EXPERIMENT NO 6

Date of performance :
Date of submission :

Aim: To generate target code for the code optimized, considering the target machine to be X86.

Software Used: c language

Theory:
Simple code generation
In this sectin we will discuss the method of generating targe code from address statement. In this method computed result can be kept in register as long as possible eg.
X:=a+b;
Correspounding target code is " add b,r: here r hold value of a here cost=2
Or
Mov b,r, here r holds value
Add r,r0 here cost=2
The code generator algo. Used descriptor to keep track of register contents & address for names.

1. A register descriptor is used to keep track of what is currently in each register descriptor show initially all the register are empty as the code generation for block program the register will hold value of computations.
2. the add descriptor stores the location where current value of name can be found at runtime. The information about location can be stored in symbol table & is use to access the variables

the modes of operand address reusability
s->used to indicate value of operand in storage
r->used to indicate valued of operand in register.
Is->indicates that address of operand is stored in storage
Ir->indicate that address of operand is stored in register
Eg address descriptor
The address descriptor fields
The attributes mean type of operand:-it generally refer to the name of temporary variables.
The address mode indicate whether address is in storage location /in register.
The register descriptor can be shown as

By using register descriptor we can keep track of registers which are currently occupied. The status field is of Boolean type which is used to check whether the register is occupied with some data/not when status field holds the value 'true' then operand descriptor who is having the lasts value in register.

Algorithm:

```
 Gen_code(operator,Op1,op2)
{
If(op1.addressmode='R')
{
If(operator='+')
Generate('ADD op2, R0');
Else if(operator='-')
Generate('SUB op2, R0');
Else if(operator='*')
Generate('MUL op2, R0');
Else if(operator='/')
Generate('DIV op2, R0');
}
Else if (op2.addressmode='R')
{
If(operator='+')
Generate('ADD op1, R0');
Else if(operator='-')
Generate('SUB op1, R0');
Else if(operator='*')
Generate('MUL op1, R0');
Else if(operator='/')
Generate('DIV op1, R0');
}
Else{
Generate('MOV op2, R0')
If(operator='+')
Generate('ADD op1, R0');
Else if(operator='-')
Generate('SUB op1, R0');
Else if(operator='*')
Generate('MUL op1, R0');
Else if(operator='/')
Generate('DIV op1, R0');
}
}
```

Eg:=
X:=(a+b)*(c-d)+((e/f)*(a+b))

Program:

```
import java.io.*;
public class exp6
{
   public static void main(String args[])throws IOException
          {
          DataInputStream in=new DataInputStream(System.in);
          System.out.println("Enter the equation");
          String stmt=in.readLine();
          StringBuffer ans=new StringBuffer("");
          int reg=0;
          int count=0;
          char c2='a';
          int flag=0;
          for(int i=0;i<stmt.length();i++)
          {
                    char c=stmt.charAt(i);
                    if(i>0)
                    {
                    c2=stmt.charAt(i-1);
                    }
                    switch(c)
                    {
                    case'(':count++;
                    break;
                    case')':count--;
                    break;
                    case'+':if(count>0)
                    {
                    System.out.println("MOV "+stmt.charAt(i-1)+",R"+reg);
                    System.out.println("ADD "+stmt.charAt(i+1)+",R"+reg);
                    ans.append("R"+reg);
                    reg++;
                    }
                    else
                    {
                    ans.append("+");
                    }
                    break;
                    case'-':if(count>0)
                    {
                    System.out.println("MOV "+stmt.charAt(i-1)+",R"+reg);
                    System.out.println("SUB "+stmt.charAt(i+1)+",R"+reg);
                    ans.append("R"+reg);
                    reg++;
                    }
                    else
                    {
                    ans.append("-");
                    }
                    break;
                    case'*':if(count>0)
                    {
                    System.out.println("MOV "+stmt.charAt(i-1)+",R"+reg);
                    System.out.println("MUL "+stmt.charAt(i+1)+",R"+reg);
                    ans.append("R"+reg);
                    reg++;
                    }
                    else
                    {
                    ans.append("*");
                    }
                    break;
                    case'/':if(count>0)
                    {
                    System.out.println("MOV "+stmt.charAt(i-1)+",R"+reg);
                    System.out.println("DIV "+stmt.charAt(i+1)+",R"+reg);
                    ans.append("R"+reg);
                    reg++;
                    }
                    else
                    {
                    ans.append("/");

                    }
                    break;
                    default:break;
                    }
                    flag++;
                    }
                    String ans1=new String(ans);
                    for(int i=0;i<ans1.length();i++)
                    {
                    char c=ans1.charAt(i);
                    switch(c)
```

```
                {
                case'+':System.out.println("ADD"+ans1.charAt(i-2)+ans1.charAt(i-
        1)+","+ans1.charAt(i+1)+ans1.charAt(i+2));
                          break;
                case'-':System.out.println("SUB"+ans1.charAt(i-2)+ans1.charAt(i-
        1)+","+ans1.charAt(i+1)+ans1.charAt(i+2));
                          break;
                case'*':System.out.println("MUL"+ans1.charAt(i-2)+ans1.charAt(i-
        1)+","+ans1.charAt(i+1)+ans1.charAt(i+2));
                          break;
                case'/':System.out.println("DIV"+ans1.charAt(i-2)+ans1.charAt(i-
        1)+","+ans1.charAt(i+1)+ans1.charAt(i+2));
                          break;
                   default :break;
}
}
}
}
}
```

OUTPUT-

```
Enter the equation
(a+b)*(c-d)+(e/f)*(a+b)
MOV a,R0
ADD b,R0
MOV c,R1
SUB d,R1
MOV e,R2
DIV f,R2
MOV a,R3
ADD b,R3
MUL R0,R1
ADD R1,R2
MUL R2,R3
```

Conclusion: Hence we have generated target code for code optimized considering target m/c to be x86.

SIGN AND REMARK

DATE

R1	**R2**	**R3**	**R4**	**R5**	**Total** **(15 Marks)**	**Signature**