Bits, bytes, and representation of information

- digital representation means that everything is represented by numbers only

- the usual sequence:
  - something (sound, pictures, text, instructions, ...) is converted into numbers by some mechanism
  - the numbers can be stored, retrieved, processed, transmitted
  - the numbers might be reconstituted into a version of the original

- for sound, pictures, other real-world values
  - make accurate measurements
  - convert them to numeric values

Analog versus Digital

- analog: "analogous" or "the analog of"
  - smoothly or continuously varying values
  - volume control, dimmer, faucet, steering wheel
  - value varies smoothly with something else
    - no discrete steps or changes in values
    - small change in one implies small change in another
    - infinite number of possible values
  - the world we perceive is largely analog

- digital: discrete values
  - only a finite number of different values
  - a change in something results in sudden change from one discrete value to another
    - digital speedometer, digital watch, push-button radio tuner, ...
  - values are represented as numbers
Transducers

- devices that convert from one representation to another
  - microphone
  - loudspeaker / earphones
  - camera / scanner
  - printer / screen
  - keyboard
  - mouse
  - touch screen
  - etc.

- something is usually lost by conversion (in each direction)
  - the ultimate copy is not as good as the original

Encoding sound

- need to measure intensity/loudness often enough and accurately enough that we can reconstruct it well enough
- higher frequency = higher pitch
- human ear can hear ~ 20 Hz to 20 KHz
  - taking samples at twice the highest frequency is good enough (Nyquist)

- CD audio usually uses
  - 44,100 samples / second
  - accuracy of 1 in 65,536 (= 2^16) distinct levels
  - two samples at each time for stereo
  - data rate is 44,100 x 2 x 16 bits/sample
    = 1,411,200 bits/sec = 176,400 bytes/sec ~ 10.6 MB/minute

- MP3 audio compresses by clever encoding and removal of sounds that won't really be heard
  - data rate is ~ 1 MB/minute
Discrete values vs continuous values

- another kind of conversion
  - letters are converted into numbers when you type on a keyboard
  - the letters are stored (a Word document), retrieved (File/Open...), processed (paper is revised), transmitted (submitted by email)
  - printed on paper

- letters and other symbols are inherently discrete

- encoding them as numbers is just assigning a numeric value to each one, without any intrinsic meaning

Important ideas

- number of items and number of digits are tightly related:
  - one determines the other
  - maximum number of different items = base 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?
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.

Using bits to represent information

- M/F or on/off
  - 1 bit
- Fr/So/Jr/Sr
  - 2 bits
- Add grads, auditors, faculty
  - 3 bits
- A number for each student in 109
  - 7 bits
- A number for each freshman at PU
  - 11 bits
- A number for each undergrad at PU
  - 13 bits
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 \times 1000 + 4 \times 100 + 9 \times 10 + 2 \times 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 two 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 \times 16 + 1 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 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
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 $$
    \text{ASCII character set defines values for letters, digits, punctuation, etc.}
  $$

- 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

- 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

<table>
<thead>
<tr>
<th>0</th>
<th>0000</th>
<th>1</th>
<th>0001</th>
<th>2</th>
<th>0010</th>
<th>3</th>
<th>0011</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0100</td>
<td>5</td>
<td>0101</td>
<td>6</td>
<td>0110</td>
<td>7</td>
<td>0111</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>9</td>
<td>1001</td>
<td>A</td>
<td>1010</td>
<td>B</td>
<td>1011</td>
</tr>
<tr>
<td>C</td>
<td>1100</td>
<td>D</td>
<td>1101</td>
<td>E</td>
<td>1110</td>
<td>F</td>
<td>1111</td>
</tr>
</tbody>
</table>

ASCII (better representation: uses hex)
Color

- TV & computer screens use Red-Green-Blue (RGB) model
  
  - each color is a combination of red, green, blue components
    - R+G = yellow, R+B = magenta, B+G = cyan, R+G+B = white
  
- for computers, color of a pixel is usually specified by three numbers giving amount of each color, on a scale of 0 to 255
  
  - this is often expressed in hexadecimal so the three components can be specified separately (in effect, as bit patterns)
    - 000000 is black, FFFFFF is white
  
- printers, etc., use cyan-magenta-yellow[black] (CMYK)

Things to remember

- digital devices represent everything as numbers
  - discrete values, not continuous or infinitely precise
- all modern digital devices use binary numbers (base 2)
  - instead of decimal (base 10)
- it's all bits at the bottom
  - a bit is a "binary digit", that is, a number that is either 0 or 1
  - computers ultimately represent and process everything as bits
- groups of bits represent larger things
  - numbers, letters, words, names, pictures, sounds, instructions, ...
  - the interpretation of a group of bits depends on their context
  - the representation is arbitrary: standards (often) define what it is
- the number of digits used in the representation determines how many different things can be represented
  - number of values = base number of digits
  - e.g., \(10^2\), \(2^{10}\)