Experiment No:1

Pass I:

Title: Design suitable Data Structure & implement Pass-I and Pass-II of two-pass assembler for pseudo-machine.Implementation should consist of a few instruction from each category & few assembler directives.The output of Pass-I(Intermediate code file & symbol table)should be input for pass-II.

abc.java

```
import java.io.*;

import java.util.*;

public class abc {

        public static void main(String args[]) throws
IOException,FileNotFoundException,ArrayIndexOutOfBoundsException {

                int lc=0;

                String line;

                String code;

                BufferedReader br=new BufferedReader(new FileReader("a.txt"));

                BufferedWriter bw=new BufferedWriter(new FileWriter("b.txt"));

                INSTtable lookup=new INSTtable();

                LinkedHashMap <String,TableRow> SYMTAB;

                SYMTAB=new LinkedHashMap();

                int SymIndex=0;

                //System.out.println(br.readLine());


               while((line=br.readLine())!=null)

                {

                        String parts[]=line.split("\\s+");

                        if(parts[1].equals("START"))

                        {       lc=Integer.parseInt(parts[2]);

                                code="(AD,01)"+",(c,"+lc+")";

                                bw.write(code+"\n")          }
```

```
		if(parts[1].equals("END"))
		{
			code="(AD,02)\t";
			bw.write(code+"\n")
		}

		if(parts[1].equals("DC"))
		{
			lc=lc+1;
			code="(DL,01),(c,"+Integer.parseInt(parts[2])+")";
			//System.out.println(code);
			bw.write(code+"\n");
		}
		if(parts[1].equals("DS"))
		{
			lc=lc+Integer.parseInt(parts[2]);
			code="(DL,02),(c,"+Integer.parseInt(parts[2])+")";
			bw.write(code+"\n");
		}

		if(!parts[0].isEmpty())
		{
			if(SYMTAB.containsKey(parts[0]))

	SYMTAB.put(parts[0],newTableRow(parts[0],lc,SYMTAB.get(parts[0]).getIndex());

			else

	SYMTAB.put(parts[0],newTableRow(parts[0],lc,++SymIndex));
		}
```

```
if(lookup.getType(parts[1].equals("IS")))
{
        code="(IS,0"+lookup.getCode(parts[1])+")\t";
        int j=2;
        String code2=" ";
        while(j<parts.length)
        {
                if(lookup.getType(parts[j].equals("RG")))
                {
                        code2=code2+lookup.getCode(parts[j])+"\t";
                }
                else
                {
                        if(SYMTAB.containsKey(parts[j]))
                        {
                        Int ind=SYMTAB.get(parts[j]).getIndex();
                                code2=code2+"(S,0"+ind+")";
                        }
                        else
                        {
SYMTAB.put(parts[j],new TableRow(parts[j],-1,++SymIndex));
                        int ind=SYMTAB.get(parts[j]).getIndex();
                                code2=code2+"(s,0"+ind+")";
                        }
                }
                j++;
        }       //
        lc++;
        code=code+code2;
        bw.write(code+"\n");
```

```
                }
            }
                br.close();
                bw.close();
            BufferedWriter bws=new BufferedWriter(new FileWriter("SYMTAB.txt"));
                Iterator<String>itr=SYMTAB.keySet().iterator();
                System.out.println("Symbol Table\n");

                while(itr.hasNext())
                {
                    String Key=itr.next().toString();
                    TableRow value=SYMTAB.get(Key);
System.out.println(value.getIndex()+"\t"+value.getSymbol()+"\t"+value.getAddress()+"\n");

        bws.write(value.getIndex()+"\t"+value.getSymbol()+"\t"+value.getAddress()+"\n");
                }
                bws.close();
            }
}
```

INSTtable.java

```
import java.util.*;

public class INSTtable {
        HashMap<String,Integer>AD,IS,DL,RG;
        public INSTtable()
        {
            AD=new HashMap<>();
            IS=new HashMap<>();
            DL=new HashMap<>();
            RG=new HashMap<>();


            DL.put("DC",01);
            DL.put("DS",02);

            IS.put("STOP", 00);
            IS.put("ADD", 01);
```

```
                IS.put("SUB",02);
                IS.put("MUL", 03);
                IS.put("MOVER", 04);
                IS.put("MOVEM", 05);
                IS.put("COMP", 06);
                IS.put("BC", 07);
                IS.put("DIV",8);
                IS.put("READ",9);
                IS.put("PRINT", 10);

                AD.put("START",01);
                AD.put("END", 02);
                AD.put("ORIGIN", 03);
                AD.put("EQU", 04);
                AD.put("LTOREG",05);

                RG.put("AREG", 01);
                RG.put("BREG", 02);
                RG.put("CREG", 03);

}
        public String getType(String s)
        {
                s=s.toUpperCase();
                if(AD.containsKey(s))
                        return "AD";
                else if(IS.containsKey(s))
                        return "IS";
                else if(DL.containsKey(s))
                        return "DL";
                else if(RG.containsKey(s))
                        return "RG";
                else
                        return "  ";

                }
        public int getCode(String s)
        {
                s=s.toUpperCase();
                if(AD.containsKey(s))
                        return AD.get(s);
                else if(IS.containsKey(s))
                        return IS.get(s);
                else if(DL.containsKey(s))
                        return DL.get(s);
                else if(RG.containsKey(s))
                        return RG.get(s);
                else
                        return -1;
        }
```

```
	public boolean getType(boolean equals) {
		// TODO Auto-generated method stub
		return false;
	}

}
```

TableRow.java

```
public class TableRow {


	String symbol;
	int index,address;

	public TableRow(String Symbol,int address)
	{
		this.symbol=Symbol;
		this.address=address;
		index=0;
	}

	public TableRow(String Symbol,int address,int index)
	{
		this.symbol=Symbol;
		this.address=address;
		this.index=index;
	}

	public void SetSymbol(String Symbol)
	{
		this.symbol=Symbol;
	}

	public String getSymbol()
	{
		return symbol;
	}

	public void SetAddress(int address)
	{
		this.address=address;
	}

	public int getAddress()
	{
		return address;
	}

	public void setIndex(int index)
```

```
            {
                        this.index=index;
            }

            public int getIndex()
            {
                        return index;
            }


}
```

a.txt (Input file)

Take/Read input as assembly code.

```
START 200
 ADD AREG M
 SUB BREG P
M DS 2
P DC 10
 END
```

b.txt (output file)

In pass-I of two pass assembler we get Intermediate code.

```
(AD,01),(c,200)
(DL,02),(c,2)
(DL,01),(c,10)
(AD,02)
```

SYMTAB.txt (output file)

```
1          M         202
2          P         203
```

Output:

```
1          M         202

2          P         203
```

Pass II:

```
import java.io.*;

import java.util.*;

public class pass2 {

        static int lc;

        public static void main(String[] args)throws
IOException,FileNotFoundException,ArrayIndexOutOfBoundsException {



BufferedReader br=new BufferedReader(new FileReader("C:\\Exp1_Pass2\\SYMTAB"));


                ArrayList<TableRow>SYMTAB=new ArrayList<>();

                String line;

                while((line=br.readLine())!=null)

                {

                        String parts[]=line.split("\\s+");

                        SYMTAB.add(new
TableRow(parts[1],Integer.parseInt(parts[2]),Integer.parseInt(parts[0])));

System.out.println(parts[1]+" "+Integer.parseInt(parts[2])+" "+Integer.parseInt(parts[0]));


                }br.close();


                br=new BufferedReader(new FileReader("C:\\Exp1_Pass2\\IC"));

BufferedWriter bw=new BufferedWriter(new FileWriter("C:\\Exp1_Pass2\\pass2"));


                String code;

                while((line=br.readLine())!=null)

                {

                String parts[]=line.split("\\s+");

                if(parts[0].contains("(AD,01)"))

                {
```

```
lc=Integer.parseInt(parts[1].replaceAll("[^0-9]" ,""));
//System.out.println(lc);
lc=lc-1;

}
if(parts[0].contains("DL,02"))
{int constant=Integer.parseInt(parts[1].replaceAll("[^0-9]",""));
lc=lc+constant;
}

if(parts[0].contains("AD") ||(parts[0].contains("DL,02")))
{
bw.write("\n");
}
else if(parts.length==1)
{
int opcode=Integer.parseInt(parts[0].replaceAll("[^0-9]" ,""));
code=String.format("%02d",opcode)+"\t0\t"+String.format("%03d",0)+"\n";
bw.write(code);
}
else if(parts.length==2)
{
if(parts[0].contains("DL,01"))
{
int constant=Integer.parseInt(parts[1].replaceAll("[^0-9]",""));
code=String.format("%02d",0)+"\t0\t"+String.format("%03d",constant)+"\n";
bw.write(lc+")"+code);
System.out.println(lc+")"+code);
}
else if(parts[0].contains("IS"))
```

```
                    {
                    int opcode=Integer.parseInt(parts[0].replaceAll("[^0-9]",""));
                    int symindex=Integer.parseInt(parts[1].replaceAll("[^0-9]" ,""));
                    int add=SYMTAB.get(symindex-1).getAddress();

          code=String.format("%02d",opcode)+"\t0\t"+String.format("%03d",add)+"\n";
                    bw.write(lc+")"+code);
                    System.out.println(lc+")"+code);
                    }
                    }
                    else if(parts.length==3)
                    {
                    int opcode=Integer.parseInt(parts[0].replaceAll("[^0-9]",""));

                    int regcode=Integer.parseInt(parts[1]);
                    int symindex=Integer.parseInt(parts[2].replaceAll("[^0-9]",""));
                    int add=SYMTAB.get(symindex-1).getAddress();
code=String.format("%02d",opcode)+"\t"+regcode+"\t"+String.format("%03d",add)+"\n";
                    bw.write(lc+")"+code);
                    System.out.println(lc+")"+code);
                    }
                    lc++;
                    }
                    br.close();
                    bw.close();     }
}
```

Ic.txt:

Input file

```
(AD,01) (c,200)
(IS,01) 1 (s,01)
(IS,02) 2 (s,02)
```

```
(DL,02) (c,2)
(DL,01) (c,4)
(AD,02)
```

Pass2 :

Output File

```
200)01 1        202
201)02 2        204

205)00 0        004
```

SYMTAB:

Input file

```
1 A 202
2 B 204
```

Output:

```
200)01 1        202
201)02 2        204

205)00 0        004
```