Code:

```
package com.muthadevs;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Hashtable;
import java.util.Map;


public class Main {

   public static void main(String[] args) {

      try {

         String f = "E:\\pass1_assembler\\OUTPUT\\IC.txt";
         FileReader fw = new FileReader(f);
         BufferedReader IC_file = new BufferedReader(fw);

         String f1 = "E:\\pass1_assembler\\OUTPUT\\SYMTAB.txt";
         FileReader fw1 = new FileReader(f1);
         BufferedReader symtab_file = new BufferedReader(fw1);
         symtab_file.mark(500);

         String f2 = "E:\\pass1_assembler\\OUTPUT\\LITTAB.txt";
         FileReader fw2 = new FileReader(f2);
         BufferedReader littab_file = new BufferedReader(fw2);
         littab_file.mark(500);

         String littab[][]=new String[10][2] ;

         Hashtable<String, String> symtab = new Hashtable<String, String>();
         String str;
         int z=0;
```

```
while ((str = littab_file.readLine()) != null) {

   littab[z][0]=str.split("\t")[0];
   littab[z][1]=str.split("\t")[1];
   z++;
}
while ((str = symtab_file.readLine()) != null) {
   symtab.put(str.split("\t")[0], str.split("\t")[1]);
}

String f3 = "E:\\pass1_assembler\\OUTPUT\\POOLTAB.txt";
FileReader fw3 = new FileReader(f3);
BufferedReader pooltab_file = new BufferedReader(fw3);

String f4 = "E:\\pass2_assembler\\MACHINE_CODE.txt";
FileWriter fw4 = new FileWriter(f4);
BufferedWriter machine_code_file = new BufferedWriter(fw4);

ArrayList<Integer> pooltab = new ArrayList<Integer>();
String t;
while ((t = pooltab_file.readLine()) != null) {
   pooltab.add(Integer.parseInt(t));
}


int pooltabptr = 1;
int temp1 = pooltab.get(0);        //dry run
int temp2 = pooltab.get(1);


String sCurrentLine;
sCurrentLine = IC_file.readLine();
int locptr=0;
locptr=Integer.parseInt(sCurrentLine.split("\t")[3]);

while ((sCurrentLine = IC_file.readLine()) != null) {

   machine_code_file.write(locptr+"\t");    //always write the LC
```

```
            String s0 = sCurrentLine.split("\t")[0];

            String s1 = sCurrentLine.split("\t")[1];

            //If its imperative statement
            if (s0.equals("IS")) {
               machine_code_file.write(s1 + "\t");  //checking if its IS then we take address after it (i.e s1) and write in mc.txt file
               if (sCurrentLine.split("\t").length == 5) {

                  machine_code_file.write(sCurrentLine.split("\t")[2] + "\t"); // so if here value is present it means its a register, we directly write it in mc.txt


                  if (sCurrentLine.split("\t")[3].equals("L")) {  //if its a Literal we take index followed by L and search in littab and write it into MC.txt
                     int add = Integer.parseInt(sCurrentLine.split("\t")[4]);

                     machine_code_file.write(littab[add-1][1]); //taking address from littab and writing it into machine code

                  }


                  if (sCurrentLine.split("\t")[3].equals("S")) { //if its a symbol , we take index followed by S and search in symtab and write address associated with it to MC file
                     int add1 = Integer.parseInt(sCurrentLine.split("\t")[4]);
                     int i = 1;
                     String l1;
                     for (Map.Entry m : symtab.entrySet()) {
                        if (i == add1) {
                           machine_code_file.write((String) m.getValue());
                        }
                        i++;
                     }
```

```
                }
            } else {
                machine_code_file.write("0\t000");
            }
        }
        //If its Assembler Directive :

        if (s0.equals("AD")) {
            littab_file.reset();
            if (s1.equals("05")) {      //if it is LTORG
                int j = 1;
                while (j < temp1) {
                    littab_file.readLine();
                }
                while (temp1 < temp2) {
                    machine_code_file.write("00\t0\t00" +
littab_file.readLine().split("")[1]);
                    if(temp1<(temp2-1)){
                        locptr++;
                        machine_code_file.write("\n");
                        machine_code_file.write(locptr+"\t");
                    }
                    temp1++;
                }
                temp1 = temp2;
                pooltabptr++;
                if (pooltabptr < pooltab.size()) {
                    temp2 = pooltab.get(pooltabptr);
                }
            }
            int j = 1;
            if (s1.equals("02")) {      //if it is "END" stmt
                String s;
                while ((s = littab_file.readLine()) != null) {
                    if (j >= temp1)
                        machine_code_file.write("00\t0\t00" + s.split("")[1]);
                    j++;
                }
```

```
            }
          }

          //If its a declarative statements
          if(s0.equals("DL")&&s1.equals("01")){    //if it is DC stmt
            machine_code_file.write("00\t0\t00"+sCurrentLine.split("'")[1]);

          }

          locptr++;
          machine_code_file.write("\n");
        }
        IC_file.close();
        symtab_file.close();
        littab_file.close();
        pooltab_file.close();
        machine_code_file.close();
      } catch (IOException e) {
        e.printStackTrace();
      }

    }
}
```

Output:

Machine Code:

```
200 04 1  204
201    05 1  208
202    04 2  210
203    04 3  209
204    00 0  004
205    00 0  006
206    01 3  205
207    00 0  000
208
209
210    00 0  001
```