Compiler Design Lab

Group - 08

September 19, 2023

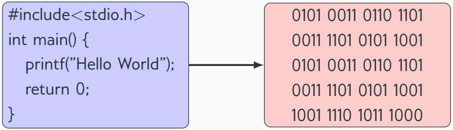
Group Members

- **Md. Masud Mazumder** (19701070)
- **Tonmoy Chandro Das** (19701066)
- **Tareq Rahman Likhon Khan** (19701068)
- **Arif Hasan** (19701060)
- **Md. Mustak Ahmed** (19701058)
- **Md. Siam** (19701075)
- **Abdullah Al Faruque** (19701052)
- **Imtiaz Ahmed Imon** (19701054)
- **Hasan Miah** (19701059)
- **Abu Noman Shawn Shikdar** (19701074)
- **Rabbi Hasan** (19701071)
- **Abdullah Siddique Mohammad Sayeed** (19701061)

Introduction

Compiler

A special program that translates high-level programming language's source code into equivalent machine code or bytecode.



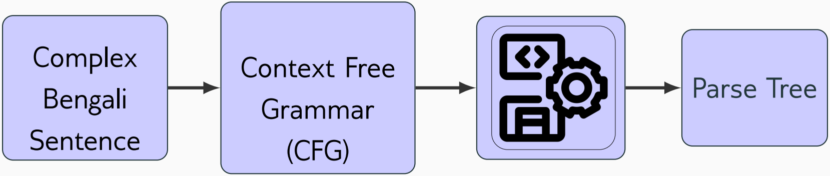
```
#include<stdio.h>
int main() {
   printf("Hello World");
   return 0;
}
```

```
0101 0011 0110 1101
0011 1101 0101 1001
0101 0011 0110 1101
0011 1101 0101 1001
1001 1110 1011 1000
```

Figure 1: Compiler Overview

- Translation from High-Level to Low-Level Code
- Error Detection
- Code Optimization
- Multiple Phases
- Platform Independence

Task Description



Complex Bengali Sentence

Context Free Grammar (CFG)

Parse Tree

Overview

- i Write CFG for Complex Bengali Sentence
- ii Generate Recursive Descending Parser
- iii Syntax Analysis using ANTLR4
- iv Syntax Analysis using BISON
- v Compare ANTLR4 and BISON

Phases of a Compiler

The compilation process is divided into phases. Each phase receives input from the previous phase, has its own representation of the source program, and outputs to the next phase of the compiler.

Phases of Compiler

- i Lexical Analysis
- ii Syntax Analysis
- iii Semantic Analysis
- iv Intermediate Code Generation
- v Code Optimization
- vi Code Generation

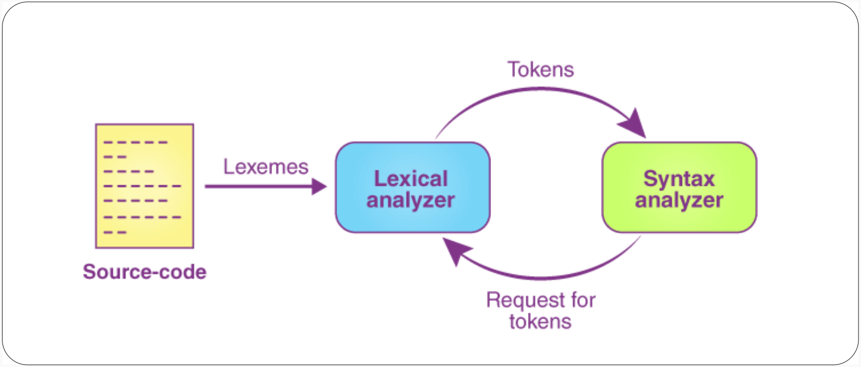
Lexical Analysis

Lexical analysis

Lexical analysis, also known as scanning, is the initial phase of the compiler where the source code is analyzed to identify and categorize lexical units or tokens such as keywords, identifiers, constants, and operators. This process utilizes regular expressions to define the patterns for recognizing these tokens.

- First phase of the compiler, analyzing source code.
- Identifies basic units (tokens) like identifiers, keywords, operators.
- Uses regular expressions to define tokens.
- Removes whitespace and comments.
- Outputs a token stream for further processing.

Lexical Analysis



Syntax Analysis

Syntax analysis

Syntax analysis, also known as parsing, is the second phase of the compiler. It verifies whether the tokens produced by the lexical analysis adhere to the syntax rules of the programming language, following a specified grammar.

- Matches the token stream against the grammar rules.
- Constructs a parse tree or syntax tree.
- Helps in identifying syntax errors in the source code.
- Provides a structured representation of the program's syntax.

ANTLR4 as a Syntax Analyzer

Overview

ANTLR (ANother Tool for Language Recognition) is a powerful parser generator that assists in constructing syntax analyzers or parsers. It employs LL(*) parsing algorithm, providing a robust tool for processing structured languages based on a specified grammar.

- Generates parsers based on a specified grammar.
- Utilizes LL(*) parsing algorithm for efficient parsing.
- Creates parse trees facilitating further analysis.
- Supports a wide range of programming languages and platforms.
  - Java, C++, C#, Python, PHP, GO, Javascript, Dart, Swift

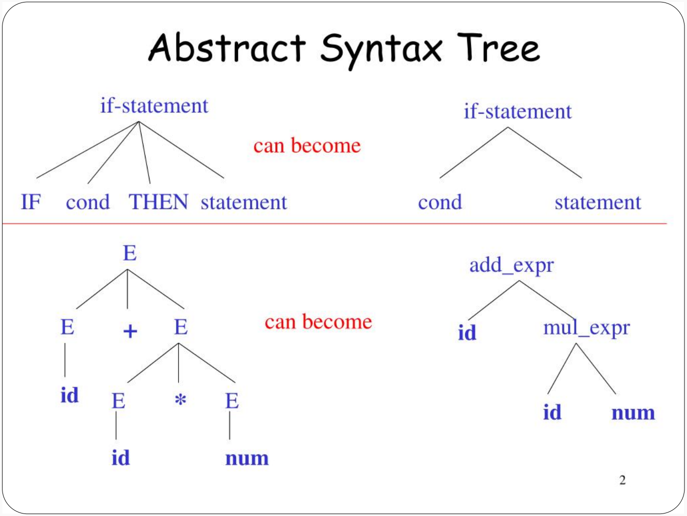
Bison as a Syntax Analyzer

Overview

BISON is a parser generator, often used with Lex, that creates syntax analyzers for compilers and interpreters and employing LALR(1) parsing algorithm.

- Bison reads a specification in the BNF notation, warns about any parsing ambiguities
- Constructs Abstract Syntax Trees (ASTs) for further analysis.
- Widely used in the development of compilers and interpreters.
  - Used by the GNU Compiler Collection for C, C++, the C preprocessor.
- Supports multiple target programming languages.

Abstract Syntax Tree



Abstract Syntax Tree

can become

can become

Comparison Between ANTLR4 and BISON - 1/2

Feature	ANTLR4	BISON
Language Support	Multiple languages like C, C++, Java, Python, etc	BISON primarily used for C/C++
Performance	Slower than BISON	Faster but may struggle with more complex grammar.
Toolchain	ANTLR4 includes lexer and parser generators	BISON requires a separate lexer like Flex

Comparison Between ANTLR4 and BISON - 2/2

Feature	ANTLR4	BISON
Error Handling	ANTLR4 provides detailed error messages	BISON provides low-level error reporting
Grammar Definition	ANTLR4 uses EBNF notation	BISON uses custom grammar rules
Real Life Usages	Twitter Search Query Parsing, Netbeans IDE C++ Parsing	GNU Compiler Collection for C, C++, the C preprocessor

Questions and Discussions

Thank You