Today:

Everything is numbers ⟶ Everything is bits

1. Representing numbers as bits
2. Representing information as numbers
   more detail

Important ideas

- **number of items and number of digits are tightly related:**
  - one determines the other
  - maximum number of different items = base $^{\text{number of digits}}$
  - e.g., 9-digit SSN: 10^9 = 1 billion possible numbers
  - e.g., to represent up to 100 "characters": 2 digits is enough
  - but for 1000 characters, we need 3 digits
- **interpretation depends on context**
  - without knowing that, we can only guess what things mean
  - what's 81615 ?

Review: What's a bit? What's a byte?

- **a bit is the smallest unit of information**
- **represents one 2-way decision or a choice out of two possibilities**
  - yes / no, true / false, on / off, M / F, ...
- **abstraction of all of these is represented as 0 or 1**
  - enough to tell which of TWO possibilities has been chosen
  - a single digit with one of two values
  - hence "binary digit"
  - hence bit
- **binary is used in computers because it's easy to make fast, reliable, small devices that have only two states**
  - high voltage/low voltage, current flowing/not flowing (chips)
  - electrical charge present/not present (RAM, flash)
  - magnetized this way or that (disks)
  - light bounces off/doesn't bounce off (cd-rom, dvd)
- **all information in a computer is stored and processed as bits**
- **a byte is 8 bits that are treated as a unit**

Why binary, from von Neumann's paper:

5.2. In a discussion of the arithmetical organs of a computing machine one is naturally led to a consideration of the number system to be adopted. In spite of the longstanding tradition of building digital machines in the decimal system, we feel strongly in favor of the binary system for our device. Our fundamental unit of memory is naturally adapted to the binary system since we do not attempt to measure gradations of charge at a particular point in the Selectron but are content to distinguish two states.

The flip-flop again is truly a binary device. On magnetic wires or tapes and in acoustic delay line memories one is also content to recognize the presence or absence of a pulse or (if a carrier frequency is used) of a pulse train, or of the sign of a pulse. (We will not discuss here the ternary possibilities of a positive-or-negative-or-no-pulse system and their relationship to questions of reliability and checking,

A review of how decimal numbers work

- **how many digits?**
  - we use 10 digits for counting: "decimal" numbers are natural for us
  - other schemes show up in some areas
    clocks use 12, 24, 60; calendars use 7, 12
    other cultures use other schemes (quatre-vingts)
- **what if we want to count to more than 10?**
  - 0 1 2 3 4 5 6 7 8 9
    1 decimal digit represents 1 choice from 10; counts 10 things; 10 distinct values
  - 00 01 02 … 10 11 12 … 20 21 22 … 98 99
    2 decimal digits represents 1 choice from 100; 100 distinct values
    we usually elide zeros at the front
  - 000 001 … 099 100 101 … 998 999
    3 decimal digits …
- **decimal numbers are shorthands for sums of powers of 10**
  - 1492 = 1 x 1000 + 4 x 100 + 9 x 10 + 2 x 1
  - = $1 \times 10^3 + 4 \times 10^2 + 9 \times 10^1 + 2 \times 10^0$
- **counting in "base 10", using powers of 10**

Binary numbers: using bits to represent numbers

- **just like decimal except there are only <u>two</u> digits: 0 and 1**
- **everything is based on powers of 2 (1, 2, 4, 8, 16, 32, …)**
  - instead of powers of 10 (1, 10, 100, 1000, …)
- **counting in binary or base 2:**
  0 1
  1 binary digit represents 1 choice from 2; counts 2 things; 2 distinct values
  00 01 10 11
  2 binary digits represents 1 choice from 4; 4 distinct values
  000 001 010 011 100 101 110 111
  3 binary digits …
- **binary numbers are shorthands for sums of powers of 2**
  11011 = 1 x 16 + 1 x 8 + 0 x 4 + 1 x 2 + 1 x 1
  = $1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$
- **counting in "base 2", using powers of 2**

Using bits to represent information

- **M/F or on/off**
- **Fr/So/Jr/Sr**
- **add grads, auditors, faculty**
- **a number for each student in 109**
- **a number for each freshman at PU**
- **a number for each undergrad at PU**

Number of items represent with n bits?
Largest magnitude represent with n bits?

13th century wine units

2 gills = 1 chopin
2 chopins = 1 pint
2 pints = 1 quart
2 quarts = 1 pottle
2 pottles = 1 gallon
2 gallons = 1 peck
2 pecks = 1 demibushel
2 demibushels = 1 bushel or firkin
2 firkins = 1 kilderkin
2 kilderkins = 1 barrel
2 barrels = 1 hogshead
2 hogspeads = 1 pipe
2 pipes = 1 tun

– from D E Knuth, *The Art of Computer Programming, v 2*

Binary (base 2) arithmetic

- **works like decimal (base 10) arithmetic, but simpler**
- **addition:**

  0 + 0 = 0
  0 + 1 = 1
  1 + 0 = 1
  1 + 1 = 10

- **subtraction, multiplication, division are analogous**

Bytes

- **"byte" = group of 8 bits**
  - on modern machines, the fundamental unit of processing and memory addressing
  - can encode any of 2^8 = 256 different values, e.g:
    - numbers 0 .. 255 or
    - a single letter like A or digit like 7 or punctuation like $
  - ASCII character set defines values for letters, digits, punctuation, etc.
- **group 2 bytes together to hold larger entities**
  - two bytes (16 bits) holds 2^{16} = 65536 values
  - a bigger integer, a character in a larger character set
    Unicode character set defines values for almost all characters anywhere

Bytes cont.

- **group 4 bytes together to hold even larger entities**
  - four bytes (32 bits) holds 2^{32} = 4,294,967,296 values
    - an even bigger integer,
    - a number with a fractional part (floating point),
    - a memory address
- **etc.**
  - recent machines use 64-bit integers and addresses (8 bytes)
    2^{64} = 18,446,744,073,709,551,616

Interpretation of bits depends on context

- **meaning of a group of bits depends on how they are interpreted**
- **1 byte could be**
  - 1 bit in use, 7 wasted bits (e.g., M/F in a database)
  - 8 bits storing a number between 0 and 255
  - an alphabetic character like W or + or 7
  - part of a character in another alphabet or writing system (2 bytes)
  - part of a larger number (2 or 4 or 8 bytes, usually)
  - part of a picture or sound
  - part of an instruction for a computer to execute
    - instructions are just bits, stored in the same memory as data
    - different kinds of computers use different bit patterns for their instructions
      laptop, cellphone, game machine, etc., all potentially different
  - part of the location or address of something in memory
  - ...
- **one program's instructions are another program's data**
  - when you download a new program from the net, it's data
  - when you run it, it's instructions

Powers of two, powers of ten

1 bit = 2 possibilities
2 bits = 4 possibilities
3 bits = 8 possibilities
...
n bits = 2^n

2^{10} = 1,024 is about 1,000 or 1K or 10^3
2^{20} = 1,048,576 is about 1,000,000 or 1M or 10^6
2^{30} = 1,073,741,824 is about 1,000,000,000 or 1G or 10^9
the approximation is becoming less good
but it's still good enough for estimation

- **terminology is often imprecise:**
  - " 1K " might mean 1000 or 1024 (10^3 or 2^{10})
  - " 1M " might mean 1000000 or 1048576 (10^6 or 2^{20})

Converting between binary and decimal (version 1)

- **binary to decimal:**
  $1101 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$
  $= 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1$
  $= 13$
- **decimal to binary:**
  - start with largest power of 2 smaller than the number
  - for each power of 2 down to 2^0
  - if you can subtract that power of 2, do so and write "1"
  - otherwise write "0"

  - start with 13, subtract 8, write "1"
  - with 5, subtract 4, write "1"
  - with 1, can't subtract 2, write "0"
  - with 1, subtract 1, write "1"
  - answer is 1101

Converting between binary and decimal (version 2)

- **decimal to binary (from right to left):**
  - repeat while the number is > 0:
  - divide the number by 2
  - write the remainder (0 or 1)
  - use the quotient as the number and repeat
  - answer is the resulting sequence in reverse (right to left) order

  - divide 13 by 2, write "1", number is 6
  - divide 6 by 2, write "0", number is 3
  - divide 3 by 2, write "1", number is 1
  - divide 1 by 2, write "1", number is 0
  - answer is 1101

Hexadecimal notation

- **binary numbers are bulky**
- **hexadecimal notation is a shorthand**
- **it combines 4 bits into a single digit, written in base 16**
  - a more compact representation of the same information
- **hex uses the symbols A B C D E F for the digits 10 .. 15**
  0 1 2 3 4 5 6 7 8 9 A B C D E F

0	0000	1	0001	2	0010	3	0011
4	0100	5	0101	6	0110	7	0111
8	1000	9	1001	A	1010	B	1011
C	1100	D	1101	E	1110	F	1111

Example:

- **10100110110110**