EXPERIMENT- 11

INTERMEDIATE CODE GENERATION - QUADRUPLE, TRIPLE, INDIRECT TRIPLE

AIM: A program to implement intermediate code generation

ALGORITHM: Takes 3 address as input. a := b op c perform various actions.

1. Invoke a function getreg to find out location L where result of computation b op c should be stored.
2. Consult address description for y to determine y'. If the value of y currently in memory and register both the prefer the register y'. If value of y is not already in L then generate instruction MOV y', L to place copy of y in L.
3. Generate instruction OP z', L where z' is used to show current location of z. If z is in both then prefer a register to a memory location. Update address descriptor of x to indicate that x is in location L. If x is in L then update its descriptor & remove x from all other descriptors.
4. If current value of y or z have no next uses or not live on exit from block or in register then alter the register descriptor to indicate that after execution of x := y op z those register no longer contain y or z.

PROGRAM:

```
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <string.h>
```

Output: Input the expression:

Intermediate code generation

$a = b^* - c + b^* b - c$

prefix : $- + - * = * \bullet bbc$

postfix : $a = b^* c - bb^* + c -$

$t_1 := = * b$

$t_2 := t_1 - c$

$t_3 := b^* b$

$t_4 := t_2 + t_3$

$t_5 = t_4 - c$

the Quadruple for expression

OP	Arg1	Arg2	Result
*	=	b	t(1)
-	c	(-)	t(2)
+	t(1)	t(2)	t(3)
*	b	b	t(4)
+	t(3)	t(4)	t(5)
-	c	(-1)	t(6)
=	t(5)	t(6)	t(7)

The triple for given expression :

OP	Arg1	Arg2
*	=	b
-	c	(-1)
+	(0)	(1)
*	b	b'
+	(2)	(3)
-	c	
+	(4)	(-1)
		(5)

```
void done (int i);
int p[5] = {0,1,2,3,4}, c=x, i, k, l, m, pi;
char sw[5] = {'=','-','+','/','*'}, j[20], a[5], b[5];
void main()
{ printf("Enter the expression : ");
 scanf("%s", j);
printf("The Intermidiate code is:\n");
small(); }
void done (int i) {
a[0] = b[0] = '\0';
if(!isdigit(j[i+2]) && !isdigit(j[i-2]))
{ a[0] = j[i-1];
   b[0] = j[i+1]; }
if(isdigit(j[i+2]))
{ a[0] = j[i-1];
   b[0] = 't';
   b[1] = j[i+2]; }
if(isdigit(j[i-2])) {
b[0] = j[i+1];
a[0] = 't';
a[1] = j[i-2];
b[1] = '\0'; }
if(isdigit(j[i+2]) && isdigit(j[i-2]))
{ a[0] = 't';
   b[0] = 't';
   a[1] = j[i-2];
   b[1] = j[i+2];
   sprintf(ch, "%d", c);
   j[i+2] = j[i-2] = ch[0]; }
if(j[i] == '*') printf("t%d = %s * %s\n", c, a, b);
if(j[i] == '/') printf("t%d = %s / %s\n", c, a, b);
if(j[i] == '+') printf("t%d = %s + %s\n", c, a, b);
if(j[i] == '-')
```

```
printf("%d.c = t%d.d", j[i-1], --c);
sprintf(ch, "%d.d", c);
j[i] = ch[0];
c++;
small(); }
void small()
{ pi = 0; l=0;
 for (i=0; i < strlen(j); i++)
{ for (m=0; m<5; m++)
  if (j[i] == sw[m])
  if (pi <= p[m])
{ pi = p[m];
   l=i;
   k=i;
}}
if (k==1)
done(k);
 else
exit(0); }
```

RESULT: : A program to implement intermediate code generation - Quadruple, triple, Indirect triple has been compiled and run successfully.