CIS1400 – Programming Logic and Technique

Topic 1 → Computers and Programming

Chapter Topics

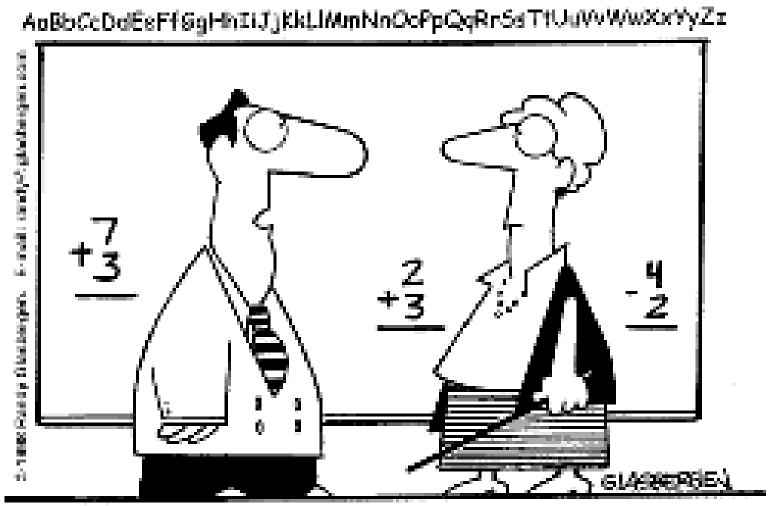
1.1 Introduction

1.2 Hardware

1.3 How Computers Store Data

1.4 How a Program Works

1.5 Types of Software



"My computer is going to die on January 1, 2000. Can you teach me how to add and spell again?"

> "It's hardware that makes a machine fast. It's software that makes a fast machine slow."
> – *Craig Bruce*

1.1 Introduction

Program » A **set of instructions** that a computer follows to perform a task

- We use programs every time we use a computer
- Programs are sometimes called software

Microsoft Word, Adobe Photoshop are examples of software programs.

Programmer » A **person** with the training and skills necessary to **design, create, and test computer programs**

- Programmers are sometimes called software developers

Programmers work in a variety of fields: business, medicine, law enforcement, academics, entertainment, etc.

1.2 Hardware

Hardware » The physical devices that make up a computer

- CPU
  - Most important part of the computer; actually runs the programs
- Main memory
  - Where programs are stored while running; also called RAM
- Secondary storage devices
  - Memory that holds data for a long period of time, even when the computer is shut down such as a memory stick or DVD
- Input devices
  - Keyboard, mouse, scanner
- Output devices
  - Printer, video display

"All parts should go together without forcing. You must remember that the parts you are reassembling were disassembled by you. Therefore, if you can't get them together again, there must be a reason. By all means, do not use a hammer."
– IBM Manual, 1925

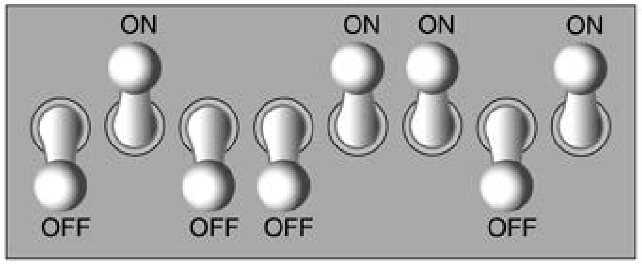
1.3 How Computers Store Data

All data that is stored in a computer is converted to sequences of 0s and 1s, called **bits**.

- There are 8 bits in a **byte**.
- One byte is only large enough to store a letter of the alphabet or a small number

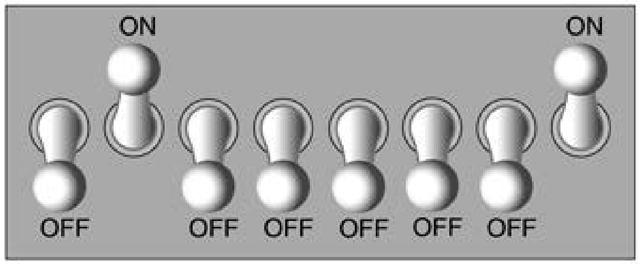
Figure 1.7 Bit patterns for the number 77 and the letter A

01001101 = 1 + 4 + 8 + 64 = 77



The number 77 stored in a byte.

01000001 = 1 + 64 = 65



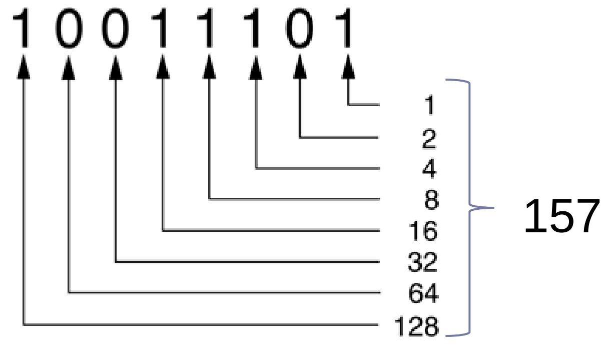
The letter A stored in a byte.

1.3 How Computers Store Data

Storing numbers

- If a bit is turned off, it represents a binary 0.
- If a bit is turned on, it represents a binary 1.
- Each digit in a binary number has a value assigned to it.
- Each value that is turned on (1) is then added up to find the number that it represents.

Figure 1.9 The values of binary digits



1.3 How Computers Store Data

Storing characters

- When a character is stored, it must first be converted to a numeric code.
- ASCII is the common coding scheme.
  - It has a 128 set of common codes
  - Example: a capital A is 65, or binary 01000001
- Unicode capable of representing many writing systems currently in use today
  - ASCII incorporated as first 128 character of Unicode

π Я 音 æ∞

1.4 How a Program Works

A computer's CPU can only understand instructions that are written in **machine language**.

The process of interpreting and executing these instructions is referred to as the **machine cycle**.

1. Fetch – reads the next instruction from memory into the CPU.
2. Decode – the CPU decodes the instruction to determine what should happen first.
3. Execute – perform the operation.
4. Store – write result to memory.

Both statements print the letter "A" 1000 times on the screen.

Machine Language

```
  169 1 160 0 153 0 128 153 0 129 153 130 153 0 131 200 208 241 96
```

BASIC

```
   5 FOR I=1 TO 1000: PRINT "A";: NEXT I
```

1.4 How a Program Works

Machine Language to **Assembly Language**

- Instead of binary numbers, assembly language uses short words that are known as mnemonics.
  - mul means to multiply
  - add was used for addition
- An assembler was then used to translate it into machine language.

Assembly Language

```
MOV AX, 47104
MOV DS, AX
MOV [3998], 36
INT 32
```

1.4 How a Program Works

Assembly Language to **High-Level Languages**

- Allowed programmers to create powerful and complex programs without knowing how the CPU works.
- Common High-Level Languages
  - C and C++
  - C#
  - Java
  - Visual Basic
  - Python
  - Ruby

See Table 1-1 for more examples of programming languages.

1.4 How a Program Works

A **compiler** is a program that ***translates*** a high-level language program into a separate machine language program, or an ***executable***.

A **interpreter** is a program that both ***translates and executes*** the instructions.

Compiled programs *generally* execute faster than interpreted because they are already translated entirely to machine language.

1.4 How a Program Works

The statements that a programmer writes are called **source code**.

The code is then **compiled** or **interpreted**.

All **syntax errors** must be corrected in order for the statement to execute.

- Syntax errors are mistakes with the rules of the language.

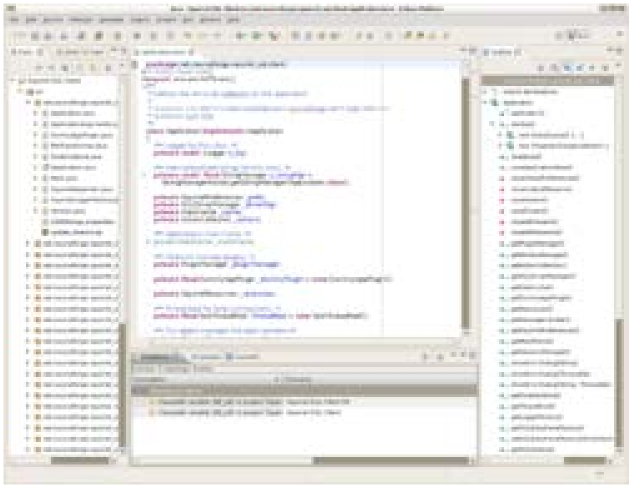
Compiler checks for syntax errors after **all** statements have been entered.

Interpreter checks for syntax errors after **each** statement has been entered.

1.4 How a Program Works

Integrated Development Environments

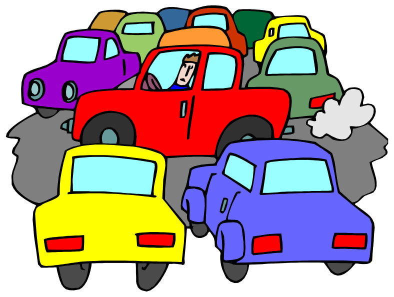
- IDEs are specialized software packages that have a text editor, a compiler, and tools for testing programs and locating errors.
  - Microsoft Visual Studio
  - Eclipse
  - Dev C++
  - NetBeans
  - jGRASP



1.5 Types of Software

Programs generally fit into one of two categories

- System software
  - The set of programs that control or enhance the operation of a computer such as an Operating System, Utility Programs, or Software Development Tools.
- Application software
  - Programs that make a computer useful for every day tasks such as Microsoft Word, email programs, and Web browsers.



"If you think your users are idiots, only idiots will use it."
– *Linus Torvalds*