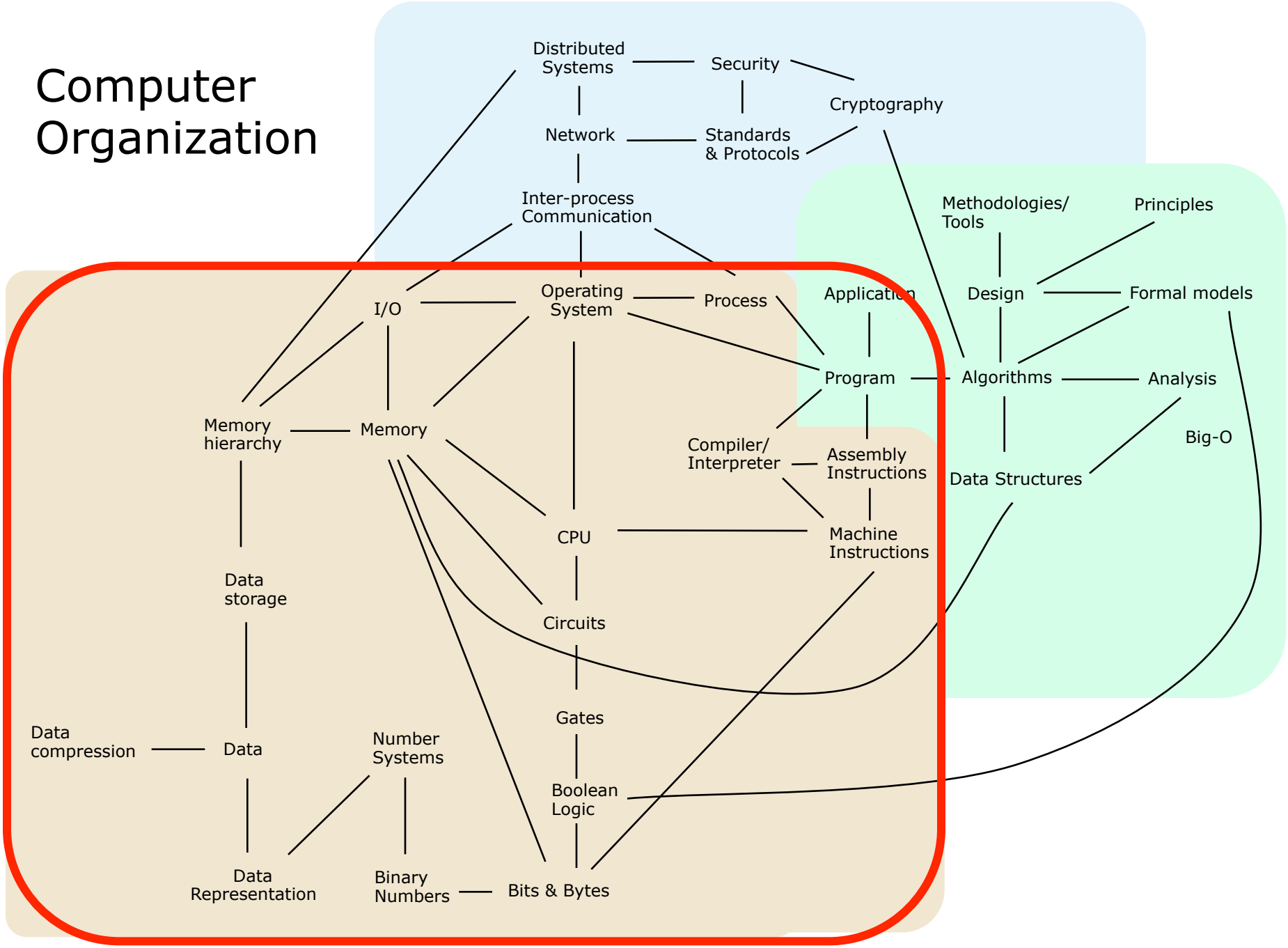
i206: Lecture 2:
Computer Architecture, Binary Encodings, and Data Representation

Tapan Parikh

Spring 2013

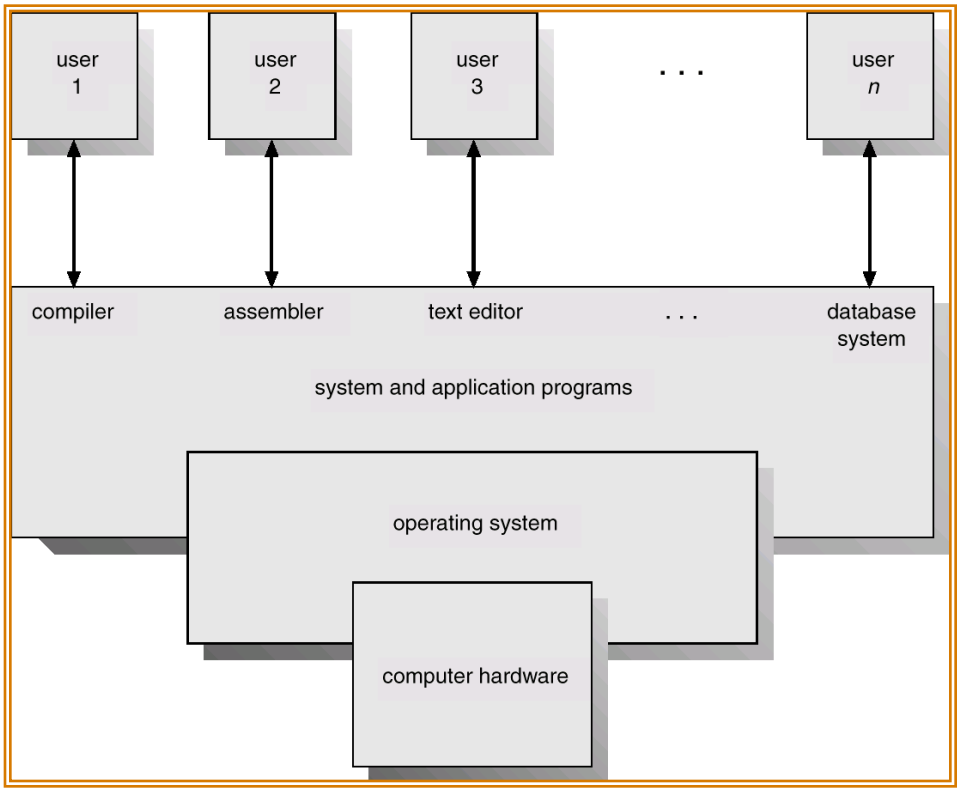
Some slides courtesy Marti Hearst, John Chuang and others

Computer Organization



Computers of Different Shapes

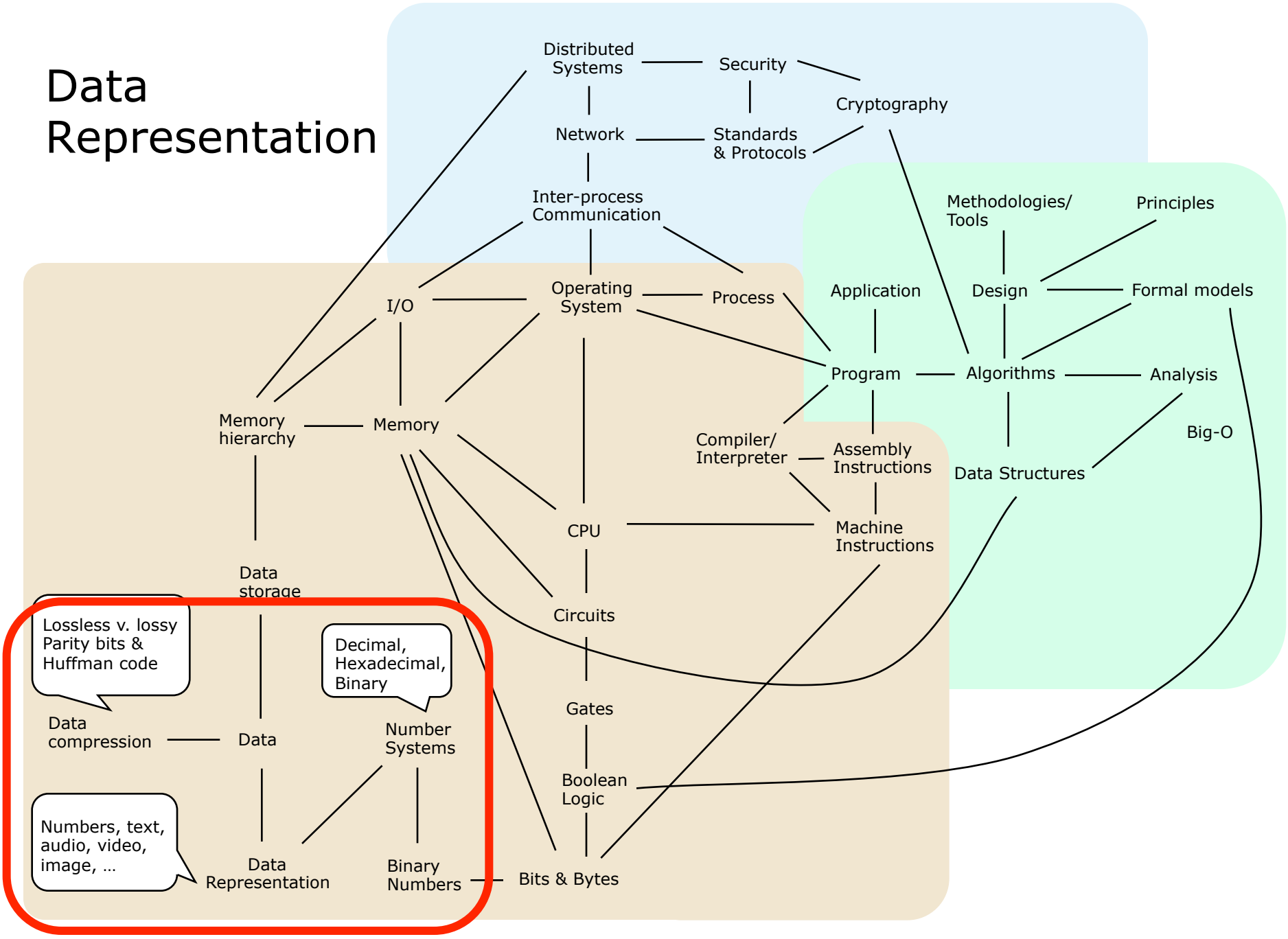
Computer System Components



Source: Silberschatz, Galvin, Gagne

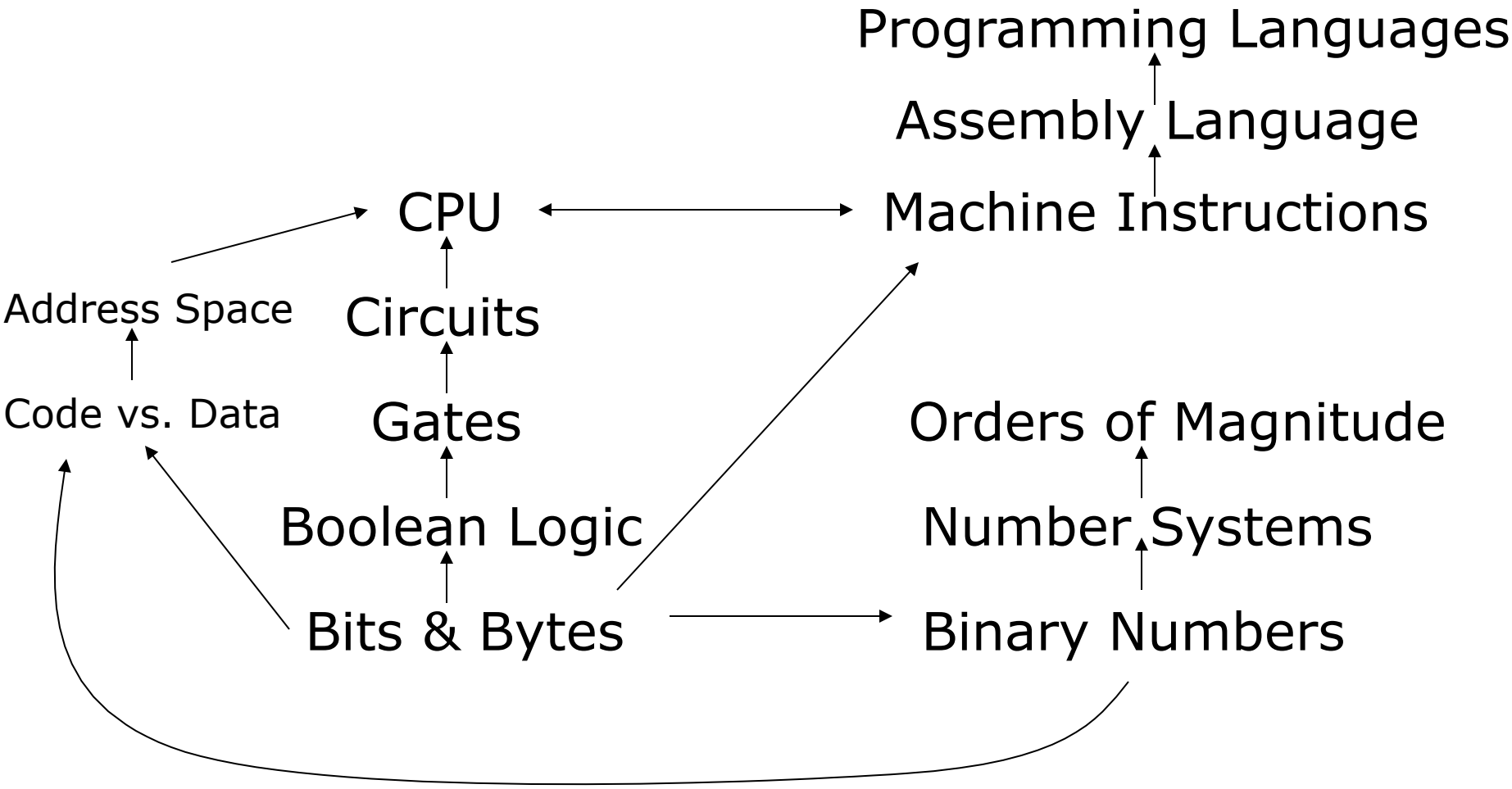
1. Hardware – provides basic computing resources (CPU, memory, I/O devices, network)
2. Operating system – controls and coordinates the use of the hardware among the various application programs for the various users
3. Application programs – define the ways in which the system resources are used to solve the computing problems of the users (compilers, database systems, video games, business programs)
4. Users (people, machines, other computers)

Data Representation



How Do Computers Work?

We are going bottom-up, with digressions



Data Representation

- All data stored in and manipulated by a *digital* computer are represented by patterns of bits:
  - Numbers
  - Text characters
  - Sound
  - Images and videos
  - Anything else…
- Bit = Binary Digit
- Binary: takes on values of ‘0’ or ‘1’
  - Or equivalently, “FALSE” or “TRUE”, “OFF” or “ON”

Bytes

- A sequence of bits
- 8 bits = 1 byte
- 2 bytes = 1 word (sometimes 4 or 8 bytes)
- How do binary numbers work?

Number Systems

$$DIGIT * BASE^{POSITION\ \#}$$

$$
\begin{array}{lcll}
4*10^{2} & = & 4*100 & = & 400. \\
6*10^{1} & = & 6*10 & = & 60. \\
2*10^{0} & = & 2*1 & = & 2. \\
1*10^{-1} & = & 1*.1 & = & 0.1 \\
5*10^{-2} & = & 5*.01 & = & +\ 0.05 \\
 & & & & \underline{462.15}
\end{array}
$$

Images from http://courses.cs.vt.edu/~csonline/MachineArchitecture/Lessons/Circuits/index.html

Binary Number System

$$DIGIT * 2^{POSITION\ \#}$$

110

Fours	*Twos*	*Ones*
2^2	2^1	2^0
1	**1**	**0**

$$1*2^2 = 1*4 = 4$$
$$1*2^1 = 1*2 = 2$$
$$0*2^0 = 0*1 = 0$$

= 6 decimal

Images from http://courses.cs.vt.edu/~csonline/MachineArchitecture/Lessons/Circuits/index.html

Powers of Two

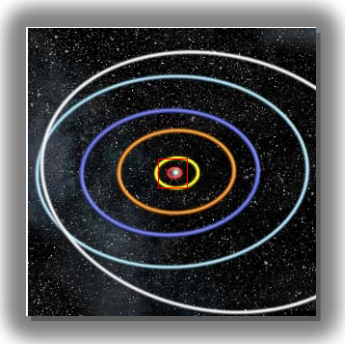
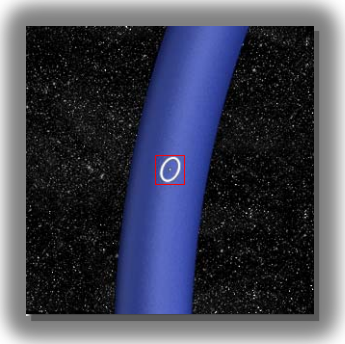
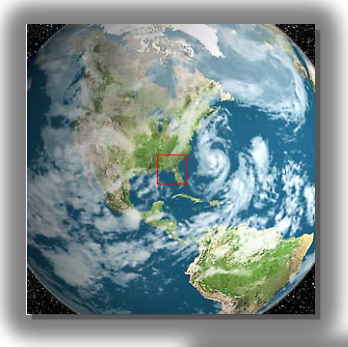
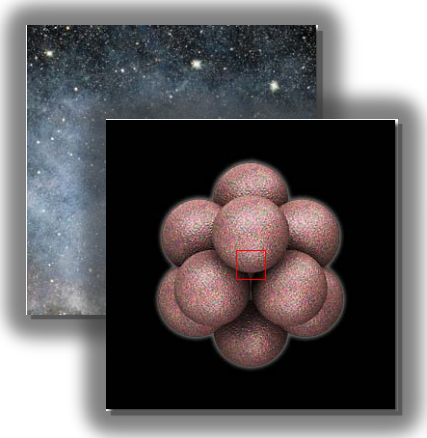
Decimal	Binary	Power of 2
1	1	2^0
2	10	2^1
4	100	2^2
8	1000	2^3
16	10000	2^4
32	100000	2^5
64	1000000	2^6
128	10000000	2^7

Famous Powers of Two

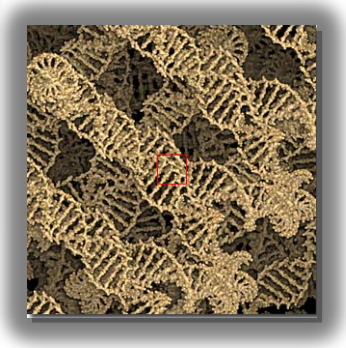
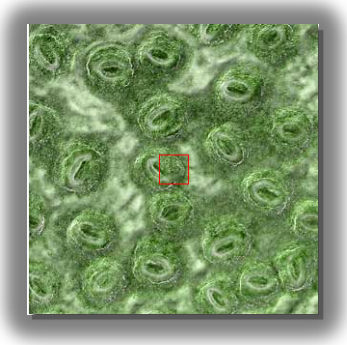
Kilobyte (KB)	1024 or 2^{10} bytes	1,024 bytes	Thousands of bytes
Megabyte (MB)	1024^2 or 2^{20} bytes	1,048,578 bytes	Millions of bytes
Gigabyte (GB)	1024^3 or 2^{30} bytes	1,073,741,824 bytes	Billions of bytes
Terabyte (TB)	1024^4 or 2^{40} bytes	1,099,511,627,776 bytes	Trillions of bytes

As Brookshear points out,
these are non-standard uses of K,M,G,B
but we're stuck with them.

Images from http://courses.cs.vt.edu/~csonline/MachineArchitecture/Lessons/Circuits/index.html


Orders of Magnitude

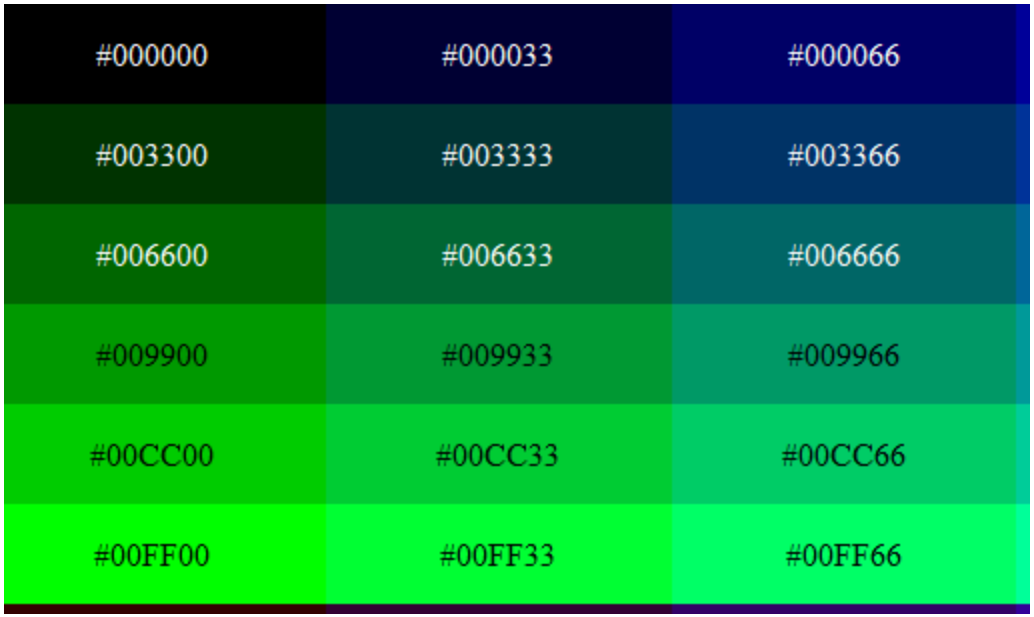

- Each increase in the power of 10 is an increase in the order of magnitude
- http://micro.magnet.fsu.edu/primer/java/scienceopticsu/powersof10/index.html

Other Number Systems

Binary	Octal	Decimal	Hexadecimal
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	10	8	8
1001	11	9	9
1010	12	10	A
1011	13	11	B
1100	14	12	C
1101	15	13	D
1110	16	14	E
1111	17	15	F
Base-2	*Base-8*	*Base-10*	*Base-16*

Images from http://courses.cs.vt.edu/~csonline/MachineArchitecture/Lessons/Circuits/index.html

Where do we see Hex on the Web?



Binary Addition

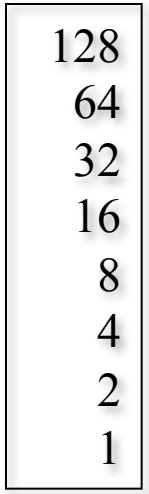
Rule 1	Rule 2	Rule 3	Rule 4
0 + 0 0	0 + 1 1	1 + 0 1	1 + 1 10

Also: 1 + 1 + 1 = 1 with a carry of 1

Images from http://courses.cs.vt.edu/~csonline/MachineArchitecture/Lessons/Circuits/index.html

Convert Decimal to Binary

$$13 = 8 + 4 + 1 = 1101$$



Images from http://courses.cs.vt.edu/~csonline/MachineArchitecture/Lessons/Circuits/index.html

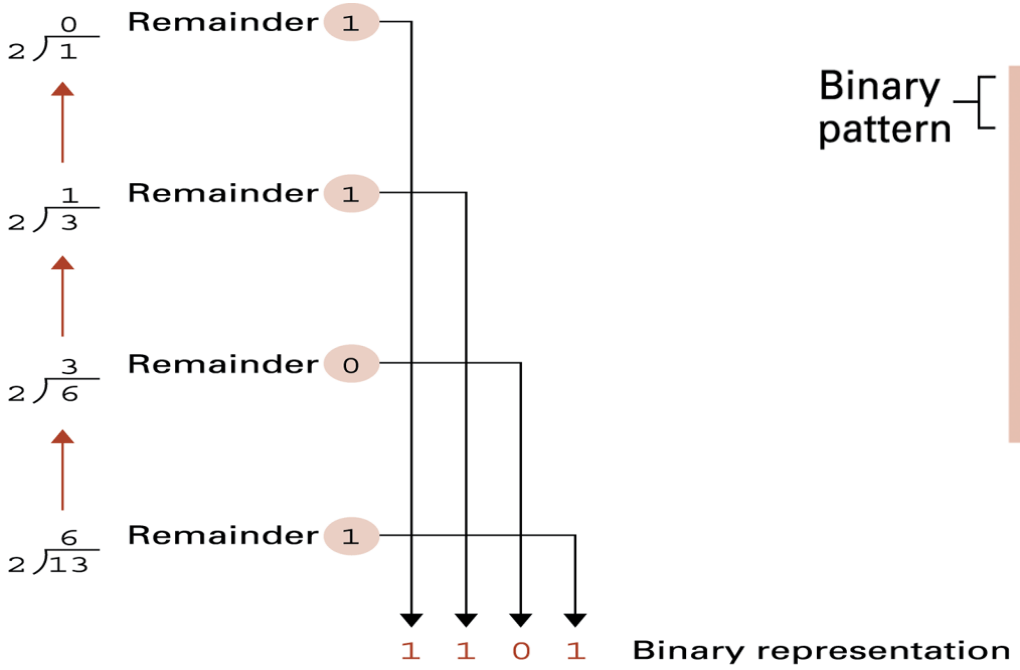
Decimal-Binary Conversion

Decimal to binary:

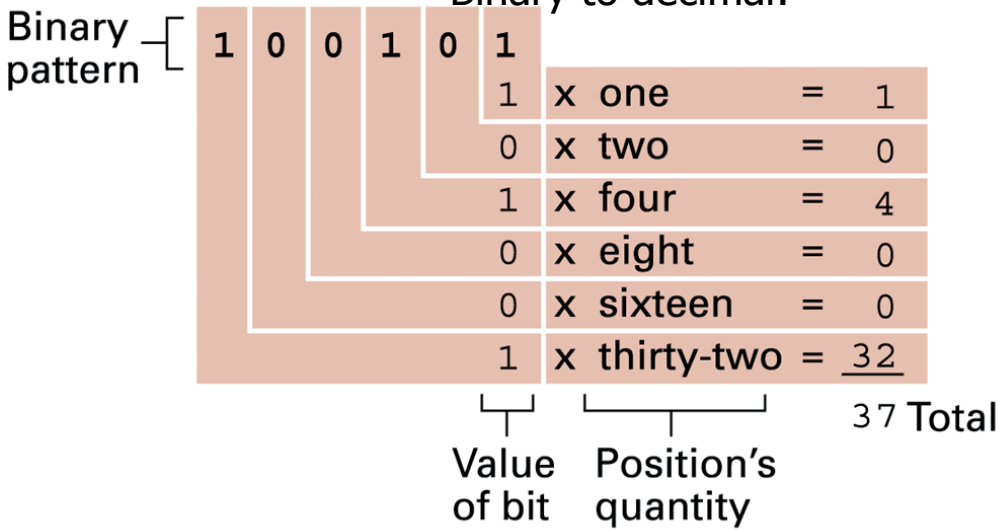
Step 1. Divide the value by two and record the remainder.

Step 2. As long as the quotient obtained is not zero, continue to divide the newest quotient by two and record the remainder.

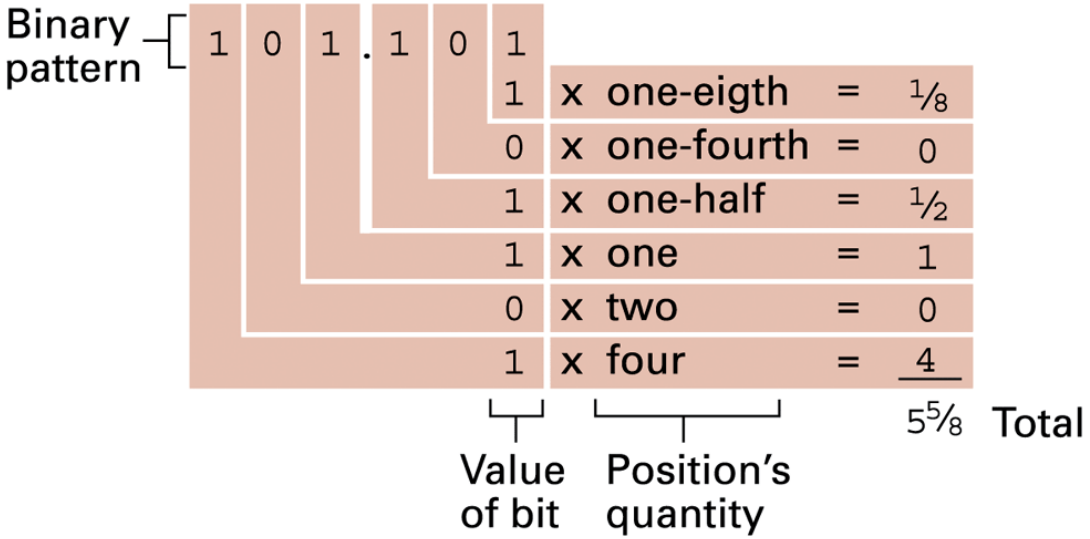
Step 3. Now that a quotient of zero has been obtained, the binary representation of the original value consists of the remainders listed from right to left in the order they were recorded.

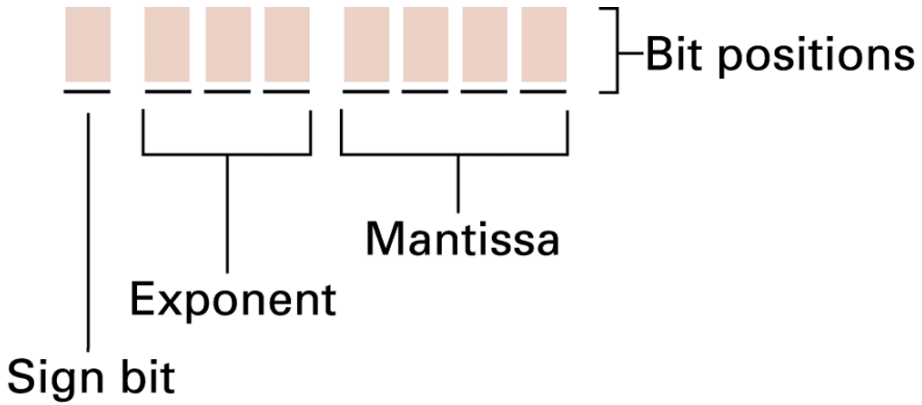


Binary to decimal:



Binary Representation of Fractions and Floating-Point Numbers





Representing text

- Each printable character (letter, punctuation, etc.) is assigned a unique bit pattern.
  - ASCII = 7-bit values for most symbols used in written English text (see Appendix A in Brookshear)
  - Unicode = 16-bit values for most symbols used in most world languages today
  - ISO proposed standard = 32-bit values

- Example: the message ‘Hello.’ in ASCII encoding

01001000	01100101	01101100	01101100	01101111	00101110
H	e	l	l	o	.

Write “Go Bears!” in hex or binary

(the code below is decimal)

128
64
32
16
8
4
2
1

Letter	ASCII Code
a	097
b	098
c	099
d	100
e	101
f	102
g	103
h	104
i	105
j	106
k	107
l	108

m	109
n	110
o	111
p	112
q	113
r	114
s	115
t	116
u	117
v	118
w	119
x	120
y	121
z	122

Letter	ASCII Code
A	065
B	066
C	067
D	068
E	069
F	070
G	071
H	072
I	073
J	074
K	075
L	076

M	077
N	078
O	079
P	080
Q	081
R	082
S	083
T	084
U	085
V	086
W	087
X	088
Y	089
Z	090

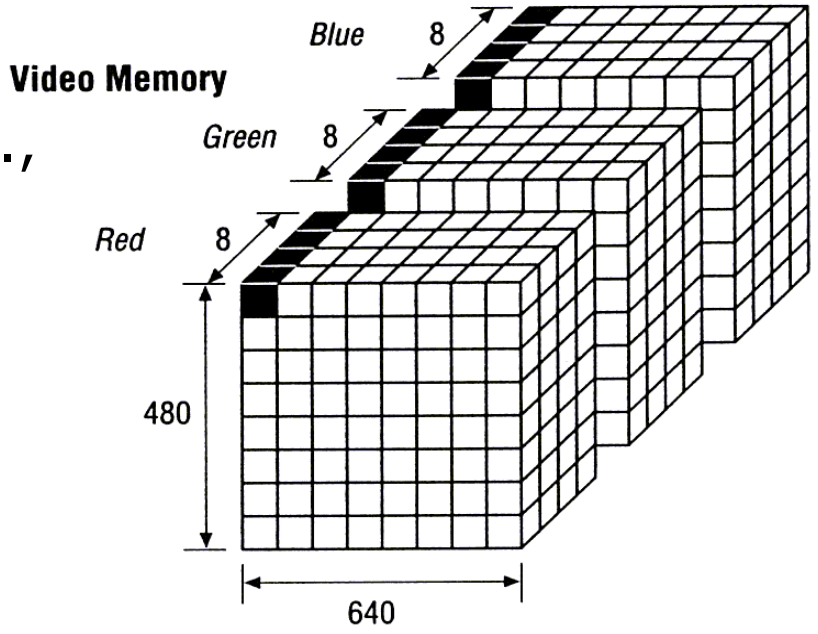
! is 33, space is 32

Write “Go Bears!” in binary / hex

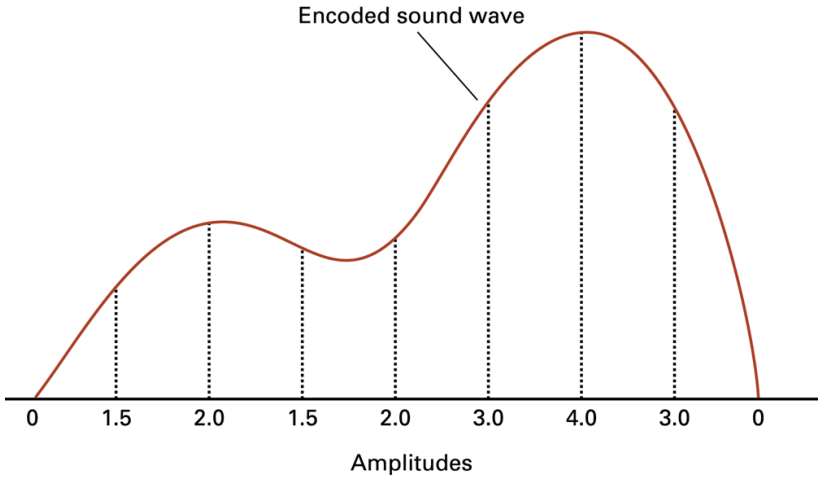
01000111	47
01101111	6F
00100000	20
01000010	42
01100101	65
01100001	61
01110010	72
01110011	73
00100001	21

Representation of Images and Video

- Image encoding:
  - Number of bits per pixel (e.g., 24 bits for RGB color)
  - Number of pixels in image (e.g., 480*640)
- Video encoding:
  - Number of bits per pixel
  - Number of pixels per frame
  - Number of frames per second



Audio Encoding



The sound wave represented by the sequence 0, 1.5, 2.0, 1.5, 2.0, 3.0, 4.0, 3.0, 0

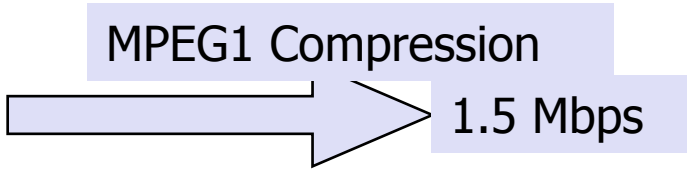
- Telephone:
  - 8,000 samples per second
  - 8 bits per sample
  - → 64 kbps
- CD-quality audio:
  - 44,100 samples per second
  - 16 bits per sample
  - 2 channels for stereo
  - → 1.4 Mbps
  - → 60 minute music CD takes ~630MB
- There are many audio compression standards (e.g., H.323 for Voice-over-IP, MP3 for music)

Data Compression

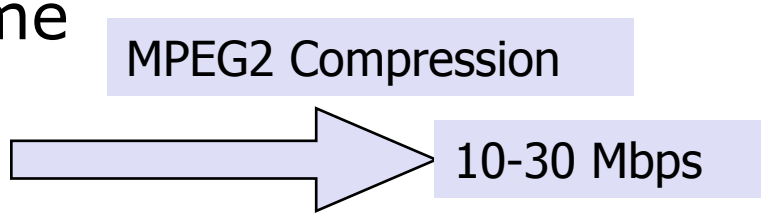
- Encoding data using fewer bits
- There are many encoding standards
  - (e.g., JPEG, MPEG, MP3) include data compression
- Lossless
  - data received = data sent
  - used for executables, text files, numeric data
- Lossy
  - data received != data sent
  - used for images, video, audio

Video Encoding Examples

- Uncompressed video:
  - 8 bits per pixel
  - 480*640 pixels per frame
  - 24 frames per second
  - → 56 Mbps



- Uncompressed HDTV:
  - 24 bits per pixel
  - 1920*1080 pixels per frame
  - 24-60 frames per second
  - → 1000-3000 Mbps



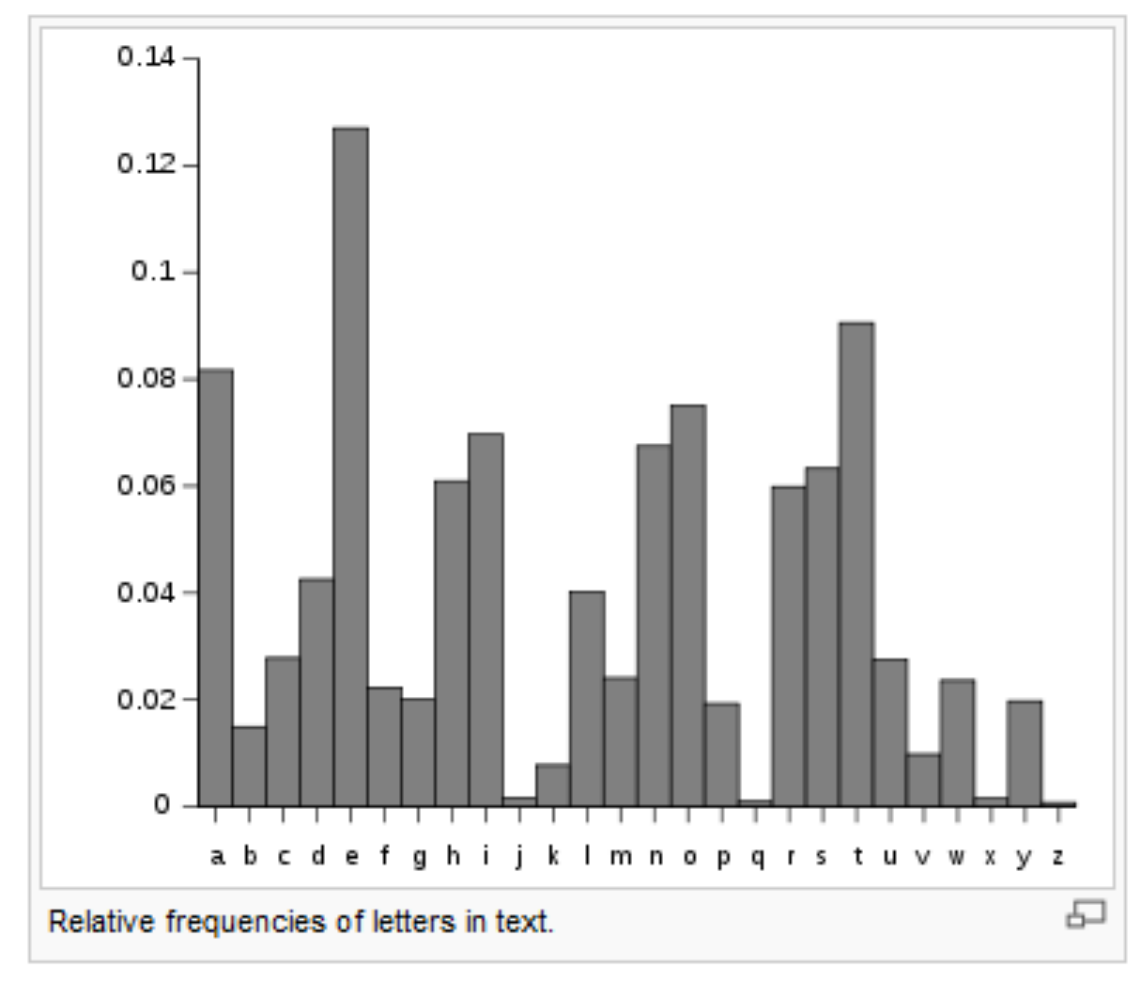
Data Compression: Run-length encoding

If there is a long sequence of repeated numbers, just say how many of each.

- 0000000000000100000000000
- 13 0's
- 1 1
- 11 0's
- How many bits long is this in binary?

Data Compression: Frequency-dependent encoding

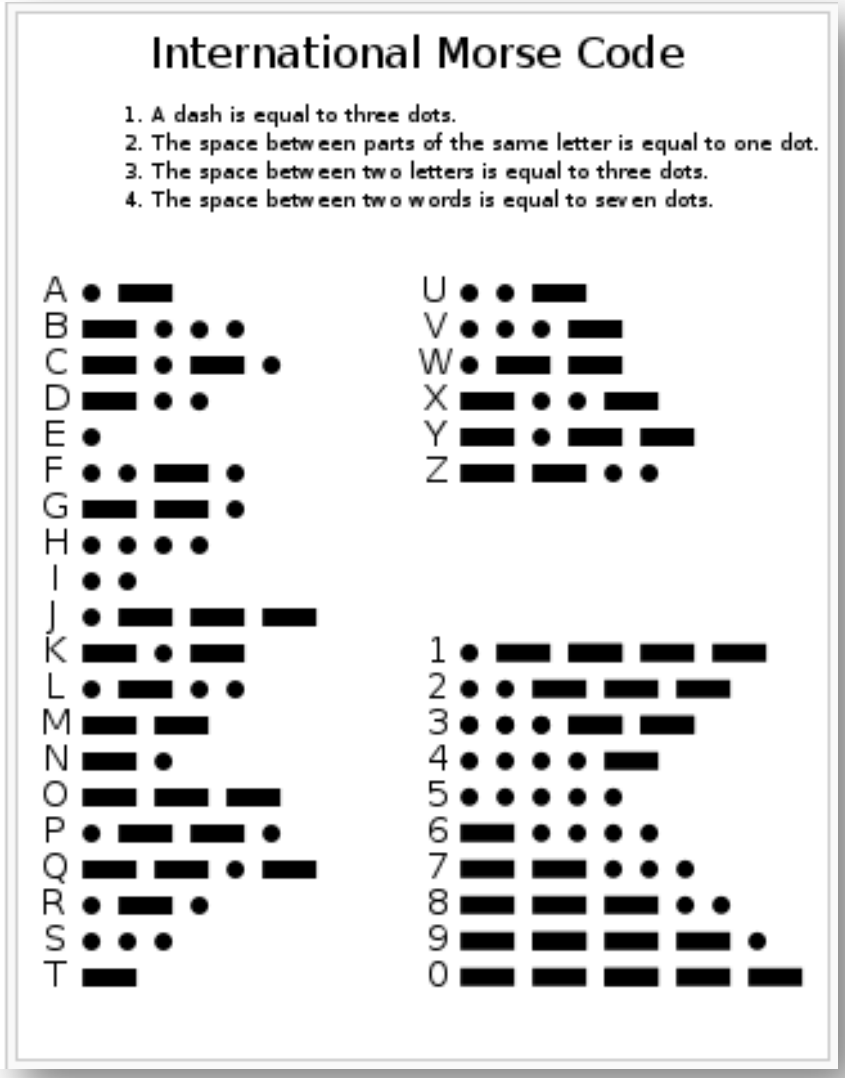
- The length of the representation of the bit pattern is inversely related to its frequency of occurrence.
- Which letters of English are most frequent?
- Give these shorter representations.



Relative frequencies of letters in text.

Wikipedia (via google cache)

Where else do we see this used?



Wikipedia (via google cache)

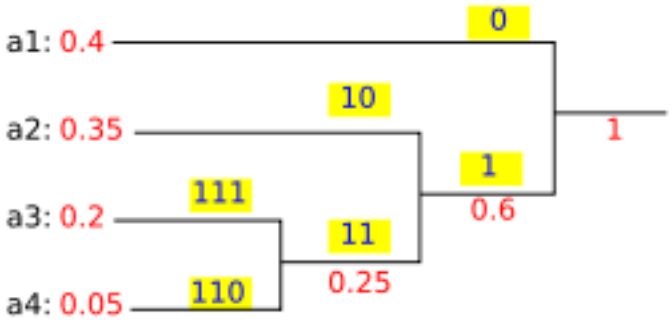
Data Compression
Frequency Dependent Encoding

- ASCII coding uses 7 bits to represent characters
- Can we devise a more intelligent encoding scheme given knowledge that some characters (e.g., “a”, “e”) occur more frequently than others (e.g., “q”, “z”)?
- Huffman code: variable length frequency dependent encoding
  - Intuition: assign shorter codes for characters that occur more frequently

http://en.wikipedia.org/wiki/Huffman_code

Huffman Coding

- Choose codes such that less likely messages have longer lengths.
- Given: messages with known weights (probabilities of occurrence)
  - Construct binary tree with each message as a leaf node
  - Iteratively merge two nodes with lowest weights
  - Assign bits (0/1) to each branch
  - Read binary tree from root to leaf to obtain code for each message



Message	Code
a1	0
a2	10
a3	111
a4	110

Error-Correction Codes

- Parity Bits: add an additional bit to the code
  - If the sum of the bits is even, add a 1 to make it odd
- What if there are two errors?
  - Checkbyte for long codes
  - Example: 1 bit for every 8^{th} bit in the code
- But this doesn't correct the error!
- How can we do this?

Error-Correction Codes

- Hamming Distance:
  - Number of bits in which the codes differ
- Assign codes so they all have a Hamming Difference of at least 3
- If a code differs by 1, you can tell which bit flipped.
- If a code differs by 2, you can tell there is an error of some kind.

Error-Correction Codes

Pattern: 010100

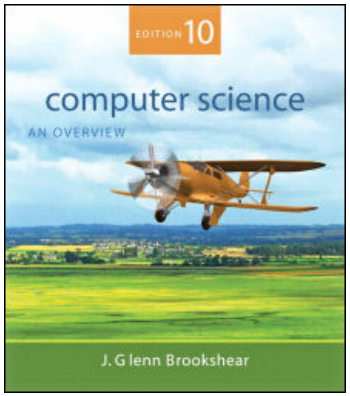
Symbol	Code
A	000000
B	001111
C	010011
D	011100
E	100110
F	101001
G	110101
H	111010

Character	Distance between the received pattern and the character being considered
A	2
B	4
C	3
D	1 *Smallest distance*
E	3
F	5
G	2
H	4

Summary

- Different types of data (e.g., numbers, text characters, sounds, images, videos) can all be represented using binary bits
- Compression techniques allow compact encoding of information
- There are many standards for encoding different data types
  - Compression often part of encoding standard
- Up next: how these binary represented data can be stored, communicated, and operated upon

Next



- Read: Brookshear 1.1, 2.1 – 2.4
- Read: Brookshear 6.1-6.5 (for a review of OO for lab)