BasicCompiler Documentation

Class Overview

The BasicCompiler class is a simple compiler implementation that processes a subset of C-like code. It performs lexical analysis, syntax analysis, semantic analysis, intermediate representation (IR) generation, optimization, code generation, and assembly.

Method	Description
__init__	Initializes the compiler
lexical_analysis	Breaks down source code into tokens
syntax_analysis	Builds an Abstract Syntax Tree (AST) from tokens
semantic_analysis	Performs type checking and ensures variables are declared
generate_ir	Converts AST to intermediate representation
optimize_ir	Placeholder for IR optimizations
code_generation	Generates x86 assembly code from IR
assemble	Converts assembly to simplified machine code
compile	Coordinates all steps of the compilation process

Compilation Process

• Lexical Analysis
• Syntax Analysis
• Semantic Analysis
• IR Generation
• IR Optimization
• Code Generation
• Assembly

Limitations

• Supports only a small subset of C-like syntax
• Limited to integer operations
• No support for control structures (if, loops, etc.)
• No function definitions or calls
• Simplified register allocation and variable management
• No actual machine code generation (uses a simplified representation)

Usage Example

```
compiler = BasicCompiler() source_code = """ int a = 5 + 3; int b = a * 2; """
machine_code = compiler.compile(source_code) print("Generated Machine Code:\n",
machine_code)
```