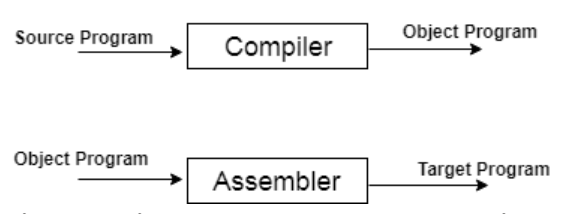
**Chapter 1: INTRODUCTION**

**Topic – 1: Compiler Introduction**

**Introduction**

* **Compiler:** A translator that converts **high-level language** into **machine language**.
* High-level language is often referred as **HLL** & **low-level** language as **LLL**.
* **Object program:** Low-level language, like **assembly**.



**Expected Topics To Be Covered**

* Introduction
* Grammar
* Parsing
* Syntax directed translation
* Symbol table
* Course optimization
* Code generation

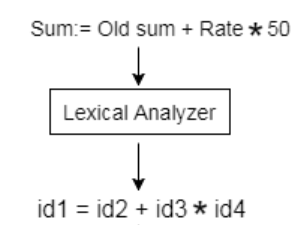
**Topic – 2: Compiler Phases**

**Introduction**

**Source program 🡪 Lexical analyser 🡪 Syntax analyser 🡪 Semantic analyser 🡪 Intermediate code generation 🡪 Code optimization 🡪 Code generation 🡪 Target program**

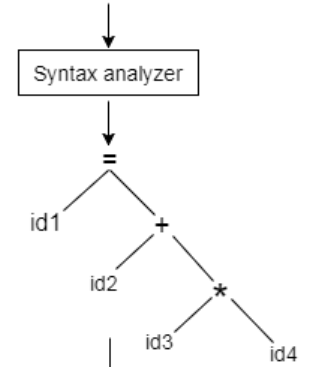
**Lexical Analysis**

* Analyses **source program** character-by-character.
* Converts program into meaningful ***lexemes***.
* **Lexemes** are represented in form of **tokens**.



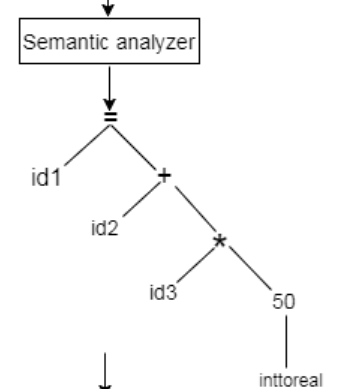
**Syntax Analysis**

* Takes **tokens** (**lexemes**) as input.
* Converts it into a **parse tree**.
* Then parser checks if the syntax is written the **right way** or **not**.



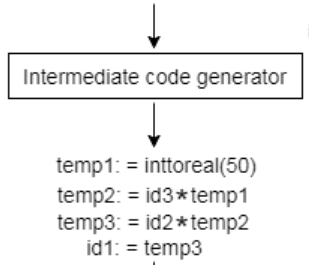
**Semantic Analysis**

* In this phase, it checks if parse tree follows **language rules** or **not**.
* Then it tracks all involved **identifiers**, their **types** & the **expression**.



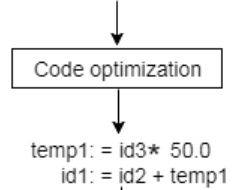
**Intermediate Code Generation**

* Code which is then converted into final **machine code**.



**Code Optimization (\*Optional)**

* Converting **intermediate code** in a form that will **run faster** & **take less space**.
* Like **removing unnecessary lines**, **arranging code lines in an order** etc.



**Code Generation**

* Takes the **optimized** **intermediate code** or just **intermediate code** as input.
* Converts that into processor’s **machine code**.

