Program:
PASS 1:

```
import java.io.*;

import java.util.*;

class MnemonicTable {

          public String mnemonic;

          public String opcode;

          public int num;

          public MnemonicTable(String mnemonic,String opcode,int num ){

                    this.mnemonic=mnemonic;

                    this.opcode=opcode;

                    this.num=num;

          }
}

public class Pass_1 {

   Map<String,MnemonicTable> is=new Hashtable<String,MnemonicTable>();

   ArrayList<String>symtab=new ArrayList<>();

   ArrayList<Integer> symaddr=new ArrayList<>();

   ArrayList<String>littab=new ArrayList<>();

   ArrayList<Integer> litaddr=new ArrayList<>();

   ArrayList<Integer>pooltab=new ArrayList<>();

   int LC=0;

   public void createIS() throws Exception {

         Scanner input=new Scanner(System.in);

      MnemonicTable m1=new MnemonicTable("STOP","00", 0);

      is.put("STOP",m1);

      MnemonicTable m2=new MnemonicTable("ADD","01", 0);

      is.put("ADD",m2);

      MnemonicTable m3=new MnemonicTable("SUB","02", 0);
```

```
    is.put("SUB",m3);
    MnemonicTable m4=new MnemonicTable("MULT","03", 0);
    is.put("MULT",m4);
    MnemonicTable m5=new MnemonicTable("MOVER","04", 0);
    is.put("MOVER",m5);
    MnemonicTable m6=new MnemonicTable("MOVEM","05", 0);
    is.put("MOVEM",m6);
    MnemonicTable m7=new MnemonicTable("COMP","06", 0);
    is.put("COMP",m7);
    MnemonicTable m8=new MnemonicTable("BC","07", 0);
    is.put("BC",m8);
    MnemonicTable m9=new MnemonicTable("DIV","08", 0);
    is.put("DIV",m9);
    MnemonicTable m10=new MnemonicTable("READ","09", 0);
    is.put("READ",m10);
    MnemonicTable m11=new MnemonicTable("PRINT","10", 0);
      is.put("PRINT",m11);
      /*BufferedWriter wr=new BufferedWriter(new FileWriter("ic.txt"));
      String string=input.next();
      wr.write(string);
      wr.flush();
      wr.close();      */
}
public void generateIC() throws Exception {
    BufferedWriter wr=new BufferedWriter(new FileWriter("ic.txt"));
    BufferedReader br=new BufferedReader(new FileReader("input.asm"));
    String line=" ";
    pooltab.add(0, 0);
    wr.write("--------------------\n  Intermediate Code\n-------------------- \n");
    while((line=br.readLine())!=null) {


      String[] split=line.split("\\s+");
```

```
if(split[0].length()>0) {
    //it is a label
    if(!symtab.contains(split[0])) {
        symtab.add(split[0]);
        symaddr.add(LC);
    }
    else {
        int index=symtab.indexOf(split[0]);
        symaddr.remove(index);
        symaddr.add(index,LC);
    }
}

if(split[1].equals("START")) {
    LC=Integer.parseInt(split[2]);
    wr.write("(AD,01)(C,"+split[2]+") \n");
}
else if(split[1].equals("ORIGIN")) {
    if(split[2].contains("+") || split[2].contains("-")) {
        LC=getAddress(split[2]);
    }
    else {
        LC=symaddr.get(symtab.indexOf(split[2]));
    }
}
else if(split[1].equals("EQU")) {
    int addr=0;
    if(split[2].contains("+") || split[2].contains("-")) {
        addr=getAddress(split[2]);
    }
    else {
        addr=symaddr.get(symtab.indexOf(split[2]));
    }
    if(!symtab.contains(split[0])) {
        symtab.add(split[0]);
        symaddr.add(addr);
    }
    else {
        int index=symtab.indexOf(split[0]);
        symaddr.remove(index);
        symaddr.add(index,addr);
    }
}
else if(split[1].equals("LTORG") || split[1].equals("END")) {
    if(litaddr.contains(0)) {
        for(int i=pooltab.get(pooltab.size()-1);i<littab.size();i++) {
            if(litaddr.get(i)==0) {
                litaddr.remove(i);
                litaddr.add(i, LC);
                LC++;
            }
        }
        if(!split[1].equals("END")) {
            pooltab.add(littab.size());
            wr.write("\n(AD,05)\n");
        }
        else
            wr.write("(AD,04) \n");
    }
}
else if(split[1].contains("DS")) {
    LC+=Integer.parseInt(split[2]);
    wr.write("(DL,01) (C,"+split[2]+") \n");
}
```

```
else if(split[1].equals("DC")) {
   LC++;
   wr.write("\n(DL,02) (C,"+split[2].replace("'", "").replace("'", "")+") \n");
}
else if(is.containsKey(split[1])) {
   wr.write("(IS,"+is.get(split[1]).opcode+") ");
   if(split.length>2 && split[2]!=null) {
      String reg=split[2].replace(",","");
      if(reg.equals("AREG")) {
         wr.write("(1) ");
      }
      else if(reg.equals("BREG")) {
         wr.write("(2) ");
      }
      else if(reg.equals("CREG")) {
         wr.write("(3) ");
      }
      else if(reg.equals("DREG")) {
         wr.write("(4) ");
      }
      else {
         if(symtab.contains(reg)) {
            wr.write("(S,"+symtab.indexOf(reg)+")\n");
         }
         else {
            symtab.add(reg);
            symaddr.add(0);
            wr.write("(S,"+symtab.indexOf(reg)+") \n");


         }
      }
   }
```

```
        if(split.length>3 && split[3]!=null) {
            if(split[3].contains("=")) {
                String norm=split[3].replace("=","").replace("'", "").replace("'", "");
                if(!littab.contains(norm)) {
                    littab.add(norm);
                    litaddr.add(0);
                    wr.write("(L,"+littab.indexOf(norm)+")");
                }
                else {
                    wr.write("L,"+littab.indexOf(norm)+")");
                }

            }
            else if(symtab.contains(split[3])) {
                wr.write("(S,"+symtab.indexOf(split[3])+") \n");

            }
            else {
                symtab.add(split[3]);
                symaddr.add(0);
                wr.write("(S,"+symtab.indexOf(split[3])+") \n");

            }
        }
        LC++;
    }
}
wr.flush();
BufferedReader icr=new BufferedReader(new FileReader("ic.txt"));
while(icr.ready()){
System.out.print((char)icr.read());
```

```
    }
    icr.close();
    wr.close();
    BufferedWriter br1=new BufferedWriter(new FileWriter("sym.txt"));
    br1.write("------------------\n    Symbol Table\n------------------\nSymbol    Address\n");
    for(int i=0;i<symtab.size();i++) {
      br1.write("  "+symtab.get(i)+"       "+symaddr.get(i)+"\n");
    }
    br1.flush();
    BufferedReader br1r=new BufferedReader(new FileReader("sym.txt"));
    while(br1r.ready()){
    System.out.print((char)br1r.read());
    }
    br1r.close();
    br1.close();
    BufferedWriter br2=new BufferedWriter(new FileWriter("lit.txt"));
    br2.write("----------------------\n    Literal Table\n ----------------------\nLiteral
Address\n");
    for(int i=0;i<littab.size();i++) {
      br2.write("='"+littab.get(i)+"'          "+litaddr.get(i)+"\n");
    }
    br2.flush();
    BufferedReader br2r=new BufferedReader(new FileReader("lit.txt"));
    while(br2r.ready()){
    System.out.print((char)br2r.read());
    }
    br2r.close();
    br2.close();
    BufferedWriter br3=new BufferedWriter(new FileWriter("pool.txt"));
    BufferedReader br3r=new BufferedReader(new FileReader("pool.txt"));
    br3.write("______________________________\n        Pool Table\n______________________________\nPool Index
Literal Index\n");
    for(int i=0;i<pooltab.size();i++){
            br3.write("     "+i+"                "+pooltab.get(i)+"\n");
        }
        br3.flush();
        while(br3r.ready()){
            System.out.print((char)br3r.read());
        }
        br3r.close();

    }
    private int getAddress(String string) {
        int temp=0;
        if(string.contains("+")) {
            String sp[]=string.split("\\+");
            int ad=symaddr.get(symtab.indexOf(sp[0]));
            temp=ad+Integer.parseInt(sp[1]);
        }
        else if(string.contains("-")) {
            String sp[]=string.split("\\-");
            int ad=symaddr.get(symtab.indexOf(sp[0]));
            temp=ad-Integer.parseInt(sp[1]);
        }
        return temp;
    }
    public static void main(String[] args) throws Exception {
        Pass_1 p=new Pass_1();
        p.createIS();
        p.generateIC();
    }
}
```

Input:

```
START    100
A    DS    3
L1    MOVEM    AREG,    B
   ADD    AREG,    C
   MOVER    AREG,    ='12'
D    EQU    A+1
   LTORG
L2    PRINT    D
   ORIGIN     A-1
   MOVER    AREG,    ='5'
C     DC     '5'
   ORIGIN    L2+1
   STOP
B    DC    '19'
   END
```

Output:

```
--------------------
  Intermediate Code
--------------------
(AD,01)(C,100)
(DL,01) (C,3)
(IS,05) (1) (S,2)
(IS,01) (1) (S,3)
(IS,04) (1) (L,0)
(AD,05)
(IS,10) (S,4)
(IS,04) (1) (L,1)
(DL,02) (C,5)
(IS,00)
(DL,02) (C,19)
(AD,04)
------------------
   Symbol Table
--------------------
Symbol    Address
  A       100
  L1       103
  B       109
  C       100
  D       101
  L2       107
---------------------
   Literal Table
- - - - - - - - - - - - -
Literal      Address
='12'         106
='5'         110
----------------------------
       Pool Table
----------------------------
Pool Index    Literal Index
    0             0
    1             1
```

PASS 2:

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.util.ArrayList;

class TableRow {
	String symbol;
	int  address;
	int index;

	public TableRow(String symbol, int address) {
		super();
		this.symbol = symbol;
		this.address = address;

	}
	public TableRow(String symbol, int address,int index) {
		super();
		this.symbol = symbol;
		this.address = address;
		this.index=index;

	}
	public int getIndex() {
		return index;
	}
	public void setIndex(int index) {
		this.index = index;
	}
	public String getSymbol() {
		return symbol;
	}
	public void setSymbol(String symbol) {
		this.symbol = symbol;
	}
	public int getAddress() {
		return address;
	}
	public void setAddress(int address) {
		this.address = address;
	}

}
public class Pass_2 {
	ArrayList<TableRow> SYMTAB,LITTAB;

	public Pass_2()
	{
		SYMTAB=new ArrayList<>();
		LITTAB=new ArrayList<>();
	}
	public static void main(String[] args) {
		Pass_2 pass2=new Pass_2();

		try {
			pass2.generateCode("IC.txt");
		} catch (Exception e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}
	public void readtables()
	{
		BufferedReader br;
		String line;
		try
		{
			br=new BufferedReader(new FileReader("SYMTAB.txt"));
			while((line=br.readLine())!=null)
			{
				String parts[]=line.split("\\s+");
				SYMTAB.add(new TableRow(parts[1],
Integer.parseInt(parts[2]),Integer.parseInt(parts[0]) ));
			}
			br.close();
			br=new BufferedReader(new FileReader("LITTAB.txt"));
			while((line=br.readLine())!=null)
			{
				String parts[]=line.split("\\s+");
				LITTAB.add(new TableRow(parts[1],
Integer.parseInt(parts[2]),Integer.parseInt(parts[0])));
			}
			br.close();
		}
		catch (Exception e) {
			System.out.println(e.getMessage());
		}
	}

	public void generateCode(String filename) throws Exception
	{
		readtables();
		BufferedReader br=new BufferedReader(new FileReader(filename));

		BufferedWriter bw=new BufferedWriter(new FileWriter("PASS2.txt"));
		String line,code;
		while((line=br.readLine())!=null)
		{
			String parts[]=line.split("\\s+");
			if(parts[0].contains("AD")||parts[0].contains("DL,02"))
			{
				bw.write("\n");
				continue;
			}
			else if(parts.length==2)
			{
				if(parts[0].contains("DL")) //DC INSTR
```

```
                                        {
                                                parts[0]=parts[0].replaceAll("[^0-9]", "");
                                                if(Integer.parseInt(parts[0])==1)
                                                {
                                                        int
constant=Integer.parseInt(parts[1].replaceAll("[^0-9]", ""));
                                                        code="00\t0\t"+String.format("%03d",
constant)+"\n";
                                                        bw.write(code);


                                                }
                                        }
                                        else if(parts[0].contains("IS"))
                                        {
                                                int opcode=Integer.parseInt(parts[0].replaceAll("[^0-9]",
""));
                                                if(opcode==10)
                                                {
                                                        if(parts[1].contains("S"))
                                                        {
                                                                int
symIndex=Integer.parseInt(parts[1].replaceAll("[^0-9]", ""));
                                                                code=String.format("%02d",
opcode)+"\t0\t"+String.format("%03d", SYMTAB.get(symIndex-1).getAddress())+"\n";
                                                                bw.write(code);
                                                        }
                                                else if(parts[1].contains("L"))
                                                        {
                                                                int
symIndex=Integer.parseInt(parts[1].replaceAll("[^0-9]", ""));
                                                                code=String.format("%02d",
opcode)+"\t0\t"+String.format("%03d", LITTAB.get(symIndex-1).getAddress())+"\n";
                                                                bw.write(code);
                                                        }

                                                }
                                        }
                                }
                                else if(parts.length==1 && parts[0].contains("IS"))
                                {
                                        int opcode=Integer.parseInt(parts[0].replaceAll("[^0-9]", ""));
                                        code=String.format("%02d",
opcode)+"\t0\t"+String.format("%03d", 0)+"\n";
                                        bw.write(code);
                                }
                                else if(parts[0].contains("IS") && parts.length==3) //All OTHER IS
INSTR
                                {
                                int opcode=     Integer.parseInt(parts[0].replaceAll("[^0-9]", ""));

                                int regcode=Integer.parseInt(parts[1]);

                                if(parts[2].contains("S"))
                                {
                                        int symIndex=Integer.parseInt(parts[2].replaceAll("[^0-9]", ""));
                                        code=String.format("%02d",
opcode)+"\t"+regcode+"\t"+String.format("%03d", SYMTAB.get(symIndex-
1).getAddress())+"\n";
                                        bw.write(code);
                              }
                              else if(parts[2].contains("L"))
                              {
                                        int symIndex=Integer.parseInt(parts[2].replaceAll("[^0-9]", ""));
                                        code=String.format("%02d",
opcode)+"\t"+regcode+"\t"+String.format("%03d", LITTAB.get(symIndex-
1).getAddress())+"\n";
                                        bw.write(code);
                              }
                              }
                    }
                    bw.close();
                    br.close();
                    System.out.println("Pass2 Processing done............. )");
          }
}
```

Input:

```
--------------------
  Intermediate Code
----------------
(AD,01)         (C,100)
(IS,04) 1       (L,1)
(IS,05) 2       (S,02)
(IS,01) 1       (L,2)
(DL,01)         (C,5)
(DL,01)         (C,2)
(IS,04) 1       (S,03)
(DL,01)         (C,5)
(DL,02)         (C,2)
(AD,02)
----------------
   Symbol Table
------------------
Index  Symbol Address
1      L1     100
2      X      106
3      Y      107

----------------------
   Literal Table
----------------------
Literal     Address
5           104
2           105
```

Output:

```
Pass2 Processing done.............)

Pass_2 Output-
04      1       104
05      2       106
01      1       105
00      0       005
00      0       002
04      1       107
00      0       005
```