Stages of program compilation

In programming language there are 4 general stages which are

Preprocessing

The first stage of compilation is called preprocessing. In this stage, lines starting with a # character are interpreted by the preprocessor as preprocessor commands. These commands form a simple macro language with its own syntax and semantics. This language is used to reduce repetition in source code by providing functionality to inline files, define macros, and to conditionally omit code.

Before interpreting commands, the preprocessor does some initial processing. This includes joining continued lines (lines ending with a \) and stripping comments.

Compilation

The second stage of compilation is confusingly enough called compilation. In this stage, the preprocessed code is translated to assembly instructions specific to the target processor architecture. These form an intermediate human readable language.

The existence of this step allows for C code to contain inline assembly instructions and for different assemblers to be used.

Some compilers also supports the use of an integrated assembler, in which the compilation stage generates machine code directly, avoiding the overhead of generating the intermediate assembly instructions and invoking the assembler.

Assembly

During this stage, an assembler is used to translate the assembly instructions to object code. The output consists of actual instructions to be run by the target processor.

Linking

The object code generated in the assembly stage is composed of machine instructions that the processor understands but some pieces of the program are out of order or missing. To produce an executable program, the existing pieces have to be rearranged and the missing ones filled in. This process is called linking.

The linker will arrange the pieces of object code so that functions in some pieces can successfully call functions in other ones. It will also add pieces containing the instructions for library functions used by the program. In the case of the "Hello, World!" program, the linker will add the object code for the puts function.

Levels of programming

Low level

A low-level or first-generation function 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.

The use of the word low in its name does not imply that the language is less powerful than a high-level language, but rather refers to the reduced abstraction between the language and the hardware.

Mid level

Mid-level language is a computer programming language such as the C language, which are among the high-level languages and the low-level languages.

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

High level

The high-level language is one that is closer to human natural language than to the binary language of computers, which is known as low-level language.

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 a specific hardware. The only condition is that the PC has a program known as

Keys to improve

When we speak of low-level language we are not referring to a specific one, in fact this term encompasses three different types of low-level language, although they all share similar characteristics.

The first one we come across is the famous Binary Code. Surely you have heard of it on more than one occasion and it is the most basic language that is part of all computer systems.

In a second step is machine language. Also widely used since, as its name indicates, this is going to be the code by which instructions are going to be communicated to the machine.

Finally, we have assembly language, something more complicated because the codes it uses are not decrypted directly by the computer, so it will have to be converted to machine language so that the computer understands the order that we are trying to transmit to it.

References

https://nivelesdeprogramacion.blogspot.com/2019/07/niveles-de-programacion.html

https://www.calleluks.com/the-four-stages-of-compiling-a-c-program/

https://www.bbc.co.uk/bitesize/guides/zmthsrd/revision/3