Compiler Design

An Introduction

Samit Biswas[1]

[1] Department of Computer Science and Technology,
Indian Institute of Engineering Science and Technology, Shibpur
Email: samit@cs.iiests.ac.in

Table of Contents

1. Pre-requisite courses
2. Introduction To Compiler Design
3. The Phases of a Compiler
4. Compiler Construction Tools

IIEST, Shibpur
आई आई ई एस टि, शिवपुर
Erstwhile B E College (Estd 1856)

Pre-requisite courses

- Strong **programming** background in C, C++ or Java

Pre-requisite courses

- Strong **programming** background in C, C++ or Java
- Formal Language and Automata Theory (REs, NFAs, DFAs, CFG, PDA).

Pre-requisite courses

- Strong **programming** background in C, C++ or Java
- Formal Language and Automata Theory (REs, NFAs, DFAs, CFG, PDA).
- Assembly Language Programming and Machine Architecture.

Table of Contents

1. Pre-requisite courses
2. Introduction To Compiler Design
3. The Phases of a Compiler
4. Compiler Construction Tools

Introduction To Compiler Design

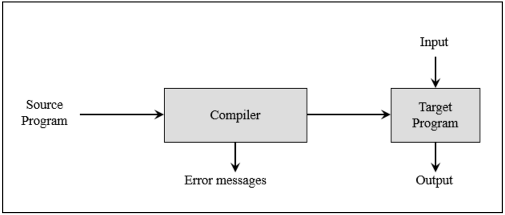
- **What is a compiler?**

Introduction To Compiler Design

- **What is a compiler?**
  - Before a program can be run, it first must be translated into a form (Target Program), which can be executed by a computer.

Introduction To Compiler Design

- **What is a compiler?**
  - Before a program can be run, it first must be translated into a form (Target Program), which can be executed by a computer.
  - **Translation** of a program written in a source language into a semantically equivalent program written in a target language



Introduction To Compiler Design

- A compiler is a program translates (or compiles) a program written in a **high-level** programming language (the source language) that is suitable for human programmers into the **low-level** machine language (target language) that is required by computers.

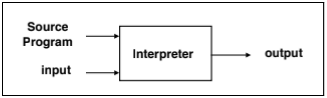
Introduction To Compiler Design

- A compiler is a program translates (or compiles) a program written in a **high-level** programming language (the source language) that is suitable for human programmers into the **low-level** machine language (target language) that is required by computers.
- During this process, the compiler will also attempt to spot and report obvious programmer mistakes that detect during the translation process.

IIEST, Shibpur
आई आई ई एस टि, शिवपुर
Erstwhile B E College (Estd 1856)

Introduction To Compiler Design

- **Interpreter:** An interpreter is another common kind of language processor. Instead of producing a target program as a translation, an interpreter appears to directly execute the operations specified in the source program on inputs supplied by the user.



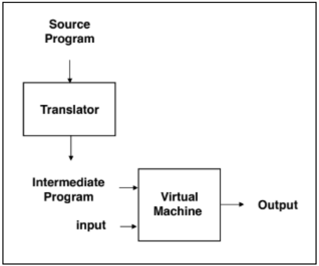
Compiler vs. Interpreter

- **Compiler vs. Interpreter**

Compiler	**Interpreter**
Takes the entire program as input	Take single instruction as input
It is Faster	It is Slower
Intermediate object code is generated.	no intermediate code is generated;
Errors are displayed after the entire program has been checked.	Errors are displayed for every instruction interpreted.
Ex: C, C++.	Ex: python, Ruby, basic.

Introduction To Compiler Design

- **Bytecode & Virtual Machine**
  A Java source program may first be compiled into an intermediate form called bytecodes. The bytecodes are then interpreted by a virtual machine. A benefit of this arrangement is that bytecodes compiled on one machine can be interpreted on another machine, perhaps across a network.

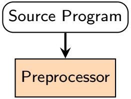


A typical Language Processing System

A typical Language Processing System

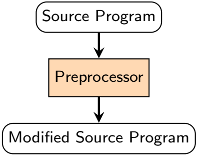
Source Program

A typical Language Processing System



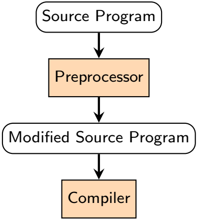
A typical Language Processing System

A typical Language Processing System



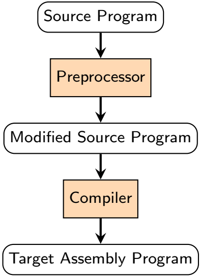
A typical Language Processing System

A typical Language Processing System



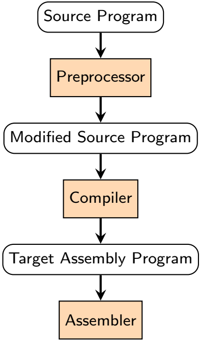
A typical Language Processing System

A typical Language Processing System



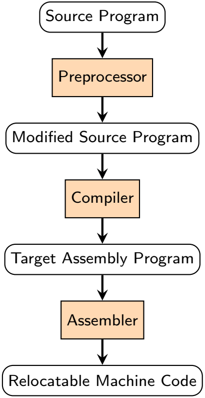
A typical Language Processing System

A typical Language Processing System



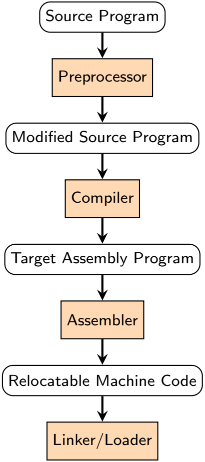
A typical Language Processing System

A typical Language Processing System



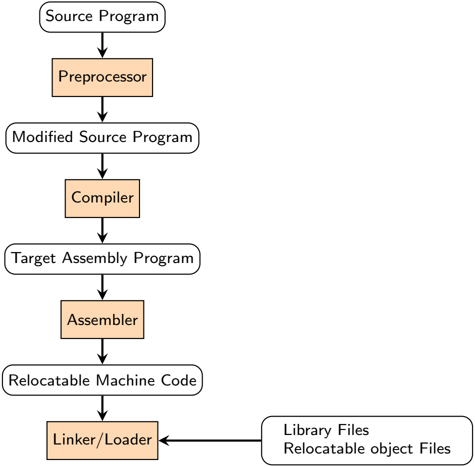
A typical Language Processing System

A typical Language Processing System



A typical Language Processing System

A typical Language Processing System



Source Program

Preprocessor

Modified Source Program

Compiler

Target Assembly Program

Assembler

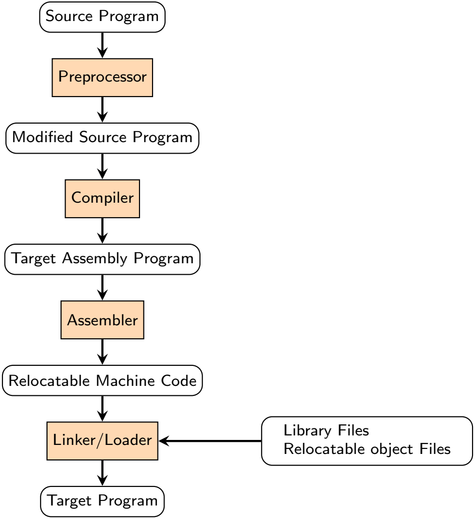
Relocatable Machine Code

Linker/Loader

Library Files
Relocatable object Files

A typical Language Processing System

A typical Language Processing System



Issues related to Compiler Design

Issues related to Compiler Design:

- Correctness

Issues related to Compiler Design

Issues related to Compiler Design:

- Correctness
- Speed (runtime and compile time)
  - Degrees of optimization
  - Multiple passes

Issues related to Compiler Design

Issues related to Compiler Design:

- Correctness
- Speed (runtime and compile time)
  - Degrees of optimization
  - Multiple passes
- Space

Issues related to Compiler Design

Issues related to Compiler Design:

- Correctness
- Speed (runtime and compile time)
  - Degrees of optimization
  - Multiple passes
- Space
- Feedback to user

Issues related to Compiler Design

Issues related to Compiler Design:

- Correctness
- Speed (runtime and compile time)
  - Degrees of optimization
  - Multiple passes
- Space
- Feedback to user
- Debugging

Table of Contents

1 Pre-requisite courses

2 Introduction To Compiler Design

3 The Phases of a Compiler

4 Compiler Construction Tools

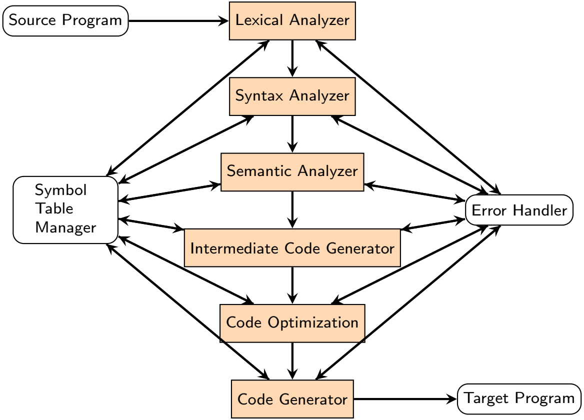
Analysis - Synthesis Model of Compilation

There are two parts of compilation:-

- **Analysis** - It breaks up the source program into constituent pieces and creates an intermediate representation of the source program. If the analysis part detects that the source program is either syntactically ill formed or semantically unsound, then it must provide informative messages, so the user can take corrective action.
- **Synthesis** - It constructs the desired target program from the intermediate representation and the information in the symbol table.

The Phases of a Compiler

- Conceptually, a compiler operates in phases, each of which translates the source program from one representation to another.



IIEST, Shibpur
आई आई ई एस टि, शिवपुर
Erstwhile B E College (Estd 1856)

Translation of an Assignment Statement

Translation of an Assignment Statement:

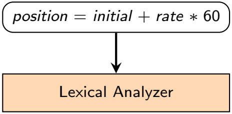
- Lexical Analyser takes the source program as input and produces a long string of tokens.
- Syntax Analyser takes an out of lexical analyser and produce a large tree.
- Semantic Analyser takes an output of Syntax analyser and produces another tree.
- Similarly Intermediate code generator takes a tree as an input produced by Semantic analyser and produces Intermediate code.

The Phases of a Compiler

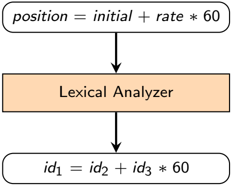
$\boxed{position = initial + rate * 60}$

IIEST, Shibpur
आई आई ई एस टि, शिवपुर
Erstwhile B E College (Estd 1856)

The Phases of a Compiler



The Phases of a Compiler

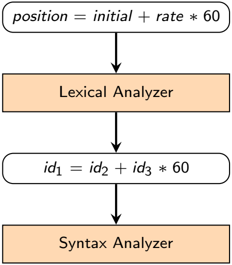


$position = initial + rate * 60$

Lexical Analyzer

$id_1 = id_2 + id_3 * 60$

The Phases of a Compiler



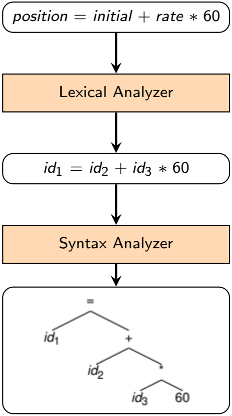
$position = initial + rate * 60$

Lexical Analyzer

$id_1 = id_2 + id_3 * 60$

Syntax Analyzer

The Phases of a Compiler



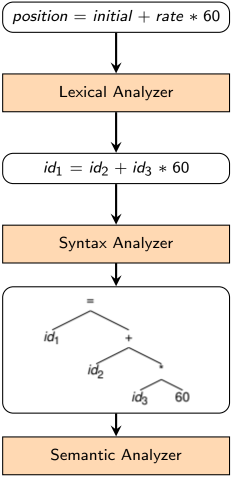
$position = initial + rate * 60$

Lexical Analyzer

$id_1 = id_2 + id_3 * 60$

Syntax Analyzer

The Phases of a Compiler



$position = initial + rate * 60$

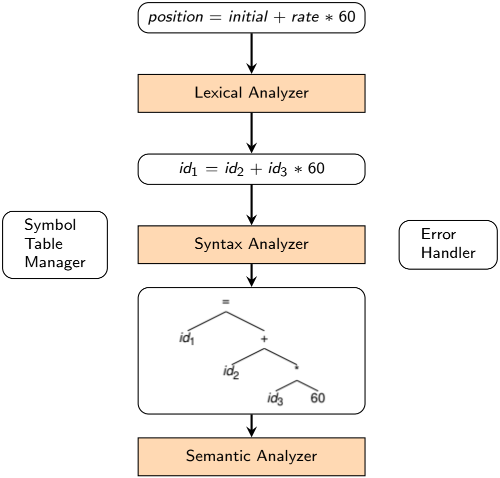
Lexical Analyzer

$id_1 = id_2 + id_3 * 60$

Syntax Analyzer

Semantic Analyzer

The Phases of a Compiler



$position = initial + rate * 60$

Lexical Analyzer

$id_1 = id_2 + id_3 * 60$

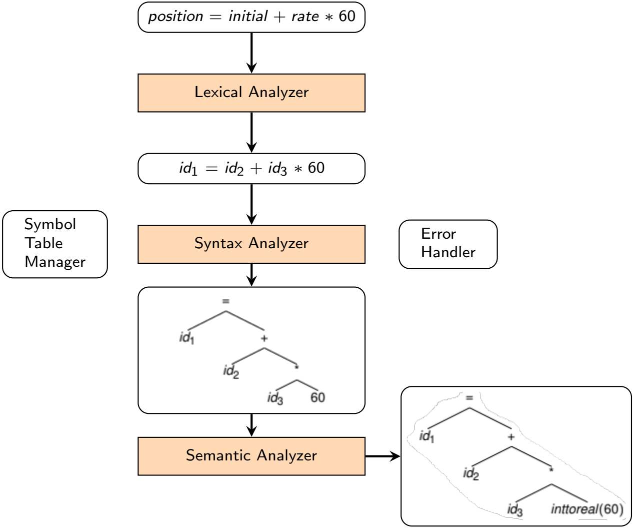
Syntax Analyzer

Semantic Analyzer

Symbol Table Manager

Error Handler

The Phases of a Compiler



$position = initial + rate * 60$

Lexical Analyzer

$id_1 = id_2 + id_3 * 60$

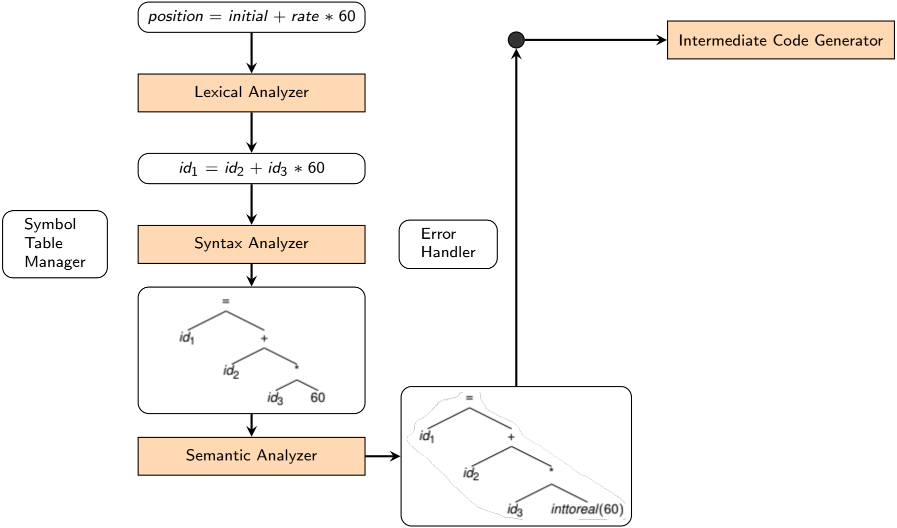
Syntax Analyzer

Symbol Table Manager

Error Handler

Semantic Analyzer

The Phases of a Compiler



$position = initial + rate * 60$

Lexical Analyzer

$id_1 = id_2 + id_3 * 60$

Syntax Analyzer

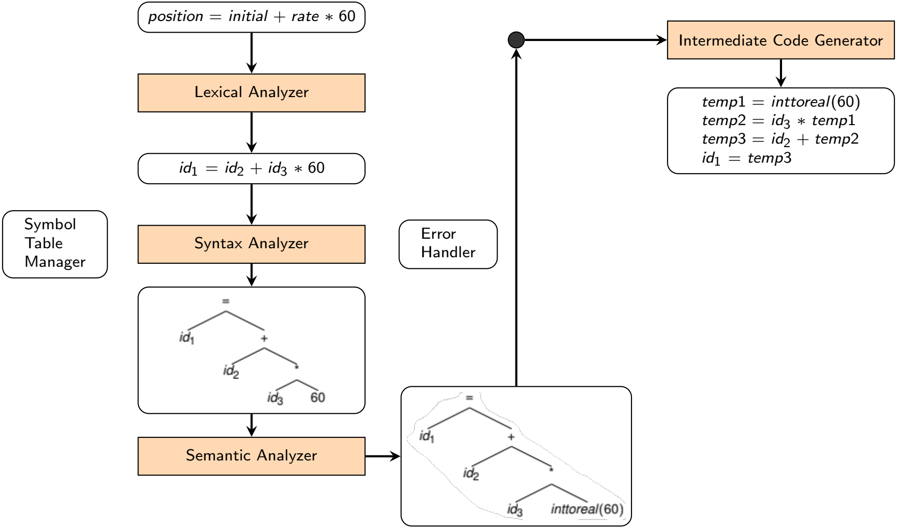
Semantic Analyzer

Intermediate Code Generator

Symbol Table Manager

Error Handler

The Phases of a Compiler



$position = initial + rate * 60$

Lexical Analyzer

$id_1 = id_2 + id_3 * 60$

Syntax Analyzer

Semantic Analyzer

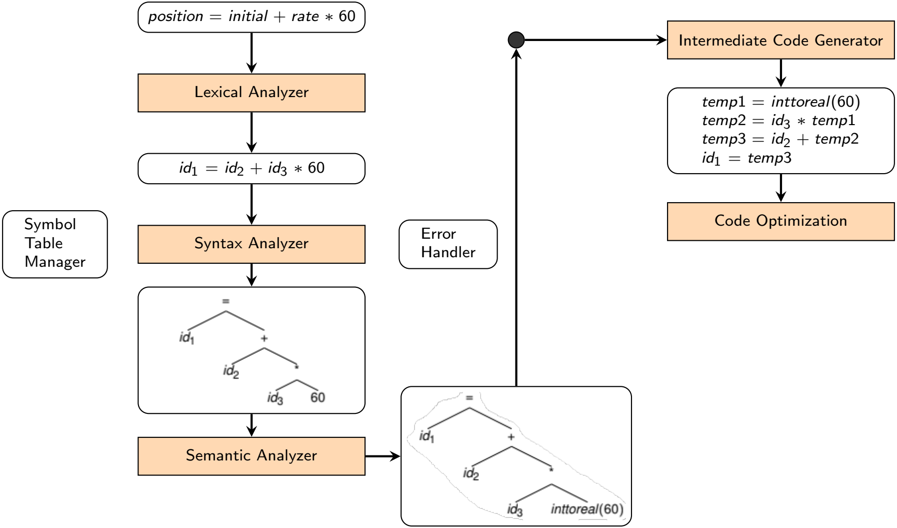
Symbol Table Manager

Error Handler

Intermediate Code Generator

$temp1 = inttoreal(60)$
$temp2 = id_3 * temp1$
$temp3 = id_2 + temp2$
$id_1 = temp3$

The Phases of a Compiler



$position = initial + rate * 60$

Lexical Analyzer

$id_1 = id_2 + id_3 * 60$

Syntax Analyzer

Semantic Analyzer

Symbol Table Manager

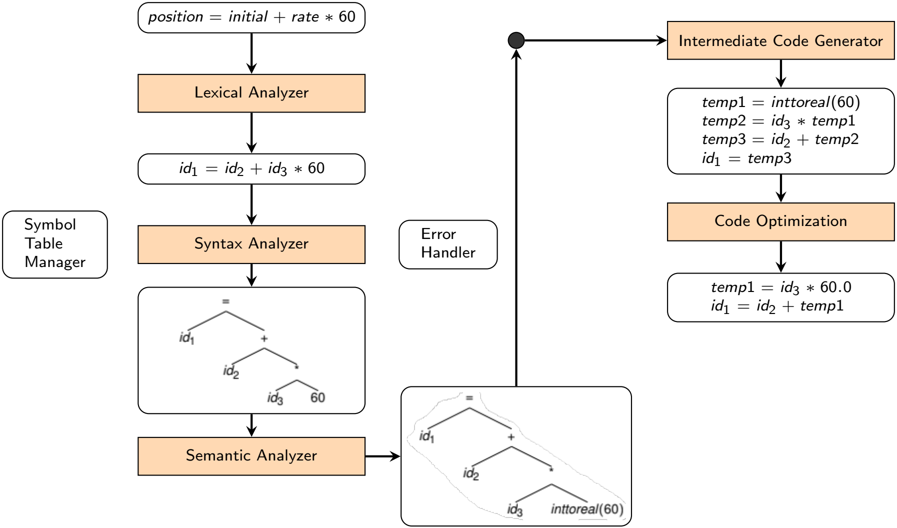
Error Handler

Intermediate Code Generator

$temp1 = inttoreal(60)$
$temp2 = id_3 * temp1$
$temp3 = id_2 + temp2$
$id_1 = temp3$

Code Optimization

The Phases of a Compiler



$position = initial + rate * 60$

Lexical Analyzer

$id_1 = id_2 + id_3 * 60$

Syntax Analyzer

Semantic Analyzer

Symbol Table Manager

Error Handler

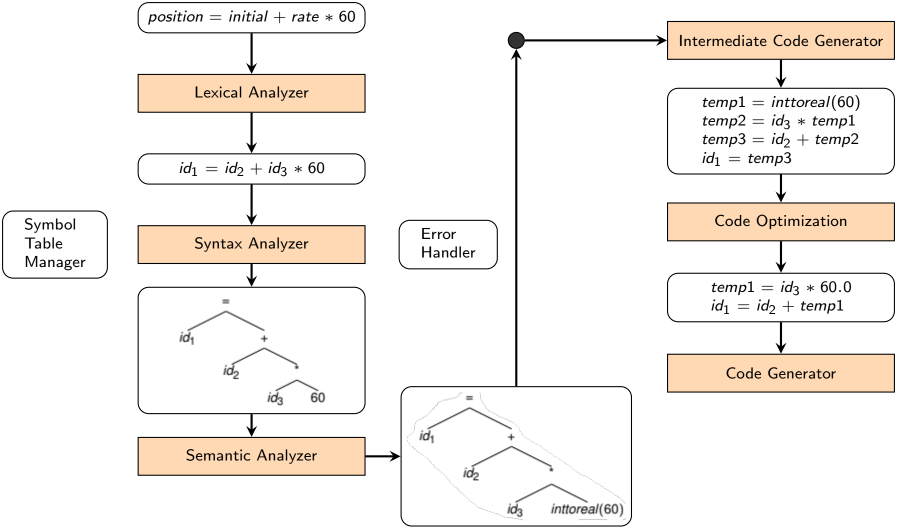
Intermediate Code Generator

$$\begin{aligned} temp1 &= inttoreal(60) \\ temp2 &= id_3 * temp1 \\ temp3 &= id_2 + temp2 \\ id_1 &= temp3 \end{aligned}$$

Code Optimization

$$\begin{aligned} temp1 &= id_3 * 60.0 \\ id_1 &= id_2 + temp1 \end{aligned}$$

The Phases of a Compiler



$position = initial + rate * 60$

Lexical Analyzer

$id_1 = id_2 + id_3 * 60$

Syntax Analyzer

Semantic Analyzer

Symbol Table Manager

Error Handler

Intermediate Code Generator

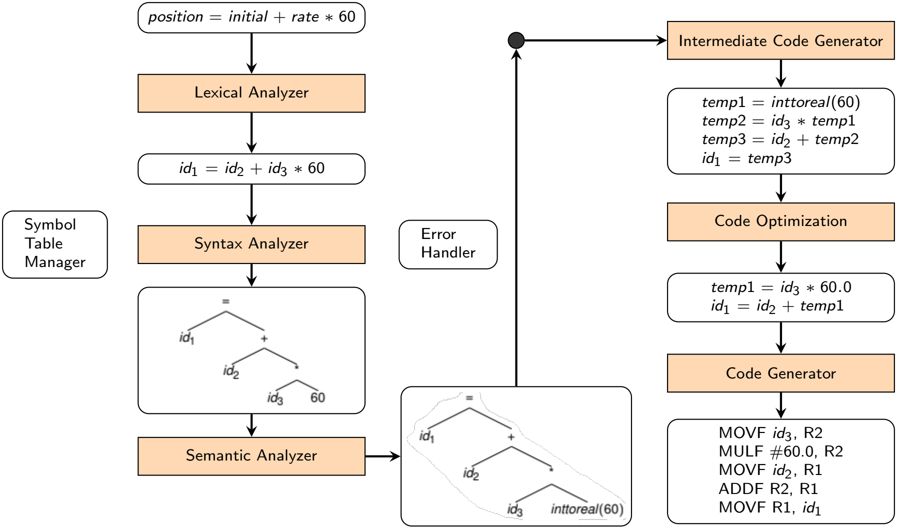
$temp1 = inttoreal(60)$
$temp2 = id_3 * temp1$
$temp3 = id_2 + temp2$
$id_1 = temp3$

Code Optimization

$temp1 = id_3 * 60.0$
$id_1 = id_2 + temp1$

Code Generator

The Phases of a Compiler



$position = initial + rate * 60$

Lexical Analyzer

$id_1 = id_2 + id_3 * 60$

Syntax Analyzer

Semantic Analyzer

Symbol Table Manager

Error Handler

Intermediate Code Generator

$temp1 = inttoreal(60)$
$temp2 = id_3 * temp1$
$temp3 = id_2 + temp2$
$id_1 = temp3$

Code Optimization

$temp1 = id_3 * 60.0$
$id_1 = id_2 + temp1$

Code Generator

MOVF id_3, R2
MULF #60.0, R2
MOVF id_2, R1
ADDF R2, R1
MOVF R1, id_1

Table of Contents

1. Pre-requisite courses

2. Introduction To Compiler Design

3. The Phases of a Compiler

4. Compiler Construction Tools

IIEST, Shibpur
आई आई ई एस टि, शिवपुर
Erstwhile B E College (Estd 1856)

Compiler Construction Tools

Software Tools are available to implement compiler phases-

Compiler Construction Tools

Software Tools are available to implement compiler phases-

- **Scanner Generators** that produce lexical analysers from a regular expression.

Compiler Construction Tools

Software Tools are available to implement compiler phases-

- **Scanner Generators** that produce lexical analysers from a regular expression.
- **Parser Generators** that automatically produce syntax analyser.

Compiler Construction Tools

Software Tools are available to implement compiler phases-

- **Scanner Generators** that produce lexical analysers from a regular expression.
- **Parser Generators** that automatically produce syntax analyser.
- Syntax Directed translation Engine.

Compiler Construction Tools

Software Tools are available to implement compiler phases-

- **Scanner Generators** that produce lexical analysers from a regular expression.
- **Parser Generators** that automatically produce syntax analyser.
- Syntax Directed translation Engine.
- **Data-flow analysis** engines that facilitate the gathering of information about how values are transmitted from one part of a program to each other part. Data-flow analysis is a key part of code optimization.

Compiler Construction Tools

Software Tools are available to implement compiler phases-

- **Scanner Generators** that produce lexical analysers from a regular expression.
- **Parser Generators** that automatically produce syntax analyser.
- Syntax Directed translation Engine.
- **Data-flow analysis** engines that facilitate the gathering of information about how values are transmitted from one part of a program to each other part. Data-flow analysis is a key part of code optimization.
- **Code Generator** that produce a code generator from a collection of rules for translating each operation of the intermediate language into the machine language for a target machine.

Compiler Construction Tools

Software Tools are available to implement compiler phases-

- **Scanner Generators** that produce lexical analysers from a regular expression.
- **Parser Generators** that automatically produce syntax analyser.
- Syntax Directed translation Engine.
- **Data-flow analysis** engines that facilitate the gathering of information about how values are transmitted from one part of a program to each other part. Data-flow analysis is a key part of code optimization.
- **Code Generator** that produce a code generator from a collection of rules for translating each operation of the intermediate language into the machine language for a target machine.

The Evolution of Programming Language

Classification by generation

- **First-generation languages:** machine languages.
- **Second-generation:** assembly languages
- **Third-generation:** higher-level languages like Fortran, Cobol, Lisp, C, C++, C#, and Java.
- **Fourth-generation languages:** languages designed for specific applications like NOMAD for report generation, SQL for database queries, and Postscript for text formatting.
- **Fifth-generation language** has been applied to logic- and constraint-based languages like Prolog and OPS5.

Impacts on Compilers

- The advances in programming languages placed new demands on compiler writers.
- Compiler writers would take maximal advantage of the new hardware capabilities.
- Good software-engineering techniques are essential for creating and evolving modern language processors.

Other Applications

- Techniques used in a lexical analyzer can be used in **text editors**, **information retrieval system**, and **pattern recognition** programs.

Other Applications

- Techniques used in a lexical analyzer can be used in **text editors**, **information retrieval system**, and **pattern recognition** programs.
- Techniques used in a parser can be used in a query processing system such as SQL.

IIEST, Shibpur
आई आई ई एस टि, शिवपुर
Erstwhile B E College (Estd 1856)

Other Applications

- Techniques used in a lexical analyzer can be used in **text editors**, **information retrieval system**, and **pattern recognition** programs.
- Techniques used in a parser can be used in a query processing system such as SQL.
- Many software having a complex front-end may need techniques used in compiler design. (A symbolic equation solver which takes an equation as input. That program should parse the given input equation)

Other Applications

- Techniques used in a lexical analyzer can be used in **text editors**, **information retrieval system**, and **pattern recognition** programs.
- Techniques used in a parser can be used in a query processing system such as SQL.
- Many software having a complex front-end may need techniques used in compiler design. (A symbolic equation solver which takes an equation as input. That program should parse the given input equation)
- Most of the techniques used in compiler design can be used in **Natural Language Processing (NLP)** systems

References

- Alfred V. Aho, Ravi Sethi, Jeffrey D Ullman, “Compilers Principles Techniques and Tools”, Pearson Education.