Introduction to Computers and C

CEN111 Algorithms and Programming I

Introduction

- C is one of the world's most senior computer programming languages and, according to the Tiobe Index, one of the world's most popular
- Software (that is, the C instructions you write, which are also called code) controls hardware (that is, computers and related devices)
- C is widely used in industry for a wide range of tasks
- Today's popular desktop operating systems—Windows, macOS and Linux—are partially written in C
- Many popular applications are partially written in C, including popular web browsers (e.g., Google Chrome and Mozilla Firefox), database management systems (e.g., Microsoft SQL Server, Oracle and MySQL) and more

Introduction

- Many development environments are available in which you can compile, build and run C applications
- You can compile and execute C code using:
  - Microsoft Visual Studio Community edition for Windows
  - Clang in X code on mac OS
  - GNU GCC in a shell on Linux

Hardware and Software

- Computers can perform calculations and make logical decisions phenomenally faster than human beings can
- Today’s personal computers and smartphones can perform billions of calculations in one second—more than a human can perform in a lifetime
- Supercomputers already perform thousands of trillions (quadrillions) of instructions per second!

Hardware and Software

- Computers process data under the control of sequences of instructions called computer programs (or simply programs)
- Specified by people called computer programmers
- A computer consists of various physical devices referred to as hardware
  - keyboard, screen, mouse, hard disks, memory, and processing units
- Computing costs are dropping dramatically due to rapid developments in hardware and software technologies
  - Computers that might have filled large rooms and cost millions of dollars decades ago are now inscribed on silicon computer chips smaller than a fingernail, costing perhaps a few dollars each
  - Silicon-chip technology has made computing so economical that computers and computerized devices have become commodities.

Moore's Law

- Every year, you probably expect to pay at least a little more for most products and services
- The opposite has been the case in the computer and communications fields
- Over the years, hardware costs have fallen rapidly
- For decades, every couple of years, computer processing power approximately doubled inexpensively
- This trend often is called Moore's Law
  - Named for Gordon Moore, co-founder of Intel and the person who identified this trend in the 1960s

Moore’s Law

- Key executives at computer-processor companies NVIDIA and Arm have indicated that Moore’s Law no longer applies
- Computer processing power continues to increase but relies on new processor designs, such as multicore processors
- Moore’s Law and related observations apply especially to
  - the amount of memory that computers have for programs,
  - the amount of secondary storage (such as hard disks and solid-state drive storage) they have to hold programs and data, and
  - their processor speeds—that is, the speeds at which computers execute programs to do their work.

Computer Organization

- Regardless of physical differences, computers can be envisioned as divided into various logical units or sections
- Input Unit—This “receiving” section obtains information (data and computer programs) from input devices and places it at the other units’ disposal for processing
- Output Unit—This “shipping” section takes information the computer has processed and places it on various output devices to make it available outside the computer

Computer Organization

- Memory Unit—This rapid-access, relatively low-capacity “warehouse” section retains information entered through the input unit, making it immediately available for processing when needed.
  - The memory unit also retains processed information until it can be placed on output devices by the output unit
  - Information in the memory unit is volatile—it’s typically lost when the computer’s power is turned off
  - The memory unit is often called either memory, primary memory or RAM (Random Access Memory). Main memories on desktop and notebook computers contain as much as 128 GB of RAM, though 8 to 16 G B is most common. GB stands for gigabytes; a gigabyte is approximately one billion bytes. A byte is eight bits. A bit (short for “binary digit”) is either a 0 or a 1.

Computer Organization

- Arithmetic and Logic Unit (ALU)—This “manufacturing” section performs calculations (e.g., addition, subtraction, multiplication and division) and makes decisions (e.g., comparing two items from the memory unit to determine whether they’re equal)
  - Part of the CPU
- Central Processing Unit (CPU)—This “administrative” section coordinates and supervises the operation of the other sections
  - Tells the input unit when to read information into the memory unit
  - Tells the ALU when to use information from the memory unit in calculations
  - Tells the output unit when to send information from the memory unit to specific output devices

Computer Organization

- Most computers today have multicore processors that economically implement multiple processors on a single integrated circuit chip
  - Such processors can perform many operations simultaneously
  - A dual-core processor has two CPUs, a quad-core processor has four and an octa-core processor has eight.

Computer Organization

- Secondary Storage Unit—This is the long-term, high-capacity “warehousing” section
  - Programs and data not actively being used by the other units are placed on secondary storage devices until they’re again needed, possibly hours, days, months or even years later
  - Information on secondary storage devices is persistent
  - Secondary storage information takes much longer to access than information in primary memory, but its cost per byte is much less
  - Examples of secondary storage devices include solid-state drives (SSDs), USB flash drives, hard drives and read/write Blu-ray drives
  - Many current drives hold terabytes (TB) of data
  - A terabyte is approximately one trillion bytes

Languages

- Programmers write instructions in various programming languages, some directly understandable by computers and others requiring intermediate translation steps
- Hundreds of such languages are in use today
- These may be divided into three general types:
  - Machine languages.
  - Assembly languages.
  - High-level languages.

Machine Languages

- Any computer can directly understand only its own machine language, defined by its hardware design
- Machine languages generally consist of strings of numbers (ultimately reduced to 1s and 0s) that instruct computers to perform their most elementary operations one at a time
- Machine languages are machine-dependent—a particular machine language can be used on only one type of computer
- Such languages are cumbersome for humans

Assembly Languages and Assemblers

- Programming in machine language was simply too slow and tedious for most programmers
- Instead of using the strings of numbers that computers could directly understand, programmers began using English-like abbreviations to represent elementary operations
- These abbreviations formed the basis of assembly languages
- Translator programs called assemblers were developed to convert assembly-language programs to machine language at computer speeds

High-Level Languages and Compilers

- To speed the programming process, high-level languages were developed in which single statements could accomplish substantial tasks
- A typical high-level-language program contains many statements, known as the program's source code
- Translator programs called compilers convert high-level-language source code into machine language
- High-level languages allow you to write instructions that look almost like everyday English and contain common mathematical notations
- C is among the world's most widely used high-level programming languages.

Interpreters

- Compiling a large high-level language program into machine language can take considerable computer time
- Interpreters execute high-level language programs directly
- Interpreters avoid compilation delays, but your code runs slower than compiled programs
- Some programming languages, such as Java and Python, use a clever mixture of compilation and interpretation to run programs

Operating Systems

- Operating systems are software that make using computers more convenient for users, software developers and system administrators
- Provide services that allow applications to execute safely, efficiently and concurrently with one another.
- The software that contains the core operating-system components is called the kernel
- Linux, Windows and mac O S are popular desktop computer operating systems
  - Each is partially written in C
- The most popular mobile operating systems used in smartphones and tablets are Google’s Android and Apple’s iOS

The C Programming Language

- C evolved from two earlier languages, BCPL and B
  - BCPL was developed in 1967 by Martin Richards as a language for writing operating systems and compilers
  - Ken Thompson modeled many features in his B language after their counterparts in BCPL, and in 1970 he used B to create early versions of the UNIX operating system at Bell Laboratories.
- The C language was evolved from B by Dennis Ritchie at Bell Laboratories and was originally implemented in 1972
- C initially became widely known as the development language of UNIX
- Many of today's leading operating systems are written in C and/or C++
- C is mostly hardware-independent—with careful design, it's possible to write C programs that are portable to most computers.

Built for Performance

- C is widely used to develop systems that demand performance
  - operating systems
  - embedded systems
  - real-time systems
  - communications systems:
- By the late 1970s, C had evolved into what's now referred to as "traditional C."

The C Standard Library and Open-Source Libraries

- C programs consist of pieces called functions
- You can program all the functions you need to form a C program
- However, most C programmers take advantage of the rich collection of existing functions in the C standard library
- Two parts to learning C programming:
  - learning the C language itself, and
  - learning how to use the functions in the C standard library.
- Throughout the book, we discuss many of these functions

The C Standard Library and Open-Source Libraries

- When programming in C, you'll typically use the following building blocks:
  - C standard library functions,
  - open-source C library functions,
  - functions you create yourself, and
  - functions other people have created and made available to you.
- Throughout the book, we focus on using the existing C standard library to leverage your program-development efforts and avoid "reinventing the wheel."
- This is called software reuse.

Open-Source Libraries

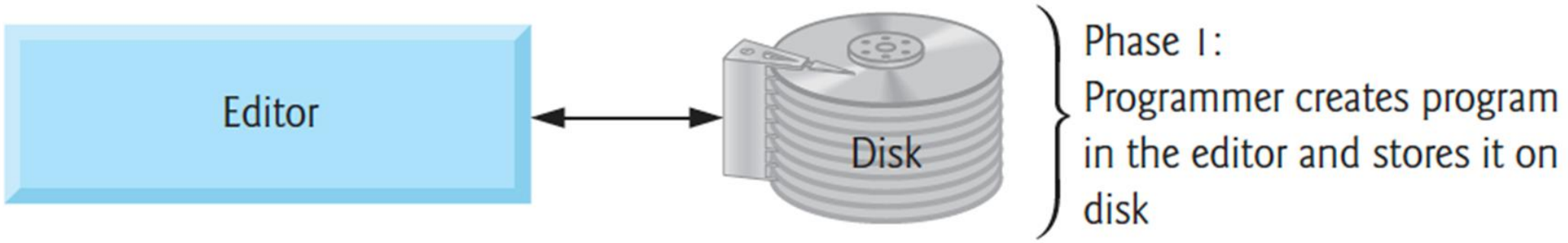
- There are enormous numbers of third-party and open-source C libraries that can help you perform significant tasks with modest amounts of code. GitHub lists over 32,000 repositories in their C category:
  - https://github.com/topics/c
- In addition, pages such as Awesome C provide curated lists of popular C libraries for a wide range of application areas.
  - https://github.com/kozross/awesome-c

Typical C Program-Development Environment

- C systems generally consist of several parts
  - a program-development environment
  - the language
  - the C standard library
- C programs typically go through six phases to be executed
  - Edit
  - Preprocess
  - Compile
  - Link
  - Load
  - Execute

Phase 1: Creating a Program

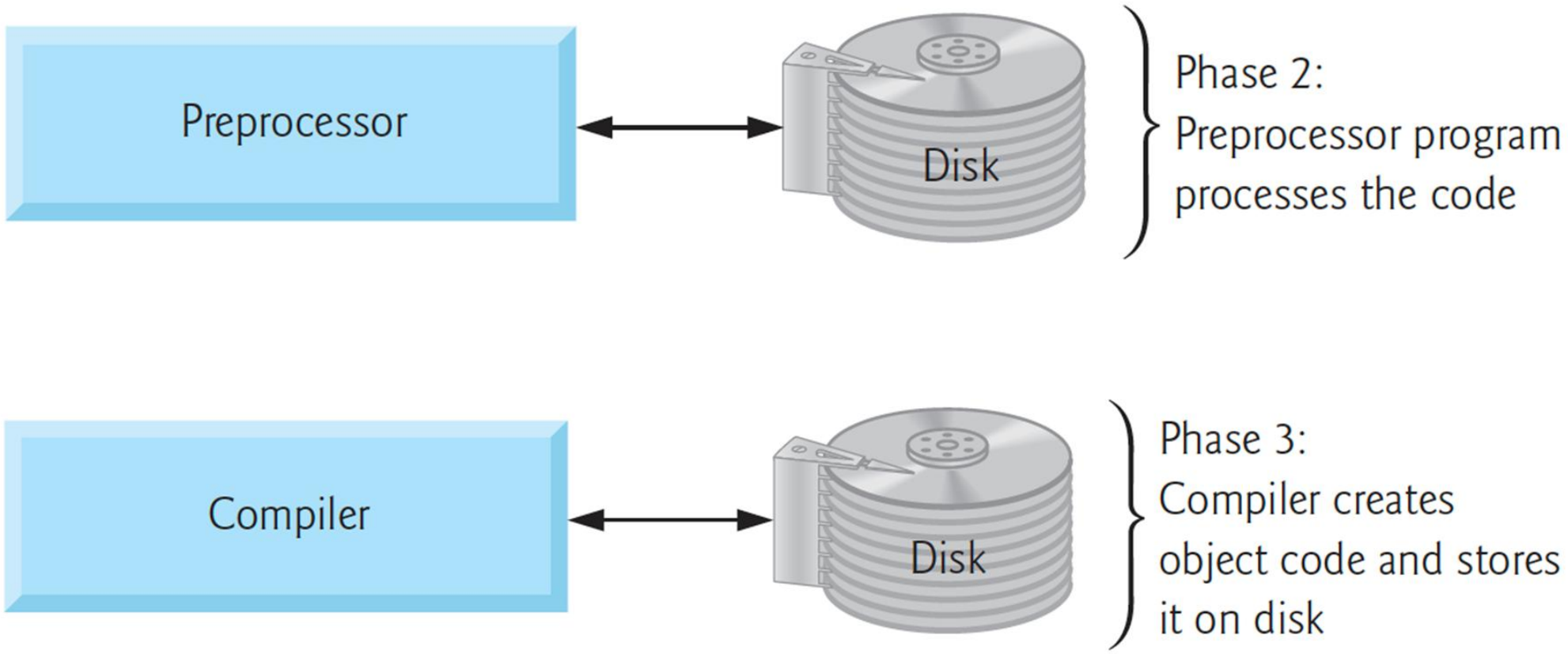
- Consists of editing a file in an editor program:



Phases 2 and 3: Preprocessing and Compiling a C Program

- You give the command to compile the program
- Compiler translates it into machine-language code
- The compilation command invokes a preprocessor first
  - Performs text manipulations on a program's source-code files
  - Inserting the contents of other files
  - Text replacements

Phases 2 and 3: Preprocessing and Compiling a C Program

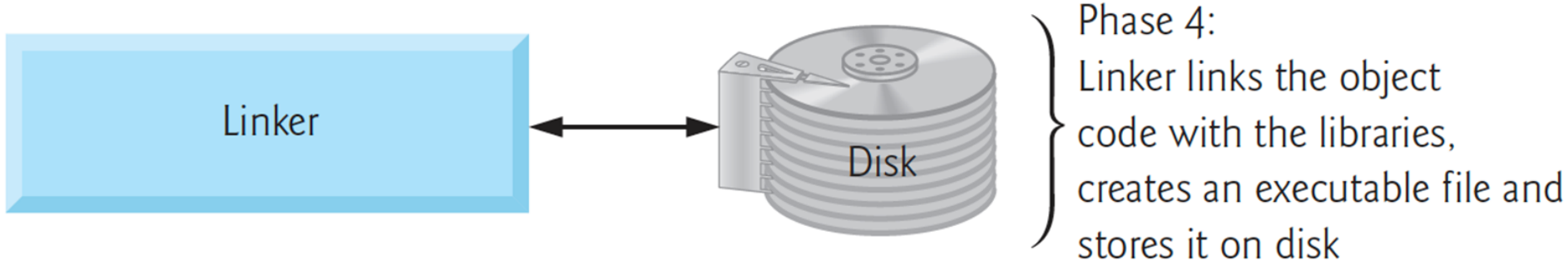


Phases 2 and 3: Preprocessing and Compiling a C Program

- A syntax error occurs when the compiler cannot recognize a statement because it violates the language rules
  - The compiler issues an error message to help you locate and fix the incorrect statement.
- The C standard does not specify the wording for error
- Syntax errors are also called compile errors or compile-time errors.

Phase 4: Linking

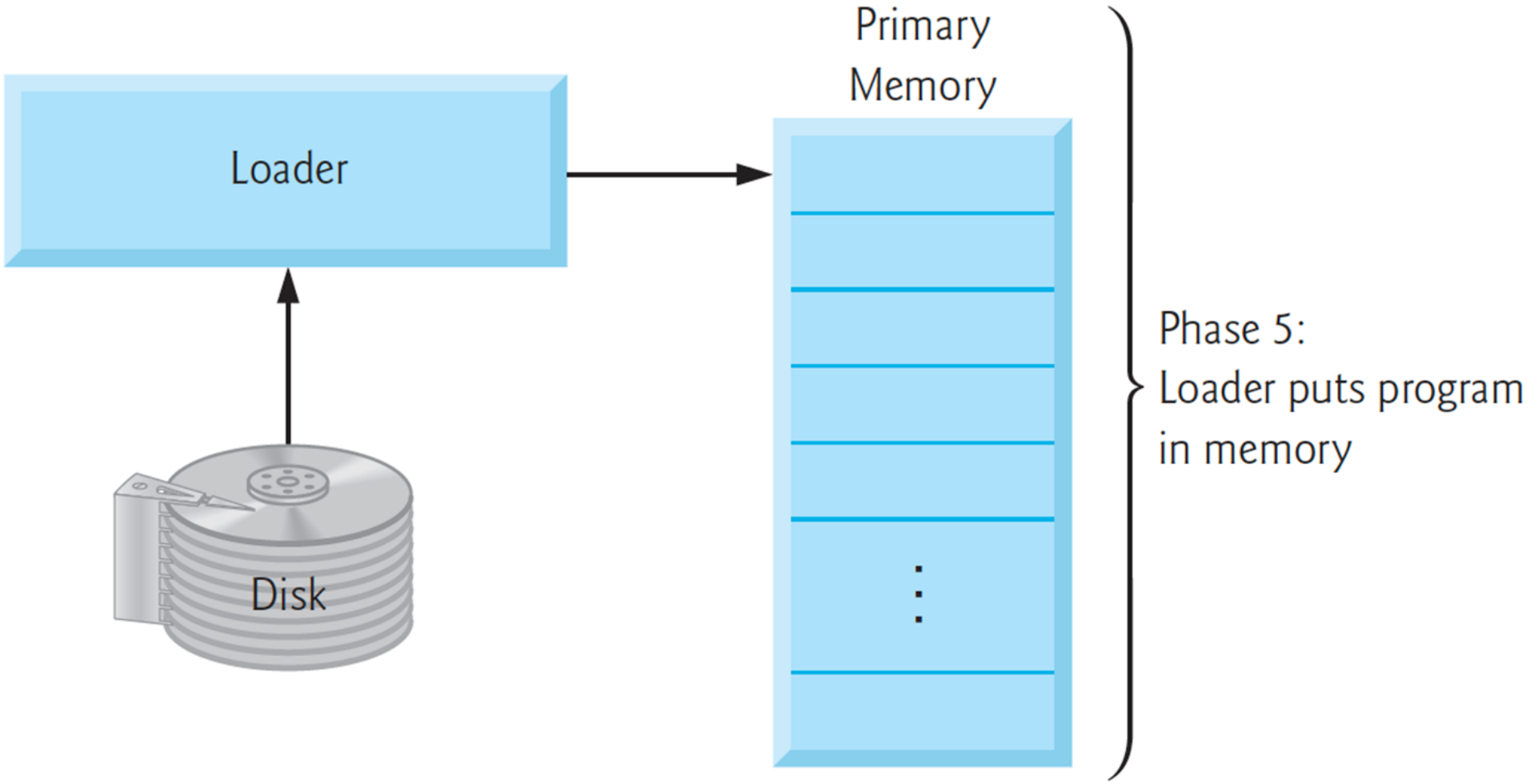
- C programs use functions defined elsewhere
  - standard libraries, open-source libraries or private libraries of a particular project.
- The object code produced by the C compiler typically contains “holes”
- Linker links a program’s object code with the code for the missing functions to produce an executable image (with no missing pieces)



Phase 5: Loading

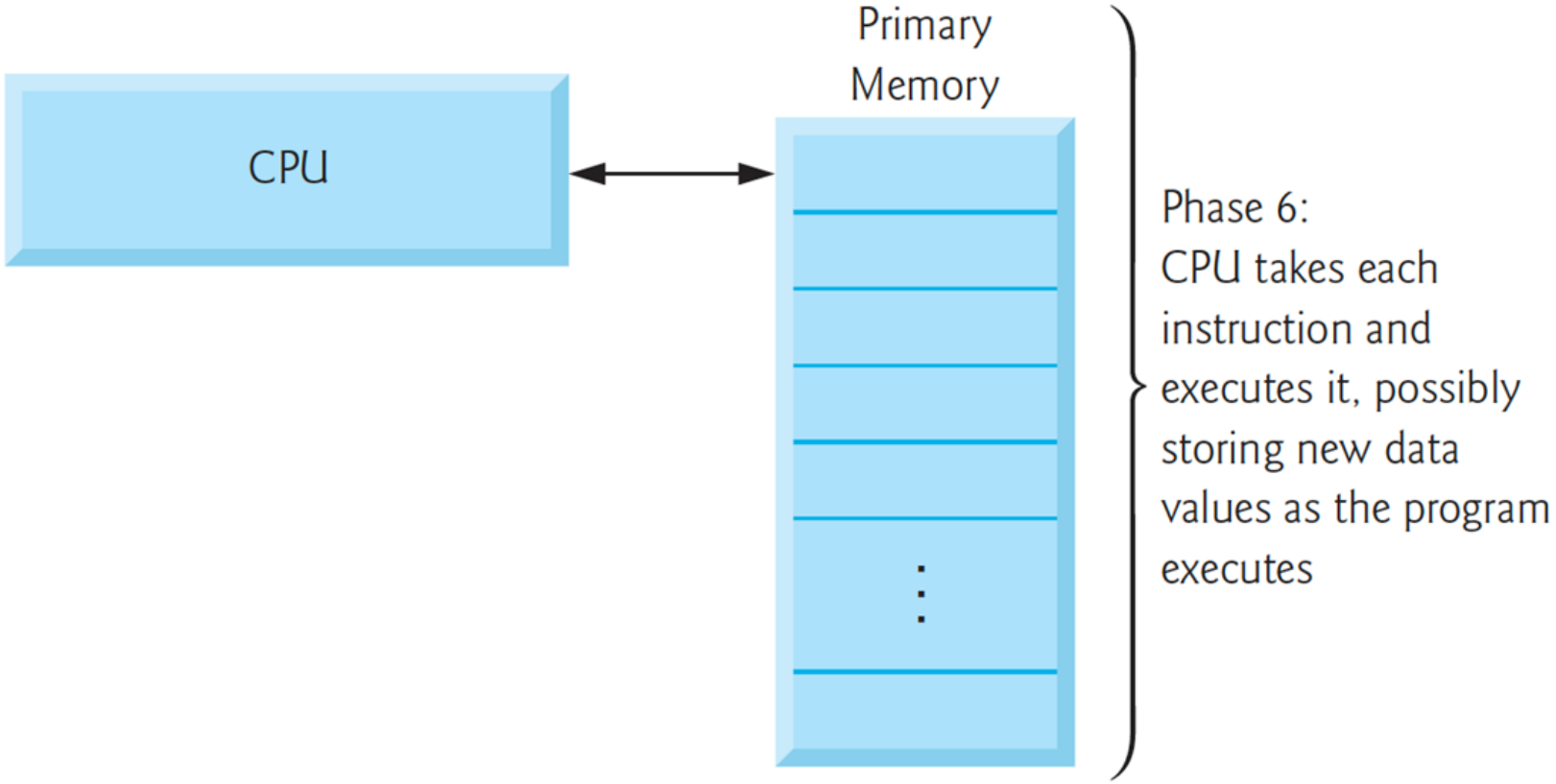
- Before a program can execute, the operating system must load it into memory
- Loader transfers the executable image from disk to memory
- Additional components from shared libraries that support the program also are loaded

Phase 5: Loading



Phase 6: Execution

- Finally, the computer, under the control of its CPU, executes the program one instruction at a time



Problems That May Occur at Execution Time

- Errors that occur as programs run are called runtime errors or execution-time errors
- Fatal errors cause a program to terminate immediately without successfully performing its job
- Nonfatal errors allow programs to run to completion, often producing incorrect results

The Software Development Method

1. Specify the problem requirements.
2. Analyze the problem.
3. Design the algorithm to solve the problem.
4. Implement the algorithm
5. Test and verify the completed program.
6. Maintain and update the program.

Reference

- C How to Program, Ninth Edition by Deitel & Deitel, Pearson, 2022.