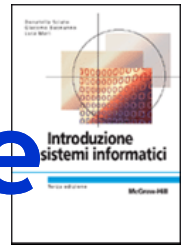
Capitolo 2
L'informazione e la sua codifica

Informatica e Informazione

La codifica dell'informazione

Informazione e Informatica

Informatica e telecomunicazione



- **Cos'è l'informatica?**
  - lo studio sistematico degli algoritmi che descrivono e trasformano l'informazione: la loro teoria, analisi, progetto, efficienza, realizzazione e applicazione
    [ACM – Association for Computing Machinery]
  - **la scienza della rappresentazione e dell'elaborazione dell'informazione**
- **Cos'è la telecomunicazione?**
  - la trasmissione **rapida** a **distanza** dell'informazione

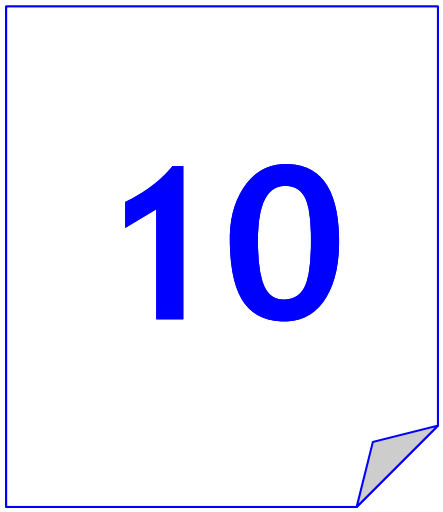
Informazione e supporto

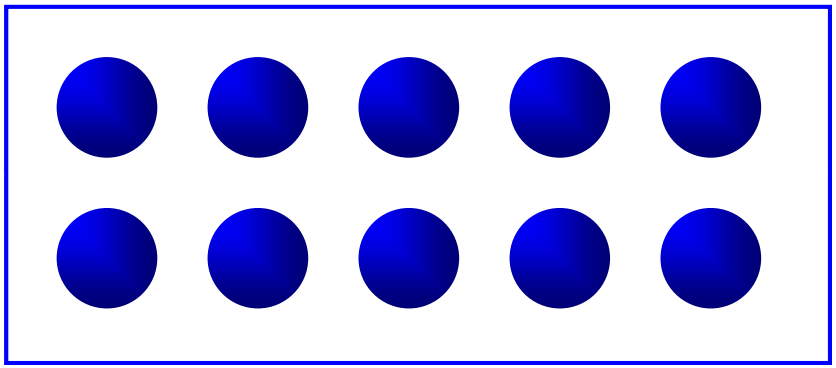
- L'informazione è "portata da", o "trasmessa su", o "memorizzata in", o "contenuta in" qualcosa; questo "qualcosa" però non è l'informazione stessa.

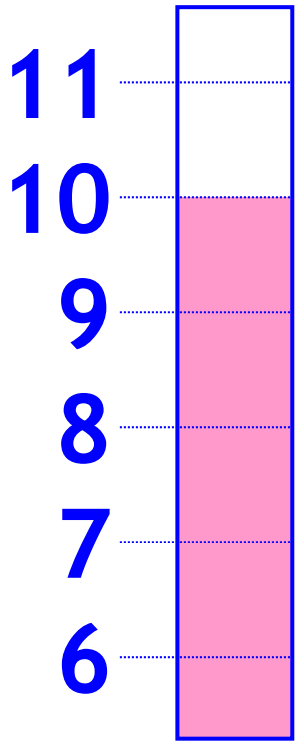
- Ogni supporto ha le sue caratteristiche in quanto mezzo su cui può essere scritta dell'informazione.

Informazione e supporti (1)

La stessa informazione può essere scritta su supporti differenti.







Informazione e supporto (2)

- **Distinguere informazione e supporto fisico è distinguere tra "entità logiche" ed "entità fisiche":**
  - l'informazione **richiede un supporto fisico**, ma non coincide con esso;
  - l'informazione è un'entità **extra-fisica**, non interpretabile in termini di materia-energia e sottoposta alle leggi della fisica solo perché basata su un supporto fisico.

- **L'informazione si può creare e distruggere.**

Quali caratteristiche deve avere un sistema fisico per supportare informazioni?

- Si ottiene informazione quando, dato un insieme di alternative possibili, la lettura del supporto ne elimina alcune e ne seleziona altre.
- Condizione necessaria perché un supporto possa portare informazione è che possa assumere configurazioni differenti, a ognuna delle quali venga associata una differente entità di informazione.

Supporto fisico: 1ª condizione

- **Deve consentire di potere identificare delle differenze**
  - Es: voglio rappresentare 2 alternative

- **Cosa rappresenta ciascuna configurazione?**

Configurazioni e codici

- **A ogni configurazione del supporto deve essere associata un'entità di informazione:**
  - Prima Configurazione = interruttore "**ON**" = "**Divina Commedia**";
  - Seconda Configurazione = interruttore "**OFF**" = "**I Promessi Sposi**".

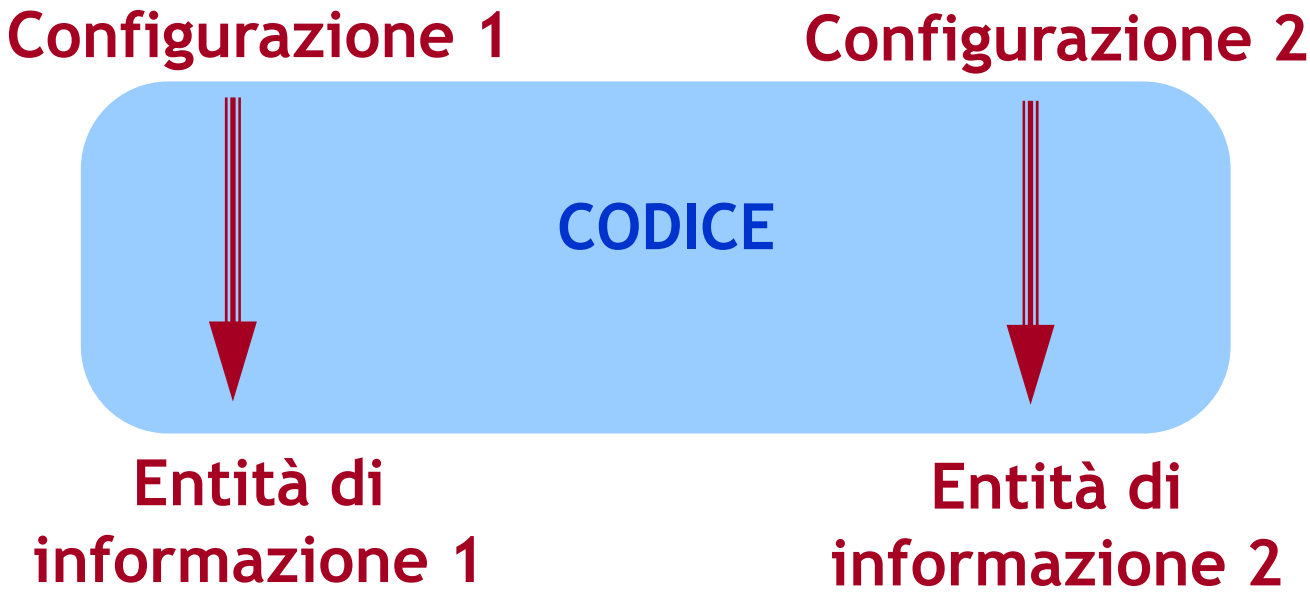
- **Per interpretare le differenti configurazioni del supporto in termini di informazione è necessario conoscere il codice (cioè la regola) che a ogni configurazione ammessa del supporto associa un'entità di informazione.**

- **La definizione di un codice comporta che siano identificati in modo non ambiguo l'insieme delle possibili configurazioni del supporto e l'insieme delle possibili entità di informazione a cui ci si vuole riferire.**

- **Variando il codice è possibile riferirsi a entità di informazione differenti utilizzando uno stesso supporto fisico.**

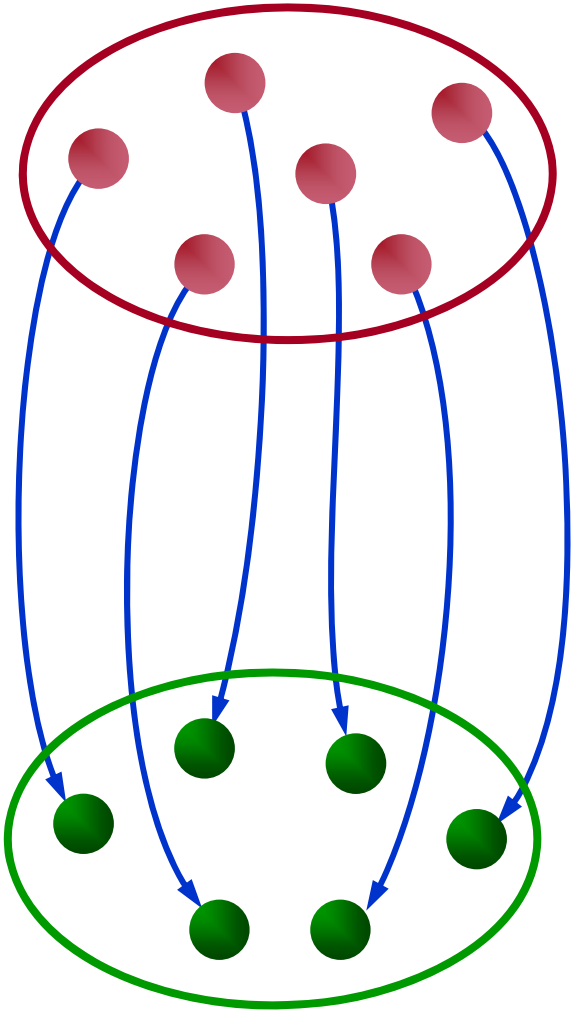
Supporto fisico: 2a condizione

Deve essere condivisa una regola per attribuire un significato a ciascuna configurazione



Definire un codice

- Identificare
  - { Configurazioni }
  - { Entità informazione }
- Associare gli elementi dei 2 insiemi



Teoria dell'informazione

- **Ambito di applicazione: caratterizzare le condizioni per la trasmissione di segnali in termini di**
  - **adeguatezza** del supporto adottato per la trasmissione
  - **accuratezza** della trasmissione stessa.
- **Quali problemi si pone:**
  - un certo supporto può essere utilizzato per la memorizzazione di una certa quantità di informazione?
  - con quale velocità una certa quantità di informazione può essere trasferita a distanza mediante un certo supporto?
  - con quale grado di accuratezza un certo messaggio è stato trasmesso?
- **La presenza di informazione è condizionata dal fatto che il supporto sia in grado di assumere diverse configurazioni.**

La codifica dell'informazione

Codifica dati e istruzioni

- **Algoritmo**
  - **descrizione** della **soluzione di problema** scritta in modo da poter essere eseguita da un **esecutore** (eventualmente diverso dall'autore dell'algoritmo)
  - sequenza di **istruzioni** che operano su **dati**.
- **Programma**
  - **algoritmo** scritto in modo da poter essere eseguito da un **calcolatore** (esecutore automatico)

- **Per scrivere un programma è necessario rappresentare istruzioni e dati in un formato tale che l'esecutore automatico sia capace di memorizzare e manipolare.**

Codifica dati e istruzioni

- **Alfabeto dei simboli**
  - cifre "0", "1", …, "9", separatore decimale (","), separatore delle migliaia (".") e segni positivo ("+") o negativo ("-").
- **Regole di composizione (sintassi), che definiscono le successioni "ben formate"**
  - "1.234,5" è la rappresentazione di un numero;
  - "1,23,45" non lo è.
- **Codice (semantica)**
  - "1.234,5" = $1\times10^3 + 2\times10^2 + 3\times10^1 + 4\times10^0 + 5\times10^{-1}$
  - "1,23,45" = ??

- **Lo stesso alfabeto può essere utilizzato con codici diversi:**
  - "123,456" = $1\times10^2 + 2\times10^1 + 3\times10^0 + 4\times10^{-1} + 5\times10^{-2} + 6\times10^{-3}$, [IT]
  - "123,456" = $1\times10^5 + 2\times10^4 + 3\times10^3 + 4\times10^2 + 5\times10^1 + 6\times10^0$, [UK]

Codifica Binaria

- **Alfabeto binario**: usiamo dispositivi con solo due stati
- Problema: assegnare un **codice univoco** a tutti gli oggetti compresi in un insieme predefinito (e.g. studenti)

- Quanti **oggetti** posso codificare con **k** bit:
  - 1 bit $\Rightarrow$ 2 stati (0, 1) $\Rightarrow$ 2 oggetti (e.g. Vero/Falso)
  - 2 bit $\Rightarrow$ 4 stati (00, 01, 10, 11) $\Rightarrow$ 4 oggetti
  - 3 bit $\Rightarrow$ 8 stati (000, 001, …, 111) $\Rightarrow$ 8 oggetti
  - …
  - **k bit $\Rightarrow$ 2^k stati $\Rightarrow$ 2^k oggetti**

- Quanti **bit** mi servono per codificare **N** oggetti:
  - $N \leq 2^k \Rightarrow k \geq \log_2 N \Rightarrow \mathbf{k = \lceil \log_2 N \rceil}$ (intero superiore)

- Attenzione:
  ipotesi implicita che i codici abbiano tutti la **stessa lunghezza**

Esempio di codifica binaria

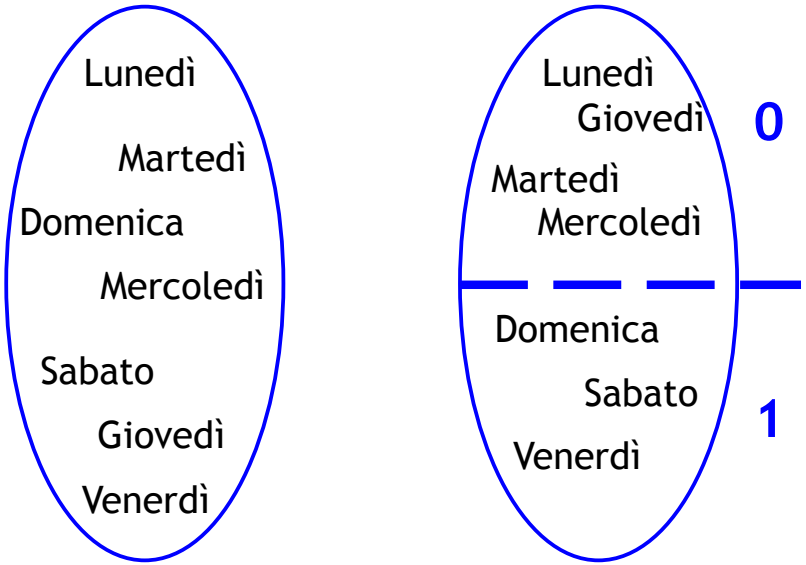
- **Problema:**
  assegnare un codice binario univoco a tutti i giorni della settimana

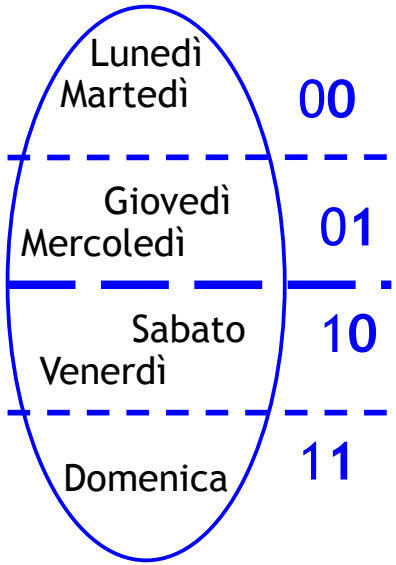
- **Giorni della settimana: $N = 7 \Rightarrow k \geq \log_2 7 \Rightarrow k = 3$**

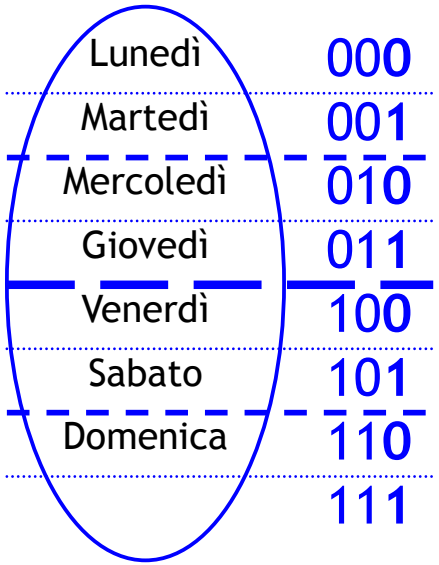
- **Con 3 bit possiamo ottenere 8 diverse configurazioni:**
  - Ne servono 7, quali utilizziamo?
  - Quale configurazione associamo a quale giorno?

- **Attenzione:**
  ipotesi che i codici abbiano tutti la stessa lunghezza

I giorni della settimana in binario (1)





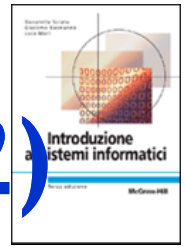


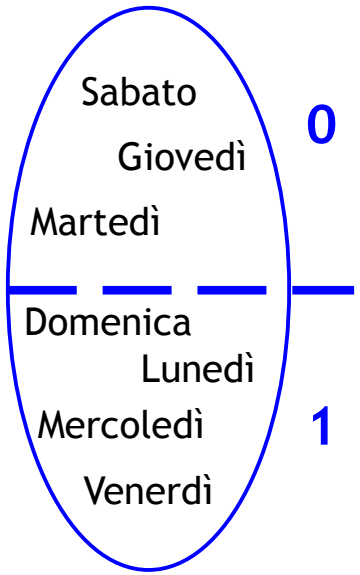
1 bit
2 “gruppi”

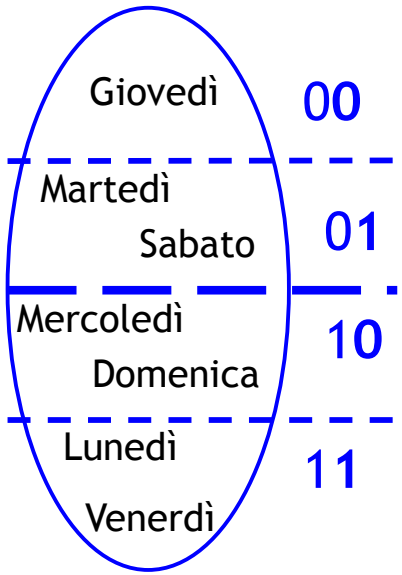
2 bit
4 “gruppi”

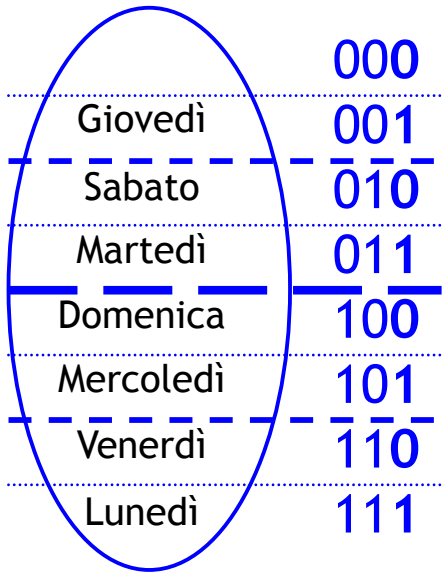
3 bit
8 “gruppi”

I giorni della settimana in binario (2)









1 bit
2 “gruppi”

2 bit
4 “gruppi”

3 bit
8 “gruppi”

Codifica binaria dei caratteri

- **Quanti sono gli oggetti compresi nell'insieme?**
  - 26 lettere maiuscole + 26 minuscole ⇒ 52
  - 10 cifre
  - Circa 30 segni d'interpunzione
  - Circa 30 caratteri di controllo (EOF, CR, LF, …)

  circa 120 oggetti complessivi ⇒ $k = \lceil \log_2 120 \rceil = 7$
- **Codice ASCII: utilizza 7 bit e quindi può rappresentare al massimo 2^7=128 caratteri**
  - Con 8 bit (= byte) rappresento 256 caratteri (ASCII esteso)
  - Si stanno diffondendo codici più estesi (e.g. UNICODE) per rappresentare anche i caratteri delle lingue orientali

ASCII su 7 bit

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
010	sp	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
011	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
101	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
110	`	a	b	c	d	e	f	g	h	I	j	k	l	m	n	o
111	p	q	r	s	t	u	v	w	x	Y	z	{	|	}	~	canc

bit, Byte, KiloByte, MegaByte, …

bit = solo due stati, “0” oppure “1”.

Byte = 8 bit, quindi $2^8 = 256$ stati

KiloByte [KB] = 2^{10} Byte = 1024 Byte ~ 10^3 Byte

MegaByte [MB] = 2^{20} Byte = 1'048'576 Byte ~ 10^6 Byte

GigaByte [GB] = 2^{30} Byte ~ 10^9 Byte

TeraByte [TB] = 2^{40} Byte ~ 10^{12} Byte

PetaByte [PB] = 2^{50} Byte ~ 10^{15} Byte

ExaByte [EB] = 2^{60} Byte ~ 10^{18} Byte

Lo standard IEC per i prefissi binari

Grandezza	Nome	Simbolo		Dimensione	SI	Diff. %
Kilo binario	Kibi	Ki	2^{10}	1'024	10^3	2.40%
Mega binario	Mebi	Mi	$(2^{10})^2$	1'048'576	$(10^3)^2$	4.86%
Giga binario	Gibi	Gi	$(2^{10})^3$	1'073'741'824	$(10^3)^3$	7.37%
Tera binario	Tebi	Ti	$(2^{10})^4$	1'099'511'627'776	$(10^3)^4$	9.95%
Peta binario	Pebi	Pi	$(2^{10})^5$	1'125'899'906'842'624	$(10^3)^5$	12.59%
Exa binario	Exbi	Ei	$(2^{10})^6$	1'152'921'504'606'846'976	$(10^3)^6$	15.29%
Zetta binario	Zebi	Zi	$(2^{10})^7$	1'180'591'620'717'411'303'424	$(10^3)^7$	18.06%
Yotta binario	Yobi	Yi	$(2^{10})^8$	1'208'925'819'614'629'174'706'176	$(10^3)^8$	20.89%

- **Usando questi prefissi si può risolvere l'ambiguità**
  - capacità di un disco fisso: 120 GB = 111.76 GiB,
  - capacità di un floppy: 1.406 MiB = 1.475 MB

La codifica delle istruzioni

- **Si segue lo schema presentato per i caratteri alfanumerici:**
  - **quali e quante** sono le istruzioni da codificare?
  - qual è la **lunghezza** delle successioni di bit da utilizzare ?
  - qual è la **corrispondenza** tra istruzioni e successioni di bit ?

Istruzioni aritmetico-logiche	
Codice	Istruzione
0111 1100	**ADD**
0111 1101	**SUB**
0111 1110	**AND**
… … …	**… … …**

Istruzioni per il trasferimento dati	
Codice	Istruzione
1110 1000	**LOAD**
1111 1000	**STORE**
… … …	**… … …**
… … …	**… … …**

Istruzioni di controllo	
Codice	Istruzione
0100 1001	**IF_EQ**
0100 1000	**GOTO**
0100 1100	**RETURN**
… … …	**… … …**

Numeri naturali

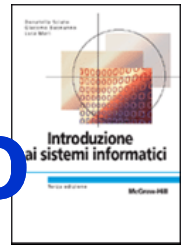
- **Sistema di numerazione posizionale in base** b
  - $c_k c_{k-1} \ldots c_0$ rappresenta $c_k \times b^k + c_{k-1} \times b^{k-1} + \ldots + c_0 \times b^0$
  - b=10 $\Rightarrow$ 1101_{dieci} indica $1 \times 10^3 + 1 \times 10^2 + 0 \times 10 + 1 \times 10^0$
- **Conversione binario $\Rightarrow$ decimale**
  - basta scrivere il numero secondo la notazione posizionale utilizzando già il sistema decimale
  - b=2 $\Rightarrow$ 1101_{due} indica $1 \times 2^3 + 1 \times 2^2 + 0 \times 2 + 1 \times 2^0 = 13_{dieci}$
- **Conversione decimale $\Rightarrow$ binario**
  - Si potrebbe utilizzare lo stesso metodo indicato sopra, ma è molto complesso
  - b=10 $\Rightarrow$ 345_{dieci} indica $11 \times 1010^{10} + 100 \times 1010^1 + 101 \times 1010^0$

Conversione binario ⇒ decimale

$$
\begin{aligned}
101100_{due} = {} & 1_{dieci} \times 2^{5}_{dieci} + 0_{dieci} \times 2^{4}_{dieci} + 1_{dieci} \times 2^{3}_{dieci} + 1_{dieci} \times 2^{2}_{dieci} + 0_{dieci} \times 2^{1}_{dieci} + \\
& + 0_{dieci} \times 2^{0}_{dieci} = \\
= {} & 1_{dieci} \times 32_{dieci} + 0_{dieci} \times 16_{dieci} + 1_{dieci} \times 8_{dieci} + 1_{dieci} \times 4_{dieci} + 0_{dieci} \times 2_{dieci} \\
& + 0_{dieci} \times 1_{dieci} = \\
= {} & 32_{dieci} + 8_{dieci} + 4_{dieci} = \\
= {} & 44_{dieci}
\end{aligned}
$$

$$
\begin{aligned}
101110101_{due} = {} & 1_{dieci} \times 2^{8}_{dieci} + 0_{dieci} \times 2^{7}_{dieci} + 1_{dieci} \times 2^{6}_{dieci} + 1_{dieci} \times 2^{5}_{dieci} + 1_{dieci} \times 2^{4}_{dieci} + \\
& 0_{dieci} \times 2^{3}_{dieci} + 1_{dieci} \times 2^{2}_{dieci} + 0_{dieci} \times 2^{1}_{dieci} + 1_{dieci} \times 2^{0}_{dieci} = \\
= {} & 1_{dieci} \times 256_{dieci} + 0_{dieci} \times 128_{dieci} + 1_{dieci} \times 64_{dieci} + 1_{dieci} \times 32_{dieci} + \\
& 1_{dieci} \times 16_{dieci} + 0_{dieci} \times 8_{dieci} + 1_{dieci} \times 4_{dieci} + 0_{dieci} \times 2_{dieci} + 1_{dieci} \times 1_{dieci} = \\
= {} & 256_{dieci} + 64_{dieci} + 32_{dieci} + 16_{dieci} + 4_{dieci} + 1_{dieci} = \\
= {} & 373_{dieci}
\end{aligned}
$$

Conversione decimale ⇒ binario



- Sistema di numerazione posizionale in base B che, in questo contesto si può ipotizzare diversa da dieci

  $c_{n-1}c_{n-2}\ldots c_1c_0 = c_{n-1}\times B^{n-1} + c_{n-2}\times B^{n-2} + \ldots + c_1\times B^1 + c_0\times B^0$

  $c_{n-1}c_{n-2}\ldots c_1c_0 = c_{n-1}\times B^{n-1} + c_{n-2}\times B^{n-2} + \ldots + c_1\times B + c_0$

  (infatti $B^1 = B$ e $B^0 = 1$)

- Dividendo il numero per il valore della base, il risultato che si ottiene è:

  $(c_{n-1}\times B^{n-1} + c_{n-2}\times B^{n-2} + \ldots + c_1\times B + c_0)/B =$

  $= c_{n-1}\times B^{n-2} + c_{n-2}\times B^{n-3} + \ldots + c_1 + c_0/B$

- che può essere scomposto in modo da evidenziare quoziente e resto:

  quoziente $= c_{n-1}\times B^{n-2} + c_{n-2}\times B^{n-3} + \ldots + c_1$

  resto $= c_0$

- Il resto della divisione corrisponde all'ultima cifra della rappresentazione in base B del numero, ma il suo valore è indipendente dalla base che si utilizza per effettuare i conti.
- Applicando lo stesso procedimento al quoziente si ottiene la penultima cifra della rappresentazione in base B
- Ripetendo la procedura è possibile ottenere tutte le altre cifre.

Conversione decimale ⇒ binario

(cifra binaria meno significativa)

$573_{dieci} : 2_{dieci}$	⇒	*quoziente*	286_{dieci}	*resto* 1_{dieci}
$286_{dieci} : 2_{dieci}$	⇒	*quoziente*	143_{dieci}	*resto* 0_{dieci}
$143_{dieci} : 2_{dieci}$	⇒	*quoziente*	71_{dieci}	*resto* 1_{dieci}
$71_{dieci} : 2_{dieci}$	⇒	*quoziente*	35_{dieci}	*resto* 1_{dieci}
$35_{dieci} : 2_{dieci}$	⇒	*quoziente*	17_{dieci}	*resto* 1_{dieci}
$17_{dieci} : 2_{dieci}$	⇒	*quoziente*	8_{dieci}	*resto* 1_{dieci}
$8_{dieci} : 2_{dieci}$	⇒	*quoziente*	4_{dieci}	*resto* 0_{dieci}
$4_{dieci} : 2_{dieci}$	⇒	*quoziente*	2_{dieci}	*resto* 0_{dieci}
$2_{dieci} : 2_{dieci}$	⇒	*quoziente*	1_{dieci}	*resto* 0_{dieci}
$1_{dieci} : 2_{dieci}$	⇒	*quoziente*	0_{dieci}	*resto* 1_{dieci}

(cifra binaria più significativa)

$$1\ 000\ 111\ 101_{due} = 573_{dieci}$$

Conversione decimale ⇒ binario

Si calcolano i resti delle divisioni per due

18 : 2 = 9 resto 0
9 : 2 = 4 resto 1
4 : 2 = 2 resto 0
2 : 2 = 1 resto 0
1 : 2 = 0 resto 1

10010

137 : 2 = 68 resto 1
68 : 2 = 34 resto 0
34 : 2 = 17 resto 0
17 : 2 = 8 resto 1
8 : 2 = 4 resto 0
4 : 2 = 2 resto 0
2 : 2 = 1 resto 0
1 : 2 = 0 resto 1

10001001

Conversione decimale ⇒ binario

			18	2			
128	64	32	16	8	4	2	1
0	0	0	1	0	0	1	0

Numeri binari: operazioni

- Operazioni di somma di numeri binari naturali.
- Con gli 8 bit utilizzati negli esempi qui riportati si possono rappresentare i numeri naturali fino a 255dieci.
- Operazioni che producono un risultato maggiore provocano il superamento della capacità di rappresentazione (indicato in gergo dal termine inglese overflow).

1 1		1 1	
$0\,0\,0\,1\,1\,0\,0\,1_{due}$ +	$2\,5_{dieci}$ +	$0\,0\,1\,0\,1\,1\,0\,0_{due}$ +	$4\,4_{dieci}$ +
$0\,0\,1\,1\,1\,1\,0\,0_{due}$ =	$6\,0_{dieci}$ =	$0\,1\,0\,0\,1\,1\,1\,0_{due}$ =	$7\,8_{dieci}$ =
$0\,1\,0\,1\,0\,1\,0\,1_{due}$	$8\,5_{dieci}$	$0\,1\,1\,1\,1\,0\,1\,0_{due}$	$1\,2\,2_{dieci}$

Numeri naturali binari nei calcolatori

- Per la codifica dei numeri naturali (interi positivi) si utilizzano abitualmente successioni di 32 bit (4 byte) con cui si possono rappresentare i numeri compresi tra 0 e $2^{32}-1$ = 4'294'967'295 ≈ 4×10^{9}.

- Aumentando la lunghezza delle successioni il massimo numero rappresentabile cresce esponenzialmente: passando da 32 a 64 bit il massimo numero rappresentabile diventa $2^{64}-1$ ≈ 16×10^{18} = 1.6×10^{19}.

- A causa del minor numero di simboli dell'alfabeto binario rispetto a quello decimale, un numero codificato in notazione binaria richiede più cifre rispetto a quelle impiegate in notazione decimale.

Numeri interi

- **Alfabeto binario**
  - anche il segno è rappresentato da 0 o 1
  - è indispensabile indicare il numero **k** di bit utilizzati
- **Modulo e segno**
  - **1** bit di segno (0 positivo, 1 negativo)
  - **k - 1** bit di modulo
    - Esempio: $+6_{dieci} = 0110_{ms}$ $-6_{dieci} = 1110_{ms}$
  - si rappresentano i valori da $-2^{k-1}+1$ a $2^{k-1}-1$
    - con 4 bit i valori vanno da –7 a +7
    - con 8 bit i valori vanno da –127 a +127
  - Attenzione: ci sono due rappresentazioni dello 0
    - con 4 bit sono $+0_{dieci} = 0000_{ms}$ $-0_{dieci} = 1000_{ms}$