Binary representation and data

Loriano Storchi

loriano@storchi.org

http:://www.storchi.org/

Binary representation of numbers

- In a **positional numbering system** given **the base this directly defines the number of symbols (digits) that are used** to write the number.
  - For example, in the decimal system we use 10 symbols (0,1,2,3,4,5,6,7,8,9)
- The modern numbering systems are positional, **so the number is written specifying the order of the digits, and each digit takes on a value depending on its position**
  - Example $423 = 4 * 10^2 + 2 * 10^1 + 3 * 10^0$

If you want 4 hundred 2 tens and 3 units

Binary representation of numbers

- In general given a **base b** I will have b symbols (digits) and then **an integer N will be written as**:
  - Value $N = c_n * b^n + c_{n-1} * b^{n-1} + \ldots + c_0 * b^0$
- Similarly if I have a number $N = 0.c_1 c_2 \ldots c_n$
  - Value $N = c_1 * b^{-1} + c_2 * b^{-2} + \ldots + c_n * b^{-n}$
- Now consider the simplest case that of the representation of **unsigned integers**
- If I use a **numbering system with base b with n digits I can represent a maximum of b^n different numbers, so all the numbers from 0 up to $b^n - 1$**
- For example in **base 10 it is clear that using two digits can represent all the numbers from 0 to 99 then $10^2 = 100$ distinct numbers**

Binary representation of numbers

- Using a **base 2**, then only **two symbols 0 and 1**, always remaining within the representation of positive integers using **n digits could represent at most all the numbers between 0 and 2^n -1**
- For example using two digits could represent 4 distinct numbers:
  - $00 = 0 * 2^1 + 0 * 2^0 = 0$
  - $01 = 0 * 2^1 + 1 * 2^0 = 1$
  - $10 = 1 * 2^1 + 0 * 2^0 = 2$
  - $11 = 1 * 2^1 + 1 * 2^0 = 3$

DIGITIZATION

DIGITIZATION

- The process of converting information into digital format
- **Information is thus organized into bits**
- **Images, sounds, documents or any analog signal is converted in bits, thus generating a series of numbers that describe a discrete set of its points or samples.**
- In modern practice, the digitized data is in the form of binary numbers, which facilitate computer processing and other operations, but, strictly speaking, digitizing simply means the conversion of analog source material into a numerical format

DIGITIZATION

- **A byte (one bite) represents the sequence of 8 bits** in a modern way and has historically become the basic element of the addressability and therefore the basic unit of information.
- **8 bit means that with 1 byte I can represent 2^8 = 256 different values**. So in the case of integers unsigned the numbers from **0 to 255**.
  - If **I use a bit to indicate the sign for example 0 positive and 1 negative, I can represent the integers from -128 to 127 (from 10000000 to 01111111)**
- **Or with 8 bits I can represent 256 different characters.**

ASCII

- Extended ASCII uses 8-bit
- ASCII initially US-ASCII 7-bit

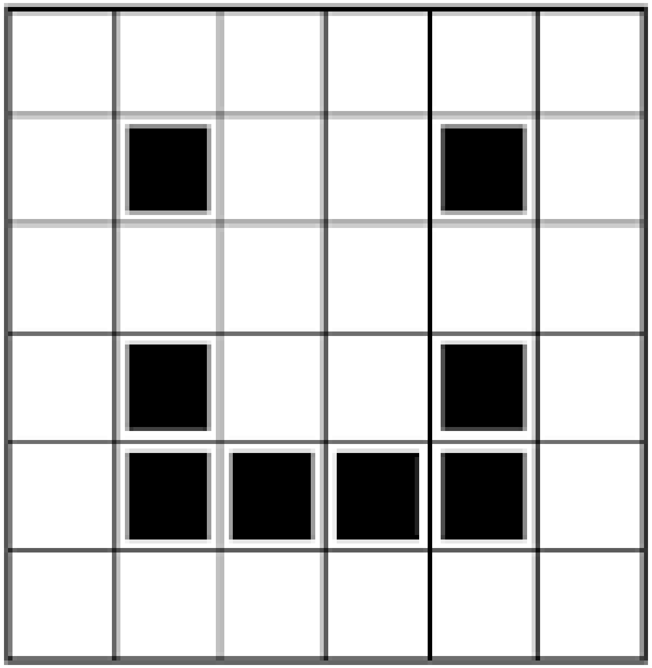
000	NUL	033	!	066	B	099	c	132	ä	165	Ñ	198	ã	231	þ
001	Start Of Header (SOH)	034	"	067	C	100	d	133	à	166	ª	199	Ã	232	Þ
002	Start Of Text (STX)	035	#	068	D	101	e	134	å	167	º	200	╚	233	Ú
003	End Of Text (ETX)	036	$	069	E	102	f	135	ç	168	¿	201	╔	234	Û
004	End Of Transmission (EOT)	037	%	070	F	103	g	136	ê	169	®	202	╩	235	Ù
005	Enquiry	038	&	071	G	104	h	137	ë	170	¬	203	╦	236	ý
006	Acknowledge (ACK)	039		072	H	105	i	138	è	171	½	204	╠	237	Ý
007	Bell	040	(	073	I	106	j	139	ï	172	¼	205	═	238	¯
008	Backspace (BS)	041	)	074	J	107	k	140	î	173	¡	206	╬	239	´
009	Horizontal Tab	042	*	075	K	108	l	141	ì	174	«	207	¤	240	-
010	Line Feed **(LF)**	043	+	076	L	109	m	142	Ä	175	»	208	ð	241	±
011	Vertical Tab	044	,	077	M	110	n	143	Å	176	░	209	Ð	242	‗
012	Form Feed **(FF)**	045	-	078	N	111	o	144	É	177	▒	210	Ê	243	¾
013	Carriage Return **(CR)**	046	.	079	O	112	p	145	æ	178	▓	211	Ë	244	¶
014	Shift Out	047	/	080	P	113	q	146	Æ	179	│	212	È	245	§
015	Shift In	048	0	081	Q	114	r	147	ô	180	┤	213	ı	246	÷
016	Dataline Escape (DLE)	049	1	082	R	115	s	148	ö	181	Á	214	Í	247	¸
017	DC 1 (XON)	050	2	083	S	116	t	149	ò	182	Â	215	Î	248	°
018	DC 2	051	3	084	T	117	u	150	û	183	À	216	Ï	249	¨
019	DC 3 (XOFF)	052	4	085	U	118	v	151	ù	184	©	217	┘	250	·
020	DC 4	053	5	086	V	119	w	152	ÿ	185	╣	218	┌	251	¹
021	Negative Acknowledge (NAK)	054	6	087	W	120	x	153	Ö	186	║	219	█	252	³
022	Synchronous Idle	055	7	088	X	121	y	154	Ü	187	╗	220	▄	253	²
023	End Of Transmission Block	056	8	089	Y	122	z	155	ø	188	╝	221	¦	254	■
024	Cancel	057	9	090	Z	123	{	156	£	189	¢	222	Ì	255	
025	End Of Medium	058	:	091	[	124	|	157	Ø	190	¥	223	▀		
026	Substitude	059	;	092	\	125	}	158	×	191	┐	224	Ó		
027	Escape **(ESC)**	060	<	093	]	126	~	159	ƒ	192	└	225	ß		
028	File Seperator	061	=	094	^	127 (DEL)	⌂	160	á	193	┴	226	Ô		
029	Group Seperator	062	>	095	_	128	Ç	161	í	194	┬	227	Ò		
030	Record Seperator	063	?	096	`	129	ü	162	ó	195	├	228	õ		
031	Unit Seperator	064	@	097	a	130	é	163	ú	196	─	229	Õ		
032	*SPACE* (SP)	065	A	098	b	131	â	164	ñ	197	┼	230	µ		

Information encoding

- **An encoding is a convention**
  - As seen before then **the bit sequence 01001100 can represent the character L (capital L) or if instead we mean an integer value unsigned it represents the decimal number 76**.
- For example **every image is composed of pixels, if I used only 1 bit for each pixel I could have only black and white images (1 black pixel, 0 white pixel)**.

Information encoding

- **Example:**

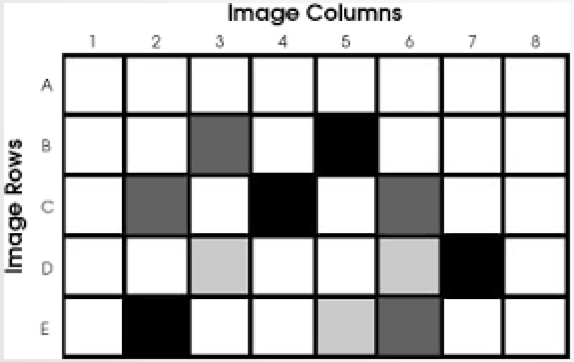


1	1	1	1	1	1
1	0	1	1	0	1
1	1	1	1	1	1
1	0	1	1	0	1
1	0	0	0	0	1
1	1	1	1	1	1

Information encoding

- if I use more bits to represent each pixel, I can have gray ranges or colors instead. And from here you can play, video ...

The pixel represents the smallest autonomous element image. Each pixel is therefore characterized by its own position



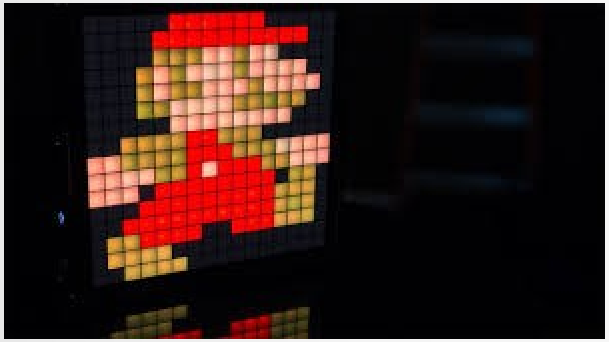
The total number of pixels in a digital image is called a resolution. For example, if I have a 10 x 10 grid, the image will consist of 100 pixels

dpi = number of dots per inch, for example in a monitor will typically have 72 pixels per inch

Depth: in the case of a grayscale an image can use 8 bits for each pixel thus having available 2^8 = 256 shades of gray

Information encoding

- Color images



In the case of color images each pixel is characterized by three color scales for the three fundamental colors, **RGB (Red, Green, Blue)**

For example an 8-bit image will have for each color 256 possible shades for red, 256 for green and 256 for blue. Therefore in total 16777216 possible shades of color. In the case of **12-bit images we will have 2^{12} = 4096 shades for each color, in total 68718476736 possible colors for each pixel.**

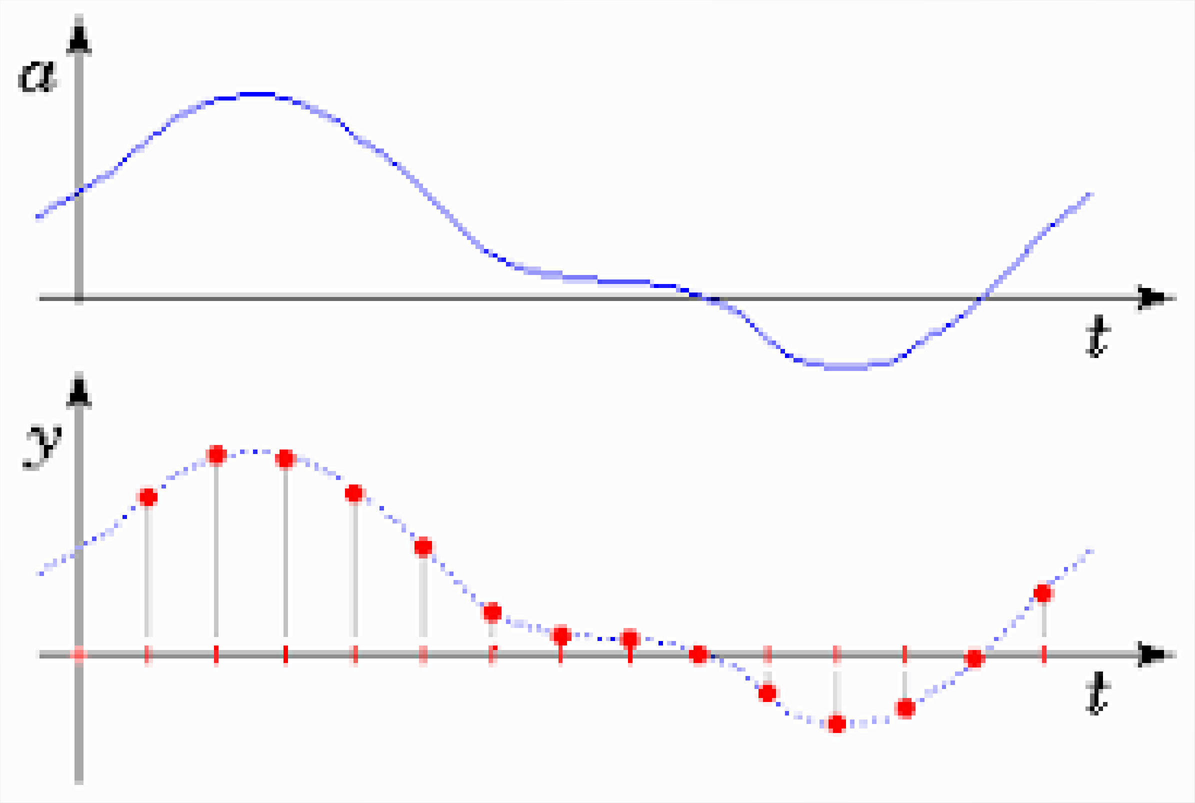
Analog signal

- Digital Analog Conversion

sampling - time discretization (Nyquist-Shannon sampling theorem)

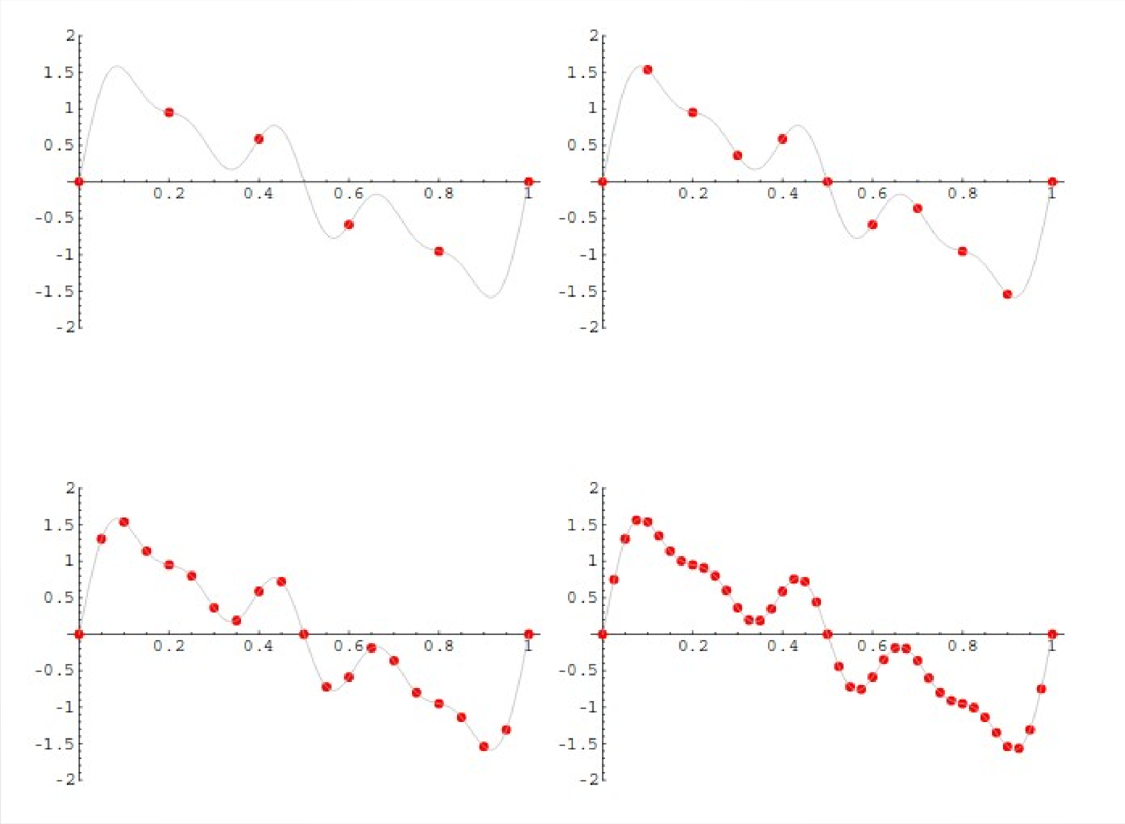
quantization - discretization of the amplitude

coding - use of binary "words" to express the value of the signa



Analog signal

- Obviously, fidelity improves as the number of samples per unit of time increases (sampling frequency) and the number of quantization levels



Digital audio

- Since the **human ear can hear frequencies in a certain range (20, 20000 Hz) the sampling theorem tells us that we need to sample at 40 kHz**
- Typically, we use a number of **quantization levels much larger than 256, often 16 bits**
- For example in the case of a **CD we have two channels (stereo) at 44.1 kHz and 16 bits (2^{16} = 65536)**
- So the **bit rate = 44100 samples / s * 16 bits * 2 channels = 1411200 bits / s = 176400 Bytes / s**
- If we want **1 minute of music we need 60 * 176400 Byte / s = 10584000 Bytes which is about 10 MiB**

Final considerations

- Compression of images and sound (therefore also video) is essentially based on the elimination of information to which the human ear and the human eye are not very sensitive
- Not only that ….

OVERFLOW, UNDERFLOW, VARIABLES

Computer memory is limited

- **OVERFLOW**: the available bits are not sufficient to represent the result. **UNDERFLOW** number too small 3/2 I can not represent 1.5 using integer, I can only represent 1 or 2
- For example **if I use a 8 bits registers** to perform a computation with positive integers I can only represent numbers from **0 to 255, so what happens if I try to add 1 to 255 ?**

1 1 1 1 1 1 1 1 +

0 0 0 0 0 0 0 1 =

____

1 0 0 0 0 0 0 0 0 I need more than 8 bits to store the result

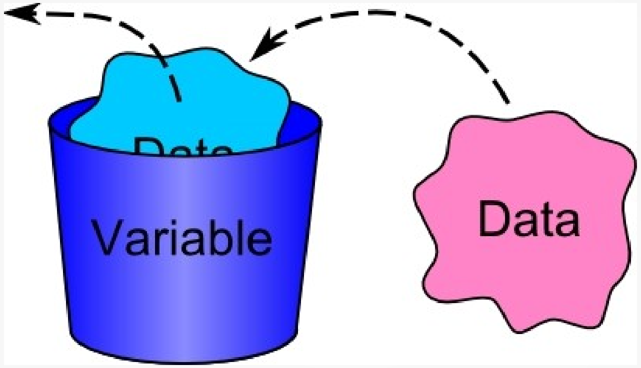
Data type

- See the difference between strongly typed and non-typed languages, dynamic and static typing
- Programming languages have native data types, such as integer, floating-point, boolean and character. **Different types different sizes and therefore different ranges** (exact Algebra and not ...)

label	size (bytes)	smallest value	largest value
byte	1	-128	127
short	2	-32768	32767
int	4	-2147483648	2147483647
long	8	-9223372036854775808	9223372036854775807
char	2	0	65535

Variable

- **Variable** - A variable is a value that can change during the execution of a program.



- **Declaration**: A declaration ensures that sufficient memory is reserved in which to store the values and also states the variable’s data type
- **Scope** - Scope indicates whether a variable can be used by all parts of a program or only within limited sections of the program – for example, within a subroutine.