Assembly is further translated into machine language based on a processor's instruction set architecture. The pr
code being translated into machine language is often called encoding. The inverse process of machine languag
into Assembly is often called decoding.

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

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

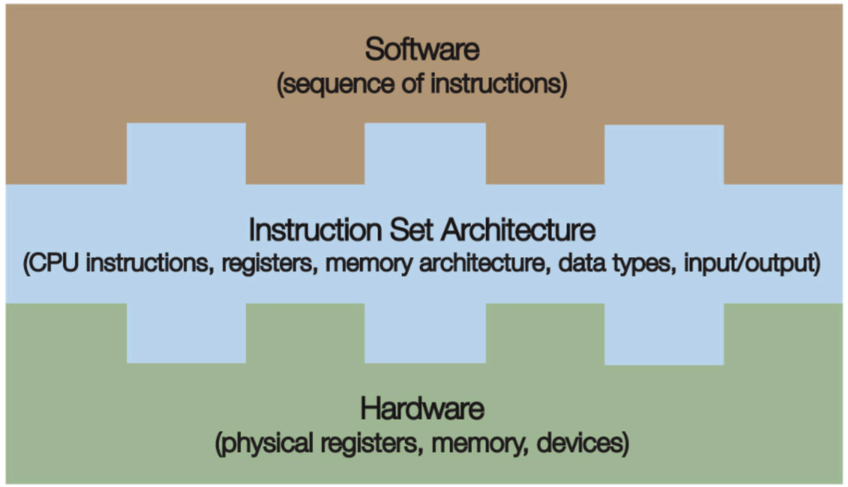

Figure 1.3 Instruction set architecture

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 when instructions are discussed in more detail in **Chapters 4 and 5**. Also, when **disassembly** examples are provided, such as in **Chapters 6** and **11**, the varying length and complexity of instructions will be evident. You can also see variable length instructions in the **Chapter 1 page 7** PROGRAMMING note on relocatable machine language.