Chapter 1: Introduction to Computers and Programming

Starting Out with C++
Early Objects
Sixth Edition

by Tony Gaddis, Judy Walters, and Godfrey Muganda

Topics

1.1 Why Program?

Computer – programmable machine designed to follow instructions

Program – instructions in computer memory to make it do something

Programmer – person who writes instructions (programs) to make computer perform a task

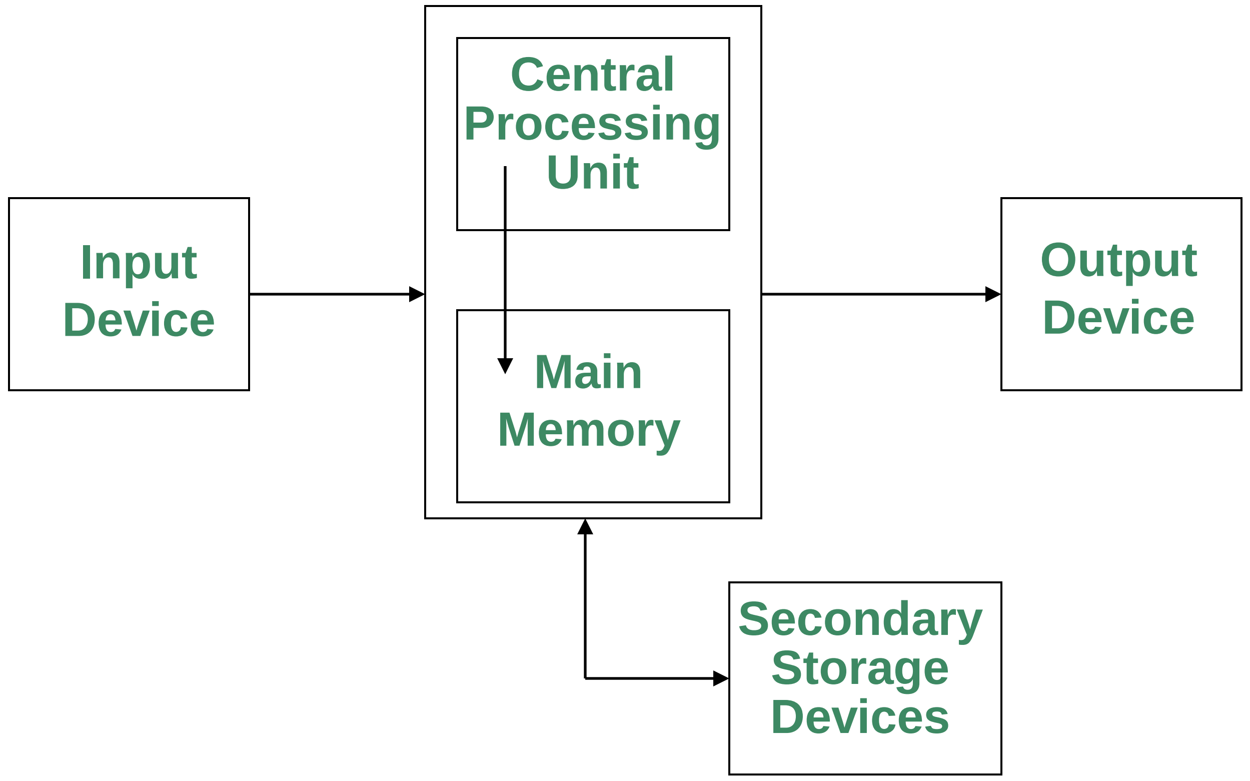
SO, without programmers, no programs; without programs, the computer cannot do anything

1.2 Computer Systems: Hardware and Software

Main Hardware Component Categories

1. Central Processing Unit (CPU)
2. Main Memory
3. Secondary Memory / Storage
4. Input Devices
5. Output Devices

Main Hardware Component Categories



Central Processing Unit (CPU)

Includes

- Control Unit
  - Retrieves and decodes program instructions
  - Coordinates computer operations
- Arithmetic & Logic Unit (ALU)
  - Performs mathematical operations

The CPU's Role in Running a Program

Cycle through:

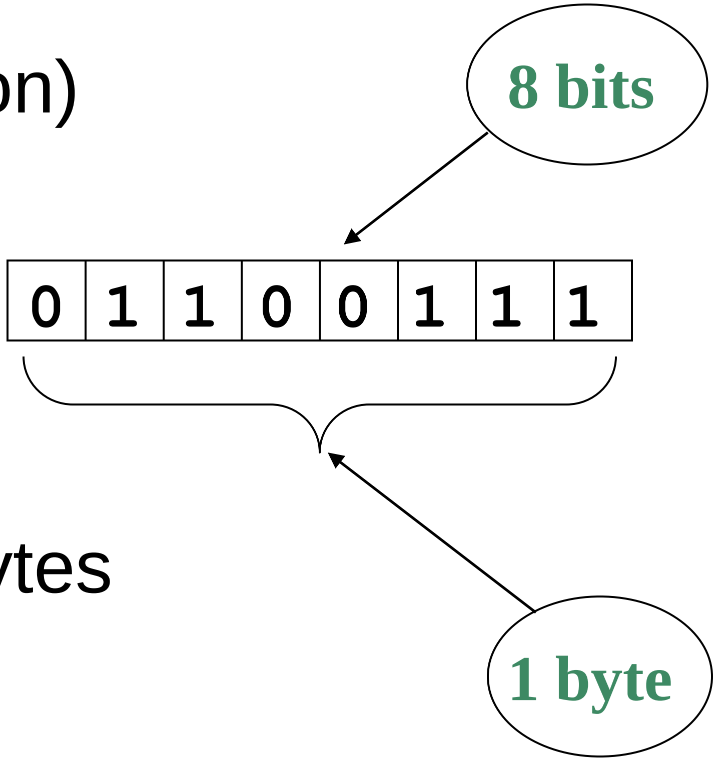
- Fetch: get the next program instruction from main memory
- Decode: interpret the instruction and generate a signal
- Execute: route the signal to the appropriate component to perform an operation

Main Memory

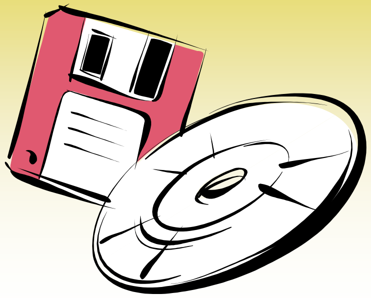
- Holds both program instructions and data
- Volatile – erased when program terminates or computer is turned off
- Also called Random Access Memory (RAM)

Main Memory Organization

- Bit
  - Smallest piece of memory
  - Stands for **b**inary dig**it**
  - Has values 0 (off) or 1 (on)
- Byte
  - Is 8 consecutive bits
- Word
  - Usually 4 consecutive bytes
  - Has an address



Secondary Storage

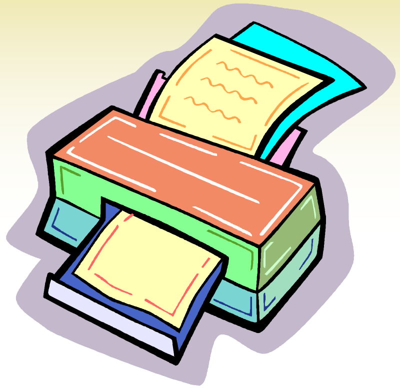


- Non-volatile - data retained when program is not running or computer is turned off
- Comes in a variety of media
  - magnetic: floppy disk, zip disk, hard drive
  - optical: CD
  - flash: thumb or flash drive

Input Devices

- Used to send information to the computer from outside
- Many devices can provide input
  – keyboard, mouse, microphone, scanner, digital camera, disk drive, CD drive, flash drive

Output Devices



- Used to send information from the computer to the outside
- Many devices can be used for output
  - Computer screen, printer, speakers, disk drive, writable CD drive, flash drive

Software Programs That Run on a Computer

- Operating system software
  - programs that manage the compute hardware and the programs that run on them

    Ex: Windows versions, UNIX
- Application software
  - programs that provide services to the user.

    Ex: word processing, games, programs to solve specific problems

1.3 Programs and Programming Languages

- Program

  a set of instructions directing a computer to perform a task

- Programming Language

  a language used to write programs

Programs and Programming Languages

Types of languages

- Low-level: used for communication with computer hardware directly. Often written in binary machine code using 0 and 1.
- High-level: closer to human language

From a High-level Program to an Executable File

a) Create file containing the program with a text editor.

b) Run preprocessor to convert source file directives to source code program statements.

c) Run compiler to convert source program statements into machine instructions.

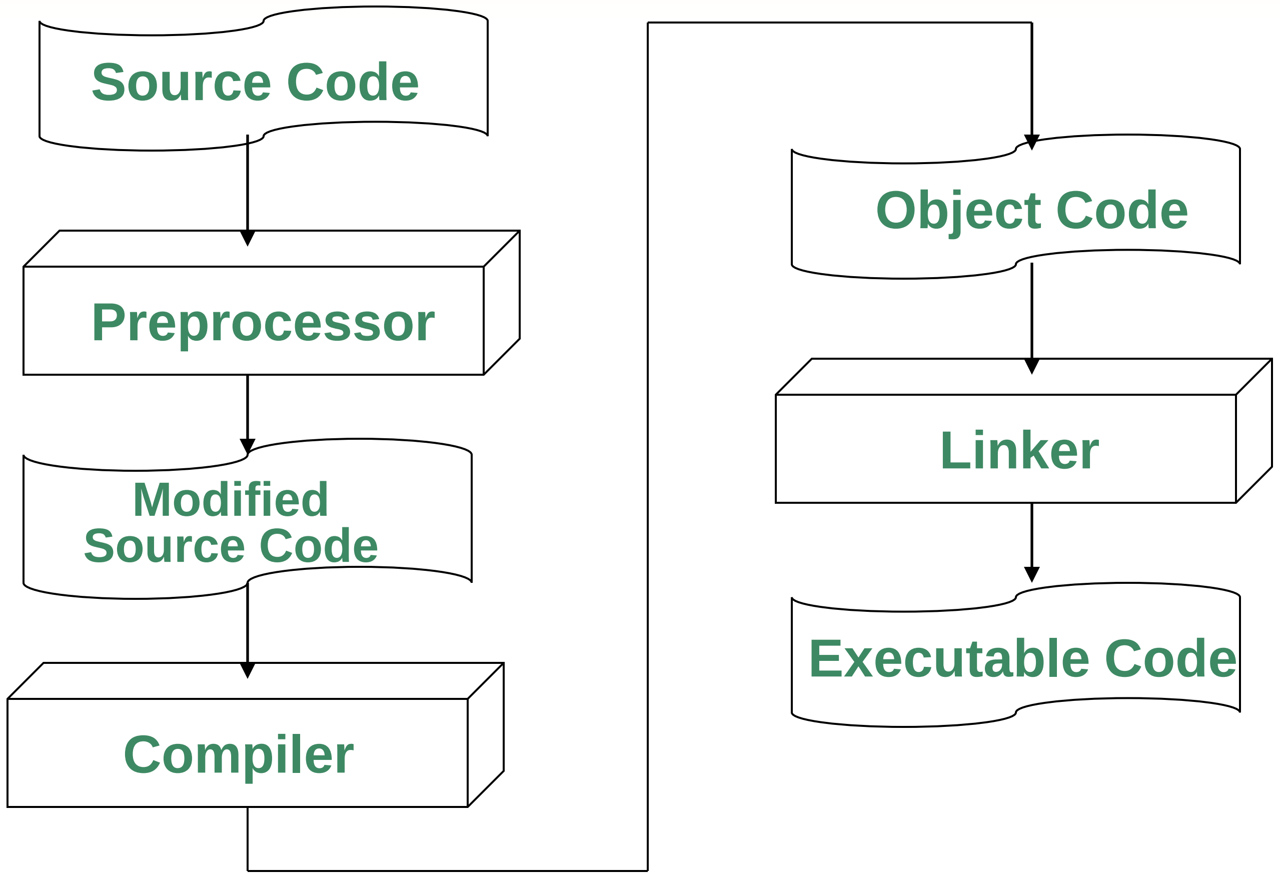
From a High-level Program to an Executable File

d) Run linker to connect hardware-specific library code to machine instructions, producing an executable file.

Steps b)–d) are often performed by a single command or button click.

Errors detected at any step will prevent execution of the following steps.

From a High-level Program to an Executable File



1.4 What Is a Program Made Of?

Common elements in programming languages:

- Key Words
- Programmer-Defined Symbols
- Operators
- Punctuation
- Syntax

Example Program

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
  string name;
  cout << "What is your name? ";
  cin  >> name;
  cout << "Hello there, " << name;
  return 0;
}
```

Key Words

- Also known as reserved words
- Have a special meaning in C++
- Can not be used for another purpose
- Written using lowercase letters
- Examples in program (shown in green):

```
using namespace std;
int main()
```

Programmer-Defined Symbols

- Names made up by the programmer
- Not part of the C++ language
- Used to represent various things
  - variables (memory locations), functions, etc.
- Example in program (shown in green):

```
string name;
```

Operators

- Used to perform operations on data
- Many types of operators
  - Arithmetic: `+, -, *, /`
  - Assignment: `=`
- Examples in program (shown in green):

```
cout << "What is your name? ";
cin  >> name;
```

Punctuation

- Characters that mark the end of a statement, or that separate items in a list
- Example in program (shown in green):

```
string name;
cin >> name;
```

Lines and Statements

In a source file,

- lines are adjacent characters before a carriage return. Empty lines improve the readability of a program.
- statements are instructions to the computer to perform an action. Statements may contain keywords, operators, programmer-defined identifiers, and punctuation. A statement may fit on one line, or it may occupy multiple lines.

Variable

- A named location in the computer (in RAM)
- Holds a piece of data
- Must be *defined* before it can be used
- Example definition:
  - `string name;`

1.5 Input, Processing, and Output

Three steps that many programs perform

1) Gather input data
   - from keyboard
   - from files on disk drives
2) Process the input data
3) Display the results as output
   - send it to the screen or a printer
   - write it to a file

1.6 The Programming Process

1. Define what the program is to do.
2. Visualize the program running on the computer.
3. Use design tools to create a model of the program.

   Hierarchy charts, pseudocode, flowcharts, etc.
4. Check the model for logical errors.
5. Write the program source code.
6. Compile the source code.

The Programming Process

7. Correct any errors found during compilation.
8. Link the program to create an executable file.
9. Run the program using test data for input.
10. Correct any errors found while running the program.

Repeat steps 4 - 10 as many times as necessary.

11. Validate the results of the program.
    Does the program do what was defined in step 1?

1.7 Procedural and Object-Oriented Programming

- Procedural programming
  - Focus is on the process
  - Procedures/functions are written to process data
- Object-Oriented programming
  - Focus is on objects, which contain data and the means to manipulate the data
  - Messages are sent to objects to perform operations

Chapter 1: Introduction to Computers and Programming

Starting Out with C++
Early Objects
Sixth Edition

by Tony Gaddis, Judy Walters, and Godfrey Muganda