unacademy

Lexical Analyzer - Part II

Complete Course on Compiler Design

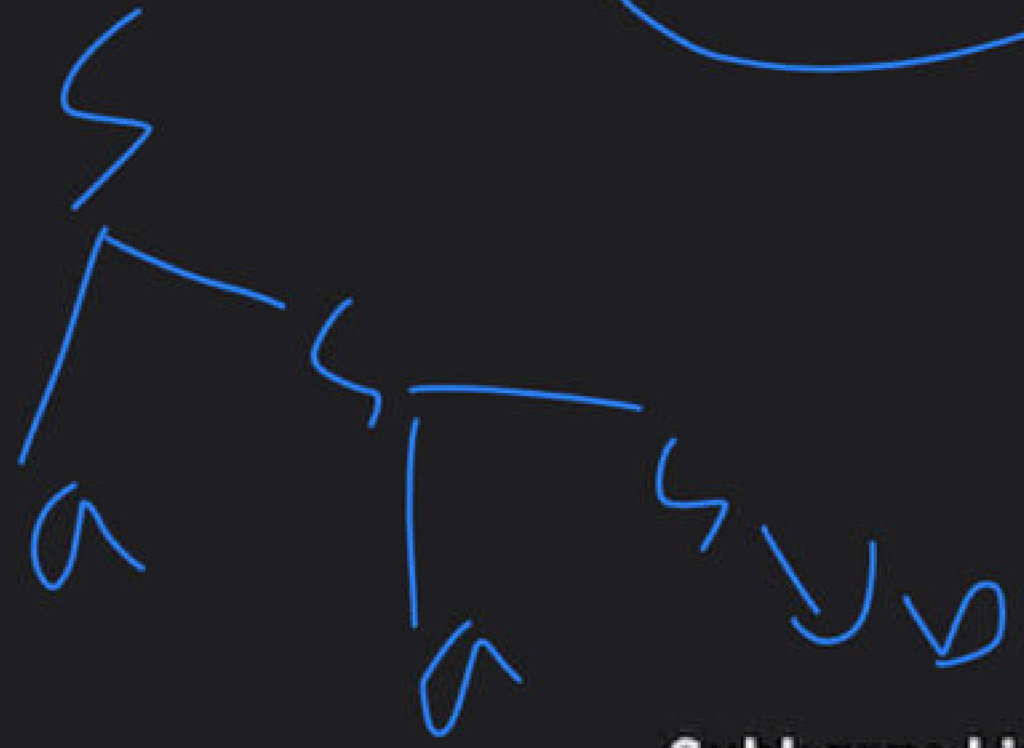
$S \to aS|b$

aab?

ab

b, ab, $\underline{aab}$

aaab



Subbarao Lingamgunta • Lesson 2 • Aug 16, 2023

~~ex~~

$x = a + b * 60$

$\Downarrow$

$\boxed{LA}$

$\Downarrow$

Tokens

x — identi — Token — (id, 1)

= — Assign — operator token

a — id — token — (id, 2)

+ — addi op — token

b — id — token — (id, 3)

* — mult op — token

60 — int const

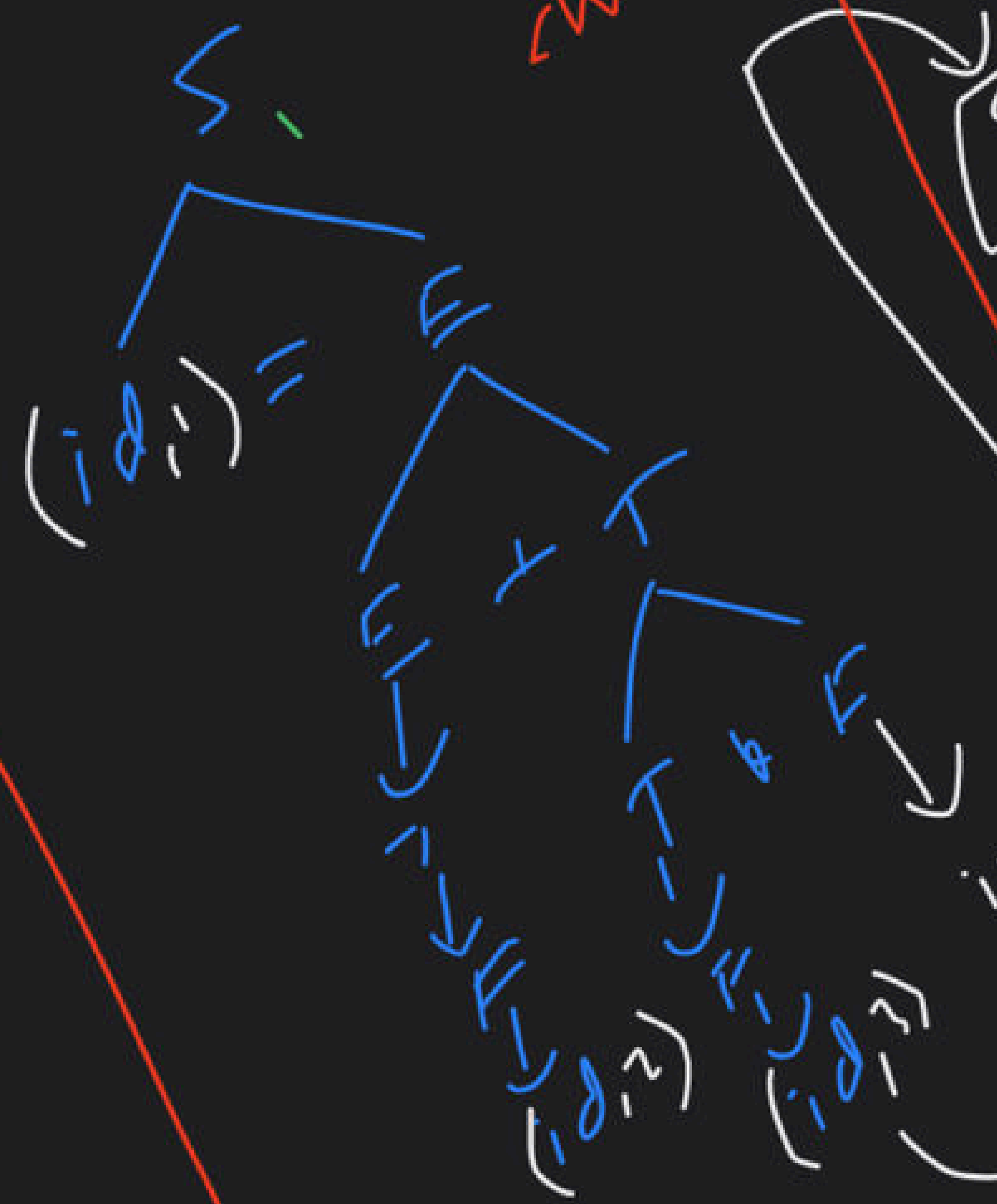
(id,1) = (id,2) + (id,3) * 60

$\Downarrow$

$\boxed{\text{Syn. An}}$

$\Downarrow$

Syntax checking

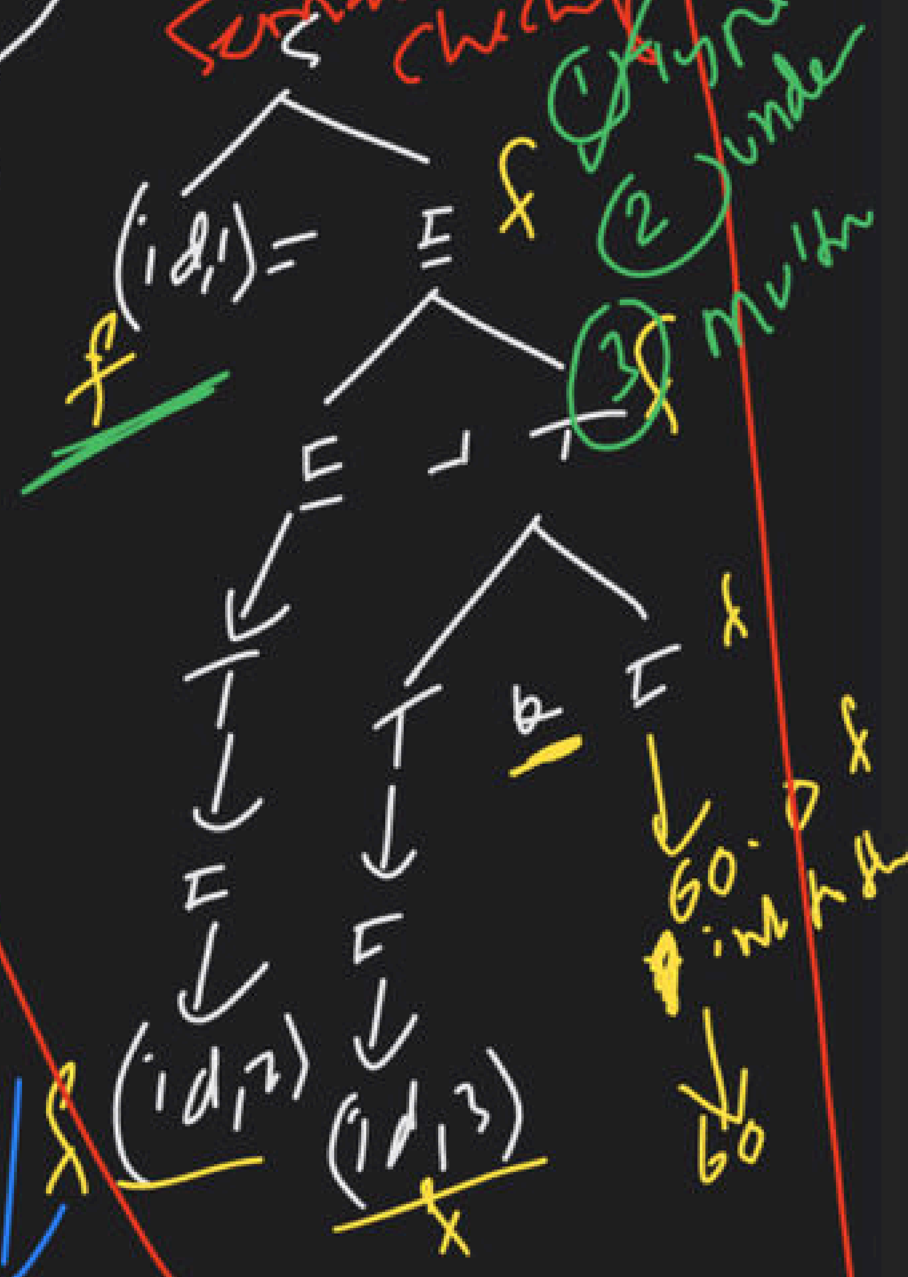


S.N	V.N	V.T
1	x	
2	a	
3	b	

$\boxed{\text{Sem. A}}$

$\Downarrow$

Semantic checking



$$x = \underbrace{a + \underbrace{b * 60.0}_{t_1}}$$

$\Downarrow$

I.C.G

$\Downarrow$

$t_1 = b * 60.0$

$t_2 = a + t_1$

$x = t_2$

$\Downarrow$

C.O

$t_1 = b * 60.0$

$x = a + t_1$

$\Downarrow$

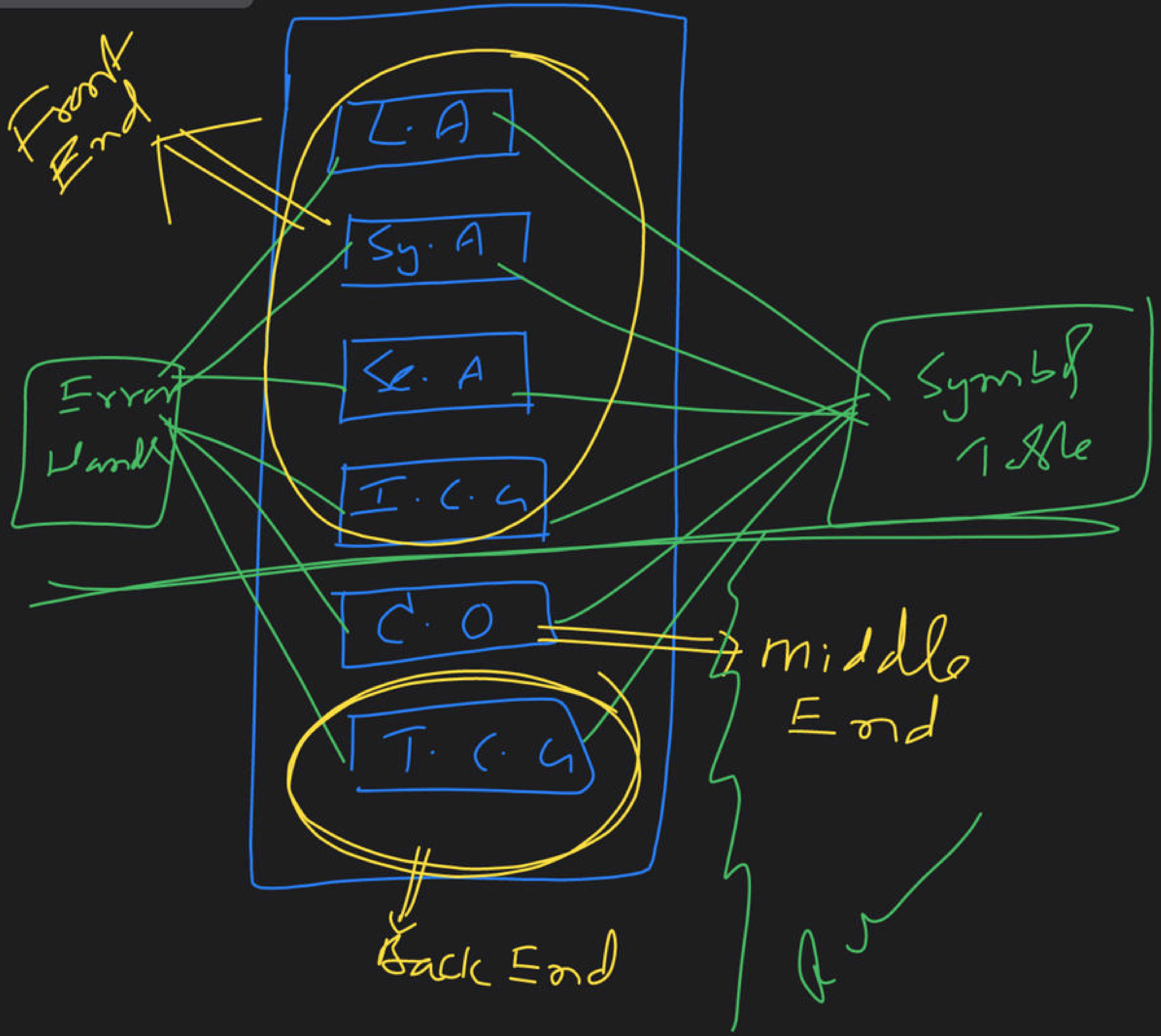
T.C.G

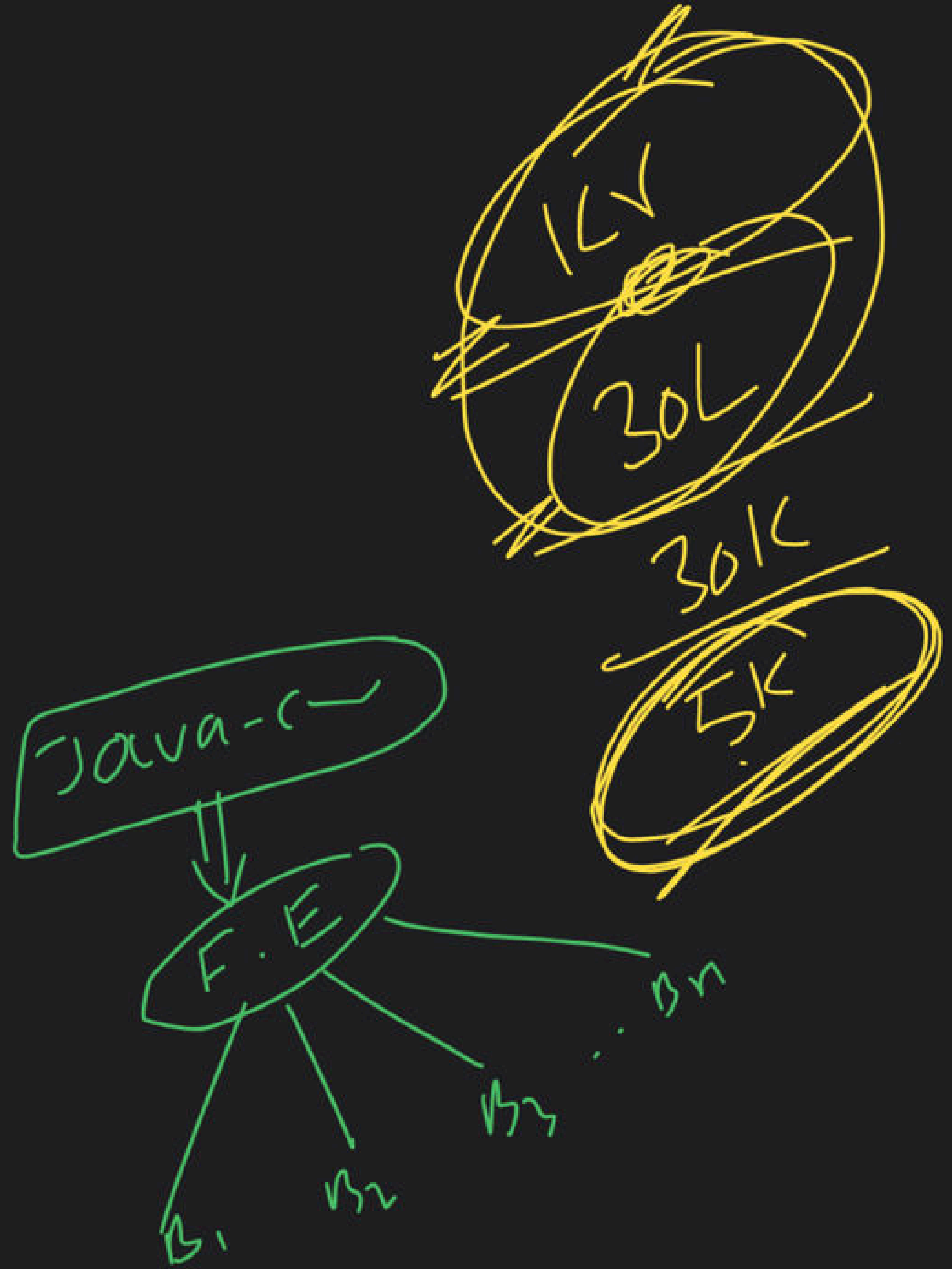
$\Downarrow$

MUL

ADD

Store


Front End
L.A
Sy.A
Se.A
I.C.G
C.O
T.C.G
Error Handler
Symbol Table
Middle End
Back End


1CV
30L
30K
5K
Java-c
F.E
B1
B2
B3
.. Bn

L.A

Sy.A

Se.A

I.C.G

C.O

T.C.G

Total compile at a time

⇓

Single pass compiler

- adv: Less time
- draw: more space

Total compilation multiple passes

⇓

Multipass compiler

- adv: Less space
- draw: more time

Lexical Analyzer