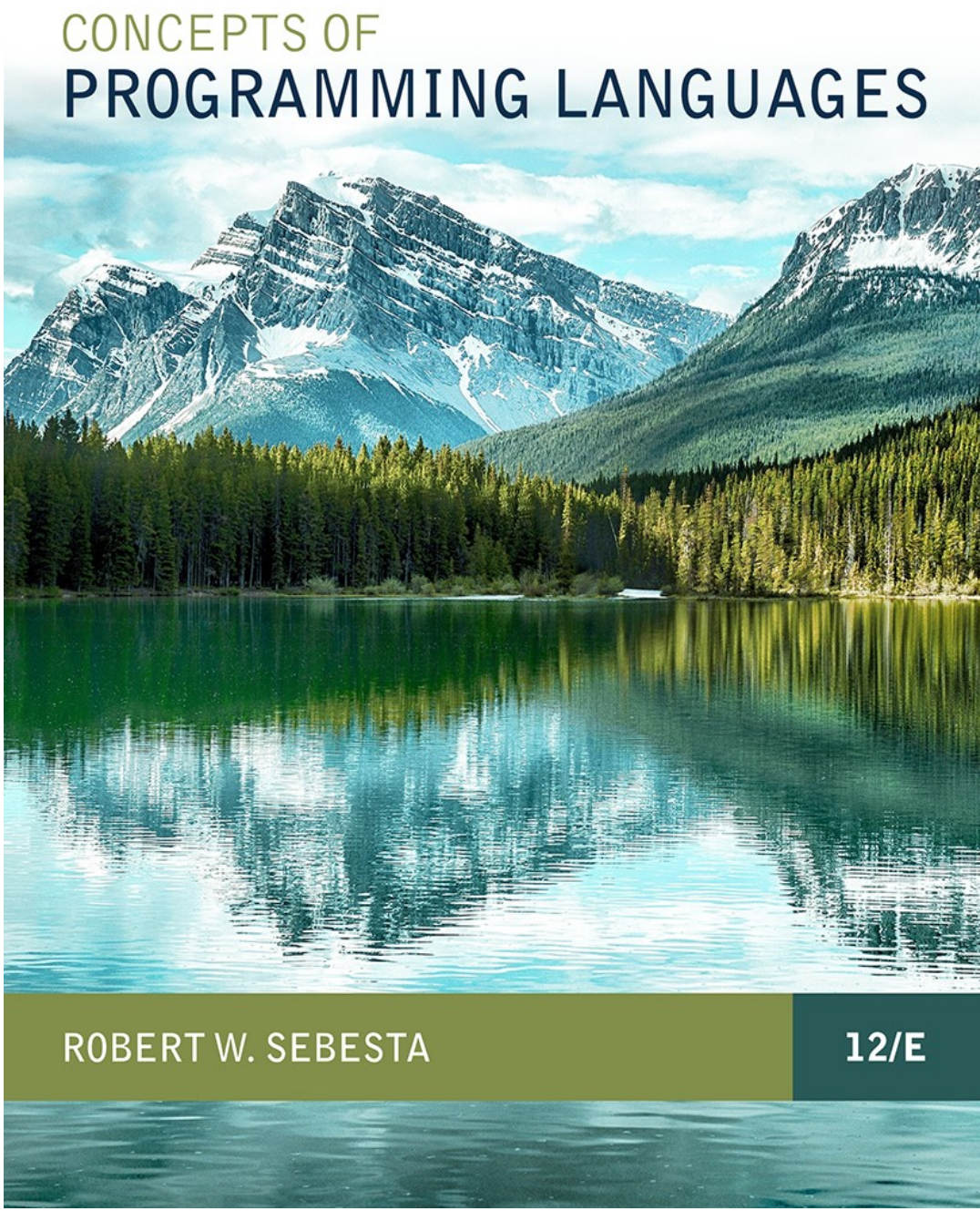
Chapter 6

Data Types



ISBN 0-321-49362-1

Chapter 6 Topics

- Introduction
- Primitive Data Types
- Character String Types
- Enumeration Types
- Array Types
- Associative Arrays
- Record Types
- Tuple Types
- List Types
- Union Types
- Pointer and Reference Types
- Optional Types
- Type Checking
- Strong Typing
- Type Equivalence
- Theory and Data Types

Introduction

- A *data type* defines a collection of data objects and a set of predefined operations on those objects
- A *descriptor* is the collection of the attributes of a variable
- An *object* represents an instance of a user-defined (abstract data) type
- One design issue for all data types: What operations are defined and how are they specified?

Primitive Data Types

- Almost all programming languages provide a set of *primitive data types*
- Primitive data types: Those not defined in terms of other data types
- Some primitive data types are merely reflections of the hardware
- Others require only a little non-hardware support for their implementation

Primitive Data Types: Integer

- Almost always an exact reflection of the hardware so the mapping is trivial
- There may be as many as eight different integer types in a language
- Java's signed integer sizes: **`byte`**, **`short`**, **`int`**, **`long`**

Python

- Integer arithmetic operations in Python that produce values too large to be represented with int type store them as long integer type values.

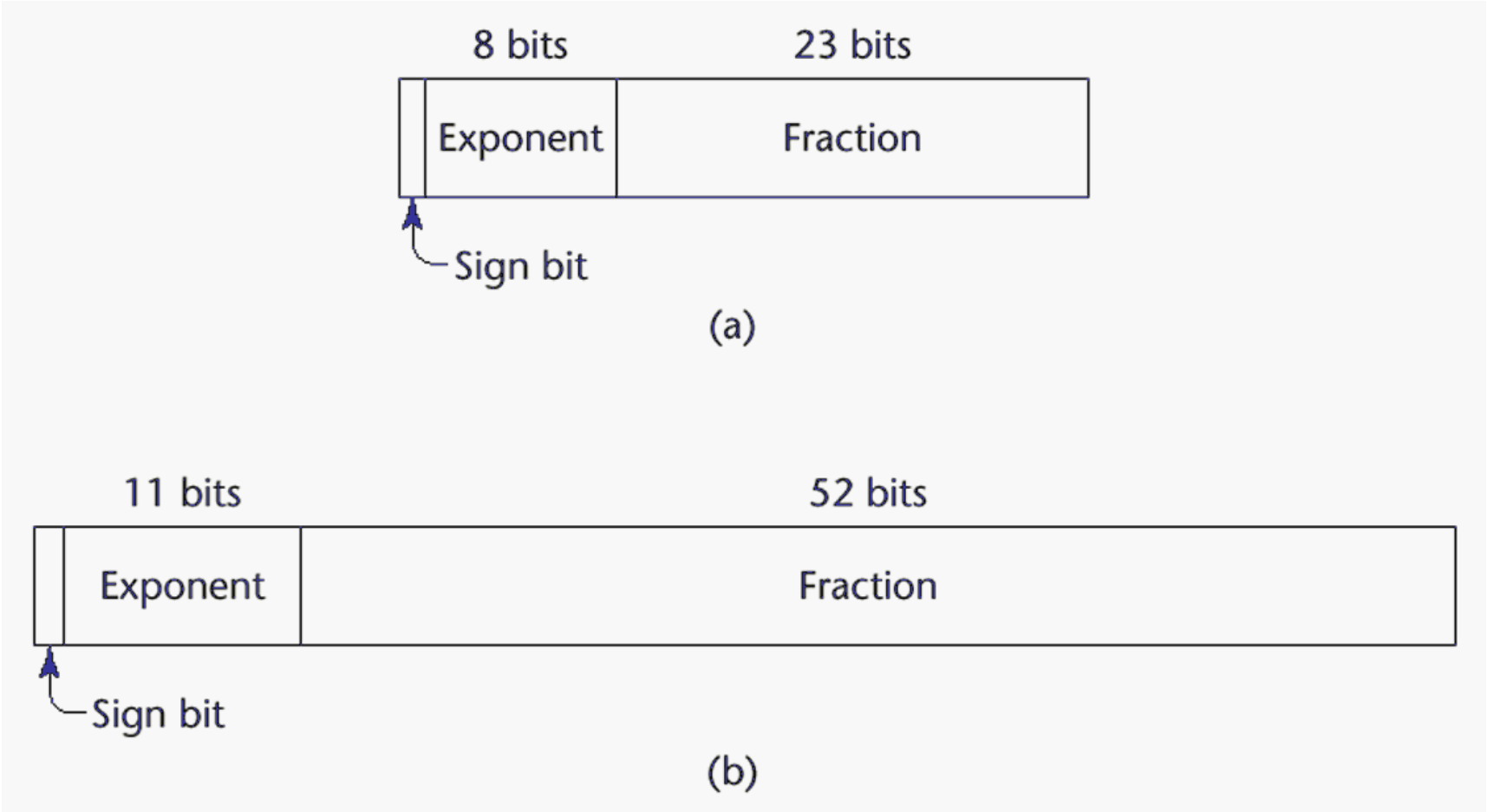
Primitive Data Types: Floating Point

- Model real numbers, but only as approximations
- Languages for scientific use support at least two floating-point types (e.g., **`float`** and **`double`**; sometimes more
- Usually exactly like the hardware, but not always

Primitive Data Types: Floating Point

- On most computers, floating-point numbers are stored in binary, which exacerbates the problem. For example, even the value 0.1 in decimal cannot be represented by a finite number of binary digits.1 Another problem with floating-point types is the loss of accuracy through arithmetic operations

IEEE Floating-Point Standard 754



Primitive Data Types: Decimal

- For business applications (money)
  - Essential to COBOL
  - C# offers a decimal data type
- Store a fixed number of decimal digits, in coded form (BCD)
- *Advantage*: accuracy
- *Disadvantages*: limited range, wastes memory

Primitive Data Types: Boolean

- Simplest of all
- Range of values: two elements, one for "true" and one for "false"
- Could be implemented as bits, but often as bytes
  - Advantage: readability

Primitive Data Types: Character

- Stored as numeric codings
- Most commonly used coding: ASCII
- An alternative, 16-bit coding: Unicode (UCS-2)
  - Includes characters from most natural languages
  - Originally used in Java
  - Now supported by many languages
- 32-bit Unicode (UCS-4)
  - Supported by Fortran, starting with 2003