AssemblerP1.java

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

public class Ass_1 {
    static int address = 0;
    static int sadd[] = new int[10];
    static int ladd[] = new int[10];

    public static void main(String args[]) {
        BufferedReader br;
        OutputStream oo;
        String input = null;
        String IS[] = { "ADD", "SUB", "MUL", "MOVR", "MOVM" };
        String UserReg[] = { "AREG", "BREG", "CREG", "DREG" };
        String AD[] = { "START", "END", "$$", "$$", "LTORG" };
        String DL[] = { "DC", "DS" };
        int lc = 0;
        int scount = 0, lcount = 0, pcount = 0, cc = 0, ca = 0;
        int flag = 0, flag2 = 0, stored = 0, i, kk = 0, j;
        String tokens[] = new String[30];
        String tt = null;
        String sv[] = new String[10];
        String lv[] = new String[10];
        try {
            br = new BufferedReader(new FileReader("initial.txt"));
            File f = new File("IM.txt");
            File f1 = new File("ST.txt");
            File f2 = new File("LT.txt");
            PrintWriter p = new PrintWriter(f);
            PrintWriter p1 = new PrintWriter(f1);
            PrintWriter p2 = new PrintWriter(f2);
            int k = 0, l = 0, o = 0;
            while ((input = br.readLine()) != null) {
                StringTokenizer st = new StringTokenizer(input, " ");
                while (st.hasMoreTokens()) {
                    tt = st.nextToken();

                    if (tt.matches("\\d*") && tt.length() > 2) {
                        lc = Integer.parseInt(tt);
                        p.println(lc);
                        address = lc - 1;
                    } else {
                        for (i = 0; i < AD.length; i++) {
                            if (tt.equals(AD[i])) {
                                p.print("AD " + (i + 1) + " ");
                                if (tt.equals(AD[4])) {
                                    if (ca == 1) {
                                        --address;
                                        for (j = 0; j < cc; j++) {
                                            p2.println(j + "\t" + lv[j] + "\t" + address);
                                            address++;
                                        }
                                        kk = j;
                                        pcount = cc;
                                    }
                                    if (ca == 2) {
                                        for (j = kk; j < cc; j++) {
                                            p2.println(j + "\t" + lv[j] + "\t" + address);
                                            address++;
                                        }
                                        kk = j;
                                        pcount = cc;
                                    }
                                    o++;
                                }
                            }
                        }
                        for (i = 0; i < IS.length; i++) {
                            if (tt.equals(IS[i])) {
                                p.print("IS " + (i + 1) + " ");
                            }
                        }
                        for (i = 0; i < UserReg.length; i++) {
                            if (tt.equals(UserReg[i])) {
```

```
                        p.print((i + 1) + " ");
                        flag = 1;
                    }
                }
                for (i = 0; i < DL.length; i++) {
                    if (tt.equals(DL[i])) {
                        p.print("DL " + (i + 1) + " ");
                    }
                }
                if (tt.length() == 1 && !(st.hasMoreTokens()) && flag == 1) {
                    if (Arrays.asList(sv).contains(tt)) {
                        for (i = 0; i < scount; i++) {
                            if (sv[i].equals(tt)) {
                                p.print("S" + i);
                                flag2 = 1;
                            } else {
                                flag2 = 0;
                            }
                        }
                    } else {
                        p.print("S" + scount);
                        sv[scount] = tt;
                        flag2 = 1;
                        scount++;
                    }
                }
                if (tt.length() == 1 && (st.hasMoreTokens())) {
                    p.print(tt + " ");
                    sadd[k] = address;
                    k++;
                }
                if (tt.charAt(0) == '=') {
                    p.print("L" + lcount);
                    lv[lcount] = tt;
                    lcount++;
                    cc++;
                    if (ca == 0)
```

```
                        ca = 1;
                    else
                        ca = 2;
                }
                if (!st.hasMoreTokens()) {
                    p.println();
                }
                if (tt.equals("DS")) {
                    int a = Integer.parseInt(st.nextToken());
                    address = address + a - 1;
                    p.println();
                }
            }
        }
        address++;
        address = address - o;
        o = 0;
    }
    p.close();
    if (pcount == 0) {
        if (ca == 1) {
            address--;
            for (i = 0; i < cc; i++) {
                p2.println(i + "\t" + lv[i] + "\t" + address);
                address++;
            }
            kk = i;
        }
        if (ca == 2) {
            for (i = kk; i < cc; i++) {
                p2.println(i + "\t" + lv[i] + "\t" + address);
                address++;
            }
        }
    }
    for (i = 0; i < scount; i++) {
        p1.println(i + "\t" + sv[i] + "\t" + sadd[i]);
```

```
            }
            p1.close();
            p2.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Assembly Code:

START 100

MOVM AREG A

MOVM BREG B

MOVR CREG =2

MOVR DREG =3

LTORG

ADD AREG BREG

MOVR CREG =5

MOVR CREG =5

SUB AREG A

LTORG

A DC 05

B DS 03

END

Symbol Table:

0	A	112
1	B	113

Literals Table:

0	=2	104
1	=3	105
2	=5	110

3 =5 111

Output Intermediate code:

```
AD   1   100
IS   5   1   S0
IS   5   2   S1
IS   4   3   L0
IS   4   4   L1
AD   5
IS   1   1   2
IS   4   3   L2
IS   4   3   L3
IS   2   1   S0
AD   5
A    DL  1
B    DL  2
AD   2
```