**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");

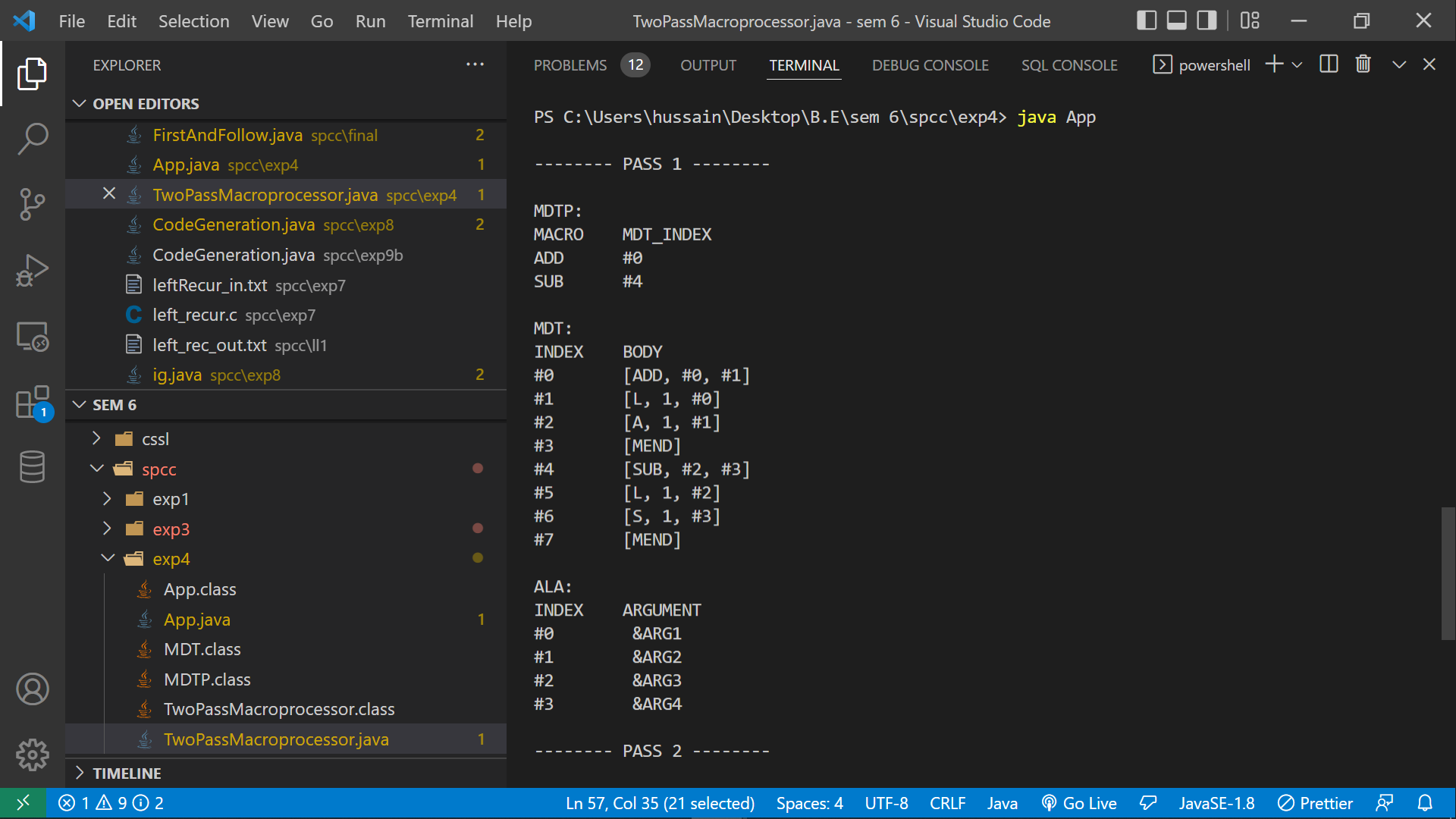
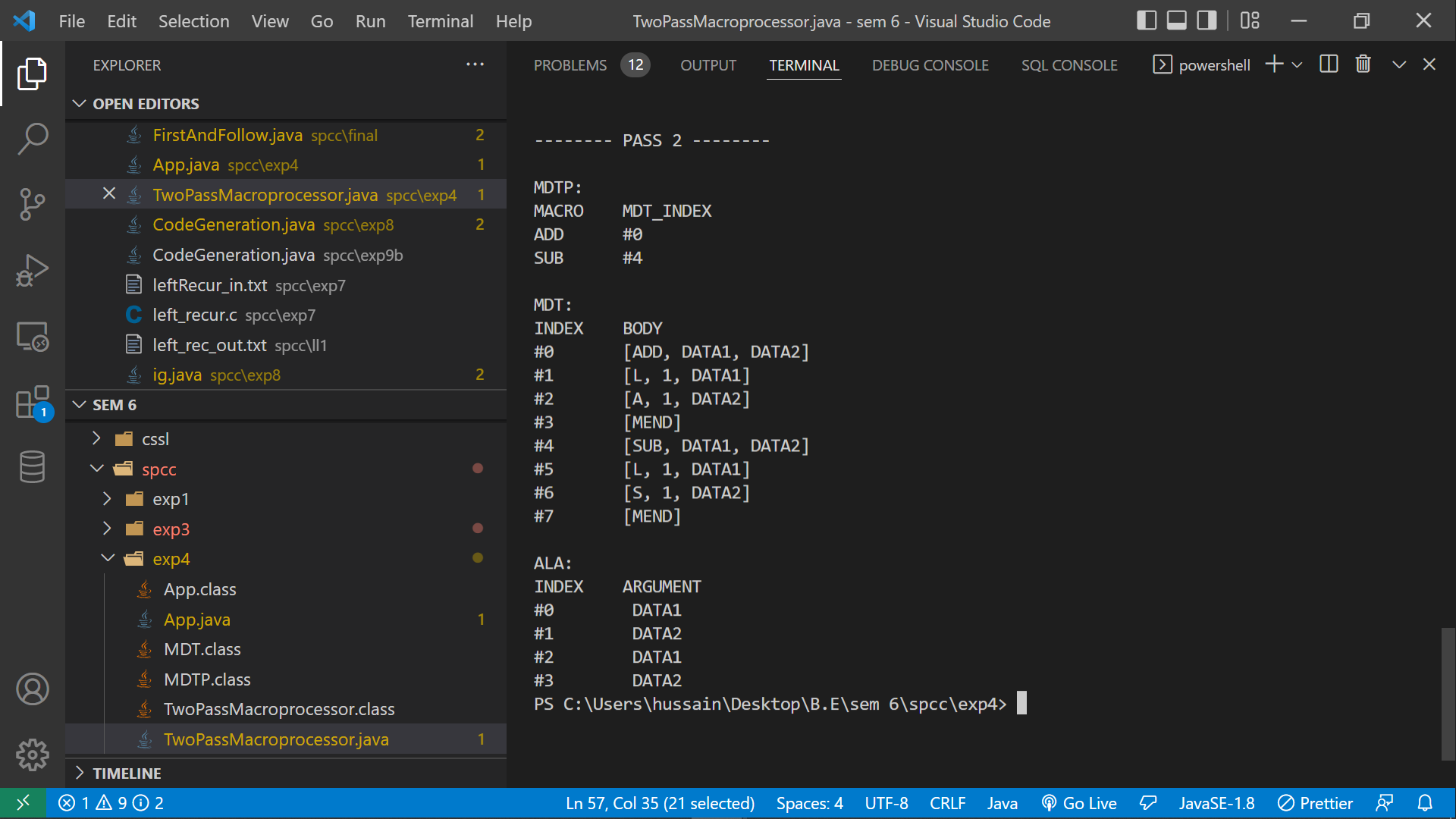
printTables();

}

}

**OUTPUT:**

**- Snapshots of the tables in pass1 and pass 2**

**CONCLUSION:**

A two pass assembler for an 8086 microprocessor is implemented.