Bits, Bytes, and Representation of Information

- **digital versus analog**
  - computers use discrete, discontinuous values, not smoothly varying
- **binary versus decimal**
  - computers 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 <u>everything</u> 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
- **these are basic ideas that underlie all computing**

Analog versus Digital

- **analog: "analogous" or "the analog of"**
  - smoothly or continuously varying values
  - like a volume control, a tap, a car steering wheel or brakes
    - louder/softer on phone, light dimmer, ...
  - 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 in a car, digital watch, push-button radio tuner, ...
  - values are represented as <u>numbers</u>

Binary versus Decimal

- **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 others (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

- **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 ...

Converting between binary and decimal

- **binary to decimal:**

  1101 = 1 x 2^3 + 1 x 2^2 + 0 x 2^1 + 1 x 2^0
  = 1 x 8 + 1 x 4 + 0 x 2 + 1 x 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"

Bits

- **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)
  - magnetized this way or that (disks)
  - light bounces off/doesn't bounce off (cd-rom, dvd)
  - ...
- **"bit" = "binary digit" (coined by John Tukey *38)**
  - only two values, 0 and 1
  - compare to decimal digits, with values 0 1 2 3 4 5 6 7 8 9
- **1 bit represents one decision or choice from two possibilities**
  - yes / no
  - true / false
  - on / off
  - M / F

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
- **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.**
  - newest machines are starting to use 64-bit integers

Hexadecimal notation

- **binary numbers are bulky**
- **hexadecimal notation 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**

Binary (base 2) arithmetic

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

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

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

Interpretation of bits depends on context

- **the meaning of a group of bits depends on the context in which they are being interpreted**
- **1 byte could be**
  - 1 bit in use, 7 wasted bits (e.g., M/F in a database)
  - 8 bits storing a small 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 an instruction for the CPU to execute
    - instructions are just bits, stored in the same memory as data
    - different kinds of CPUs use different bit patterns for their instructions
      - Pentium, PowerPC, Palm, game machines, etc., all potentially different
- **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

- **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**
  - notice that the approximation is becoming less good

- **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})

Other kinds of computers

- **not all computers are PCs (or Macs)**
- **multiple CPUs with shared memory (symmetric multiprocessor)**
  - e.g., Phoenix, Yuma
- **"supercomputers"**
  - specialized to do some kind of numeric computation very fast
- **"distributed" computers**
  - loosely coupled; memory is not shared
- **embedded computers**
  - appliances, cell phones, games, PDAs, prox card

- **each represents some set of tradeoffs among cost, computing power, size, speed, ...**

Important Hardware Ideas

- **programmable computer: a single general-purpose machine can be programmed to do an infinite variety of tasks**
- **simple instructions that do arithmetic, compare items, select next instruction based on results**
- **all machines have the same logical capabilities (Turing)**
  - architecture is mostly unchanged since 1940's (von Neumann)
  - physical properties evolve rapidly
- **bits at the bottom**
  - everything ultimately reduced to representation in bits (binary numbers)
  - groups of bits represent larger entities: numbers of various sizes, letters in various character sets, instructions, memory addresses
  - interpretation of bits depends on context
    one person's instructions are another's data
- **there are many things that we do not know how to represent as bits, nor how to process by computer**