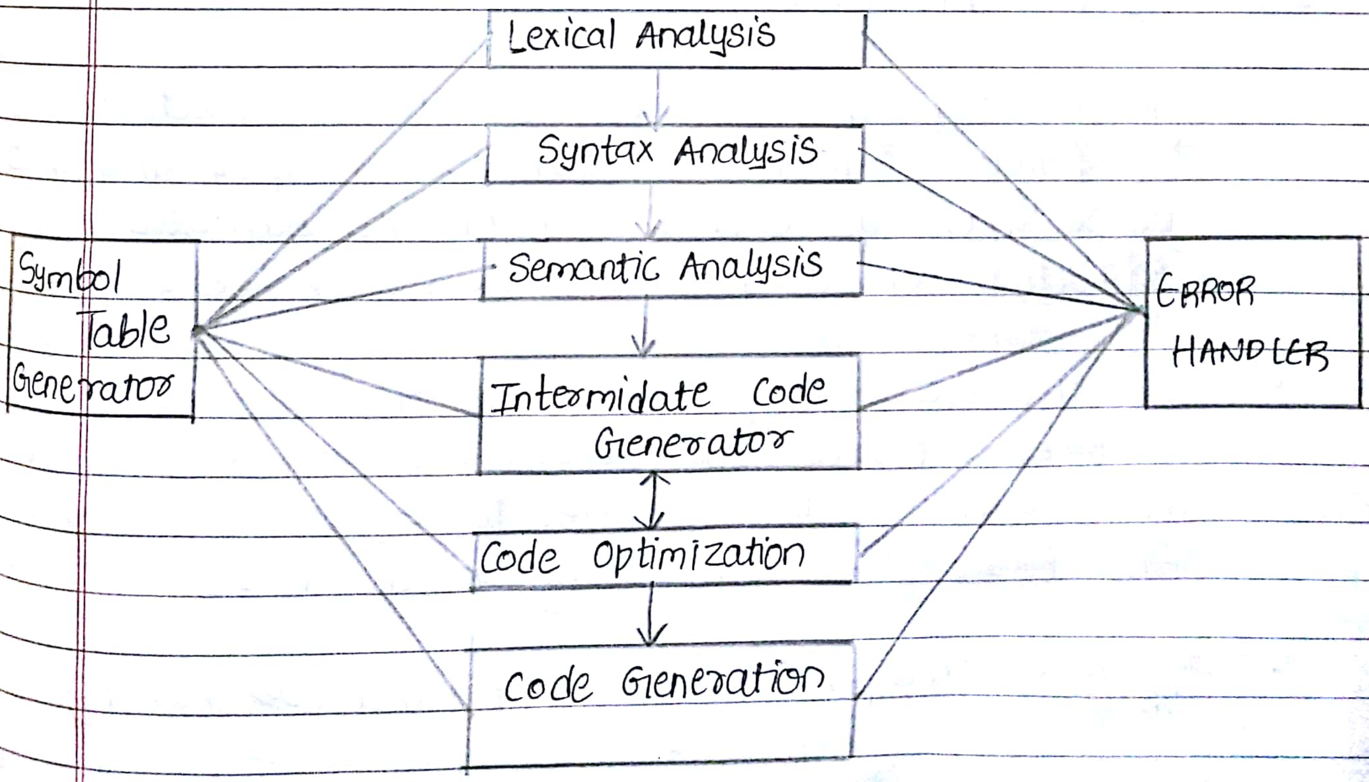
# COMPILERS

## DEFINITION :

- A complier is a program that reads any source program and translates it into an equivalent program in another or target language.
- As an important part of translation process, the complier reports to its user the presence of the errors in source Program.

## BLOCK DIAGRAM OF COMPILERS



## I) LEXICAL ANALYSIS

- The lexical analyzer is the first phase of complier. Its main task is to read the i/p characters and produces o/p a sequence of tokens that the praser uses for syntax analysis

- In this phase, stripping out of user comments, white spaces, tab, new line character and blanks from the program are being taken care of.

- Another task is correlating error message from the complier with the source program.

## II) Syntax Analysis:

- In the second phase of Compiler Construction. It takeas stream of tokens as input and gives parse tree as output. It checks syntatic structure of a program & checks for errors.

- It involves the grouping of the source program tokens into grammatical phases that are used by the complier to synthesize output. The Parse Trees are being made here.

- Syntax Analysis is also termed as Parsing

## III) SEMANTIC ANALYSIS

- It checks for the semantic errors and gathers type info, uses the hierarchical structure determined by the previous phase to identify the operators and the operands of expressions & statements

- Important component here is type checking, where the complier check that each operator has operands that are allowed by the source language specification.

- In other words, this phase adds meaning to various tokens, identifiers, operators, expressions etc.

## IV) INTERMIDATE CODE GENERATOR

- In this many compliers generate an explicit low level or machine like intermidate representation

For eg:-

$t_1$ = int to float (60)
$t2$ = $id3 * t3$
$t3$ = $id2 + t2$
$id1$ = $t3$

this type of representation should have two important properties. ~~Str~~ It should be easy to produce & easy to translate into target m/c

Types of IR are:

1) Structural 2) Linear 3) Hybrid

## V CODE OPTIMIZATION

- This phase attempts to improve the intermidate code so that faster running machine code will result
- The basic idea of this field is to improve the execution efficiency of the program. For eg:-

$t_1 = id3 * 60.0$
$id1 = id2 + t1$

## VI CODE GENERATION

- The final phase of comptilation process is the generation of target code, consisting normally of relesable machine code or assembly code.

- Memory locations are selected for each of the variable used by the program, then IR are translated into a sequence of m/c instn that perform the same task

- To avoid redundant codes, the code generator must keep track of the register contents at runtime

- A good code generator would attempt to use these registers as efficiently as possible. This aspect of code generation is called register allocation.