C++ Programming: From Problem Analysis to Program Design, Sixth Edition

Chapter 1: An Overview of Computers and Programming Languages

Elements of a Computer System

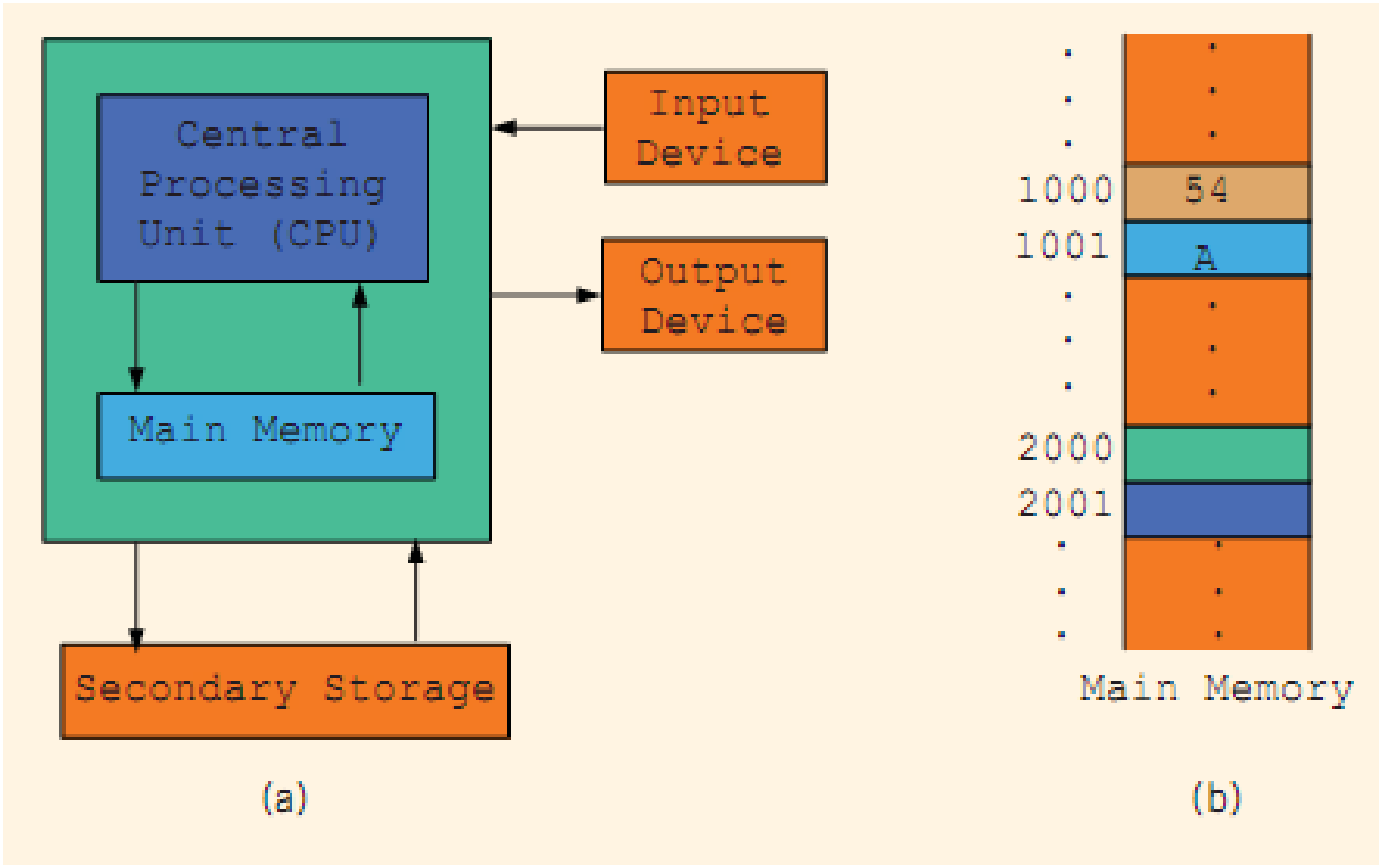

FIGURE 1-1 Hardware components of a computer and main memory

Elements of a Computer System

Digital signals are processed inside a computer, the language of a computer, called machine language, is a sequence of 0s and 1s. The digit 0 or 1 is called a binary digit, or bit. Sometimes a sequence of 0s and 1s is referred to as a binary code or a binary number.

Main memory,or random access memory, is connected directly to the CPU. Main memory is an ordered sequence of cells, **called memory cells**. Each cell has a unique location in main memory, called the address of the cell. These addresses help you access the information stored in the cell. Figure 1-1(b) shows main memory with some data.

The device that stores information **permanently** (unless the device becomes unusable or you change the information by rewriting it) is called **secondary storage**.

Software

Software are programs written to perform specific tasks. For example, word processors are programs that you use to write letters, papers, and even books. All software is written in programming languages.
There are two types of programs: system programs and application programs.

System programs control the computer. The system program that loads first when you turn on your PC is called the operating system. Without an operating system, the computer is useless. The operating system monitors the overall activity of the computer and provides services. Some of these services include memory management, input/output activities, and storage management.

Application programs perform a specific task. Word processors, spreadsheets, and games are examples of application programs. The operating system is the program that runs application programs.

The Evolution of Programming Languages (cont'd.)

- High-level languages include Basic, FORTRAN, COBOL, Pascal, C, C++, C#, and Java
- Compiler: A program translates a program written in a high-level language in machine language
- Assembler: A program that translates a program written in assembly language into an equivalent program in machine language

```
LOAD   rate
MULT   hours
STOR   wages
```

Assembly Language

```
wages = rate * hours;
```

High level language

The Evolution of Programming Languages (cont'd.)

TABLE 1-2 Examples of Instructions in Assembly Language and Machine Language

Assembly Language	Machine Language
LOAD	100100
STOR	100010
MULT	100110
ADD	100101
SUB	100011

Processing a C++ Program

```
#include <iostream>
using namespace std;
int main()
{
    cout << "My first C++ program." << endl;
    return 0;
}
```

Sample Run:

```
My first C++ program.
```

Processing a C++ Program (cont'd.)

- To execute a C++ program:
  - Use a text editor to create a <u>source program</u> in C++.
  - Preprocessor directives begin with # and are processed by a the <u>preprocessor</u>
  - Use the <u>compiler</u> to:
    - Check that the program obeys the rules
    - Translate into machine language (<u>object program</u>)

Processing a C++ Program (cont'd.)

- To execute a C++ program (cont'd.):
  - <u>Linker</u>:
    - Combines object program with other programs provided by the SDK to create executable code
  - <u>Loader</u>:
    - Loads executable program into main memory
  - The last step is to execute the program

Processing a C++ Program (cont'd.)

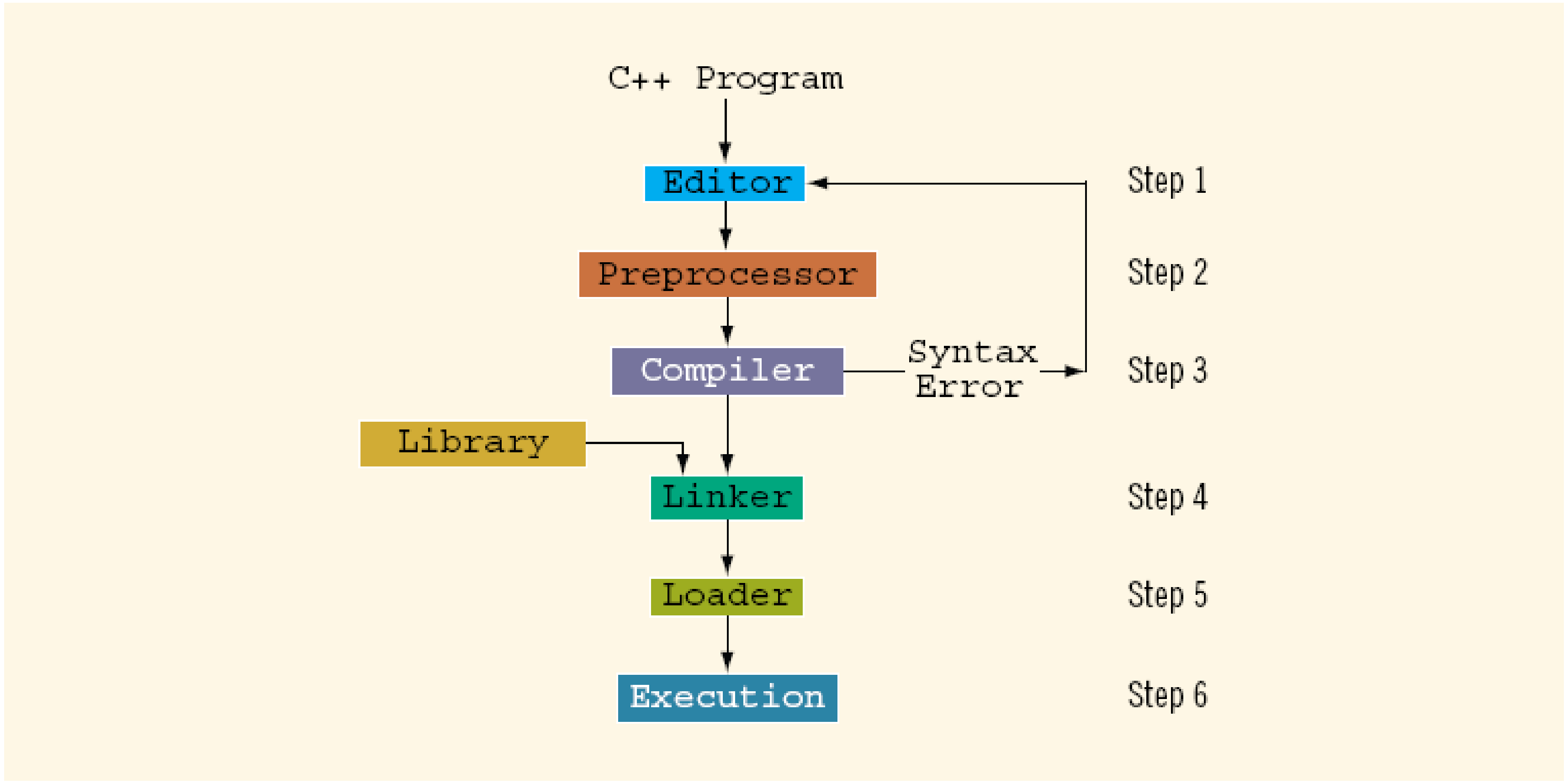

FIGURE 1-3 Processing a C++ program

Programming with the Problem Analysis–Coding–Execution Cycle

- Programming is a process of problem solving
- One problem-solving technique:
  - Analyze the problem
  - Outline the problem requirements
  - Design steps (algorithm) to solve the problem
- Algorithm:
  - A Step-by-step problem-solving process in which a solution is arrived at in a finite amount of time.

The Problem Analysis–Coding–Execution Cycle (cont'd.)

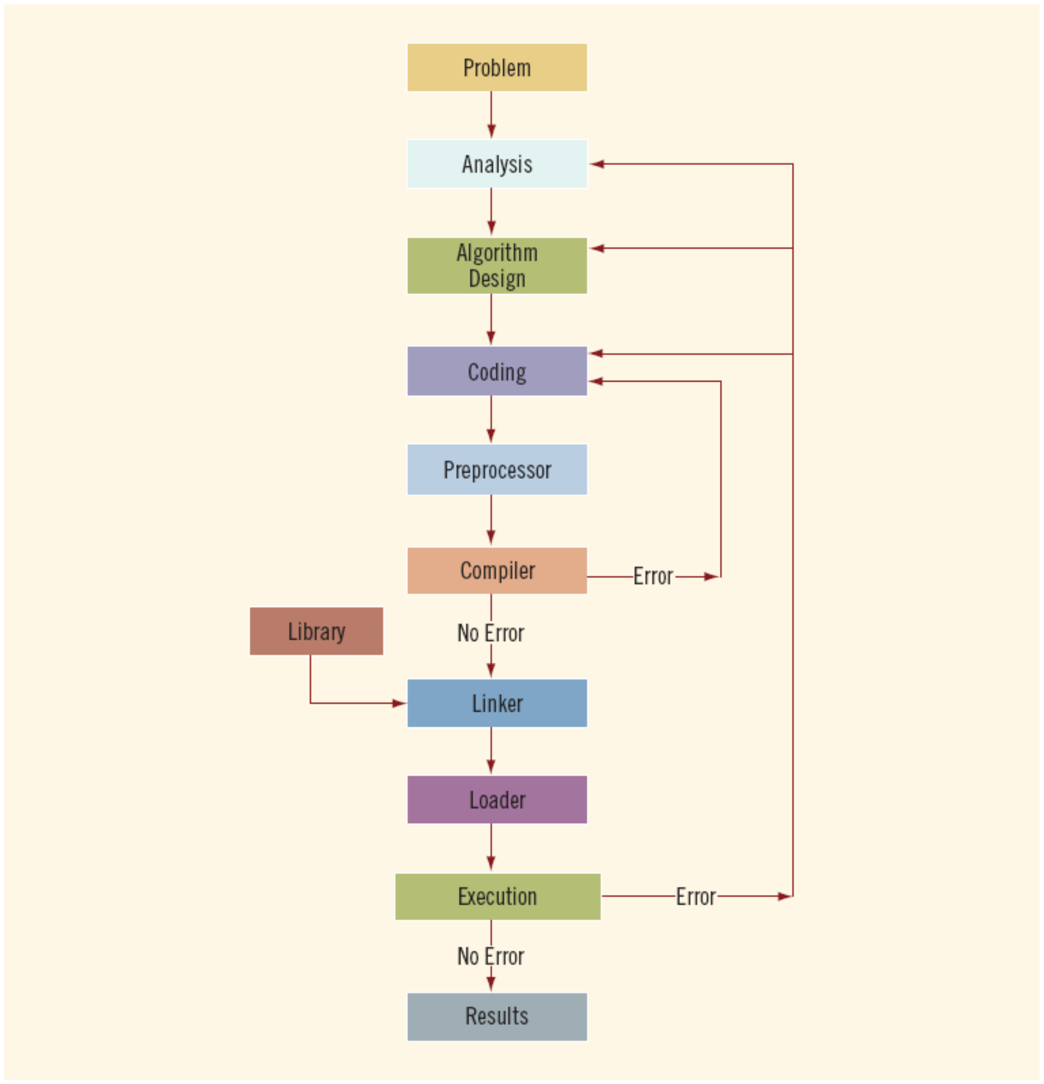

FIGURE 1-4 Problem analysis–coding–execution cycle

The Problem Analysis–Coding–Execution Cycle (cont'd.)

- Run code through compiler
- If compiler generates errors
  - Look at code and remove errors
  - Run code again through compiler
- If there are no syntax errors
  - Compiler generates equivalent machine code
- Linker links machine code with system resources

The Problem Analysis–Coding–Execution Cycle (cont'd.)

- Once compiled and linked, loader can place program into main memory for execution
- The final step is to execute the program
- Compiler guarantees that the program follows the rules of the language
  - Does not guarantee that the program will run correctly

Example 1-1

- Design an algorithm to find the perimeter and area of a rectangle
- The perimeter and area of the rectangle are given by the following formulas:

```
perimeter = 2 * (length + width)
area = length * width
```

Example 1-1 (cont'd.)

- Algorithm:
  - Get length of the rectangle
  - Get width of the rectangle
  - Find the perimeter using the following equation:

    ```
    perimeter = 2 * (length + width)
    ```

  - Find the area using the following equation:

    ```
    area = length * width
    ```

Programming Methodologies

- Two popular approaches:
  - Structured approach and
  - Object-oriented approach

Programming Methodologies (contd.)

- Structured approach

Dividing a problem into smaller subproblems is called structured design. Each subproblem is then analyzed, and a solution is obtained to solve the subproblem. The solutions to all of the subproblems are then combined to solve the overall problem. This process of implementing a structured design is called structured programming. The structured-design approach is also known as top-down design, bottom-up design, stepwise refinement, and modular programming.

Programming Methodologies (contd.)

- Object Oriented Design

In OOD, the first step in the problem-solving process is to identify the components called objects, which form the basis of the solution, and to determine how these objects interact with one another.

For example, suppose you want to write a program that automates the video rental process for a local video store. The two main objects in this problem are the video and the customer.

Programming Methodologies (contd.)

- Object Oriented Design

After identifying the objects, the next step is to specify for each object the **relevant data and possible operations** to be performed on that data. For example, for a video object, the data might include:

- movie name
- starring actors
- producer
- production company
- number of copies in stock

Programming Methodologies (contd.)

- Object Oriented Design

Some of the operations on a video object might include:

• checking the name of the movie

• reducing the number of copies in stock by one after a copy is rented

• incrementing the number of copies in stock by one after a customer returns a particular video