

# Chapter 1

# Getting Started

## 1.1 What Do Computers Do?

- A ***computer system*** is an integrated collection of hardware and software components.
- ***Hardware*** refers to the electronics inside a computer.
- ***Software*** consists of programs that tell the hardware what to do.

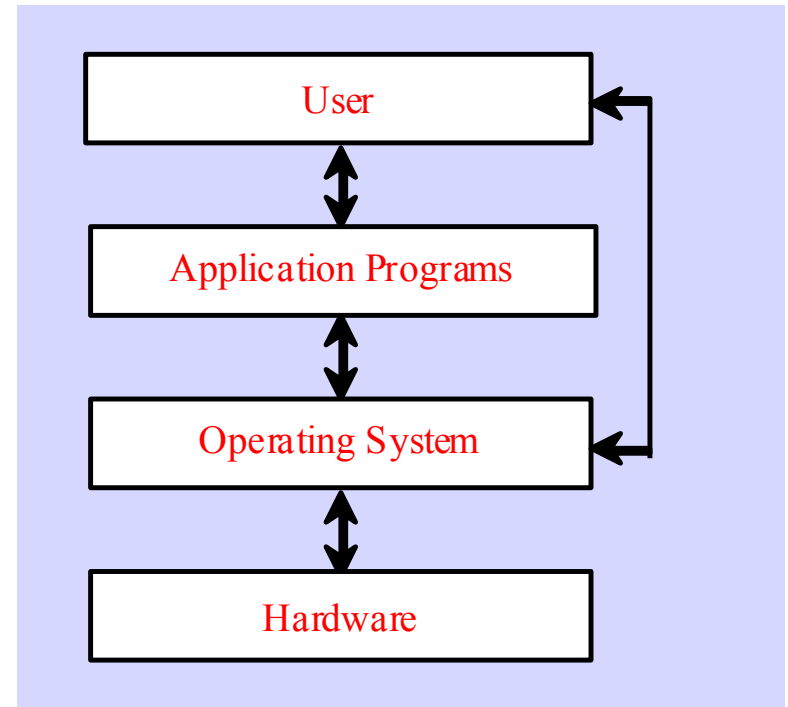
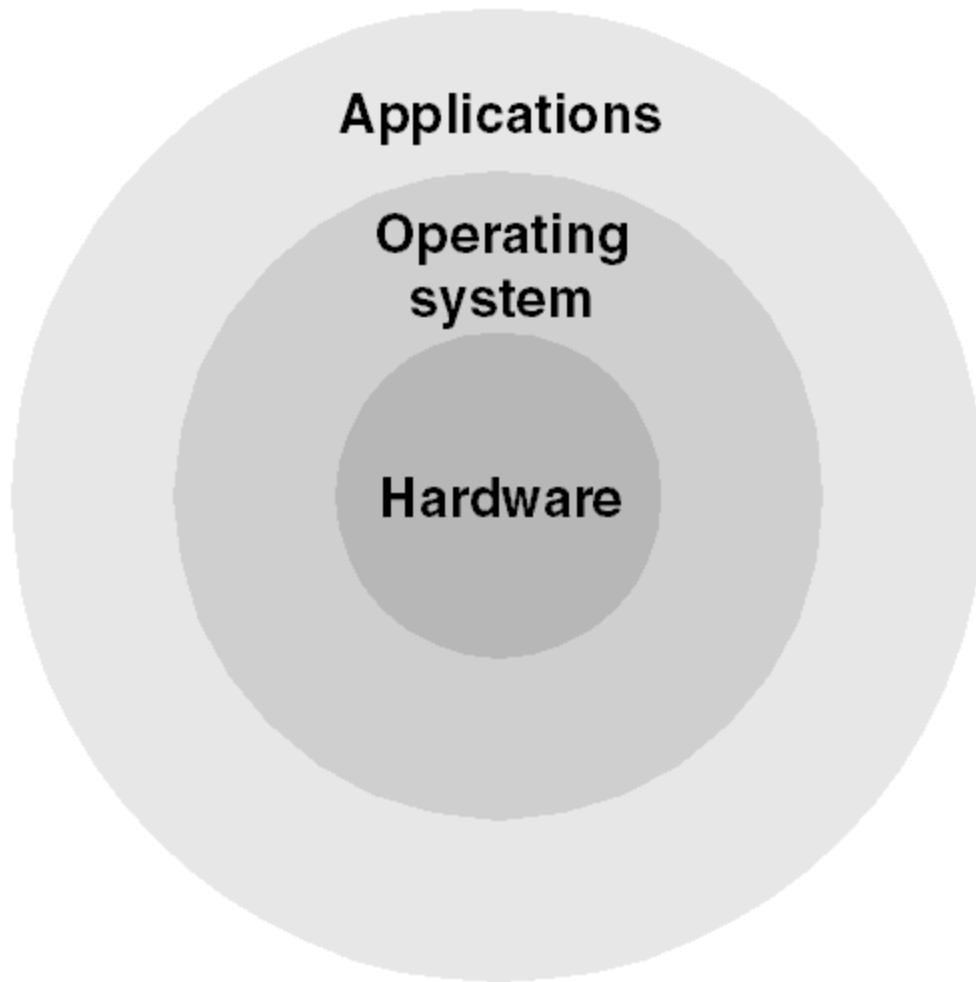
## Hardware

- ***Processors***
  - *Central processing unit, or CPU*
  - Specialized processors, such as a graphics processor
- ***Memory***
  - *Main memory, or RAM (random-access memory)*
  - *ROM (read-only memory)*
  - Hard disks, floppy disks, and other storage media
- ***Peripheral devices***
  - Provide an *interface* to the world outside the system
  - Include keyboards, mice, monitors, printers, and modems

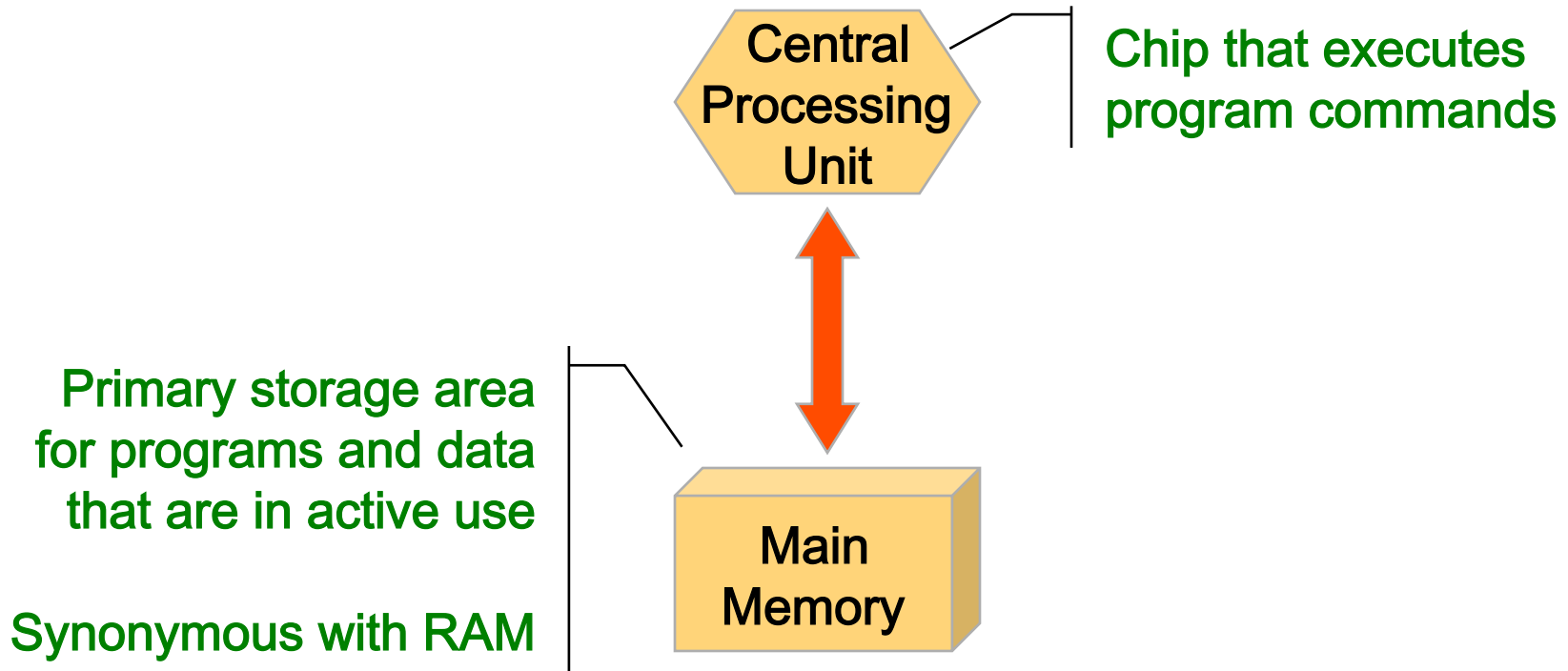
# Software

- Software consists of programs that instruct the hardware how to perform operations.
- A *program* is a step-by-step set of instructions.
- Categories of software:
  - *Operating systems*. A collection of programs that interact directly with the computer's hardware.
  - *Applications*. Programs designed to perform useful tasks for humans.
- An operating system serves as a bridge between hardware and applications.

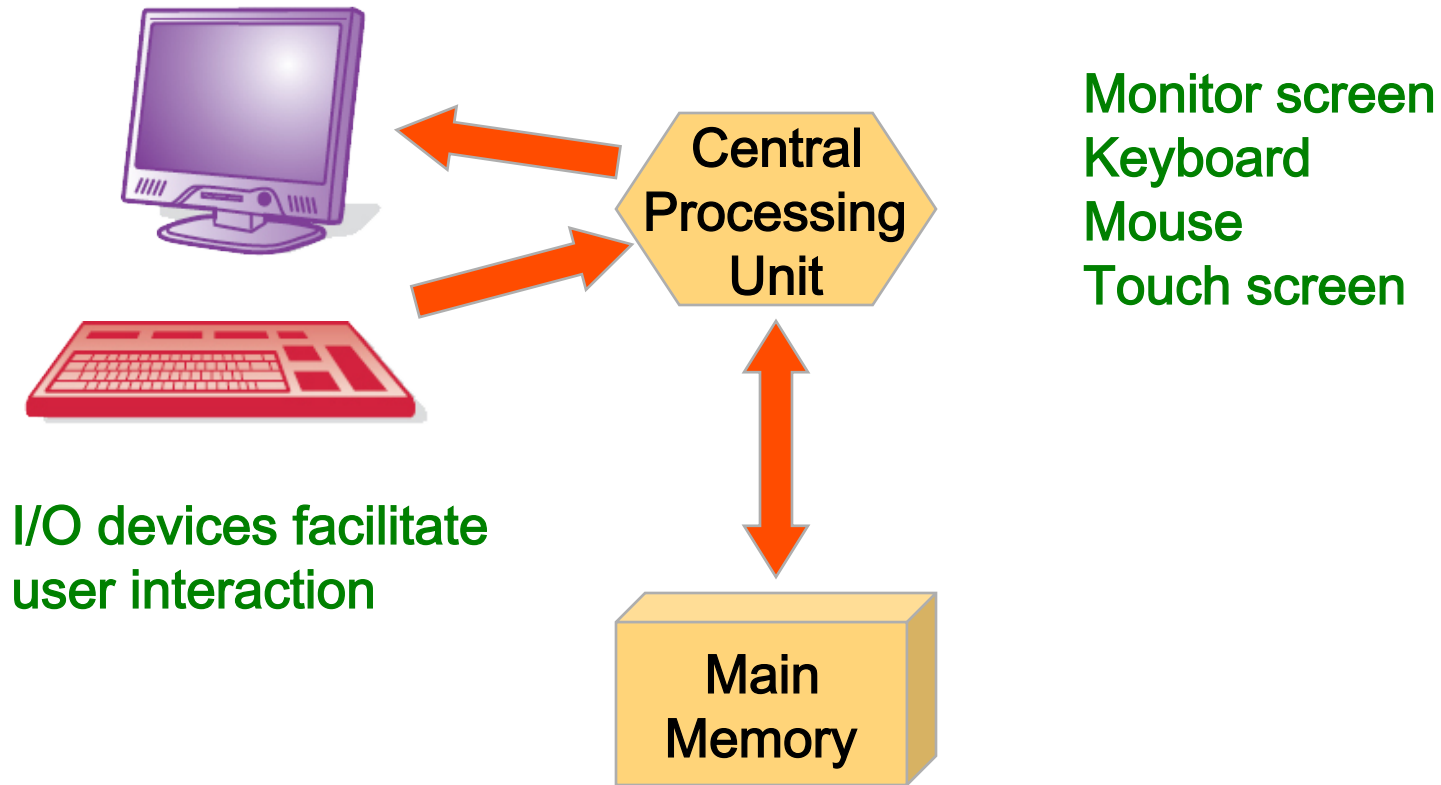
## Chapter 1: Getting Started



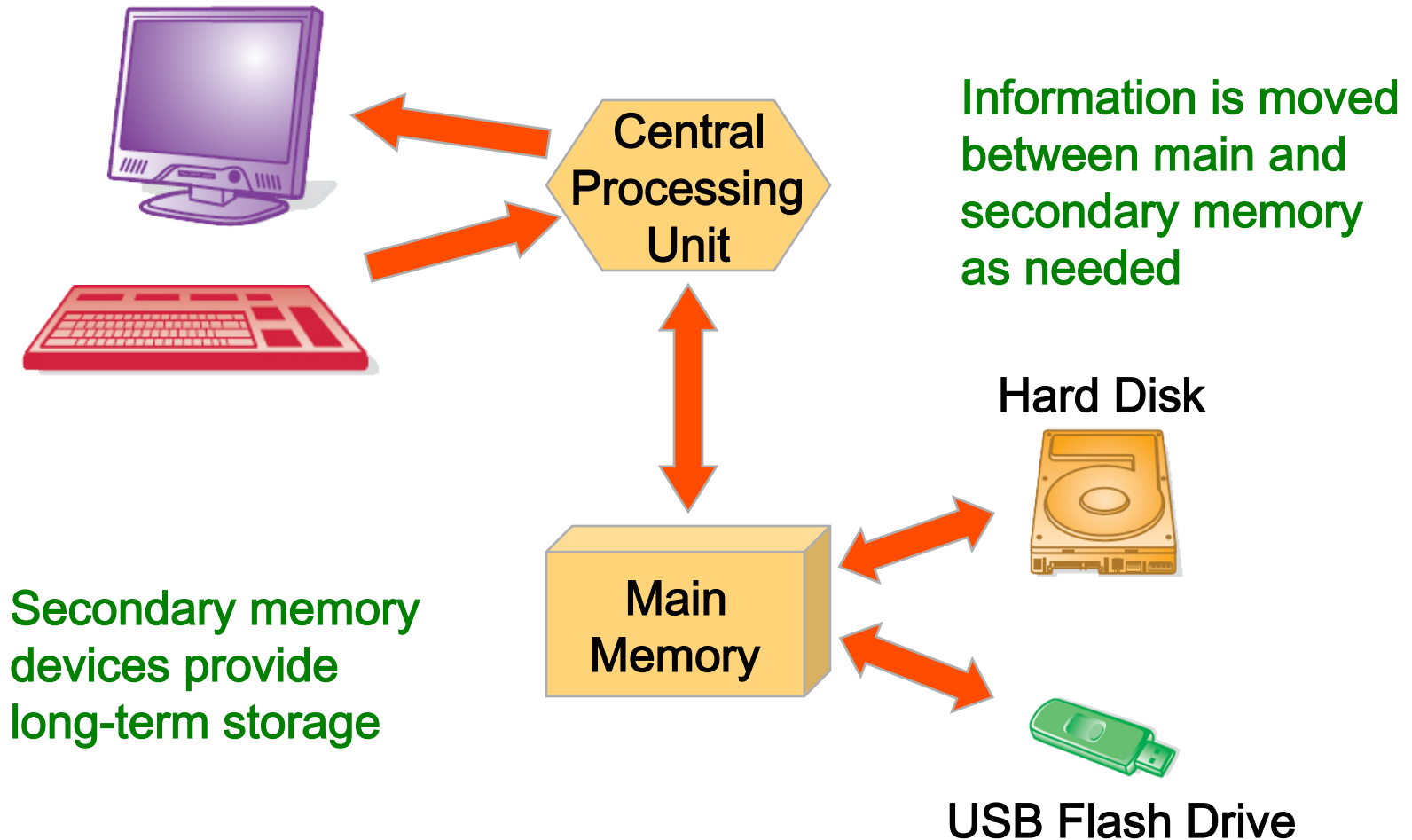
# CPU and Main Memory



# Input / Output Devices

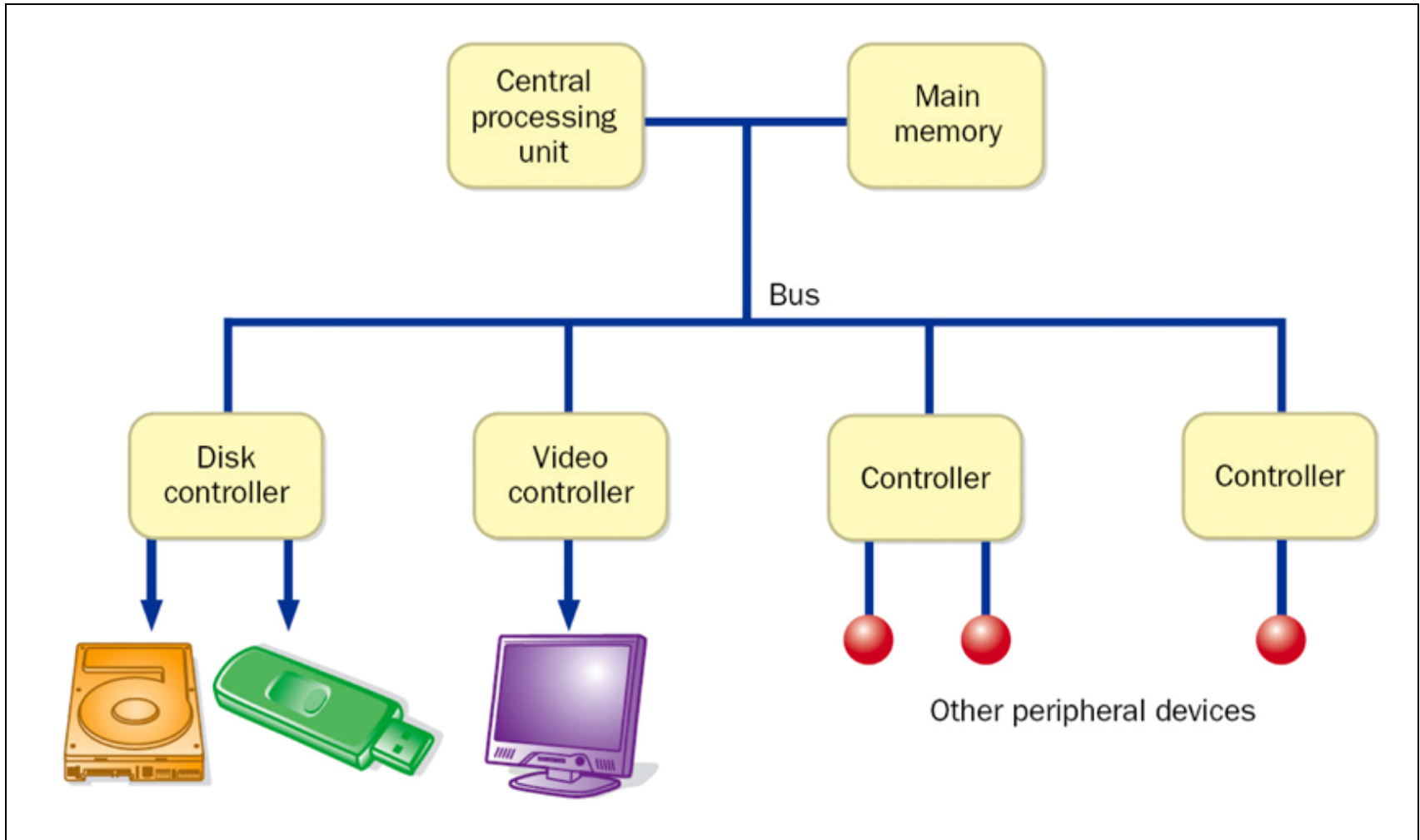


# Secondary Memory Devices





# Computer Architecture



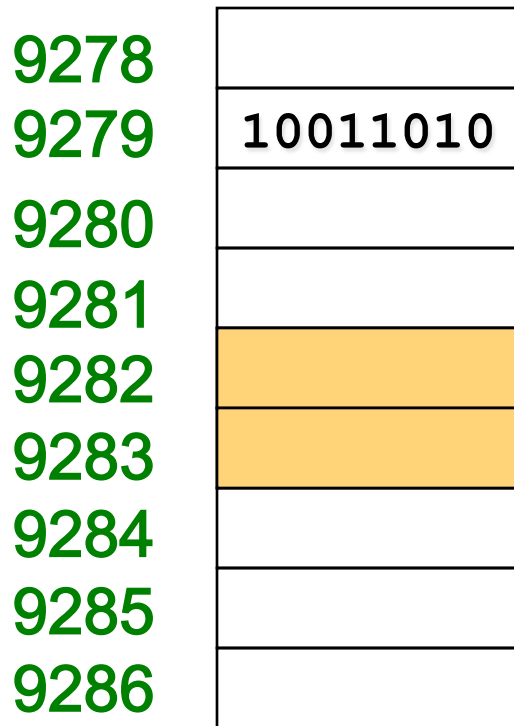
# Memory



Main memory is divided into many memory locations (or *cells*)

Each memory cell has a numeric *address*, which uniquely identifies it

# Storing Information



— Each memory cell stores a set number of bits (usually 8 bits, or one *byte*)

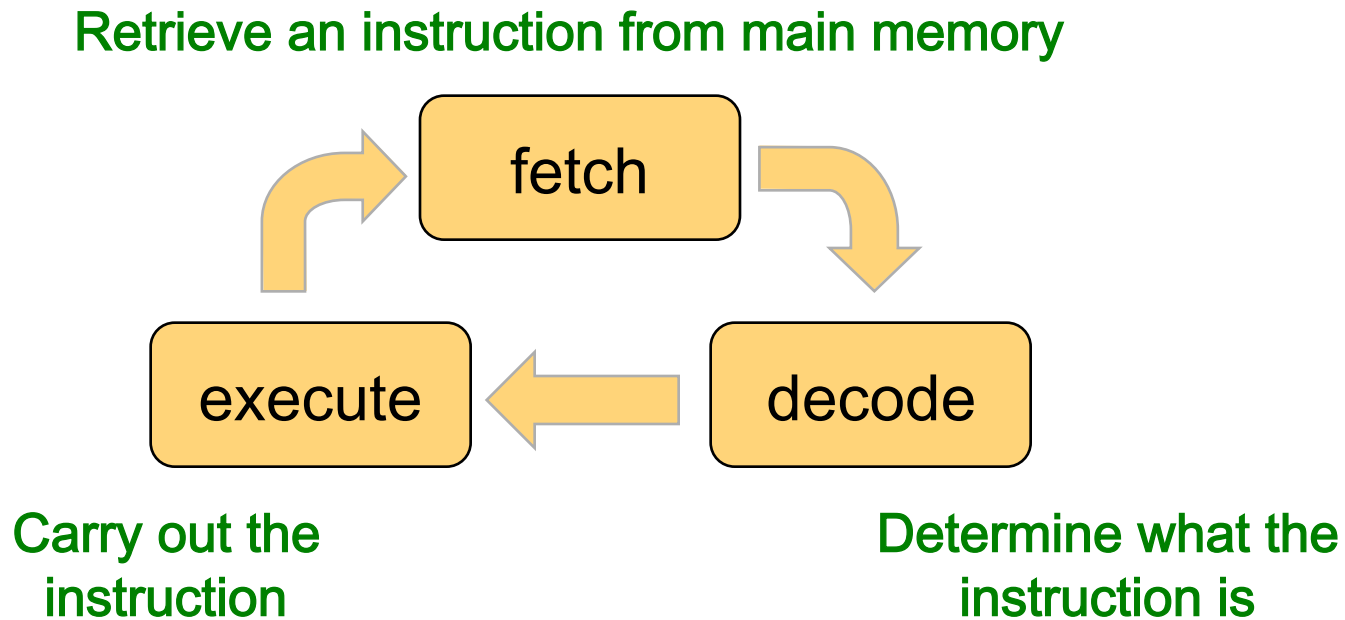
} Large values are stored in consecutive memory locations

# Memory

- Main memory is *volatile* - stored information is lost if the electric power is removed
- Secondary memory devices are *nonvolatile*
- Main memory and disks are *direct access* devices - information can be reached directly
- The terms *direct access* and *random access* often are used interchangeably
- A magnetic tape is a *sequential access* device since its data is arranged in a linear order - you must get by the intervening data in order to access other information

## The Central Processing Unit

- A CPU is on a chip called a *microprocessor*
- It continuously follows the *fetch-decode-execute cycle*:



## Platforms

- The combination of an operating system and a particular type of CPU is often called a *platform*.
- Software usually works only on a single platform.
- Java programs, however, will run on multiple platforms without change.
- Most of the time, a computer system has only one operating system but many applications.
- Applications are usually designed for one particular version of an operating system.

## 1.2 Ways of Interacting with Computers

- Most applications need to communicate, or “interface,” with the user by displaying information for the user to see and accepting commands from the user.
- Primary types of user interfaces:
  - Graphical user interfaces
  - Text-based interfaces

## Graphical User Interfaces

- Most applications now rely on a *graphical user interface*, or **GUI** (pronounced “gooey”) built out of visual components.
- When a GUI program is run, it displays a window on the screen.
- The window is composed of thousands of tiny *pixels* (picture elements), each with its own color.



## Dialog Boxes

- Performing certain actions will cause other windows to appear.
- These *dialog boxes* or *dialogs* are used to display information to the user and/or accept input from the user.
- One type of dialog box is called a *file dialog box* or a *file dialog*.
- A file dialog allows the user to choose a file.

## Origin of the Graphical User Interface

- Graphical user interfaces were developed during the 1970s at Xerox Corporation's Palo Alto Research Center.
- Apple Computer incorporated the GUI approach into a computer called the Lisa, which flopped.
- Apple tried again and made a success of the Macintosh, the first widely used computer to support a graphical user interface.
- Microsoft Corporation later developed Windows for the IBM Personal Computer and its clones.

## Text-Based Interfaces

- Before the advent of graphical user interfaces, programs used a *text-based interface*, in which all input and output consisted of characters.
- In a text-based interface, no graphics are displayed, and user commands are entered from the keyboard.
- Text-based programs are normally run from a *command line*.

## Command-Line Prompts

- Typical Unix command-line prompt:  
%
- Typical DOS command-line prompt:  
C:>
- The DOS prompt is often configured to display the “current directory”:  
C:\WINDOWS>

## DOS Windows

- Operating systems that provide a graphical user interface may still allow the user to run text-based programs from a command line.
- In Windows, the user can open a “DOS window,” which allows programs to be run from a DOS-like command line.
- A DOS window is normally capable of displaying up to 25 lines, with each line limited to 80 characters.
- Several DOS windows can be open at a time.

## 1.3 What Is Programming?

- **Programming** means writing down a series of instructions that tell a computer what to do.
- Properties of these instructions:
  - Computation proceeds in discrete steps.
  - Each step is precisely defined.
  - The order in which steps are performed may be important.

# Algorithms

- A set of instructions with these properties is said to be an *algorithm*.
- The steps in an algorithm are not always short and simple.
  - Some steps may involve a series of smaller steps.
  - Some steps may involve making decisions.
  - Some steps may repeat.
- Algorithms are common in the real world.

## When to talk to me ?

### Algorithm 1

- 1: send questions to me at [cao@cs.gsu.edu](mailto:cao@cs.gsu.edu)
- 2: else  
    use office hours 12:50-1:50pm MW.
- 3: else  
    make an appointment
- 4: goto 1



## Computer Algorithms

- Computer algorithms often involve obtaining input, performing a calculation, and producing output.
- An algorithm for converting from Fahrenheit to Celsius:
  1. Display a message asking the user to enter a Fahrenheit temperature.
  2. Obtain the input entered by the user.
  3. Convert the user's input into numerical form.
  4. Calculate the equivalent Celsius temperature, using the formula
$$C = (F - 32) \times (5 / 9)$$
  5. Convert the Celsius temperature into character form and display the result.

# Analysis of the Temperature Conversion Algorithm

- It's not clear how to display information to the user and obtain the user's input. That will depend on whether the program is GUI or text-based.
- Step 3 (converting the user's input to numerical form) is a bit fuzzy: What action should be taken if the input is not in the form of a number?

## Ways to Express Algorithms

- ***Natural languages.*** Allows anyone who understands that language to read the algorithm, but lacks precision.
- ***Programming languages.*** Precise, yet simple enough for computers to understand.
- ***Pseudocode.*** A mixture of natural language and a programming language. More precise than natural language but less precise than a programming language. Often easier to read (and to write) than a programming language.

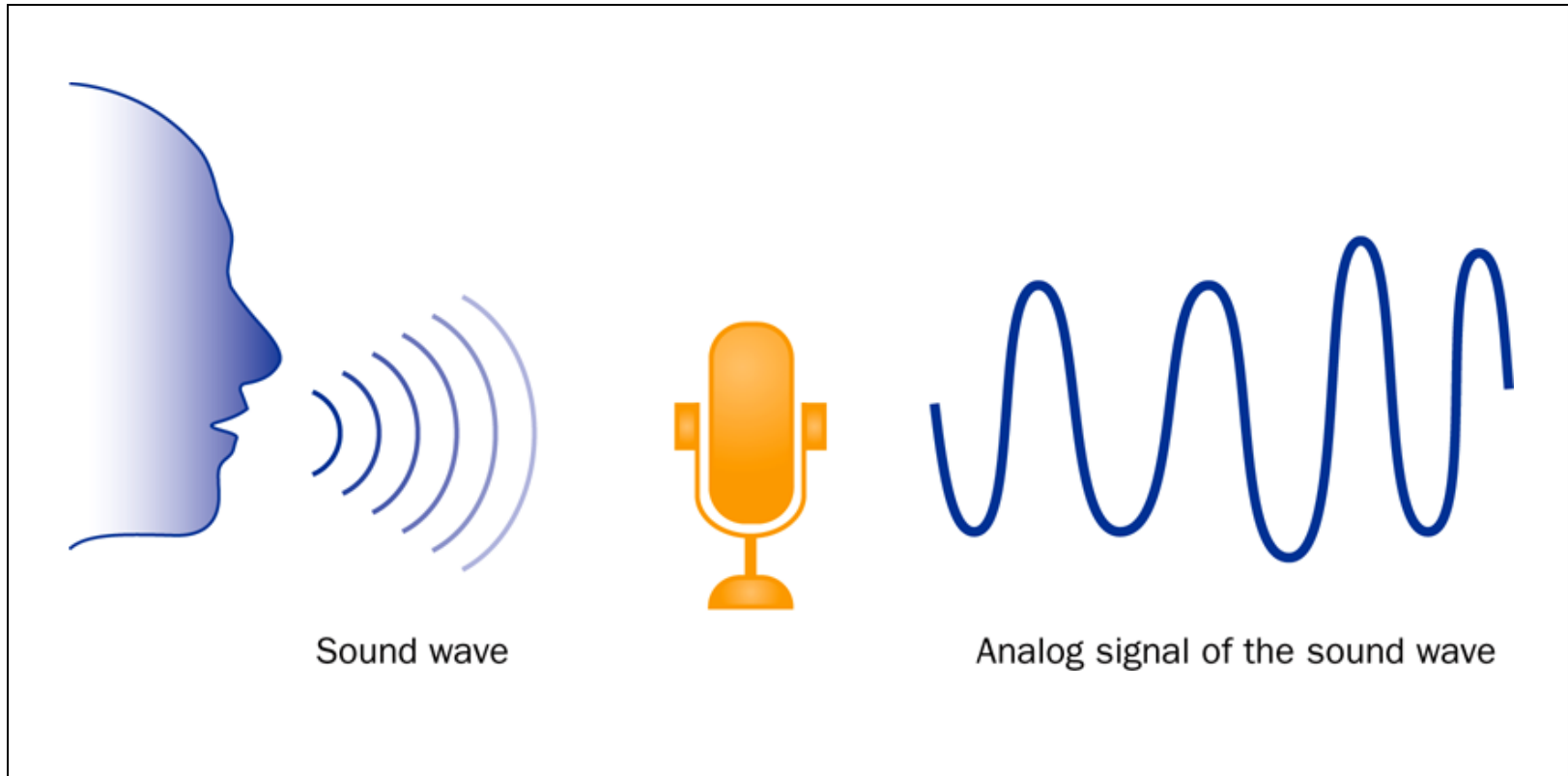
## 1.4 Storing Data

- Computer algorithms manipulate data.
- The term *data* refers to information, particularly information that's stored in a uniform and systematic fashion.
- The Fahrenheit-to-Celsius algorithm deals with several items of data, including two numbers (a Fahrenheit temperature and its Celsius equivalent).

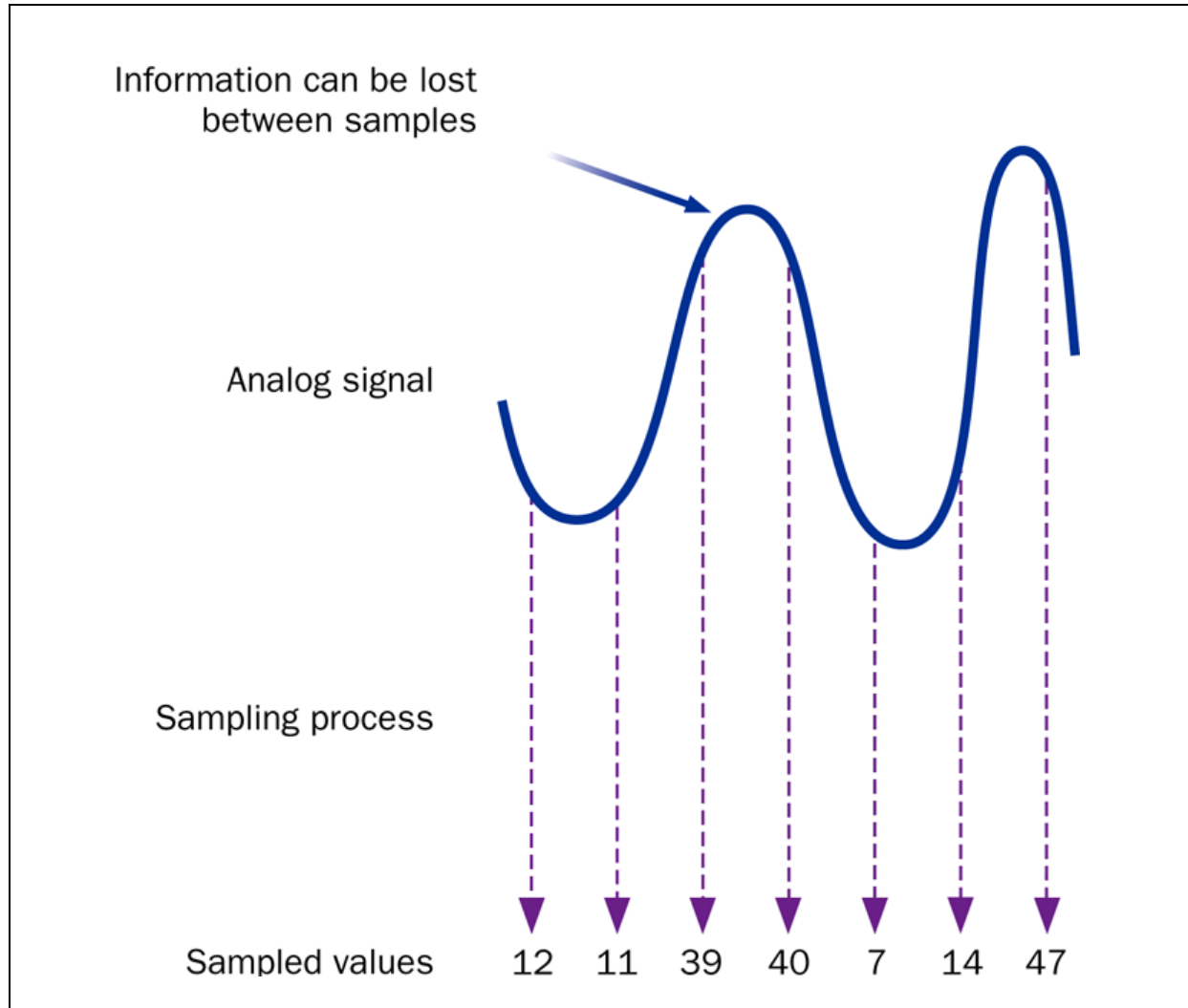
# Analog vs. Digital

- There are two basic ways to store and manage data:
- *Analog*
  - continuous, in direct proportion to the data represented
  - music on a record album - a needle rides on ridges in the grooves that are directly proportional to the voltages sent to the speaker
- *Digital*
  - the information is broken down into pieces, and each piece is represented separately
  - *sampling* – record discrete values of the analog representation
  - music on a compact disc - the disc stores numbers representing specific voltage levels sampled at specific times

# Analog Information



# Sampling



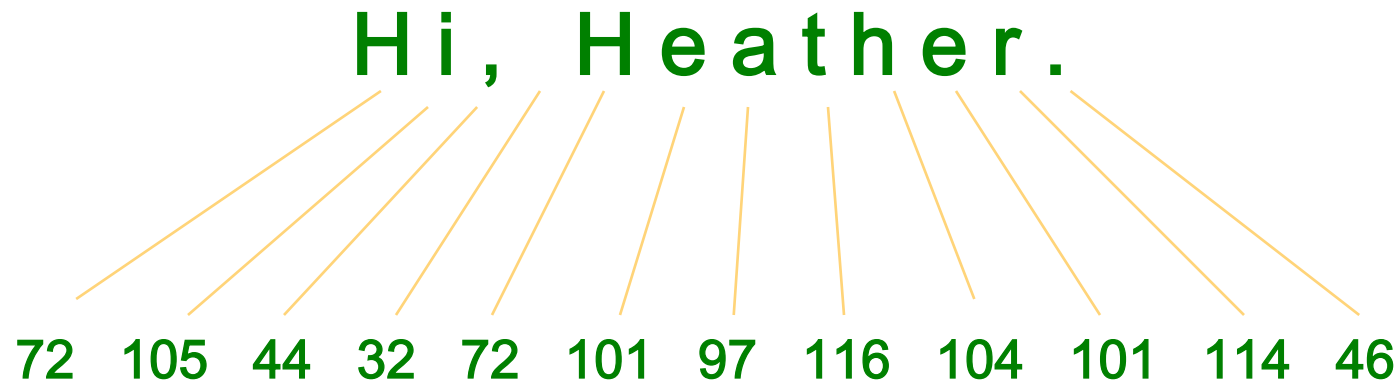
## Digital Information

- Computers store all information digitally:
  - numbers
  - text
  - graphics and images
  - audio
  - video
  - program instructions
- In some way, all information is *digitized* - broken down into pieces and represented as numbers



## Representing Text Digitally

- For example, every character is stored as a number, including spaces, digits, and punctuation
- Corresponding upper and lower case letters are separate characters



## Binary Numbers

- Once information has been digitized, it is represented and stored in memory using the *binary number system*
- A single binary digit (0 or 1) is called a *bit*
- Devices that store and move information are cheaper and more reliable if they have to represent only two states
- A single bit can represent two possible states, like a light bulb that is either on (1) or off (0)
- Permutations of bits are used to store values

## Bit Permutations

### 1 bit

0  
1

### 2 bits

00  
01  
10  
11

### 3 bits

000  
001  
010  
011  
100  
101  
110  
111

### 4 bits


0000 1000  
0001 1001  
0010 1010  
0011 1011  
0100 1100  
0101 1101  
0110 1110  
0111 1111

Each additional bit doubles the number of possible permutations

## Bit Permutations

- Each permutation can represent a particular item
- There are  $2^N$  permutations of N bits
- Therefore, N bits are needed to represent  $2^N$  unique items

**How many  
items can be  
represented by**



<b>1 bit ?</b>	<b><math>2^1 = 2</math> items</b>
<b>2 bits ?</b>	<b><math>2^2 = 4</math> items</b>
<b>3 bits ?</b>	<b><math>2^3 = 8</math> items</b>
<b>4 bits ?</b>	<b><math>2^4 = 16</math> items</b>
<b>5 bits ?</b>	<b><math>2^5 = 32</math> items</b>

## Quick Check

How many bits would you need to represent each of the 50 United States using a unique permutation of bits?

## Quick Check

How many bits would you need to represent each of the 50 United States using a unique permutation of bits?

Five bits wouldn't be enough, because  $2^5$  is 32.

**Six bits** would give us 64 permutations, and some wouldn't be used.

000000	Alabama
000001	Alaska
000010	Arizona
000011	Arkansas
000100	California
000101	Colorado
	etc.

## Numeric Data

- Humans usually write numbers in *decimal* (base 10), using the *digits* 0 through 9.
- Computers store numbers in *binary* (base 2).
- In binary, numbers consist of *bits* (binary digits), each of which is either 0 or 1.
- Inside a computer, each number is represented by a fixed number of 0s and 1s.
- Typical representations of  $-97$  and  $31.125$ :

<b>-97</b>	111111111111111111111111111111110011110
<b>31.125</b>	01000001111110010000000000000000000000

## Numeric Data

- The encoding scheme for integers is different from the one for nonintegers.
- Numbers that contain a decimal point are often called *floating-point numbers*, because they're stored in a form of scientific notation that allows the decimal point to “float.”



## Character Data

- Characters are stored as numeric codes.
- There are several codes in common use; the one that Java uses is named *Unicode*.
- In Unicode, each character has a 16-bit code.

## Character Data

- How “Susan Cole” would be stored in Unicode:

<b>S</b>	0000000001010011
<b>u</b>	0000000001110101
<b>s</b>	0000000001110011
<b>a</b>	0000000001100001
<b>n</b>	0000000001101110
<i>space</i>	0000000001000000
<b>C</b>	0000000001000011
<b>o</b>	0000000001101111
<b>l</b>	0000000001101100
<b>e</b>	0000000001100101

# How Data is Stored?

Data of various kinds, such as numbers, characters, and strings, are encoded as a series of bits (zeros and ones). Computers use zeros and ones because digital devices have two stable states, which are referred to as *zero* and *one* by convention. The programmers need not to be concerned about the encoding and decoding of data, which is performed automatically by the system based on the encoding scheme. The encoding scheme varies. For example, character 'J' is represented by 01001010 in one byte. A small number such as three can be stored in a single byte. If computer needs to store a large number that cannot fit into a single byte, it uses a number of adjacent bytes. No two data can share or split a same byte. A byte is the minimum storage unit.

Memory address	Memory content	
.	.	
.	.	
.	.	
2000	01001010	Encoding for character 'J'
2001	01100001	Encoding for character 'a'
2002	01110110	Encoding for character 'v'
2003	01100001	Encoding for character 'a'
2004	00000011	Encoding for number 3

## Bytes

- Most computers group bits into larger units called *bytes*, which usually contain eight bits.
- A number typically occupies four bytes (32 bits).
- A Unicode character occupies two bytes (16 bits).

## Variables

- Locations that are used to store data within a program are known as *variables*.
- Variables are given names by the programmer.
- It's best to choose a name that suggests what data the variable stores.

*Good names:*

```
fahrenheitTemperature  fahrenheitTemp  
fahrenheit
```

*Poor names:*

```
fahr f a x temp
```

# Types

- Each variable stores a particular type of data.
- In the Fahrenheit-to-Celsius algorithm, the user's input will be a sequence of characters.
- The Fahrenheit and Celsius temperatures will be numbers, possibly with digits after the decimal point.

## Temperature Conversion Algorithm with Variable Names Added

1. Display a message asking the user to enter a Fahrenheit temperature.
2. Obtain the input entered by the user and store it into `userInput`.
3. Convert `userInput` into numerical form and store the result into `fahrenheit`.
4. Calculate the equivalent Celsius temperature using the formula
$$\text{celsius} = (\text{fahrenheit} - 32) \times (5 / 9)$$
5. Convert the value of `celsius` to character form and display the result.

## 1.5 Programming Languages

- Creating programs requires that algorithms be expressed in a highly precise language that's specifically designed for computers.
- Every computer comes with such a language, known as *machine language*.
- Each CPU has its own machine language.
- Machine language is extremely primitive, making it difficult to write even simple programs.
- Q: What are Computer Programs?



## Writing and Executing a Program

- Writing a program in a high-level language requires creating a file containing *source code*.
- Source code is not *executable*—there is no direct way for a computer to follow the commands that it contains.
- *Executing* (or *running*) the program requires special software.
- Approaches to executing a program:
  - Compilation
  - Interpretation

# Programming Languages

Machine Language    Assembly Language    High-Level Language

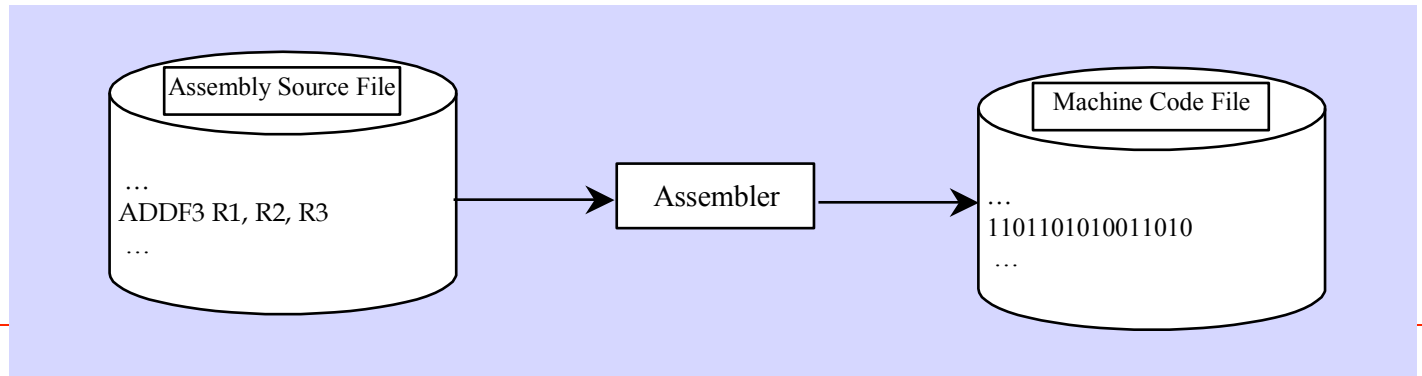
- A set of primitive instructions built into every computer
- In the form of binary code
- Program with native machine language is a tedious process.
  - highly difficult to read and modify.
- For example, to add two numbers:  
1101101010011010

# Programming Languages

Machine Language    **Assembly Language**    High-Level Language

- Developed to make programming easy.
- A program called assembler is used to convert assembly language programs into machine code.
- For example, to add two numbers:

```
ADDF3 R1, R2, R3
```



# Programming Languages

Machine Language

Assembly Language

High-Level Language

- English-like and easy to learn and program.
- For example, the following is a high-level language statement that computes the area of a circle with radius 5:

```
area = 5 * 5 * 3.1415;
```

# Popular High-Level Languages

Language	Description
Ada	Named for Ada Lovelace, who worked on mechanical general-purpose computers. The Ada language was developed for the Department of Defense and is used mainly in defense projects.
BASIC	Beginner's All-purpose Symbolic Instruction Code. It was designed to be learned and used easily by beginners.
C	Developed at Bell Laboratories. C combines the power of an assembly language with the ease of use and portability of a high-level language.
C++	C++ is an object-oriented language, based on C.
C#	Pronounced "C Sharp." It is a hybrid of Java and C++ and was developed by Microsoft.
COBOL	COmmon Business Oriented Language. Used for business applications.
FORTRAN	FORmula TRANslation. Popular for scientific and mathematical applications.
Java	Developed by Sun Microsystems, now part of Oracle. It is widely used for developing platform-independent Internet applications.
Pascal	Named for Blaise Pascal, who pioneered calculating machines in the seventeenth century. It is a simple, structured, general-purpose language primarily for teaching programming.
Python	A simple general-purpose scripting language good for writing short programs.
Visual Basic	Visual Basic was developed by Microsoft and it enables the programmers to rapidly develop graphical user interfaces.

# Interpreting/Compiling Source Code

A program written in a high-level language is called a *source program* or *source code*.

A computer cannot understand a source program.

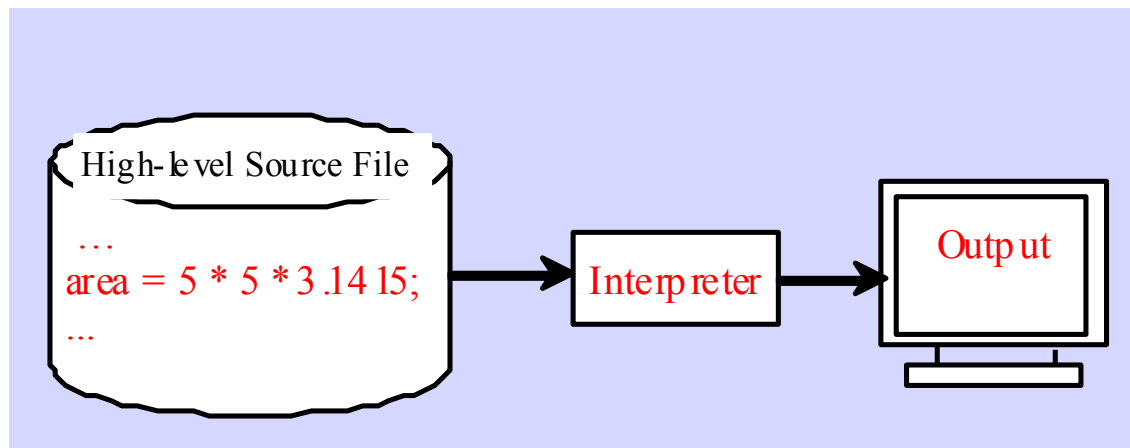
A source program must be translated into machine code for execution. The translation can be done using another programming tool called an *interpreter* or a *compiler*.

## Interpretation

- The program's source code is given to a program known as an *interpreter*.
- The interpreter executes the program without first translating it to machine instructions.
- The interpreter itself is normally a compiled program, so it can execute machine instructions corresponding to the source code.

## Interpreting Source Code

An interpreter reads one statement from the source code, translates it to the machine code or virtual machine code, and then executes it right away, as shown in the following figure. Note that a statement from the source code may be translated into several machine instructions.



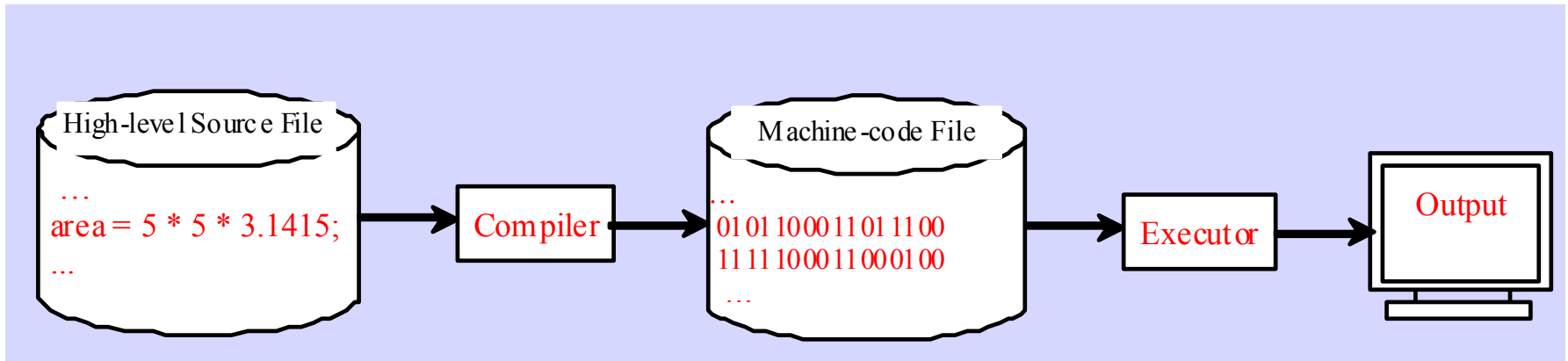


## Compilation

- The program's source code is given to a program called a *compiler*.
- The compiler checks that the source code is valid (obeys the rules of the language) and translates it to machine instructions for a particular CPU.
- The compiled program is stored in a file, and it can be run as many times as desired.

# Compiling Source Code

A compiler translates the entire source code into a machine-code file, and the machine-code file is then executed, as shown in the following figure.



## Java's Approach

- Java employs a combination of compilation and interpretation.
- The Java compiler translates the original program into ***bytecode instructions*** for a computer called the ***Java Virtual Machine***.
- The resulting ***bytecode program*** is then executed by an interpreter.
- One advantage of Java's approach is that programs don't need a particular CPU or operating system.

- Java Standard Edition (J2SE)
  - J2SE can be used to develop client-side standalone applications or applets.
- Java Enterprise Edition (J2EE)
  - J2EE can be used to develop server-side applications such as Java servlets, Java ServerPages, and Java ServerFaces.
- Java Micro Edition (J2ME).
  - J2ME can be used to develop applications for mobile devices such as cell phones.

This book uses J2SE to introduce Java

## 1.6 Why Java?

- Simple
- Object-oriented
- Distributed
- Robust
- Architecture-neutral
- Portable
- Interpreted
- Multithreaded

## 1.7 The Programming Process

1. Write a specification for the program.
2. Design the program.
3. Choose algorithms and decide how data will be stored.
4. Write the program.
5. Compile the program.
6. Execute the program.
7. Debug the program.

## Program Maintenance

- Most of the time, there's an additional step in the programming process: *maintenance*.
- Reasons for maintenance:
  - Fix bugs
  - Add enhancements
  - Adapt to changes in the program's specification
- Maintenance is often the costliest step in the programming process.

## 1.8 What You Need to Know

- A *file* is a collection of related data.
- In many operating systems, a file name includes an *extension* that indicates the type of data stored in the file.
- Common Windows file extensions:
  - .exe (executable program)
  - .doc (document)
  - .gif, .jpg (image)



## File Operations

- Basic file operations:
  - Create
  - Edit
  - Copy
  - Rename
  - Delete
- A file can be created or edited by using an editor or word processor.
- An *editor* is a program that can create or modify a file containing *text* (ordinary characters).
- A *word processor* has the added ability to format text.

## Directories

- A *directory* is a place where files can be kept.
- Directories are also known as *folders*.
- Directories are normally organized in a tree-like fashion, with a “root” directory that contains other directories as well as files.
- Basic directory operations:
  - Create a directory
  - Move from one directory to another
  - List the files in a directory

## Executing Programs

- In a GUI environment, a program is *executed* (or *launched*) by clicking on an icon or by choosing the program from a menu.
- In a text-based environment, a program is executed by typing its name.
- A program that isn't in the current directory can still be executed as long as the operating system knows where to look for the program.

## Exercise: install JDK

- Download JDK
- Set path variable
- Use notepad and cmd to run HelloJava.java

```
public class HelloJava
```

```
{
```

```
    public static void main (String [] args)
```

```
    {
```

```
        System.out.println ("Hello World csc2310!");
```

```
    }
```

```
}
```