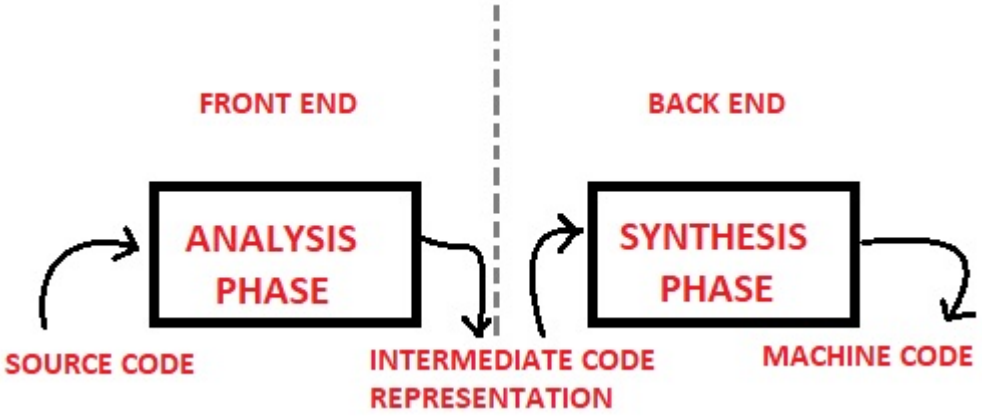
A compiler can be divided into two phases based on the way they compile:

1. Analysis phase, also known as front end as shown in diagram below.
2. Synthesis phase, also known as back end as shown in diagram below.



The code optimization in the synthesis phase is a program transformation technique, which tries to improve the intermediate code by making it consume fewer resources (i.e. CPU, Memory) so that faster-running machine code will result.

Compiler optimizing process should meet the following objectives:

- The optimization must be correct, it must not , in any way, change the meaning of the program.
- Optimization should increase the speed and performance of the program.
- The compilation time must be kept reasonable.
- The optimization process should not delay the overall compiling process.

When to optimize the code?

Optimization of the code is often performed at the end of the development stage since it

reduces readability and adds code that is used to increase the performance.

Types of code optimization:

1. Machine independent optimization
2. Machine dependent optimization

1. Machine Independent Optimization

It is done before the target code get generated as the output. It does not involve any CPU registers or memory hierarchy.

2. Machine Dependent Optimization

It is done after the target code get generated as the output. I involves the maximum use of CPU registers or memory hierarchy.

Unoptimized code

```
repeat
S1;
i := j;
s2
until p;
```

Optimized code

```
i := j;
```

```
repeat
Sl;
s2
until p;
```