COMP 204: Computer Programming for Life Sciences

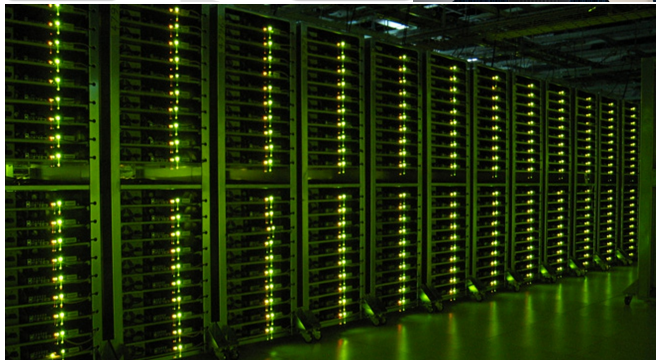## What is a computer: CPU, RAM, storage, communication. Binary numbers, instructions

Mathieu Blanchette
based on slides by Yue Li, Mathieu Blanchette, Christopher Cameron, and Carlos Oliver
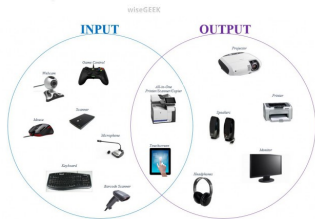
Reminders

- Tutorials: Monday and Tuesday, 5:00-6:00pm, TR3120.
- First quiz available at 12:25 today, due at 23:59.

Modern computers appear in variety of form factors

Key physical components of modern computer devices

Modern computers consist of two classes of components:

1. **Hardware:** physical machinery
2. **Software:** instructions and data executed by the hardware (focus of the course)

Typical hardware components are:

- ► Random Access Memory (RAM)
- ► Mass storage device
- ► Input device(s)
- ► Output device(s)
- ► **Central Processing Unit (CPU)**

Central Processing Unit (CPU)

▶ Commonly referred to as the 'brains' of a computer

▶ Responsible for executing sets of software instructions (called 'programs')

▶ Programs take input from input devices, process data, and provide output to an output device

▶ CPUs aren't limited to desk/laptop computers

    ▶ Can be found in mobile phones, watch, media players, gaming consoles, loundry machines, etc.

Moore's Law: transistors doubles every two years

Moore's Law – The number of transistors on integrated circuit chips (1971-2016)

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years.
This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are strongly linked to Moore's law.
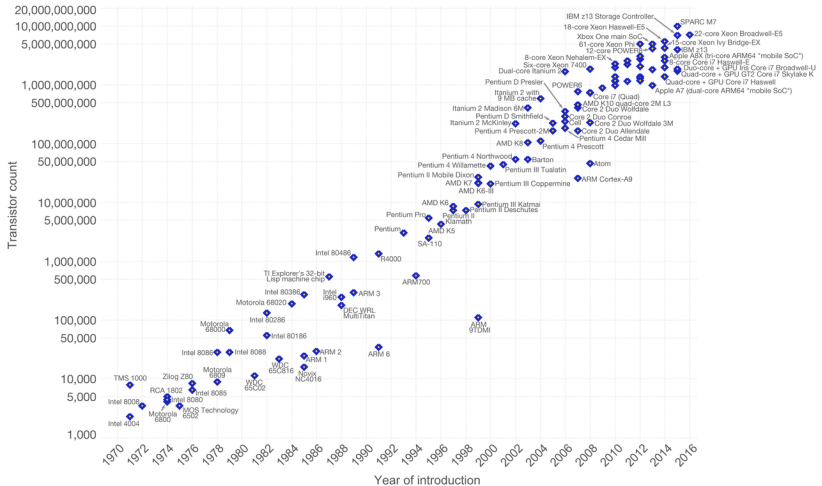
Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)
The data visualization is available at OurWorldInData.org. There you find more visualizations and research on this topic.   Licensed under CC-BY-SA by the author Max Roser.

Computer memory (or 'primary storage')

- Refers to hardware devices that allow for the storage of data and programs, at least temporarily
- These devices operate at high-speeds, which is a distinction from mass storage devices

Volatile memory
Electrical power must be maintained for stored information not to be lost by memory (e.g., **Random Access Memory (RAM)**).
Typical laptop computer: 8 Gb = 8 Billion bytes = 8,000,000,000.

Non-volatile memory
Hardware retains stored information even when not powered.
Examples include:

- Flash memory
- Read Only Memory (ROM)
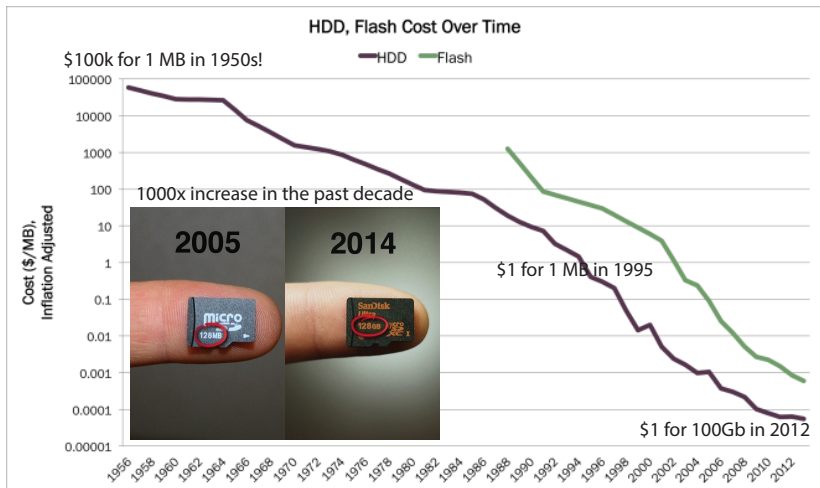
Mass storage (or 'secondary storage')

Differs from primary storage in the following ways:

- ▶ Typically, not directly accessible by the CPU
- ▶ Much slower to access
- ▶ Non-volatile
- ▶ Typically costs much less (per Giga-byte) than computer memory
- ▶ Much larger in capacity than computer memory. Typical laptop: 250 Gb = 250 Billion bytes.

Examples of storage devices:

- ▶ hard disk drives
- ▶ optical (CD/DVD/Blu-ray)
- ▶ punch cards

Drastically decreasing cost of mass storage
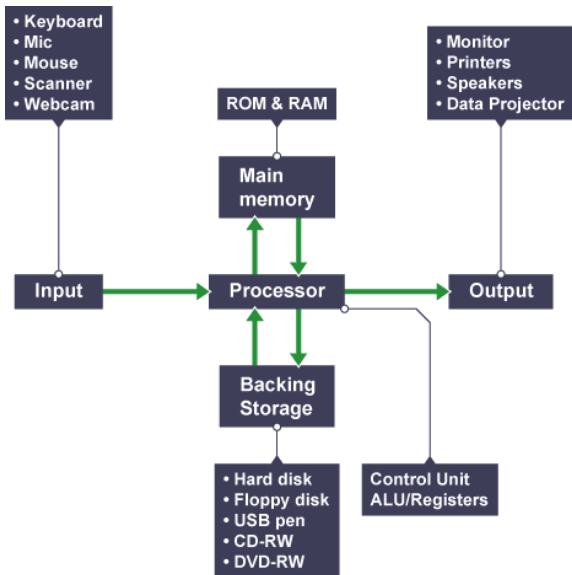

source:

https://www.schoolsofkingedwardvi.co.uk/ks2-computing-computing-theory-5-computer-networks/

Input/output devices allow communicate with computers
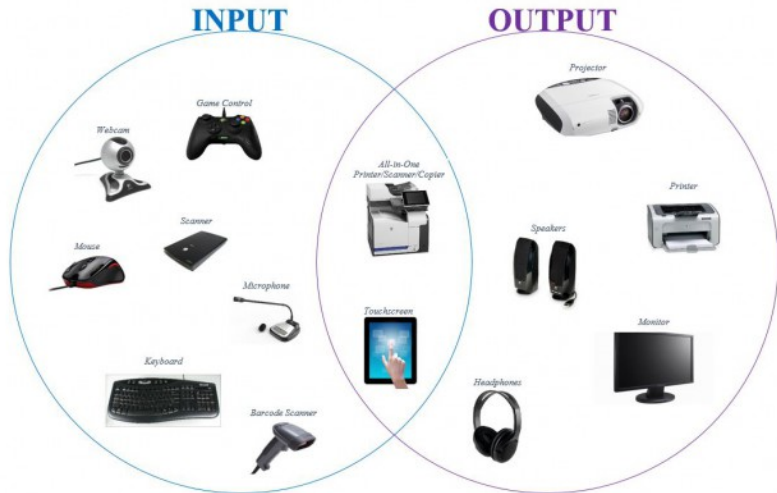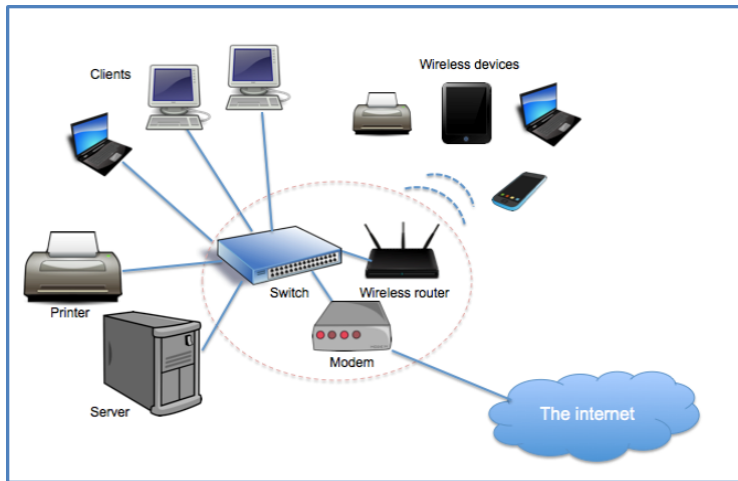
Summary of computer components

Computer network

A computer network is a number of computers linked together to allow them to "talk" to each other and share resources. Networked computers can share hardware, software and data.

Binary numbers

All data stored in primary and secondary storage is stored as <u>bits</u>:
0 and 1. Sequence of bits can be used to represent:

1. Numbers (next slides)
2. Text (ASCII characters): 'A' = 10000001, 'B' = 10000010, ...
3. Images: one pixel at a time, RGB encoding (e.g., Red is rgb(255,0,0), Yellow is rgb(255,255,0), etc)
4. Digital audio: encoder and decoder for rate, bit depth and bit rate
5. Any type of information can be stored as bits!!

But... dealing directly with bits is cumbersome for humans. That's why the computer's operating system allows you to interact with the computer with text.

Back to basics - The Decimal number system

- The number system that you use every day
- Contains ten digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9
- How do we count to numbers greater than 9?
    - Start counting: 0... 1... 2... 3... 4... 5... 6... 7... 8... 9... ?
    - We're out of digits
    - Add a second column worth ten times the value of the first
    - Continue counting: 10... 11... 12... 13... and so on

Expanded notation:
$365_{10} = (3 \times 10^2) + (6 \times 10^1) + (5 \times 10^0)$
$2032_{10} = (2 \times 10^3) + (0 \times 10^2) + (3 \times 10^1) + (2 \times 10^0)$

Binary number system

- ▶ The binary number system is the exact same <u>except</u>
- ▶ Contains **only two** digits: 0 and 1
  - ▶ *'It was just a dream, Bender.*
    There's no such thing as two' - Philip J. Fry I
- ▶ How do we count to numbers greater than 1?
  - ▶ Start counting: 0... 1... ?
  - ▶ We're out of digits...again
  - ▶ Let's try adding a second column again
  - ▶ Continue counting: 10 (2)... 11 (3)...

Expanded notation:

$101101_2 = (1 \times 2^5) + (0 \times 2^4) + (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) = 45$

Converting from decimal to binary

How to go from decimal to binary? $233_{10} = ?_2$
The algorithm to convert decimal to binary is called "Divide by 2"
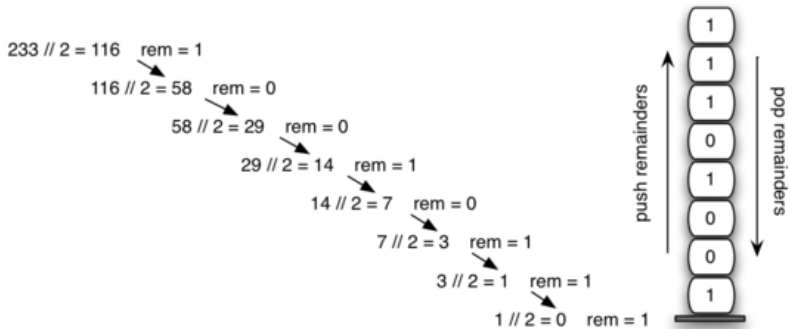
So: $233_{10} = 11101001_2$
Check: $233_{10} = (1 \times 2^7) + (1 \times 2^6) + (1 \times 2^5) + (0 \times 2^4) + (1 \times 2^3) + (0 \times 2^2) + (0 \times 2^1) + (1 \times 2^0)$

Let's do another example

$89_{10} = ?_2$

	result	remainder
divide 89 by two	44	1
divide 44 by two	22	0
divide 22 by two	11	0
divide 11 by two	5	1
divide 5 by two	2	1
divide 2 by two	1	0
divide 1 by two	0	1

So: $89_{10} = 1011001_2$.
Check : $89_{10} =$
$(1 \times 2^6) + (0 \times 2^5) + (1 \times 2^4) + (1 \times 2^3) + (0 \times 2^2) + (0 \times 2^1) + (1 \times 2^0)$.

Computers operate on Bytes

- Computers almost always operate on at least 8 bits at a time.
  1 byte = 8 bits. So...
  $0_{10} = 00000000_2$
  $1_{10} = 00000001_2$
  $89_{10} = 01011001_2$

- To store larger numbers, we need more bits, e.g. 16 or 32.
  Programmers can choose how many bits they want to use:
  8 bits = byte. 16 bits = word, 32 bits = double word, etc.

- For example, our computers often operate on a 64-bit
  operating system, that means it uses 64 bits to represent
  numbers and instructions. The largest positive integer a
  64-bit system can represent by binary is:
  0111111111111111111111111111111111111111111111111111111111111111
  which is $2^{63} - 1 = 9,223,372,036,854,775,807$ or over 9,223
  trillion)

Beyond positive integers

Signed integers: How to represent -13 on a byte?

- ▶ The Most Significant Bit (MSB) = leftmost bit is used to represent the sign: 0 = positive, 1 = negative
  $+13_{10} = 0$0001101 as a signed byte
  $-13_{10} = 1$0001101 as a signed byte

- ▶ So to know the value represented by a byte, we must know if it is a signed byte or an unsigned byte:
  unsigned byte value of $10001101 = 141_{10}$
  signed byte value of $10001101 = -13_{10}$

- ▶ How does computer know whether it is an unsigned or signed byte?
  Answer: The type unsigned and signed are stored in other memory location and specified by the programmers.

Computer instructions

- How to tell a computer what to it is supposed to do?
  Give it *instructions*.
- Instructions inform a computer's processor to perform specific basic operations
  - Add/subtract/multiply/divide two numbers,
  - Retrieve or store value at specific address in memory
  - Jump to another instruction
  - etc.
- Instructions are represented in binary (usually 32 or 64 bits)
- Computers can be programmed by writing the sequence of instructions to be performed (using assembly language). This very tedious, error prone.
- Instead, programmers use *high-level languages* (i.e., Python, Java, C) that are easier for humans to write and understand. Programs written in text get translated to instructions by an interpreter (for Python) or a compiler (for Java, C++).

Python code for converting decimal to binary

```python
from pythonds.basic.stack import Stack

def decimal_to_binary(decNumber):
    remstack = Stack()

    while decNumber > 0:
        rem = decNumber % 2
        remstack.push(rem)
        decNumber = decNumber // 2

    binString = ""
    while not remstack.isEmpty():
        binString = binString + str(remstack.pop())

    return binString
```