Translators: Compilers and Interpreters

A computer is an integrated collection of hardware components, which are capable of executing instructions called “object code” stored in a computer's memory component. The computer's control component takes the object code stored as a string of binary bits (i.e. 0's and 1's), converts the bits to voltage levels, and transmits the voltage levels to its hardware components which carry out, or execute, operations as directed by the voltages, as specified in the object code. The steps of conversion, transmission, and execution by hardware components are called interpretation.

Writing instructions in object code (also called the target language, machine language, or a low level language for some computer hardware) is not easy for humans to do. A low level target language is a language appropriate for computer hardware, not for humans. Languages that are appropriate for the problems that humans work on are called higher level or source languages. Many higher level source languages have been defined over the years, including Java and C++.

When humans write computations in a source language, there needs to be a way to translate that source language to the low level target language of the computer that is being used. This is the task of a compiler - namely, to translate source language instructions to a target machine language. This is not a trivial task. A collection of source language instructions, called a source program, is translated in stages by a compiler to a target language program. Typical, the stages performed by a compiler are: 1) analysis of the source program to check that its grammar or syntax has no errors, 2) parse the source program to identify its syntactic elements, e.g. operations and operands, and 3) organize the elements into an overall structure and modify this syntax structure to optimize it (e.g. removing redundancies and converting the syntax structure to target language instructions). The target language program is then saved in the computer memory and if it is at the level of the machine instructions it can be executed i.e. interpreted by the computer hardware to perform the computations specified in the source program. If the translation is done correctly and the results of the operations are correct, we say that the source program and the target program have the same semantics, or meaning. Thus, the compilation is a translation process which is followed by an interpretation process.

So, a compiler translates a program from a high level language to, usually, low level machine instructions. The translated program can be saved and executed at a later time. Note also that the translated program is much larger than the source program. This is because the source program is at a higher level and the target program is at a lower level. For example, a high level operation might be to multiply while a low level operation involves manipulation of binary bits. Suppose your computer hardware only has a binary adder; the high level multiply instruction would have to be translated into computations that perform multiplication by manipulating bits and using the adder. One multiply instruction might be translated into up to ten or more machine instructions.

An interpreter is also a translator, but instead of translating the source program to a target program, it interprets each source instruction and produces the results e.g. the multiplication result (as opposed to the target instructions to compute the multiplication result, which is what a compiler would produce). There is no saved translation or target program, only the results of the computations. This assumes that as the interpreter is analyzing the source program it can produce the appropriate binary code for the voltages that direct the operation of the hardware components.

An interpreter is faster than a compiler because it has fewer stages e.g. no optimization stage; and it produces the results of the computations as it translates the source instructions. An interpreter is also simpler than a compiler, but the computations specified in the source program are carried out less efficiently.

A compiler is more complicated than an interpreter, simply because it has more stages for doing analysis and optimizations, and because it focuses on the translation of the entire program (whereas an interpreter works on one instruction as a time). However, the resulting target program translation is more efficient when it is executed.

In practice, a combination of compliers and interpretation is typically used. For example, a compiler translates a source language to an intermediate language, which is at an in-between level between the high level language and the low level language, and then an interpreter executes the intermediate language to a low level language.