Digital Data

Patrice Koehl
Computer Science
UC Davis

Digital Data

- Binary and Hexadecimal numbers
- ASCII code and UNICODE
- Sampling and Quantitizing
  - *Example: sound*

Digital Data

- Binary and Hexadecimal numbers
- ASCII code and UNICODE
- Sampling and Quantitizing
  - *Example: sound*

Number representation

We are used to counting in base 10:

1000	100	10	1
10^3	10^2	10^1	10^0
..... thousands	hundreds	tens	units

Example:

1	7	3	2
1000	100	10	1

← *digits*

1x1000+7x100+3x10+2x1 = 1732

Number representation

Computers use a different system: base 2:

1024	512	256	128	64	32	16	8	4	2	1
2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Example:

1	1	0	1	1	0	0	0	1	0	0
1024	512	256	128	64	32	16	8	4	2	1

← *bits*

1x1024+1x512+0x256+1x128+1x64+0x32+ 0x16+ 0x8 + 1x4 + 0x2 + 0x1 = 1732

Number representation

Base 10	Base 2
0	0
1	1
2	10
3	11
4	100
5	101
6	110
…	…
253	11111101
254	11111110
255	11111111
…	…

Conversion

From base 2 to base 10:

1	1	1	0	1	0	1	0	1	0	0
1024	512	256	128	64	32	16	8	4	2	1

1x1024+1x512+1x256+0x128+1x64+0x32+ 1x16+ 0x8 + 1x4 + 0x2 + 0x1 = 1876

From base 10 to base 2:

1877 %2 = 938 Remainder 1
938 %2 = 469 Remainder 0
469 %2 = 234 Remainder 1
234 %2 = 117 Remainder 0
117 %2 = 58 Remainder 1
58 %2 = 29 Remainder 0
29 %2 = 14 Remainder 1
14 %2 = 7 Remainder 0
7 %2 = 3 Remainder 1
3 %2 = 1 Remainder 1
1 %2 = 0 Remainder 1

1877 (base10) = 11101010101 (base 2)

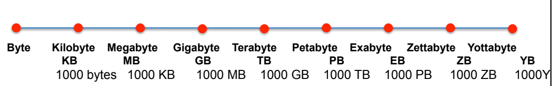
Facts about Binary Numbers

-Each "digit" of a binary number (each 0 or 1) is called a bit

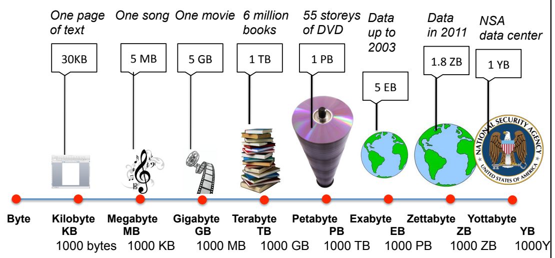
-1 byte = 8 bits
-1 KB = 1 kilobyte = 2^{10} bytes = 1024 bytes (≈1 thousand bytes)
-1 MB = 1 Megabyte = 2^{20} bytes = 1,048,580 bytes (≈ 1 million bytes)
-1 GB = 1 Gigabyte = 2^{30} bytes = 1,073,741,824 bytes (≈1 billion bytes)
-1 TB = 1 Tetabyte = 2^{40} bytes = 1,099,511,627,776 bytes (≈ 1 trillion bytes)

-A byte can represent numbers up to 255: 11111111 (base 2) = 255 (base 10)
-The largest number represented by a binary number of size N is $2^N - 1$

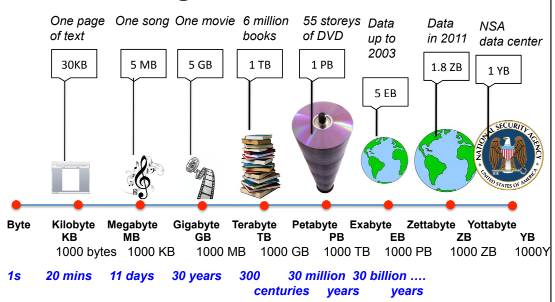
Big Data: Volume



Big Data: Volume



Big Data: Volume



Hexadecimal numbers

While base 10 and base 2 are the most common bases used to represent numbers, others are also possible: base 16 is another popular one, corresponding to hexadecimal numbers

256	16	1
16^2	16^1	16^0

The "digits" are: 0 1 2 3 4 5 6 7 8 9 A B C D E F

Example:

2	A	F
256	16	1

2x256 + 10*16 + 15x1 = 687

Hexadecimal numbers

Everything we have learned in base 10 should be studied again in other bases !!

Example: multiplication table in base 16:

x	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	2	4	6	8	A	C	E	10	12	14	16	18	1A	1C	1E
3	3	6	9	C	F	12	15	18	1B	1E	21	24	27	2A	2D
4	4	8	C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	5	A	F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
6	6	C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
7	7	E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
8	8	10	18	20	28	30	38	40	48	50	58	60	68	70	78
9	9	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A	A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96
B	B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5
C	C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4
D	D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E	E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2
F	F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1

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

Conversion: From base 2 to base 16, and back

This is in fact easy!!

-From base 2 to base 16:

Example: 11011000100

Step 1: break into groups of 4 (starting from the right):

110 1100 0100

Step 2: pad with 0, if needed:

0110 1100 0100

Step 3: convert each group of 4, using table:

6 C 4

Step 4: regroup:

6C4

11011000100 (base 2) = 6C4 (base 16)

Conversion: From base 2 to base 16, and back

From base 16 to base 2:

Example: 4FD

Step 1: split:

4 F D

Step 2: convert each "digit", using table:

0100 1111 1101

Step 3: Remove leading 0, if needed

100 1111 1101

Step 4: regroup:

10011111101

4FD (base 16) = 10011111101 (base 2)

Digital Data

- Binary and Hexadecimal numbers
- ASCII code and UNICODE
- Sampling and Quantitizing
  - *Example: sound*

ASCII

American Standard Code for Information Interchange

So far, we have seen how computers can handle numbers.
What about letters / characters?
The ASCII code was designed for that: it assigns a number to each character:

A-Z: 65-90
a-z: 97-122
0-9: 48-57

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

UNICODE

ASCII only contains 127 characters (though an extended version exists with 257 characters).
This is by far not enough as it is too restrictive to the English language.

UNICODE was developed to alleviate this problem: the latest version, UNICODE 5.1.0 contains more than 100,000 characters, covering most existing languages.

For more information, see:

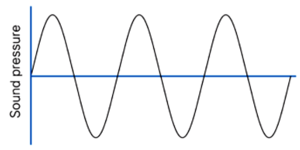
http://www.unicode.org/versions/Unicode5.1.0/

Digital Data

- Binary and Hexadecimal numbers
- ASCII code and UNICODE
- Sampling and Quantitizing
  - *Example: sound*

Digital Sound

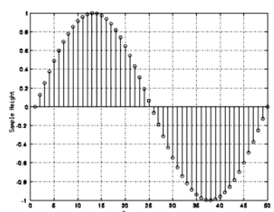
Sound is produced by the vibration of a media like air or water. Audio refers to the sound within the range of human hearing.
Naturally, a sound signal is **analog**, i.e. **continuous in both time and amplitude.**



To store and process sound information in a computer or to transmit it through a computer network, we must first **convert the analog signal to digital form** using an analog-to-digital converter (ADC); the conversion involves two steps: (1) **sampling**, and (2) **quantization**.

Sampling

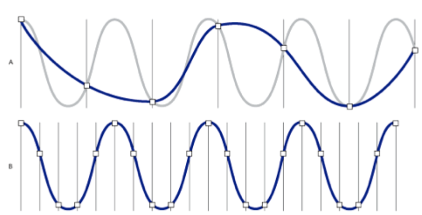
Sampling is the process of examining the value of a continuous function at regular intervals.



Sampling usually occurs at uniform intervals, which are referred to as sampling intervals. The reciprocal of sampling interval is referred to as the sampling frequency or sampling rate.
If the sampling is done in time domain, the unit of sampling interval is second and the unit of sampling rate is Hz, which means cycles per second.

Sampling

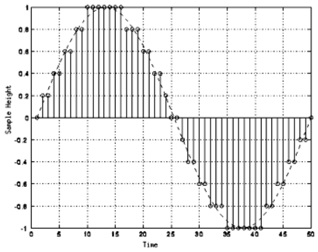
Note that choosing the sampling rate is not innocent:



A higher sampling rate usually allows for a better representation of the original sound wave. However, when the sampling rate is set to twice the highest frequency in the signal, the original sound wave can be reconstructed without loss from the samples. This is known as the Nyquist theorem.

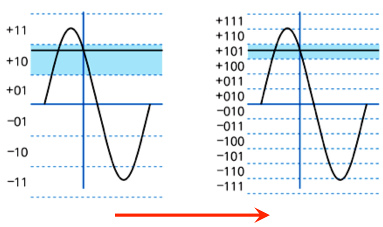
Quantization

Quantization is the process of limiting the value of a sample of a continuous function to one of a predetermined number of allowed values, which can then be represented by a finite number of bits.



Quantization

The number of bits used to store each intensity defines the accuracy of the digital sound:



Adding one bit makes the sample twice as accurate

Audio Sound

Sampling:

The human ear can hear sound up to 20,000 Hz: a sampling rate of 40,000 Hz is therefore sufficient. The standard for digital audio is 44,100 Hz.

Quantization:

The current standard for the digital representation of audio sound is to use 16 bits (i.e 65536 levels, half positive and half negative)

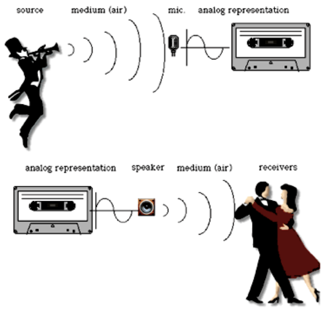
How much space do we need to store one minute of music?

- 60 seconds
- 44,100 samples
- 16 bits (2 bytes) per sample
- 2 channels (stereo)

S = 60x44100x2x2 = 10,534,000 bytes ≈ 10 MB !!

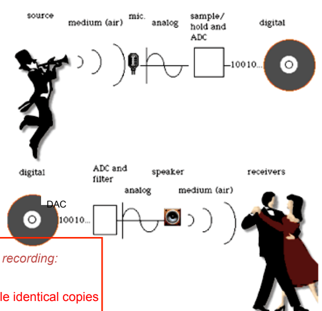
1 hour of music would be more than 600 MB !

Analog Recording



www.atpm.com/6.02/digitalaudio.shtml

DIGITAL RECORDING



Advantages of digital recording:

-Faithful
- can make multiple identical copies

-Can be processed
- compression (MP3)

www.atpm.com/6.02/digitalaudio.shtml