Compiler Design

Lecture-12

Syntactic Analysis Parsing

Topics Covered

Compilers

Parsing

Compilers

- A Compiler is a program that reads a program written in one language (the *source language*) and translates it into another (the *target language*)
- A compiler operates in phases, each of which transforms the source program from one representation to the other

Source program → Lexical Analyzer → Syntax Analyzer → Semantic Analyzer → Intermediate Code Generator → Code Optimizer → Code Generator → *Target Program*

- The part of the compiler we will focus on in this part of the course is the Syntax Analyzer or Parser.

Parsing

- Parsing is the process of determining whether a string of tokens can be generated by a grammar.
- Most parsing methods fall into one of two classes, called the *top-down* and *bottom-up* methods.
- In top-down parsing, construction starts at the root and proceeds to the leaves. In bottom-up parsing, construction starts at the leaves and proceeds towards the root.
- Efficient top-down parsers are easy to build by hand.
- Bottom-up parsing, however, can handle a larger class of grammars. They are not as easy to build, but tools for generating them directly from a grammar are available.

Part I
Top Down Parsing

- Basic Ideas behind Top-Down Parsing
- Predictive Parsers
  - Left Recursive Grammars
  - Left Factoring a grammar
  - Constructing a Predictive Parser
- LL(1) Grammars

Basic Idea behind Top-Down Parsing

- Top-Down Parsing is an attempt to find a left-most derivation for an input string
- Example:

S → cAd	Find a derivation for
A → ab | a	for w → cad