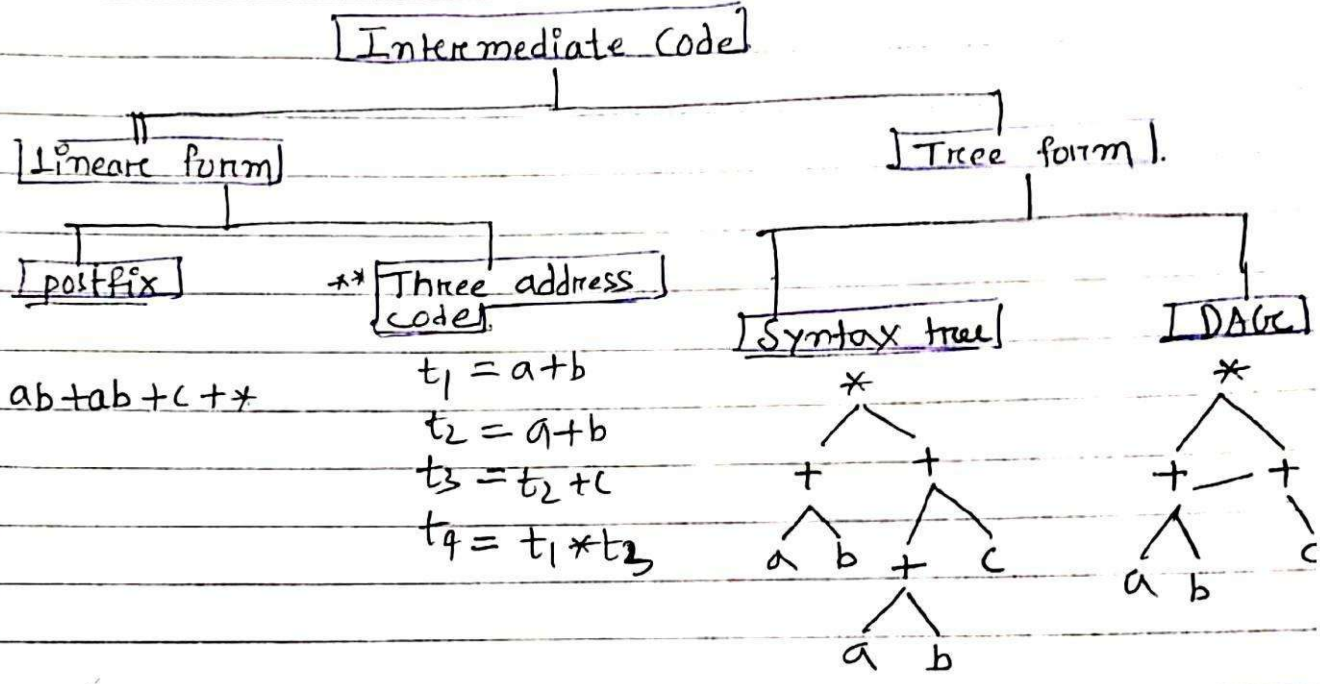
INTERMEDIATE CODE GENERATION

- Intermediate code generation =



ab+ab+c+*

$t_1 = a+b$
$t_2 = a+b$
$t_3 = t_2+c$
$t_4 = t_1 * t_3$

- Types of 3 address Code :

① $x = y \text{ op } z$
② $x = \text{op } y$
③ $x = y$
④ if x (relation op) y goto L.
⑤ goto L.
⑥ $A[i] = x$
   $y = A[i]$.
⑦ $x = *p$
   $y = \&y$.

- Various way to represent three address code =

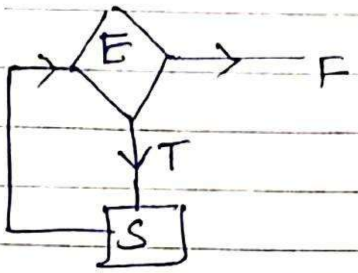
given instruction : $(a+b) * (c+d) + (a+b+c)$

1) $t_1 = a+b$
2) $t_2 =$ ~~$a+b$~~ $-t_1$
3) $t_3 = c+d$
4) $t_4 = t_2 * t_3$
5) $t_5 = a+b$
6) $t_6 = t_5 + c$
7) $t_7 = t_4 + t_6$.

3 ways =

Quadruples				TRIPLE			Indirect triples.
opr	op1	op2	result	opr	op1	op2	
1) +	a	b	t_1	1) +	a	b	1)(i) (1)
2) −	t_1	—	t_2	2) −	t_1	.	2)(ii) (2)
3) +	c	d	t_3	3) +	c	d	3)(iii) (3)
4) *	t_2	t_3	t_4	4) *	t_2	t_3	4)(iv) (4)
5) +	a	b	t_5	5) +	a	b	5)(v) (5)
6) +	t_5	c	t_6	6) +	t_5	c	6)(vi) (6)
7) +	t_4	t_6	t_7	7) +	t_4	t_6	7)(vii) (7)
adv: statement can be moved around.				adv: space is not wasted.			adv: statements can be moved.
dis: too much of space wasted.				dis: statements cannot be moved.			dis: Two memory access.

ex — While E do S. (how to convert while loop into three address code)



```
L: if (E==0) goto L1
   S
   goto L
L1:
```

→ when E is false goto L1. otherwise S.

or

```
L: if(E) goto L1
   goto last
L1: S
   goto L
last:
```

ex —

```
while (a<b) do
   x = y+z
```

```
L : if (a<b) goto L1
    goto Last
L1: t = y+z
    x = t
    goto L
Last:
```

• Back paetcing :

→ Whenever ~~are~~ statement are not ~~int~~ in the tree add code, then by using all 'if' statements make them into tree address code.

ex =

if(a<b) then t=1
else t=0. → not in the form of three add code.

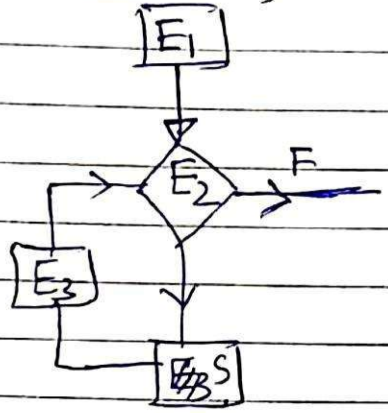
(i): if (a<b) goto (i+3)
(i+1): t = 0
(i+2): goto (i+4)
(i+3): t = 1
(i+4) → now in the form of three add code.

ex =

if$\left(\overset{t_1}{(a<b)}\text{ and }\overset{t_2}{(c<d)}\text{ or }\overset{t_3}{(e<f)}\right)$ then 1
else 0

100) if (a<b) goto 103
101) $t_1 = 0$
102) goto 104
103) $t_1 = 1$
104) if (c<d) goto 107
105) $t_2 = 0$
106) goto 108
107) $t_2 = 1$
108) if (e<f) goto 111
109) $t_3 = 0$
110) goto 112
111) $t_3 = 1$
112) $t_4 = t_1$ and t_2
113) $t_5 = t_4$ or t_3 .

$\boxed{ex}$ = for (E_1; E_2; E_3) (for loop to 3-add code)
{S ; }



(how for loop work)

for loop represent in the form of three address code =

```
f(i=0; i<10; i++)
    a=b+c;
```

```
   i=0
L: if (i<10) goto L1
   goto Last
L1: t1 = b+c
   a = t1
   t = i+1
   i = t
   goto L
Last:
```

$\boxed{ex}$ = switch (i+j) (switch case to 3-add code)

```
{
   case(1): a=b+c;
            break
   case(11): P=Q+R
            break;
   default: n=Y+Z;
            break;
}
```

switch statement convert into three add code =

```
    t = i+j
     goto test
L1: t1 = b+c
     a = t1
     goto Last
L2: t2 = Q+R
     P = t2
     goto Last
L3: t3 = y+z
     x = t3
     goto Last.
test: if(t==1) goto L1
       if (t==2) goto L2
       goto L3
Last:
```

[ex]= Two Dimentionay array convert into 3-~~code~~ add code =

A [4] [4]

00	01	02	03
10	11	12	13
20	21.	22	23
30	31	32	33

A [2 3]

$2 * 4 + 3 = 11$
(no. of R) (E) (no. of C)

after cross 11 element ~~to~~ you will get output.

RMO :
00 01 02 03 10 11 12 13 20 21 22 23 . . .

CMO : 00 10 20 30 01, 11, 21 31 02 12 22 32 . .

RMO: (R * no. of element + C) * element size.

$x = A[y, z]$ $\boxed{A : 10 \times 20} \rightarrow$ 2D array.

$$\underset{(100)}{\text{Base add}} + \underbrace{(y * 20 + z) \times 4}$$

('4' is element size in bit)

→ after crossing this amount of number we get result of $A[Y, Z]$.

3-add code =

$t_1 = y * 20$

$t_2 = t_1 + z$

$t_3 = t_2 * 4$

$t_4 =$ base address of A

$x = t_4[t_3]$. (Base Address add with offset).