

Practical No 02

```
package A2;

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

// Tuple class
class Tuple {
    String mnemonic, m_class, opcode;
    int length;

    Tuple() {}

    Tuple(String s1, String s2, String s3, String s4) {
        mnemonic = s1;
        m_class = s2;
        opcode = s3;
        length = Integer.parseInt(s4);
    }
}

// Symbol Table Tuple
class SymTuple {
    String symbol, address, length;

    SymTuple(String s1, String s2, String s3) {
        symbol = s1;
        address = s2;
        length = s3;
    }
}

// Literal Table Tuple
class LitTuple {
    String literal, address, length;

    LitTuple() {}

    LitTuple(String s1, String s2, String s3) {
        literal = s1;
        address = s2;
        length = s3;
    }
}

public class Assembler_PassTwo {

    static int lc, iSymTabPtr = 0, iLitTabPtr = 0, iPoolTabPtr = 0;
    static int poolTable[] = new int[10];
    static Map<String, Tuple> MOT;
    static ArrayList<SymTuple> symtable;
```

```

static ArrayList<LitTuple> littable;
static Map<String, String> regAddressTable;
static PrintWriter out_pass2;

static void initializeTables() throws Exception {
    symtable = new ArrayList<>();
    littable = new ArrayList<>();
    regAddressTable = new HashMap<>();
    MOT = new HashMap<>();

    String s;
    BufferedReader br;

    // Reading Symbol Table
    br = new BufferedReader(new InputStreamReader(new
FileInputStream("A2/symtable.txt")));
    while ((s = br.readLine()) != null) {
        StringTokenizer st = new StringTokenizer(s, "\t", false);
        symtable.add(new SymTuple(st.nextToken(), st.nextToken(), ""));
    }
    br.close();

    // Reading Literal Table
    br = new BufferedReader(new InputStreamReader(new
FileInputStream("A2/littable.txt")));
    while ((s = br.readLine()) != null) {
        StringTokenizer st = new StringTokenizer(s, "\t", false);
        littable.add(new LitTuple(st.nextToken(), st.nextToken(), ""));
    }
    br.close();

    // Initialize register address table
    regAddressTable.put("AREG", "1");
    regAddressTable.put("BREG", "2");
    regAddressTable.put("CREG", "3");
    regAddressTable.put("DREG", "4");
}

static void pass2() throws Exception {
    BufferedReader input = new BufferedReader(new InputStreamReader(new
FileInputStream("A2/output_pass1.txt")));
    out_pass2 = new PrintWriter(new FileWriter("A2/output_pass2.txt"), true);

    String s;

    // Process each line from intermediate code
    while ((s = input.readLine()) != null) {
        String output_str = "";
        String[] ic_tokens = s.split("\\s+");

        if (ic_tokens.length < 2) continue;

```

```

// Example processing for a DC directive
if (ic_tokens[1].equalsIgnoreCase("(DL,02)")) {
    // DL 02 -> Define Constant
    String[] opr_tokens = tokenizeString(ic_tokens[2], ",");
    output_str = ic_tokens[0] + "\t" + "00\t0\t" + opr_tokens[1];
}

// You can add more processing logic here as per requirement

System.out.println(output_str);
out_pass2.println(output_str);
}
input.close();
out_pass2.close();
}

static String[] tokenizeString(String str, String separator) {
    StringTokenizer st = new StringTokenizer(str, separator, false);
    String[] s_arr = new String[st.countTokens()];
    for (int i = 0; i < s_arr.length; i++) {
        s_arr[i] = st.nextToken();
    }
    return s_arr;
}

public static void main(String[] args) throws Exception {
    initializeTables();
    pass2();
}
}

//output_pass1.txt
100 (IS,04) (R,1) (S,1)
101 (IS,05) (R,2) (S,2)
102 (DL,02) (C,5)
103 (DL,02) (C,3)
104 (IS,02) (R,3) (L,1)
105 (IS,01) (R,1) (L,2)
106 (AD,02)
107 (IS,06) (R,2) (S,3)

//littable.txt
='5'    201
='3'    202

//symtable.txt
LOOP 101
VALUE    102
TEMP 104

```

```
PS D:\Suyash Birar\LP1 Pr> java -cp . A2.Assembler_PassTwo  
>>
```

```
102    00    0    5)  
103    00    0    3)
```

```
PS D:\Suyash Birar\LP1 Pr> █
```

≡ *output_pass2.txt* X

A2 > ≡ output_pass2.txt

1

2

3 102 00 0 5)

4 103 00 0 3)

5

6

7