Cosa?

Il calcolatore memorizza ed elabora vari tipi di informazioni

- Numeri
- Testi
- Immagini
- Suoni
- ...

Come?

I calcolatori usano la **codifica binaria** (a due valori) per rappresentare l'informazione.

Definizione

Chiameremo **BIT** (**BI**nary digi**T**) l'unità minimale di rappresentazione: `0` oppure `1`.

Il computer si basa quindi su un alfabeto binario $\{0, 1\}$ e le informazioni vengono "*associate*" a stringhe su tale alfabeto.

Example

- `0` NO
- `1` SI

Example

- `00` Inverno
- `01` Primavera
- `10` Estate
- `11` Autunno

Perché?

I computer **non sono in grado** di effettuare operazioni (e dunque di immagazzinare informazioni) utilizzando un linguaggio più vicino al nostro modo di pensare e di intendere le cose!

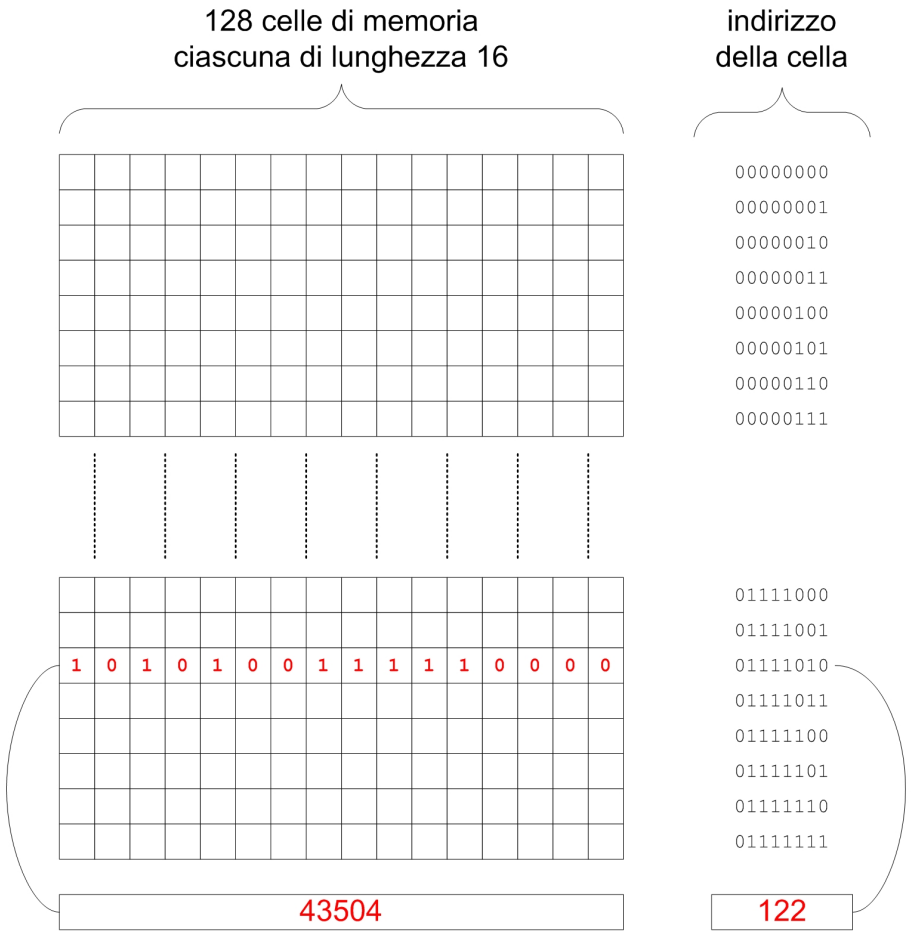
A livello fisico **i computer lavorano su segnali elettrici**, ed i segnali elettrici possono trovarsi solo in due stati (**acceso** e **spento**).

Definizione

Ogni informazione gestita dal computer deve necessariamente essere convertita in formato **binario**.

Quanti bit servono per codificare i nomi dei giorni della settimana?

Celle di Memoria



Il Byte

Definizione

Con **N** bit (ognuno dei quali può assumere 2 valori) possiamo rappresentare 2^N informazioni diverse.

Definizione

Per rappresentare **M** informazioni dobbiamo usare (almeno) **N** bit, in modo che $2^N \geq$ **M**

Definizione

Una sequenza di **8 bit** viene chiamata **Byte**. Con un byte possiamo codificare $2^8 = 256$ informazioni diverse.

Come si misura la memoria di un calcolatore?

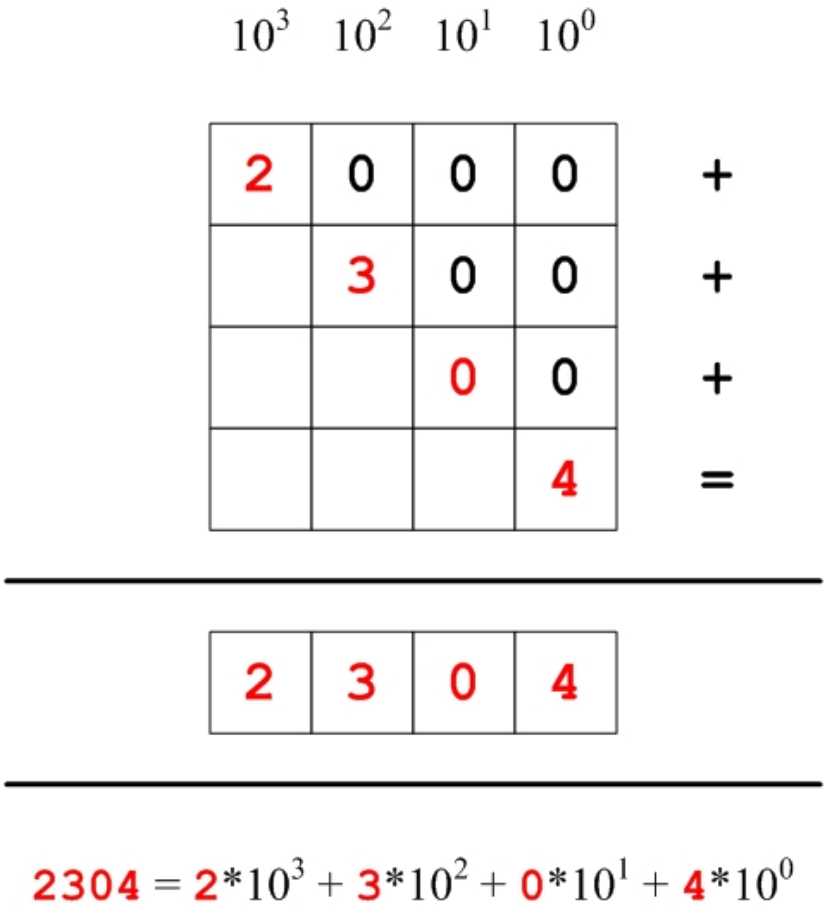
Definizione

Il **byte** è definito come unità di misura di memoria.

I multipli del byte sono:

- KiloByte (**KB**) - 1 **KB** = 2^{10} **byte** = 1024 **byte**
- MegaByte (**MB**) - 1 **MB** = 2^{20} **byte** = 1024 **KB** $\simeq$ 1 milione di **byte**
- GigaByte (**GB**) - 1 **GB** = 2^{30} **byte** = 1024 **MB** $\simeq$ 1 miliardo di **byte**
- TeraByte (**TB**) - 1 **TB** = 2^{40} **byte** = 1024 **GB** $\simeq$ 1000 miliardi di **byte**
- ...

Notazione decimale



Notazione binaria

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
1	0	0	0	0	0	0	0	+
	0	0	0	0	0	0	0	+
		1	0	0	0	0	0	+
			0	0	0	0	0	+
				0	0	0	0	+
					1	0	0	+
						0	0	+
							1	=
1	0	1	0	0	1	0	1	

$$165 = 1*2^7 + 1*2^5 + 1*2^2 + 1*2^0 = 128 + 32 + 4 + 1$$

Come codifichiamo i caratteri?

Per rappresentare i simboli dell'alfabeto anglosassone (`0 1 2 ... A B ... a b ...`) bastano 7 bit ($2^7 = 128$)

- *Nota*: `B` e `b` sono simboli diversi
- *Contiamo*: 26 maiuscole + 26 minuscole + 10 cifre + 33 segni di interpunzione + 33 caratteri di controllo = 128 oggetti

Per l'alfabeto esteso con simboli quali `è ½ © ...` bastano 8 bit ($2^8 = 256$) come nella codifica accettata universalmente chiamata **ASCII** (*American Standard Code for Information Interchange*) esteso

Per manipolare un numero maggiore di simboli si utilizza la codifica **UNICODE** a 16 bit ($2^{16} = 65.536$)

Codifica ASCII (standard)

Byte	Cod.	Char
00000000	0	Null
00000001	1	Start of heading
00000010	2	Start of text
00000011	3	End of text
00000100	4	End of transmit
00000101	5	Enquiry
00000110	6	Acknowledge
00000111	7	Audible bell
00001000	8	Backspace
00001001	9	Horizontal tab
00001010	10	Line feed
00001011	11	Vertical tab
00001100	12	Form Feed
00001101	13	Carriage return
00001110	14	Shift out
00001111	15	Shift in
00010000	16	Data link escape
00010001	17	Device control 1
00010010	18	Device control 2
00010011	19	Device control 3
00010100	20	Device control 4
00010101	21	Neg. acknowledge
00010110	22	Synchronous idle
00010111	23	End trans. block
00011000	24	Cancel
00011001	25	End of medium
00011010	26	Substitution
00011011	27	Escape
00011100	28	File separator
00011101	29	Group separator
00011110	30	Record Separator
00011111	31	Unit separator

Byte	Cod.	Char
00100000	32	Spc
00100001	33	!
00100010	34	"
00100011	35	#
00100100	36	$
00100101	37	%
00100110	38	&
00100111	39	'
00101000	40	(
00101001	41	)
00101010	42	*
00101011	43	+
00101100	44	,
00101101	45	-
00101110	46	.
00101111	47	/
00110000	48	0
00110001	49	1
00110010	50	2
00110011	51	3
00110100	52	4
00110101	53	5
00110110	54	6
00110111	55	7
00111000	56	8
00111001	57	9
00111010	58	:
00111011	59	;
00111100	60	<
00111101	61	=
00111110	62	>
00111111	63	?

Byte	Cod.	Char
01000000	64	@
01000001	65	A
01000010	66	B
01000011	67	C
01000100	68	D
01000101	69	E
01000110	70	F
01000111	71	G
01001000	72	H
01001001	73	I
01001010	74	J
01001011	75	K
01001100	76	L
01001101	77	M
01001110	78	N
01001111	79	O
01010000	80	P
01010001	81	Q
01010010	82	R
01010011	83	S
01010100	84	T
01010101	85	U
01010110	86	V
01010111	87	W
01011000	88	X
01011001	89	Y
01011010	90	Z
01011011	91	[
01011100	92	\
01011101	93	]
01011110	94	^
01011111	95	_

Byte	Cod.	Char
01100000	96	`
01100001	97	a
01100010	98	b
01100011	99	c
01100100	100	d
01100101	101	e
01100110	102	f
01100111	103	g
01101000	104	h
01101001	105	i
01101010	106	j
01101011	107	k
01101100	108	l
01101101	109	m
01101110	110	n
01101111	111	o
01110000	112	p
01110001	113	q
01110010	114	r
01110011	115	s
01110100	116	t
01110101	117	u
01110110	118	v
01110111	119	w
01111000	120	x
01111001	121	y
01111010	122	z
01111011	123	{
01111100	124	|
01111101	125	}
01111110	126	~
01111111	127	Del

Codifica ASCII (esteso)

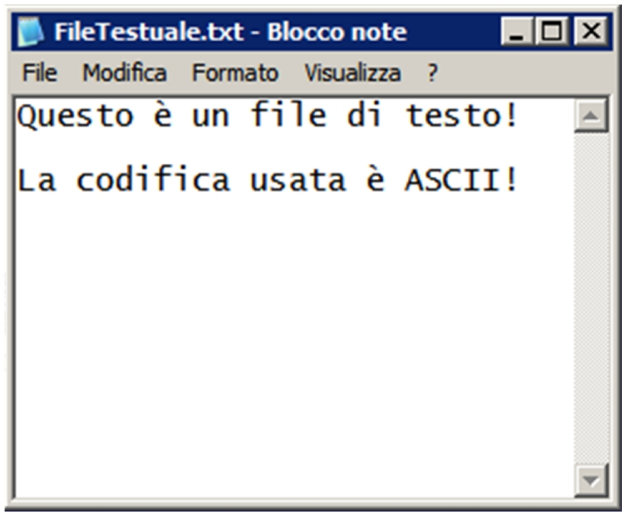
Byte	Cod.	Char
10000000	128	Ç
10000001	129	ü
10000010	130	é
10000011	131	â
10000100	132	ä
10000101	133	à
10000110	134	å
10000111	135	ç
10001000	136	ê
10001001	137	ë
10001010	138	è
10001011	139	ï
10001100	140	î
10001101	141	ì
10001110	142	Ä
10001111	143	Å
10010000	144	É
10010001	145	æ
10010010	146	Æ
10010011	147	ô
10010100	148	ö
10010101	149	ò
10010110	150	û
10010111	151	ù
10011000	152	ÿ
10011001	153	Ö
10011010	154	Ü
10011011	155	ø
10011100	156	£
10011101	157	Ø
10011110	158	×
10011111	159	ƒ

Byte	Cod.	Char
10100000	160	á
10100001	161	í
10100010	162	ó
10100011	163	ú
10100100	164	ñ
10100101	165	Ñ
10100110	166	ª
10100111	167	º
10101000	168	¿
10101001	169	®
10101010	170	¬
10101011	171	½
10101100	172	¼
10101101	173	¡
10101110	174	«
10101111	175	»
10110000	176	▒
10110001	177	▒
10110010	178	▒
10110011	179	¦
10110100	180	¦
10110101	181	Á
10110110	182	Â
10110111	183	À
10111000	184	©
10111001	185	¦
10111010	186	¦
10111011	187	+
10111100	188	+
10111101	189	¢
10111110	190	¥
10111111	191	+

Byte	Cod.	Char
11000000	192	+
11000001	193	-
11000010	194	-
11000011	195	+
11000100	196	-
11000101	197	+
11000110	198	ã
11000111	199	Ã
11001000	200	+
11001001	201	+
11001010	202	-
11001011	203	-
11001100	204	¦
11001101	205	-
11001110	206	+
11001111	207	¤
11010000	208	ð
11010001	209	Ð
11010010	210	Ê
11010011	211	Ë
11010100	212	È
11010101	213	ı
11010110	214	Í
11010111	215	Î
11011000	216	Ï
11011001	217	+
11011010	218	+
11011011	219	▒
11011100	220	▒
11011101	221	¦
11011110	222	Ì
11011111	223	▒

Byte	Cod.	Char
11100000	224	Ó
11100001	225	ß
11100010	226	Ô
11100011	227	Ò
11100100	228	õ
11100101	229	Õ
11100110	230	µ
11100111	231	þ
11101000	232	Þ
11101001	233	Ú
11101010	234	Û
11101011	235	Ù
11101100	236	ý
11101101	237	Ý
11101110	238	¯
11101111	239	´
11110000	240	-
11110001	241	±
11110010	242	‗
11110011	243	¾
11110100	244	¶
11110101	245	§
11110110	246	÷
11110111	247	¸
11111000	248	°
11111001	249	¨
11111010	250	·
11111011	251	¹
11111100	252	³
11111101	253	²
11111110	254	■
11111111	255	_

Esempio



testo visualizzato	*txt*	*dec*	*bin*
Ques	Ques	081117101115	01010001011101010110010101110011
to è	to è	116111032138	01110100011011110010000010001010
un	un	032117110032	00100000011101010110111000100000
file	file	102105108101	01100110011010010110110001100101
di	di	032100105032	00100000011001000110100100100000
test	test	116101115116	01110100011001010111001101110100
o!	o!	111033013010	01101111001000010000110100001010
La	La	013010076097	00001101000010100100110001100001
cod	cod	032099111100	00100000011000110110111101100100
ific	ific	105102105099	01101001011001100110100101100011
a us	a us	097032117115	01100001001000000111010101110011
ata	ata	097116097032	01100001011101000110000100100000
è AS	è AS	138032065083	10001010001000000100000101010011
CII!	CII!	067073073033	01000011010010010100100100100001