CS & IT ENGINEERING

Compiler Design

(5 - 8) marks

Compiler Introduction

One Shot

By- Venkat sir

Topics to be Covered

(2-3) 5

1 Topic — Lexical Alalysis

2 Topic — Parsing — 50%

3 Topic — Syntax Directed Translation — 20%

4 Topic — Intemediate Code Genaration

5 Topic — Code Optimisation(Dataflow Analysis) — 30%

6 Topic — Runtime Environment

Topics to be Covered

Topic ?????

Introduction

Phases of Compiler

Books for Reference

Topic ?????

1. Ullman

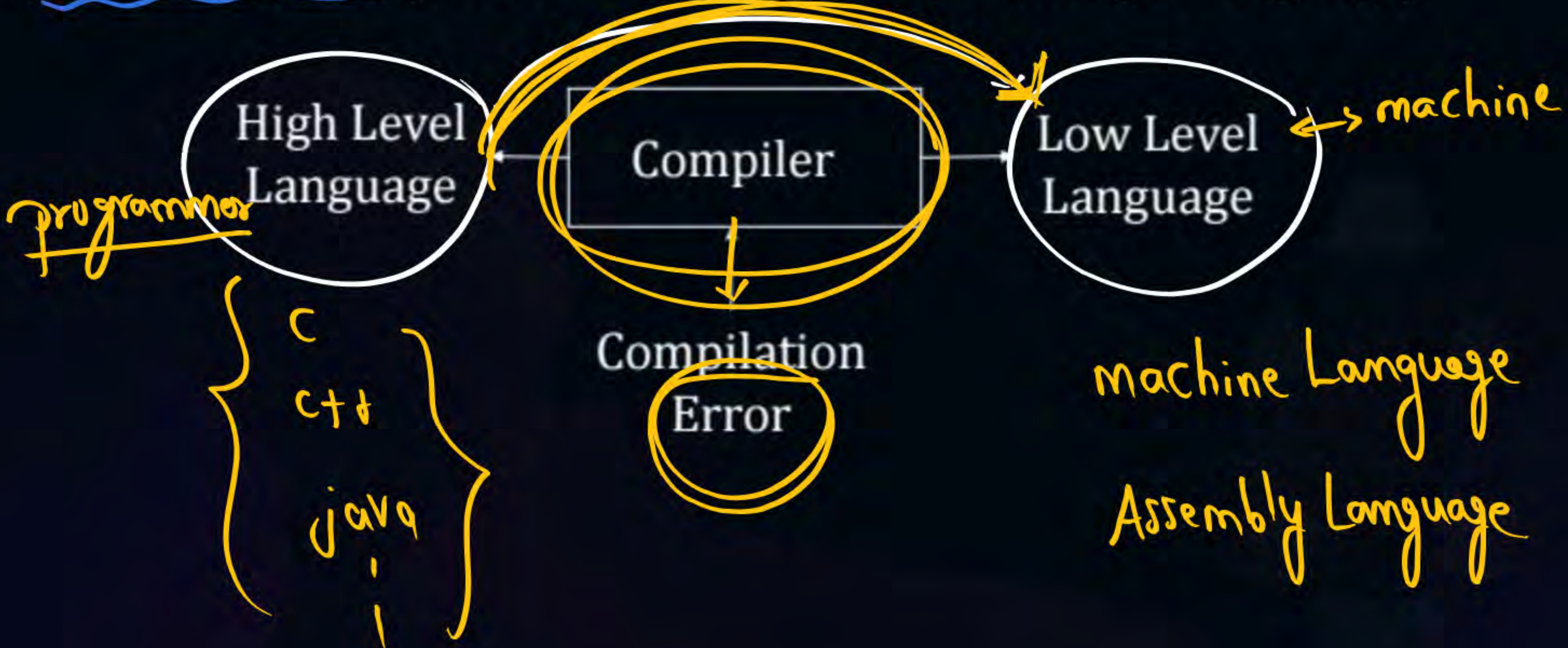
2. GATE PYQ's

Topic : Compiler Design

- Lexical Analysis
- Parsing
- Syntax Directed Translation
- Intermediate Code Generation
- Runtime Environment

Topic : Introduction of Compiler Design

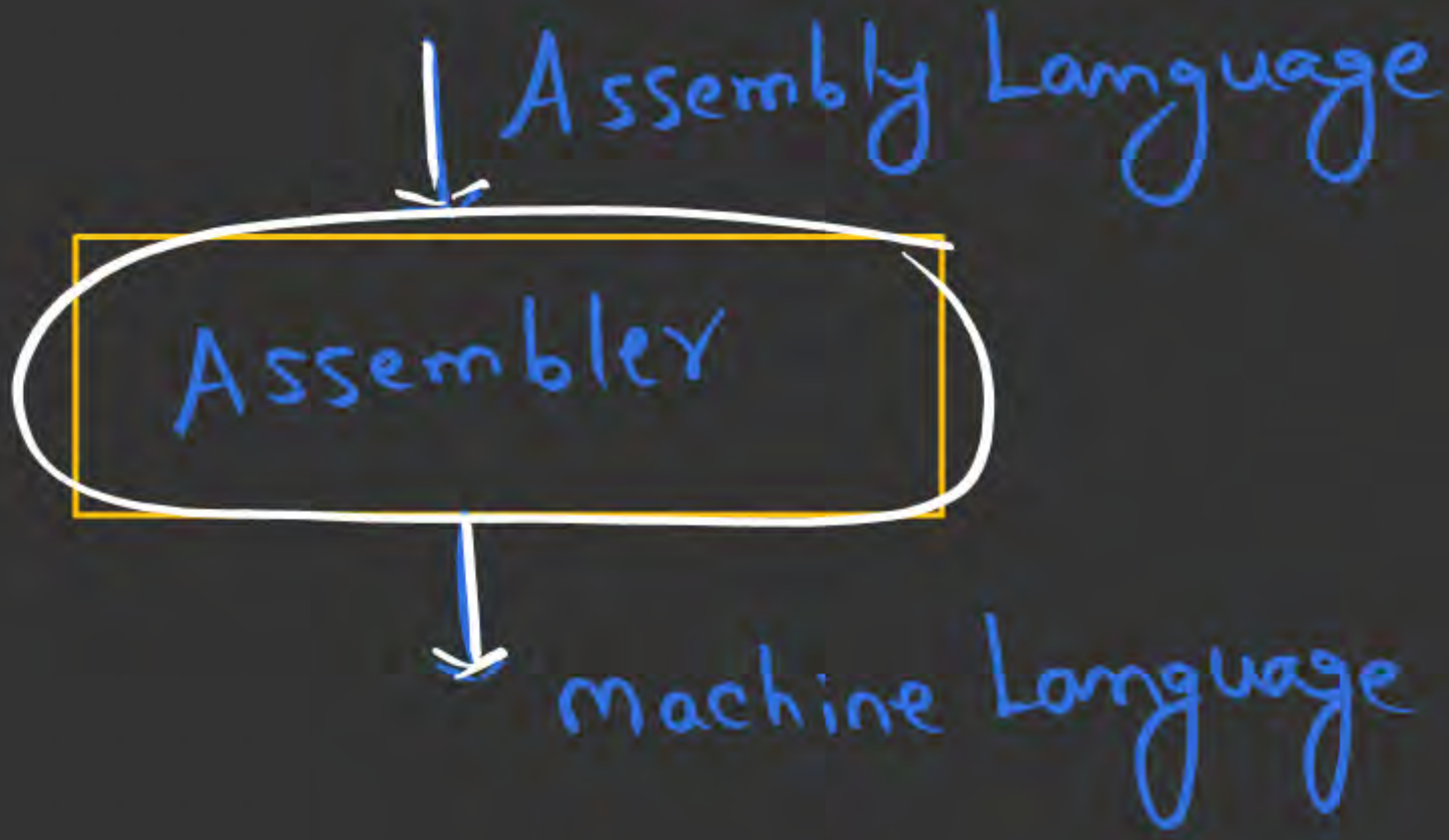
- Compiler is a translator which converts a program written in one language (Source Language) to other language (Object/Target Language).



$$\begin{tikzcd} \text{C} \arrow[r, "\text{Compile}"] & \text{Assembly Language} \end{tikzcd}$$

$$\begin{tikzcd} \text{java} \arrow[r, "\text{javaC}"] & \text{Byte Codes.} \end{tikzcd}$$

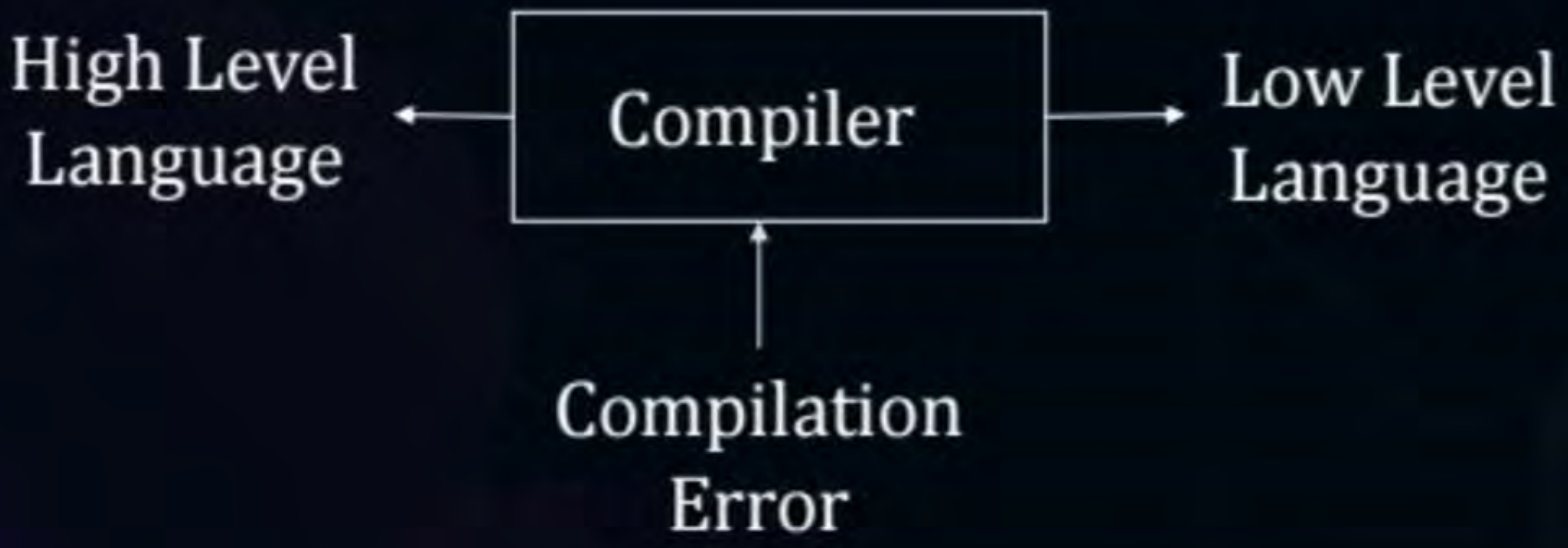
1. ✓ Error detection
2. ✓ Translation


Assembly Language
Assembler
machine Language

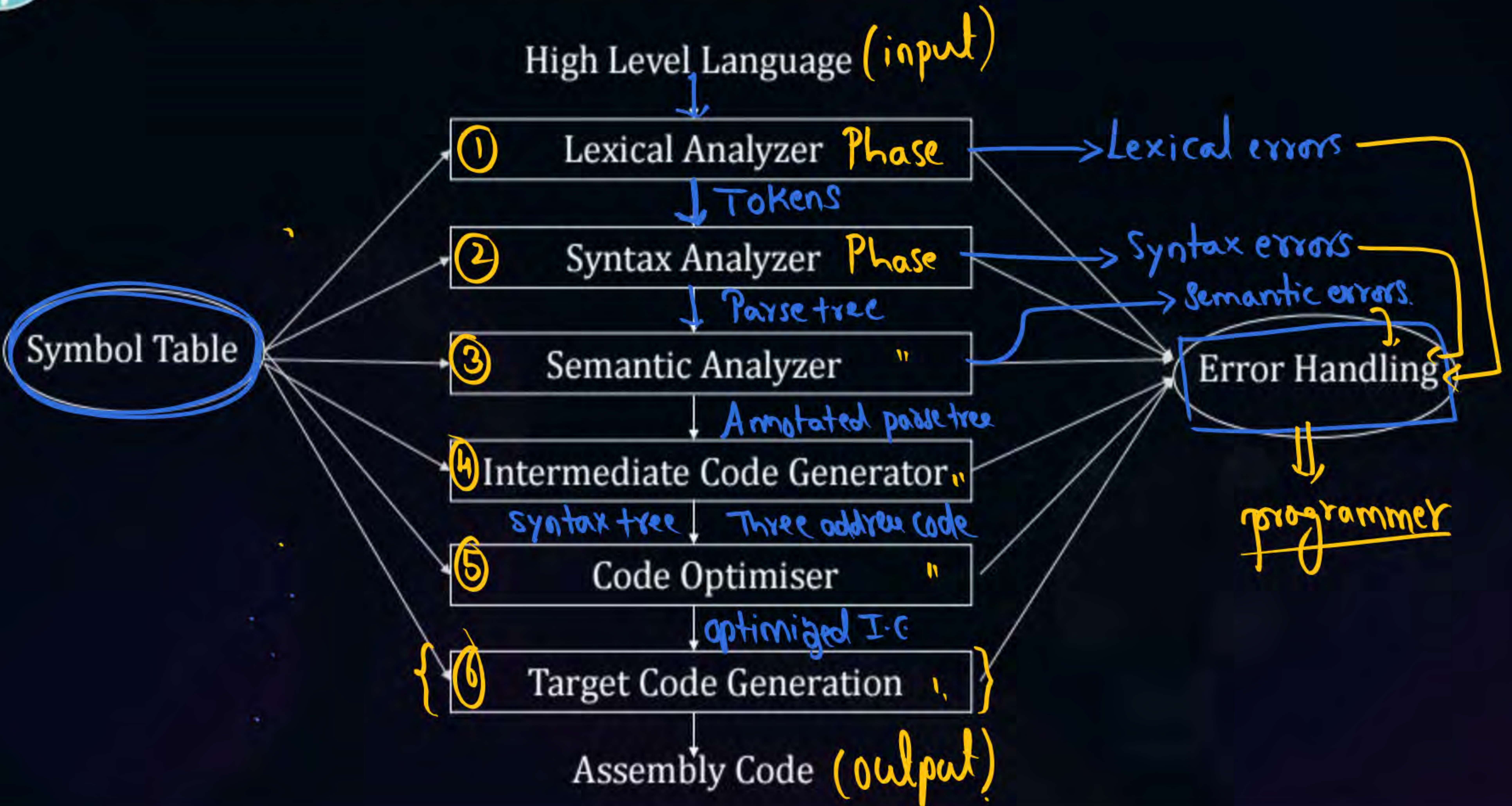
Topic : Introduction of Compiler Design

Program

- Compiler is a software which converts a program written in high level language (Source Language) to low level language (Object/Target/Machine Language).



Topic : Introduction of Compiler Design



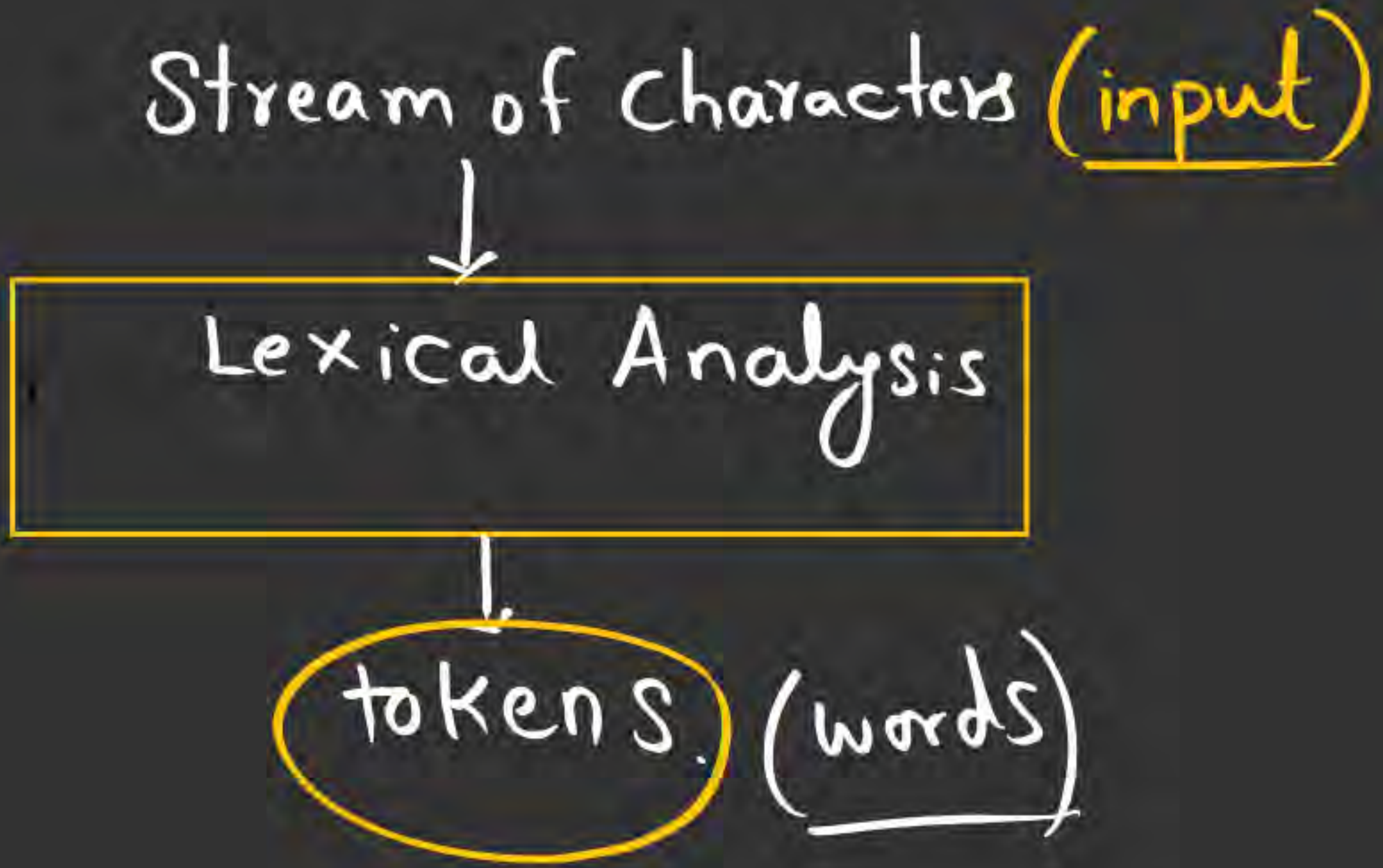
CFG

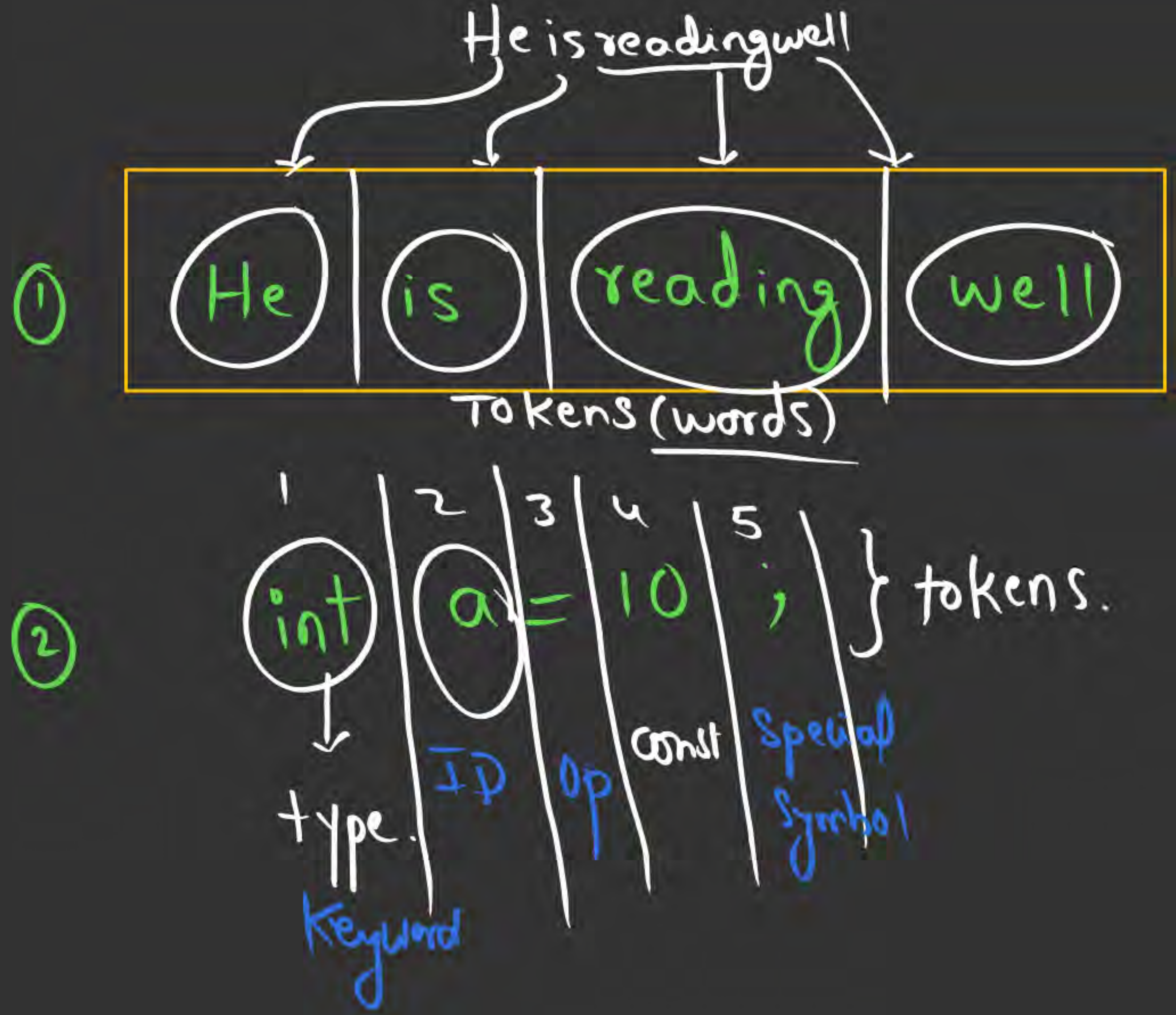
Contextfree Grammar

Topic : Lexical Analysis Phase

1 ➤ It is a program that takes high level language as input and produces tokens as output.

2 ➤ It also detects lexical errors present in the program.


Stream of Characters (input)
Lexical Analysis
tokens.
(words)

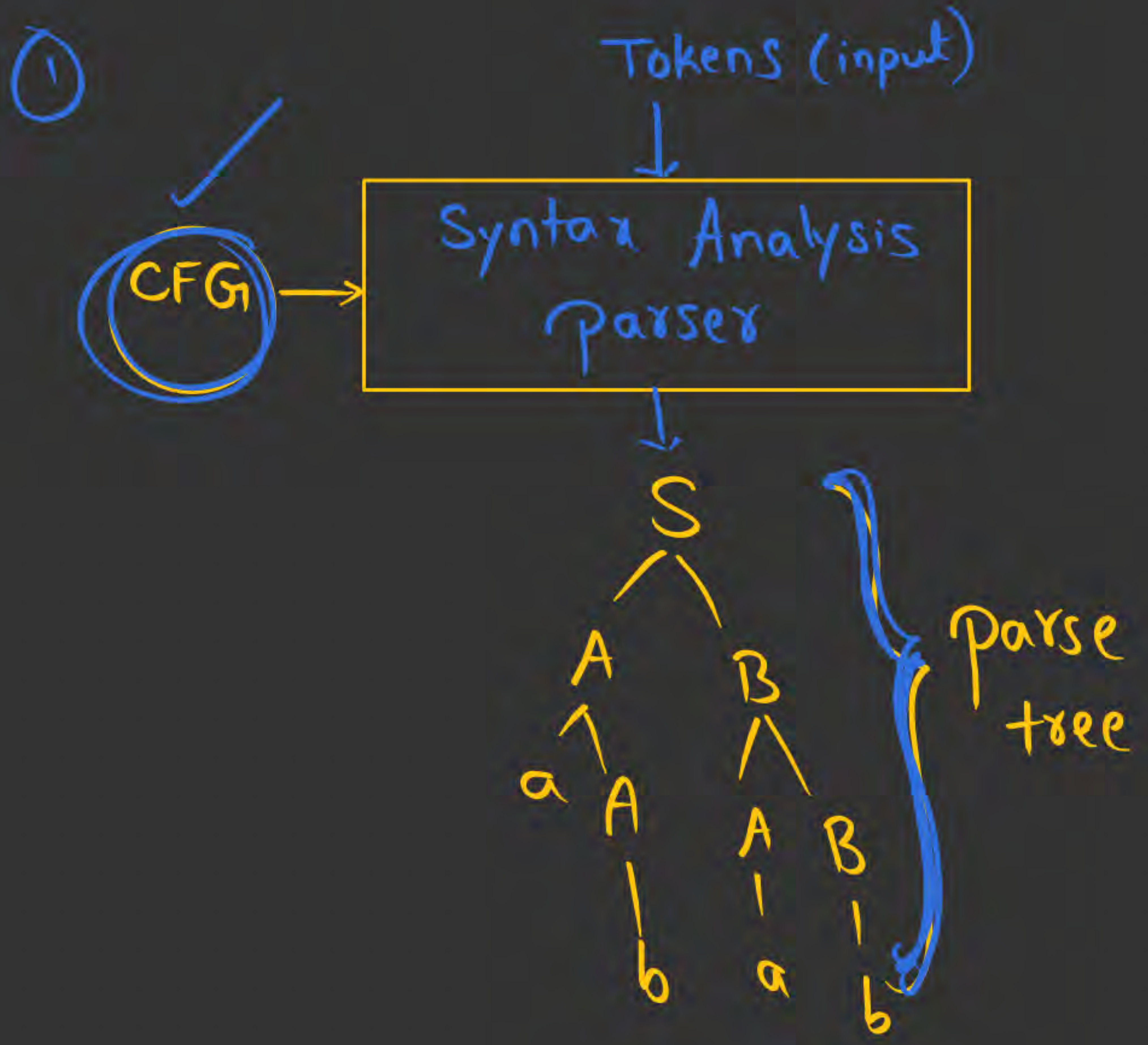

He is readingwell
①
He
is
reading
well
Tokens (words)
②
1
2
3
4
5
int
a
=
10
;
} tokens.
type.
Keyword
ID
Op
const
Special Symbol

Topic : Syntax Analysis Phase

translation

- It is a program that takes tokens as input and produces parse tree as output.
- It detects syntax errors present in the program.

Grammar checking is done by this phase


1
Tokens (input)
CFG
Syntax Analysis
Parser
S
A
B
a
A
A
B
b
a
b
Parse tree

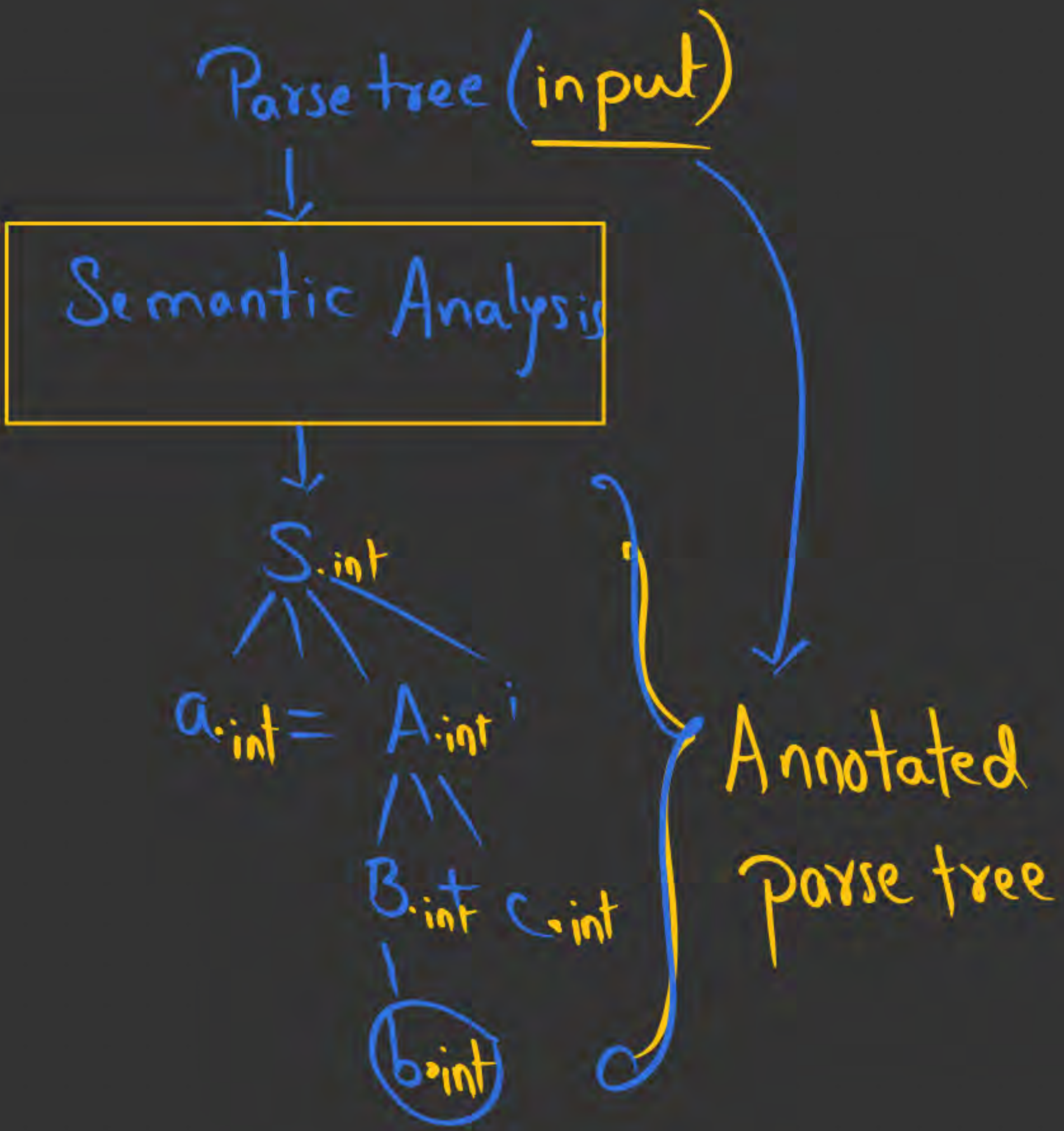
Topic : Semantic Analysis Phase

translation

1. It is a program that takes parse tree as input and produces annotated parse tree as output.
2. It also detects sematic errors present in the program.(type checking)

Meaning verification is done by this phase.

$$\boxed{a = \underset{\text{(int)}}{b} + \underset{\text{(string)}}{c}\ ;}$$ (Semantic error)


Parse tree (input)
Semantic Analysis
S.int
a.int
=
A.int
;
B.int
c.int
b.int
Annotated
parse tree

Topic : Intermediate Code Generation Phase

- It is a program that translates high level code into intermediate code.
- Advantage of generating intermediate code is to perform optimization in simple way.

Annotated Parse tree (input)

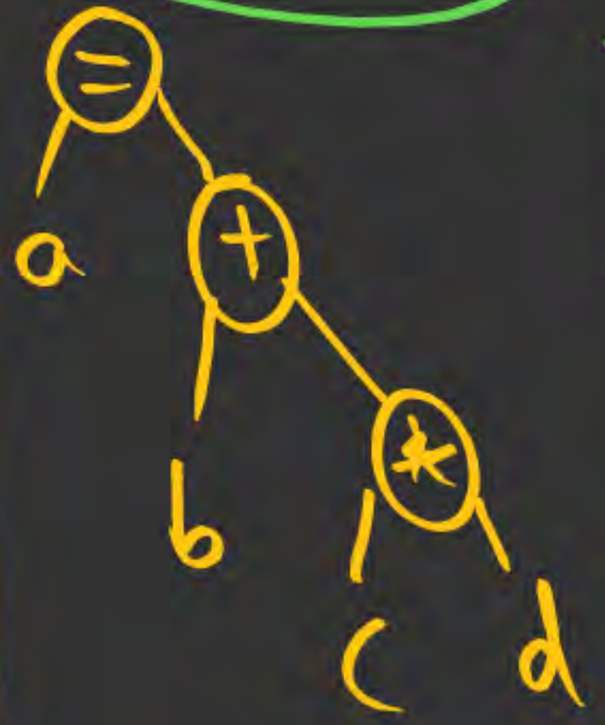
$$\begin{tikzcd} \text{Annotated Parse tree (input)} \arrow[d] \\ \boxed{\text{I. C. G}} \arrow[d] \\ \text{Three address Code} \end{tikzcd}$$

Three address Code (or) Syntax tree ($a = b + c * d$)

$t_1 = b + c$

$t_2 = a + t_1$

$t_4 = c * t_2$



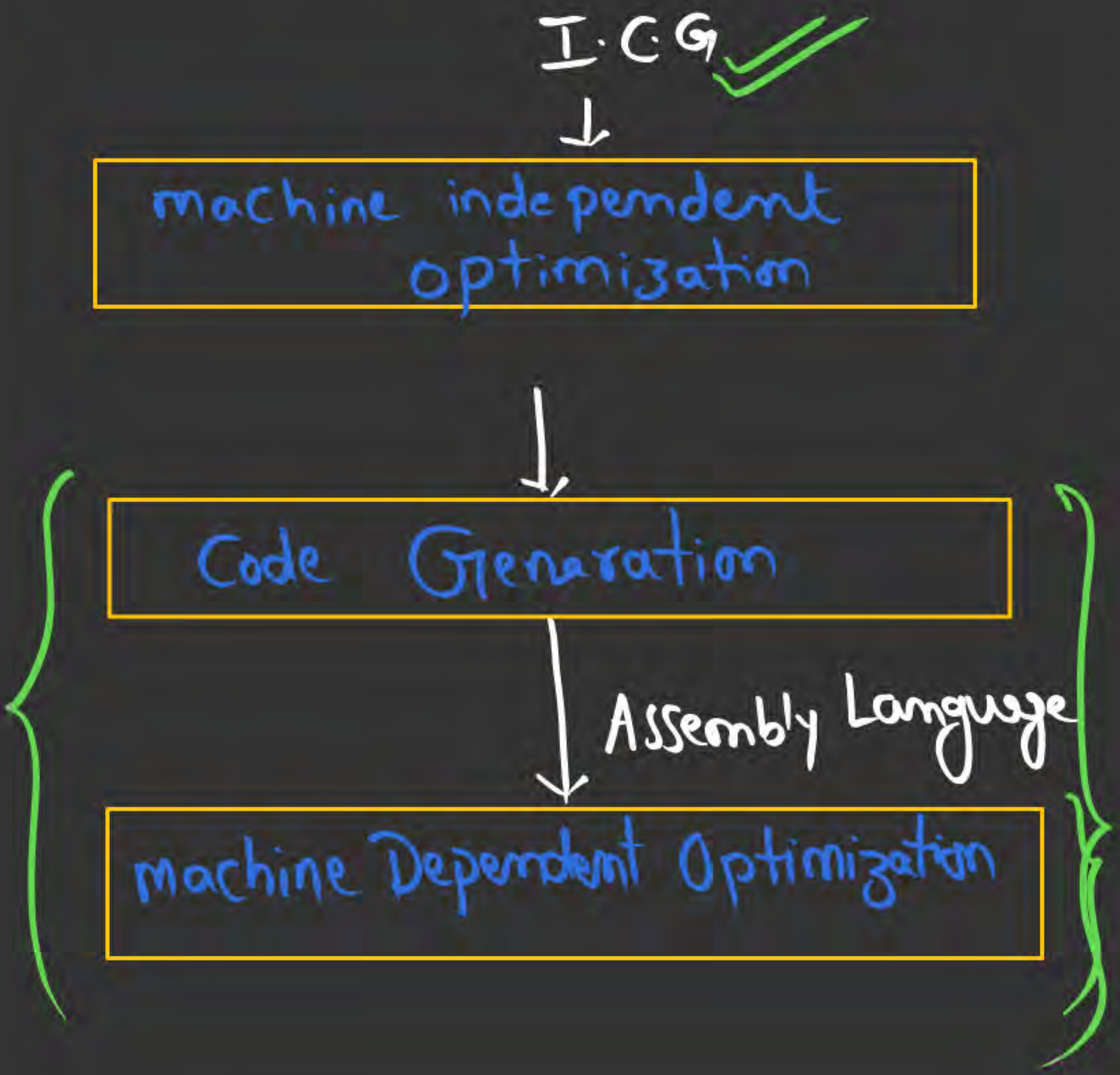
output

Topic : Code Optimization Phase

$a = 5 * 10;$

$\hookrightarrow a = 50;$ ✓

- It is a program that reduces time and space required by the target machine by removing some unnecessary code .
- There are two types of optimizations performed by compilers known as machine independent optimization and machine dependent optimization.


I.C.G
machine independent optimization
Code Generation
Assembly Language
machine Dependent Optimization

Topic : Code Generation Phase

- It is a program that translates optimized intermediate code into assembly language or target code.

$$\begin{tikzcd}
\text{Optimized I.C.G} \arrow[d] \\
\boxed{\text{Code Geranation}} \arrow[d] \\
\text{Assembly Language (low level Language)}
\end{tikzcd}$$

Topic : Symbol Table

int a = 10;

Symbol Table:

➤ It is data structure that contains information about identifiers and constants present in the program.

	type	Id	value
1	**int**	**a**	10
2	float	b	2.5
3	char	c	'a'

Topic

Error Handler:

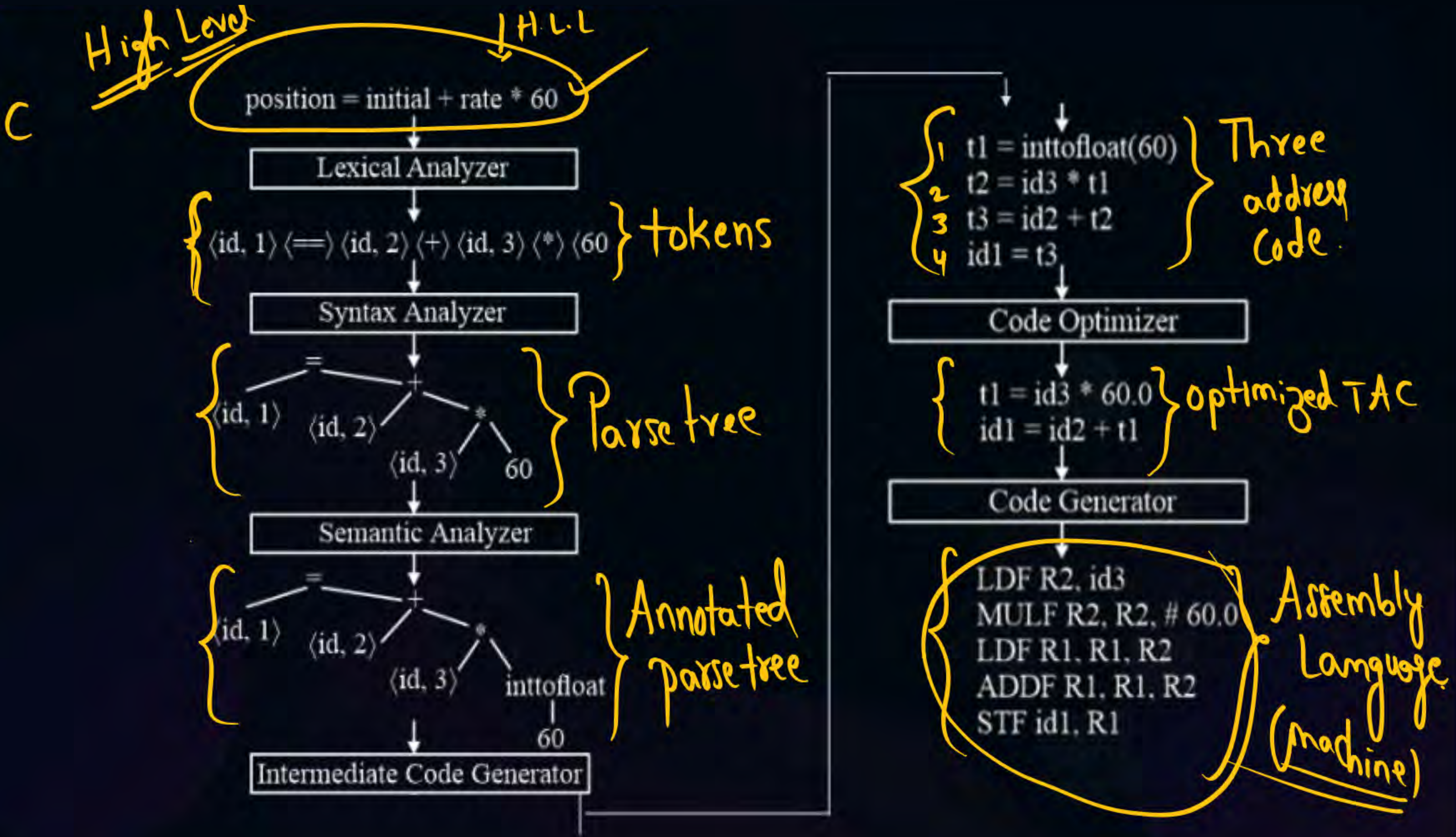
- If any phase of compiler detects error then it is stored in error handler.

Errors

1. Lexical errors ✓
2. Syntax errors ✓
3. Semantic errors ✓

Compiler detects only these 3 type of errors

4. Runtime errors → can't detected by Compiler



```
position = initial + rate * 60
```

```
⟨id, 1⟩ ⟨=⟩ ⟨id, 2⟩ ⟨+⟩ ⟨id, 3⟩ ⟨*⟩ ⟨60⟩
```

```
1 t1 = inttofloat(60)
2 t2 = id3 * t1
3 t3 = id2 + t2
4 id1 = t3
```

```
t1 = id3 * 60.0
id1 = id2 + t1
```

```
LDF R2, id3
MULF R2, R2, # 60.0
LDF R1, R1, R2
ADDF R1, R1, R2
STF id1, R1
```

Front end of Compiler

Phases of Compiler dependent on Source Language and independent on target (machine) Language

$$\left\{\begin{array}{l}\text{Lexical Analysis}\\ \text{Syntax Analysis}\\ \text{Semantic Analysis}\\ \text{Intermediate code Generation}\\ \text{m/c independent optimization.}\end{array}\right\}\underline{\underline{\text{Front end}}}$$

Back end of Compiler :-

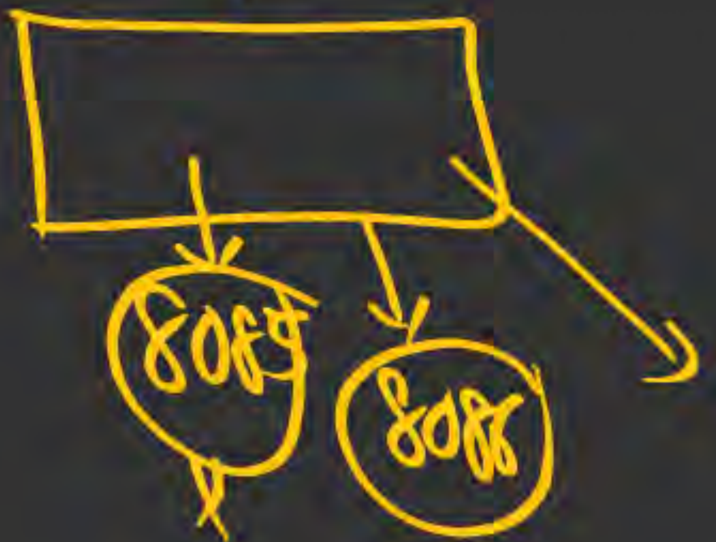
Phases of Compiler Depends on Target (machine) Language and independent on Source Language.

Backend

Code Genaration

machine dependent optimization

C ampiler



input	Phase of Compiler	output
Stream of characters	Lexical Analysis	Tokens
Tokens	Syntax Analysis	Parse tree
Parse tree	Semantic Analysis	Annotated P.T
Annotated P.T	Intermediate Code Genaration	TAC (or) Syntax tree
I.C.G	Code Optimization	Optimized I.C.G
opti I.C.G	Code Genation	Assembly Language

[MCQ]

#Q. Consider the following two sets:

o/p

	Set X			Set Y
P.	Lexical Analyzer $\longleftrightarrow$ tokens		1.	Abstract Syntax Tree
Q.	Syntax Analyzer $\longrightarrow$ P.T		2.	Token
R.	Intermediate Code Generator - 1		3.	Parse Tree
S.	Code Optimizer	$\longleftrightarrow$	4.	Constant Folding

Which one of the following options is the CORRECT match from Set X to Set Y ?

[GATE-CS-shift-II-24: 1M]

A P-4: Q-1: R-3 ; S-2

B P-2: Q-1: R-3 ; S-4

[✓ C P-2: Q-3: R-1 ; S-4]

D P-4: Q-3: R-2 ; S-1

[MCQ]

#Q. Consider the following two sets:

	Set X	o/p		Set Y
P.	Lexical Analyzer	$\leftrightarrow$ tokens	1.	Abstract Syntax Tree
Q.	Syntax Analyzer	$\rightarrow$ P.T	2.	Token
R.	Intermediate Code Generator	– 1	3.	Parse Tree
S.	Code Optimizer	$\longleftrightarrow$	4.	Constant Folding

Which one of the following options is the CORRECT match from Set X to Set Y ?

[GATE-CS-shift-II-24: 1M]

A. P-4: Q-1: R-3 ; S-2

B. P-2: Q-1: R-3 ; S-4

[✓ C. P-2: Q-3: R-1 ; S-4]

D. P-4: Q-3: R-2 ; S-1

#Q. Match the following according to input (from the left column) to the compiler phase (in the right column) that processes it:

List-I	List-II
(P) Parse tree	(i) Code generator
(Q) Character stream	(ii) Syntax analyzer
(R) Intermediate representation	(iii) Semantic analyzer
(S) Token stream	(iv) Lexical analyzer

A P → (ii), Q → (iii), R→ (iv), S → (i)

B P → (ii), Q → (i), R→ (iii), S → (iv)

[✓ C] P → (iii), Q→ (iv), R→ (i), S → (ii) ✓

D P → (i), Q → (iv), R→ (ii), S→ (iii)

#Q. In a compiler the module that checks every character of the source text is called

A The code generator

B The code optimizer

C The lexical analyzer

D The syntax analyzer

THANK - YOU