADD (IS,1) (31) (C,1)
LTORG (AD,4)
MUL (IS,3) (32) (S,3)
END (AD,2)

SYMBOL TABLE

symbol	Address	index
A	200	2
B	202	3

Literal TABLE

Literal	Address
=10	203