Assignment No. 1

Name : Aryan Shinde
Div : B
Batch : C
Roll no : 58
Sub : LP-1

Pass 1 Program

```
 import java.io.*; import java.util.*; class Symbol {     String
name;
   int address;

   Symbol(String name, int address) {        this.name
= name;
       this.address = address;
   }
} class Literal {
String value;
int address;

   Literal(String value, int address) {
       this.value = value;
       this.address = address;
   }
}

class Opcode {
String mnemonic;
String classType;      int
code;
   int length;

   Opcode(String mnemonic, String classType, int code, int length) {
this.mnemonic = mnemonic;        this.classType = classType;        this.code
= code;
       this.length = length;
   }
}
// ---------------- Pass-1 ----------------- public class Pass1 {     public
static void main(String[] args) throws Exception {
Map<String, Opcode> OPTAB = new HashMap<>();
       OPTAB.put("START", new Opcode("START", "AD", 1, 0));
       OPTAB.put("END", new Opcode("END", "AD", 2, 0));
       OPTAB.put("LTORG", new Opcode("LTORG", "AD", 5, 0)); OPTAB.put("DS", new
Opcode("DS", "DL", 1, 0));
        OPTAB.put("DC", new Opcode("DC", "DL", 2, 0));
       OPTAB.put("EQU", new Opcode("EQU", "AD", 3, 0));
       OPTAB.put("MOVER", new Opcode("MOVER", "IS", 4, 1));
       OPTAB.put("MOVEM", new Opcode("MOVEM", "IS", 5, 1));
       OPTAB.put("ADD", new Opcode("ADD", "IS", 1, 1));
       OPTAB.put("SUB", new Opcode("SUB", "IS", 2, 1));

       Map<String, Integer> REGMAP = new HashMap<>();
       REGMAP.put("AREG", 1);
REGMAP.put("BREG", 2);
       REGMAP.put("CREG", 3);
       REGMAP.put("DREG", 4);

       List<Symbol> SYMTAB = new ArrayList<>();
       List<Literal> LITTAB = new ArrayList<>();
       List<Integer> POOLTAB = new ArrayList<>();
       List<String> IC = new ArrayList<>();

       POOLTAB.add(1);         int LC = 0, littabPtr
= 1, pooltabPtr = 0;         List<String[]> source
= new ArrayList<>();
       BufferedReader br = new BufferedReader(new FileReader("Input.txt"));
       String line;

       while ((line = br.readLine()) != null) {
line = line.trim();
            if (line.isEmpty()) continue;            String[]
parts = line.split("\\s+", 3);
source.add(parts);
            if (parts[0].equals("END") || (parts.length > 1 && parts[1].equals("END"))) {
break;
            }
       }
       br.close();

       // --- PASS 1 Logic ---         for
(String[] parts : source) {
            String label = "", opcode = "", operands = "";

            if (!OPTAB.containsKey(parts[0])) {
                label = parts[0];                opcode =
(parts.length > 1) ? parts[1] : "";                operands
= (parts.length > 2) ? parts[2] : "";
            } else {            opcode = parts[0];
  operands = (parts.length > 1) ? parts[1] : "";
            }
            if (!label.isEmpty() && !OPTAB.containsKey(label)) {
                boolean exists = false;     for
            (Symbol s : SYMTAB) {     if
            (s.name.equals(label)) {
            s.address = LC;
                    exists = true;
                    break;
                 }
              }
              if (!exists) SYMTAB.add(new Symbol(label, LC));
            }

            if (opcode.isEmpty()) continue;

            if (OPTAB.containsKey(opcode)) {
Opcode op = OPTAB.get(opcode);

               if (op.classType.equals("AD")) {
if (opcode.equals("START")) {                       LC
= Integer.parseInt(operands);
                    IC.add("(AD,01) (C," + operands + ")");
                 } else if (opcode.equals("END") || opcode.equals("LTORG"))
{                       for (int i = POOLTAB.get(pooltabPtr) - 1; i < LITTAB.size();
i++) {                          if (LITTAB.get(i).address == -1) {
                          LITTAB.get(i).address = LC;
                          IC.add("(DL,02) (C," + LITTAB.get(i).value.substring(1) + ")");
                          LC++;
                       }
                    }
                    pooltabPtr++;
                    POOLTAB.add(littabPtr);
                    if (opcode.equals("END")) IC.add("(AD,02)");
                 } else if (opcode.equals("EQU")) {                       int
addr = Integer.parseInt(operands);
                    if (!SYMTAB.isEmpty()) SYMTAB.get(SYMTAB.size() - 1).address = addr;
IC.add("(AD,03) (C," + operands + ")");
                 }
              } else if (op.classType.equals("DL")) {                  if
(opcode.equals("DS")) {
                    IC.add("(DL,01) (C," + operands + ")");
                    LC += Integer.parseInt(operands);
                 } else if (opcode.equals("DC")) {
                    IC.add("(DL,02) (C," + operands + ")");
                    LC++;
                 }
              } else if (op.classType.equals("IS")) {
String icEntry = "(IS," + op.code + ") ";                  if
(!operands.isEmpty()) {
                    String[] ops = operands.split(",");                  for
   (String opnd : ops) {
                      opnd = opnd.trim();               if
```

```
        (REGMAP.containsKey(opnd)) {
                        icEntry += "(RG," + REGMAP.get(opnd) + ") ";
                    } else if (opnd.startsWith("=")) {
            int litIndex = -1;               boolean exists =
            false;
                        for (int i = 0; i < LITTAB.size(); i++) {
                            if (LITTAB.get(i).value.equals(opnd)) {
        exists = true;
        litIndex = i + 1;
        break;
                            }
}
                        if (!exists) {
                            LITTAB.add(new Literal(opnd, -1));
                            litIndex = LITTAB.size();
                            littabPtr++;
                        }
                        icEntry += "(L," + litIndex + ") ";
                    } else {
                        int symIndex = -1;
boolean exists = false;                            for (int i = 0; i <
SYMTAB.size(); i++) {                               if
(SYMTAB.get(i).name.equals(opnd)) {
                                symIndex = i + 1;
exists = true;                              break;
                            }
}                           if
(!exists) {
                            SYMTAB.add(new Symbol(opnd, -1));
                            symIndex = SYMTAB.size();
                        }
                        icEntry += "(S," + symIndex + ") ";
                    }
                }
            }
            IC.add(icEntry.trim());
            LC += op.length;
        }
    }
}
try (PrintWriter icFile = new PrintWriter("IC.txt");             PrintWriter
symFile = new PrintWriter("SYMTAB.txt");
     PrintWriter litFile = new PrintWriter("LITTAB.txt");
     PrintWriter poolFile = new PrintWriter("POOLTAB.txt")) {

    for (String i : IC) icFile.println(i);

     for (int i = 0; i < SYMTAB.size(); i++)
```

```
            symFile.println((i + 1) + " " + SYMTAB.get(i).name + " " + SYMTAB.get(i).address);
          if (LITTAB.size() == 0) {
          litFile.println("null");
          } else {  for (int i = 0; i <
            LITTAB.size(); i++)
              litFile.println((i + 1) + " " + LITTAB.get(i).value + " " + LITTAB.get(i).address); }

          if (POOLTAB.size() == 0) {
        poolFile.println("null");
          } else {
            for (int i = 0; i < POOLTAB.size(); i++)
              poolFile.println((i + 1) + " " + POOLTAB.get(i));
          }
        }
        System.out.println("\nPASS-1 complete. Tables + IC written to files.");

        System.out.println("\n--- INTERMEDIATE CODE ---");        for
(String i : IC) System.out.println(i);

        System.out.println("\n--- SYMTAB ---");        for
(int i = 0; i < SYMTAB.size(); i++)
          System.out.println((i + 1) + " " + SYMTAB.get(i).name + " " + SYMTAB.get(i).address);

        System.out.println("\n--- LITTAB ---");
        if (LITTAB.size() == 0) {            System.out.println("null");
        } else {
          for (int i = 0; i < LITTAB.size(); i++)
            System.out.println((i + 1) + " " + LITTAB.get(i).value + " " + LITTAB.get(i).address);
        }

        System.out.println("\n--- POOLTAB ---");
        if (POOLTAB.size() == 0) {            System.out.println("null");
        } else {
          for (int i = 0; i < POOLTAB.size(); i++)
            System.out.println((i + 1) + " " + POOLTAB.get(i));
        }
    }
}
```

INPUT.txt :

```
START 100
MOVER AREG, ='5'
MOVEM AREG, B
ADD BREG,C
SUB CREG,D
D DS 1
B       DC 10
```

```
C          DC 20  END
```

OUTPUT :

```
C:\Users\durve\Desktop\Assembler>javac Pass1.java

C:\Users\durve\Desktop\Assembler>java Pass1

PASS-1 complete. Tables + IC written to files.

--- INTERMEDIATE CODE --- (AD,01)
(C,100)
(IS,4) (RG,1)
(IS,5) (RG,1)
(IS,1) (RG,2) (S,1)
(IS,2) (RG,3) (S,2)
(DL,01) (C,1)
(DL,02) (C,10)
(DL,02) (C,20)
(AD,02)

--- SYMTAB ---
1 C 106
2 D 104
3 B 105

--- LITTAB ---
1 ='5' 107
--- POOLTAB --- 1
1
```

Pass 2
Program :

```
import java.io.File; import
java.io.PrintWriter; import
java.util.HashMap;
import java.util.Scanner;

public class Pass2 {
  public Pass2() {
  }

  public static void main(String[] var0) throws Exception {
HashMap var1 = new HashMap();
    Scanner var2 = new Scanner(new File("SYMTAB.txt"));
    int var6;
    try {
        while(var2.hasNextLine()) {       String
      var3 = var2.nextLine().trim();            if
      (!var3.isEmpty()) {
            String[] var4 = var3.split("\\s+");               if
(var4.length >= 3) {                 int var5 =
Integer.parseInt(var4[0]);                var6 =
Integer.parseInt(var4[2]);
var1.put(var5, var6);
            }
          }
        }
      } catch (Throwable var19) {
        try {
          var2.close();
        } catch (Throwable var12) {
          var19.addSuppressed(var12);
        }

        throw var19;
      }

      var2.close();
      HashMap var20 = new HashMap();
      Scanner var21 = new Scanner(new File("LITTAB.txt"));
       try
{
        while(var21.hasNextLine()) {
String var22 = var21.nextLine().trim();            if
(!var22.isEmpty()) {
             String[] var24 = var22.split("\\s+");               if
(var24.length >= 3) {                  var6 =
Integer.parseInt(var24[0]);                 int var7 =
Integer.parseInt(var24[2]);
var20.put(var6, var7);
             }
          }
        }
      } catch (Throwable var18) {
        try {            var21.close();
} catch (Throwable var13) {
          var18.addSuppressed(var13);
       }
       throw var18;
     }
```

```
      var21.close();
      var21 = new Scanner(new File("IC.txt"));
try {
       PrintWriter var23 = new PrintWriter("MachineCode.txt");

       try {
          while(var21.hasNextLine()) {
            String var25 = var21.nextLine().trim();              if
(!var25.isEmpty() && !var25.startsWith("(AD")) {
               String[] var26;                  String var27;
if
(var25.startsWith("(DL")) {
var26 = var25.split("\\s+");                    if
(var26[0].equals("(DL,02)")) {
                   var27 = var26[1].replace("(C,", "").replace(")", "");
var23.println("00 00 " + var27);
                 } else {
                   var23.println("00 00 00");
                 }
               } else if (var25.startsWith("(IS")) {                    var26
= var25.split("\\s+");
                 var27 = var26[0].replaceAll("[^0-9]", "");
String var8 = "0";                    if (var26.length > 1 &&
var26[1].startsWith("(RG")) {                        var8 =
var26[1].replaceAll("[^0-9]", "");
                 }

                 String var9 = "000";
if (var26.length > 2) {
String var10 = var26[2];                      int
var11;
                   if (var10.startsWith("(S")) {                          var11 =
Integer.parseInt(var10.replaceAll("[^0-9]", ""));
var9 = String.valueOf(var1.get(var11));                      } else if
(var10.startsWith("(L")) {
                     var11 = Integer.parseInt(var10.replaceAll("[^0-9]", ""));
var9 = String.valueOf(var20.get(var11));                      } else if
(var10.startsWith("(C")) {                          var9 =
var10.replaceAll("[^0-9]", "");
                   }
                 }
                var23.println(var27 + " " + var8 + " " + var9);
              }
            }
          }
        } catch (Throwable var16) {
          try {
            var23.close();
          } catch (Throwable var15) {
            var16.addSuppressed(var15);
          }

          throw var16;
        }

        var23.close();
      } catch (Throwable var17) {
        try {
          var21.close();
        } catch (Throwable var14) {
          var17.addSuppressed(var14);
        }

        throw var17;
      }

      var21.close();
      System.out.println("PASS-2 complete Machine code written to MachineCode.txt");
    }
  }
```

Intermediate Code :

```
(AD,01) (C,100)
(IS,4) (RG,1)
(IS,5) (RG,1)
(IS,1) (RG,2) (S,1)
(IS,2) (RG,3) (S,2)
(DL,01) (C,1)
(DL,02) (C,10)
(DL,02) (C,20)
(AD,02)
```

Symbol Table :

```
1 C 106
2 D 104
3 B 105
```

Literal Table :

```
1 ='5' 107
```

OUTPUT :

```
C:\Users\durve\Desktop\Assembler>javac Pass2.java

C:\Users\durve\Desktop\Assembler>java Pass2
PASS-2 complete Machine code written to MachineCode.txt
```

MachineCode.txt :

```
4 1 107
5 1 104
1 2 106
2 3 105
00 00 00
00 00 10
00 00 20
```