NUMERICAL PROBLEMS

Prob.1. Translate the expression a * – (b + c) into –

(i) Postfix notation

(ii) Three address code.

Sol. (i) Postfix Notation –

a * – (b + c)

a * – (bc+)

a *(bc + –)

abc + - * Ans

(ii) Three Address Code –

a * – (b + c)

$T_1 = b + c$

$T_2 = - T_1$

$T_3 = a * T_2$ Ans.

Prob.2. Construct syntax tree and postfix notation for the following expression –

$$(a + b\ (b * c))\ \uparrow d\ - e \mid (f + g)$$

(R.G.P.V., Dec. 2008)

Sol. The syntax tree for the given expression is shown in fig. 3.26.

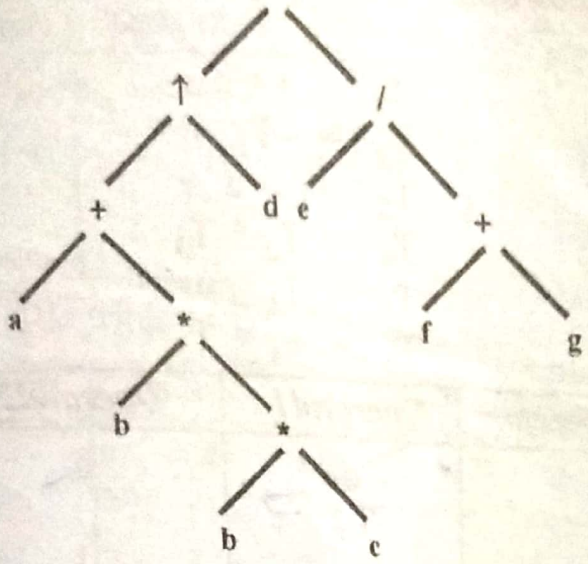

Fig. 3.26 Syntax Tree

Postfix Notation – The given expression is –

$$(a + b*(b*c)) \uparrow d - e/(f + g)$$

$$= (a + b*(bc*)) \uparrow d - e/(f + g)$$

Let $T_1 = bc*$

$$= (a + b\,(T_1)) \uparrow d - e/(f + g)$$

$$= (a + bT_1*) \uparrow d - e/(f + g)$$

Let $T_2 = bT_1*$

$$= (a + T_2) \uparrow d - e/(f + g)$$

$$= (aT_2+) \uparrow d - e/(fg +)$$

Let $T_3 = aT_2+$, and

$T_4 = fg+$

$$= T_3 \uparrow d - e/T_4$$

$$= T_3d \uparrow - e/T_4$$

$$= T_3d \uparrow - eT_4/$$

$$= T_3d \uparrow eT_4/-$$

Now, put the value of T_4, T_3, T_2, T_1 respctively

$$= T_3d \uparrow e\ fg\ +/-$$

$$= aT_2 + d \uparrow efg + /-$$

$$= abT_1* + d\uparrow efg\ /-$$

$$= abbc** + d\uparrow efg\ +/-$$

Prob.3. Translate the expression – (a + b) * (c + d) + (a + b + c) into –
(i) Quadruples (ii) Triples (iii) Indirect triples.
(R.G.P.V., June 2003, 2006)

Sol. (i) Quadruple – The three address code of the given expression is as follows –

$T_1 := a + b$
$T_2 := -T_1$
$T_3 := c + d$
$T_4 := T_2 * T_3$
$T_5 := T_1 + c$
$T_6 := T_4 + T_5$

	Operator	*Operand1*	*Operand2*	*Result*
(1)	+	a	b	T_1
(2)	–	T_1		T_2
(3)	+	c	d	T_3
(4)	*	T_2	T_3	T_4
(5)	+	T_1	c	T_5
(6)	+	T_4	T_5	T_6

(ii) Triples – The triple of the given expression is as follows –

	Operator	*Operand 1*	*Operand2*
(1)	+	a	b
(2)	–	(1)	
(3)	+	c	d
(4)	*	(2)	(3)
(5)	+	(1)	c
(6)	+	(4)	(5)

(iii) Indirect Triple – The indirect triple of the given expression is as follows –

	STATEMET
(1)	(1)
(2)	(2)
(3)	(3)
(4)	(4)
(5)	(5)
(6)	(6)

	Operator	*Operand1*	*Operand2*
(1)	+	a	b
(2)	–	(1)	
(3)	+	c	d
(4)	*	(2)	(3)
(5)	+	(1)	c
(6)	+	(4)	(5)

Prob.4. Translate the expression y : = (4 * 7 + 1) *2 into –
(i) Quadruple, (ii) Triple, (iii) Indirect triple.

Sol. (i) Quadruple – The three address code of the expression y : = (4*7 + 1) * 2 is as follows :

$T_1 = 4 * 7$
$T_2 := T_1 + 1$
$T_3 := T_2 * 2$
$y := T_3$

The quadruple of the given expression is shown below –

	Operator	Operand1	Operand2	Result
(1)	*	4	7	T_1
(2)	+	T_1	1	T_2
(3)	*	T_2	2	T_3
(4)	=	T_3		Y

(ii) Triple – The triple of the given expression is shown below –

	Operator	Operand1	Operand2
(1)	*	4	7
(2)	+	(1)	1
(3)	*	(2)	2
(4)	=	y	(3)

(iii) Indirect Triple – Indirect triple of the given expression is shown below –

		Operator	Operand 1	Operand2
(1)	(1)	*	4	7
(2)	(2)	+	(1)	1
(3)	(3)	*	(2)	2
(4)	(4)	=	y	(3)

Prob.5. Write quadruples, triples and indirect triples for the expression

$$-(a + b) * (c + d) - (a + b + c)$$

(R.G.P.V., June 2004, 2005. 2007, 2008)

Sol. (i) Quadruples – The three address code of the given expression is as follows –

$T_1 := a + b$

$T_2 := -T_1$

$T_3 := c + d$

$T_4 := T_2 * T_3$

$T_5 := T_1 + c$

$T_6 := T_4 - T_5$

The quadruples of the given expression is shown below –

	Operator	Operand1	Operand2	Result
(1)	+	a	b	T_1
(2)	–	T_1		T_2
(3)	+	c	d	T_3
(4)	*	T_2	T_3	T_4
(5)	+	T_1	c	T_5
(6)	–	T_4	T_5	T_6

(ii) Triple – The triple of the given expression is shown below –

	Operator	Operand1	Operand2
(1)	+	a	b
(2)	–	(1)	
(3)	+	c	d
(4)	*	(2)	(3)
(5)	+	(1)	c
(6)	–	(4)	(5)

(iii) Indirect Triple – The indirect triple of the given expression is shown below –

(1)	(i)
(2)	(2)
(3)	(3)
(4)	(4)
(5)	(5)
(6)	(6)

	Operator	Operand1	Operand2
(1)	+	a	b
(2)	–	(1)	
(3)	+	c	d
(4)	*	(2)	(3)
(5)	+	(1)	c
(6)	–	(4)	(5)

Prob.6. Translate the following expression to quadruple, triple and indirect triple –

$$-(x + y)*(z + c) - (x + y + z)$$ ***(R.G.P.V., June 2009)***

Sol. **Quadruple** – The three address code of the given expression is as follows –

$$T_1 = x + y$$

$$T_2 = -T_1$$

$$T_3 = z + c$$

$$T_4 = T_2 * T_3$$

$$T_5 = T_1 + z$$

$$T_6 = T_4 - T_5$$

The quadruples of the given expression is shown below –

S. No.	Operator	Operand 1	Operand 2	Result
(1)	+	x	y	T_1
(2)	–	T_1		T_2
(3)	+	z	c	T_3
(4)	*	T_2	T_3	T_4
(5)	+	T_1	z	T_5
(6)	–	T_4	T_5	T_6

(ii) Triple – The triple of the given expression is shown below –

S. No.	Operator	Operand 1	Operand 2
(1)	+	x	y
(2)	–	(1)	
(3)	+	z	c
(4)	*	(2)	(3)
(5)	+	(1)	z
(6)	–	(4)	(5)

(iii) Indirect Triple – The indirect triple of the given expression is shown below –

	Statement
(1)	(1)
(2)	(2)
(3)	(3)
(4)	(4)
(5)	(5)
(6)	(6)

S.No.	Operator	Operand 1	Operand 2
(1)	+	x	y
(2)	–	(1)	
(3)	+	z	c
(4)	*	(2)	(3)
(5)	+	(1)	z
(6)	–	(4)	(5)

Prob. 7. Construct 3-Address Code for the following –

If [(a < b) and ((c ≥ d) or (a > d))], then

$z = x + y * z$

else

$z = z + 1$

(R.G.P.V., Dec. 2008)

Sol. The three address code for the following program is as follows –

(1) if a < b goto (3)
(2) goto (11)
(3) if c ≥ d goto (7)
(4) goto (5)
(5) if a > d goto (7)
(6) goto (11)

(7) $t_1 = y * z$
(8) $t_2 = x + t_1$
(9) $z = t_2$
(10) goto (13)
(11) $t_3 = z + 1$
(12) $z = t_3$
(13) Exit;

Prob.8. Generate the three address code for the following statements

```
While a < b do
    if c < d then
    x := y + z
    else
    x := y – z
```

Sol. The three address code is –

(1) if a < b goto (3)
(2) goto (11)
(3) if c < d goto (5)
(4) goto (8)
(5) $t_1 = y + z$
(6) $x = t_1$
(7) goto (1)
(8) $t_2 = y - z$
(9) $x = t_2$
(10) goto (1)
(11) Exit;

Prob.9. Generate the three address code for the following C program

```
main ( )
{
    int i := 1 ;
    int a [10];
    while (i <= 10)
    a[i] = i
}
```

Sol. The three address code for the above C program is –

(1) i := 1
(2) if i <= 10 goto (4)
(3) goto (8)
(4) t_1 := i * width

(5) $t_2 := addr\ (a) - width$
(6) $t_2\ [t_1] = q$
(7) goto (2)
(8) Exit,

Where, width is the number of bytes requried for each element.

Prob.10. Consider the following while-statement –

while A > B and A <= 2*B – 5 do
A : A + B;

(i) Construct the parse tree for the given above while-statement.
(ii) Write the intermediate code for while-statement.

(R.G.P.V., Dec. 2007)

Sol. (i) Parse Tree – The parse tree for this statement is shown in fig. 3.27. We use "exp" for expression, "relop" for relational operator and we indicate parenthetically the particular name or constant to which each instance of token **id** and **const** refer.

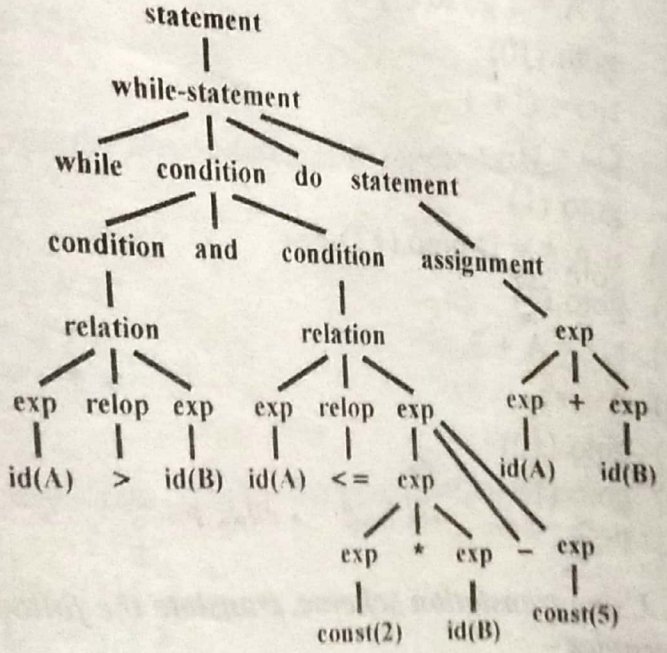

Fig. 3.27 Parse Tree for while-statement

(ii) Intermediate Code –

L_1 : if A > B goto L_2
goto L_3
L_2 : $T_1 := 2 * B$
$T_2 := T_1 - 5$
if A ≤ T_2 goto L_4
goto L_3
L_4 : A := A + B
goto L_1
L_3 : END

Fig. 3.28 Intermediate Code for while-statement

Prob.11. Generate the three address code for the following program fragment :

```
while (A < C and B > D) do
    if A = 1 then
        C := C + 1
    else
        while A <= D do
        A := A + 3
```

(R.G.P.V., Dec. 2006)

Sol. The three address code for the given program fragment is as follows–

(1) if A < C goto (3)
(2) goto (16)
(3) if B > D goto (5)
(4) goto (16)
(5) if A = 1 go to (7)
(6) goto (10)
(7) t_1 : = C + 1
(8) C : = t1
(9) goto (1)
(10) if A < = D goto (12)
(11) goto (1)
(12) t_2 : = A + 3.
(13) A : = t_2
(14) goto (10)
(15) goto (1)
(16) Exit;

Prob.12. Using translation scheme, translate the following segment of codes to quadruples –

```
while (A < C and B < D)do
if A = 1 then C : = C + 1
else while A ≤ D do A : = A + 2
```

(R.G.P.V., June 2007)

Sol. Same as Prob.11.

Prob.13. Consider the program fragment –

```
Sum : = 0
for (i = 1; i < = 20; i ++)
sum : = sum + a[i] + b[i];
```

and generate the three address code for it, there are four bytes per word.

Sol. The three address code is –

(1) Sum : = 0
(2) i : = 1
(3) if i <= 20 goto (8)
(4) goto (18)
(5) t_1 : = i + 1
(6) i : = t_1
(7) goto (3)
(8) t_2 : = i * 4
(9) t_3 : = addr (a) – 4
(10) t_4 : = $t_3[t_2]$
(11) t_5 = i * 4
(12) t_6 : = addr (b) – 4
(13) t_7 : = t_6 $[t_5]$
(14) t_8 : = sum + t_4
(15) t_8 : = t_8 + t_7
(16) sum : = t_8
(17) goto (5)
(18) Exit;

Prob.14. Generate the three address code for the following switch statement –

```
Switch (i + j)
{
case 1 : x : = y + z
case 2 : p : = q + r
case 3 : u: = v + w
}
```

Sol. The three address code for the given switch statements is as follows –

(1) t_1 : = i + j
(2) goto (11)
(3) t_1 : = y + z
(4) x : = t_1
(5) goto (6)
(6) t_2 : = q + r
(7) p : = t_2
(8) goto (9)
(9) t_3 : = u + w
(10) u : = t_3
(11) goto (15)
(12) if t_1 = 1 goto (3)
(13) if t_1 = 2 goto (6)
(14) if t_1 = 3 goto (9)
(15) Exit;

Prob.15. Generate the three address code for the following switch statement –

```
Switch (a + b)
    {
    case 2 :
        {
        x : = y;
        break;
        }
    case 5 :
        {
        switch (x)
            {
            case 0 :
                {
                a : = b + 1;
                break;
                }
            case 1 :
                {
                a : = b + 3'
                break;
                }
            default :
                {
                a : = 2;
                break ;
                }
            }
        break;
        }
    case 9 :
        {
        x : = y – 1;
        break;
        }
    default :
        {
        a : = 2;
        break;
        }
```

Sol. The three address code for the given program is as follows –

(1) $t_1 := a + b$
(2) goto (23)
(3) $x := y$
(4) goto NEXT
(5) goto (14)
(6) $t_3 := b + 1$
(7) $a := t_3$
(8) goto NEXT
(9) $t_4 := b + 3$
(10) $a := t_4$
(11) goto NEXT
(12) $a := 2$
(13) goto NEXT
(14) if $x = 0$ goto (6)
(15) if $x = 1$ goto (9)
(16) goto (12)
(17) goto NEXT
(18) $t_5 := y - 1$
(19) $x := t_5$
(20) goto NEXT
(21) $a := 2$
(22) goto NEXT
(23) if $t_1 = 2$ goto (3)
(24) if $t_1 = 5$ goto (5)
(25) if $t_1 = 9$ goto (18)
(26) goto (21)

Prob.16. ***To evaluate the following instruction give a sequence of quadruples generated in each case –***

(i) ***If P & Q then x = x + 1***
ELSE x = x + 2

(ii) ***X GT Y OR Y LT Z & U EQ V***

(iii) ***IF X LT Y then P = P + 2***
ELSE P = P – 2

(iv) ***X = P AND Q OR R***

Sol. (i) ***IF P & Q then x = x + 1***
Else x = x + 2

(a) (AND, P,Q,T_1)
(b) (IF, T_1, (6,3) (6,6))
(c) (+,x, 1,T_2)
(d) (= , T_2 , , x)
(e) GoTo, , , (6,8))
(f) (+ , x , 2, T_3)
(g) (= , T_3 , , x)
(h) END

(ii) X GT Y Or Y LT Z & U EQ V

(a) (GT, X, Y, T_1)
(b) (LT, Y,Z, T_2)
(c) (EQ, U,V, T_3)
(d) (OR, T_1, T_2, T_4)
(e) (AND, T_3 , T_4, T_5)

(iii) IF X LT Y then P = P + 2 ELSE P = P –2

(a) (LT, X, Y, T_1)
(b) (IF, T_1, (6,3) (6,6))
(c) (+, P,2, T_2)
(d) (= , T_2, , P)
(e) (GoTo , , , (6,8))
(f) (– , P, 2,T_3)
(g) (= T_3, , P)
(h) END

(iv) X = P AND Q OR R

(a) (AND, P,Q,T_1)
(b) (OR, T_1, R, X)

Prob.17. Generate code for the following C statements for the target machine assuming all variables are static. Assume three registers are available –

```
x = a[i] + 1
a[i] = b[c[i]]
a[i][j] = b[i][k]*c[k][j]
a[i] = a[i] + b[j]
a[i] + = b[j]
```

(R.G.P.V., June 2006)

Sol. Three-address code for each of given C statements are obtained as follows –

x = a [i] + 1 →

$t_1 = i*4$
$t_2 = addr(a) - 4$
$t_3 = t_2 [t_1]$
$t_4 = t_3 + 1$
$x = t_4$

a[i] = b [c(i)]

$t_1 = i*4$
$t_2 = addr(a) - 4$
$t_3 = t_2 [t_1]$
$t_4 = i * 4$
$t_5 = addr(c) - 4$
$t_6 = t_5[t_4]$
$t_7 = t_6*4$
$t_8 = addr(b) - 4$
$t_9 = t_8[t_7]$
$t_3 = t_9$

a[i][j] = b[i][k] *c[k][j]

Let all a, b and c be a 10 × 20 arrays.

Therefore

$t_1 = i*20$
$t_1 = t_1 + j$
$t_1 = t_1*4$
$t_2 = addr(a) - 84$
$t_3 = t_2[t_1]$
$t_4 = i*20$
$t_4 = t_4 + k$
$t_4 = t_4*4$
$t_5 = addr(b) - 84$
$t_6 = t_5[t_4]$
$t_7 = k*20$
$t_7 = t_7 + j$
$t_7 = t_7*4$
$t_8 = addr(c) - 84$
$t_9 = t_8[t_7]$

$t_{10} = t_6 * t_9$

$t_3 = t_{10}$

a[i] = a[i] + b[j]

$t_1 = i*4$

$t_2 = addr(a) - 4$

$t_3 = t_2[t_1]$

$t_4 = j*4$

$t_5 = addr(b) - 4$

$t_6 = t_5[t_4]$

$t_7 = t_3 + t_6$

$t_3 = t_7$

a[i] + = b[j]

$t_1 = j*4$

$t_2 = addr(b) - 4$

$t_3 = t_2[t_1]$

$t_4 = i*4$

$t_5 = addr(a) - 4$

$t_6 = t_5[t_4]$

$t_7 = t_6 +$

$t_7 = t_3$