Stages of compilation

To program a code we need tools such as compilers and interpreters to transform programs written into code that machines can understand, so the compilation process are the phases that the code has to go through to become an interpretable code.

Steps to create source code:

I. **Lexical análisis:** Lexical analysis is the first step of the compilation process.

- Comments and unnecesary space are removed: The first part consists in removing the unnecessary characters for the codes, the programming code is mainly made by humans and we need to read those codes, that's why we add spaces and make them readable, organized, and understandable for anyone who reads them, this is indispensable for us but unnecessary for the executable code for that reason the compiler removes them during the analysis.

- Keywords, constants and identifiers are replaced by 'tokens': This is a process in which each line of code is analyzed and some characters of the line of code are replaced by a 'token' depending on its function, for example, "user_name" will be an identifier, "=" will be an operator and "Jorge" and "Mario" will be literal.

II. **Syntax Analysis:** This step is similar to find grammatical errors in normal sentences, once the tokens have been assigned depending on their use, the compiler verifies that they comply with the rules of the programming language and that they follow a correct order (*For example, in Python the command print(user_name) is syntactically correct, in that it follows the rules for a print statement: print in lowercase immediately followed by a bracket followed by an identifier and closed by a righthand bracket*.)
The compiler proceeds to detect that each structure is correct and if required tokens are missing from the tree, or in the wrong place, the compiler will report an error and we need to take into a consideration that each programming language has a different syntaxys.

III. **Code generation:** In this step, the program is generated that is different from the original code source, so the machine code is generated. This is the executable version of the code, before the linked libraries are included.

IV. **Code optimisation:** In this step, the code is optimized, the optimization identifies parts of the code that are repeated or are not necessary and can eliminate or reorganize, this optimization makes the code more efficient, fast and does not use so many resources.

Levels of programming

A programming language is a set of instructions that are compiled to take specific actions, each programming language is unique in keywords and syntax and at this moment we have a lot of different programming languages with different purposes.

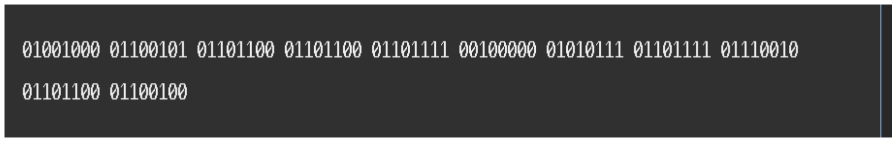
These languages vary in the level of abstraction they provide from the hardware. Some programming languages provide less or higher abstraction.

I. **Low-level Programming Language:** The low-level language does not represent much abstraction for the hardware and is represented by or and 1 and in this way, instructions are given to the machine. The languages included in this level contain loops, procedures, functions, and some typing.

Some programming languages that are included in this category are: FORTRAN, COBOL, BASIC, arguably C

II. **Machine level language:** This language is a language that consists of a set of instructions made by 0 and 1 in other words, in a binary form. As all of us know computers only can understand orders in binary code and for that reason create a code in this programming language is so difficult for programmers because it is a very big challenge and its maintenance is also very complicated.
At the same time, this language is difficult to move to other machines because each computer has its own machine instructions so a program written on one computer will no longer be valid on another.
As an example, we have the different architectures of processors that uses different machine codes, for example a PowerPC processor contains RISC architecture, which requires different code than intel x86 processor, which has a CISC architecture.

Example of machine language (binary) for the text "Hello Wold"

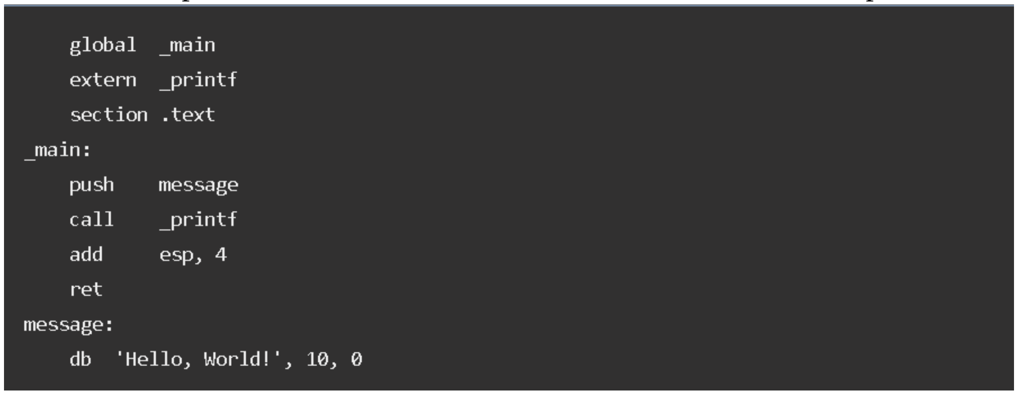


```
01001000 01100101 01101100 01101100 01101111 00100000 01010111 01101111 01110010
01101100 01100100
```

III. **Assembly Language:** The assembly language adds some human-readable commands like "add", "sub", "mov" and many others. The difficulty of the language at the machine level is reduced a bit since the instructions allow the use of the commands mentioned above so it is easier to write and understand.
Computers can only understand instructions at the machine level, so a translator is needed to convert them and the translator used to translate the

code is known as an assembler, Another aspect to take into account is that the data of this code is not transferable since it is stored in the computer registers.

Example : Here is "Hello, World" written for a 32-bit Intel processor.



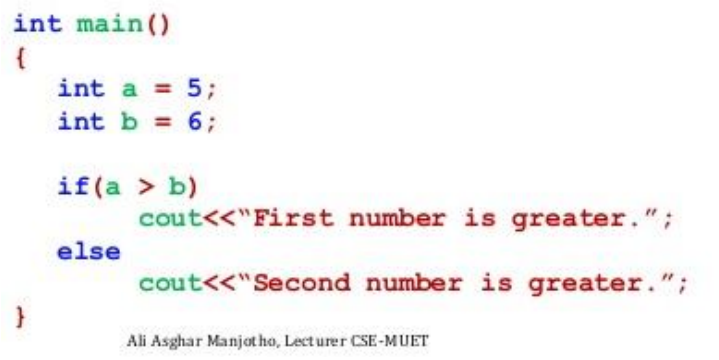
```
    global  _main
    extern  _printf
    section .text
_main:
    push    message
    call    _printf
    add     esp, 4
    ret
message:
    db  'Hello, World!', 10, 0
```

Examples of mid-level programming languages include C, C++, Ada, Nim, and Rust

IV. **High-level Programming Language:** High-level languages give a programmer the ability to write programs that are independent of a single computer or a particular type of machine. These languages are considered high-level because they are the most similar to human languages and are the most distant from machine-level languages.

Examples of high-level programming languages in active use today include Python, Visual Basic, Delphi, Perl, PHP, ECMAScript, Ruby, C#, Java and many others. The terms high-level and low-level are inherently relative.

High level language program example



```
int main()
{
   int a = 5;
   int b = 6;

   if(a > b)
        cout<<"First number is greater.";
   else
        cout<<"Second number is greater.";
}
```

References

BBC. (20 of 6 de 2018). *Bitesize*. Obtenido de Program construction: https://www.bbc.co.uk/bitesize/guides/zmthsrd/revision/3

Education, N. C. (7 de 2 de 2016). *isaac computer science*. Obtenido de Stages of compilation: https://isaaccomputerscience.org/concepts/sys_trans_stages

JavaTpoint. (19 de 8 de 2016). *JavaTpoint*. Obtenido de What is a programming language?: https://www.javatpoint.com/classification-of-programming-languages

Penn, G. (3 de 10 de 2017). *University of Toronto*. Obtenido de Levels of Programming Languages: https://www.cs.toronto.edu/~gpenn/csc324/lecture2.pdf