classical examples. You may encounter the term in other texts and explanations of multilevel design. Virtual machines can be implemented in hardware or software.

A program written in a high-level language must be translated in order to run at a lower level. Translation has two forms: (1) **interpreting**, which is translating line-by-line as the program executes, and (2) **compiling**, which is translating all the code in a single step before execution. An example of a high-level interpreted language is Python. An example of a high-level compiled language is C++.

A high-level language statement such as `int x = y + 2;` has a one-to-many relationship with machine language; that is, each high-level statement corresponds to multiple Assembly and machine instructions.

Assembly is an intermediate form in the translation pipeline. A high-level program is translated into Assembly and the Assembly is further translated into machine language based on a processor's instruction set architecture. The process of Assembly code being translated into machine language is often called encoding. The inverse process of machine language being translated into Assembly is often called decoding.

Assembly language statements have a one-to-one relationship with machine language; that is, each Assembly language instruction corresponds to a single machine language instruction. Machine language instructions are typically represented in an intermediate numeric form such as hexadecimal, which is then translated and implemented physically as binary on the hardware, what we call digital logic.

PROGRAMMING: Some compilers, such as Microsoft's C compiler, translate high-level code directly to machine language, and creating Assembly is an option. Other compilers, such as GCC, translate to Assembly first, then the Assembly code is translated to machine code.

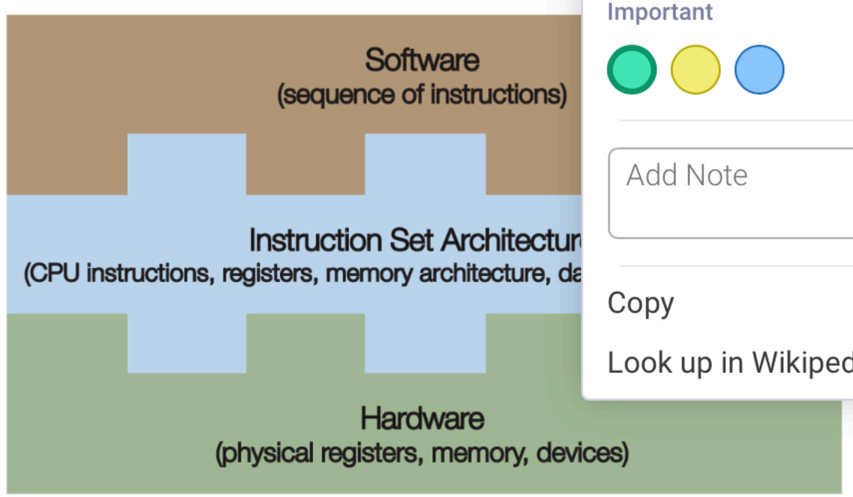

Figure 1.3 Instruction set architecture

An **instruction set architecture (ISA)** is the programming-related aspect of **computer architecture**. The ISA specifies the instructions, registers, memory architecture, data types, and other attributes native to a particular processor that are available to a programmer. Think of an ISA as the language a computer speaks. As shown in Figure 1.3, an ISA facilitates communication between software and hardware.

We can think of instruction set architectures as complex or reduced. **Complex instruction set computing (CISC)** architectures have instructions that are of varying length (in bytes) and are complex in the sense that a single instruction may perform more than one task (e.g., access a memory location and perform arithmetic). An alternative ISA design is **reduced instruction set computing (RISC)** in which all instructions are the same length and perform only one task (e.g., access a memory location).

x86 and x86_64 are CISC architectures, while the majority of other ISAs are RISC architectures. CISC attributes of x86 will be shown

Important

Add Note

Copy

Look up in Wikipedia