EXPERIMENT – 4

Aim: IMPLEMENTATION OF TWO PASS ASSEMBLER

IMPLEMENTATION:
- CODE (C / Java / any language you are comfortable)
App.java

```
import java.util.List;

public class App{
        public static void main(String[] args) {
                TwoPassMacroprocessor twoPassMacroprocessor = new TwoPassMacroprocessor();
    String input_string = "START\n" +
            "\n" +
            "MACRO\n" +
            "ADD &ARG1,&ARG2\n" +
            "L 1,&ARG1\n" +
            "A 1,&ARG2\n" +
            "MEND\n" +
            "\n" +
            "MACRO\n" +
            "SUB &ARG3,&ARG4\n" +
            "L 1,&ARG3\n" +
            "S 1,&ARG4\n" +
            "MEND\n" +
            "\n" +
            "ADD DATA1,DATA2\n" +
            "SUB DATA1,DATA2\n" +
            "\n" +
            "DATA1 DC F'9'\n" +
            "DATA2 DC F'5'\n" +
            "\n" +
            "END";

    List<List<String []>> input = twoPassMacroprocessor.parseInput(input_string);
    twoPassMacroprocessor.generatePasses(input);
    twoPassMacroprocessor.populateALA();
    twoPassMacroprocessor.performPass1();
    twoPassMacroprocessor.performPass2();
  }
}
```

TwoPassMacroprocessor.java

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

class MDTP{
  private String macroName;
```

```
    private int MDT_index;

    MDTP(String macroName, int MDT_index){
        this.macroName = macroName;
        this.MDT_index = MDT_index;
    }

    public String getMacroName() {
        return macroName;
    }

    public void setMacroName(String macroName) {
        this.macroName = macroName;
    }

    public int getMDT_index() {
        return MDT_index;
    }

    public void setMDT_index(int MDT_index) {
        this.MDT_index = MDT_index;
    }

    @Override
    public String toString() {
        return  macroName + "      #" + MDT_index ;
    }
}

class MDT{
    private List<String> body;

    MDT(List<String> body){
        this.body = body;
    }

    public List<String> getBody() {
        return body;
    }

    public void setBody(List<String> body) {
        this.body = body;
    }

    @Override
    public String toString() {
        return Arrays.toString(body.toArray()) ;
    }
}

public class TwoPassMacroprocessor {
```

```
static List<List<String []>> passes = new ArrayList<>();
static List<List<String []>> non_passes = new ArrayList<>();
static List<String> ALA = new ArrayList<>();
static List<MDTP> mdtp = new ArrayList<>();
static List<MDT> mdt = new ArrayList<>();
static List<String> MACROS = new ArrayList<>(Arrays.asList("ADD", "SUB"));

public void populateALA(){
   for(List<String []> pass : passes){
      for(String [] p : pass){
         if(MACROS.contains(p[0])){
            for(String arg : Arrays.copyOfRange(p, 1, p.length)){
               ALA.add(arg);
            }
         }
      }
   }
}

public void generatePasses(List<List<String []>> input){

   for(List<String []> ip : input){
      if(ip.get(0)[0].equals("MACRO")){
         ip.remove(0);
         passes.add(ip);
      }
      else{
         non_passes.add(ip);
      }
   }
}

public List<List<String []>> parseInput(String input){
   String [] lines = input.split("\n");
   String [][] full_split = new String[lines.length][];

   for(int i=0; i < lines.length; i++){
      full_split[i] = lines[i].split("\\s+|,");
   }

   List<Integer> blank_indexes = new ArrayList<>();

   for(int i=0; i<full_split.length; i++){
      if(full_split[i][0].equals("")){
         blank_indexes.add(i);
      }
   }

   List<int []> paired_blank_indexes = new ArrayList<>();

   for(int i=0; i<blank_indexes.size()-1; i++){
```

```
        paired_blank_indexes.add(new int []{blank_indexes.get(i),blank_indexes.get(i+1)});
    }

    List<List<String []>> inputs = new ArrayList<>();

    for(int [] pairs : paired_blank_indexes){
        List<String []> temp = new ArrayList<>();
        for(int i=pairs[0]+1; i<pairs[1]; i++){
            temp.add(full_split[i]);
        }
        inputs.add(temp);
    }

    return inputs;
}

static void printTables(){
    System.out.println("MDTP: ");
    System.out.println("MACRO    MDT_INDEX");
    for(MDTP m : mdtp){
        System.out.println(m.toString());
    }

    System.out.println("\nMDT: ");
    System.out.println("INDEX    BODY");
    for(int i=0; i<mdt.size(); i++){
        System.out.println("#"+i + "       " + mdt.get(i).toString());
    }

    System.out.println("\nALA: ");
    System.out.println("INDEX    ARGUMENT");
    for(int i=0; i<ALA.size(); i++){
        System.out.println("#"+i + "        " + ALA.get(i));
    }
}

public void performPass1(){
    //populate MDT and MDTP simultaneously
    for(List<String []> pass : passes){
        for(String [] p : pass){
            // if the first element is ADD or SUB
            if(MACROS.contains(p[0])){
                MDTP temp_mdtp = new MDTP(p[0], 0);
                MDT temp_mdt = new MDT(Arrays.asList(p));
                mdt.add(temp_mdt);
                temp_mdtp.setMDT_index(mdt.indexOf(temp_mdt));
                mdtp.add(temp_mdtp);
            }
            else{
                mdt.add(new MDT(Arrays.asList(p)));
            }
```

```
            }
        }

        //replacing arguments by respective indexes
        for(MDT mdtp : mdt){
            List<String> body = mdtp.getBody();
            for(int i=1; i<body.size(); i++){
                if(ALA.contains(body.get(i))) {
                    body.set(i, "#" + ALA.indexOf(body.get(i)));
                }
            }
        }
        System.out.println("\n-------- PASS 1 --------\n");
        printTables();
    }

    public void performPass2(){
        for(List<String []> non_pass : non_passes){
            for(String [] np : non_pass){
                if(MACROS.contains(np[0])){
                    // Go inside MDTP Table and if first element is a valid macro name then get MDT index
                    // and then get argument index
                    for(MDTP mdtp : mdtp){
                        if(mdtp.getMacroName().equals(np[0])){
                            List<String> body = mdt.get(mdtp.getMDT_index()).getBody();
                            // change value inside ALA
                            ALA.set(Integer.valueOf(body.get(1).replace("#", "")), np[1]);
                            ALA.set(Integer.valueOf(body.get(2).replace("#", "")), np[2]);
                        }
                    }
                }
            }
        }

        // After changing value inside ALA then change the same in MDT Table.
        for(MDT mdt : mdt){
            List<String> body = mdt.getBody();
            for(int i=1; i<body.size(); i++){
                if(body.get(i).startsWith("#")) {
                    body.set(i, String.valueOf(ALA.get(Integer.valueOf(body.get(i).replace("#", "")))));
                }
            }
        }

        System.out.println("\n-------- PASS 2 --------\n");
        printTables();
    }
}
```

OUTPUT:

- Snapshots of the tables in pass1 and pass 2

```
-------- PASS 1 --------

MDTP:
MACRO    MDT_INDEX
ADD      #0
SUB      #4

MDT:
INDEX    BODY
#0       [ADD, #0, #1]
#1       [L, 1, #0]
#2       [A, 1, #1]
#3       [MEND]
#4       [SUB, #2, #3]
#5       [L, 1, #2]
#6       [S, 1, #3]
#7       [MEND]

ALA:
INDEX    ARGUMENT
#0        &ARG1
#1        &ARG2
#2        &ARG3
#3        &ARG4
```

```
-------- PASS 2 --------

MDTP:
MACRO    MDT_INDEX
ADD      #0
SUB      #4

MDT:
INDEX    BODY
#0       [ADD, DATA1, DATA2]
#1       [L, 1, DATA1]
#2       [A, 1, DATA2]
#3       [MEND]
#4       [SUB, DATA1, DATA2]
#5       [L, 1, DATA1]
#6       [S, 1, DATA2]
#7       [MEND]

ALA:
INDEX    ARGUMENT
#0        DATA1
#1        DATA2
#2        DATA1
#3        DATA2
```

CONCLUSION:

A two pass assembler for an 8086 microprocessor is implemented.