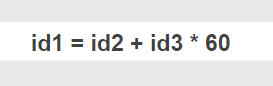
**Stages of program compilation:**

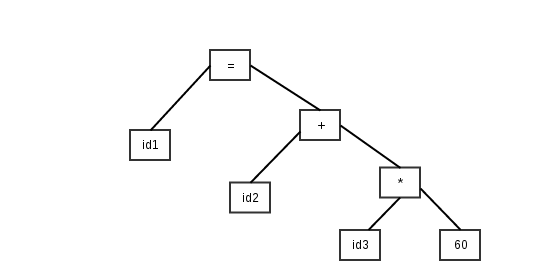
Compilation is the process in which a high-level language is translated into machine language. During this stage, at least four phases can be recognized:

Suppose we need to compile the following expression: **price = cost + tax \* 60**

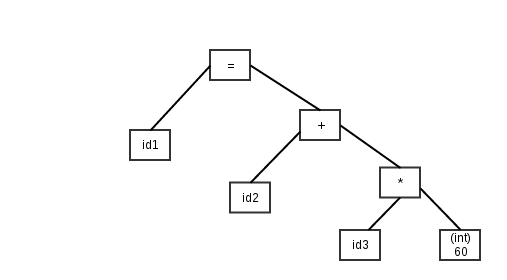
1. **Lexical Analysis:**

It extracts from some file all the strings that it can recognize as part of the vocabulary and in this way generates a set of output tokens. If any part of the input file is not recognized as a valid language, the corresponding error messages will be generated.

1. **Syntactic Analysis:**

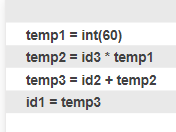
During the parsing, the sequence of tokens that was generated above is processed and an intermediate representation is built, but without being a machine language yet. This allows the compiler to do its job more easily in successive phases.

1. **Semantic Analysis:**

In this step, the generated tree diagram above is used to detect possible violations to the semantics of the programming language, such as the declaration and consistent use of identifiers (for example, that the data type on the right-hand side of an assignment is consistent with the type data of the target variable, on the left side of the assignment).

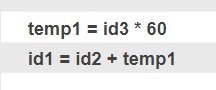
1. **Code generation:**

After having conducted the semantic analysis, the compiler generates an intermediate code between the source code and the target machine code. This intermediate code must be generated in such a way that it is easily translated into a low-level machine language.



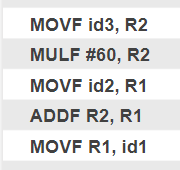
1. **Optimization:**

It can be assumed as something that removes lines of code that are unnecessary and reorders a sequence of statements which speed up the execution of the program without wasting RAM memory.



**6) Code generation:**

In this phase, the optimized version of the intermediate code is taken and mapped to the target machine language. The intermediate code is translated into a relocatable sequence of machine code, later these instructions are taken by the linker of the operating system to allocate a space in it. memory and in this way make it work.



**Levels of programming:**

**-** **Low level:**

A low-level or first-generation feature programming language is one in which its instructions exert direct control over the hardware and are conditioned by the physical structure of the computers that support it.

Example: machine language and assembly language.

**-** **Middle level:**

They are between high-level languages ​​and low-level languages ​​and are accurate for certain applications such as building operating systems, as they allow abstract handling (machine independent, as opposed to assembler), but without losing much of the power and efficiency.

Ejemplos: BCPL, C

**-** **High level:**

It is the one that is closer to human natural language than to the binary language of computers, its main function is that, from its development, there is the possibility that the same program can be used on different machines, that is, it is independent of specific hardware.

Ejemplos: Java, C++, Clipper, Modula, Prolog, Visual Basic.

**References:**

Aho, Alfred V., Ullman, Jeffrey D. y Hopcroft, John E. - Data Structures and Algorithms - Addison Wesley, EE.UU, 1983.

https://ricardogeek.com/fases-del-compilador/

http://conogasi.org/articulos/lenguaje-de-programacion/