SSC LAB ASSIGNMENT NO.1

DESIGN OF PASS 1 OF 2 PASS ASSEMBLER

NAME : KETAKI PATIL

ROLL NO : PA-17

BATCH : A1

BATCH A1 INPUT CODE :

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

CODE :

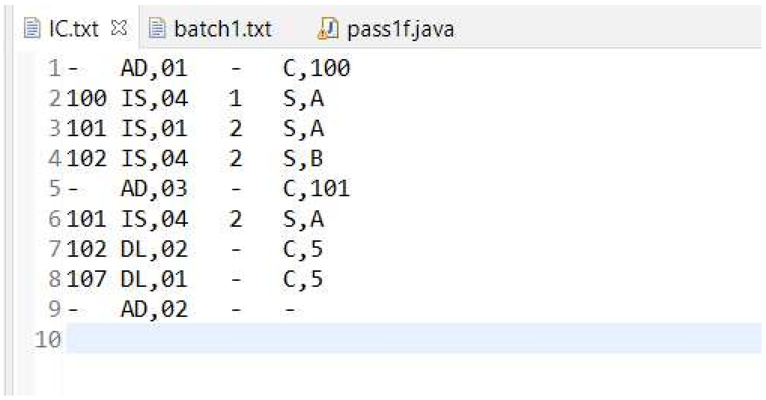
```
/*
 SSC LAB ASSIGNMENT NO.1
DESIGN OF PASS 1 OF 2 PASS ASSEMBLER
NAME : KETAKI PATIL
ROLL NO : PA-17
BATCH : A1
 */

package ssc_lab;

import java.util.*;
import java.io.*;

public class pass1f{

    public static void main(String[] args) throws IOException{
       System.out.println("SSC LAB 1 :");
         System.out.println("PASS 1 OF ASSEMBLER");
         BufferedReader br=null;
         FileReader fr=null;
         FileWriter fw=null;
         BufferedWriter bw=null;

         int LC=0;
         String Instropcode=null;

         String inputfilename = "C:\\Users\\ketak\\Desktop\\batch1.txt";
         fr = new FileReader(inputfilename);
         br = new BufferedReader(fr);

         String OUTPUTFILENAME ="IC.txt";
         fw= new FileWriter(OUTPUTFILENAME);
         bw= new BufferedWriter(fw);

         Hashtable<String, String> is = new Hashtable<>();
         is.put("STOP", "00");
         is.put("ADD", "01");
         is.put("SUB", "02");
         is.put("MULT", "03");
         is.put("MOVER", "04");
         is.put("MOVEM", "05");
```

```java
is.put("COMP", "06");
is.put("BC", "07");
is.put("DIV", "08");
is.put("READ", "09");
is.put("PRINT", "10");
System.out.println("Mappings of imperative statements : " + is);

Hashtable<String, String> ad = new Hashtable<>();
ad.put("START", "01");
ad.put("END", "02");
ad.put("ORIGIN", "03");
ad.put("EQU", "04");
ad.put("LTORG", "05");
System.out.println("Mappings of Assembler Directive : " + ad);

Hashtable<String, String> d1 = new Hashtable<>();
d1.put("DC", "01");
d1.put("DS", "02");
System.out.println("Mappings of Declarative Statement : " + d1);
Hashtable<String, Integer> LC1 = new Hashtable<String, Integer>();

String name=null;
Hashtable<String, String> symtab = new Hashtable<>();
String reg="";
String ICcode=null;
while ((name = br.readLine()) != null) {
    //System.out.println(name);
    String s1 = name.split(" ")[0];
    String temp;
    if (s1.equals("START")){
        String s2 = name.split(" ")[1];
        LC=Integer.parseInt(s2);
        for (Map.Entry m : ad.entrySet()) {
            if (s1.equals(m.getKey())) {
                Instropcode=(String)m.getValue();
            }
        }
        ICcode="-\t"+ "AD,"+ Instropcode + "\t-" + "\tC," + LC;

    }
    else if (s1.equals("END")){
        for (Map.Entry m : ad.entrySet()) {
            if (s1.equals(m.getKey())) {
                Instropcode=(String)m.getValue();
            }
        }
        ICcode="-\t"+ "AD,"+ Instropcode+"\t-\t-";
    }
    else if(s1.equals("ORIGIN")){
        String s2 = name.split(" ")[1];
        boolean isNumeric = s2.chars().allMatch( Character::isDigit );
        if(isNumeric){
            LC= Integer.parseInt(s2);
        }
        else{

            String ss1=s2.split("\\+")[0];
            int ss2= Integer.parseInt(s2.split("\\+")[1]);
            LC= LC1.get(ss1)+ss2;
        }
        for (Map.Entry m : ad.entrySet()) {
            if (s1.equals(m.getKey())) {
                Instropcode=(String)m.getValue();
            }
        }
```

```
            ICcode="-\t"+ "AD,"+ Instropcode + "\t-" + "\tC," + LC;

        }
        else if((name.split(" ")[1]).equals("EQU")){
            //symtab.put(s1,""+LC);
            String s2= name.split(" ")[1];
            String s3= name.split(" ")[2];
            int t= LC1.get(s3);
            symtab.put(s1,""+t);

            ICcode= "-\t"+ "AD,04\t-\tS,"+s3;
        }
        else if(is.containsKey(s1)){
            for (Map.Entry m : is.entrySet()) {
                if (s1.equals(m.getKey())) {
                    Instropcode=(String)m.getValue();
                }
            }
            String s2= name.split(" ")[1];
            if(s2.equals("AREG")){reg="1";}
            if(s2.equals("BREG")){reg="2";}

            String s3= name.split(" ")[2];
            symtab.put(s3, "");
            ICcode=LC+"\t"+ "IS,"+ Instropcode + "\t" + reg +"\tS,"+s3;
            LC=LC+1;
        }
        else if((name.split(" ")[1]).equals("DS")|| name.split(" ")[1].equals("DC")){
            symtab.replace(s1,""+LC);
            String s2= name.split(" ")[1];
            String s3= name.split(" ")[2];

            if(s2.equals("DS")){ICcode=LC+"\tDL,02\t-\tC,"+s3;}
            if(s2.equals("DC")){ICcode=LC+"\tDL,01\t-\tC,"+s3;}

            LC=LC+ (Integer.parseInt(s3));
        }
        else {
            //System.out.println(name);
            symtab.put(s1, ""+LC);
            String s2 = name.split(" ")[1];
            for (Map.Entry m : is.entrySet()) {
                if (s2.equals(m.getKey())) {
                    Instropcode=(String)m.getValue();
                }
            }

            String s3 = name.split(" ")[2];
            if (s3.equals("AREG")) { reg="1";}
            if(s3.equals("BREG")){reg="2";}
            String s4 = name.split(" ")[3];
            symtab.put(s4,"");
            LC1.put(s1,LC);
            ICcode=LC+"\t"+ "IS,"+ Instropcode + "\t" + reg +"\tS,"+s4;
            LC=LC+1;
        }
        bw.write(ICcode+"\n");
    }
    br.close();
    bw.close();
    System.out.println("\n-----------------------");
    System.out.println("SYMTAB is:");
    System.out.println("------------------------");
    for (Map.Entry m : symtab.entrySet()) {
        System.out.println(m.getKey()+"\t"+m.getValue()+"\n");
```

```
            }
        }
}
```

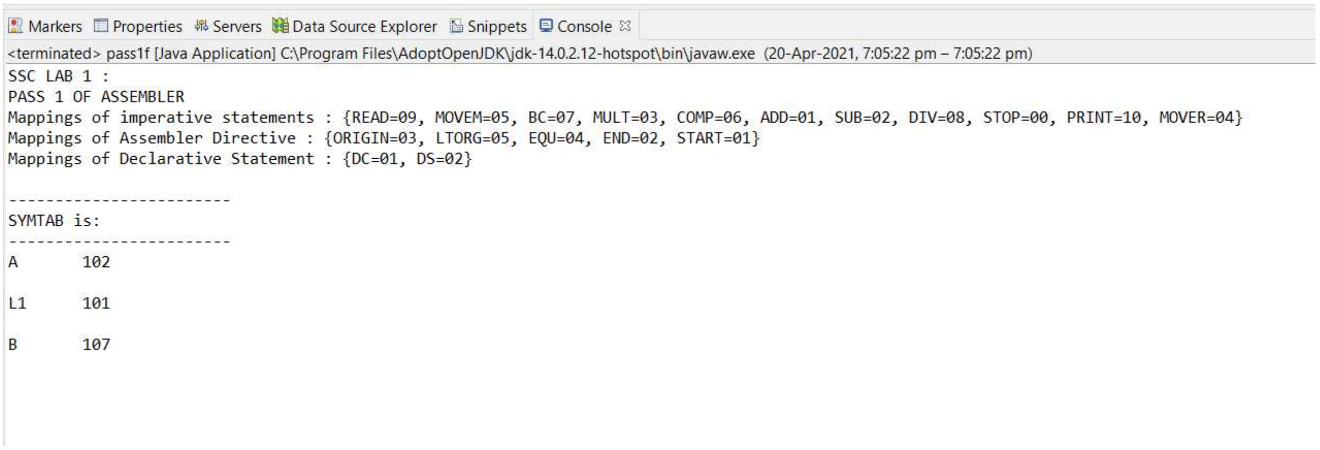
OUTPUT IC SS :



IC.txt batch1.txt pass1f.java

```
-    AD,01    -    C,100
100  IS,04    1    S,A
101  IS,01    2    S,A
102  IS,04    2    S,B
-    AD,03    -    C,101
101  IS,04    2    S,A
102  DL,02    -    C,5
107  DL,01    -    C,5
-    AD,02    -    -
```

OUTPUT SYMTAB SS:



Markers Properties Servers Data Source Explorer Snippets Console

```
<terminated> pass1f [Java Application] C:\Program Files\AdoptOpenJDK\jdk-14.0.2.12-hotspot\bin\javaw.exe  (20-Apr-2021, 7:05:22 pm – 7:05:22 pm)
SSC LAB 1 :
PASS 1 OF ASSEMBLER
Mappings of imperative statements : {READ=09, MOVEM=05, BC=07, MULT=03, COMP=06, ADD=01, SUB=02, DIV=08, STOP=00, PRINT=10, MOVER=04}
Mappings of Assembler Directive : {ORIGIN=03, LTORG=05, EQU=04, END=02, START=01}
Mappings of Declarative Statement : {DC=01, DS=02}

------------------------
SYMTAB is:
------------------------
A       102

L1      101

B       107
```