CS & IT ENGINEERING

Compiler Design

Lexical Analysis & Syntax Analysis

Lecture No. 2

By- DEVA Sir

- Phases of a Compiler
- Errors
- Lexical Analysis
  - Functionality
  - Types of tokens
  - Token Finding

H.W.

Data | Code | Statement | program | Language

C Language

C compiler

C Language → set of programs

```
Void main( )
{
    int x=10;
    Printf ("%d", x);
}
```

function

Data ✓

Code

Statement

Program

Language

function

Declaration

Program → set of statements

10 → data

x → variable

Statement → simple / compound

Code → Handles your data
[I/P or O/P]

Data → Requirement

C Language

I) High Level Code

II) Machine Independent Code

III) Portable

code can run on any machine

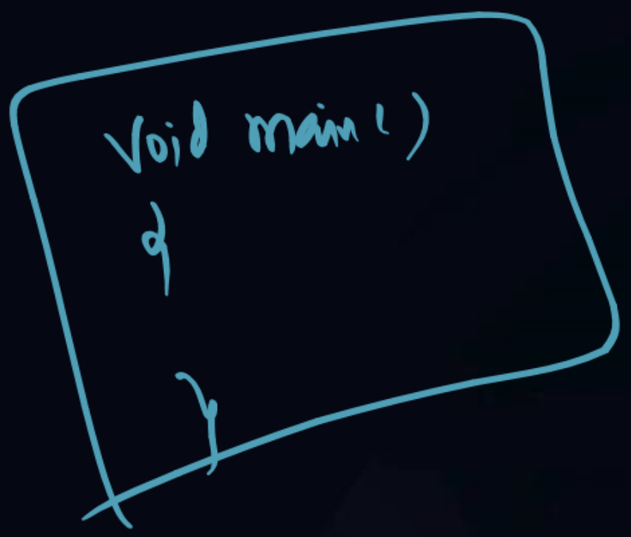
C compiler

I) Machine Code [Binary]
[executable]
[Low-level]

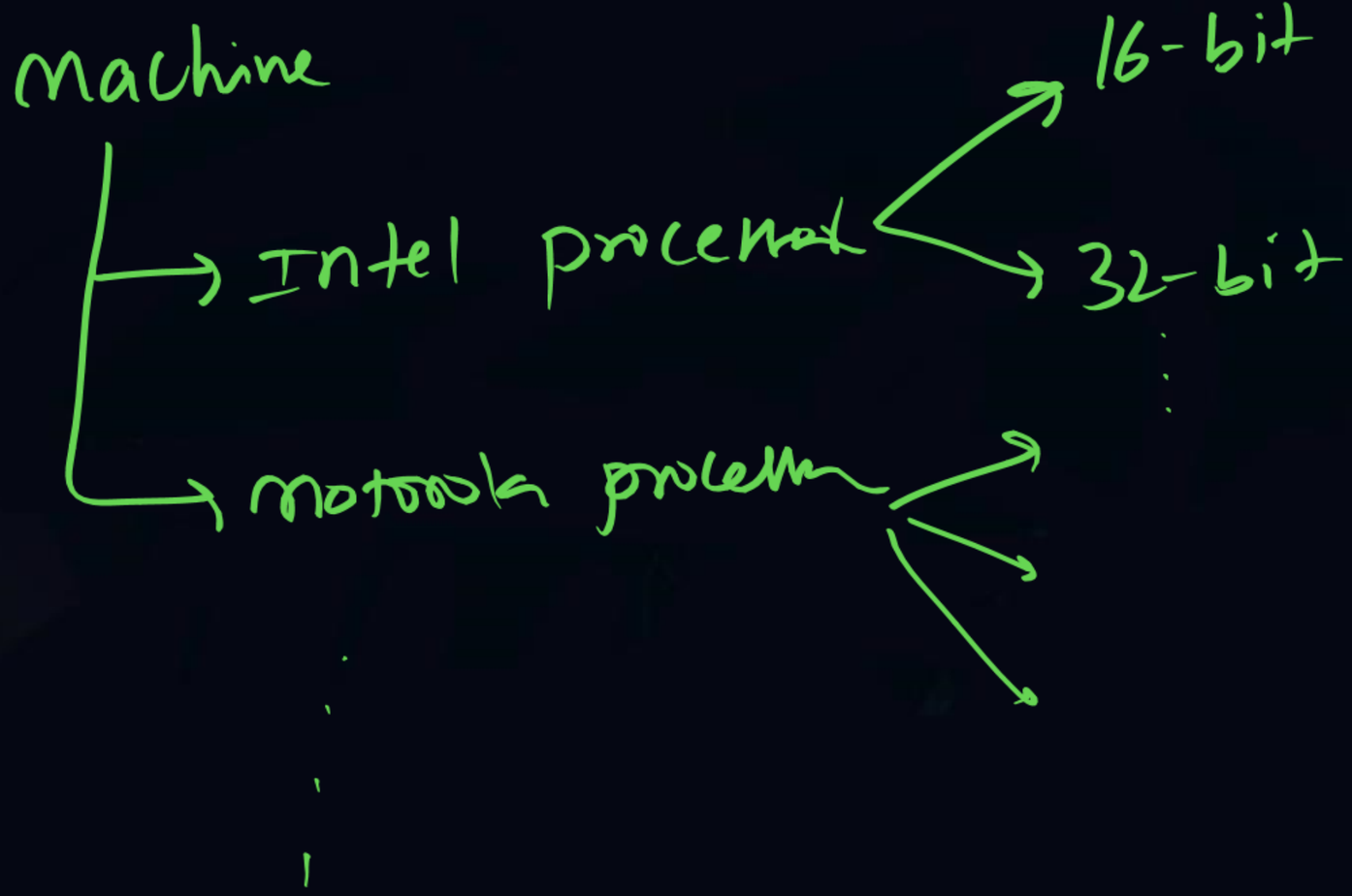
II) Machine dependent Code

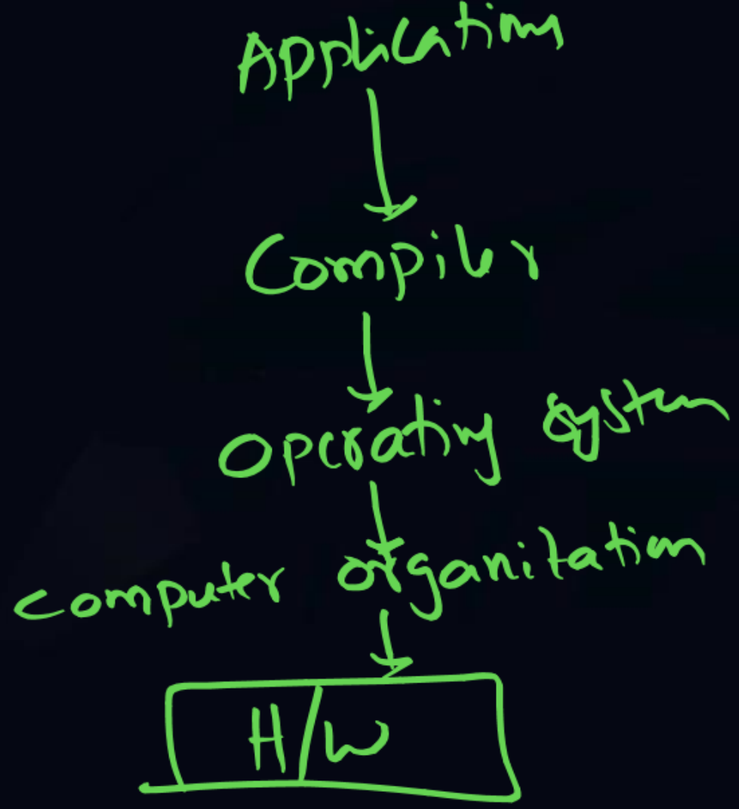
III) Not Portable

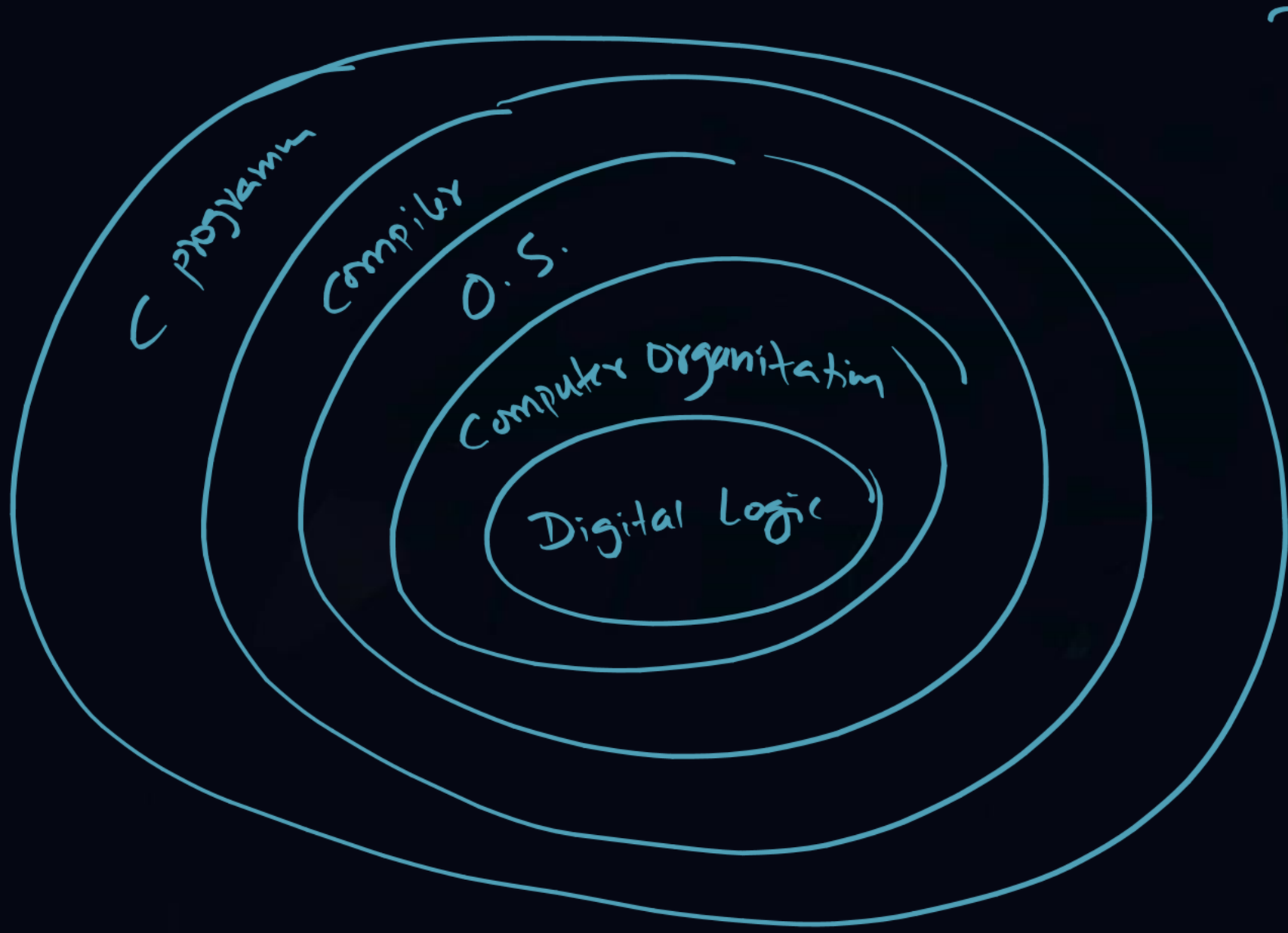
Code can run on specific machine

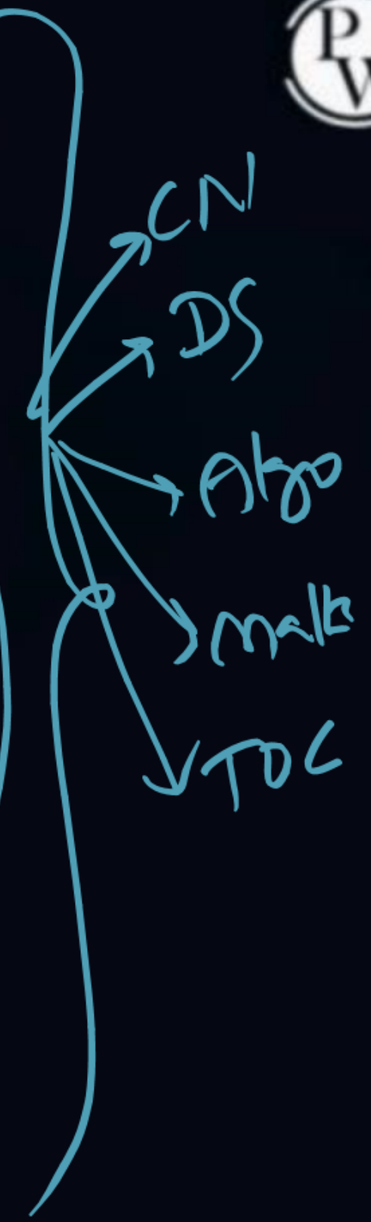


C language

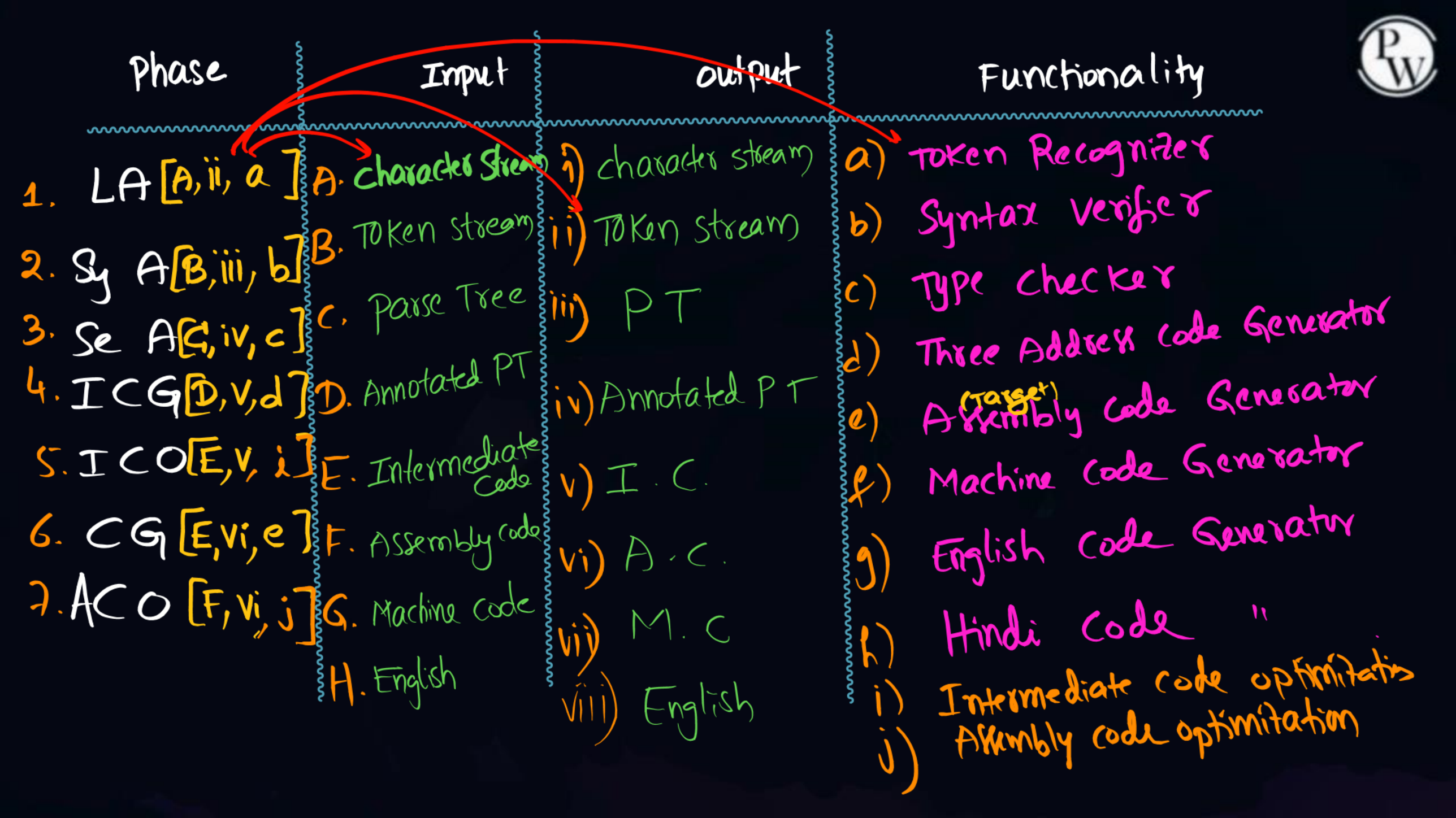

Machine
Intel processor
16-bit
32-bit
Motorola processor


Application
Compiler
Operating System
computer organization
H/W


C programs
Compiler
O.S.
Computer organisation
Digital Logic


CN
DS
Algo
Maths
TOC


LA
Sy A
Se A
ICG
ICO
CG
ACO


Phase
Input
output
Functionality
1. LA [A, ii, a]
2. Sy A[B, iii, b]
3. Se A[C, iv, c]
4. ICG[D, v, d]
5. ICO[E, v, i]
6. CG [E, vi, e]
7. ACO [F, vi, j]
A. Character Stream
B. Token stream
C. Parse Tree
D. Annotated PT
E. Intermediate Code
F. Assembly code
G. Machine code
H. English
i) character stream
ii) Token Stream
iii) PT
iv) Annotated PT
v) I.C.
vi) A.C.
vii) M.C
viii) English
a) Token Recognizer
b) Syntax Verifier
c) Type Checker
d) Three Address code Generator
e) Assembly (Target) code Generator
f) Machine Code Generator
g) English Code Generator
h) Hindi Code "
i) Intermediate code optimitation
j) Assembly code optimitation

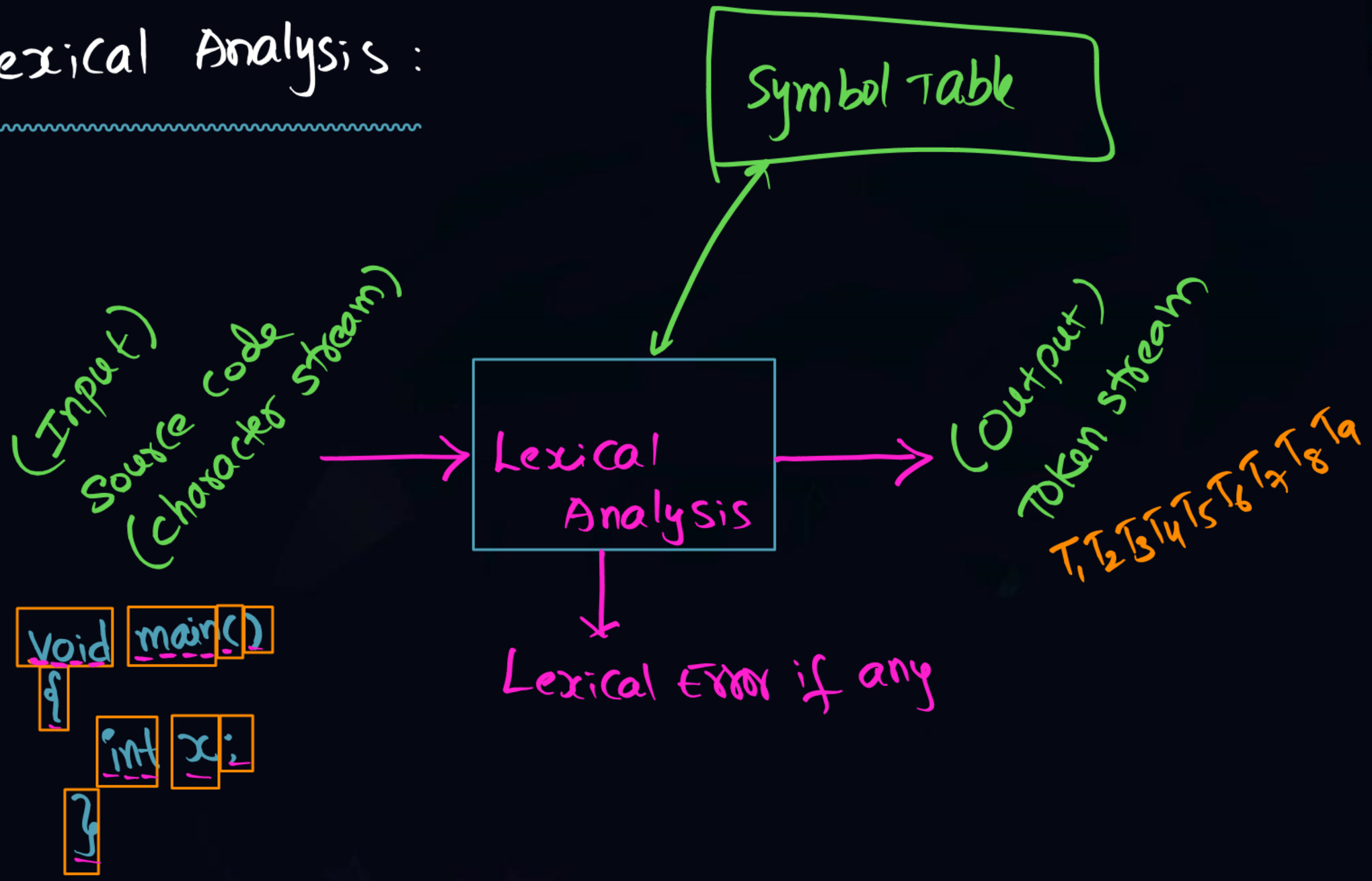
Introduction

- Translator ✓
- Compiler ✓
- Preprocessor ✓
- Assembler ✓
- Linker ✓
- Loader ✓
- phases ✓

Errors

- Preprocessor Error
  - #Conclude<...>
- Compiler Error
  - lexical
  - Syntax
  - Semantic
- Linker Error
- Runtime Error
  - Divide by zero
  - Array: out of bound access
  - pointer problems
  - Inf loop
  - Inf recursion
  - ...

Lexical Analysis :



Lexical Analysis

- Lexical phase
- Lexical Analyzer
- 1st phase of compiler
- Scanner
- Linear phase
- Token Recognizer
- Token Generator

Lexical Analysis:

I) It scans whole program character by character [Top to bottom, Left to Right]

II) It groups characters into tokens

III) It ignores white spaces and Comments

IV) It uses "longest Prefix Rule" [Maximal munch]

V) It also produces lexical errors if any

VI) It creates a symbol table and stores Some attribute information of Identifiers (Variable names, function names, ..)

VII) To design lexical analysis, Regular expression / FA / Regular Grammar can be used.

Compiler scans from ____

Compilation begins from 1st character

Execution begins from main()

A) First character of program ✓

B) From main()

C) A & B

D) None

I are student.
S + V + O

I am student

Lexical: I
am
Student

Syntax: S + V + O

Semantic: Meaning

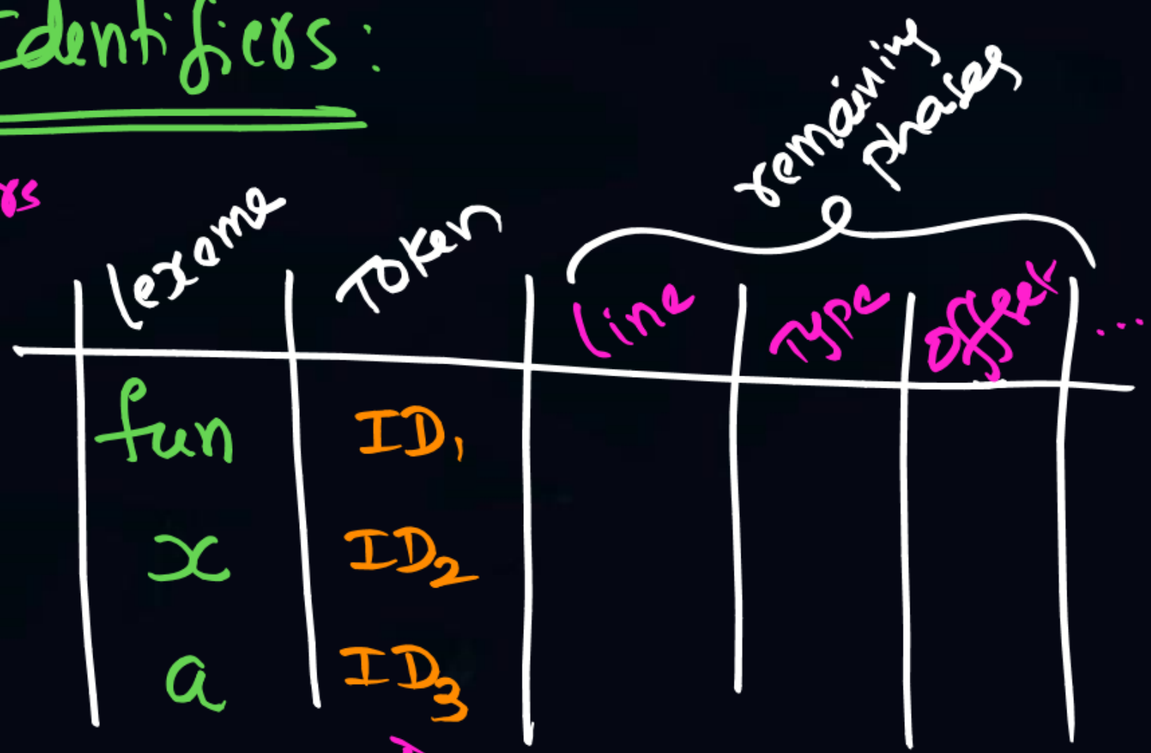
void main()
{
}

Symbol Table:

- It is a data structure
- It stores attribute information of identifiers
- It is accessible to all phases of Compiler

```
Void fun(int x)
{
    int a;
}
```

Identifiers:



Lexeme: Smallest unit of (logic) program

Token: It uniquely represents lexeme

Internal representation of lexeme

Lexical Analysis

- Lexeme & Token
- Types of Tokens
  - Keywords
  - Operators
  - Identifiers
  - Literals / constants
  - Special tokens (punctuations)
- How to find Tokens?
- What are Lexical Errors?

I) Keywords:

↳ 32 Keywords

① int, float, double, char, long, short, void, signed, unsigned, volatile, const, auto, static, extern, register

② if, else, switch, case, default, while, do, for, break, continue, return, goto

③ struct, union, enum, typedef

④ Sizeof()

II) operators

H.W.

III) Identifiers

H.W.

IV) Constants

H.W.

V) Special Tokens

H.W.

Q1) Is "main" keyword ?

Yes ×

NO ✓

- It is not a keyword
- It is userdefined function
- It is identifier

Q2) IS "exit()" Keyword?

Yes

NO

It is a function to terminate the program

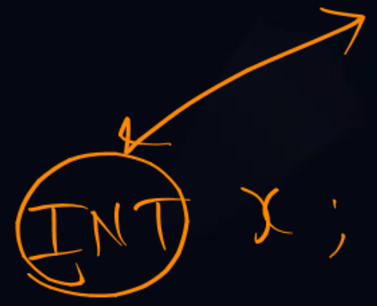
Q3) IS "sizeof()" Keyword ?

- Yes ✓
- NO

It is Keyword & operator

typedef int INT;

this is analyzed in Semantic Analysis



#define INT int

It is analyzed in preprocessing

INT x;

int x;

Summary

- LA
  - What is LA?
  - Functionality?

Next:
- How to find Tokens?
- Lexical Errors?


THANK YOU GW
SOLDIERS !