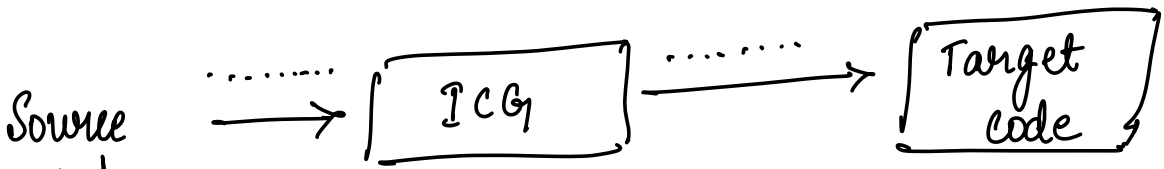
Intermediate code Generation : Chapter 5



In between all the phases

→ Machine Independent code

↓↓

run on different machines

⇒ Types :

① AST (Abstract Syntax Tree)

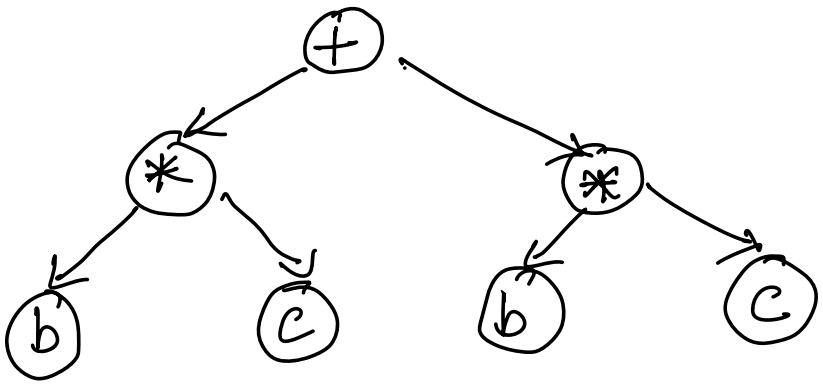
② DAG (Direct Acyclic Graph)

③ Postfix Notation

④ 3 Address code

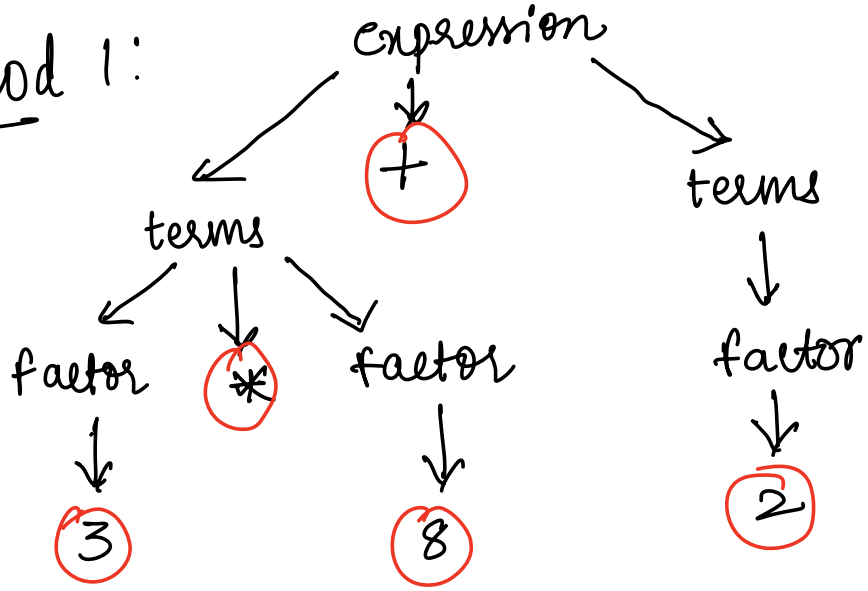
(i) AST: Tree representation of abstract syntatic structure of source code

Ex: (b * c) + (b * c)

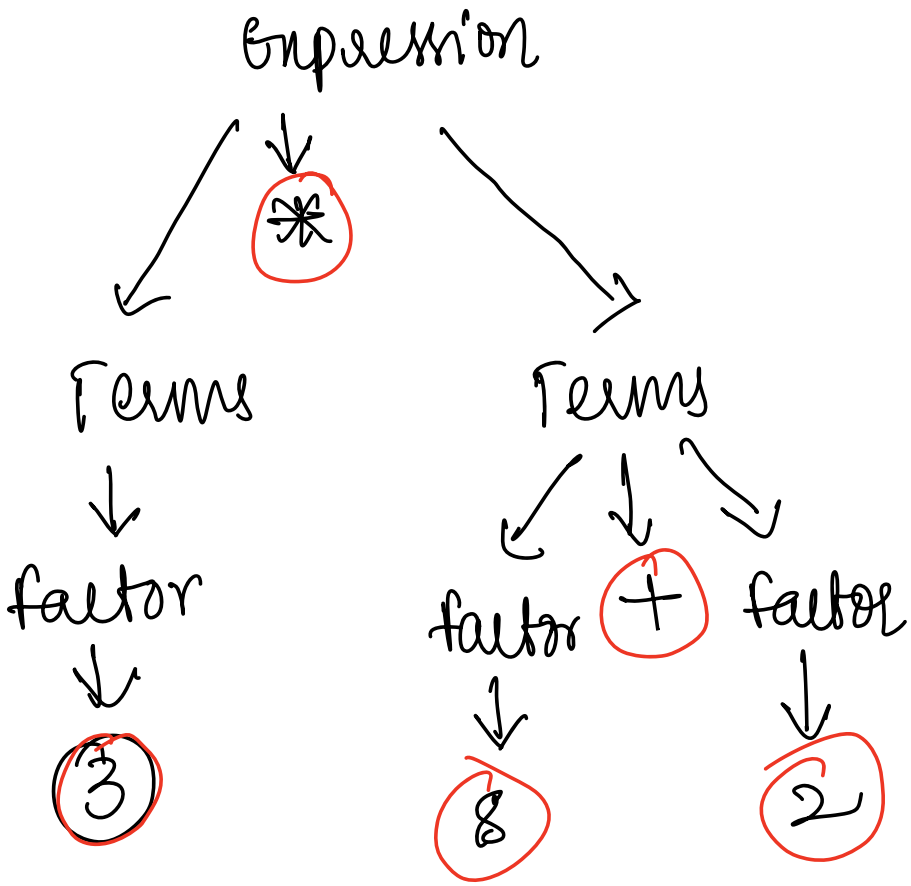


Ex: 3 * 8 + 2

Method 1:



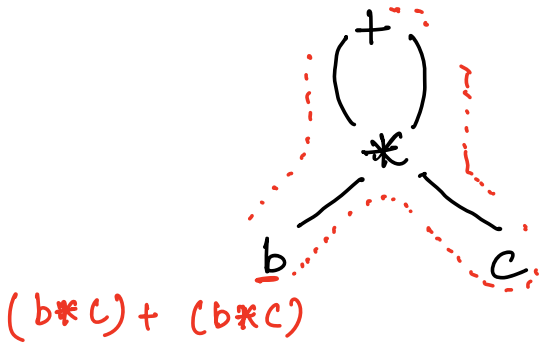
Method 2:



(2) Direct Acyclic Graphs: (DAG):

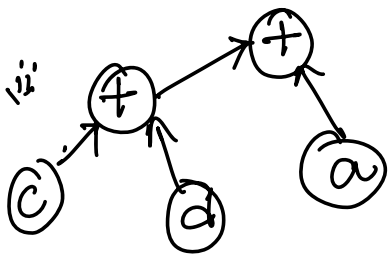
→ to represent structure of blocks.

Example: (b * c) + (b * c)

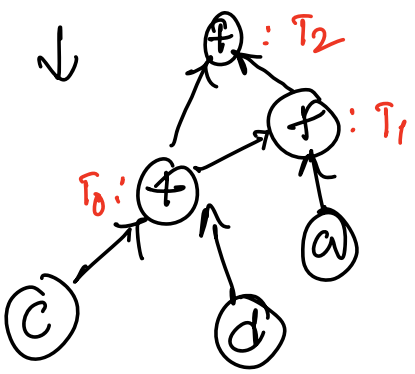


(ii) T_0 : c + d

(ii) T_1 : T_0 + a :



(iii) T_2 : T_0 + T_1



(3) Postfix Notation

Operator comes after the operand

→ operator follows operand.

Example: (b * c) + (b * c)

→ (b * c) (b * c) +

→ bc* bc* +

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

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

$ab*\ cd+\ -$

Example: $(ab)+(cd)$

$\Rightarrow\ ab\ cd+$

example: $(a+b)*c$

$\Rightarrow\ ab+\ c*$

④ 3 Address code:

Example: $(b*c)+(b*c)$

$b*c = T_1$

$b*c = T_2$

$T_3 = T_1+T_2$ (choose any alphabets)

Example: $d = a * b + c$

$t_1 = a * b$

$t_2 = t_1 + c$

$d = t_2$

Various statements in 3-address code are:

1. Assignment:

   1. $x = y \ op \ z$
   2. $x = op \ y$
   3. $x = y$

   $x = y + z$

   $x = + y$

   $x = y$

2. Jump:

   - Conditional: if x relop (relational op) y, goto L
     ex: if $x \geq y$, goto L
   - Unconditional: Go to L

3. Array Assignment:

$x = y[i]$

$x[i] = y$

If $i=0$, $x = y[0]$

$x[0] = y$

4. Pointer, address assignment:

$x = \&y$

$x = *y$