MCA 18 302
PRINCIPLES OF COMPILERS

MODULE 1

1. **Introduction to compiling**
    1. Definition of compiler, translator, interpreter
    2. Analysis of the source program
    3. The phases of a compiler
    4. Compiler construction tools
2. **Programming language basics**
3. **Lexical analysis**
    1. Role of lexical analyzer
    2. Input buffering
    3. Specification of tokens
    4. Recognition of tokens using finite automata
    5. Regular expressions and finite automata
    6. From NFA to DFA
    7. Regular expression to an NFA

Introduction to compiling

Compiler construction tools

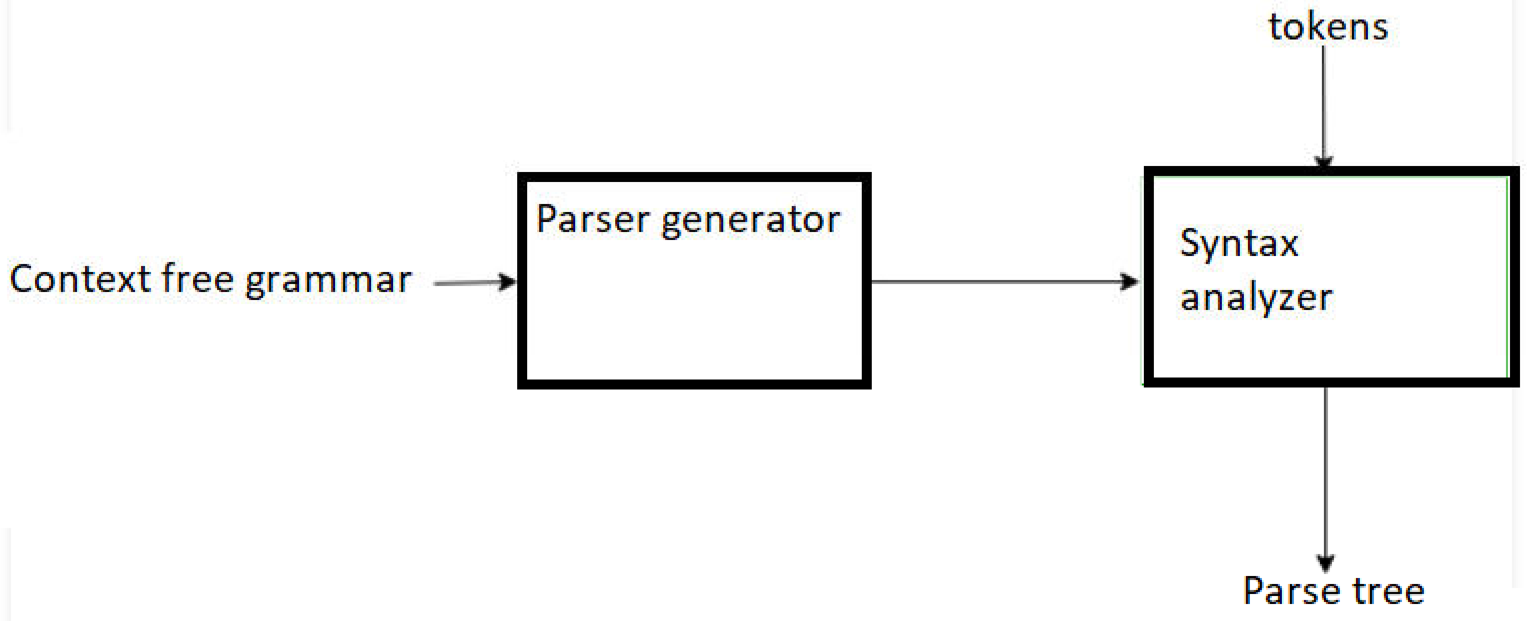
- The compiler construction tools are:
  1. Parser generator
  2. Scanner generator
  3. Syntax directed translation engines
  4. Automatic code generators
  5. Data flow Engines
  6. Compiler construction toolkit

1. **Parser generator:**

❖ It produces syntax analyzers(parsers) from the input that is based on a grammatical description of programming language or on a context free grammar.

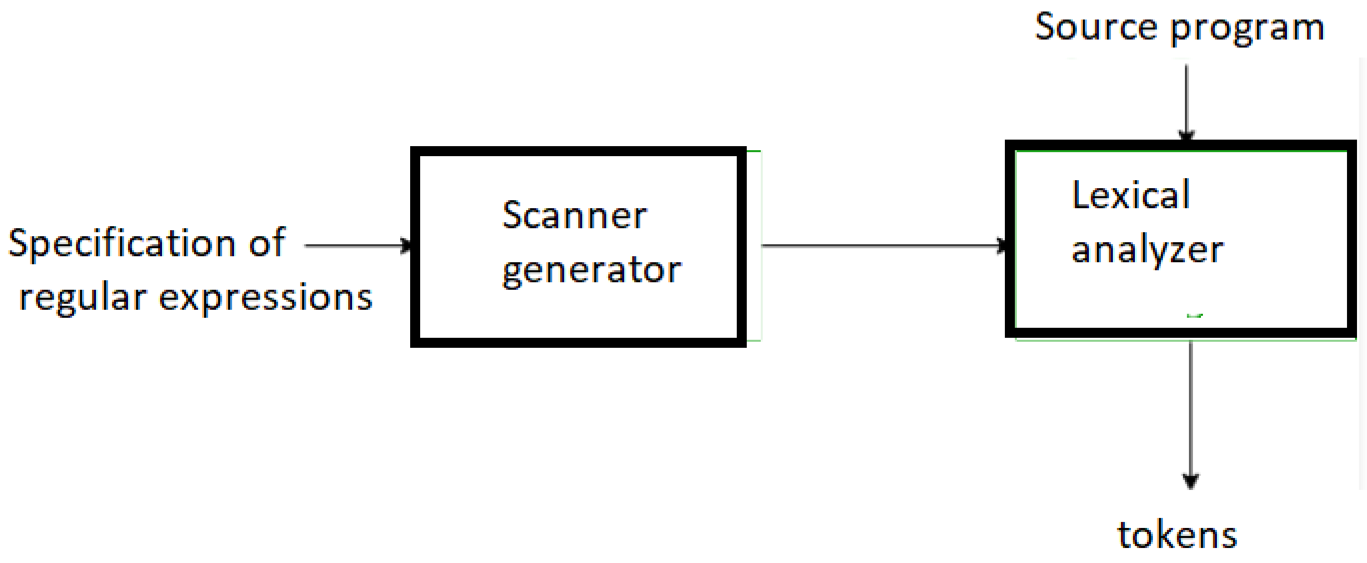
❖ It is useful as the syntax analysis phase is highly complex and consumes more manual and compilation time.

❖ Example:PIC,EQM


tokens
Context free grammar
Parser generator
Syntax analyzer
Parse tree

2. Scanner generator:

- ❖ It generates lexical analyzers from the input that consists of regular expression description based on tokens of a language.
- ❖ It generates a FA to recognize the regular expression.
- ❖ Example:Lex


Specification of regular expressions
Scanner generator
Source program
Lexical analyzer
tokens

3. Syntax directed translation engines:

- ❖ It generates intermediate code with three address format from the input that consists of a parse tree.
- ❖ These engines have routines to traverse the parse tree and then produce the intermediate code. In this, each node of the parse tree is associated with one or more translations.

4. Automatic code generators:

- ❖ It generates the machine language for a target machine. Each operation of the intermediate language is translated using a collection of rules and then is taken as an input by the code generator.
- ❖ Template matching process is used.
- ❖ An intermediate language statement is replaced by its equivalent machine language statement using templates.

5. **Data flow analysis engines:**
   - ❖ It is used in code optimization. Data flow analysis is a key part of the code optimization that gathers the information, that is the values that flow from one part of a program to another.
6. **Compiler construction tool kits:**
   - ❖ It provides an integrated set of routines that aids in building compiler components or in the construction of various phase of compiler.