PROGRAM

```
package Assembler.AssemblerPass2;

import java.io.*;
import java.util.*;

public class PassII {
    public static void main(String[] args) throws IOException {
        //Creating file object to read data from 3 txt files
        BufferedReader br = new BufferedReader(new
FileReader("E:\\Coding\\TE_SEM_V\\SPOS\\TE_SPOS\\src\\Assembler.AssemblerPass2\\intermedi
ate.txt"));
        BufferedReader b2 = new BufferedReader(new
FileReader("E:\\Coding\\TE_SEM_V\\SPOS\\TE_SPOS\\src\\Assembler.AssemblerPass2\\symtab.tx
t"));
        BufferedReader b3 = new BufferedReader(new
FileReader("E:\\Coding\\TE_SEM_V\\SPOS\\TE_SPOS\\src\\Assembler.AssemblerPass2\\littab.tx
t"));

        //Creating file writer object
        FileWriter fw = new FileWriter("P2.txt");

        //Map the pointer and the symbol/literal addresses
        HashMap<Integer, String> symSymbol = new HashMap<Integer, String>();
        HashMap<Integer, String> litSymbol = new HashMap<Integer, String>();
        HashMap<Integer, String> litAddr = new HashMap<Integer, String>();
        int symtabPointer=1,littabPointer=1;
        String line,s;

        //Mapping symbol addresses with symbol ptr
        while((s=b2.readLine())!=null)
        {
            String word[]=s.split("\t\t\t");
            symSymbol.put(symtabPointer++,word[1]);
        }

        //Mapping literal addresses with literal ptr
        while((s=b3.readLine())!=null)
        {
            String word[]=s.split("\t\t");
            litSymbol.put(littabPointer,word[0]);
            litAddr.put(littabPointer++,word[1]);
        }

        //Reading intermediate code
        while((line=br.readLine())!=null){
            //Skipping processing of Assembler Directives
            if(line.contains("AD")) {
                fw.write("\n");
                continue;
            }
            //Processing imperative statements
            else if(line.contains("IS")){
                if(line.contains("IS,00"))
                    fw.write("+ 00  0  000\n");
                else
                    fw.write("+ " + line.substring(4,6) + "  " + line.charAt(8) + "  ");
                if(line.contains("(S,")){

fw.write(symSymbol.get(Integer.parseInt(line.substring(13,14)))+"\n");
                }
                else if(line.contains("(L,")){
```

```
                    fw.write(litAddr.get(Integer.parseInt(line.substring(13,14)))+"\n");
                }
            }
            //Process DS statement
            else if(line.contains("DL,01")){
                fw.write("+ 00  0  00" + line.charAt(10) + "\n");
            }
            else if(line.contains("DL,02")){
                fw.write("\n");
            }
        }
        fw.close();
        br.close();
        b2.close();
        b3.close();
    }
}
```

INPUT

intermediate.txt

```
1   (AD,01)(C,200)
2   (IS,04)(1)(L,1)
3   (IS,05)(1)(S,1)
4   (IS,04)(1)(S,1)
5   (IS,04)(3)(S,3)
6   (IS,01)(3)(L,2)
7   (IS,07)(6)(S,4)
8   (DL,01)(C,5)
9   (DL,01)(C,1)
10  (IS,02)(1)(L,3)
11  (IS,07)(1)(S,5)
12  (IS,00)
13  (AD,03)(S,2)+2
14  (IS,03)(3)(S,3)
15  (AD,03)(S,6)+1
16  (DL,02)(C,1)
17  (DL,02)(C,1)
18  (AD,02)
19  (DL,01)(C,1)
```

Input 1 – Intermediate Code

littab.txt

```
1   5   206
2   1   207
3   1   213
```

Input 2 – Literal Table

symtab.txt

```
1   A       211     1
2   LOOP        202     1
3   B       212     1
4   NEXT        208     1
5   BACK        202     1
6   LAST        210     1
```

Input 3 – Symbol Table

OUPUT

```
1
2   + 04  1  206
3   + 05  1  211
4   + 04  1  211
5   + 04  3  212
6   + 01  3  207
7   + 07  6  208
8   + 00  0  005
9   + 00  0  001
10  + 02  1  213
11  + 07  1  202
12  + 00  0  000
13
14  + 03  3  212
15
16
17
18
19  + 00  0  001
20
```

Ouput – Machine Code