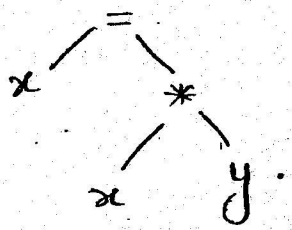
Intermediate code forms.

- Tree
  - Syntax tree
    - eg. $x = x * y$



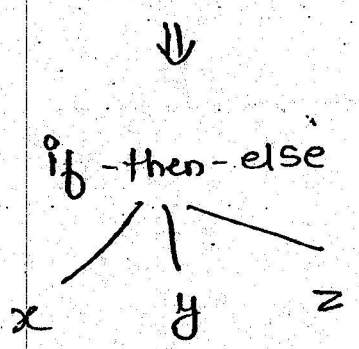
    - Here leaf nodes = operands
      Intermediate nodes = operators.

A syntax tree of a operation represented with all interior nodes as the operator and leaf nodes as operands. The operators and keywords labels of the parsetree are moved to their parent node and chain of single productions is replaced by single link.

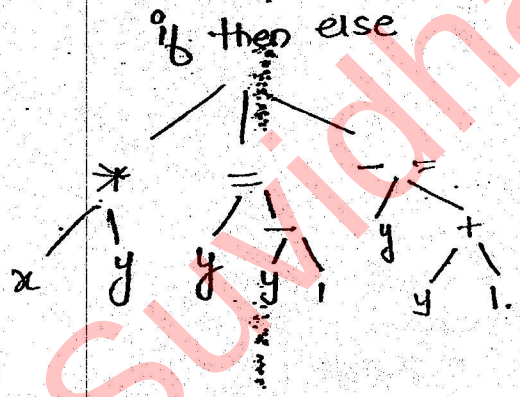
- Linear form
  - 3 address code
  - Postfix Notation

Syntax tree

eg. if x then y else z.



eg. if x * y then y = y - 1 else y = y + 1.



Postfix

In a postfix notation, the operator comes after the operands.

for eg:- expression A + B gives AB+

The statement if x then y else z, is write it as xyz?

3. Address code

The general form of the 3-address code is

A = B op C. Where A, B, C - operands

op is operator.

all modern compilers now a days generates 3 address code as intermediate code. Every statement in 3 address code contains atmost 3 operands.

~~Order~~ ∄

Other forms of 3-address code is -:

1. A = opC
2. A = B
3. A = B[i]
4. B[i] = A

eg. $x = x+1$

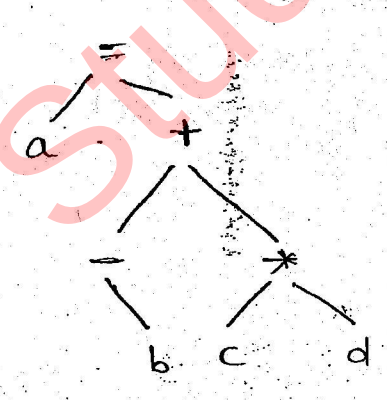
3AC

$t = x+1$
$x = t$

Q. Represent the following statements in syntax tree, postfix and 3-addr. code.

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

Syntax tree



Reverse polish notation (postfix).

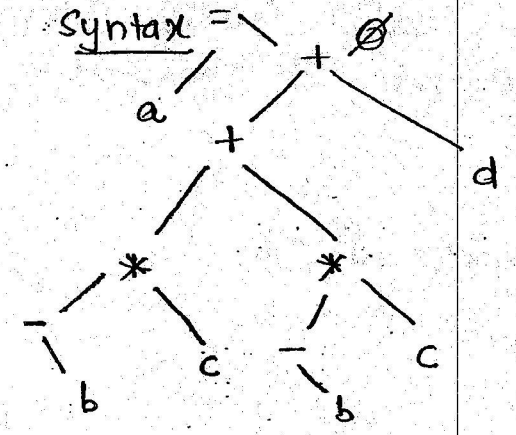
$a\ b - c\ d * + =$

3 address code

$t_1 = -b$
$t_2 = c*d$
$t_3 = t_1 + t_2$
$a = t_3$

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

⇒ Syntax =



Postfix

a b - c * b - c * + d + =

3 AC code

$t_1 = -b$

$t_2 = t_1 * c$

$t_3 = -b$

$t_4 = t_3 * c$

$t_5 = t_2 + t_4$

$t_6 = t_5 + d$

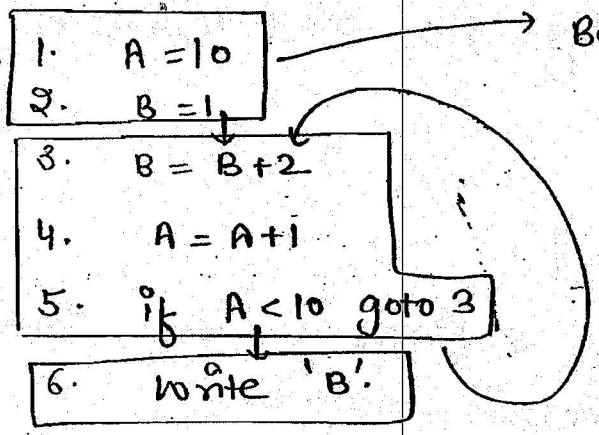
$a = t_6$

3 AC code mai kabhi bhi optimization nai lagate jo hai as it is kate hai (care vo result already ho).

1. A = 10
2. B = 1
3. while (A < 10)
   {
   B = B + 2
   A = A + 1
   }
   print (B);

3 AC code does not use while statement (instead of that it uses, if goto etc)

In 3AC form, above code is written as -:



↓ This have 3 basic block.

Whenever code is written as Basic Block the it is called control flow Graph or just flow graph.

Block is set of statement.

★ Every high level ~~statement~~ program represented as a basic blocks in the 3-address code intermediate forms as shown in the figure.

★ Implementation of 3AC

A 3AC can be implemented in 3 ways -:

① Quadruples | Operand | Arg1 | Arg2 | Result | → Prob. lots of temp. variab

② Triples

③ Indirect Triples.

★ Quadruple -: A Quadruple is a data structure with 4 fields as shown in the figure -:

OP	Arg1	Arg2	Result

In a Quadruples, more no. of temporary variables are used to store the intermediate results. This problem can be overcome by using the point addresses in the triple system.

* Tripple

Op	Arg1	Arg2

* Indirect tupple :- Even 3 fields are not there, they maintain pointers.

Q. Consider the $a = -b + c * d$, represent in Quadruples, tripples ~~and~~ ~~Indirect~~.

Quadruple :-

op	Arg1	Arg2	Result
−	b		t_1
*	c	d	t_2
+	t_1	t_2	t_3
=	t_3		a

Triple :-

pointer address	op	Arg1	Arg2
→ (1)	−	b	
(2)	*	c	d
(3)	+	(1)	(2)
(4)	=	(3)	a

Q. Represent the following expression as a quadruple & triple.

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

Quadruple

op	arg 1	arg 2	Result
-	b		t_1
*	t_1	c	t_2
-	b		t_3
*	t_3	c	t_4
+	t_2	t_4	t_5
=	t_5		a

Triple

	op.	arg 1	arg 2.
(1).	-	b	
(2).	*	(1)	c
(3).	-	b	
(4).	*	(3)	c
(5)	+	(2)	(4)
(6).	=	(5)	a.

Q. Consider the following 3 AC.

$t_1 = -e$

$t_2 = b * c$

$t_3 = d * t_1$

$t_4 = t_2 + t_3$

$a = t_4$

$a = b * c + d * -e$

Which of the following expression represents above 3 AC.

(a). $a = b * (c + d) * -e$

(b). $a = b * c + d * -e$

(c). $a = b * c + -d * e$

(d). None of the above

Q. Consider the following expression -

$a = b * -c + d$

Which operator contains in both arg. fields point address if it is represented in triples?

(a) *

(b) −

(c) +

(d) None of these.

	op	arg1	arg2
(1)	−	c	
(2)	*	b	(1)
(3)	+	(2)	d
(4)	=	(3)	a