

**Polytechnic University of Yucatan**

“Task 1”

**Student’s name:**

* Christian Isaac Dzul Canul

**Teacher’s name:** Luis Gerardo Cámara Salinas

**Subject:** Structured Programming

**Group:** Robotics 2A

**Date:** 09/05/2021

**“Stages of program compilation”**

The compilation is a process in which the source code is converted into object code. This is possible with the help of the compiler, which has the function of check the source code for the syntactical or structural errors, and if there are no error in the source code, then is possible to generate the object code.

The compilation process can be divided into four steps:

1. **Pre-Processing**

In this stage, lines starting with the character “#” are interpreted by the preprocessor as preprocessor commands. These commands form a simple macro language with is own syntax and semantics, and then certain pieces of text are replaced by other text according to the system of macros. This source code is first passed to the preprocessor, and then the preprocessor expands this code. After expanding the code, the expanded code is passed to the compiler.

1. **Compiling (compiler)**

In the second stage, the code which is expanded by the preprocessor is passed to the compiler. The compiler converts this code into assembly code. In other words, the compiler converts the pre-processed code into assembly code.

1. **Assembling**

In this part of the process, after the code transform the code into assembly code, then with the help of an assembler, it has the task of translate the assembly instructions to object code.

1. **Linking**

After the object code is generated in the assembly stage, it is composed of machine instructions that the processor understands but some pieces of the program are out of order or missing, that is why is necessary the linking process to reorganize (arrange) the pieces that are not in order or that are not completed in order to produce an executable program.

**Other processes during the compilation**

There also exist six stages in the program compilation, which are:

* Lexical analysis

In this part the source code created (which is a text document that contains a lot of characters) is tokenized for translation into executable code. It also has the purpose of remove any non-program elements, like for example comments and unnecessary spaces.

* Symbol table construction

It is a table that stores the names and addresses of all the variables, constants, and arrays. In this part, the variables are checked to confirm that they have been declared (if variables were assigned a name and data type) and to determine the data types used.

* Syntax analysis

It is the compilation stage in which after the tokens have been assigned to the code elements, then the compiler determines if the tokens are in the correct order and follow the rules of the language.

* Semantic analysis

Then, in this phase the analysis consists principally of type inference, which if successful, produces the most general type of an expression or declaration. Also type error messages may occur during this phase. This stage also detects whether any members of a sequence are not of type unit.

* Code generation

After the stages of lexical and syntax analysis, the code generation appears. In this process, the code generated is known as the object code, which is the binary equivalent of the source code, which is the executable version of the code.

* Optimisation

In this part, Code optimisation makes the program more efficient, so it runs faster and uses fewer resources.

**Levels of programming**

Each programming language contains a unique set of keywords and syntax, which are used to create a set of instructions. These languages vary in the level of abstraction they provide from the hardware. Some programming languages provide less or no abstraction while some provide higher abstraction.

They can be classified into two categories:

* **Low-level language**

It has a limited number of programming constructs, with selection and iteration being performed using ‘compare and branch’ instructions. This language is much harder to learn, and it takes too much time to code.

Low-level language can be also divided into:

**Machine language:**

It consists of a set of instructions that are in the binary form 0 or 1. In this level, the computers can understand only machine instructions, which are in binary digits, so the instructions given to the computer can be only in binary codes. It is not portable as each computer has its machine instructions.

**Assembly language:**

Assembly language instructions are written in English words like mov, add, sub, so it is easier to write and understand. We use it as a translator that converts the assembly code into machine code. It is not portable because the data is stored in computer registers, and the computer must know the different sets of registers. The assembly code is not faster than machine.

* **High-level language**

It 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; and a compiler is required to translate a high-level language into a low-level language.

Example of this level language:

-Python

-Visual Basic

-Java

-C++

**References**

Jaiswal, S. (NA). *Compilation process in c.* Retrieved from: https://www.javatpoint.com/compilation-process-in-c.

Jaiswal, S. (NA). *Classification of Programming Languages.* Retrieved from: https://www.javatpoint.com/classification-of-programming-languages.

Luks, C. (2015, August 7). *The Four Stages of Compiling a C Program.* Retrieved from: https://www.calleluks.com/the-four-stages-of-compiling-a-c-program/.

TeacherComputerScience. (2019). *The four stages of compilation.* Retrieved from: https://teachcomputerscience.com/the-four-stages-of-compilation/.

TeacherComputerScience. (2019). *The four stages of compilation.* Retrieved from: https://teachcomputerscience.com/the-four-stages-of-compilation/.

IsaacComputerScience. (NA). *Stages of compilation.* Retrieved from: https://isaaccomputerscience.org/concepts/sys\_trans\_stages.