
# T.Y.B.Tech (CSE)

## System Software and Compilers(SSC)

## Lab Assignment No – 1

**Name: Aniruddha Shende**

**Roll number: PD-05**

**Batch: D1**

**Panel: D**

Name:- Aniruddha Arun Shende
Roll no:- PD-05
Batch :- D1
Panel :- D PRN: 1032190079

MIT-WPU MIT WORLD PEACE UNIVERSITY

Subject : SSC

Lab Assignment No -1.

Assignment Title : Design of Pass I of Two Pass Assembler.

Aim :- Design suitable data structures & implementation of Pass I & of 2 Pass Assembler for pseudo machine.

Objective :- Design suitable data structures & implement Pass I of 2 Pass Assembler for pseudo machine. Subset should consist of a few instructions from each category & few assembler directives.

Theory:-

1. Design specification of an Assembler: Analysis Phase.

Ans:1. The main task of the Assembler is building the Symbol Table.
For this, it must determine the address with which the symbolic name is associated. To determine the address of a particular symbolic name, we must fix the address of all elements preceding it.

So following are the main tasks of the analysis phase:-

1. Separate the label, opcode & operand.
2. Build the symbol table.
3. Perform LC processing
4. Construct IC.

2. Design of a Two Pass Assembler: Algorithm for Pass I.

Ans 2. Step : 1 loc_cntr := 0; pooltab_ptr := 1; POOLTAB[1] := 1;
Littab_ptr := 1; symtab_ptr := 1;

Step : 2 While next stmt is not an END stmt

(a) If label is present then
this_label = symbol in label field
Enter (this_label, loc_cntr) in SYMTAB

(b) If an LTORG stmt then
i. Process literals
LITTAB[POOLTAB[pooltab_ptr]]....
LITTAB[POOLTAB[pooltab_ptr+1]]-1]
to allocate memory & put the address field
Update loc_cntr.
ii. Pooltab_ptr := pooltab_ptr + 1;
iii. POOLTAB[pooltab_ptr] := littab_ptr;

(c) If START or ORIGIN stmt then
loc_cntr := value in operand field

(d) If an EQU stmt then
this_addr = value of <address spec>
Update the SYMTAB entry

(e) If a declaration stmt then
code = code of declaration stmt,
size = size req by DC/DS
update the SYMTAB entry
loc-cntr = loc_cntr + size
generate intermediate code

(f) If an imperative stmt then
code = m/c code from MOT,
loc_cntr = loc-cntr + length of instr
If operand is literal then
this_lit = literal in operand field
LITTAB[littab_ptr] = this_lit
littab_ptr = littab_ptr+1
else this_entry = SYMTAB entry no of operand.
symtab_ptr = symtab_ptr+1.

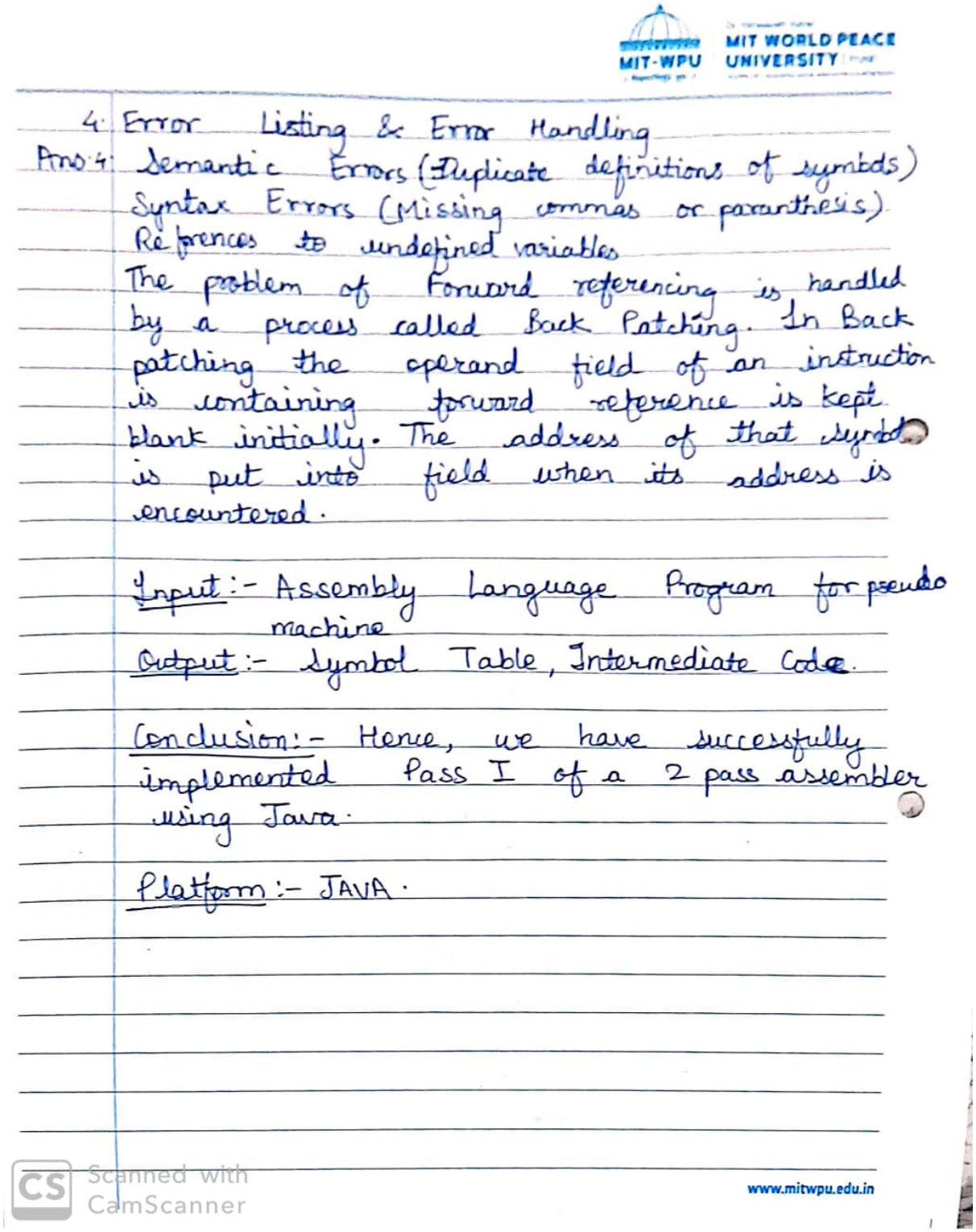
3. Contents of OPTAB:-

Ans.3 Following are the contents of OPTAB

① Mnemonic opcode - shows the name of the instruction.

② Class - Shows whether the instruction is IS, DL & AD.

③ Mnemonic info - shows the machine code & instruction length. For DL & AL stmt this field contains the address of the routine which finds the length of the statement.



MIT-WPU MIT WORLD PEACE UNIVERSITY

4. Error Listing & Error Handling

Ans 4: Semantic Errors (Duplicate definitions of symbols)
Syntax Errors (Missing commas or parenthesis)
References to undefined variables
The problem of Forward referencing is handled by a process called Back Patching. In Back patching the operand field of an instruction is containing forward reference is kept blank initially. The address of that symbol is put into field when its address is encountered.

Input :- Assembly Language Program for pseudo machine
Output :- Symbol Table, Intermediate Code.

Conclusion :- Hence, we have successfully implemented Pass I of a 2 pass assembler using Java.

Platform :- JAVA.





**Code and Output from next page onwards :**

## Java Code:

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

public class assign1 {
    public static void main(String[] args) {
        HashMap<String, ArrayList<String>> optab = new
HashMap<String, ArrayList<String>>();
        HashMap<String, String> registertable = new
HashMap<String, String>();
        HashMap<String, String> bctable = new HashMap<String,
String>();

        LinkedHashMap<String, ArrayList<String>> symtab_table =
new LinkedHashMap<String, ArrayList<String>>();

        //to store the address of the labels/symbols
        HashMap<String, Integer> symbol_addresses = new
HashMap<String, Integer>();

        //all the location_counter, symbol_id, literal_id
        int location_counter = 0;
        int symbol_id = 0;
        int literal_id = 0;
        int start_pointer = 0;

        optab.put("STOP", new
ArrayList<String>(Arrays.asList("(IS,00)")));
        optab.put("ADD", new
ArrayList<String>(Arrays.asList("(IS,01)")));
        optab.put("SUB", new
ArrayList<String>(Arrays.asList("(IS,02)")));
        optab.put("MULT", new
ArrayList<String>(Arrays.asList("(IS,03)")));
        optab.put("MOVER", new
ArrayList<String>(Arrays.asList("(IS,04)")));
        optab.put("MOVEM", new
ArrayList<String>(Arrays.asList("(IS,05)")));
        optab.put("COMP", new
ArrayList<String>(Arrays.asList("(IS,06)")));
```

```
        optab.put("BC", new
ArrayList<String>(Arrays.asList("(IS,07)")));
        optab.put("DIV", new
ArrayList<String>(Arrays.asList("(IS,08)")));
        optab.put("READ", new
ArrayList<String>(Arrays.asList("(IS,09)")));
        optab.put("PRINT", new
ArrayList<String>(Arrays.asList("(IS,10)")));
        optab.put("DC", new
ArrayList<String>(Arrays.asList("(DL,01)")));
        optab.put("DS", new
ArrayList<String>(Arrays.asList("(DL,02)")));
        optab.put("START", new
ArrayList<String>(Arrays.asList("(AD,01)")));
        optab.put("END", new
ArrayList<String>(Arrays.asList("(AD,02)")));
        optab.put("ORIGIN", new
ArrayList<String>(Arrays.asList("(AD,03)")));
        optab.put("EQU", new
ArrayList<String>(Arrays.asList("(AD,04)")));
        optab.put("LTORG", new
ArrayList<String>(Arrays.asList("(AD,05)")));

        registertable.put("AREG", "1");
        registertable.put("BREG", "2");
        registertable.put("CREG", "3");
        registertable.put("DREG", "4");

        bctable.put("LT", "01");
        bctable.put("LE", "02");
        bctable.put("EQ", "03");
        bctable.put("GT", "04");
        bctable.put("GE", "05");
        bctable.put("ANY","06");
        System.out.println();
        File file = new File("input.txt");
        Scanner sc = null;
        try {
            sc = new Scanner(file);
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
        ArrayList<String> lines = new ArrayList<String>();
        while (sc.hasNextLine()) {
```

```
            lines.add(sc.nextLine());
        }
        System.out.println("The given code is : \n");
        ArrayList<ArrayList<String>> ictab = new
ArrayList<ArrayList<String>>();
        for (int i = 0; i < lines.size(); i++) {
            String command = lines.get(i);
            ArrayList<String> words = new ArrayList<String>();
            StringTokenizer tokenizer = new
StringTokenizer(command, " ");
            while (tokenizer.hasMoreTokens()) {
                words.add(tokenizer.nextToken());
            }
            for (int j = 0; j < words.size(); j++) {
                System.out.print(words.get(j)+"\t");
            }
            for (int j = 0; j < words.size(); j++) {
                words.set(j, words.get(j).replace(",", ""));
            }
            System.out.println();


            if (words.get(0).equals("START")) {
                if (words.size() == 2) {
                    ictab.add(new ArrayList<String>(
                            Arrays.asList("-", "(AD,01)", "-",
"(C," + words.get(1) + ")")));
                    location_counter =
Integer.parseInt(words.get(1)) - 1;
                    start_pointer = location_counter;
                } else {
                    ictab.add(new
ArrayList<String>(Arrays.asList("-", "(AD,01)", "-")));
                }
            }

            //end
            else if (words.get(0).equals("END")) {
                location_counter++;
                ictab.add
                    (new
ArrayList<String>(Arrays.asList(Integer.toString(location_counter)
, "(AD,02)", "-", "-")));
                continue;
            }
```

```
            else if (words.get(0).equals("ORIGIN")) {
                //System.out.println("Inside origin");
                if (words.size() == 1) {
                    ictab.add(
                            new ArrayList<String>(Arrays.asList("-
", "(AD,03)", "-", "-")));
                    location_counter = start_pointer;
                } else if (words.size() == 2) {
                    int flag = 0;//if flag becomes 1 then we can
note that a +/- is encountered
                    for (int i1 = 0; i1 < words.get(1).length();
i1++) {
                        if (words.get(1).charAt(i1) == '+') {
                            flag = 1;
                            break;
                        } else if (words.get(1).charAt(i1) == '-')
{
                            flag = 2;
                            break;
                        }
                    }
                    //if the flag is 1 then we have +/- present
                    if (flag == 1 || flag == 2) {
                        String str1 = words.get(1);
                        if (flag == 1) {
                            ArrayList<String> tempo = new
ArrayList<>();
                            String temporary = "";
                            for (int kk = 0; kk < str1.length();
kk++) {
                                if (str1.charAt(kk) == '+') {
                                    tempo.add(temporary);
                                    temporary = "";
                                    continue;
                                }
                                temporary += str1.charAt(kk);
                            }
                            tempo.add(temporary);
                            int loc =
Integer.parseInt(symtab_table.get(tempo.get(0)).get(0));
                            int additional =
Integer.parseInt(tempo.get(1));
```

```
                            location_counter = loc + additional -
1;
                            ictab.add(
                                    new
ArrayList<String>(Arrays.asList("-", "(AD,03)", "-",
                                                "(S," +
symtab_table.get(tempo.get(0)).get(2) + ")")));
                        } else if (flag == 2) {
                            ArrayList<String> tempo = new
ArrayList<>();
                            String temporary = "";
                            for (int kk = 0; kk < str1.length();
kk++) {
                                if (str1.charAt(kk) == '-') {
                                    tempo.add(temporary);
                                    temporary = "";
                                    continue;
                                }
                                temporary += str1.charAt(kk);
                            }
                            tempo.add(temporary);
                            int loc =
Integer.parseInt(symtab_table.get(tempo.get(0)).get(0));
                            int additional =
Integer.parseInt(tempo.get(1));
                            location_counter = loc - additional -
1;
                            ictab.add(
                                    new
ArrayList<String>(Arrays.asList("-", "(AD,03)", "-",
                                                "(S," +
symtab_table.get(tempo.get(0)).get(2) + ")")));
                        }
                    } else {
                        String str = words.get(1);
                        int loc =
Integer.parseInt(symtab_table.get(str).get(0));
                        location_counter = loc - 1;
                        ictab.add(
                                new ArrayList<String>(
                                        Arrays.asList("-",
"(AD,03)", "-", "(S," + symbol_addresses.get(str) + ")")));
                    }
                }
```

```
            }

            // equ implementation
            else if (words.get(0).equals("EQU")) {
                String str1 = words.get(0);
                String str2 = words.get(2);
                // int loc =
Integer.parseInt(symtab_table.get(str1).get(1));
                ictab.add(
                        new ArrayList<String>(
                                Arrays.asList("-", "(AD,04)", "-",
"(S," + symbol_addresses.get(str2) + ")")));
                String str3 = symtab_table.get(str2).get(0);
                symtab_table.get(str1).set(0, str3);
            }
            else if (optab.containsKey(words.get(0)) == false) {
                if (symtab_table.containsKey(words.get(0)) ==
false) {
                    symbol_id++;
                    symtab_table.put(words.get(0),
                            new ArrayList<String>(Arrays.asList("-
", "-", Integer.toString(symbol_id))));
                    symbol_addresses.put(words.get(0), symbol_id);
                }
                if (words.size() == 4) {
                        location_counter++;
                        symtab_table.get(words.get(0)).set(0,
Integer.toString(location_counter));
                        symtab_table.get(words.get(0)).set(1,
"1");
                        words.remove(words.get(0));
                        if (optab.containsKey(words.get(0))) {
                        String operand = words.get(1);
                        String opcode =
optab.get(words.get(0)).get(0);
                        if (registertable.containsKey(operand)) {
                            operand = registertable.get(operand);
                        }
                        if (symtab_table.containsKey(words.get(2))
== false) {
                            symbol_id++;
                            symtab_table.put(words.get(2),
```

```
                                    new
ArrayList<String>(Arrays.asList("-", "-",
Integer.toString(symbol_id))));
                                symbol_addresses.put(words.get(2),
symbol_id);
                            }
                            ictab.add(new
ArrayList<String>(Arrays.asList(Integer.toString(location_counter)
,opcode, operand, "(S," +symbol_addresses.get(words.get(2))+
")")));
                        }
                    }
                    else if (words.size() == 3) {
                        if (words.get(1).equals("EQU")) {

                        }
                        else if (words.get(1).equals("DS")) {
                            location_counter++;
                            ictab.add(
                                    new
ArrayList<String>(Arrays.asList(Integer.toString(location_counter)
, "(DL,02)", "-", "(C," + words.get(2) + ")")));
                            symtab_table.get(words.get(0)).set(0,
Integer.toString(location_counter));
                            symtab_table.get(words.get(0)).set(1,
"1");
                            location_counter +=
Integer.parseInt(words.get(2))-1;
                        }
                        else if (words.get(1).equals("DC")) {
                            location_counter++;
                            ictab.add(
                                    new
ArrayList<String>(Arrays.asList(Integer.toString(location_counter)
, "(DL,01)", "-", "(C," + words.get(2) + ")")));
                            symtab_table.get(words.get(0)).set(0,
Integer.toString(location_counter));
                            symtab_table.get(words.get(0)).set(1,
"1");
                        }
                    }
                    else if (words.size() == 2) {

                    }
```

```
            }
            else if (optab.containsKey(words.get(0))) {
                location_counter++;
                String operand = words.get(1);
                String opcode = optab.get(words.get(0)).get(0);
                if (registertable.containsKey(operand)) {
                    operand = registertable.get(operand);
                }
                if (symtab_table.containsKey(words.get(2)) ==
false) {
                    symbol_id++;
                    symtab_table.put(words.get(2),
                            new ArrayList<String>(Arrays.asList("-
", "-", Integer.toString(symbol_id))));
                    symbol_addresses.put(words.get(2), symbol_id);
                }
                ictab.add(new
ArrayList<String>(Arrays.asList(Integer.toString(location_counter)
,opcode, operand, "(S," +symbol_addresses.get(words.get(2))+
")")));
            }
        }
        System.out.println("\n\nIC Table\n");
        for (int i = 0; i < ictab.size(); i++) {
            for (int j = 0; j < ictab.get(i).size(); j++) {
                System.out.print(ictab.get(i).get(j) + "\t    ");
            }
            System.out.println();
        }
        System.out.println();
        System.out.println("\n\nSymbol Table\n");
        for (Map.Entry<String, ArrayList<String>> entry :
symtab_table.entrySet()) {
            System.out.println(entry.getValue().get(2) + "\t" +
entry.getKey() + "\t" + entry.getValue().get(0) + "\t"
                    + entry.getValue().get(1));
        }
    }
}
```

## Output of the program:

```
ani@Aniruddhas-MacBook-Pro Assign1 %  /usr/bin/env /Library/Java/JavaVirt
ualMachines/jdk-16.jdk/Contents/Home
/bin/java -XX:+ShowCodeDetailsInExceptionMessages -cp "/Users/ani/Library
/Application Support/Code/User/works
paceStorage/2ba75a474f36b777aa8a9f82e120969e/redhat.java/jdt_ws/Assign1_5
85726ba/bin" assign1

The given code is :

START   100
MOVER   AREG,   A
L1      ADD     BREG,   A
MOVER   BREG,   B
ORIGIN  L1
MOVER   BREG,   A
A       DS      5
B       DC      5
END


IC Table

-           (AD,01)         -       (C,100)
100         (IS,04)         1       (S,1)
101         (IS,01)         2       (S,1)
102         (IS,04)         2       (S,3)
-           (AD,03)         -       (S,2)
101         (IS,04)         2       (S,1)
102         (DL,02)         -       (C,5)
107         (DL,01)         -       (C,5)
108         (AD,02)         -       -



Symbol Table

1       A       102     1
2       L1      101     1
3       B       107     1
ani@Aniruddhas-MacBook-Pro Assign1 %
```