Computer Systems 1

Lecture 1

Analogue and Digital Representation

Dr. John T. O'Donnell
School of Computing Science
University of Glasgow

Topics

1. About the course
2. Computer systems
3. Data representation
   - Analogue computing
   - Digital computing
   - Bits

CS1S: Computer Systems 1

- Check Moodle page frequently: all course materials will be there
  - Computing Science
  - Level 1
  - Computing Science 1S - Systems (Semester 2)
- Schedule
  - Two lectures every week
  - One 2-hour lab every week (but no lab in first week)
  - One quiz on Moodle every week
- Assessment
  - 80% degree examination in May
  - 10% quiz average
  - 10% assessed exercise in assembly language programming

Time management

- You need regular study, so budget your time.
- You can't learn computer systems overnight! Work at it steadily, every week
- Revise the lecture slides and handouts every week.
- Think about tricky points and solve problems.
- Prepare the lab exercises before your lab.
- Don't fall into the trap of spending all your time trying to get more marks on assessed exercises.

Study tips

- Focus on main concepts
- Don't spend time memorising numbers!
- Look over slides again shortly after lecture
- Develop a working understanding
- Try some simple examples

Voting on questions (YACRS)

- We will use the phone (tablet, laptop) voting system that you used last semester in programming
- Please bring your device (smartphone is best) to next lecture

Computer systems

- How computers work
- A broad overview, all the way from electronics up to systems software

Why learn about systems?

- Intellectual: a fascinating technology and a central aspect of our culture
- Practical: knowledge of systems is helpful in programming and improving application performance
- Future-proofing: you often need to design software solutions for the systems we'll have several years in the future
- Understanding the basic concepts of systems will help you to use computers more effectively
- Computing is a deep subject, with interesting history, ideas, theory, and philosophy. It's more than just how to use packaged software!

The aim of Computer Systems

- To implement what we want using what we have
- What we have is digital electronics – transistors and wires
- What we want is a usable computer system supporting programming and applications

The challenge: Complexity

- There is a huge distance between the simple electronic components and the usable systems!
- Computer hardware
  - a processor contains around 10^7 electronic components (ten million)
- System software
  - A modern operating system contains around 10^7 lines of code (largely C and C++)

Levels of abstraction

From highest level down to lowest level

- Applications: spreadsheets, databases, your own programs, ...
- Networks
- Programming languages and compilers
- Operating System
- Computer Architecture: machine language
- Digital Circuits
- Semiconductors

Digital circuits

This subject consists of

- The basic "building block" components
- The way they behave when connected together
- How to combine components in order to make a circuit that does something useful

Surprisingly,

- There are only a few types of component
- They are very simple
- Yet when connected the right way, they give extraordinary behaviour

Computer architecture

- The *machine language*
- The interface between low-level circuits and high-level software
- A programming language executed directly by the computer hardware
- Simple enough that it's possible to design a digital circuit that executes machine language
- Powerful enough that high level languages can be translated into it

Operating Systems

- System software that provides lots of facilities needed by applications, which are too complicated to provide directly in the hardware
- Files, protection, processes, threads, virtual memory, communication,

Networks

- Local and wide area networks
- Software organisation: Protocols
- Internet

Data representation

- Data types used in programming
  - Numbers, character strings, booleans
  - Data structures
  - Computer programs
- These must all be represented in the computer hardware, which is based on basic electronic components

Physical quantity (e.g. voltage) represents information

- We can use the voltage on a wire to represent information
- Computation then requires manipulation of these voltages
- There are two completely different approaches:
  - Analogue
  - Digital

Analogue

- The idea: make the variables in your problem proportional to a physical measure (usually either length, angle, rotation, or voltage)
- For example, to represent the real number x, use
  - x volts
  - x meters of distance
  - x degrees of rotation
- The physical measure is an "analogy" or "analogue" to the measures in your problem

Using length to represent a number: slide rule

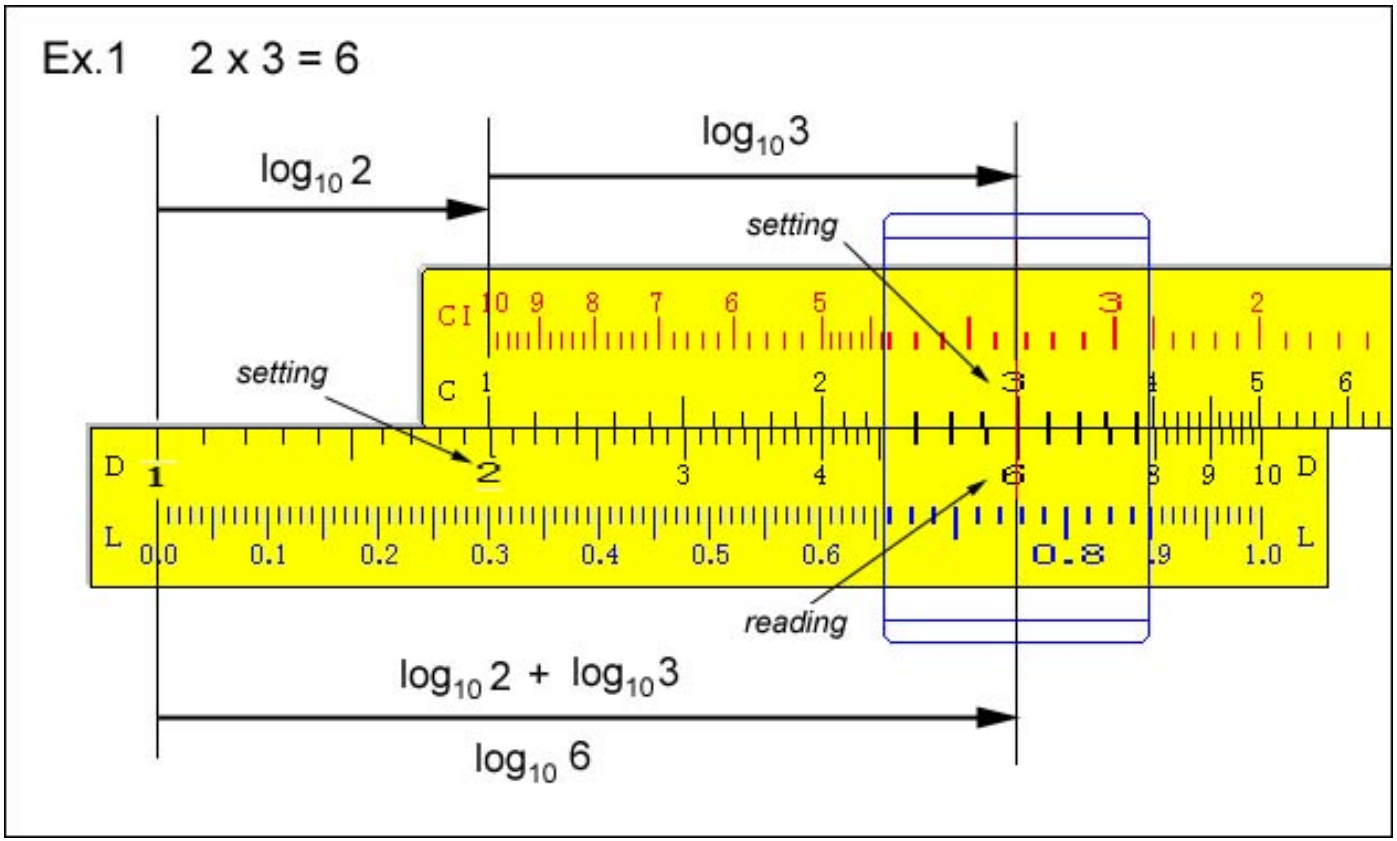
- You can add lengths by placing sticks end to end
- By marking points on two rulers, and lining them up, you can add
- Even more clever: by adding logarithms, you can multiply

$$\log(a \times b) = \log a + \log b$$

- Slide rules use this idea to perform multiplication, division, and even more complex calculations

A high quality advanced slide rule

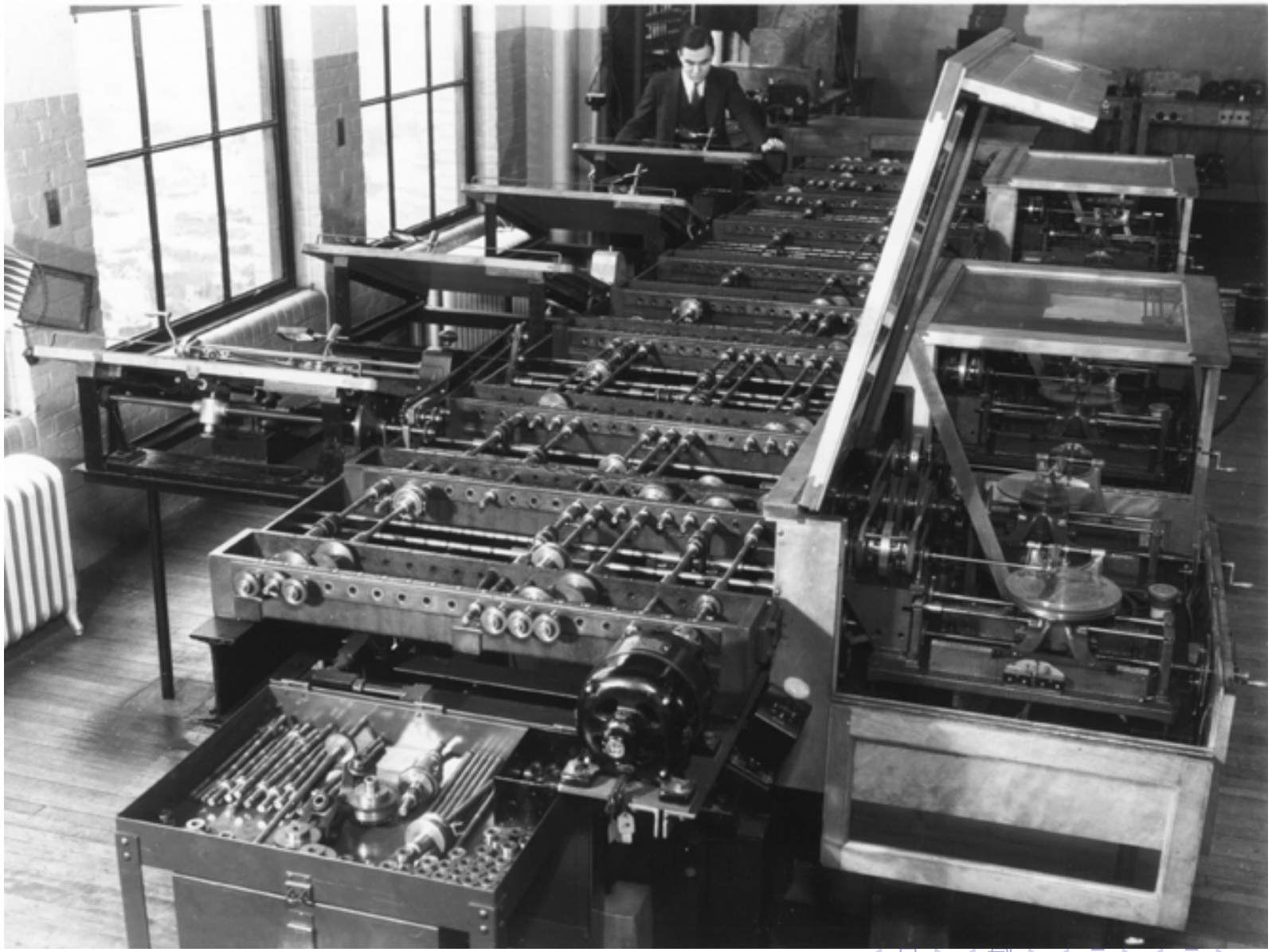
Multiplication by adding logarithms



http://www.ies-math.com/math/java/misc/slide_rule/slide_rule.html

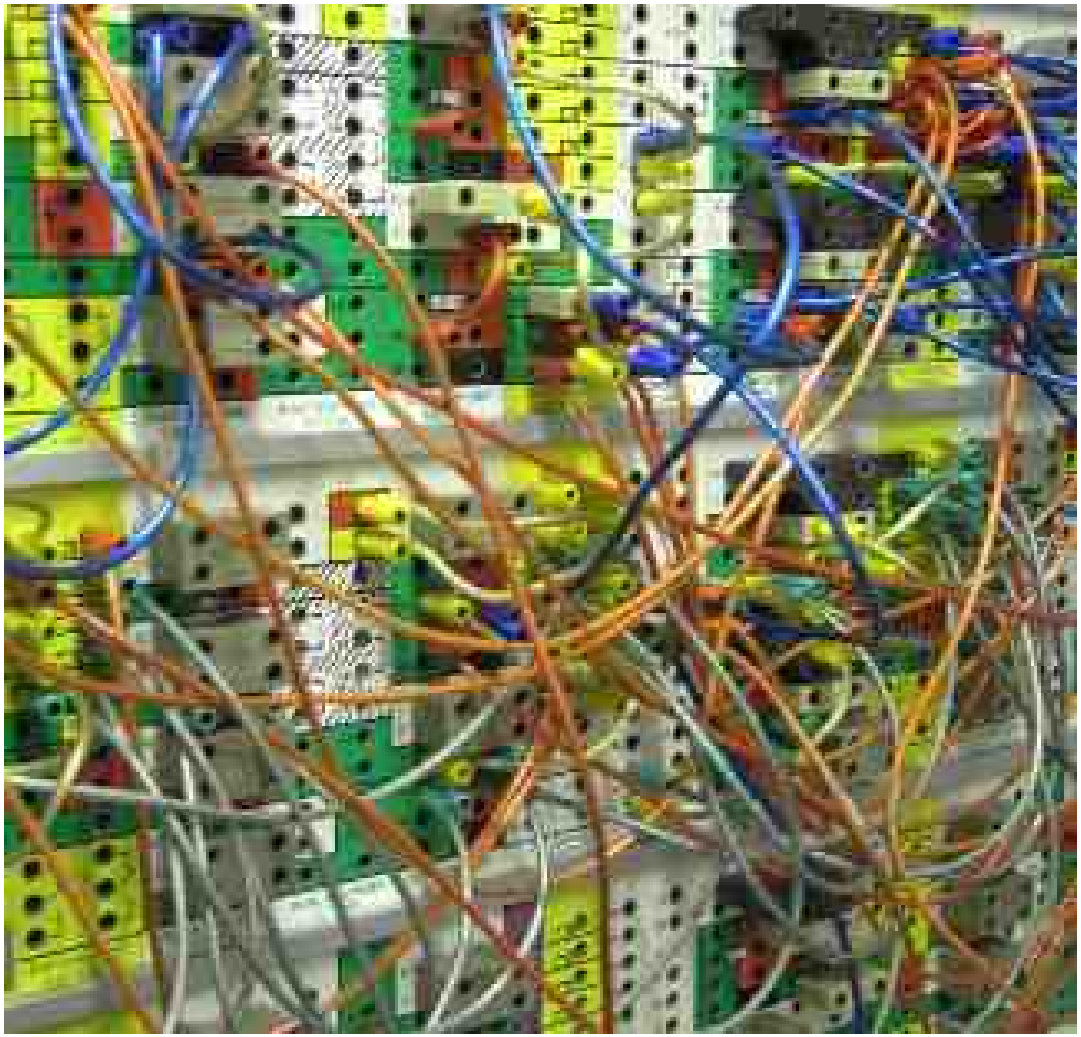
Using rotation in a mechanical analogue computer

Vannevar Bush differential analyzer (1930s)

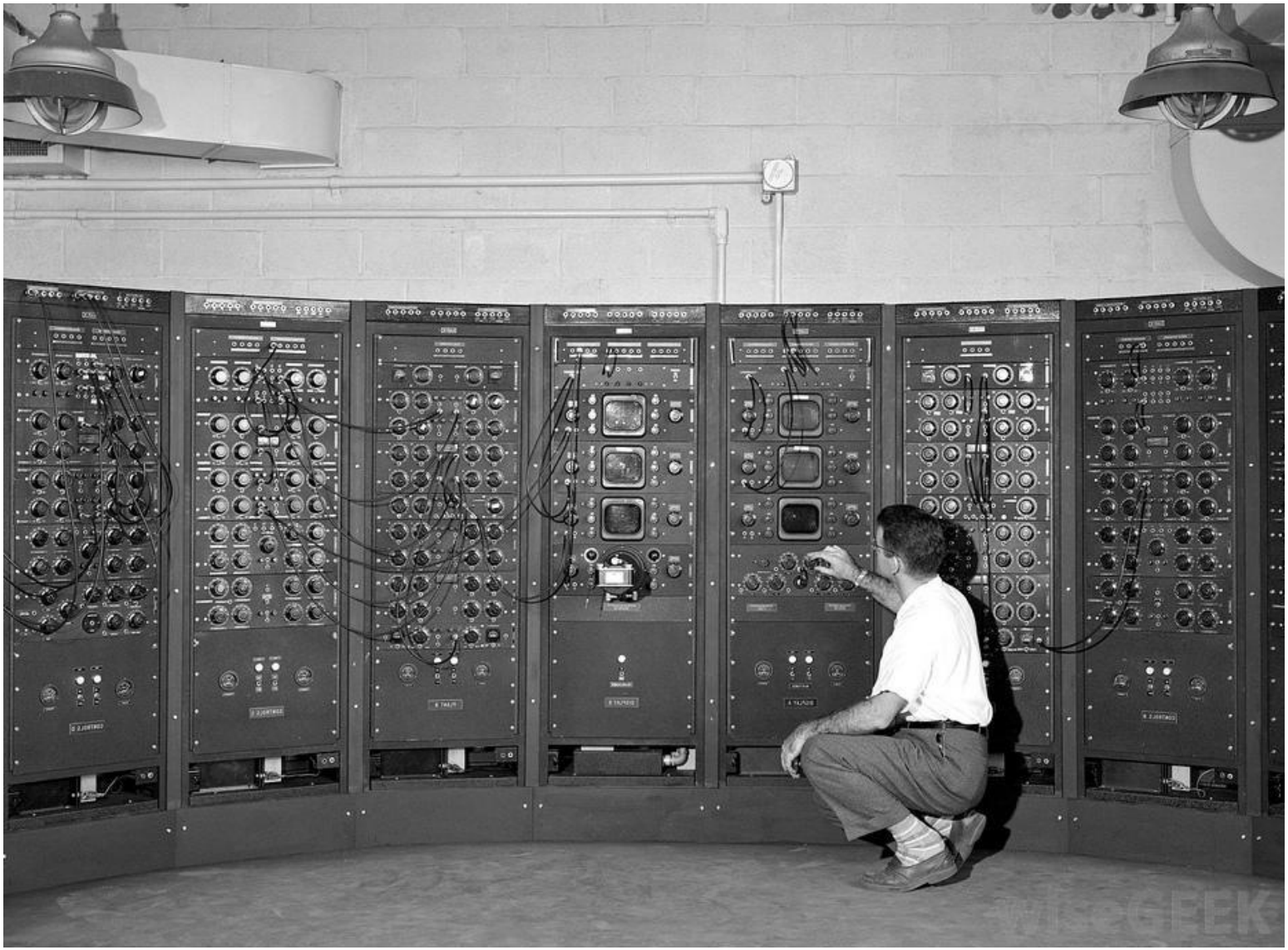


Electronic analogue computer

Programming by wiring together functional units



A large scale analogue computer



Assessment of analogue

- Can be extremely fast for differential equations
- But there are serious disadvantages...
  - Limited precision
  - After lots of calculations, errors will accumulate
  - It's hard to represent data other than real numbers (integers, strings, etc.)
  - Large numbers can be shocking!

Digital computing

- Perhaps even older than analogue computing!
- How do you add two numbers (if you don't know the addition table)? *Count on your fingers!*
- This is *digital computing*
- The word *digit* means *finger*

Advantages of digital representation

- You can get as much precision as you want, just by using more digits to represent a number
- Good immunity to noise
- Errors don't accumulate after lots of calculations
- All datatypes can be represented (integers, strings, objects, and much more)

Binary Digits: *Bits*

- Voltage is used to represent information in digital circuits
- We call one of the standard voltages 0 and the other 1
- This unit of information is a Binary Digit — a Bit

Digital computing with bits

- Use just two voltages. Doesn't matter what they are, e.g.:
  - 0 volts and 5 volts (TTL circuits)
  - -2.5 volts and 2.5 volts (CMOS circuits)
- The exact voltages are unimportant: all that matters is that we can tell them apart!
- Circuits are simpler and more reliable if we just use two different voltages
- Use these two values to represent digits, and form numbers from strings of digits

Flip flop

- A flip flop is a basic digital circuit that can
  - Take a data bit and store it ("remember it")
  - Remember the value indefinitely
  - Read out the stored value
- The computer memory consists of a large number of these basic circuits

Bytes

- A byte is a string of 8 bits
- A byte is represented in the computer by 8 copies of the basic bit storage circuit
- Examples:
  - 0000 0000
  - 0110 1001
  - 1110 0100
  - 1111 1111
- *We use spaces (not commas) to break it up into groups of 4 bits, to make it more readable*

Information capacity of a byte

- There are exactly $2^8 = 256$ distinct values that can be represented in a byte
  - 00000000, 00000001, 00000010, 00000011, $\cdots$, 11111111
- There are many different ways to utilise the information capacity of a byte
- Each basic datatype (integer, character, etc.) has its own representation method

Words

- A word is similar to a byte, but a larger amount of information:

Short word	16 bits	2 bytes
Word	32 bits	4 bytes
Long word	64 bits	8 bytes

- The term "64-bit architecture" means that the internal hardware uses (mostly) 64-bit words
- Generally, a larger word size yields faster performance

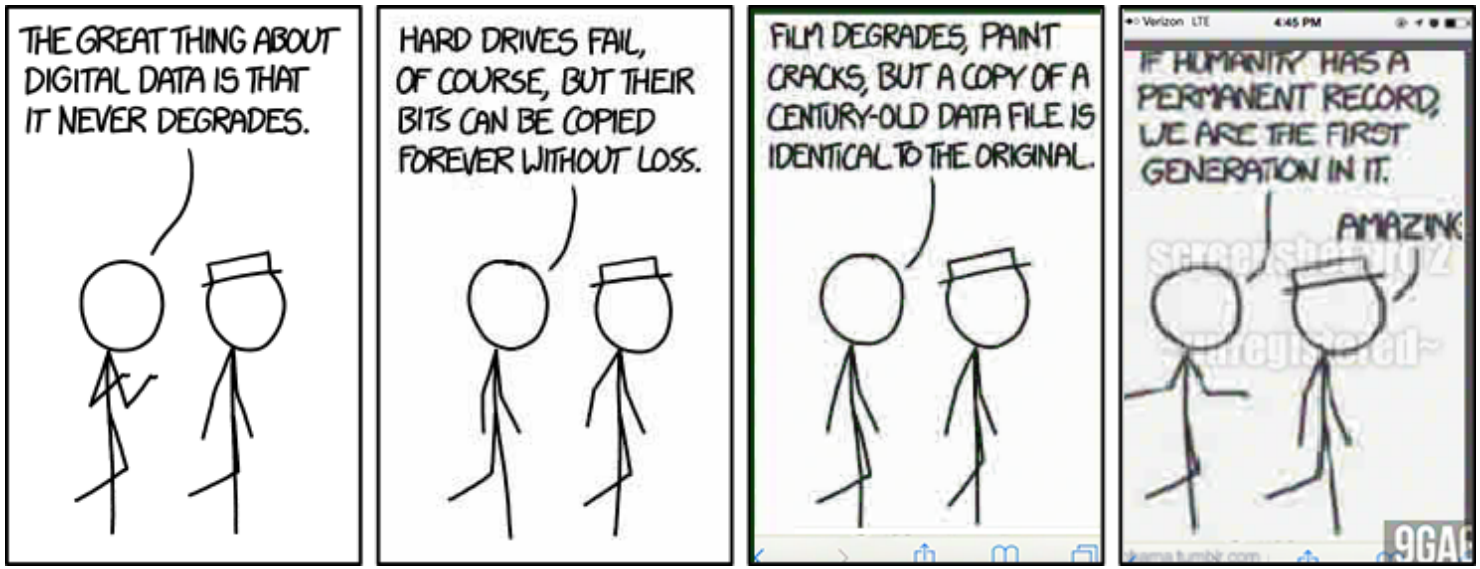
Information capacity of a word

A word containing k bits is called a k-bit word, and can represent 2^k distinct values.

Word size	Number of values	Approximately
8	$2^8 = 256$	10^2
16	$2^{16} = 65,536$	10^5
32	$2^{32} = 4,294,967,296$	10^9
64	$2^{64} =$ gigantic huge number	10^{19}

To do

- Check the Moodle page for this course
- Tutorial/lab starting next week, know which group you're in
- Reread the slides, see if you have any questions



https://xkcd.com/1683/