cc3.12/cis1.0
computing: nature, power and limits—robotics applications
fall 2007
lecture # III.1

data representation and storage

- data representation
- file storage
- speed of data transmission

resources:

- reading: Reed chapter 12 and 14

data representation: bits

- a **bit** is the smallest unit of memory
- **bit** $=$ **b**inary dig**it**
- a bit is a *switch* inside the computer; the setting (or value) of each switch is either ON $(=1)$ or OFF (-0)
- all data in a computer is represented by *bit patterns*, i.e., sequences of 0's and 1's
- all numbers can be represented by 0's and 1's in *base 2*
- hence the term *binary* computer!

data representation: bytes

- a **byte** is a sequence of 8 bits
- thus there are $2^8 = 256$ possible values that can be represented by one byte
- values range from 0 to $2^8 - 1 = 256 - 1 = 255$
- where $0 = 00000000$ and $255 = 11111111$

data representation: base 2

- in base 2, only the digits 0 and 1 are used
- just like base 10, each digit, from the right to the left, indicates how many of each base raised to a power are contained in the number that is represented...

  probably an example will help:

  – note that digits are counted from right to left, starting with 0

  – to convert a byte to base 10, multiply each digit in the byte by the value in the table below, then add them all together

digit:	7	6	5	4	3	2	1	0
power:	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
value:	128	64	32	16	8	4	2	1

  – so, to convert 00001011, look up each digit in the table above, and you get this:

digit:	7	6	5	4	3	2	1	0
power:	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
value:	128	64	32	16	8	4	2	1
× byte:	0	0	0	0	1	0	1	1
=	0×128	0×64	0×32	0×16	1×8	0×4	1×2	1×1
=	0	+ 0	+ 0	+ 0	+ 8	+ 0	+ 2	+ 1
=	11 (in base 10)							

data representation: other bases

- base 8, or *octal*—
  since $2^3 = 8$, it is often convenient to compress 3-digit binary values as base 8, or octal values
- base 16, or *hexadecimal*—
  since $2^4 = 16$, it is often convenient to compress 4-digit binary values as base 16, or hexadecimal values
- it is handy to memorize (or at least, to know how to derive) the following table of base 2, 8 and 16 numbers from 0 to 15:

base 10	base 2	base 8	base 16
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

data representation: exercise

- exercise—
  convert 13_{base8} to base 10:
  13_8
  $= (1 \times 8^1) + (3 \times 8^0)$
  $= (1 \times 8) + (3 \times 1)$
  $= 8 + 3$
  $= 11$

data storage: storing numbers in a computer

- now you know the basis for how numbers are stored
- but all numbers are not just values between 0 and 255
- some are very large, some are real (i.e., have decimal points), some are negative
- negative numbers are represented using something called **two's complement notation** in which the leftmost bit is a **sign bit** and some operations are performed on the digits in order to determine the value of the negative number;
  we will not go into this level of detail...
- real numbers are represented using something called **floating point notation** in which the whole and fractional parts of the number are stored separately and some operations are performed on the digits in order to put the pieces together and determine the value of the real number;
  we will not go into this level of detail...

data storage: storing letters in a computer

- letters, or *characters*, are stored as numbers, but are encoded so that for each character on the keyboard (or displayed on the screen) there is a (positive) number that represents that character
- the software reading the value has to know that it should be interpreted as a character rather than a number
- the standard encoding is called **ASCII** (American Standard Code for Information Interchange)
- standard ASCII encodes 128 characters
- extended ASCII encodes 128 more, to total 256 characters
- **Unicode** uses 2 bytes and encodes $2^{16} = 65536$ characters (!) in many languages

  "Unicode provides a unique number for every character,
  no matter what the platform,
  no matter what the program,
  no matter what the language."
  (from `http://www.unicode.org`)

- for more information, go to `http://www.unicode.org`
- NOTE that the digits 0, 1, 2, ..., 9 can be stored as characters and have entries in the ASCII and Unicode tables

file storage

- files can be stored on a computer as either **plain text** or **binary**
- here is the simplest way to tell which a file's type is:
  - on Windows:
    go to the DOS prompt. enter:
    `dos-prompt: type <filename>`
    where you substitute <filename> with the name of the file you want to check
  - on Mac or UNIX:
    go to the terminal prompt. enter:
    `unix-prompt: more <filename>`
    where you substitute <filename> with the name of the file you want to check
  - on either operating system, text files will be displayed as letters and numbers that you can read and should look like what you expect;
    binary files will look like garbage characters and might mess up your terminal window (in which case, just type `reset` and it will fix itself)
- HTML files are plain text files
- most image files are binary files

- some files are stored as plain text, but their content is encoded so that you need special software to read the files
- usually, plain text files take up less space than binary files
- exercise—
  go into Word and create a new document. put the text "hello world" into the document and save it (as a word document).
  then go into NotePad (or TextEdit on the Mac) and create a new document. put the same text ("hello world") into the document and save it (as plain text).
  now look at both files in the explorer (or finder on the Mac) and compare their file sizes. which is larger? are you surprised?

file sizes

- file sizes are typically quoted in **bytes**, **kilobytes**, **megabytes** or **gigabytes**
- here are some handy conversions:
  (watch your BITS and BYTES...)

1 byte (B)	$=$	8 bits
1 kilobyte (KB)	$=$	1024 bytes
	$=$	2^{10} bytes
1 megabyte (MB)	$=$	1024 KB
	$=$	1024×1024 bytes
	$=$	$2^{10} \times 2^{10}$ bytes
	$=$	2^{20} bytes
1 gigabyte (GB)	$=$	1024 MB
	$=$	$1024 \times 1024 \times 1024$ bytes
	$=$	$2^{10} \times 2^{10} \times 2^{10}$ bytes
	$=$	2^{30} bytes

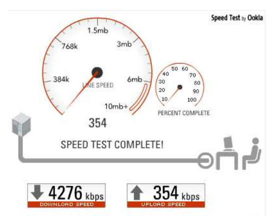
that's a lot of bytes...
today's computers must be VERY hungry ... ;-)

bandwidth

- bandwidth $=$ speed of data transmission
- data is transmitted at speeds that are measured in terms of **kilobits per second (kbits/s)**
  (1 kilobit $= 1000$ bits $= 10^3$ bits ≈ 1024 bits $= 2^{10}$ bits $= 2^7$ bytes...)
- the time it takes to **download** a file (copy it from one computer to another) depends on the size of the file, the speed of the source computer (e.g., a server), the speed of the network and the speed of the destination computer (e.g., your laptop or desktop)
- there are different ways to connect to the internet—
  - dial-up (modem)
    typically 28.8K (kilobits/sec) or $28,000$ bits per second or 56K ($56,000$ bps)
  - DSL $=$ "Digital Subscriber Line"
    part of ISDN (Integrated Services Digital Network); allows data transmission over regular telephone lines
    typically ranges from 128 K (kilobits/sec) to $24,000$ K

  - cable modem
    carries data transmissions over digital cable television lines
    typically ranges from 384 K (kilobits/sec) to 30 M (megabits/sec) for a fast business line

- exercise—
  take the **speakeasy** speed test:



`http://www.speakeasy.net/speedtest/`