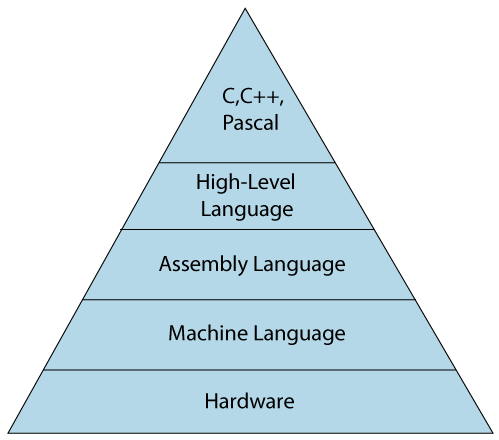
**Stages of program compilation**

There are six stages of compiling a program:

1. **lexical analysis 4. semantic analysis**
2. **symbol table construction 5. code generation**
3. **syntax análisis 6. optimisation**

* **Lexical analysis**
* Comments and unnecessary spaces are removed.
* Keywords, constants and identifiers are replaced by 'tokens', which are symbolic strings to identify what the elements are.
* **Symbol table construction**
* A table stores the names and addresses of all variables, constants and arrays.
* Variables are checked to make sure they have been declared and to determine the data types used.
* **Syntax analysis**
* Tokens are checked to see if they match the syntax of the programming language.
* If syntax errors are found, error messages are produced.
* **Semantic analysis**
* Variables are checked to make sure they have been correctly declared and contain the correct data type.
* Operations are checked to ensure that they are appropriate for the type of variable being used.
* **Code generation**
* Machine code is generated in this stage.
* **Optimisation**
* Code optimisation makes the program more efficient so it runs faster and uses fewer resources.

**Levels of programming**



* **Low-level language**

The low-level language is a programming language that provides no abstraction from the hardware, and it is represented in 0 or 1 forms, which are the machine instructions. The languages that come under this category are the Machine level language and Assembly language.

**Machine-level language**

The machine-level language is a language that consists of a set of instructions that are in the binary form 0 or 1. Since assembly language instructions are written in English words like mov, add, sub, so it is easier to write and understand.

**Assembly Language**

The assembly language contains some human-readable commands such as mov, add, sub, etc. The problems which we were facing in machine-level language are reduced to some extent by using an extended form of machine-level language known as assembly language.

* **High-Level Language**

The high-level language is a programming language that allows a programmer to write the programs which are independent of a particular type of computer. The high-level languages are considered as high-level because they are closer to human languages than machine-level languages.

When writing a program in a high-level language, then the whole attention needs to be paid to the logic of the problem.

A compiler is required to translate a high-level language into a low-level language.

**Differences between Low-Level language and High-Level language**

|  |  |
| --- | --- |
| **Low-level language** | **High-level language** |
| It is a machine-friendly language, i.e., the computer understands the machine language, which is represented in 0 or 1. | It is a user-friendly language as this language is written in simple English words, which can be easily understood by humans. |
| The low-level language takes more time to execute. | It executes at a faster pace. |
| It requires the assembler to convert the assembly code into machine code. | It requires the compiler to convert the high-level language instructions into machine code. |
| The machine code cannot run on all machines, so it is not a portable language. | The high-level code can run all the platforms, so it is a portable language. |
| It is memory efficient. | It is less memory efficient. |
| Debugging and maintenance are not easier in a low-level language. | Debugging and maintenance are easier in a high-level language. |

**Bibliographic references**

(BBC, 2022)

(National Centre for Computing Education, 2022)

(Javatpoint, 2022)