NAME : L. Surya pradeep
REG.No : 192210107

①
Lexical Analyzer :-

A lexical analyzer also known as a lexer (or) Scanner is the first phase of a compiler. It's Main task is to read the Source. Code written in a programming language and convert it into a sequence of tokens that the Subsequent phases of the Compiler can understand.

Here's a high level overview of how a lexical analyzer works :

★ Input Source Code :- The lexical analyzer takes the Source Code of program as input. The Source Code can be written in a programming language.

★ Scanning :- The lexical analyzer reads the source code character by character and groups them into tokens based on the language's grammar rules.

★ Token Generation :- When a pattern is matches a Substring of the Source Code the lexical analyzer generates a token token type and, if applicable, a value.

★ Ignoring Whitespaces And Comments : The lexical analyzer typically ignores white space (spaces, tabs, line breaks) and Comments while scanning the Source code since they don't contribute to the semantics.

★ Error Handling :- If the lexical analyzer encounters an invalid (or) unrecognized token, it raises an error and reports the issue to the compiler.

★ Token Stream :- As the lexical analyzer processes the Source code, it produces a stream of tokens, which is passed on the next phase of the compiler.

Semantic Analyzer :

The Semantic analyzer is the Second phase of a compiler, which comes after the lexical analyzer (lexer). It's Main task is to check the meaning and validity of the program according to the language's semantics and rules. The goal of the semantic analyzer is to ensure that the Source code has a Consistent and meaning ful structure.

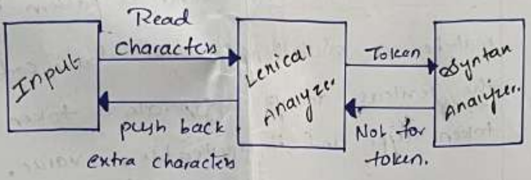
★ Input Token Stream : The Semantic analyzer takes the token stream generated by the lexical analyzer as its input. This token stream represents the - Syntactic structure of the Source Code.

Syntatic Analyzer:

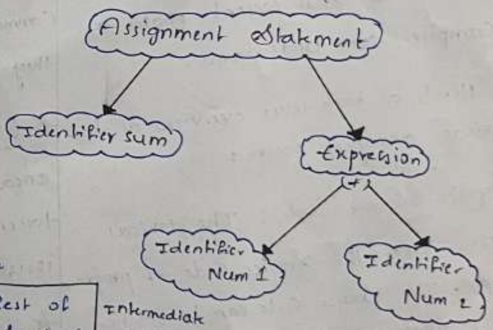
A Syntatic analyzer, also known as a parser, is a crucial Component of a Compiler. It's primary Function is to analyze the syntatic Structure of the Source Code and ensure that it adheres to the rules defined by the programming language's grammar. The parser takes the stream of tokens generated by the lexical analyzer and constructs a parse tree or an abstract syntax tree (AST) to represent the program's Syntactic structure.

By the end of the Syntactic analysis phase, the compiler has a well-structured representation of the programs syntax, which forms the basis for subsequent phases like semantic analysis, optimization and code generation.

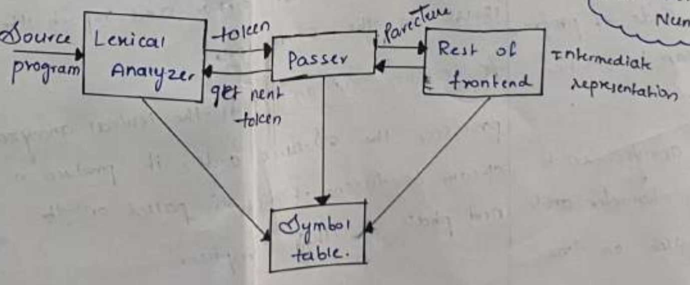
Lexical Analyzer:



Semantic Analyzer:



Syntatic Analyzer:


Source
program
Lexical
Analyzer
token
get next
token
Passer
parsetree
Rest of
frontend
Intermediate
representation
Symbol
table.