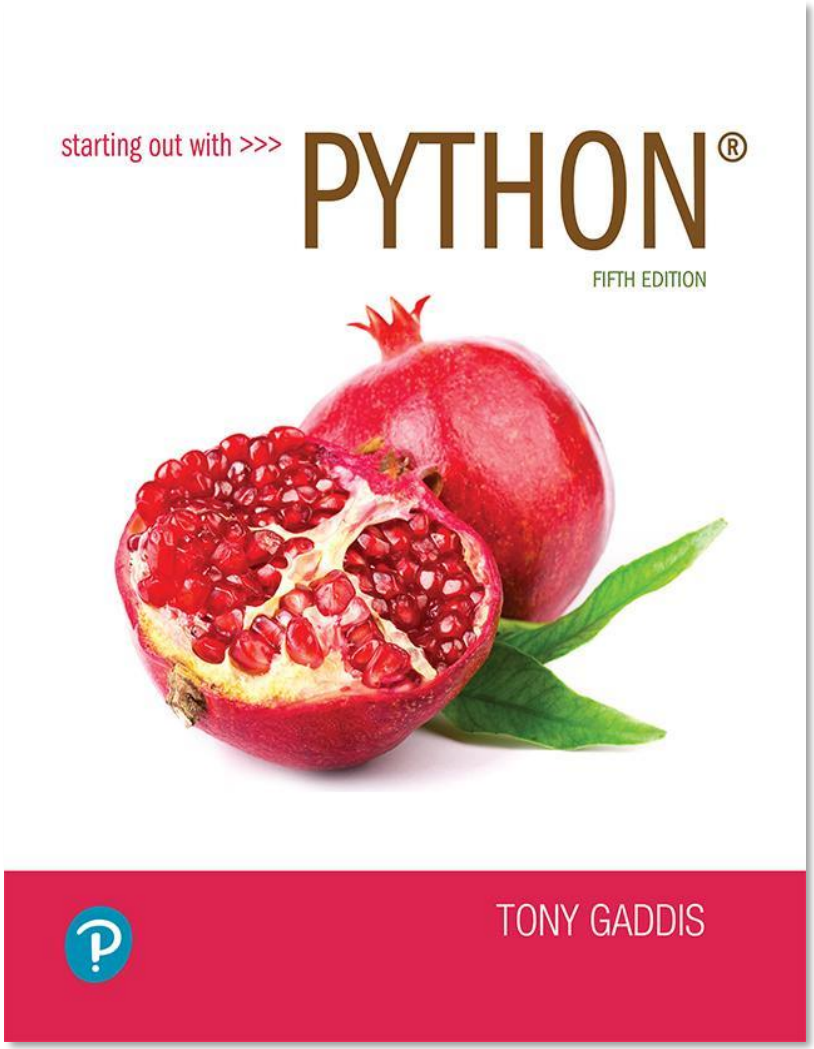
# Starting out with Python

Fifth Edition




## Chapter 1

Introduction to Computers and Programming

# Topics

- Introduction
- Hardware and Software
- How Computers Store Data
- How a Program Works
- Using Python

# Introduction

- Computers can be programmed
  - Designed to do any job that a program tells them to
- Program:
  - set of instructions that a computer follows to perform a task
  - Commonly referred to as *Software*
- Programmer:
  - person who can design, create, and test computer programs
  - Also known as software developer

# Hardware and Software

- Hardware: The physical devices that make up a computer
  - Computer is a system composed of several components that all work together
- Typical major components:
  - Central processing unit
  - Main memory
  - Secondary storage devices
  - Input and output devices

# The CPU

- Central processing unit (CPU): the part of the computer that actually runs programs
  - Most important component
  - Without it, cannot run software
  - Used to be a huge device
- Microprocessors: CPUs located on small chips
- https://www.youtube.com/watch?v=lxnlyJYZ6Vw&ab_channel=DellSupport

# Main Memory

- Main memory: where computer stores a program while program is running, and data used by the program
- Known as *Random Access Memory* or *RAM*
  - CPU is able to quickly access data in RAM
  - Volatile memory used for temporary storage while program is running
  - Contents are erased when computer is off

# Secondary Storage Devices

- Secondary storage: can hold data for long periods of time
  - Programs normally stored here and loaded to main memory when needed
- Types of secondary memory
  - Disk drive: magnetically encodes data onto a spinning circular disk
  - Solid state drive: faster than disk drive, no moving parts, stores data in solid state memory
  - Flash memory: portable, no physical disk

# Input Devices

- Input: data the computer collects from people and other devices
- Input device: component that collects the data
  - Examples: keyboard, mouse, touchscreen, scanner, camera
  - Disk drives?
  - can be considered input devices because they load programs into the main memory

# Output Devices

- Output: data produced by the computer for other people or devices
  - Can be text, image, audio, or bit stream
- Output device: formats and presents output
  - Examples:
  - video display, printer
  - Disk drives and USB drives???
  - can be considered output devices because data is sent to them to be saved

# Software (1 of 2)

- Everything the computer does is controlled by software
  - General categories:
    - Application software
    - System software
- Application software: programs that make computer useful for every day tasks
  - Examples:
  - word processing, email, games, and Web browsers

# Software (2 of 2)

- System software: programs that control and manage basic operations of a computer
  - Operating system: controls operations of hardware components
  - Utility Program: performs specific task to enhance computer operation or safeguard data
  - Software development tools: used to create, modify, and test software programs

# How Computers Store Data

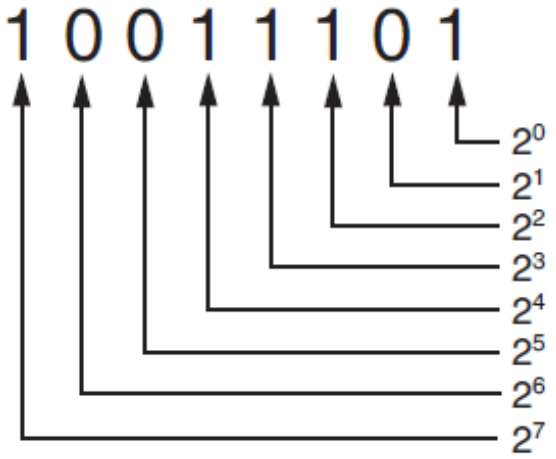
- All data in a computer is stored in sequences of 0s and 1s
- Byte: just enough memory to store letter or small number
  - Divided into eight bits
  - Bit: electrical component that can hold positive or negative charge, like on/off switch
  - The on/off pattern of bits in a byte represents data stored in the byte

# Storing Numbers

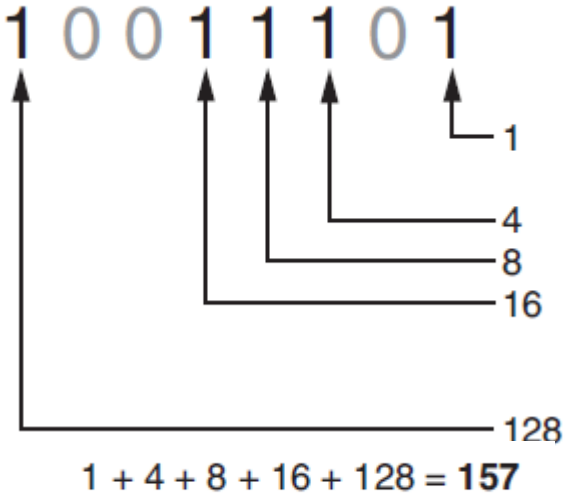
- Bit represents two values, 0 and 1
- Computers use binary numbering system
  - Position of digit `j` is assigned the value $2^{j-1}$
  - To determine value of binary number sum position values of the 1s
- Byte size limits are 0 and 255
  - 0 = all bits off; 255 = all bits on
  - To store larger number, use several bytes

# Example: Storing Numbers

The values of binary digits as powers of 2

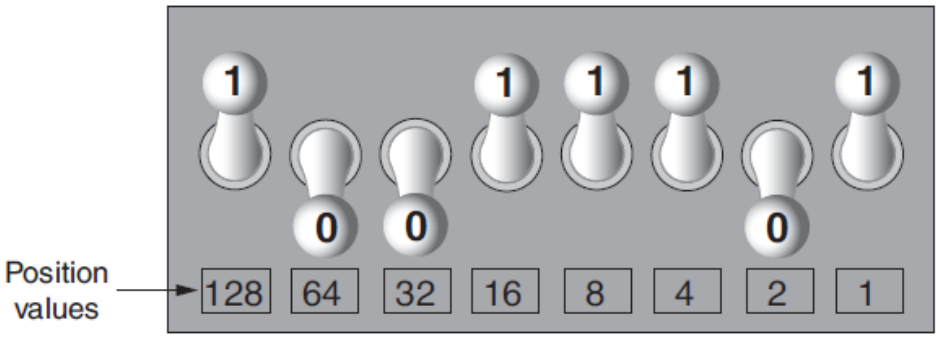



Determining the value of 10011101




1 + 4 + 8 + 16 + 128 = **157**

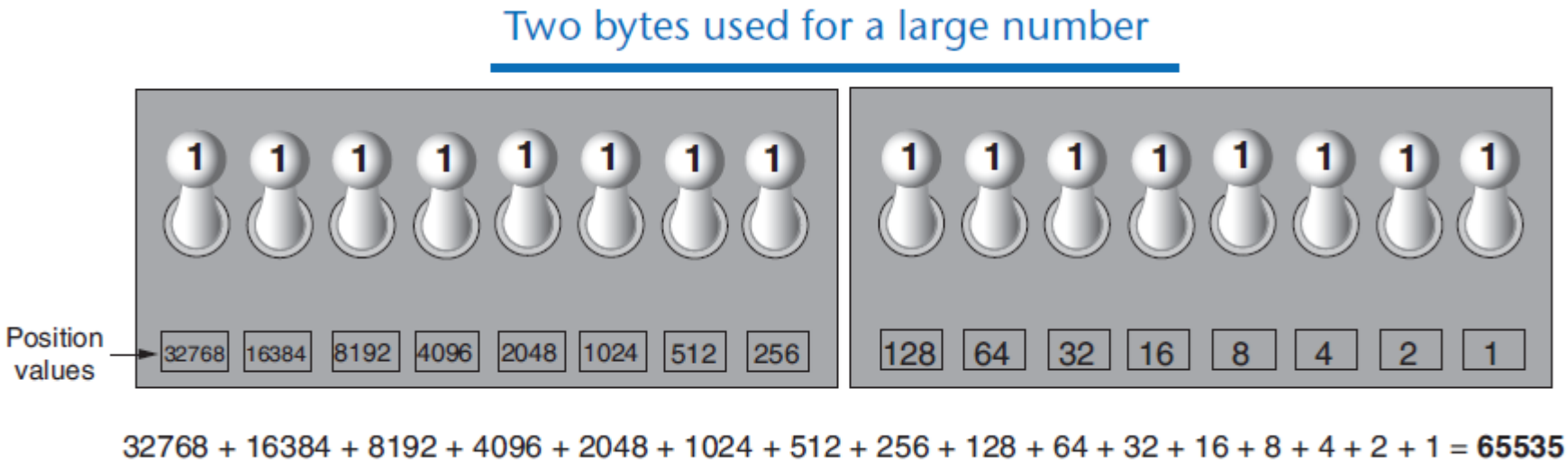
The bit pattern for 157




128 + 16 + 8 + 4 + 1 = **157**

# How to store large numbers??

- More than one byte



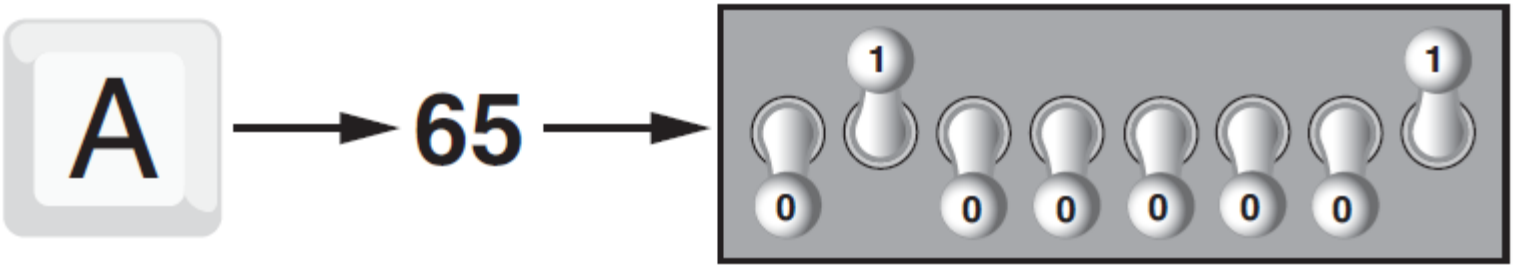

32768 + 16384 + 8192 + 4096 + 2048 + 1024 + 512 + 256 + 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = **65535**

# Storing Characters

- Data stored in computer must be stored as binary number
- Characters are converted to numeric code, numeric code stored in memory
  - Most important coding scheme is ASCII
    - ASCII is limited: defines codes for only 128 characters
  - Unicode coding scheme becoming standard
    - Compatible with ASCII
    - Can represent characters for other languages

# Example: Storing Characters

The letter A is stored in memory as the number 65



# Advanced Number Storage

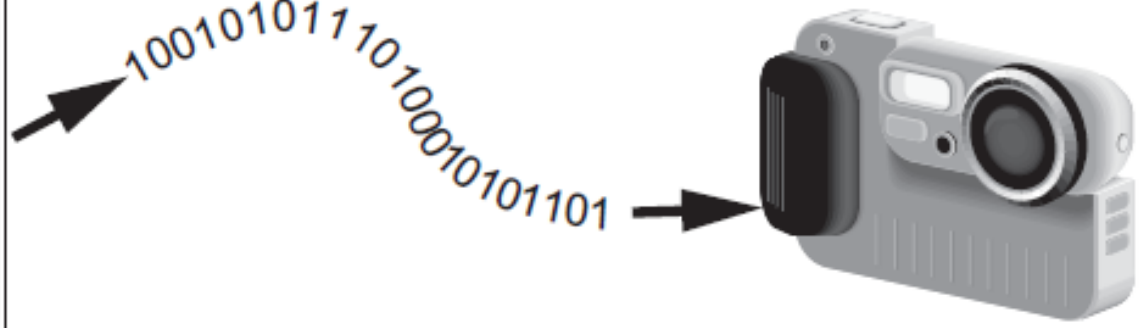
- To store negative numbers and real numbers, computers use binary numbering and encoding schemes
  - Negative numbers encoded using two's complement
  - Real numbers encoded using floating-point notation

# Other Types of Data

- Digital:
- describes any device that stores data as binary numbers
- Digital images are composed of pixels
  - To store images, each pixel is converted to a binary number representing the pixel's color
- Digital music is composed of sections called samples
  - To store music, each sample is converted to a binary number

# A digital image is stored in binary format

Jupiterimages/Getty Images



## Checkpoint

1.12 What amount of memory is enough to store a letter of the alphabet or a small number?

1.13 What do you call a tiny "switch" that can be set to either on or off?

1.14 In what numbering system are all numeric values written as sequences of 0s and 1s?

1.15 What is the purpose of ASCII?

1.16 What encoding scheme is extensive enough to represent the characters of many of the languages in the world?

1.17 What do the terms "digital data" and "digital device" mean?

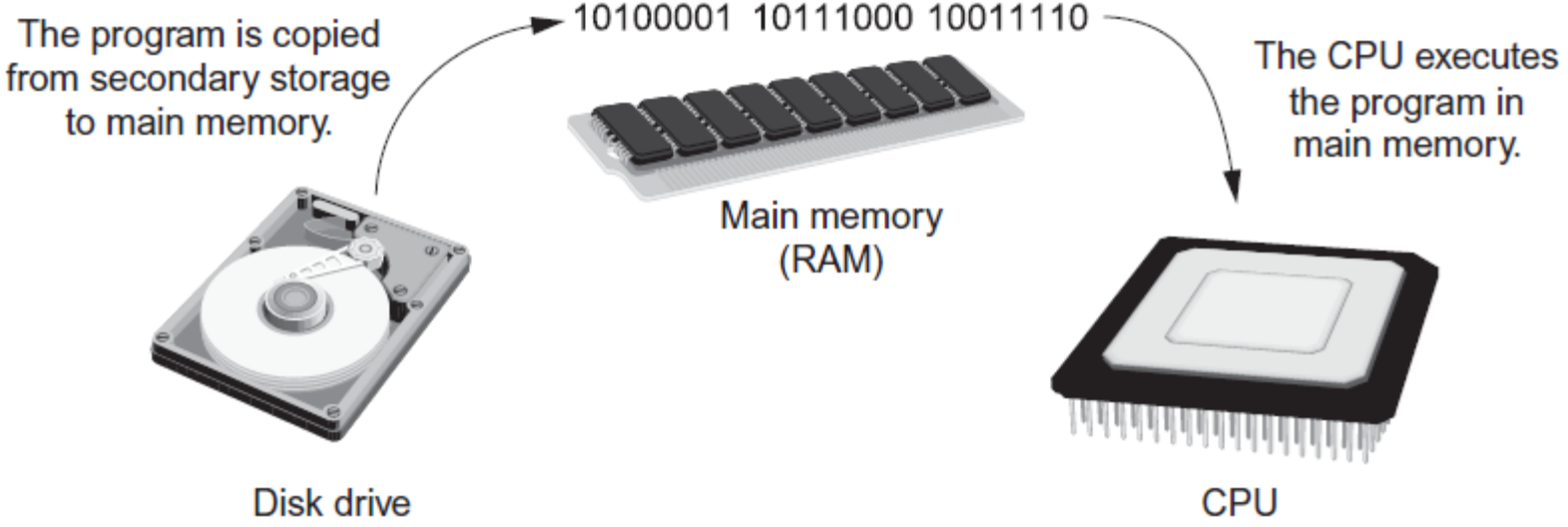
# How a Program Works (1 of 3)

- CPU designed to perform simple operations on pieces of data
  - Examples: reading data, adding, subtracting, multiplying, and dividing numbers
  - Understands instructions written in machine language and included in its instruction set
    - Each brand of CPU has its own instruction set
- To carry out meaningful calculation, CPU must perform many operations

# How a Program Works (2 of 3)

- Program must be copied from secondary memory to RAM each time CPU executes it
- CPU executes program in cycle:
  - Fetch: read the next instruction from memory into CPU
  - Decode: CPU decodes fetched instruction to determine which operation to perform
  - Execute: perform the operation

## A program is copied into main memory and then executed



# How a Program Works (3 of 3)

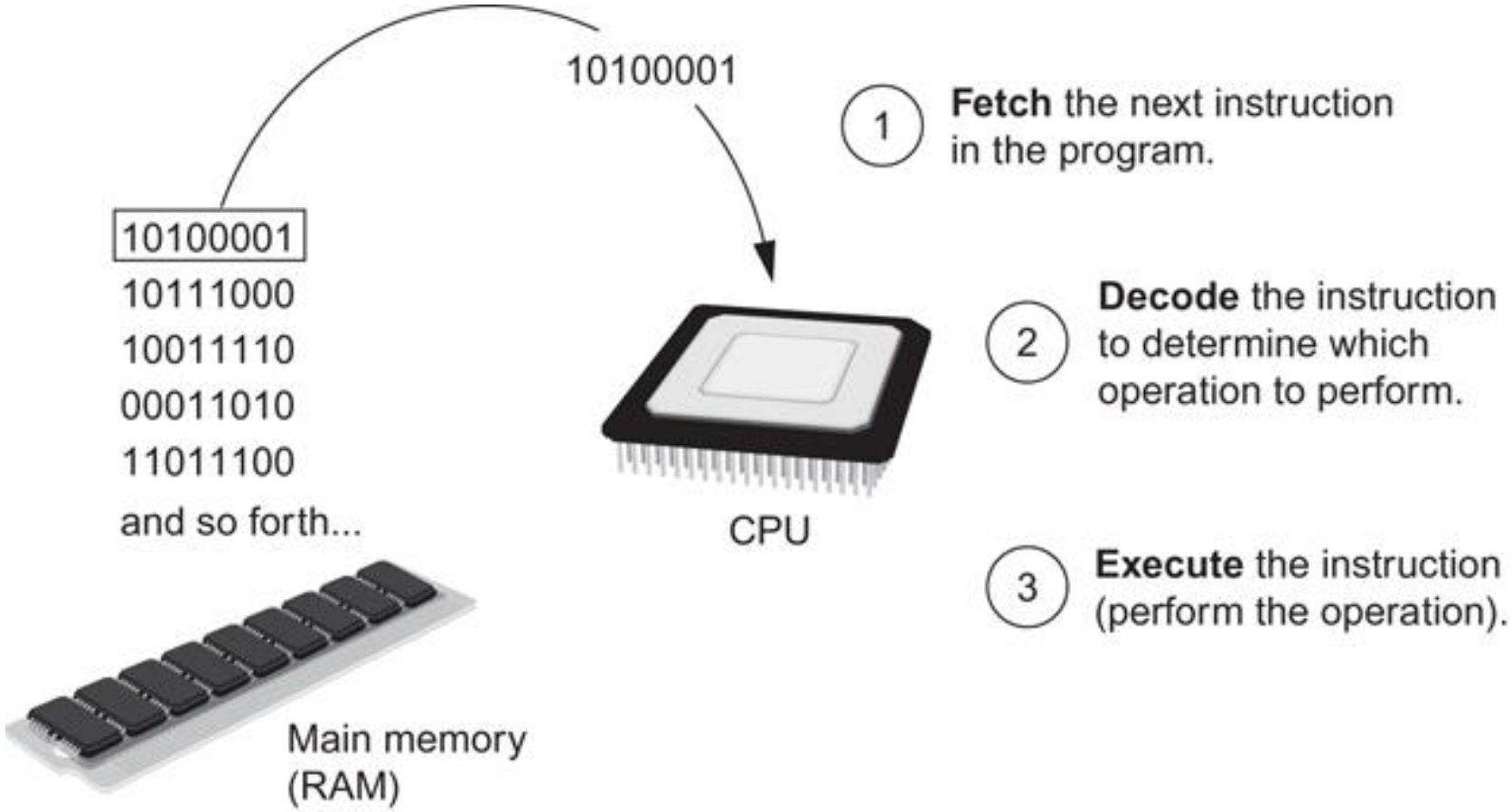
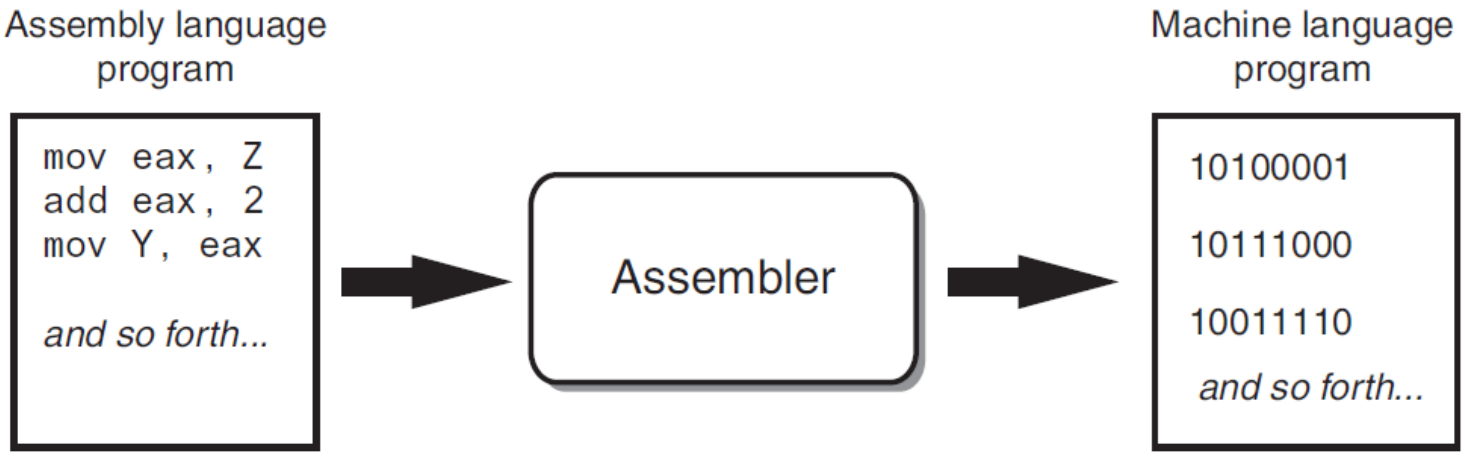



**Figure 1-16** The fetch-decode-execute cycle

# From Machine Language to Assembly Language

- Impractical for people to write in machine language
- Assembly language: uses short words (mnemonics) for instructions instead of binary numbers
  - Easier for programmers to work with
- Assembler: translates assembly language to machine language for execution by CPU



# High-Level Languages

- Low-level language: close in nature to machine language
  - Example: assembly language
- High-Level language: allows simple creation of powerful and complex programs
  - No need to know how CPU works or write large number of instructions
  - More intuitive to understand

# Keywords, Operators, and Syntax: an Overview

- Keywords: predefined words used to write program in high-level language
  - Each keyword has specific meaning
- Operators: perform operations on data
  - Example: math operators to perform arithmetic
- Syntax: set of rules to be followed when writing program
- Statement: individual instruction used in high-level language

## The Python keywords

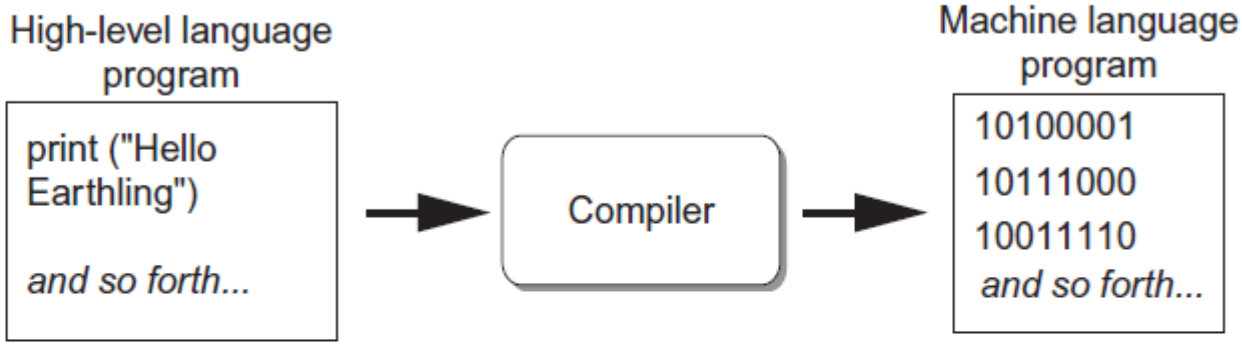
| | | | | |
|---|---|---|---|---|
| and | continue | finally | is | raise |
| as | def | for | lambda | return |
| assert | del | from | None | True |
| async | elif | global | nonlocal | try |
| await | else | if | not | while |
| break | except | import | or | with |
| class | False | in | pass | yield |

# Compilers and Interpreters (1 of 3)

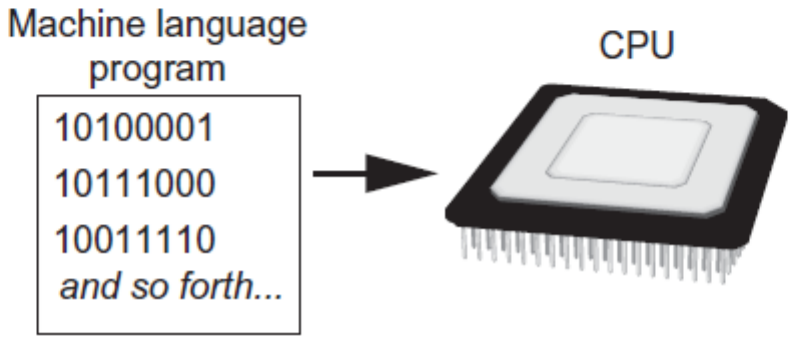
- Programs written in high-level languages must be translated into machine language to be executed
- Compiler: translates high-level language program into separate machine language program
  - Machine language program can be executed at any time

## Compiling a high-level program and executing it

1. The compiler is used to translate the high-level language program to a machine language program.




2. The machine language program can be executed at any time, without using the compiler.



# Compilers and Interpreters (2 of 3)

- Interpreter: translates and executes instructions in high-level language program
  - Used by Python language
  - Interprets one instruction at a time
  - No separate machine language program
- Source code: statements written by programmer
  - Syntax error: prevents code from being translated

# Compilers and Interpreters (3 of 3)

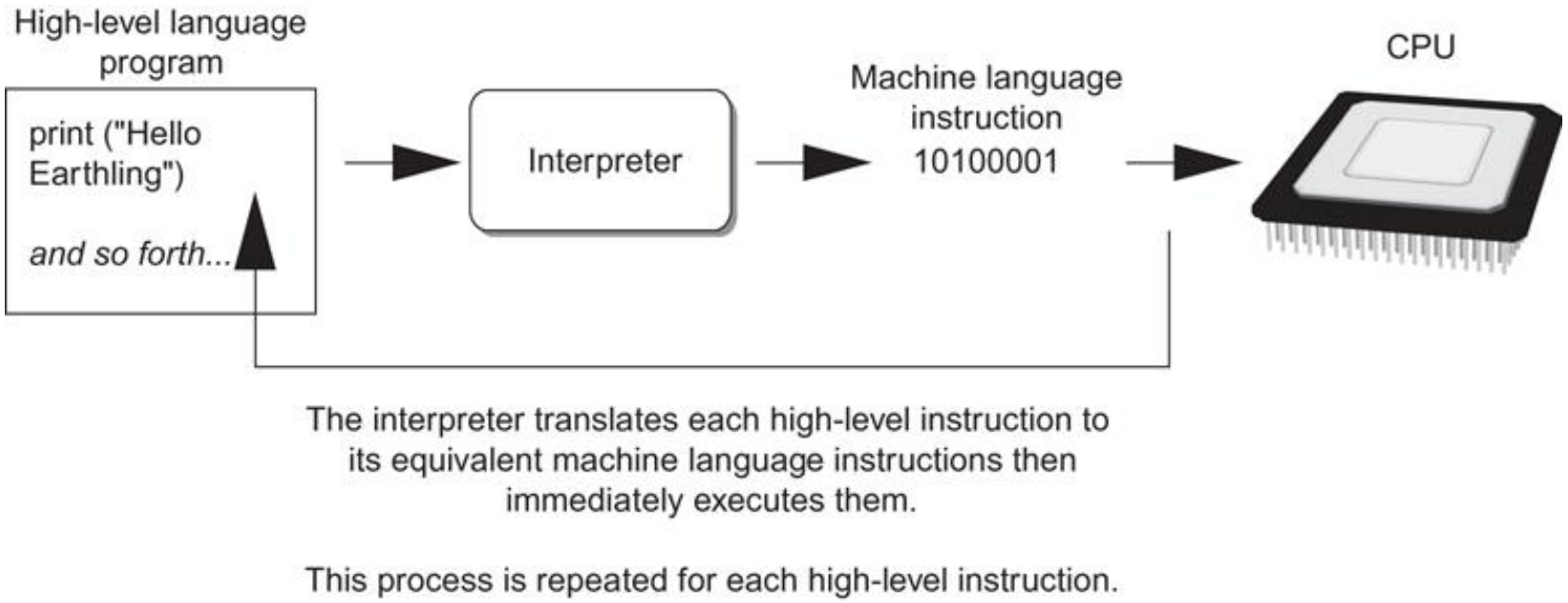



**Figure 1-19** Executing a high-level program with an interpreter

## Checkpoint

1.18 A CPU understands instructions that are written only in what language?

1.19 A program has to be copied into what type of memory each time the CPU executes it?

1.20 When a CPU executes the instructions in a program, it is engaged in what process?

1.21 What is assembly language?

1.22 What type of programming language allows you to create powerful and complex programs without knowing how the CPU works?

1.23 Each language has a set of rules that must be strictly followed when writing a program. What is this set of rules called?

1.24 What do you call a program that translates a high-level language program into a separate machine language program?

1.25 What do you call a program that both translates and executes the instructions in a high-level language program?

1.26 What type of mistake is usually caused by a misspelled keyword, a missing punctuation character, or the incorrect use of an operator?

# Using Python

- Python must be installed and configured prior to use
  - One of the items installed is the Python interpreter
- Python interpreter can be used in two modes:
  - Interactive mode: enter statements on keyboard
  - Script mode: save statements in Python script

# Summary

- This chapter covered:
  - Main hardware components of the computer
  - Types of software
  - How data is stored in a computer
  - Basic CPU operations and machine language
  - Fetch-decode-execute cycle
  - Complex languages and their translation to machine code
  - Installing Python and the Python interpreter modes