UNIT - I

Lexical Analysis.

Introduction - Language processors - Structure of a compiler - Lexical analysis - Role of Lexical Analyzer - Input Buffering - Specification of Tokens - Recognition of Tokens - The Lexical Analyzer Generator - Finite Automata From Regular expression to Automata

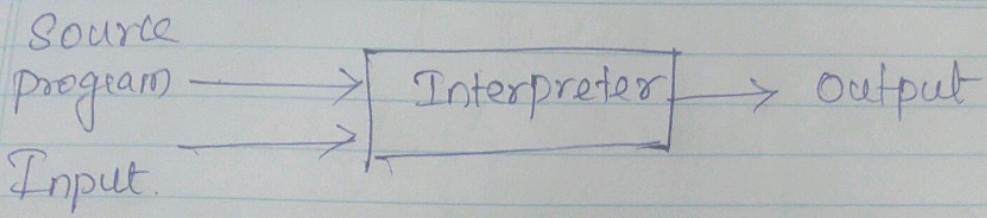
Language Processing System :-

Language processors :-

1. Interpreter :-

It is a Computer program that execute instructions written in a programming language.

It executes the source code directly or translate source code into some efficient intermediate representation and immediately executes this.



Example:- Early version of Lisp prog. Lang, BASIC.

2. Translator:-

A Software System that converts the source code from one form of the language to another form of Language.

- They are 2 types of Translators.

1. Compiler
   → Converts source code of high level language into low level language.

2. Assembler:-
   → Converts assembly language code into binary codes.

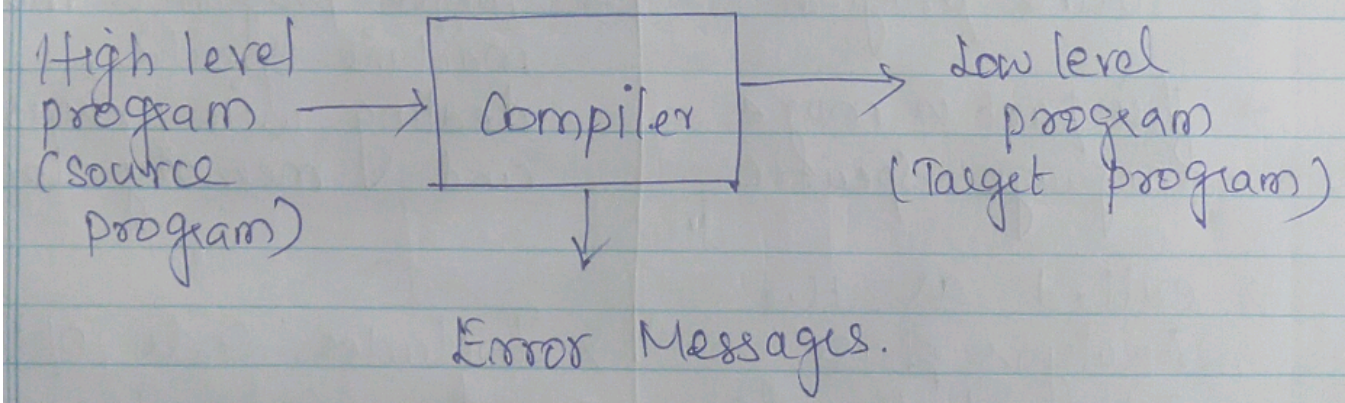
Compiler:- A Compiler is a software that translates code written in high-level language into target language.

Example:- Source languages:- C, Java.

Compilers are user friendly

Target language:-

Machine Language → Efficient for Hardware.

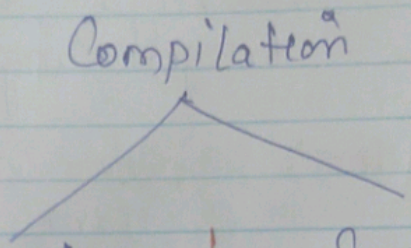


Passes:- The number of iteration to scan the source code, till to get the executable code is called as a pass.

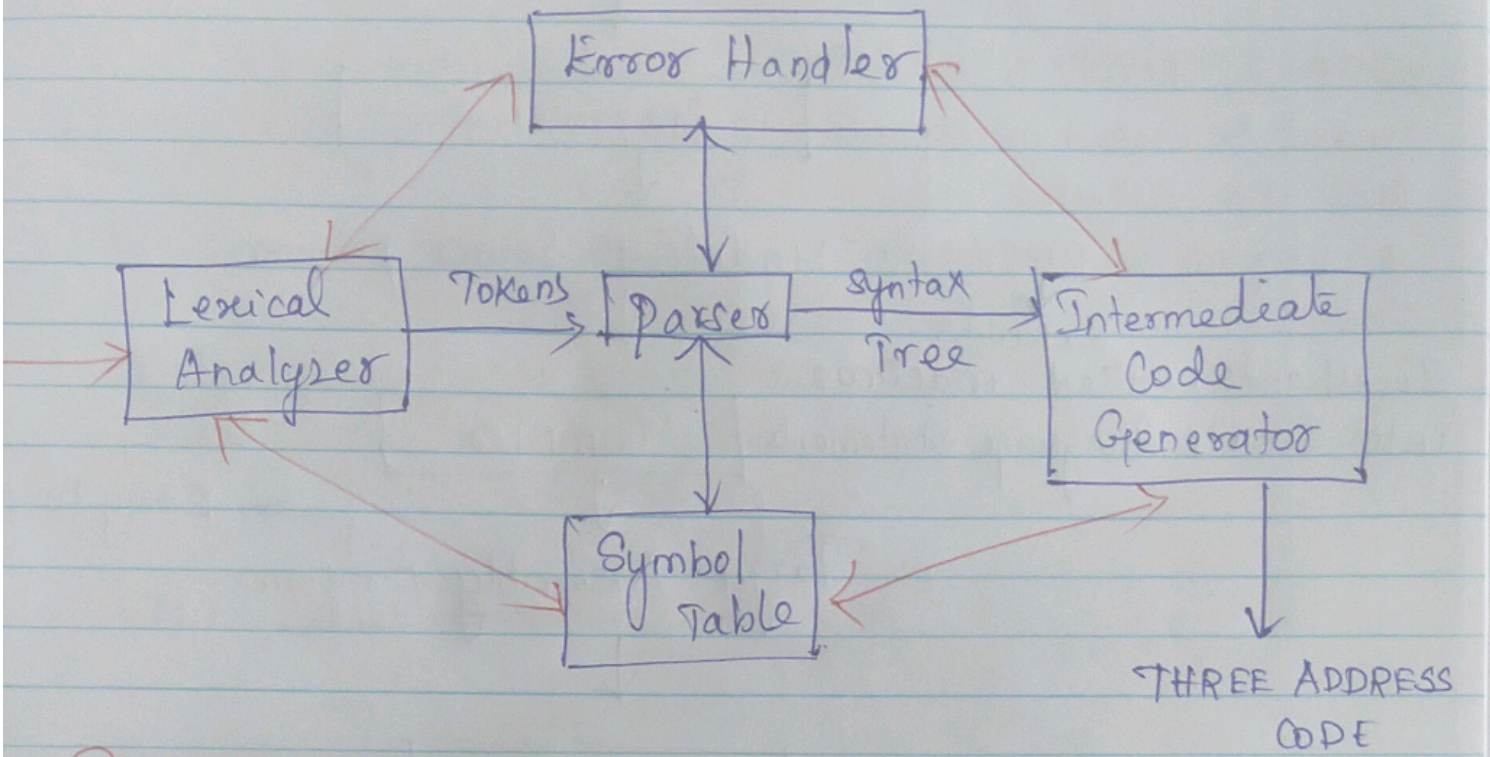
Compiler is two pass.
Single pass → Requires more memory
Multipass → Requires less memory.

Analysis - Synthesis model of Compilation :-

They are two part of compilation.



Analysis (Front end)	Synthesis (Back-end)
→ Breaks the Source program into pieces	→ It Constructs the desired target program from the intermediate representation.
→ Create Intermediate representation of the Source program	→ Target program is more machine specific, dealing with registers and memory locations.
→ This part is more language specific.	
→ called as the Analysis phase, intermediate code generator with part of code optimization.	→ Includes, code optimization and code generation phase.
	→ The back end Synthesises the target program from intermediate code.



Context of a Compiler:-

In addition to compiler, several other programs may be required to execute an executable target program.

like :- Preprocessor to macro expander.

The Target program created by a compiler may require further processing before it can be run.

Source program with macros

↓

Source program which are divided & stored in separate files are collected by a program — preprocessor.

Preprocessor

↓

Modified Source program.

↓

Expands Shorthands, called macros into Source language statements.

Compiler

↓

Target Assembly program → Easy to produce & debug

↓

Assembler

↓

Relocatable Machine code

Larger programs are often compiled into pieces. So relocatable machine code are linked together with other relocatable object files and library files.

↓

Loader | linker ← Library files, relocatable object files.

↓

Absolute machine code

Context of a Compiler

Linker:- resolves external memory addresses, where the code in one file may refer to a location in another file

Loader:- puts together all of the executable object files into memory for execution.