Practical No.1

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

class Symbol {
    String symbol;
    int address;

    Symbol(String symbol, int address) {
        this.symbol = symbol;
        this.address = address;
    }
}

class IntermediateLine {
    int address;
    String label;
    String opcode;
    String operand;

    IntermediateLine(int address, String label, String opcode, String operand) {
        this.address = address;
        this.label = label;
        this.opcode = opcode;
        this.operand = operand;
    }

    @Override
    public String toString() {
        return address + "\t" + (label == null ? "-" : label) + "\t" + opcode + "\t" + (operand
== null ? "-" : operand);
    }
}

public class TwoPassAssembler {
    private static Map<String, String> opcodeTable = new HashMap<>();
    private static Map<String, Integer> symbolTable = new HashMap<>();
    private static List<IntermediateLine> intermediateCode = new ArrayList<>();

    private static int locCounter = 0;
```

```
static {
    opcodeTable.put("MOV", "01");
    opcodeTable.put("ADD", "02");
    opcodeTable.put("SUB", "03");
    opcodeTable.put("JMP", "04");
    opcodeTable.put("DC", "DC");
    opcodeTable.put("DS", "DS");
    opcodeTable.put("START", "START");
    opcodeTable.put("END", "END");
}

public static void passOne(List<String> source) {
    locCounter = 0;

    for (String line : source) {
        line = line.trim();
        if (line.isEmpty() || line.startsWith(";")) continue;

        String label = null, opcode = null, operand = null;
        String[] parts = line.split("\\s+");
        if (parts.length == 1) {
            opcode = parts[0];
        } else if (parts.length == 2) {
            opcode = parts[0];
            operand = parts[1];
        } else if (parts.length >= 3) {
            label = parts[0];
            opcode = parts[1];
            operand = parts[2];
        }

        if ("START".equalsIgnoreCase(opcode)) {
            if (operand != null) {
                locCounter = Integer.parseInt(operand);
            }
            intermediateCode.add(new IntermediateLine(locCounter, label, opcode, operand));
            continue;
        }

        if (label != null) {
            if (symbolTable.containsKey(label)) {
```

```
                System.err.println("Error: Duplicate symbol " + label);
            } else {
                symbolTable.put(label, locCounter);
            }
        }

        intermediateCode.add(new IntermediateLine(locCounter, label, opcode, operand));

        if (opcode.equalsIgnoreCase("DC")) {
            locCounter += 1;
        } else if (opcode.equalsIgnoreCase("DS")) {
            int size = operand == null ? 1 : Integer.parseInt(operand);
            locCounter += size;
        } else if (opcode.equalsIgnoreCase("END")) {
            break;
        } else {
            locCounter += 1;
        }
    }
}

public static void passTwo() {
    System.out.println("Address\tMachine Code");
    for (IntermediateLine line : intermediateCode) {
        if (line.opcode.equalsIgnoreCase("START") || line.opcode.equalsIgnoreCase("END"))
{
            continue;
        }

        String mc = "";

        if (opcodeTable.containsKey(line.opcode)) {
            String opCodeHex = opcodeTable.get(line.opcode);

            if (opCodeHex.equals("DC")) {
                mc = String.format("%04X", Integer.parseInt(line.operand));
            } else if (opCodeHex.equals("DS")) {
                int size = line.operand == null ? 1 : Integer.parseInt(line.operand);
                for (int i = 0; i < size; i++) {
                    System.out.printf("%04X\t%04X\n", line.address + i, 0);
                }
                continue;
```

```
                } else {
                    int operandAddress = 0;
                    if (line.operand != null) {
                        if (symbolTable.containsKey(line.operand)) {
                            operandAddress = symbolTable.get(line.operand);
                        } else {
                            System.err.println("Error: Undefined symbol " + line.operand + " at
address " + line.address);
                            operandAddress = 0;
                        }
                    }
                    mc = opCodeHex + String.format("%03X", operandAddress);
                }
            } else {
                System.err.println("Error: Invalid opcode " + line.opcode + " at address " +
line.address);
                mc = "????";
            }

            System.out.printf("%04X\t%s\n", line.address, mc);
        }
    }

    public static void main(String[] args) {
        List<String> sourceProgram = Arrays.asList(
                "START 100",
                "LOOP MOV A",
                "ADD B",
                "SUB C",
                "JMP LOOP",
                "A DC 5",
                "B DC 10",
                "C DC 15",
                "END"
        );

        passOne(sourceProgram);

        System.out.println("\nIntermediate Code:");
        for (IntermediateLine line : intermediateCode) {
            System.out.println(line);
        }
```

```
        System.out.println("\nSymbol Table:");
        for (Map.Entry<String, Integer> entry : symbolTable.entrySet()) {
            System.out.printf("%s -> %04X\n", entry.getKey(), entry.getValue());
        }

        System.out.println("\nPass II: Generating Machine Code");
        passTwo();
    }
}
```

Output:-