EXPERIMENT NO : 1

Name: Fale Manish Dinkar
Class: TE
Div: A
Roll No: T211036
Batch: A2

Problem Statement: Design suitable Data structures and implement Pass-I and Pass-II of a two-pass assembler for pseudo-machine. Implementation should consist of a few instructions from each category and few assembler directives. The output of Pass-I (intermediate code file and symbol table) should be input for Pass-II.

Pass I of Two pass Assembler

```
package Pass1;
class symtab{
	int index;
	String name;
	int addr;


	symtab(int i, String s, int a){
		index = i;
		name = s;
		addr = a;
	}
}

class littab{
	int index;
	String name;
	int addr;

	littab(int i, String s, int a){
		index = i;
		name = s;
		addr = a;
	}
	 void setaddr(int a) {
		 addr = a;
	 }
}
class pooltab{
	int p_index;
	int l_index;

	pooltab(int i, int a){
		p_index = i;
		l_index = a;
	}
```

```
}

public class Pass1 {
	public static void main(String[] args) {
		String input[][] = {{null,"START","100",null},
				{null,"MOVER","AREG","A"},
				{"AGAIN", "ADD", "AREG","='2'"},
				{null,"ADD","AREG","B"},
				{"AGAIN", "ADD", "AREG","='3'"},
				{null,"LTORG",null,null},
				{"AGAIN2", "ADD", "AREG","BREG"},
				{"AGAIN2", "ADD", "AREG","CREG"},
				{"AGAIN", "ADD", "AREG","='2'"},
				{null, "DC","B","3"},
				{"Loop","DS","A","1"},
				{null,"END", null, null}};

		symtab s[] = new symtab[20];
		littab l[] = new littab[20];
		pooltab p[] = new pooltab[20];

		int loc=0, i=0;
		String m, op1,op2;
		int sn = 0, ln=0,lnc=0, pn=0;

		loc = Integer.parseInt(input[0][2]);

		m =input[1][1];
		i = 1;
		while(!m.equals("END")) {
			if(check(m) == 1) {
				if(input[i][0] == null) {
					op1 = input[i][2];
					op2 = input[i][3];
					if(comp(op2,s,sn) == 1) {
						s[sn] = new symtab(sn,op2, 0);
						sn++;
					}
					else if(comp(op2,s,sn) == 2) {
						l[ln] = new littab(ln,op2,0);
						ln++;
					}
					loc++;
					i++;
				}
				else {
					op1 = input[i][0];
					s[sn] = new symtab(sn, op1, loc);
					sn++;

					op1 = input[i][2];
					op2 = input[i][3];
					if(comp(op2,s,sn) == 1) {
						s[sn] = new symtab(sn, op2, 0);
						sn++;
					}
					else if(comp(op2,s,sn) == 2) {
						l[ln] = new littab(ln,op2,0);
						ln++;
					}
					loc++;
					i++;
```

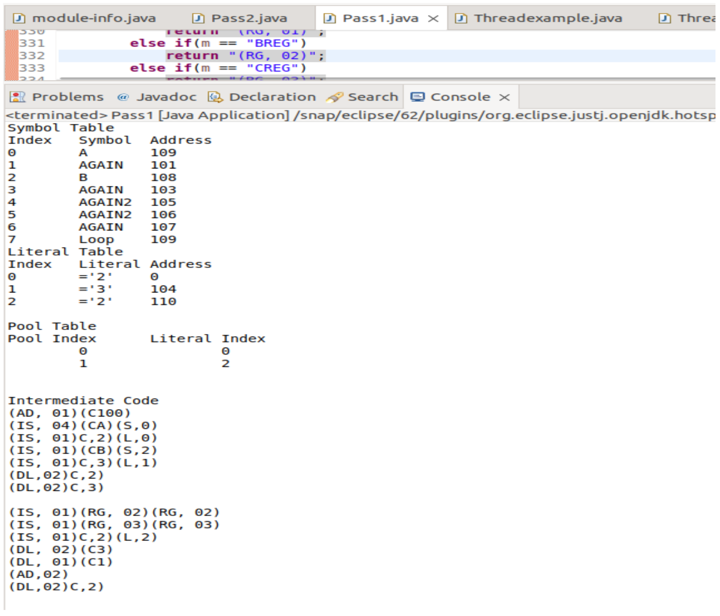
```
                }

            }
        else if(check(m) == 2) {
            if(input[i][0] == null) {
                int temp;
                op1 = input[i][2];
                op2 = input[i][3];
                temp = comps(op1,s,sn);
                if (temp!=99){
                    s[temp] = new symtab(temp,op1, loc);
                }
                loc = loc + Integer.parseInt(op2);
            }
            else {
                int temp;
                op1 = input[i][0];
                s[sn] = new symtab(sn, op1,loc);
                sn++;

                op1 = input[i][2];
                op2 = input[i][3];
                temp = comps(op1,s,sn);
                if (temp!=99){
                    s[temp] = new symtab(temp,op1, loc);
                }
                loc = loc + Integer.parseInt(op2);
                i++;
            }
        }else if(check(m) == 3) {
            if(input[i][0] == null) {
                int temp;
                op1 = input[i][2];
                op2 = input[i][3];
                temp = comps(op1,s,sn);
                if (temp!=99){
                    s[temp] = new symtab(temp,op1, loc);
                }
                loc++;
                i++;
        }
            else {
                int temp;
                op1 = input[i][0];
                s[sn] = new symtab(sn, op1, loc);
                sn++;

                op1 = input[i][2];
                op2 = input[i][3];
                temp = comps(op1,s,sn);
                if (temp!=99){
                    s[temp] = new symtab(temp,op1, loc);
                }
                loc++;
                i++;
            }
        }
        else if(check(m) == 4) {
            if(lnc != ln) {
                p[pn] = new pooltab(pn,lnc);
                lnc++;
                pn++;
```

```
				}
				while(lnc != ln) {
					l[lnc].setaddr(loc);
					lnc++;
					loc++;
				}
				i++;
			}
			m = input[i][1];
		}
		if(lnc != ln) {
			p[pn] = new pooltab(pn,lnc);
			pn++;

		}
		while(lnc != ln) {
			l[lnc].setaddr(loc);
			lnc++;
			loc++;
		}
		System.out.print("Symbol Table\nIndex\tSymbol\tAddress\n");
		for(i=0;i<sn;i++) {
			System.out.println(s[i].index + "\t" + s[i].name +"\t" + s[i].addr);
		}
		System.out.print("Literal Table\nIndex\tLiteral\tAddress\n");
		for(i=0; i<ln; i++) {
			System.out.println(l[i].index + "\t" + l[i].name +"\t" + l[i].addr);
		}
		System.out.print("\nPool Table\nPool Index\tLiteral Index\n");
		for(i=0; i<pn; i++) {
			System.out.println("\t" + p[i].p_index+"\t\t" + p[i].l_index);
		}
		System.out.print("\n\nIntermediate Code\n");
		i = 0;
		m = input[i][1];
		op1 = input[i][2];
		op2 = input[i][2];
		int point = 0,in1,in2,j=0;
		System.out.print(ic(m)+ ic(op1));
		while(!m.equals("END")) {
			if(check(m) == 1) {
				System.out.print((ic(m)+ ic(op1)));
				if(comp(op2,s,sn)==0 && comps(op2,s,sn) == 99) {
					System.out.print(ic(op2));

				}
				else if(comp(op2,s,sn)==2) {
					int temp;
					temp = compl(op2,l,ln,j);
					System.out.print("(L,"+temp+")");
					j++;
				}
				else if(comp(op2,s,sn) != 1) {
					int temp;
					temp = comps(op2,s,sn);
					System.out.print("(S,"+temp+")");

				}
			}else if(check(m) == 2 || check(m) == 3) {
				System.out.print(ic(m)+ic(op2));

			}else if(check(m) == 4) {
```

```
                    if(point + 1 != pn) {
                        in1 = p[point+1].l_index-p[point].l_index;
                        in2 = p[point].l_index;
                        point++;
                        while(in1 >0) {
                            System.out.print(ic(m) + ic(l[in2].name));
                            in2++ ;
                            in1--;
                            System.out.print("\n");
                        }
                    }else {
                        in2 = p[point].l_index;
                        while(in2 != ln) {
                            System.out.print("\n");
                        }
                    }
                }
                i++;
                m = input[i][1];
                op1 = input       [i][3];
                op2 = input[i][3];
                System.out.print("\n");
            }
            System.out.println(ic(m));
            m = "LTORG";
            if(point + 1 != pn) {
                in1 = p[point+1].l_index-p[point].l_index;
                in2 = p[point].l_index;
                point++;
                while(in1 >0) {
                    System.out.print(ic(m)+ic(l[in2].name));
                    in2++;
                    in1--;
                }
            }else {
                in2 = p[point].l_index;
                while(in2 != ln) {
                    System.out.print(ic(m) +ic(l[in2].name));
                    in2++;
                }
            }
        }

        static int check(String m) {
            if(m.equals("MOVER") || m.equals("ADD")){
                return 1;
            }
            else if(m.equals("DS")){
                return 2;
            }
            else if(m.equals("DC")){
                return 3;
            }
            else if(m.equals("LTORG")){
                return 4;
            }
            return -1;
        }
        static int comp(String m,symtab s[], int sn) {
            if(m.equals("AREG")|| m.equals("BREG") || m.equals("CREG"))
                return 0;
            else if(m.toCharArray()[0] == '=')
```

```
                        return 2;
                else if(comps(m,s,sn) == 99)
                        return 1;
                else
                        return 0;

        }
        static int compl(String m, littab l[], int ln, int j) {
                int i;
                for(i=j;i<ln;i++) {
                        if(m.equals(l[i].name))
                                return l[i].index;
                }
                return 99;
        }
        static int comps(String m,symtab s[], int sn) {
                int i;
                for(i =0; i<sn; i++) {
                        if(m.equals(s[i].name))
                                return s[i].index;
                }
                return 99;

        }
        static String ic(String m) {
                if(m == "START")
                        return"(AD, 01)";
                else if(m == "END")
                        return "(AD,02)";
                else if(m == "ORIGIN")
                        return "(AD, 03)";
                else if(m == "LTORG")
                        return "(DL,02)";
                else if(m == "ADD")
                        return "(IS, 01)";
                else if(m == "SUB")
                        return "(IS, 02)";
                else if(m == "MOVER")
                        return "(IS, 04)";
                else if(m == "MOVEM")
                        return "(IS, 05)";
                else if(m == "AREG")
                        return "(RG, 01)";
                else if(m == "BREG")
                        return "(RG, 02)";
                else if(m == "CREG")
                        return "(RG, 03)";
                else if(m == "DS")
                        return "(DL, 01)";
                else if(m == "DC")
                        return "(DL, 02)";
                else if(m.toCharArray()[0] == '=')
                        return ("C,"+ m.toCharArray()[2]+")");
                else {
                        return("(C"+m+")");
                }
        }
}
```

Output:



```
        else if(m == "BREG")
            return "(RG, 02)";
        else if(m == "CREG")
```

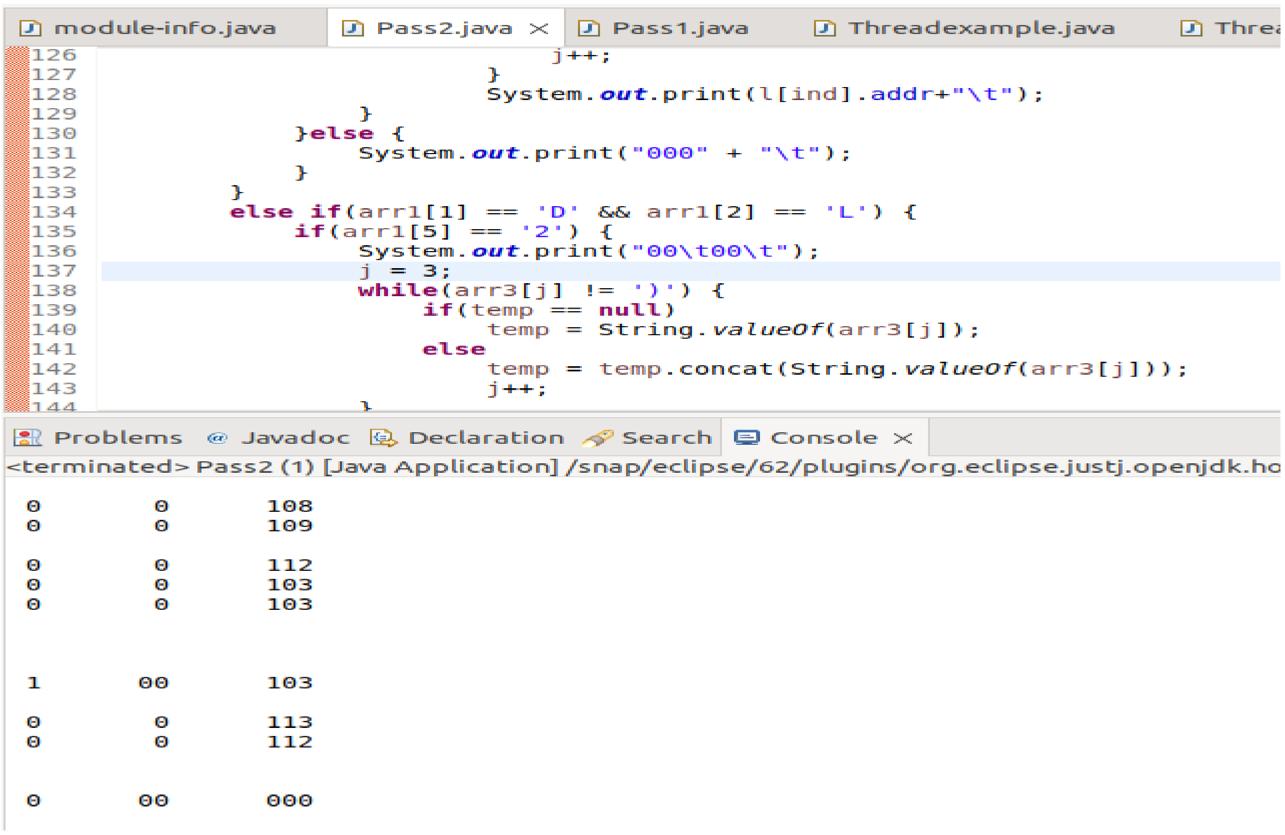
Problems Javadoc Declaration Search Console

```
<terminated> Pass1 [Java Application] /snap/eclipse/62/plugins/org.eclipse.justj.openjdk.hotsp
Symbol Table
Index    Symbol   Address
0        A        109
1        AGAIN    101
2        B        108
3        AGAIN    103
4        AGAIN2   105
5        AGAIN2   106
6        AGAIN    107
7        Loop     109
Literal Table
Index    Literal  Address
0        ='2'     0
1        ='3'     104
2        ='2'     110

Pool Table
Pool Index        Literal Index
         0                0
         1                2


Intermediate Code
(AD, 01)(C100)
(IS, 04)(CA)(S,0)
(IS, 01)C,2)(L,0)
(IS, 01)(CB)(S,2)
(IS, 01)C,3)(L,1)
(DL,02)C,2)
(DL,02)C,3)

(IS, 01)(RG, 02)(RG, 02)
(IS, 01)(RG, 03)(RG, 03)
(IS, 01)C,2)(L,2)
(DL, 02)(C3)
(DL, 01)(C1)
(AD,02)
(DL,02)C,2)
```

Pass II of Two pass Assembler

```
package Pass2;
import java.text.DecimalFormat;

class symtab{
        int index;
        String name;
        int addr;


        symtab(int i, String s, int a){
                index = i;
                name = s;
                addr = a;
        }
}

class littab{
        int index;
        String name;
        int addr;

        littab(int i, String s, int a){
                index = i;
                name = s;
                addr = a;
        }
         void setaddr(int a) {
                 addr = a;
         }
}

public class Pass2 {
        public static void main(String[] args) {
                String ic[][] = {{"(AD, 01)", null, "(c,100)"},
                                {"(IS, 04)","(RG, 01)","(L,0)"},
                                {"(IS, 01)","(RG, 03)","(L,1)"},
                                {"(DL, 01)",null ,"(C,3)"},
                                {"(IS, 04)","(RG, 01)","(S,2)"},
                                {"(IS, 01)","(RG, 01)","(S,3)"},
                                {"(IS, 05)","(RG, 01)","(S,4)"},
                                {"(DL, 02)",null ,"(C,5)"},
                                {"(DL, 02)",null ,"(C,1)"},
                                {"(AD, 04)",null ,"(C,103)"},
                                {"(IS, 10)",null,"(S,4)"},
                                {"(AD, 03)",null ,"(C,101)"},
                                {"(IS, 02)","(RG, 01)","(L,2)"},
                                {"(IS, 03)","(RG, 03)","(S,2)"},
                                {"(DL, 02)",null ,"(C,5)"},
                                {"(AD, 03)",null ,"(C,111)"},
                                {"(IS, 00)",null,null},
                                {"(DL, 02)",null ,"(C,19)"},
                                {"(AD, 02)",null,null},
                                {"(DL, 02)",null ,"(C,1)"}};

                symtab s[] = new symtab[20];
                littab l[] = new littab[20];
```

```
s[0] = new symtab(0, "A", 102);
s[1] = new symtab(1, "L1", 105);
s[2] = new symtab(2, "B", 112);
s[3] = new symtab(3, "C", 103);
s[4] = new symtab(4, "D", 103);

l[0] = new littab(0,"='5'",108);
l[1] = new littab(1,"='1'",109);
l[2] = new littab(2,"='1'",113);

int i=0, j=0, ind=0;
String m, op1,op2,temp;
char arr1[],arr2[],arr3[];

DecimalFormat df = new DecimalFormat("000");

while(i < ic.length) {
        temp = null;
        arr1 = null;
        arr2 = null;
        arr3 = null;

        m = ic[i][0];
        op1 = ic[i][1];
        op2 = ic[i][2];

        arr1 = m.toCharArray();


        if(op1 != null) {
                arr2  = op1.toCharArray();
        }

        if(op2 != null) {
                arr3  = op2.toCharArray();
        }

        if(arr1[1] == 'I' && arr1[2] == 'S') {
                System.out.print(arr1[4]+""+arr1[5]+"\t");
                if(op1 != null) {
                        System.out.print(arr2[4]+""+arr2[5]+"\t");
                }
                else {
                        System.out.print("00"+"\t");
                }

                if(op2 != null) {
                        if(arr3[1] == 'R' && arr3[2] == 'G') {
                                System.out.print(arr3[4]+arr3[5]+"\t");
                        }
                        else if(arr3[1] == 'S') {
                                ind = Character.getNumericValue(arr3[3]);
                                j = 4;
                                while(arr3[j] != ')') {
                                        ind = ind*10;
                                        ind = ind + (Character.getNumericValue(arr3[j]));
                                        j++;
                                }
                                System.out.print(s[ind].addr+"\t");
                        }
                        else if(arr3[1] == 'L') {
                                ind = (Character.getNumericValue(arr3[3]));
                                j = 4;
                                while(arr3[j] != ')') {
                                        ind = ind*10;
                                        ind = ind + (Character.getNumericValue(arr3[j]));
                                        j++;
                                }
                                System.out.print(l[ind].addr+"\t");
                        }
                }else {
                        System.out.print("000" + "\t");
                }
        }
        else if(arr1[1] == 'D' && arr1[2] == 'L') {
                if(arr1[5] == '2') {
                        System.out.print("00\t00\t");
                        j = 3;
                        while(arr3[j] != ')') {
                                if(temp == null)
                                        temp = String.valueOf(arr3[j]);
                                else
                                        temp = temp.concat(String.valueOf(arr3[j]));
                                        j++;
                        }

        System.out.print(df.format(Integer.parseInt(temp)));
                }
        }
        i++;
        System.out.print("\n");
        }
    }
}
```



module-info.java | Pass2.java | Pass1.java | Threadexample.java | Threa

```
                                   j++;
                               }
                               System.out.print(l[ind].addr+"\t");
                       }
                   }else {
                       System.out.print("000" + "\t");
                   }
               }
               else if(arr1[1] == 'D' && arr1[2] == 'L') {
                   if(arr1[5] == '2') {
                       System.out.print("00\t00\t");
                       j = 3;
                       while(arr3[j] != ')') {
                           if(temp == null)
                               temp = String.valueOf(arr3[j]);
                           else
                               temp = temp.concat(String.valueOf(arr3[j]));
                               j++;
                       }
```

Problems | Javadoc | Declaration | Search | Console

```
<terminated> Pass2 (1) [Java Application] /snap/eclipse/62/plugins/org.eclipse.justj.openjdk.ho

 0         0       108
 0         0       109

 0         0       112
 0         0       103
 0         0       103



 1        00       103

 0         0       113
 0         0       112


 0        00       000
```

Output: