Experiment No. 05

Aim- Design an implementation of pass II of 2 pass assembler

Requirement- Java and printout pages

Theory-

1. Generate object code by converting symbolic op-code into respective numeric op-code

2. Generate data for literals and look for values of symbols

3. Assemble instructions (generate opcode and look up addresses)

4. Generate data values defined by BYTE, WORD

5. Perform processing of assembler directives not done in Pass 1

6. Write the object program and the assembly listing

Algorithm-

begin

 read first input line (from intermediate file)

 if OPCODE ='START' then

 begin

 write listing line

 read next input line

 end {if START}

write Header record to object program

initialize first Text record

while OPCODE != 'END' do

begin

  if this is not a comment line then

begin

    search OPTAB for OPCODE

   if found then

   begin

    if there is a symbol in OPERAND field then

    begin

     search SYMTAB for OPERAND

     if found then

   store symbol value as operand address

     else

      begin

      store 0 as operand address

      set error flag (undefined symbol)

     end

    end {if symbol}

```
        else

          store 0 as operand address

          assemble the object code instruction

    end {if opcode found}

      else if OPCODE ='BYTE' or 'WORD' then

        convert constant to object code

  if object code will not fit into the current Text record then

      begin

        write Text record to object program

        initialize new Text record

        end

  add object code to Text record

      end {if not comment}

  write listing line

  read next input line

  end(while not END)

  write last Text record to object program

  write End record to object program

  write last listing line

end{Pass 2}
```

Code-

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

class pass2
{
    static int lc=0,sti=0,di=0,i,j,li=0,ri=0,r,lci=0,mci=0,fin=-1,index=0,base=0,ln=0;
    static int[][] symtab = new int[100][3];
    static int[][] littab = new int[100][3];
    static int[][] reg = new int[50][2];
    static String[] sym = new String[100];
    static String[] data = new String[100];
    static String[][] macode = new String[100][4];

    public static int getbr(int n)
    {
        int min=100,pos=-1;
        for(i=0;i<50;i++)
        {
            if(min>Math.abs(reg[i][0]-n) && reg[i][1]==1)
            {
                min=Math.abs(reg[i][0]-n);
                pos = i;
                //System.out.println(min+" "+pos);
            }
        }
        return pos+1;
    }

    public static int getsymlc(String s)
    {
        for(i=0;i<sti;i++)
        {
            if(s.equals(sym[i]))
            {
                return symtab[i][0];
            }
        }
        return -1;
    }

    public static int getlitlc(String s)
    {
        for(i=0;i<di;i++)
        {
            if(s.equals(data[i]))
            {
                return littab[i][0];
```

```
          }
      }
      return -1;
   }

   public static void assmc(String a, String b, String c, String d)
   {
      macode[mci][0] = a;
      macode[mci][1] = b;
      macode[mci][2] = c;
      macode[mci][3] = d;
      //System.out.println(macode[mci][0]+" "+macode[mci][1]+" "+macode[mci][2]);
      mci++;
   }

   public static void main(String []args)
   {
      BufferedReader reader;

      for(i=0;i<50;i++)
      {
         for(j=0;j<2;j++)
         {
            reg[i][j] = 0;
         }
      }

      try
      {
         reader = new BufferedReader(new FileReader("symboltable.txt"));
         String line = reader.readLine();
         while(line!=null)
         {
            String[] words = line.split("\\s+");
            sym[sti] = words[0];
            symtab[sti][0] = Integer.parseInt(words[1]);
            symtab[sti][1] = Integer.parseInt(words[2]);
            symtab[sti][2] = Integer.parseInt(words[3]);
            sti++;
            line = reader.readLine();
         }
         //for(i=0;i<sti;i++) System.out.println(sym[i]+" "+symtab[i][0]+" "+symtab[i][1]+"
"+symtab[i][2]);

         reader = new BufferedReader(new FileReader("literaltable.txt"));
         line = reader.readLine();
         while(line!=null)
         {
            String[] words = line.split("\\s+");
            data[di] = words[0];
```

```
            littab[di][0] = Integer.parseInt(words[1]);
            littab[di][1] = Integer.parseInt(words[2]);
            littab[di][2] = Integer.parseInt(words[3]);
            di++;
            line = reader.readLine();
        }
        //for(i=0;i<di;i++) System.out.println(data[i]+" "+littab[i][0]+" "+littab[i][1]+"
"+littab[i][2]);
        //System.out.println(getlitlc("=F'4'"));
    }
    catch (IOException e) { e.printStackTrace(); }
    // 1 - Base 2 - Index
    try
    {
        reader = new BufferedReader(new FileReader("prg.txt"));
        String line = reader.readLine();
        String[] words = line.split("\\s+");
        ln++;
        //System.out.println(sym[0]+" "+symtab[0][0]);
        while (!words[1].equals("END"))
        {
            if(words[1].equals("USING"))
            {
                String[] opr = words[2].split(",");
                if(opr[0].equals("*"))
                {
                    r = Integer.parseInt(opr[1]);
                    reg[r-1][0] = lc;
                    reg[r-1][1] = 1;
                    //System.out.println(r+" "+reg[r-1][0]+" "+reg[r-1][1]);
                }
                else if(opr[1].matches("[0-9]+"))
                {
                    r = Integer.parseInt(opr[1]);
                    reg[r-1][0] = getsymlc(opr[0]);
                    reg[r-1][1] = 1;
                    //System.out.println(r+" "+reg[r-1][0]+" "+reg[r-1][1]);
                }
                else
                {
                    r = getsymlc(opr[1]);
                    reg[r-1][0] = getsymlc(opr[0]);
                    reg[r-1][1] = 1;
                    //System.out.println(r+" "+reg[r-1][0]+" "+reg[r-1][1]);
                }
            }
            else if(words[1].equals("LA"))
            {
                String[] opr = words[2].split(",");
                r = Integer.parseInt(opr[0]);
```

```
            lci = Math.abs(getsymlc(opr[1])-reg[r-1][0]);
            if(fin==-1) index=0;
            assmc(Integer.toString(ln),Integer.toString(lc),words[1],opr[0]+","+Integer.toStri
ng(lci)+"("+Integer.toString(index)+","+Integer.toString(getbr(lci))+")");
            lc+=4;
        }
        else if(words[1].length()>1 && words[1].charAt(1) == 'R' &&
(words[1].equals("SR") || words[1].equals("AR") ||words[1].equals("LR")))
        {
            String[] opr = words[2].split(",");
            if(opr[0].matches("[0-9]+"))
assmc(Integer.toString(ln),Integer.toString(lc),words[1],opr[0]+","+Integer.toString(getsymlc
(opr[1])));
            else if(opr[1].matches("[0-9]+"))
assmc(Integer.toString(ln),Integer.toString(lc),words[1],Integer.toString(getsymlc(opr[0]))+",
"+opr[1]);
            else
assmc(Integer.toString(ln),Integer.toString(lc),words[1],Integer.toString(getsymlc(opr[0]))+",
"+Integer.toString(getsymlc(opr[1])));
            lc+=2;
            if(opr[1].equals("INDEX")) index=getsymlc(opr[0]);
        }
        else if(words[1].equals("L") || words[1].equals("ST") )
        {
            String[] opr = words[2].split(",");
            if(opr[1].charAt(0) == '=')
            {
                r = getsymlc(opr[0]);
                lci = Math.abs(getlitlc(opr[1])-6);
                assmc(Integer.toString(ln),Integer.toString(lc),words[1],Integer.toString(r)+","
+Integer.toString(lci)+"("+Integer.toString(index)+","+Integer.toString(getbr(r))+")");
            }
            else
            {
                String[] opr21 = opr[1].split("\\(");
                String op21 = opr21[0];
                //System.out.println(getsymlc(op21));
                opr21 = opr21[1].split("\\)");
                String op22 = opr21[0];
                //System.out.println(op22);
                if(op22.equals("INDEX") && op21.equals("DATA1"))
assmc(Integer.toString(ln),Integer.toString(lc),words[1],Integer.toString(getsymlc(opr[0]))+",
0("+Integer.toString(index)+","+Integer.toString(getbr(getsymlc(op21)))+")");
                else
assmc(Integer.toString(ln),Integer.toString(lc),words[1],Integer.toString(getsymlc(opr[0]))+",
"+Integer.toString(getsymlc(op21)-
6)+"("+Integer.toString(index)+","+Integer.toString(getbr(getsymlc(op21)))+")");
            }
            lc+=4;
        }
```

```java
        else if(words[1].equals("A") || words[1].equals("C"))
        {
          String[] opr = words[2].split(",");
          lci = getlitlc(opr[1]);
          assmc(Integer.toString(ln),Integer.toString(lc),words[1],Integer.toString(getsyml
c(opr[0]))+","+Integer.toString(lci-6)+"(0,"+Integer.toString(getbr(lci-6))+")");
          lc+=4;
        }
        else if(words[1].equals("BNE"))
        {
          lci = getsymlc(words[2]) - reg[getbr(7)-1][0];
          assmc(Integer.toString(ln),Integer.toString(lc),"BC","7,"+Integer.toString(lci)+"(
0,"+Integer.toString(getbr(7))+")");
          lc+=4;
        }
        else if(words[1].equals("BR"))
        {
          assmc(Integer.toString(ln),Integer.toString(lc),"BCR",Integer.toString(getbr(Inte
ger.parseInt(words[2])))+","+words[2]);
          lc+=4;
        }
        else if(words[1].equals("LTORG"))
        {
          //System.out.println(di);
          i=1;
          //System.out.println(data[i]);
          String[] opr21 = data[0].split("\\(");
          String op21 = opr21[0];
          //System.out.println(getsymlc(opr21[1]));
          opr21 = opr21[1].split("\\)");
          String op22 = opr21[0];
          assmc(Integer.toString(ln),Integer.toString(littab[0][0]),Integer.toString(getsyml
c(op22)),"");
          //System.out.println("in");
          opr21 = data[1].split("\\'");
          //System.out.println(opr21[0]);
          assmc(Integer.toString(ln),Integer.toString(littab[1][0]),opr21[1],"");
          opr21 = data[2].split("\\'");
          assmc(Integer.toString(ln),Integer.toString(littab[2][0]),opr21[1],"");
          opr21 = data[3].split("\\'");
          assmc(Integer.toString(ln),Integer.toString(littab[3][0]),opr21[1],"");
        }
        else if(words[1].equals("DS"))
        {
          assmc(Integer.toString(ln),Integer.toString(getsymlc(words[0])),"",".");
        }
        else if(words[1].equals("DC"))
        {
          String[] opr21 = words[2].split("\\'");
```

```
            assmc(Integer.toString(ln),Integer.toString(getsymlc(words[0])),"",opr21[1]);
         }
         line = reader.readLine();
         words = line.split("\\s+");
         ln++;
      }
      reader.close();
   }
   catch (IOException e) { e.printStackTrace(); }
   try(OutputStream fw = new FileOutputStream("machinecode.txt"))
      {
         String content = "LN LC Instruction/datum"+System.getProperty("line.separator");
         fw.write(content.getBytes(),0,content.length());
         for(i=0;i<mci;i++)
         {
            //BufferedWriter bw = new BufferedWriter(fw);
            content = macode[i][0]+"    "+macode[i][1]+"    "+macode[i][2]+"
"+macode[i][3]+System.getProperty("line.separator");
            //System.out.println(content);
            fw.write(content.getBytes(),0,content.length());
            //System.out.println(data[i]+" "+littab[i][0]+" "+littab[i][1]+" "+littab[i][2]);
         }
      }
      catch (IOException e) { e.printStackTrace(); }
      System.out.println("Check file machinecode.txt");
   }
}
```
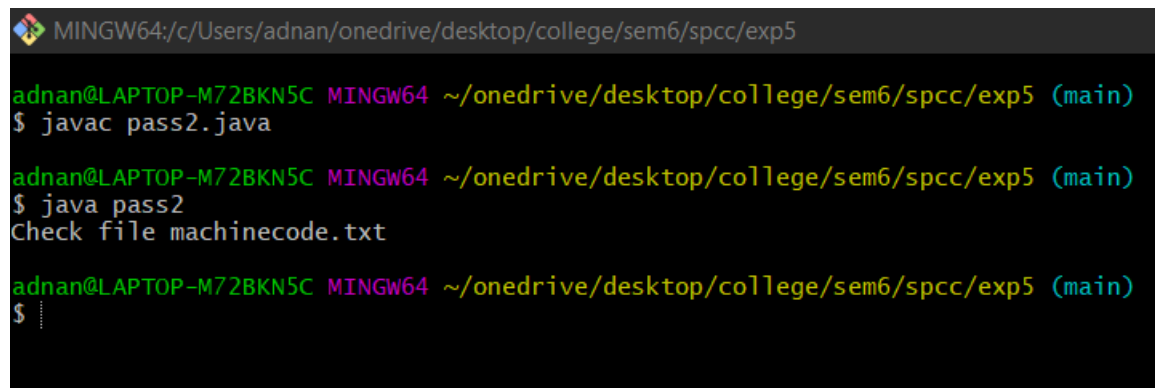
Output-

Execution:

Input File:

```
prg - Notepad

File   Edit   View

PRGAM2 START 0
        USING *,15
        LA    15,SETUP
        SR    TOTAL,TOTAL
AC      EQU   2
INDEX   EQU   3
TOTAL   EQU   4
DATABASE EQU 13
SETUP   EQU   *
        USING SETUP,15
        L     DATABASE,=A(DATA1)
        USING DATAAREA,DATABASE
        SR    INDEX,INDEX
LOOP    L     AC,DATA1(INDEX)
        AR    TOTAL,AC
        A     AC,=F'6'
        ST    AC,SAVE(INDEX)
        A     INDEX,=F'4'
        C     INDEX,=F'8000'
        BNE   LOOP
        LR    1,TOTAL
        BR    14
        LTORG
SAVE    DS    2000F
DATAAREA EQU *
DATA1   DC    F'01'
        END
```

Machine Code:

```
machinecode - Note...     —  □  ×

File    Edit    View              ⚙

LN LC Instruction/datum
3    0      LA 15,6(0,15)
4    4      SR 4,4
11   6      L 13,42(0,15)
13   10     SR 3,3
14   12     L 2,0(3,13)
15   16     AR 4,2
16   18     A 2,46(0,15)
17   22     ST 2,58(3,15)
18   26     A 3,50(0,15)
19   30     C 3,54(0,15)
20   34     BC 7,6(0,15)
21   38     LR 1,4
22   40     BCR 15,14
23   48     8064
23   52     6
23   56     4
23   60     8000
24   64     .
26   8064   01

Ln 1, Col 1   100%   Windows (CRLF)   UTF-8
```

Conclusion- Thus we have Implemented program for pass 2 of two pass Assembler.