2.1 Introduction

> "I speak Spanish to God, Italian to women, French to men, and German to my horse."
> *Charles V, Holy Roman Emperor, (1500-1558)*

To command a computer's hardware, you must speak its language. The words of a computer's language are called instructions, and its vocabulary is called an *instruction set*. In this chapter, you will see the instruction set of a real computer, both in the form written by people and in the form read by the computer. We introduce instructions in a top-down fashion. Starting from a notation that looks like a restricted programming language, we refine it step-by-step until you see the actual language of a real computer. COD Chapter 3 (Arithmetic for Computers) continues our downward descent, unveiling the hardware for arithmetic and the representation of floating-point numbers.

Instruction set: The vocabulary of commands understood by a given architecture.

You might think that the languages of computers would be as diverse as those of people, but in reality, computer languages are quite similar, more like regional dialects than independent languages. Hence, once you learn one, it is easy to pick up others.

The chosen instruction set is ARMv8, which comes from ARM Holdings plc and was announced in 2011. For pedagogic reasons, we will use a subset of ARMv8 instructions in this book. We will use the term ARMv8 when talking about the original full instruction set, and LEGv8 when referring to the teaching subset, which is of course based on ARM's ARMv8 instruction set. (LEGv8 is intended as a pun on ARMv8, but it is also a backronym for "Lessen Extrinsic Garrulity.") We'll identify differences between the two in the elaborations. Note that this chapter and several others have a section to give an overview of the rest of the features in ARMv8 that are not in LEGv8 (see COD Sections 2.19 (Real Stuff: The Rest of the ARMv8 Instruction Set), 3.8 (Real Stuff: The Rest of the ARMv8 Arithmetic Instructions), and 5.14 (Real Stuff: The Rest of the ARMv8 System and Special Instructions)).

To demonstrate how easy it is to pick up other instruction sets, we will take a quick look at three other popular instruction sets.

1. MIPS is an elegant example of the instruction sets designed since the 1980s.
2. ARMv7 is an older instruction set also from ARM Holdings plc, but with 32-bit addresses instead of ARMv8's 64 bits. More than 14 billion chips with ARM processors were manufactured in 2015, making them the most popular instruction sets in the world. Ironically, in the authors' opinion, and as shall be seen, ARMv8 is closer to MIPS than it is to ARMv7.
3. The final example is the Intel x86, which powers both the PC and the Cloud of the post-PC era.

This similarity of instruction sets occurs because all computers are constructed from hardware technologies based on similar underlying principles and because there are a few basic operations that all computers must provide. Moreover, computer designers have a common goal: to find a language that makes it easy to build the hardware and the compiler while maximizing performance and minimizing cost and energy. This goal is time-honored; the following quote was written before you could buy a computer, and it is as true today as it was in 1947:

> "It is easy to see by formal-logical methods that there exist certain [instruction sets] that are in abstract adequate to control and cause the execution of any sequence of operations .... The really decisive considerations from the present point of view, in selecting an [instruction set], are more of a practical nature: simplicity of the equipment demanded by the [instruction set], and the clarity of its application to the actually important problems together with the speed of its handling of those problems.
> *Burks, Goldstine, and von Neumann, 1947*

The "simplicity of the equipment" is as valuable a consideration for today's computers as it was for those of the 1950s. The goal of this chapter is to teach an instruction set that follows this advice, showing both how it is represented in hardware and the relationship between high-level programming languages and this more primitive one. Our examples are in the C programming language; COD Section 2.15 (Advanced material: Compiling C and interpreting Java) shows how these would change for an object-oriented language like Java.

By learning how to represent instructions, you will also discover the secret of computing: the *stored-program concept*. Moreover, you will exercise your "foreign language" skills by writing programs in the language of the computer and running them on the simulator that comes with this book. You will also see the impact of programming languages and compiler optimization on performance. We conclude with a look at the historical evolution of instruction sets and an overview of other computer dialects.

Stored-program concept: The idea that instructions and data of many types can be stored in memory as numbers and thus be easy to change, leading to the stored-program computer.

We reveal our first instruction set a piece at a time, giving the rationale along with the computer structures. This top-down, step-by-step tutorial weaves the components with their explanations, making the computer's language more palatable.

PARTICIPATION ACTIVITY 2.1.1: Instruction sets.

1) An instruction set is a particular program provided in the language of a computer.
   - ○ True
   - ○ False

2) The instruction sets of different computers are quite similar to one another.
   - ○ True
   - ○ False

3) Instructions, as well as data, can be stored in memory as numbers.
   - ○ True
   - ○ False

Provide feedback on this section