

The analysis and synthesis phases of a compiler are:

## Analysis Phase

Breaks the source program into constituent pieces and creates intermediate representation.

The analysis part can be divided along the following phases:

### 1. Lexical Analysis

The program is considered as a unique sequence of characters. The Lexical Analyzer reads the program from left-to-right and sequence of characters is grouped into tokens-lexical units with a collective meaning.

### 2. Syntax Analysis

The Syntactic Analysis is also called Parsing. Tokens are grouped into grammatical phrases represented by a Parse Tree, which gives a hierarchical structure to the source program.

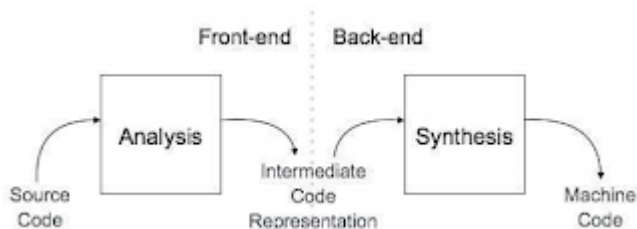
### 3. Semantic Analysis

The Semantic Analysis phase checks the program for semantic errors (Type Checking) and gathers type information for the successive phases. Type Checking check types of operands; No real number as index of array etc

## Synthesis Phase

Generates the target program from the intermediate representation.

The synthesis part can be divided along the following phases:



## Intermediate Code Generator

An intermediate code is generated as a program for an abstract machine. The intermediate code should be easy to translate into the target program.

## Code Optimizer

This phase attempts to improve the intermediate code so that faster-running machine code can be obtained. Different compilers adopt different optimization techniques.

## Code Generator

This phase generates the target code consisting of assembly code.

Here,

- Memory locations are selected for each variable
- Instructions are translated into a sequence of assembly instructions
- Variables and intermediate results are assigned to memory registers.

## Analysis synthesis model of compilation

