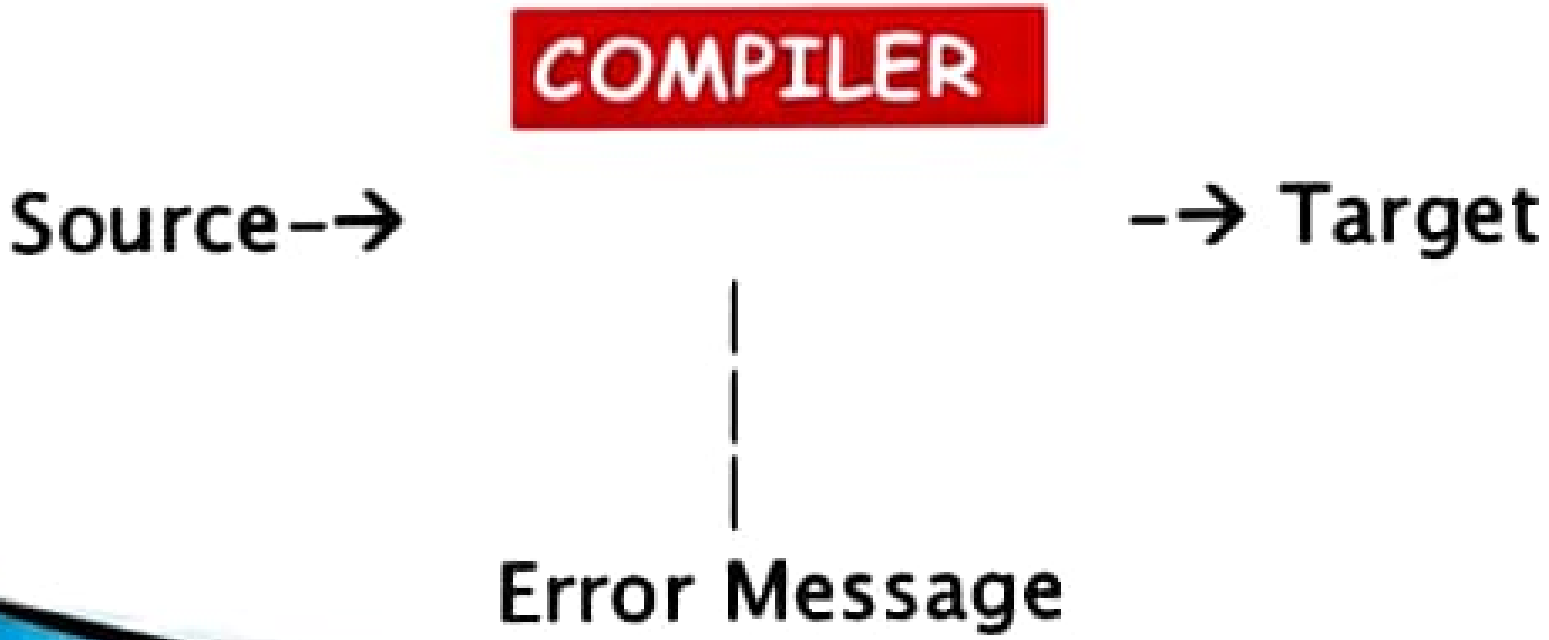
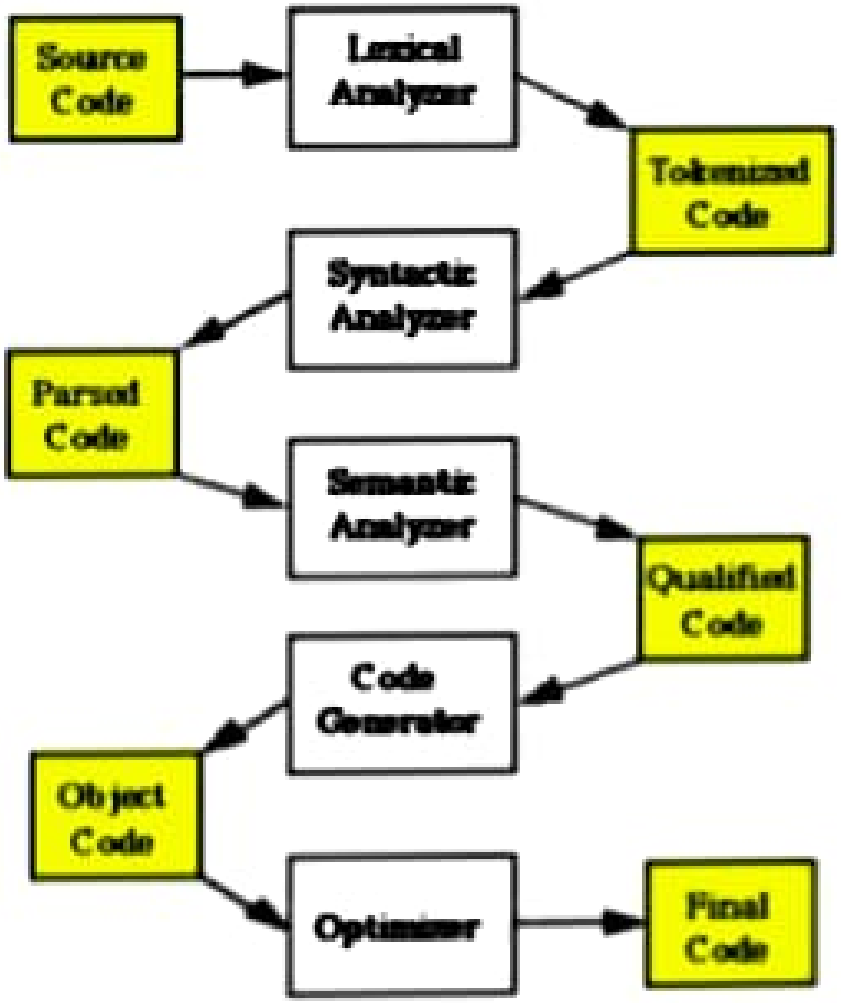
DEFINITION

- A compiler is a program that reads a program written in one language and translates into equivalent target language



Language Processing Technique



Structure of a compiler

- The **Front end** checks whether the program is correctly written in terms of the programming language syntax and semantics

- The **back end** is responsible for translating the source into assembly code.

Structure of a compiler

Front End :

- Lexical Analysis
- Preprocessing
- Syntax Analysis
- Semantic Analysis

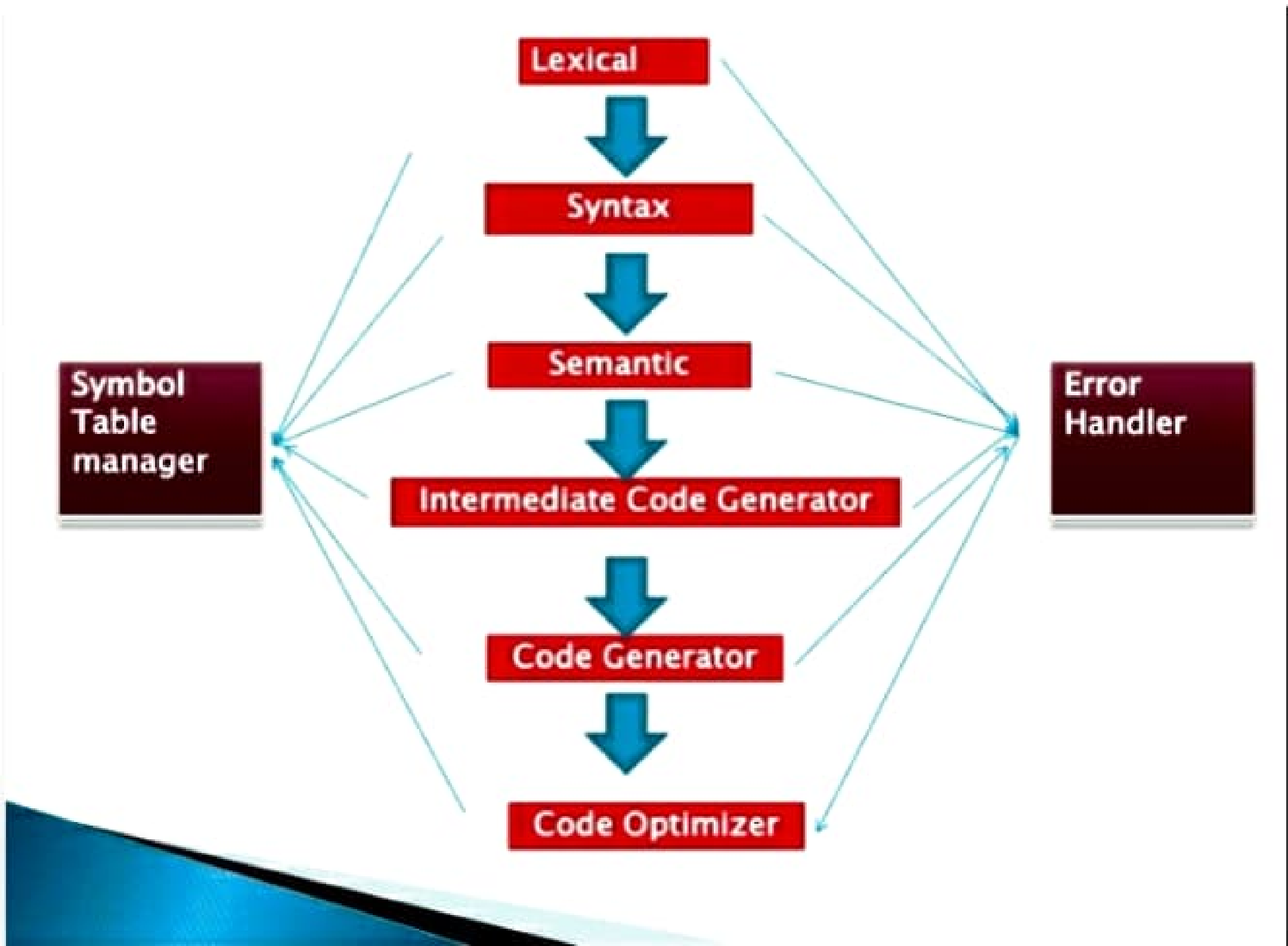
Structure of a compiler

Back End

- Analysis
- Optimization
- Code generation

Phases of a compiler

- Lexical Analyzer
- Syntax Analyzer
- Semantic Analyzer
- Intermediate code generator
- Code optimizer
- Code generator


Lexical
Syntax
Semantic
Intermediate Code Generator
Code Generator
Code Optimizer
Symbol Table manager
Error Handler

Lexical analysis

- Also called Linear Analysis
- Characters read from left to right and grouped into tokens that are a sequence of characters with a collective meaning

→Scans Input

→Removes White spaces and comments

→Manufacture Tokens

→Generate Error if Any

Lexical analysis

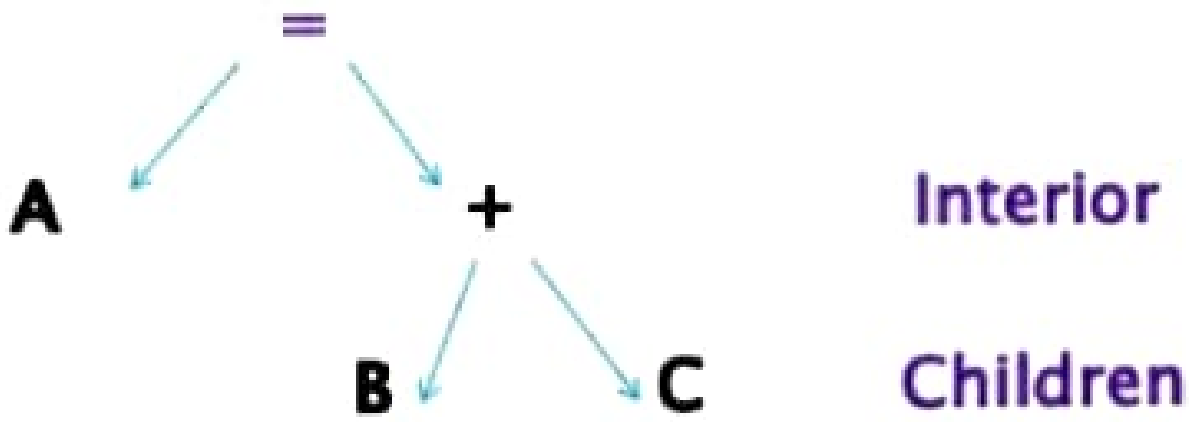
- **Example**

- **A=B+C**

- **Variable tokens –→ A ,B, C**
- **Symbolic token –-→ = +**

Syntax Analysis

- Also called as Hierarchial Analysis
- A syntax tree[also called as parse tree] is generated where
  - Operators → Interior nodes
  - Operands → Children of node for operators.



Semantic analysis

- Characters grouped as tokens in Lexical Analysis are recorded as Tables. Checks for semantic errors

- Collect TYPE information for the subsequent code generation phase

Intermediate Code Generator

- Sophisticated compilers typically perform multiple passes over various intermediate forms.
- Many algorithms for code optimization are easier to apply one at a time
- The input to one optimization relies on the processing performed by another optimization

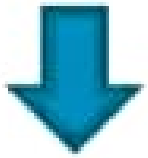
Working of ICG

- **Input** : Concrete Syntax Tree[Parse Tree] or Abstract Syntax Tree
- The tree is converted into a linear sequence of instructions, usually in an intermediate language such as three address code.
- This is an Early stage of Code generation

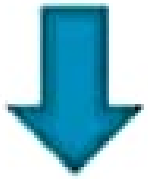
Working of ICG

Concrete Parse tree

Abstract syntax tree



Converted into a linear sequence of instructions



Results in 3AC [3 Address Code]

Code optimization

- This phase attempts to improve the intermediate code inorder to increase the running time
- Reduce the complexity of the code generated
- Leading to a faster execution of the program
- Increased Performance

Code optimization

- Platform Dependant/ Platform Independent
- Optimization can be automated by compilers or performed by programmers
- Usually, the most powerful optimization is to find a superior algorithm.
- Include activities like
  - Optimization of LOOPS
  - Optimization of Bottlenecks

Code generator

- Succeeding step of Intermediate code optimizer
- Consists of re-locatable machine code/assembly code
- Intermediate instructions are converted into a a sequence of machine instructions

Types of compilers

- One pass compilers
- Multi pass compilers
- Load and go compilers
- Optimizing compilers

One and multi pass

One pass	Multi pass
Passes through the source code of each compilation unit only once	Processes the source code of a program several times
Compilation time is faster	Compilation time is slower
Has limited scope of passes	Has wide scope of passes.
wide compilers	Narrow compilers
Pascal	Java

Load and Go

- Generates machine code and immediately executes it.

- Compilers usually produce either absolute code that is executed immediately upon conclusion of the compilation or object code that is transformed by a linking loader into absolute code.

Optimising compilers

- Loop optimization
- Data flow
- Code generation
- Functional language
- Interprocedural optimizations
- SSA [Static Single Assignment] based optimizations

Other Types of compilers

Threaded code compiler

- database lookup program.
- replaces given strings in the source with given binary code.

- **Incremental compiler**:
  - Individual functions can be compiled in a run-time environment that also includes interpreted functions.

Other Types of compilers

- **Stage compiler**
  That compiles to assembly language of a theoretical machine, like some Prolog implementations

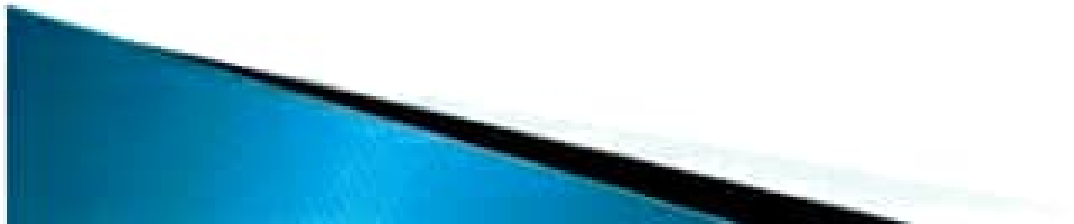
- **Just-in-time compiler**
  - Applications are delivered in byte code, which is compiled to native machine code just prior to execution

Other Types of compilers

- A Retargetable compiler
  - object code is frequently of lesser quality than that produced by a compiler developed specifically for a processor.
  - Retargetable compilers are often also cross compilers
- A parallelizing compiler
  - converts a serial input program into a form suitable for efficient execution on a parallel computer architecture.

Errors by compilation

- The compiler highlights all the possible errors which are obstacle in the program to provide a good meaning.
- Logical errors could be found only at Run time of the program.



Advantages of compiling

- Program is free from lexical errors
- Program is free from syntax errors
- Program is free from semantic errors
- Even a complex program could be compiled in a small interval of time
- In programs related to database accessing, many risks are reduced.