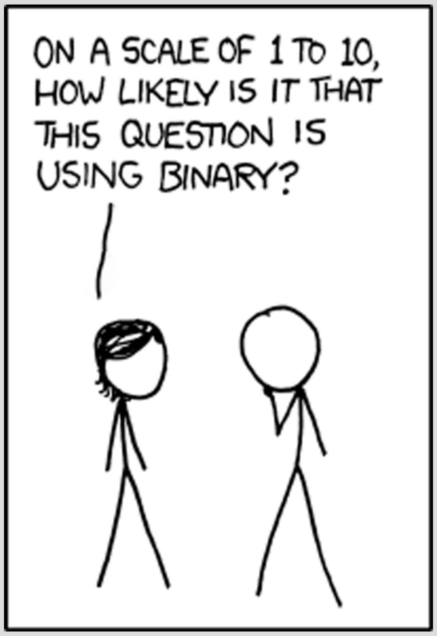
CMSC131

Today's Computers

and

Computer Programming

Information Storage


ON A SCALE OF 1 TO 10, HOW LIKELY IS IT THAT THIS QUESTION IS USING BINARY?

Information Storage

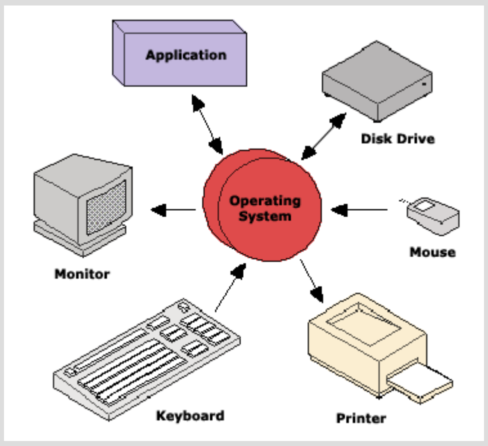
At the lowest levels, all of today's computers store and transmit information in a binary representation whether in active memory, stored on drives, or sent across a network.

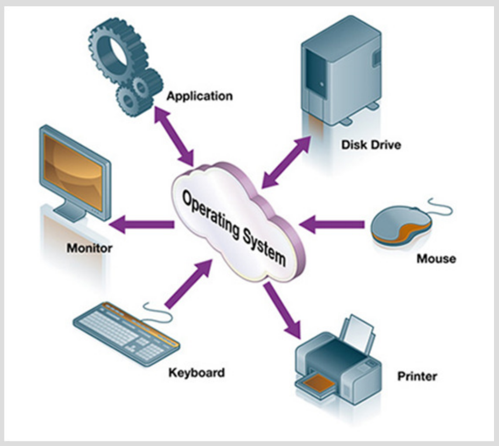
We represent binary values as 0 and 1 (also knows as BITs) but those values can be represented in other ways such on -vs- off, North/South -vs- South/North, pit -vs- no pit, low -vs- high tone.

A group of 8 bits is referred to as a byte. A group of 1024 (2^{10}) bytes is referred to as a kilobyte (K).

Hardware / OS / Software

The graphics might change but the concept is still the same…





Hardware / OS / Software

Hardware is what makes up the physical machine and its peripherals.

The **Operating System** is the piece of software that runs and controls the computer hardware. It acts as an interface between application programs and the hardware being used to run them.

While an operating system is a type of software known as *systems software* what is most commonly referred to generically as **Software** is also called *application software*.

Hardware

The key elements of a computer that we mostly talk about are pretty much the CPU, its main memory, and secondary storage.

Some things to consider are:

- Purpose -vs- Brand
- Common interfaces like USB ports, VGA, etc. and how “standards” change over time (Firewire, USB3, HDMI, etc.)
- Software drivers (basically “teach” the OS how to talk with new things)

Which is not computer hardware?

20% A. Mouse

20% B. Touchscreen

20% C. Camera

20% D. Post-It Note

20% E. All are computer hardware

Response Counter

Operating Systems

When thinking about the relationship between the operating system and the hardware, some things to consider are:

- Dual boot systems
- Software usually being compiled to a platform
- Virtual machines, JVM, bytecode idea

Software

When discussing software, things to consider include:

- Accuracy, reliability, and sustainability
- Scalability
- Proprietary vs. Open Source vs. Free software (and data formats)
- Interface: Text vs. GUI vs. Voice vs. ???

Networks

While not part of the computer itself (beyond the NIC) networks (LANs, the Internet, ad-hoc) are a key element in computing today.

Information is transmitted as BITs over the network. These transmissions might guarantee delivery (TCP) but some don't (UDP).

Things such as Internet bandwidths are often expressed in terms of megabits per second, in which case it is essentially a measurement of millions of bits per second being transmitted.

Some Example Hardware Details

- **C**entral **P**rocessing **U**nit (eg: Intel Core i7)
- Main Memory (eg: 4x1GB DIMM)
- Secondary Storage (eg: hard drive)
- **I**nput/**O**utput Devices (eg: keyboard)
- Networking (eg: wifi card)

When you double-click…

When you double-click an application icon, the operating system will begin to read that application’s code into main memory and then execute the code (instruction by instruction) on the CPU.

Things to consider in the future:

- What if an instruction says to use a peripheral?
- What if the operating system is configured to use virtual memory?
- What if the application code isn’t in the same language that your CPU speaks?

Information in Main Memory

Address	Value
0xF89E2	01010101
0xF89E3	11001100
0xF89E4	10010111
0xF89E5	00001100

Can think of it as a table of bytes where each byte has an address.

Each byte can store Base 2 values between 00000000 and 11111111 (so Base 10 values 0 to 255).

The range of addresses supported by your OS limits how much memory you can install.

Everything with computers must get converted to Base 2 (Base 10 integers, floating point numbers, characters, pictures, etc).

What is programming?

Is/was “programming your VCR/DVR or thermostat or porch lights” and example of programming?

Is making a web page an example of programming?

In this course we will define programming as taking some algorithm (a recipe or set of rules by which to accomplish a task) that we have ***been given*** or that we have ***created ourselves*** and representing it in a language that can then be turned into lower level code to be executed by a computer.

Note that this means there is some computer science going on before we get to the programming.

Writing and Following Instructions

A computer program should “start” with a clear specification of what tasks it needs to support and then be implemented in an appropriate language and tested for correctness.

While knowing how to use a language to implement things is of course required, ambiguities and assumptions regarding the specifications can often be among the largest obstacles.

Writing and Following Instructions

There is a direction-following task posted on ELMS, to be done before Friday at 1pm…

Programming Language "Generations"

- machine code 000000 00001 00010 00110 00000 100000
- assembly language (slightly more human-readable) and that was converted into machine code by a program
  addcc %r3,%r5,%r3
- more human-friendly languages (Fortran, COBOL, Lisp, C, C++, Java) x = Math.pow(2,10);
- application/domain-specific languages (SPSS, SQL, Mathematica) SELECT * FROM Book WHERE price > 100
- logic programming solve(A) :- clause(A,B), solve(B)

Compilers and Interpreters

For anything other than first-generation code, a program is used to convert it into machine code.

Compilers can be used to do all of that conversation in advance. Once the executable exists for a platform, it can typically be run without any special program on a specific machine with that platform.

Interpreters are essentially used to do the conversion in real-time as the program is being run.

There are sometimes "blurred lines" between the way these two types of tools are used.

Compilation Errors

One of the things that is done during compilation process is checking that each line of code is actually valid. If it is not, the compiler will report an error and then won't generate the machine code for the module with the error.

- Some editors, such as Eclipse will do a "spell-check" style check of the syntax before you try to compile and run the program.

It is important to note that the compiler does not look for logic/semantic errors, only syntax errors.

Common Elements of Programming

- Variables to store information in memory and ways to read/write information from/to them.
- Input/Output commands.
- Mathematical and logical operators.
- Conditional statements.
- Iterative statements.
- Subroutines to hold a set of instructions that accomplish a specific goal.
- Ways to build a structure to hold a small, related group of information.

Object Oriented Programming

Classes can be used to describe a structure to hold information (objects) as well as operations that can be performed (methods) and will often serve a dual-purpose of doing both (describing the structure of the object and the operations that can be performed on them).

We will be seeing all of these this semester and discuss what they are and how they are used as we learn Java programming.

Copyright © 2010-2019 : Evan Golub