CS & IT Engineering

Compiler Design

Intermediate Code & Code Optimization

Lecture: 1

Deva sir

Topics to be covered:

- H.W. Questions
- Intermediate code
  - Representations

7.18

① $S \rightarrow S_1 \# T \quad \{S.val = S_1.val * T.val\}$

$S \rightarrow T \quad \{S.val = T.val\}$

$T \rightarrow T_1 \% R \quad \{T.val = T_1.val / R.val\}$

$T \rightarrow R \quad \{T.val = R.val\}$

$R \rightarrow id \quad \{R.val = id.val\}$

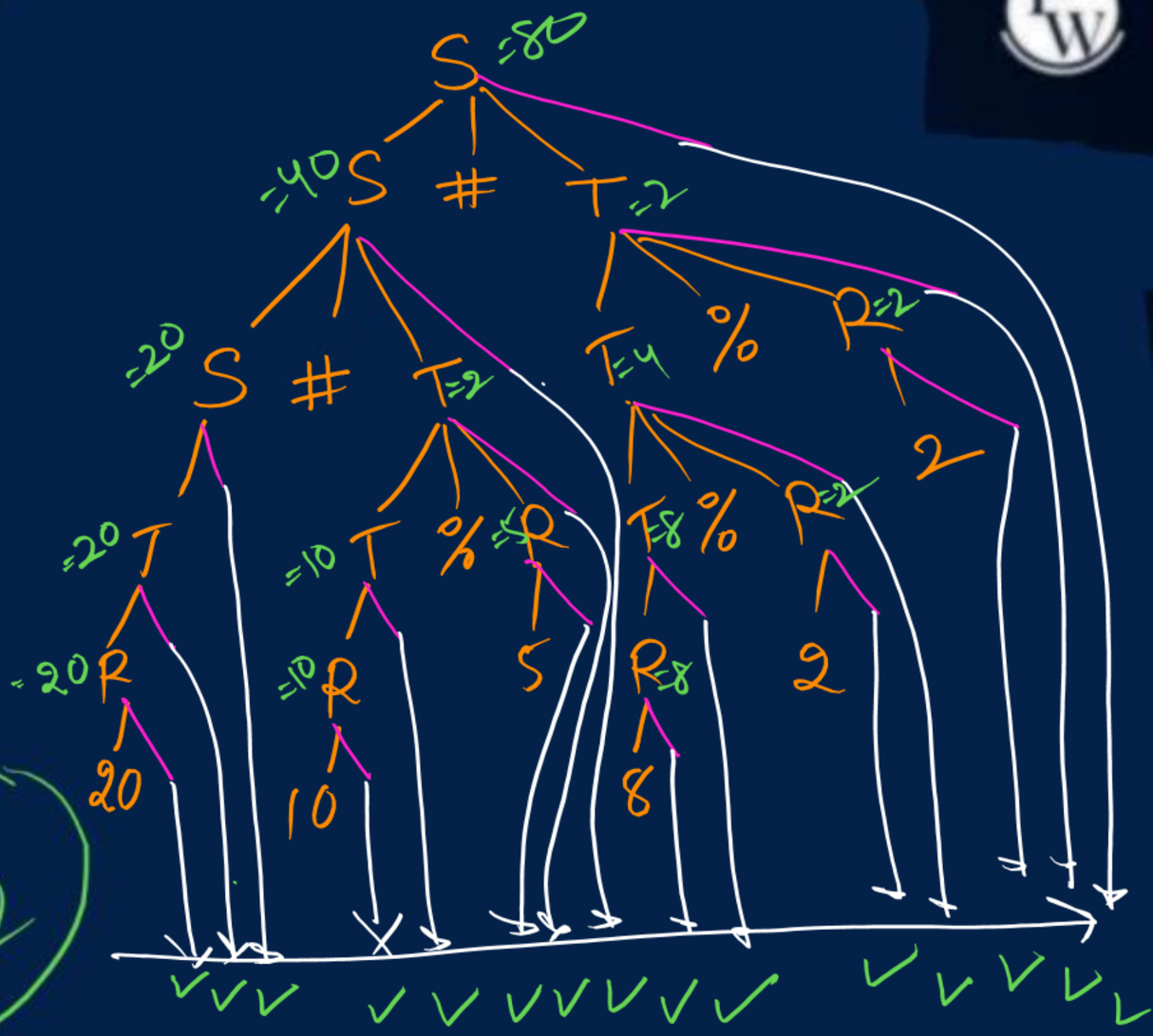
Input: 20 # 10 % 5 # 8 % 2 % 2

Compute the value at Root

20 * 10/5 * 8/2/2

2 =4 =2

40

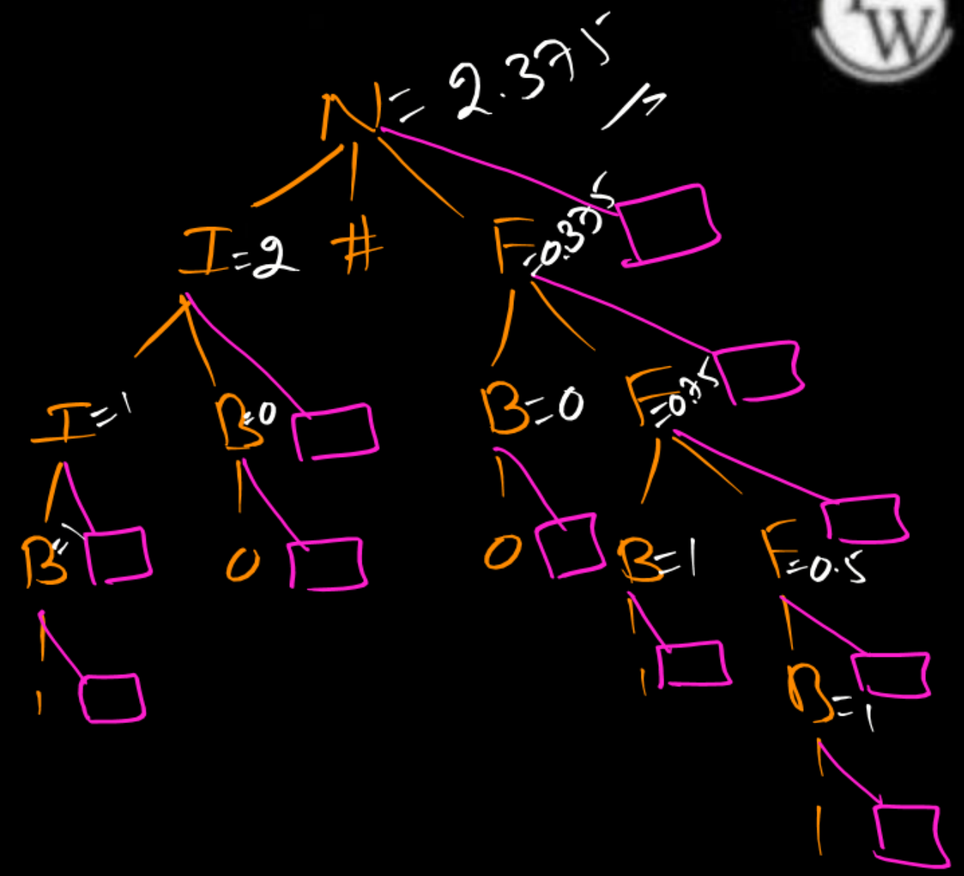


②

$N \rightarrow I \# F$	$N.val = I.val + F.val$
$I \rightarrow I_1 B$	$I.val = 2 I_1.val + B.val$
$I \rightarrow B$	$I.val = B.val$
$F \rightarrow B F_1$	$F.val = \frac{1}{2}(B.val + F_1.val)$
$F \rightarrow B$	$F.val = \frac{1}{2} B.val$
$B \rightarrow 0$	$B.val = 0$
$B \rightarrow 1$	$B.val = 1$

H.W.

Input 10 #011.



$\frac{1}{2}(0.75)$ $\frac{1}{2}(1.5)$

$=0.375$ $=0.75$

(3)

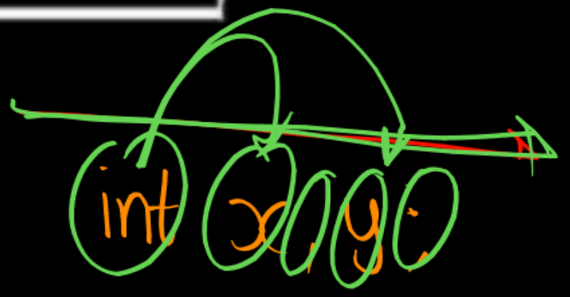
$D \rightarrow T\ L$	$L.type = T.type$
	$X_1.type = X_2.type$
$T \rightarrow int$	$T.type = int$
$T \rightarrow float$	$T.type = float$
$L \rightarrow L_1, id$	$X_3.type = X_4.type$; $AddType(id.entry, X_5.type)$
$L \rightarrow id$	$AddType(id.entry, X_6.type)$

$X_1 = L$
$X_2 = T$
$X_3 = L_1$
$X_4 = L$

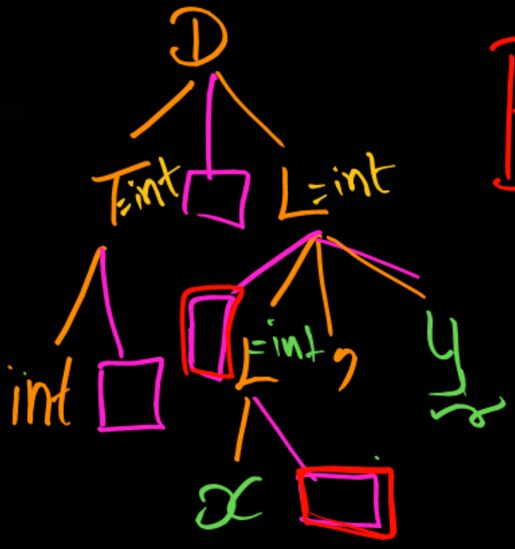
$X_5 = L$ or L_1

$X_6 = L$

x	int
y	int



AddType is a function that adds type information for id



④

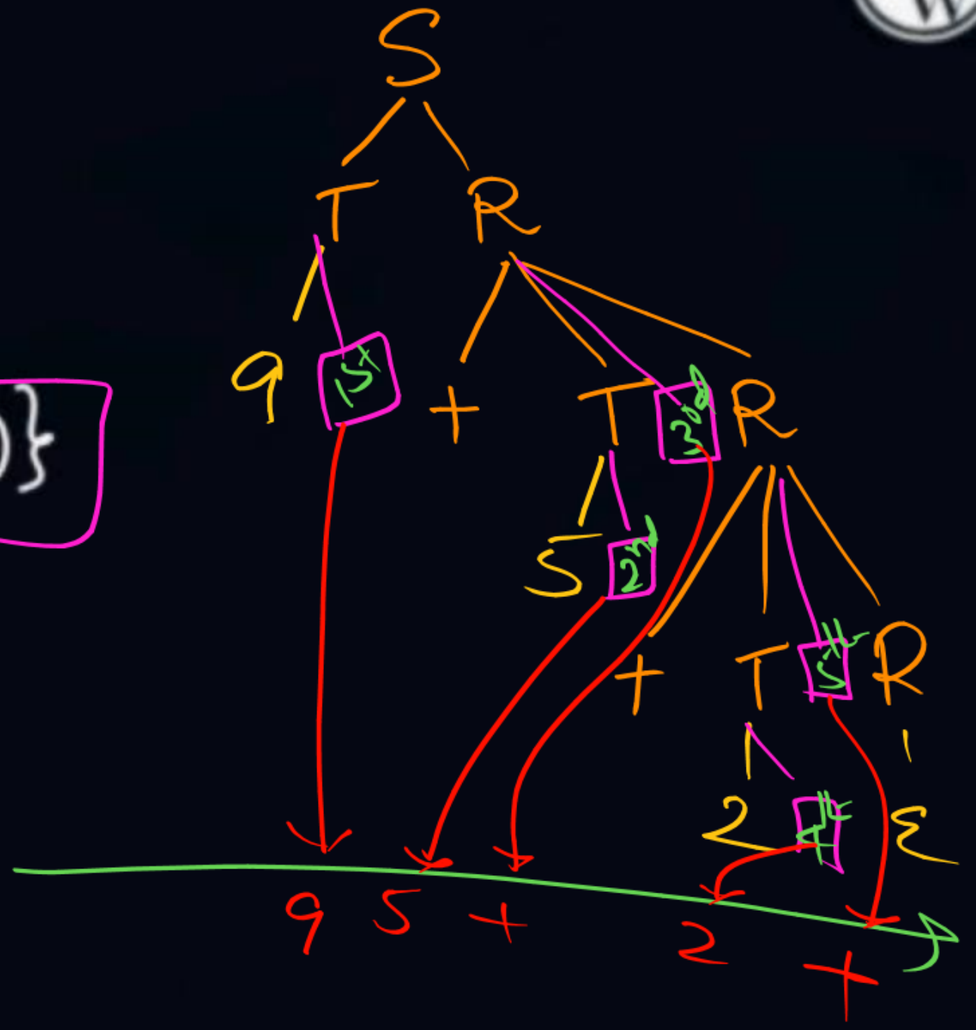
$S \rightarrow TR$

$R \rightarrow +T \{ print('+') \} R$

$R \rightarrow \varepsilon$

$T \rightarrow num \{ print(num.val) \}$

Input : 9+5+2



9 5 + 2 +

(5) $S \rightarrow id := E$ { gen(id.place = E.place) }

point

$E \rightarrow E_1 + E_2$ { t = newtemp(); E.place = t;
gen(t = E_1.place + E_2.place) }

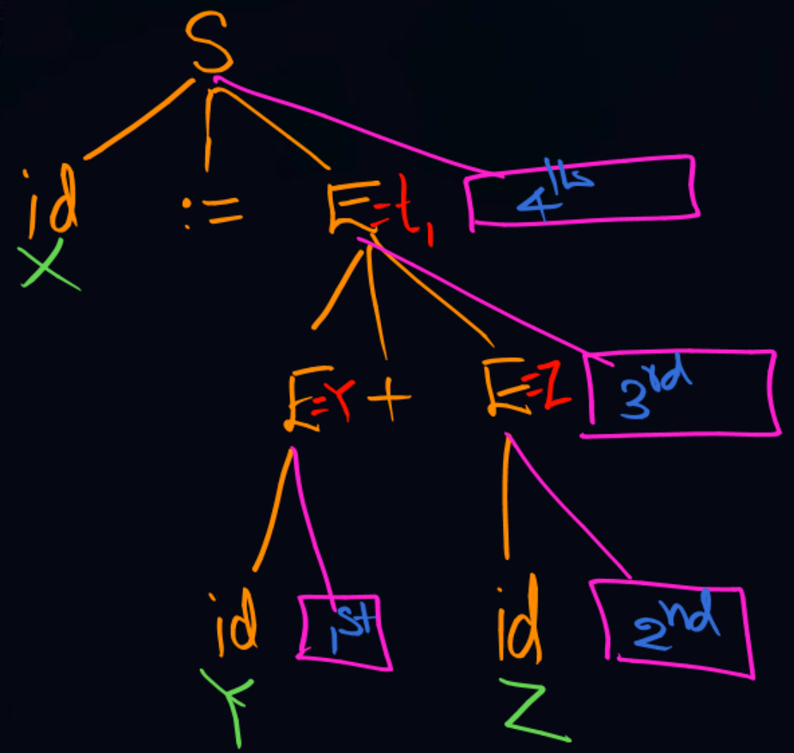
$E \rightarrow id$ { E.place = id.place }

Input. X := Y + Z

o/p:

$t_1 = Y + Z$

$X = t_1$

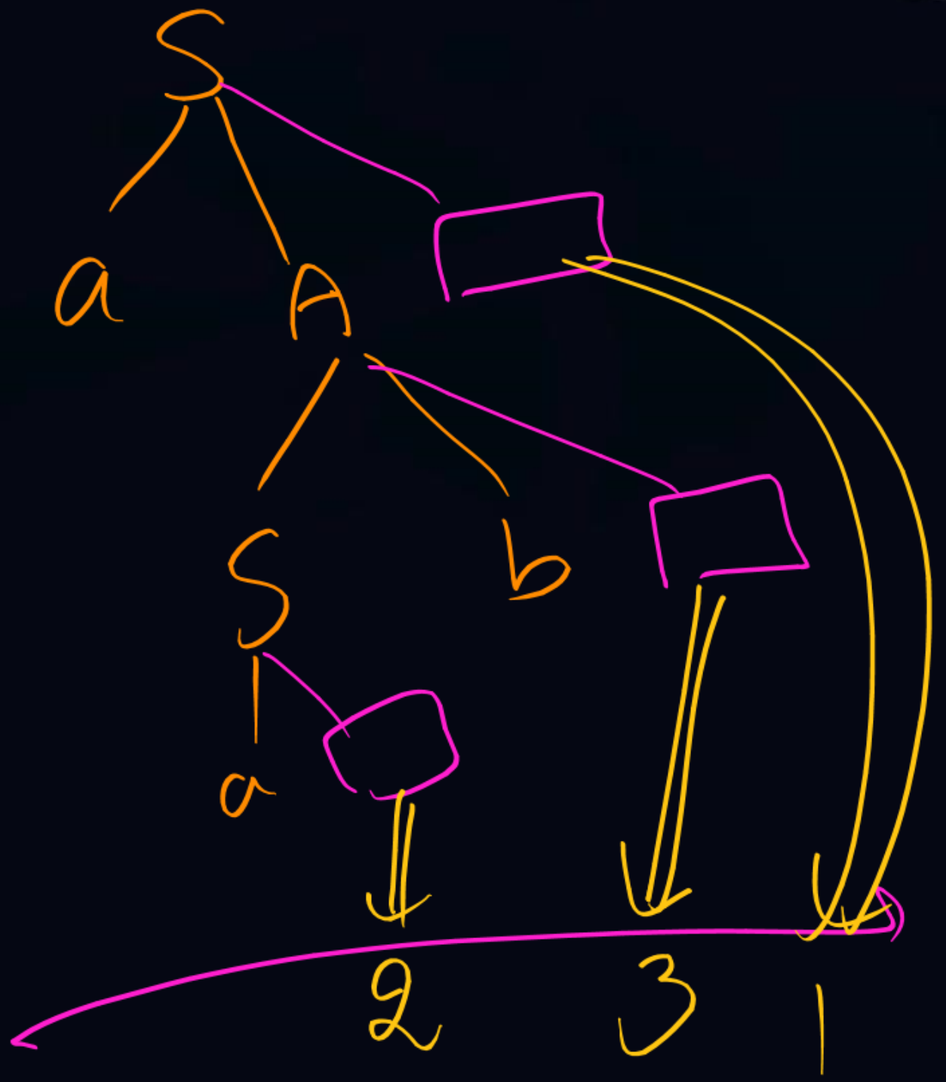


⑥ $S \rightarrow aA$ {print 1}

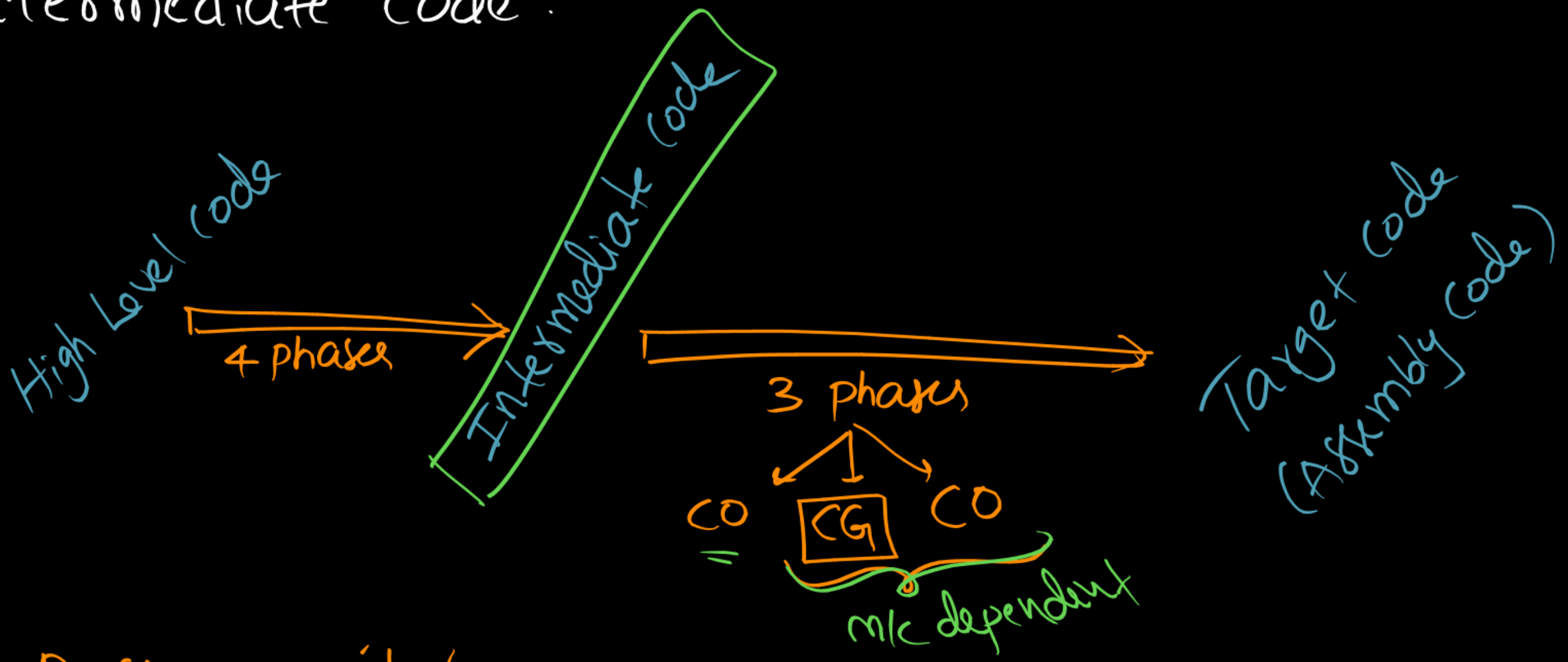
$S \rightarrow a$ {print 2}

$A \rightarrow Sb$ {print 3}

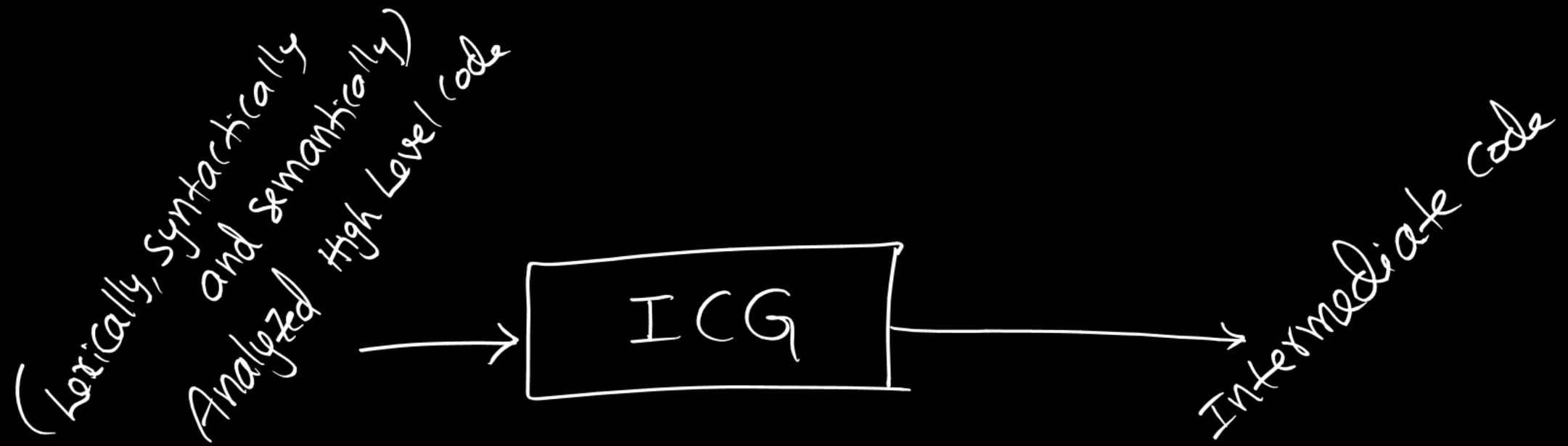
Input : aab



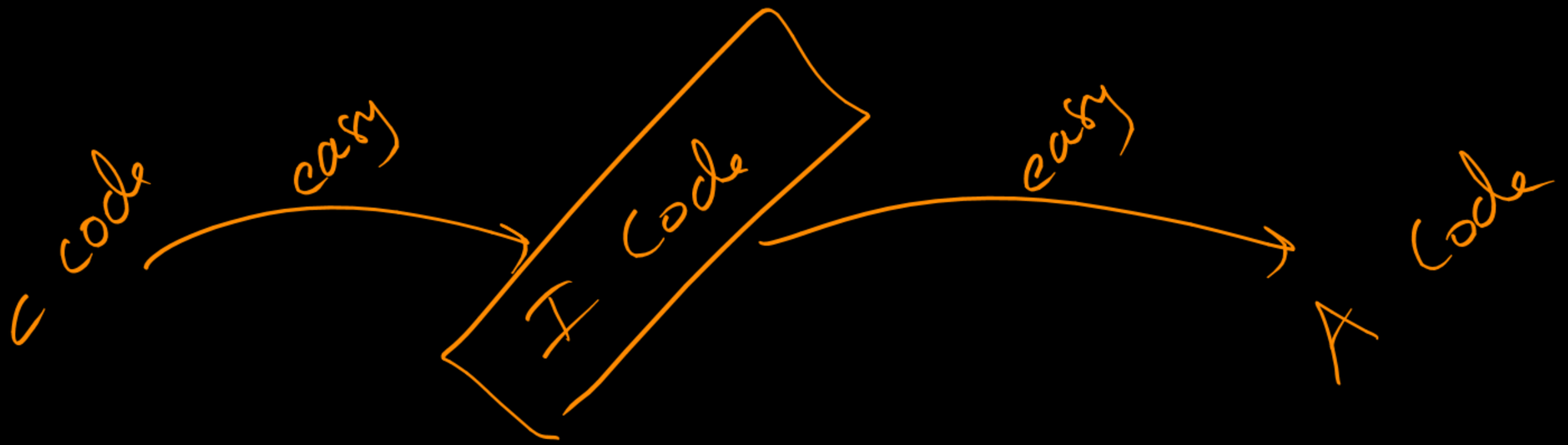
Intermediate code :



2 pass compiler


(lexically, syntactically and semantically) Analyzed High Level code
ICG
Intermediate code

$x = y * a + b$


C code
easy
I code
easy
A code

Intermediate code Representations (IR):

① Linear Form ⟶
- i) Postfix Form
- ii) 3AC / TAC / Three Address code
- iii) SSA code [Static Single Assignment]

② Non Linear Form
- i) Syntax Tree
- ii) Directed Acyclic Graph
- iii) Control Flow Graph

$x = a + b * c$

Postfix code :

$x = a + b * c$

$\Rightarrow x = a\ bc*\ +\ =$

$a+b \Rightarrow ab+$

$a*b \Rightarrow ab*$

$a-b \Rightarrow ab-$

$a=b \Rightarrow ab=$

$x = a + b * c$

Three Address code (3AC) (TAC)

↳ Every statement contains atmost 3 addresses (variables)

$t_1 = b * c$
$x = a + t_1$

3AC

5 variables

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

3AC

3 variables

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

3AC

3 variables

$x = a + b * c$

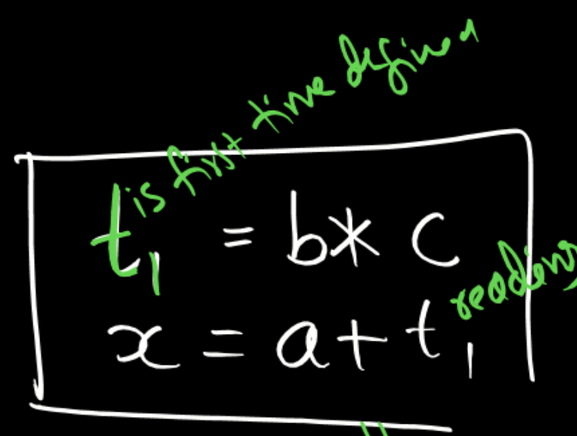
S → static

S → Single

A → Assignment (Definition)

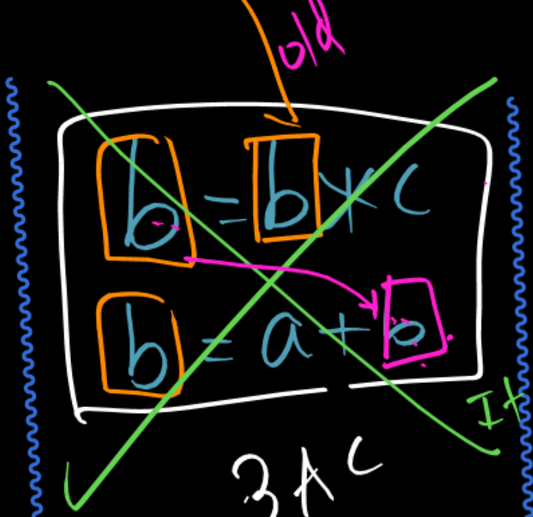
SSA code

↳ It is 3AC but every variable has single assignment.



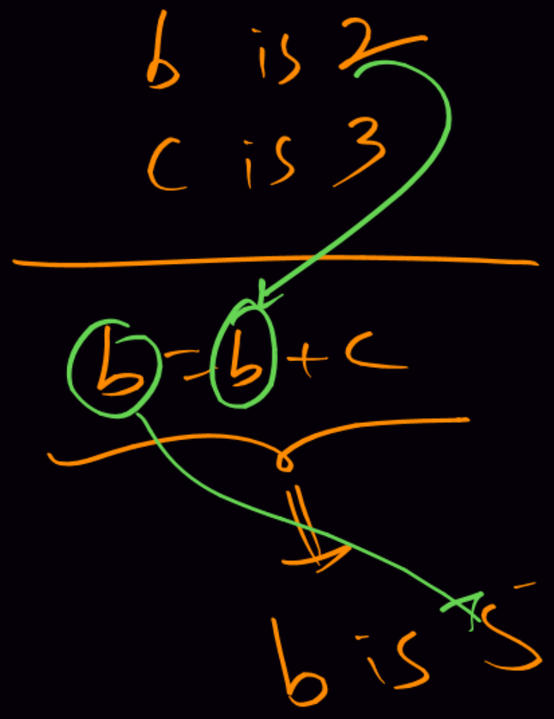
also SSA code

a, b, c, a, t_1 } Single definition



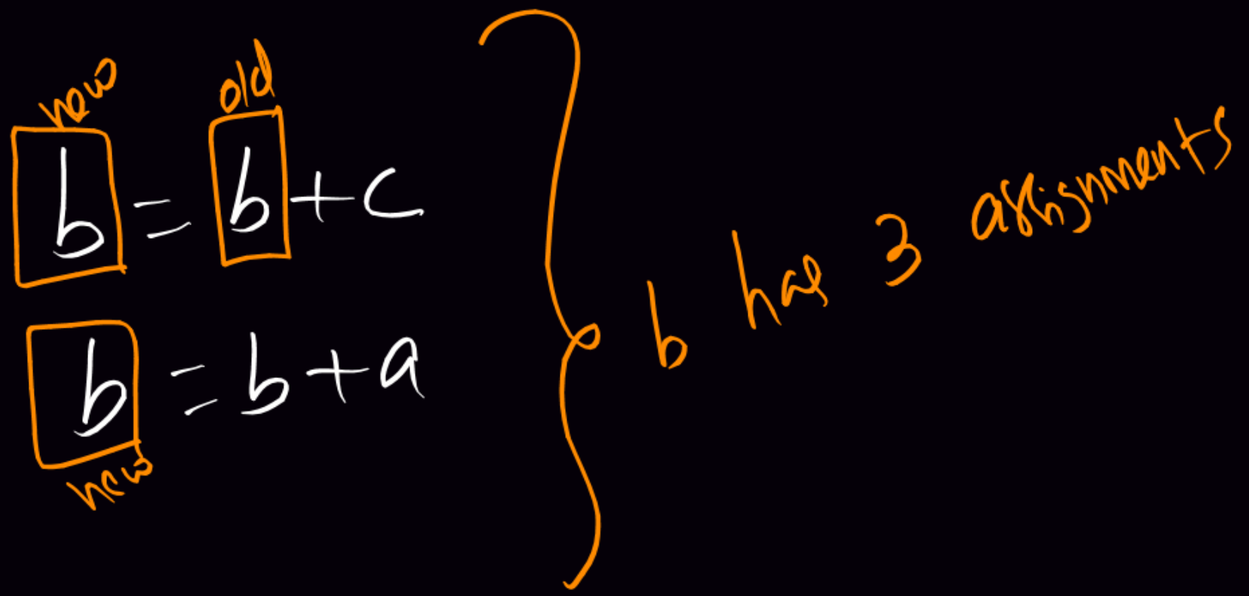
It is not SSA code

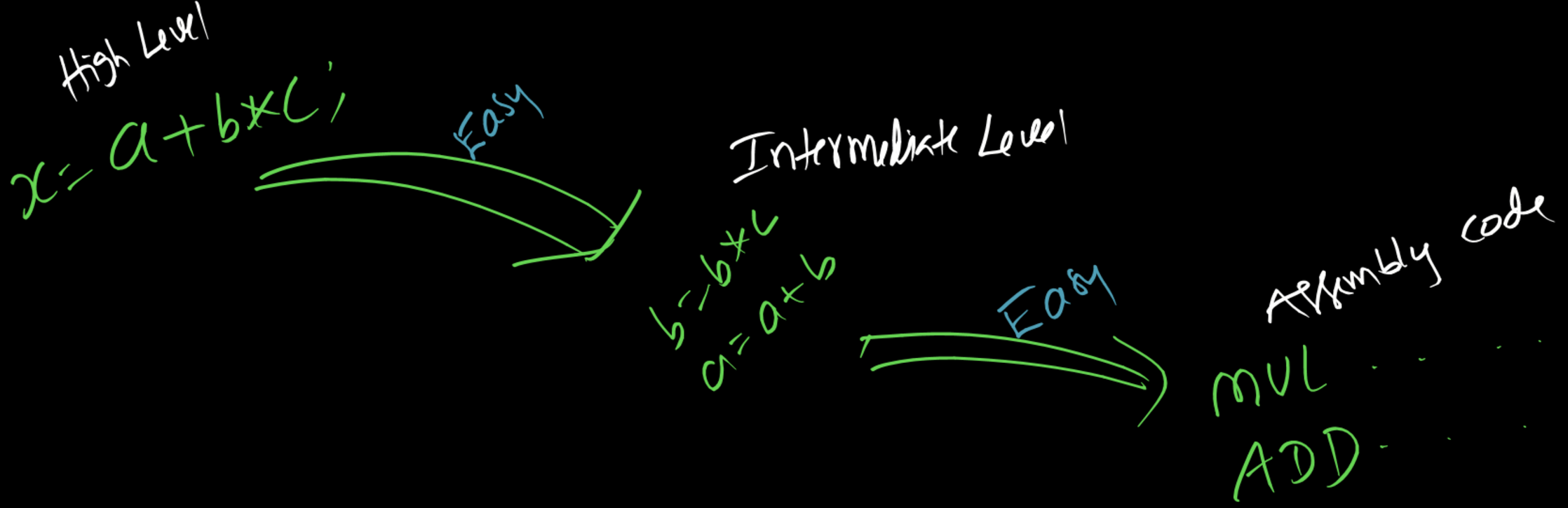
a has 1 definition
b has 3 definitions
c has 1 definition




```
b = b + c ;
```

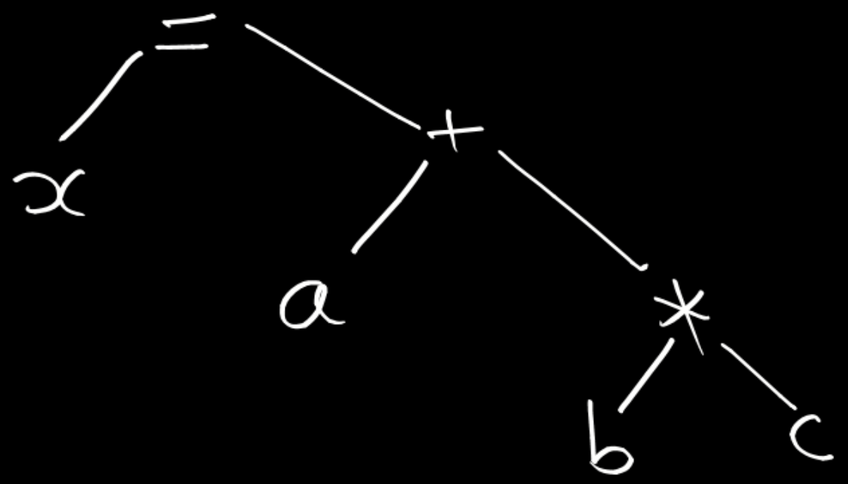
How many meanings are there for b?
(assignments)


new
old
b = b + c
b = b + a
new
b has 3 assignments


High Level
x = a + b * c ;
Easy
Intermediate Level
b = b * c
a = a + b
Easy
Assembly code
MUL . . . . .
ADD . . . . .

$$x = a + b * c ;$$

Syntax Tree (AST)(Abstract Syntax Tree) :



Leaf : operand

Non leaf : operator

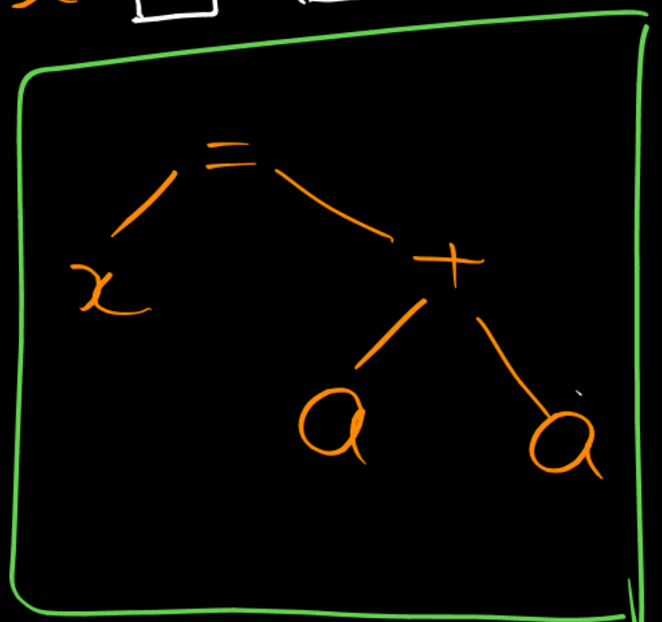
$a+b$

$a*b$

DAG : [Directed Acyclic Graph]

↳ It eliminates Common Sub expressions

$x = a + a$



Syntax Tree

$a + b$

DAG

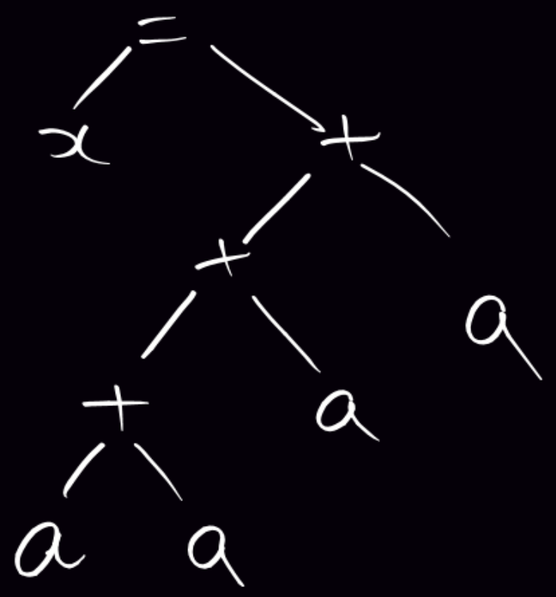
OR

DAG

2 nodes
2 edges

$$x = a + a + a + a$$

Syntax Tree



DAG

3 nodes
4 edges

Q1) $x = a + b - b + c$

Postfix

$x = ((a+b)-b)+c)$

$ab+$

$ab+b-$

$ab+b-c+$

$xab+b-c+=$

3AC with min no. of variables

$x = a+b-b+c$

$x = a+c$

$a = a+c$

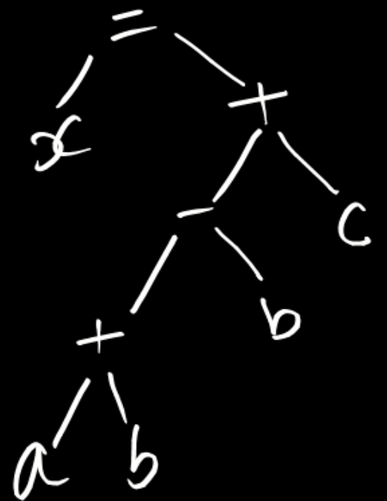
2 variables

SSA code min no. of variables

$a_1 = a+c$

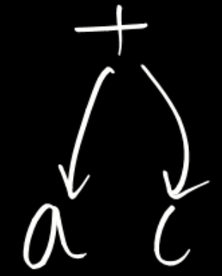
3 variables

Syntax Tree

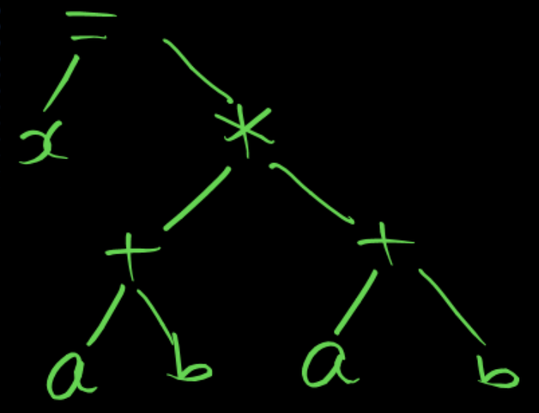


DAG

min nodes
min edges



Q2) $x = (a+b) * (a+b)$

Postfix	3AC	SSA	AST	DAG
$xab+ab+*=$	$a = a+b$ $a = a*a$ 2 variables	$a_1 = a+b$ $a_2 = a_1 * a_1$ 4 variables		 4 nodes 4 edges

(Q3) $x = a*b + a + b*c$;

3AC

Find min no. of variables

$x = (a*b) + a + (b*c)$

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

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

$c = a+c$

$b = b*c$

$b = b+a$

3 variables

SSA

Find min no. of variables

$c_1 = a+c$

$b_1 = b*c_1$

$b_2 = b_1 + a$

6 variables //

Q4) $x = a + b * c - b$

c
c
c

```
c = b * c
c = a + c
c = c - b
```

3AC

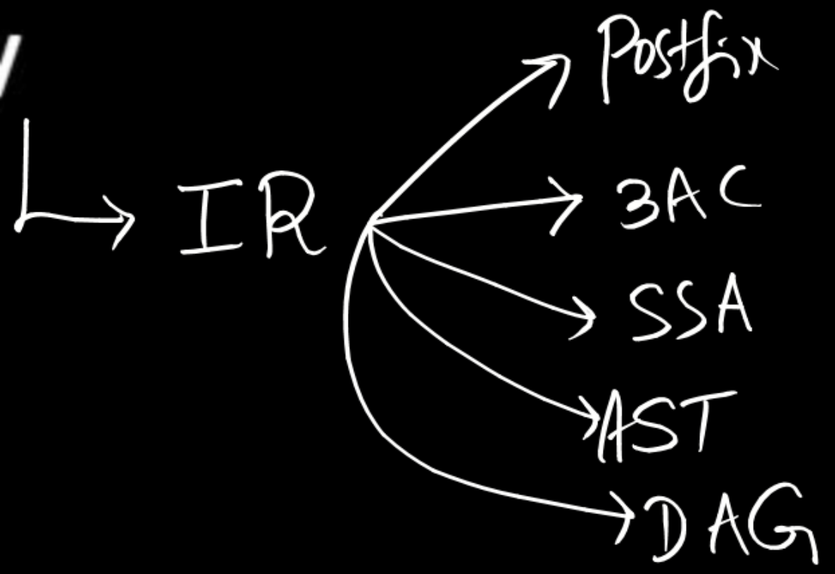
3 variables

```
c1 = b * c
c2 = a + c1
c3 = c2 - b
```

SSA

6 variables

Summary




Thank you
PW
Soldiers