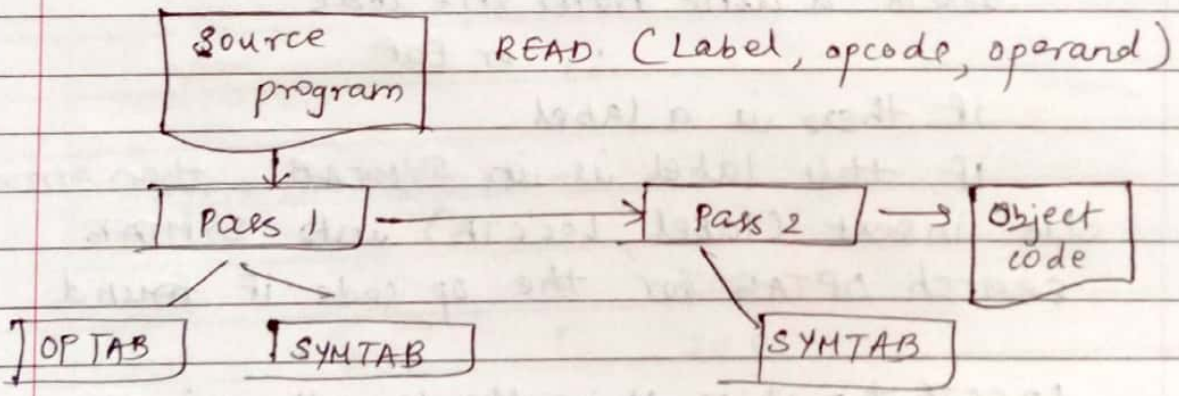
2. Pass-2 Assembler

Aim : To design data structure for Pass-2 Assembler

Problem Statement :.
Implement Pass-II of two pass assembler for pseudo-machine in Java using object oriented features. The output of assignment-1 (intermediate file and symbol table) should be input for this assignment.

Theory:-

A Simple Two Pass Assembler Implementation



Data Structures:-

Location counter (LC): points to the next location where the code will be placed.

Scanned with CamScanner

Op-code translation table :-

Symbol table (ST)

String storage buffer. (SSB)

Forward references table (FRT)

Algorithm:-

```
begin
        if starting address is given
                LOCCTR = starting address;
else
    LOCCTR =0;
    while OPCODE! =END do
    begin
    read a line from the code
                        ;; or EOF
    if there is a label
    if this label is in SYMTAB, then error
else insert (label, LOCCTR) into SYMTAB
  search OPTAB for the op code if found

  LOCCTR + = N ;; N is the length of this
  instruction (4 for MIPS)

  else if this is an assembly directive

  else error
```

Scanned with CamScanner

write line to intermediate file end

Program size = LOCCTR - starting address;

end.

Input :-

AD01	C	200		
IS	04	1	L	1
IS	05	1	S	1
IS	04	2	L	2
IS	04	3	S	3
AD	05			
IS	01	3	L	3
IS	00			
DL	02	C	1	
DL	02	C	1	
AD	02			

Expected Output :-

200	04	1	204
201	05	1	208
202	04	2	210
203	04	3	209
204	00	0	004
205	00	0	006
206	01	3	205
207	00	0	000

```
208  209
210   00   0 001
```

Conclusion:-

Thus we have generated Machine Code for the source program.

Assignment No. 02 [Pass 2 Assembler]

Problem Satement: Implement Pass-II of two pass assembler for pseudo-machine in Java using object oriented features. The output of assignment-1 (intermediate file and symbol table) should be input for this assignment.

1. Pass 2 Program:

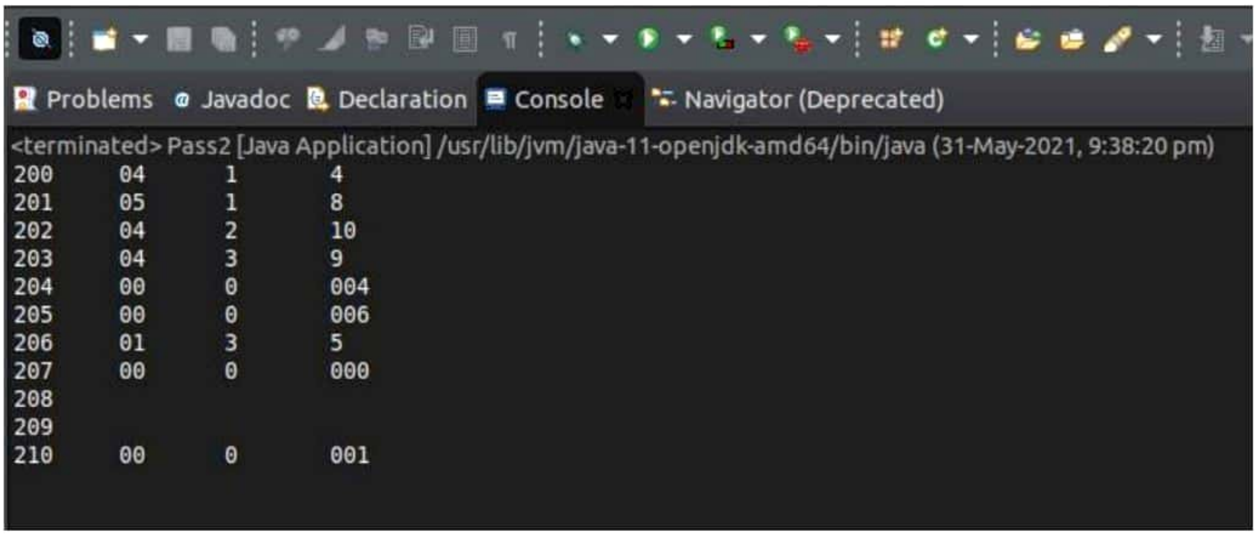
```
import java.io.BufferedReader; import
java.io.BufferedWriter; import
java.io.FileReader; import java.io.FileWriter;
import java.io.IOException; import
java.lang.reflect.Array; import
java.util.ArrayList; import
java.util.Hashtable; import java.util.Map;
public class Pass2 { public static void
main(String[] args) { try {

                    //1. Read Intermediate code file
                      String f ="/home/sagar-ravan/Desktop/IC_new.txt";
                      FileReader fw =new FileReader(f);
                      BufferedReader IC_file=new BufferedReader(fw);

                      //2.Read Symbol table file
                      String f1="/home/sagar-ravan/Desktop/SYMTAB.txt";
                      FileReader fs=new FileReader(f1);
                      BufferedReader symtab_file=new BufferedReader(fs);
                    symtab_file.mark(500);

                      //3.Read Literal table file
                      String f2="/home/sagar-ravan/Desktop/LITTAB.txt";
                      FileReader fl=new FileReader(f2);
                      BufferedReader littab_file=new BufferedReader(fl);
                    littab_file.mark(500);


                      //4.create  littab array and hashtable for symbol table


                    String littab[][]=new String[10][2] ;

                    Hashtable<String, String> symtab = new Hashtable<String,
String>();
                    String str;
                    int z=0;
                    //5.Read LITTAB.txt
                    while ((str = littab_file.readLine()) != null) {

                          littab[z][0]=str.split("\\s+")[0]; //first word
                          littab[z][1]=str.split("\\s+")[1]; //second word z++;
                    }
                    //6.Read SYMTAB.txt
```

```
                   while ((str = symtab_file.readLine()) != null) {
                   symtab.put(str.split("\\s+")[0], str.split("\\s+")[1]); }
            //7.Read POOLTAB.txt
                   String f3 = "/home/sagar-ravan/Desktop/POOLTAB.txt";
                   FileReader fw3 = new FileReader(f3);
                   BufferedReader pooltab_file = new BufferedReader(fw3);



                   ArrayList<Integer> pooltab = new ArrayList<Integer>();
                   String t;
                   while ((t = pooltab_file.readLine()) != null) {
                         pooltab.add(Integer.parseInt(t));
                   }

                   int pooltabptr = 1;
                   int temp1 = pooltab.get(0);    //dry run
                   int temp2 = pooltab.get(1);

                   //7.Read IC.txt
                   String sCurrentLine;
                   sCurrentLine = IC_file.readLine();
                   int locptr=0;
                   //locptr=Integer.parseInt(sCurrentLine.split("\\s+")[3]);
                   locptr=Integer.parseInt(sCurrentLine.split("\t")[3]);
                   while ((sCurrentLine = IC_file.readLine()) != null) {


                          System.out.print(locptr+"\t");


                         String s0 = sCurrentLine.split("\t")[0]; //contains
statement type

                         String s1 = sCurrentLine.split("\t")[1]; //contains
statement code

                         if (s0.equals("IS")) {

                               System.out.print(s1+"\t"); if
                               (sCurrentLine.split("\t").length == 5) {


                                     System.out.print(sCurrentLine.split("\t")[2]
+  "\t");
                                     //7.2 if third character is L
                                     if (sCurrentLine.split("\t")[3].equals("L"))
{ int add =
Integer.parseInt(sCurrentLine.split("\t")[4]);


      //machine_code_file.write(littab[add-1][1]);
                                           System.out.print(littab[add-1][1]);

                                         }
                                       //7.3 or if third character is S
                                     if (sCurrentLine.split("\t")[3].equals("S"))
{ int add1 =
Integer.parseInt(sCurrentLine.split("\t")[4]);

                                           //search for the 4th word in symbol
table int i = 1; String l1;
                                            for (Map.Entry m : symtab.entrySet())
{
                                  if (i == add1) {

                                                        System.out.print((String)
m.getValue());
                                                  }
                                                  i++;
                                         }

                                         }
                              }  else  {

                          System.out.print("0\t000");
                              }
                        }



                        //DRY RUN is a must

                        if (s0.equals("AD")) {
                              littab_file.reset();
                              if (s1.equals("05")) {  //if it is
LTORG int j = 1; while (j < temp1) { littab_file.readLine();
                                         }
                             while (temp1 < temp2) {

                                            System.out.print("00\t0\t00" +
littab_file.readLine().split("'")[1]);
                                            if(temp1<(temp2-1)){
                                                  locptr++;

                                                  System.out.println();

                                                  System.out.print(locptr+"\t");
                                            }
                                            temp1++;
                                    } temp1 =
                                    temp2;
                                    pooltabptr++;
                                    if (pooltabptr < pooltab.size()) {
                                    temp2 = pooltab.get(pooltabptr); }
                              } int j =
                              1;
                              if (s1.equals("02")) {  //if it is
"END" stmt
                                    String s;
                                    while ((s = littab_file.readLine()) != null)
{
                                  if (j >= temp1)

                                                  System.out.print("00\t0\t00" +
s.split("'")[1]); j++;
                                         }
                              }
                        }



                               if(s0.equals("DL")&&s1.equals("01")){     //if it
is DC stmt
```

```
                              System.out.print("00\t0\
t00"+sCurrentLine.split("'")[1]);

                         }



                         locptr++;

                         System.out.println();
                  }
                  IC_file.close();
                  symtab_file.close();
                  littab_file.close();
                  pooltab_file.close();

            }  catch (IOException e)  {
                  e.printStackTrace();
            }

      }
```

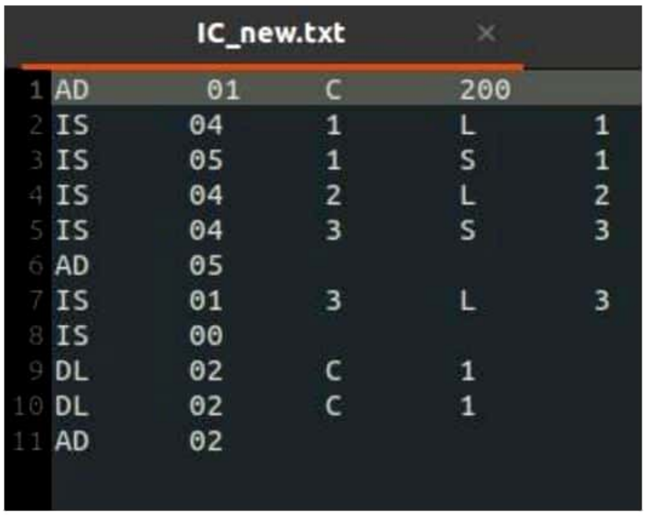
PASS 2 - ASSEMBLER OUTPUT:

PASS- 2 OUTPUT:



```
<terminated> Pass2 [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (31-May-2021, 9:38:20 pm)
200     04      1       4
201     05      1       8
202     04      2       10
203     04      3       9
204     00      0       004
205     00      0       006
206     01      3       5
207     00      0       000
208
209
210     00      0       001
```

IC_New.txt

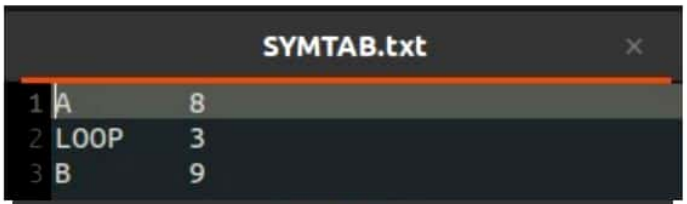


IC_new.txt

```
AD      01     C      200
IS     04     1      L      1
IS     05     1      S      1
IS     04     2      L      2
IS     04     3      S      3
AD     05
IS     01     3      L      3
IS     00
DL     02     C      1
DL     02     C      1
AD     02
```

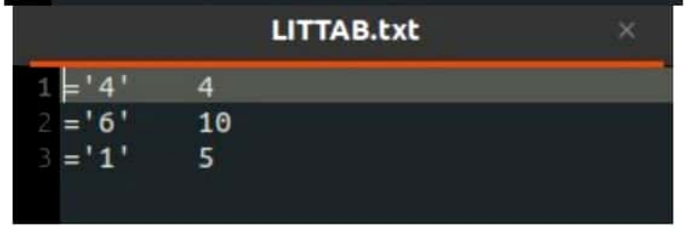
Input.txt

LITTAB.txt



SYMTAB.txt

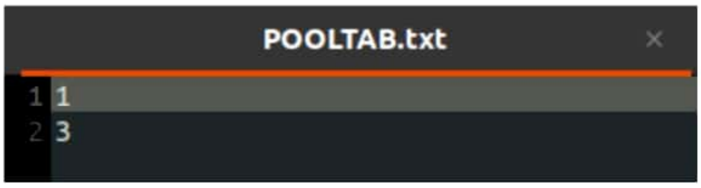
```
A       8
LOOP    3
B       9
```



LITTAB.txt

```
='4'    4
='6'    10
='1'    5
```

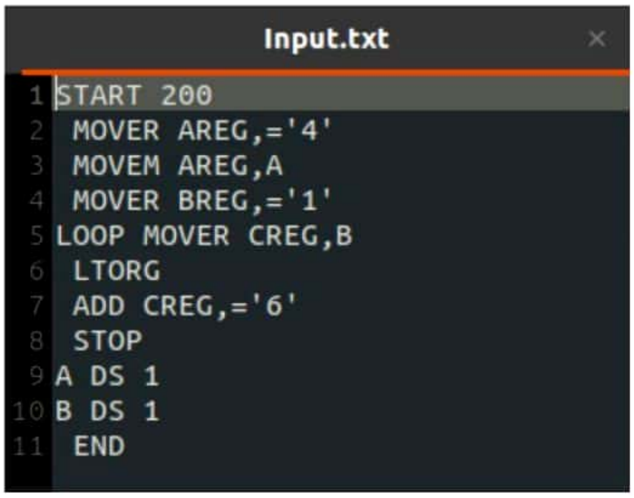
POOLTAB.txt



POOLTAB.txt

```
1
3
```

SYMTAB.txt



Input.txt

```
START 200
 MOVER AREG,='4'
 MOVEM AREG,A
 MOVER BREG,='1'
LOOP MOVER CREG,B
 LTORG
 ADD CREG,='6'
 STOP
A DS 1
B DS 1
 END
```