

UNIVERSITÀ DEGLI STUDI DI SALERNO

# Fondamenti di Informatica

---

Linguaggi, Codifica e Rappresentazione dell'Informazione

Prof. Christian Esposito

Corso di Laurea in Ingegneria Meccanica e Gestionale (Classe I)

A.A. 2016/17

# Cosa abbiamo visto la volta scorsa

---

- Gli elaboratori sono strumenti per risolvere (o aiutare a risolvere) problemi basati sulle informazioni
- Ma come ciò avviene?
  1. Abbiamo bisogno di **codificare** e memorizzare opportunamente **dati e informazioni**
  2. Abbiamo bisogno di **impartire** le giuste **istruzioni per risolvere** correttamente **problemI**

# Cosa vedremo oggi

---

- Gli elaboratori sono strumenti per risolvere (o aiutare a risolvere) problemi basati sulle informazioni
- Ma come ciò avviene?
  1. Abbiamo bisogno di codificare e memorizzare opportunamente dati e informazioni
  2. Abbiamo bisogno di impartire le giuste istruzioni per risolvere correttamente problemi

# Quale Lingua parla l'Elaboratore?

---

- Come rendere dati e informazioni comprensibili ad un elaboratore?
- **Informazioni e dati** per essere trattati da un elaboratore devono essere opportunamente **codificati**



# Linguaggio

---

- **Alfabeto**

- Collezione di simboli grafici, aventi di solito un ordine ben preciso, che servono a rappresentare le parole di una lingua
- I simboli spesso (ma non sempre) sono lettere ovvero rappresentazione scritta di suoni linguistici

- **Vocabolario (o lessico)**

- Insieme delle parole ammissibili di una lingua

- **Grammatica**

- Insieme di regole utili alla corretta costruzione di frasi, sintagmi e parole
  - Il termine si riferisce allo studio delle suddette regole

- **Semantica**

- Studia il significato delle parole (semantica lessicale), degli insiemi delle parole, delle frasi (semantica frasale) e dei testi

# La Funzione dei Linguaggi

---

- I linguaggi (verbali, non verbali, orali, scritti, etc.) sono strumenti che contribuiscono a
  - **Rappresentare le informazioni**
    - Concetti, pensieri, emozioni, etc., vengono formalizzati attraverso i linguaggi per poter essere memorizzati, trasferiti ed elaborati
  - **Memorizzare le informazioni**
    - La scrittura
  - **Trasferire le informazioni**
    - La comunicazione
  - **Elaborare le informazioni**
    - Le deduzioni nella logica

# Problemi relativi ai Linguaggi

---

- **Accordo sui simboli**

- a b c d e f g ...

- **Accordo sul lessico**

- casa, gatto, automobile, vado, ...

- **Accordo sulla grammatica**

- <soggetto verbo complemento>

- **Accordo sulla semantica**

- La nonna chiude la porta (**OK**)
  - La porta chiude la nonna (**NO**)

- **Accordo sulla codifica**

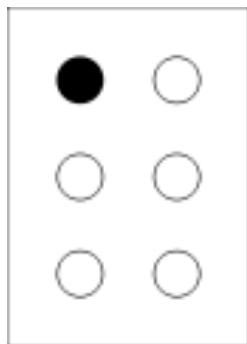
- Regole per trasformare simboli, parole e frasi di un linguaggio in una nuova rappresentazione con possibilità di effettuare in maniera corretta anche l'operazione inversa
    - “a” in codice Morse (Samuel Morse, pittore e storico inglese) è “. – ”
    - “b” in codice Morse è “– . . ”



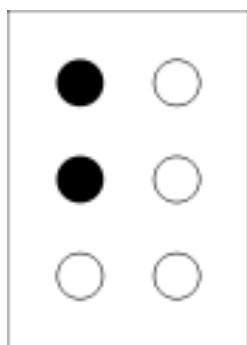
# Codice Braille

---

- Lettera “a”



- Lettera “b”



# Codifica dei Numeri

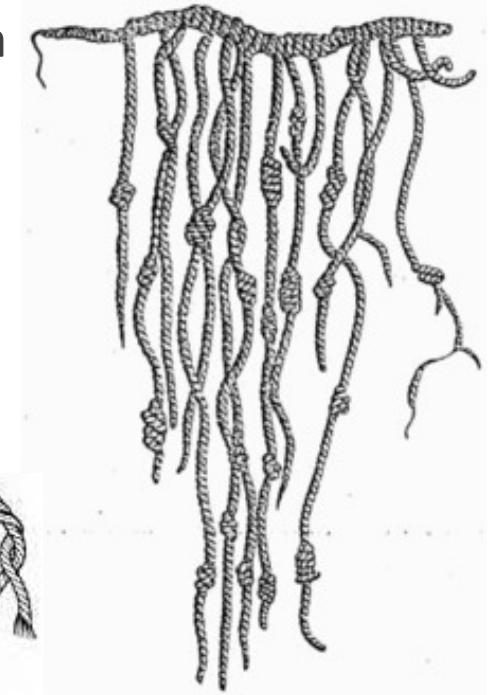
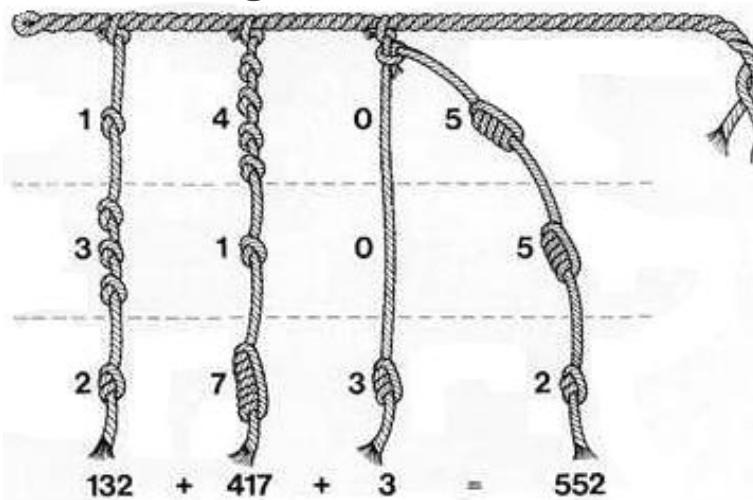
---

- **Linguaggio di partenza**
  - I numeri
- **Codifica 1**
  - Numerazione decimale
    - 5, 45, 670
- **Codifica 2**
  - Numerazione binaria
    - 101, ...
- Codifica 3



# Quipo

- **Quipo** è un insieme di cordicelle annodate, distanziate in modo sistematico tra loro e legate a una corda più grossa e corta che le sorregge.
- I quipo servivano per i calcoli matematici. I nodi delle corde sono di diversi colori: rappresentavano numeri, e dalla loro reciproca posizione se ne potevano ricavare le unità, le decine, le centinaia e le migliaia.



# I Linguaggi Naturali: Ambiguità

---

- Per comunicare tra loro gli uomini hanno sviluppato i **linguaggi naturali**
  - Italiano, inglese, francese, etc
- Una **caratteristica negativa** di tali linguaggi è la loro inerente **ambiguità**
  - Una qualsiasi **frase** formulata è potenzialmente **polisemica**
  - Il significato che viene dato alla frase da chi riceve il messaggio può essere diverso da quello datogli dal mittente
- Comprendere il significato di una frase in **linguaggio naturale** è più semplice se questa viene pronunciata in un dato **contesto**

# I Linguaggi Naturali: Disambiguazione

---

- Comprendere il significato di una frase in linguaggio naturale è più semplice se questa viene pronunciata in un dato contesto
  - Concorso Ippico
  - Circolo Scacchistico
  - Sartoria
  - Palestra

# I Linguaggi Naturali nella Comunicazione con i Calcolatori

---

- Per comunicare con un elaboratore, l'**ambiguità** dei **linguaggi naturali** rappresenta un grosso problema
- Bisognerebbe fornire all'elaboratore anche il contesto e la conoscenza (le regole) per disambiguare le frasi
  - La rappresentazione (comprendibile al calcolatore) di contesto e regole è molto complicata
  - In alcuni casi la disambiguazione è difficile anche per gli uomini
- È necessario l'utilizzo dei cosiddetti **Linguaggi Formali**

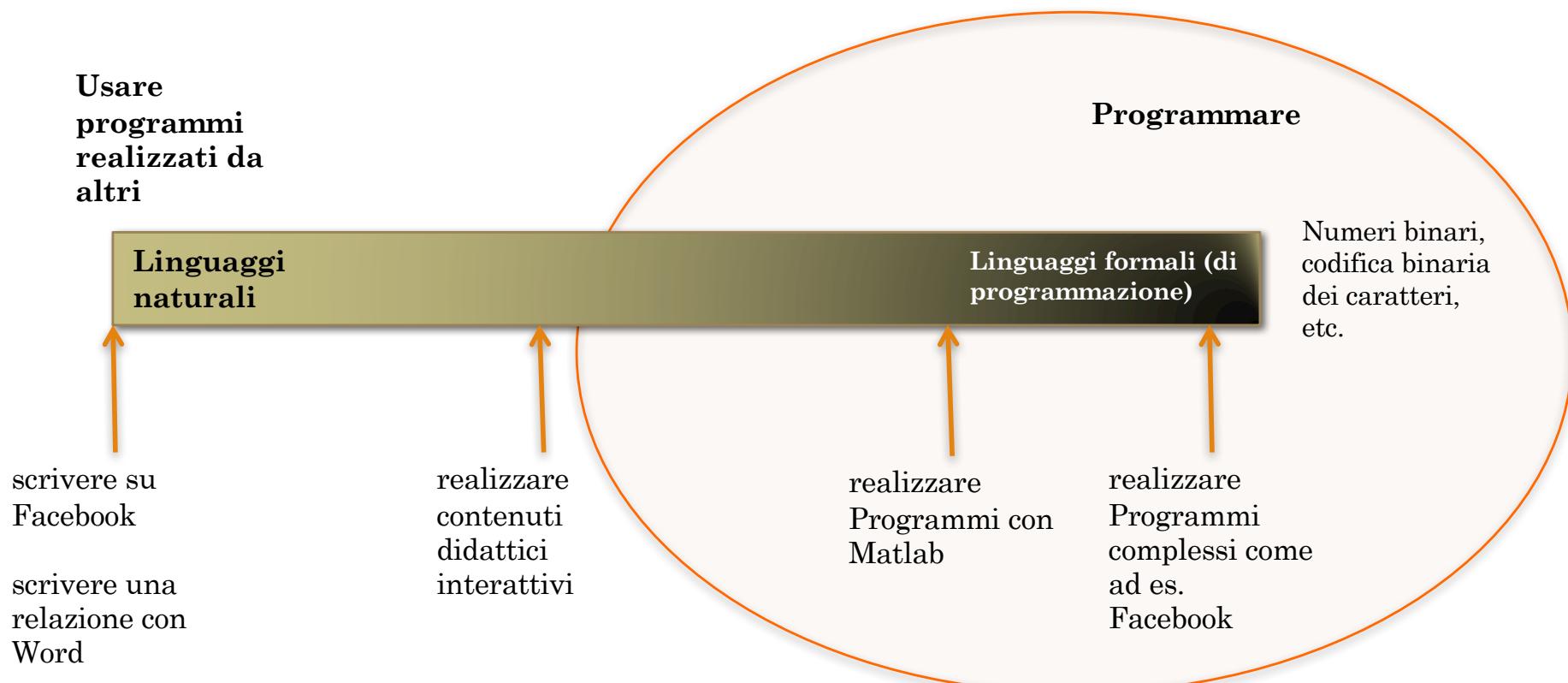
# I Linguaggi Formali

---

- Sviluppati ed impiegati in tutti gli ambiti in cui è necessario evitare l'ambiguità
  - Informatica, matematica, logica, etc
- La definizione di un Linguaggio Formale prevede
  - L'individuazione di un alfabeto, ovvero, di un elenco (finito) di simboli
  - La definizione di una grammatica formale, ovvero un insieme di regole sintattiche che specificano come i simboli dell'alfabeto possono essere combinati tra loro per costituire le frasi ben formate all'interno del linguaggio stesso
- Inoltre, le semantiche formali consentono di attribuire un significato non ambiguo alle frasi in un linguaggio formale

# Usare e Programmare il computer

- I Programmi (o software) risolvono problemi specifici con approccio basato sulle informazioni e vengono eseguiti dai computer



# Rappresentazione dell'Informazione: Accordo sui Simboli

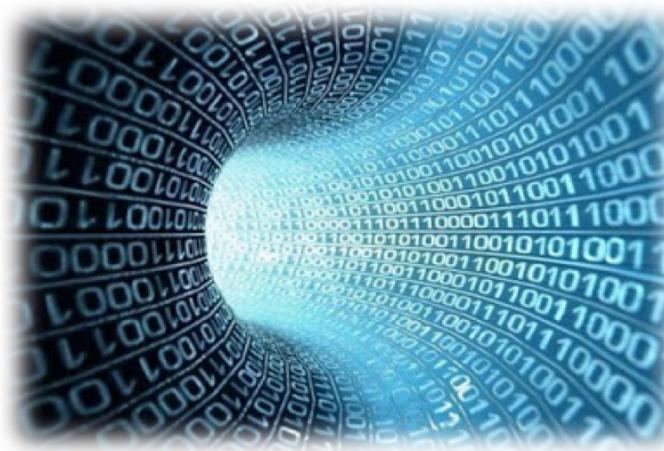
---

- L'informazione è rappresentata dai **dati**, che a loro volta sono **espressi in forma di simboli**
- La **stessa informazione** può essere **codificata con simboli e modalità diverse**
  - 1963 -> simboli "0", "1", "2", ...
  - MCMLXIII -> simboli della codifica romana
  - Milenovecentosessantatre -> rappresentazione testuale
  - ...

# Rappresentazione dell'informazione nei calcolatori – 1/2

---

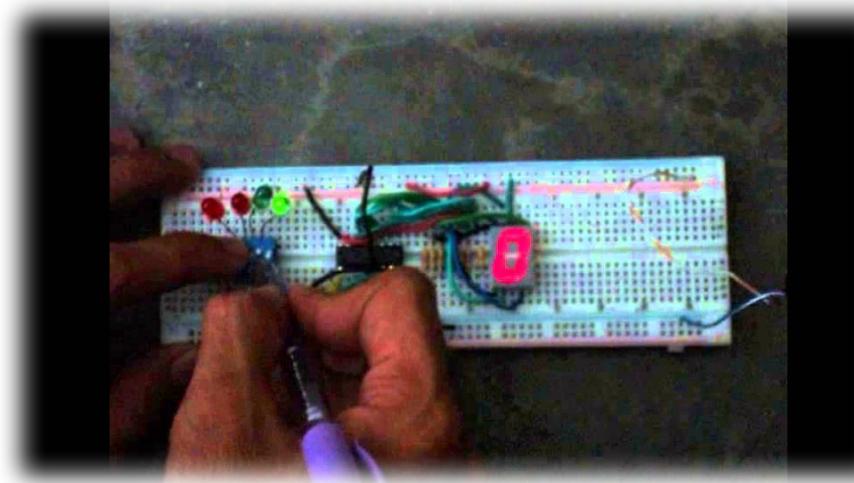
- Consideriamo un alfabeto ridotto che contiene solo i simboli “0” e “1”
- Un **bit** (contrazione di binary digit) è un simbolo scelto sull’alfabeto {0, 1}
- Nei calcolatori **ogni elemento** (numeri, testo, audio, video, istruzioni, etc) viene **rappresentato** esclusivamente con **sequenze di bit**
- I **dati** e le **istruzioni** vengono **codificati** con **sequenze di bit**



# Rappresentazione dell'informazione nei calcolatori – 2/2

---

- Per trattare (memorizzare, elaborare, trasmettere, etc.) l'informazione, il calcolatore
  - **Codifica** l'informazione disponibile scrivendo su un supporto fisico
  - **Manipola** il supporto con opportune trasformazioni fisiche, ottenendo una versione modificata del supporto
  - **Decodifica** l'informazione corrispondente al risultato dell'elaborazione leggendo dalla versione modificata del supporto



# Codifica Binaria

---

- **Alfabeto binario:** usiamo solo due cifre 0,1 (bit)
- **Problema:** assegnare una sequenza di bit univoca a tutti gli oggetti in un insieme predefinito
- **Quanti oggetti posso rappresentare con  $k$  bit?**
  - 1 bit => 2 stati (0, 1) => 2 oggetti
  - 2 bit => 4 stati (00, 01, 10, 11) => 4 oggetti
  - 3 bit => 8 stati (000, 001, ..., 111) => 8 oggetti
  - ...
  - $k$  bit =>  $2^k$  stati =>  $2^k$  oggetti
- **Quanti bit mi servono per codificare  $N$  oggetti?**
  - $N \leq 2^k \Rightarrow k \geq \log_2 N \Rightarrow k = \lceil \log_2 N \rceil$  (intero superiore)

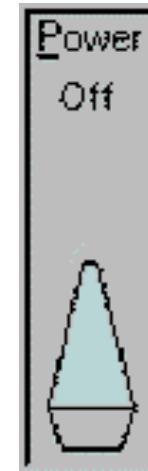
# Combinazioni di Bit

Bit a disposizione	Le combinazioni	Il numero di combinazioni
1	0, 1	$2 = 2^1$
2	00, 01, 10, 11	$4 = 2^2$
3	000, 001, 010, 011, 100, 101, 110, 111	$8 = 2^3$
4	0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111	$16 = 2^4$
5	00000, 00001, 00010, ...	$32 = 2^5$
...	...	...

# Un Primo Esempio di Codifica: Interruttore

---

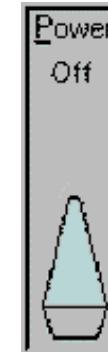
- Due sole possibilità (stati)
  - Spento
  - Acceso
- L'informazione sullo stato dell'interruttore corrisponde dunque alla scelta fra due sole alternative



# Un Primo Esempio di Codifica: Interruttore

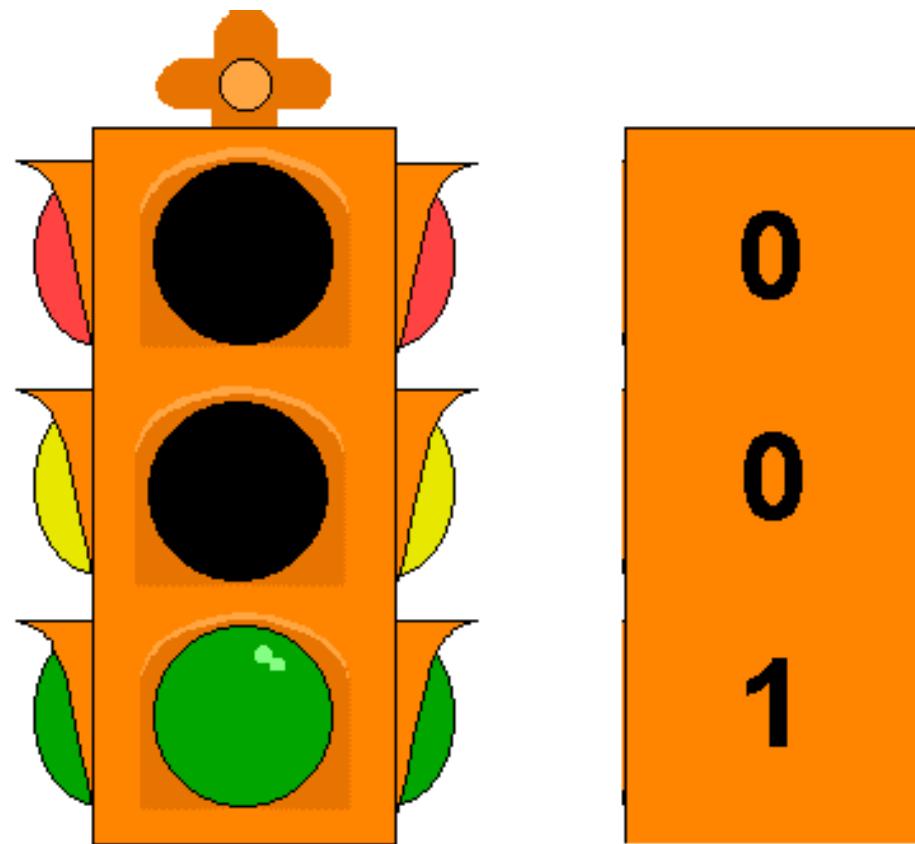
---

- 1 bit rappresenta lo stato dell'interruttore
  - Interruttore acceso: 1
  - Interruttore spento: 0



# Codifica di un'Informazione con Più di Due Stati: Il Semaforo

---



# Diverse Codifiche per le Stesse Informazioni

---

- Rappresentazione degli stati di un semaforo mediante bit

**3 bit**

Stato	Codifica
ROSSO	1 0 0
VERDE	0 0 1
GIALLO	0 1 0

**2 bit**

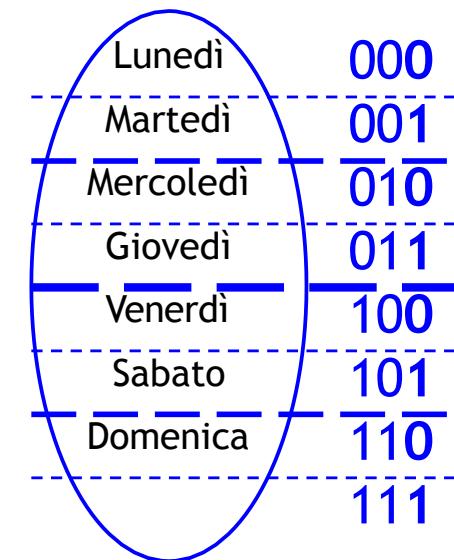
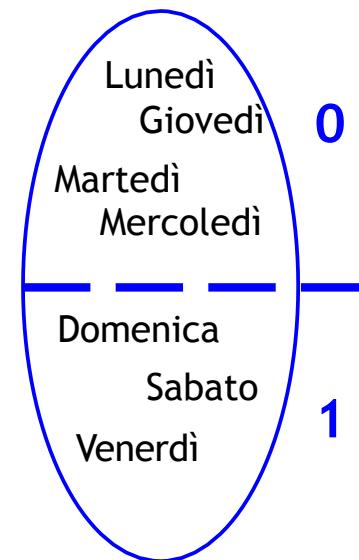
Stato	Codifica
ROSSO	0 0
VERDE	0 1
GIALLO	1 0

# Codifica di un'Informazione con Più di Due Stati: Giorni della Settimana

---

- **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 sequenze
  - Ne servono 7, quali utilizziamo?
  - Quale configurazione associamo a quale giorno?
- **Osservazione:** quanto detto fino ad ora vale sotto l'ipotesi che i codici abbiano tutti la stessa lunghezza

# I Giorni della Settimana in Binario

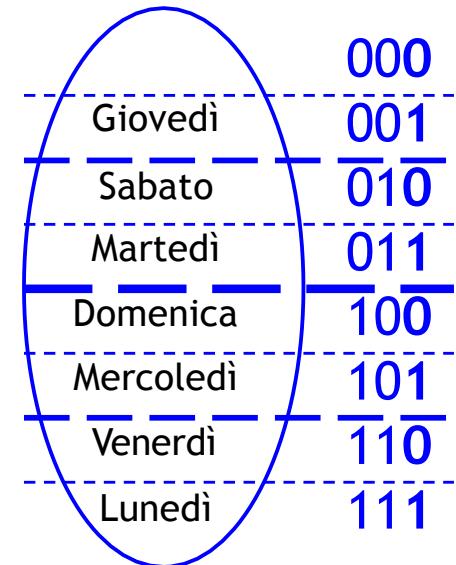
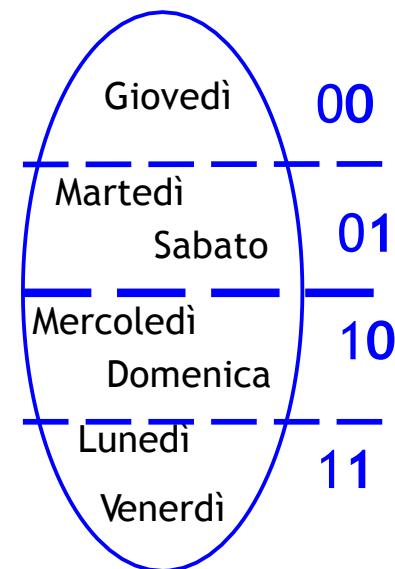
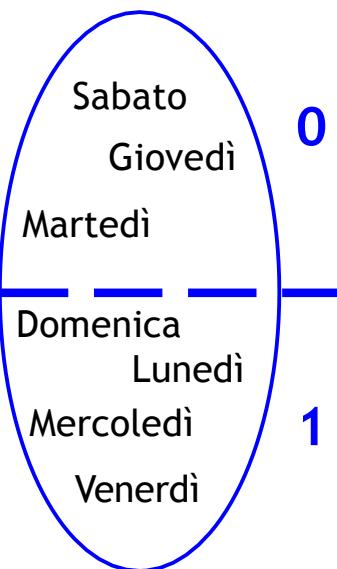


1 bit  
2 “gruppi”

2 bit  
4 “gruppi”

3 bit  
8 “gruppi”

# I Giorni della Settimana in Binario



1 bit  
2 “gruppi”

2 bit  
4 “gruppi”

3 bit  
8 “gruppi”

# Codifica dei Caratteri – 1/5

---

- **Problema:** è possibile applicare queste idee alla rappresentazione di informazione più complessa, ad esempio di un testo?
  - Un testo è rappresentato attraverso una successione di caratteri
  - Ogni carattere viene scelto all'interno di un insieme finito di simboli (alfabeto)

# Codifica dei Caratteri – 2/5

---

- Con 8 bit, è possibile rappresentare la scelta fra 256 alternative diverse ( $2^8=256$ )
  - Da 0000000... a 1111111
  - Passando per tutte le combinazioni intermedie (00000001, 00000010, ...)
- Nel caso del testo, possiamo far corrispondere diverse combinazioni di 8 bit (otto cellette, ciascuna delle quali può contenere 0 o 1) a caratteri diversi

Ogni singolo CARATTERE viene codificato con una combinazione di 8 bit

# Codifica dei Caratteri – 3/5

---

- Ad esempio:
  - 00000000 -> A
  - 00000001 -> B
  - 00000010 -> C
  - 00000011 -> D
  - 00000100 -> E
- .... e così via

# Codifica dei Caratteri – 4/5

*American Standard Code for Information Interchange – ASCII*  
(Codice Standard Americano per lo Scambio di Informazioni) è un codice standard per la codifica dei caratteri

## ASCII Code: Character to Binary

0	0011 0000	o	0100 1111	m	0110 1101
1	0011 0001	p	0101 0000	n	0110 1110
2	0011 0010	q	0101 0001	o	0110 1111
3	0011 0011	r	0101 0010	p	0111 0000
4	0011 0100	s	0101 0011	q	0111 0001
5	0011 0101	t	0101 0100	r	0111 0010
6	0011 0110	u	0101 0101	s	0111 0011
7	0011 0111	v	0101 0110	t	0111 0100
8	0011 1000	w	0101 0111	u	0111 0101
9	0011 1001	x	0101 1000	v	0111 0110
A	0100 0001	y	0101 1001	w	0111 0111
B	0100 0010	z	0101 1010	x	0111 1000
C	0100 0011	a	0110 0001	y	0111 1001
D	0100 0100	b	0110 0010	z	0111 1010
E	0100 0101	c	0110 0011	.	0010 1110
F	0100 0110	d	0110 0100	,	0010 0111
G	0100 0111	e	0110 0101	:	0011 1010
H	0100 1000	f	0110 0110	;	0011 1011
I	0100 1001	g	0110 0111	?	0011 1111
J	0100 1010	h	0110 1000	!	0010 0001
K	0100 1011	i	0110 1001	'	0010 1100
L	0100 1100	j	0110 1010	"	0010 0010
M	0100 1101	k	0110 1011	(	0010 1000
N	0100 1110	l	0110 1100	)	0010 1001
				space	0010 0000

# Codifica dei Caratteri - 5/5

---

- **Soluzione:** una parola (o più parole) sarà rappresentata dal computer come una successione di gruppi di 8 bit

O	G	G	I		P	I	O	V	E
01001111	01000111	01000111	01001001	00100000	01010000	01001001	01001111	01010110	01000101

# Sistemi di Numerazione Posizionale – 1/5

---

- Il nostro sistema di **numerazione**
  - Utilizza una notazione **posizionale** ed è in **base 10**
  - L'alfabeto utilizzato è l'insieme dei simboli  $\{0, 1, 2, \dots, 9\}$
- Essendo posizionale, il valore di una “sequenza” di simboli viene calcolata assegnando dei “pesi” ad ogni simbolo a seconda della sua posizione

Posizioni → 3 2 1 0      Base  
Stringa di simboli → 4523<sub>10</sub> =

$$4 \times 10^3 + 5 \times 10^2 + 2 \times 10^1 + 3 \times 10^0$$

migliaia      centinaia      decine      unità

# Sistemi di Numerazione Posizionale – 2/5

---

- **3251**

- **3** unità di migliaia
- **2** centinaia
- **5** decine
- **1** unità

- **745814763**

- **7** centinaia di milioni
- **4** decine di milioni
- **5** unità di milioni
- **8** centinaia di migliaia
- **1** decina di migliaia
- **4** unità di migliaia
- **7** centinaia
- **6** decine
- **3** unità

# Sistemi di Numerazione Posizionale – 3/5

- La stringa di simboli  $v$  può essere rappresentata come

$$v = d_n \times B^{n-1} + d_{n-1} \times B^{n-2} + \dots + d_2 \times B^1 + d_1 \times B^0$$

$B$  è la base del numero

$0 \leq j \leq n - 1$   
è la posizione (o peso) del numero

$n$  è il numero di simboli presenti in  $v$

$d$  è la cifra alla  $j$ -esima posizione nel numero

642 in base 10 può essere scritto come  $6_3 \times 10^2 + 4_2 \times 10^1 + 2_1 \times 10^0$

# Sistemi di Numerazione Posizionale – 4/5

---

- Concetto di **base di rappresentazione B**
- Rappresentazione del **numero** come **sequenza di simboli**, detti **cifre**
  - Appartenenti ad un alfabeto composto da **B** simboli distinti
    - Ogni simbolo rappresenta un valore fra 0 e **B-1**
- Il valore di un numero *v* espresso in questa notazione è ricavabile
  - A partire dal valore rappresentato da ogni simbolo
  - Pesato in base alla posizione che occupa nella sequenza

# Sistemi di Numerazione Posizionale – 5/5

---

- Formalmente, il valore di un numero  $v$  espresso in questa notazione è dato dalla formula

$$v = \sum_{k=0}^{n-1} d_k B^k$$

- Dove  $B$  è la base
- $d_k$  ( $0 \leq k \leq n - 1$ ) sono le cifre (comprese tra 0 e  $B - 1$ )
- **Osservazione:** una sequenza di cifre non è interpretabile se non si precisa la base in cui è espressa

# Sistemi di Numerazione più Diffusi

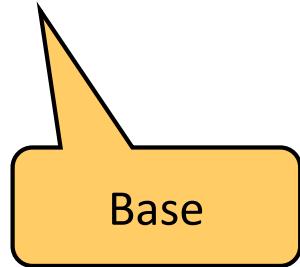
---

<b>Sistema</b>	<b>Base</b>	<b>Simboli</b>	<b>Usato dagli umani?</b>	<b>Usato dai computer?</b>
Decimale	10	0, 1, ..., 9	Si	No
Binario	2	0, 1	No	Si
Ottale	8	0, 1, ..., 7	No	No
Esadecimale	16	0, 1, ..., 9, A, B, ... F	No	No

# Esempio

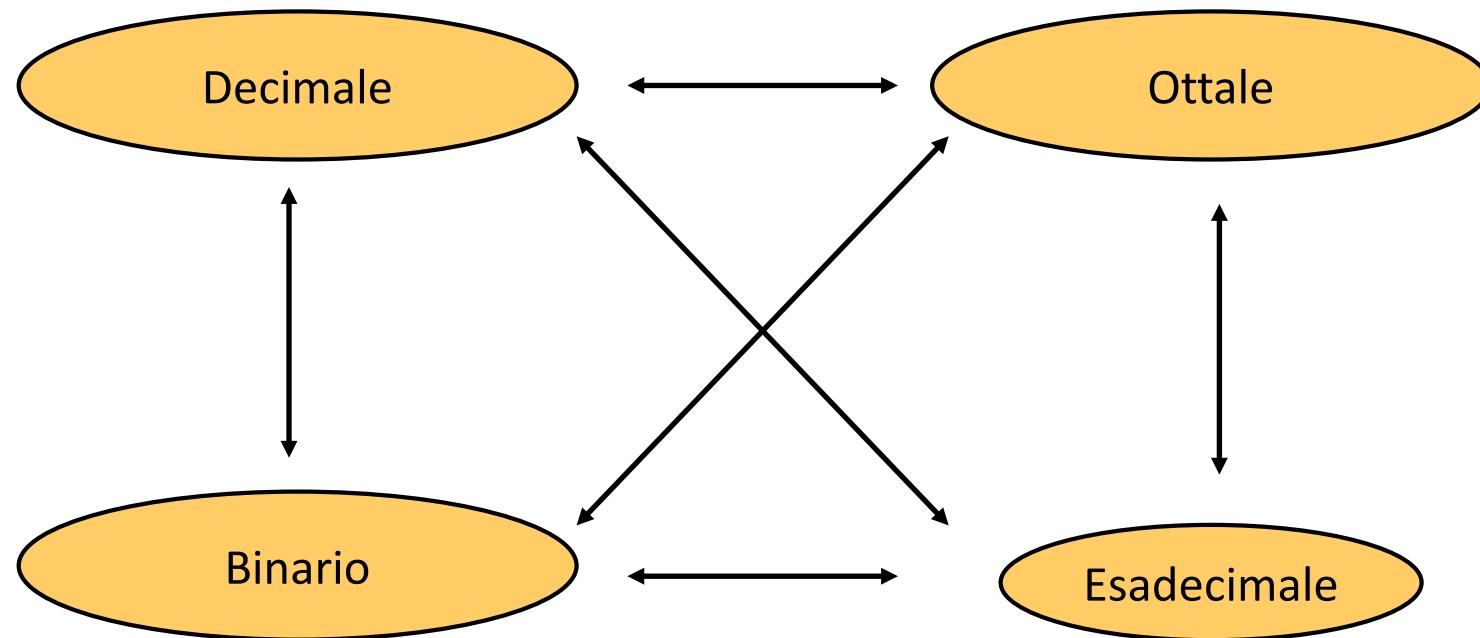
---

$$25_{10} = 11001_2 = 31_8 = 19_{16}$$



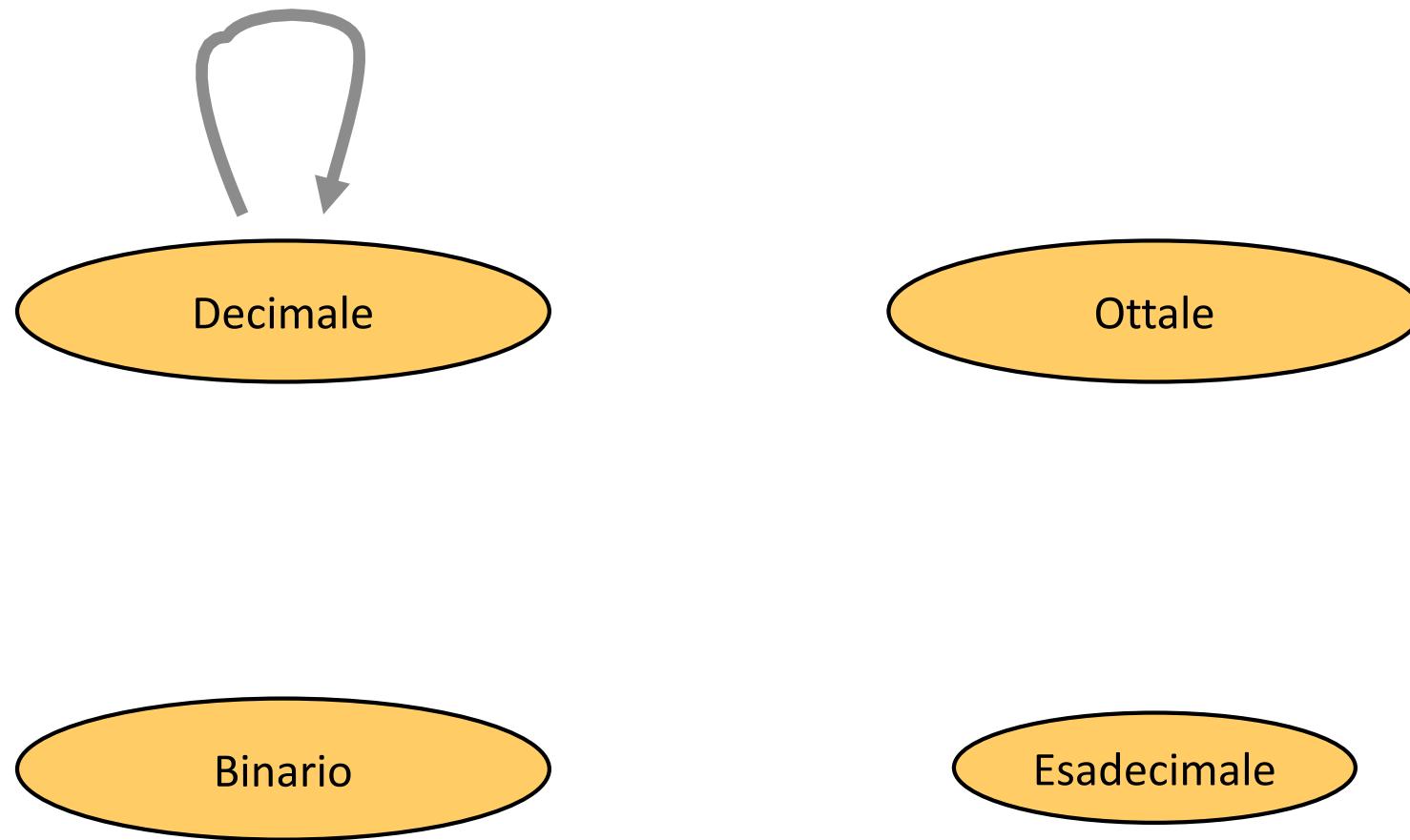
# Conversioni tra Basi (più Diffuse)

- Le possibilità



# Da Decimale a Decimale

---



# Da Decimale a Decimale

---

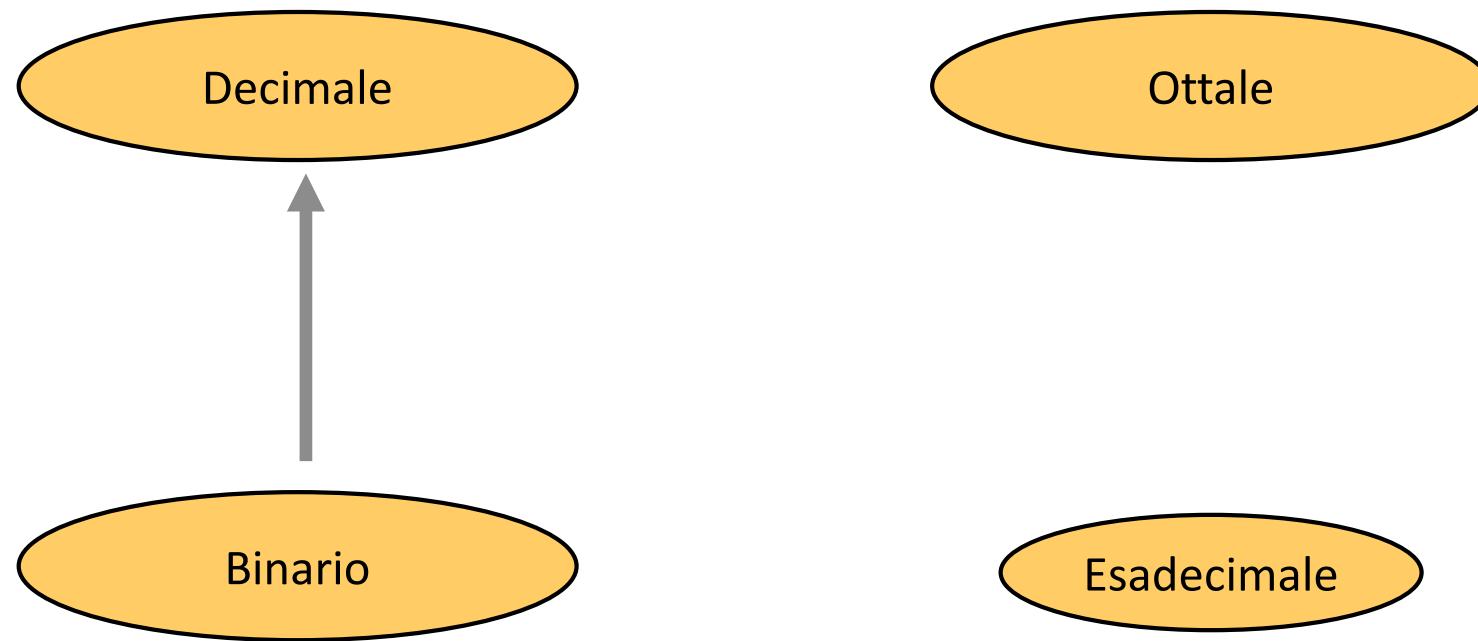
$$\begin{array}{rcl} 125_{10} \Rightarrow & 5 \times 10^0 = & 5 + \\ & 2 \times 10^1 = & 20 + \\ & 1 \times 10^2 = & \underline{100} \\ & & 125 \end{array}$$

Peso

Base

# Da Binario a Decimale

---



# Da Binario a Decimale: Tecnica

- Moltiplica ciascun bit per  $2^n$ , dove  $n$  è il “peso” del bit
  - Il peso è dato dalla posizione del bit, a partire da 0 sulla destra
- Somma i risultati


$$\begin{array}{r} \textcolor{red}{101011}_2 \Rightarrow \\ \text{Bit in} \\ \text{posizione "0"} \\ \hline \end{array} \begin{array}{rcl} \textcolor{red}{1} & \times & \textcolor{green}{2^0} = & 1 & + \\ \textcolor{red}{1} & \times & \textcolor{green}{2^1} = & 2 & + \\ \textcolor{red}{0} & \times & \textcolor{green}{2^2} = & 0 & + \\ \textcolor{red}{1} & \times & \textcolor{green}{2^3} = & 8 & + \\ \textcolor{red}{0} & \times & \textcolor{green}{2^4} = & 0 & + \\ \textcolor{red}{1} & \times & \textcolor{green}{2^5} = & 32 & \\ \hline & & & & \textcolor{red}{43}_{10} \end{array}$$

# Da Binario a Decimale: Esempi

$N_2 = 101010$       5 4 3 2 1 0 <- Posizioni

$$N_{10} = 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$$
$$= 32 + 8 + 2 = 42$$

$$N_2 = 11011$$
$$N_{10} = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$
$$= 16 + 8 + 2 + 1 = 27$$

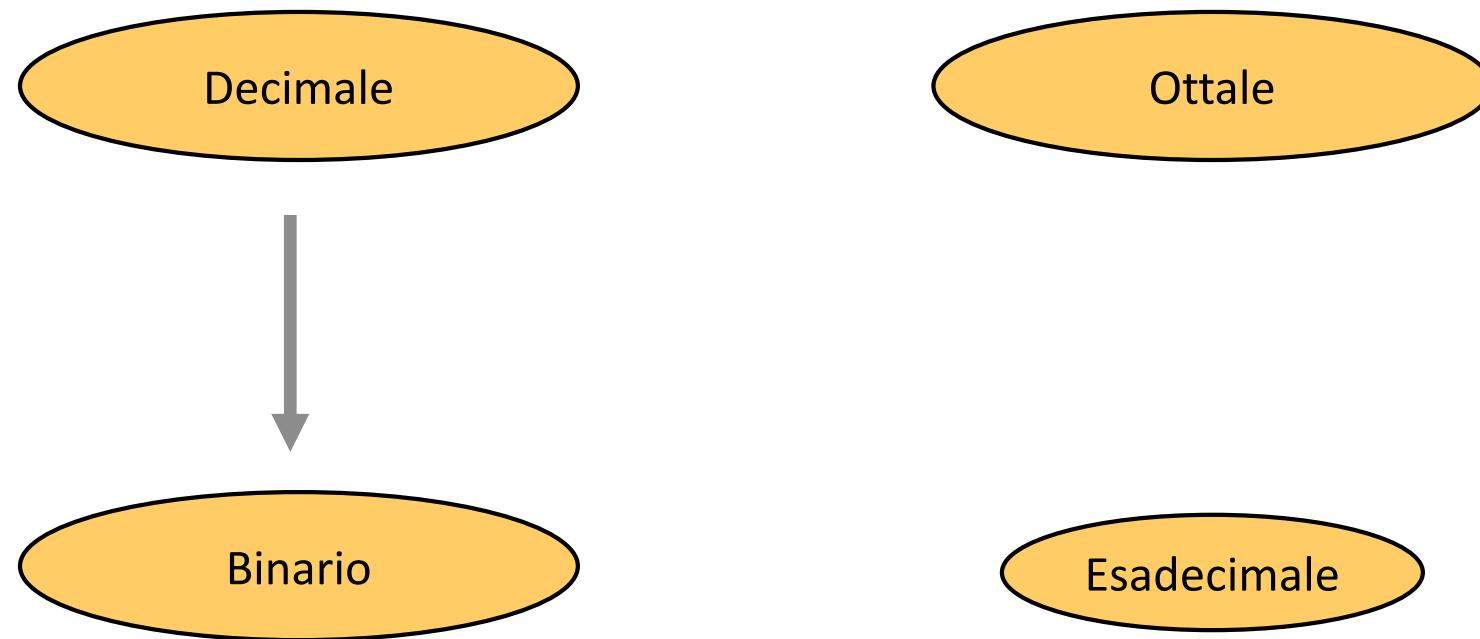
# Da Binario a Decimale: Altri Esempi

---

- $10011010 = 1 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$   
 $= 2^7 + 2^4 + 2^3 + 2^1$   
 $= 128 + 16 + 8 + 2$   
 $= 154$
- $00101001 = 0 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$   
 $= 2^5 + 2^3 + 2^0$   
 $= 32 + 8 + 1$   
 $= 41$

# Da Decimale a Binario

---



# Da Decimale a Binario: Tecnica

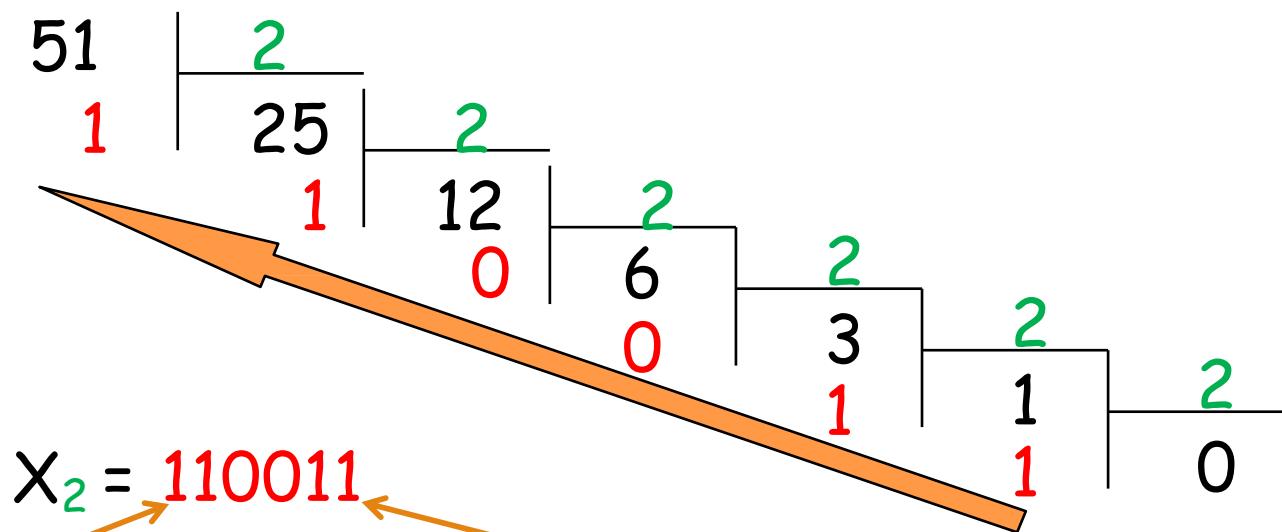
---

- Dividi per due e tieni traccia del resto (**divisione euclidea o divisione con resto**)
- Il **primo resto** è il **bit in posizione 0** (LSB, least-significant bit)
- Il **secondo resto** è il **bit in posizione 1**
- E così via...

# Da Decimale a Binario: Esempio

$51_{10}$

(Da decimale a binario)  
 $X_2 = ???$



$$51_{10} = 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

# Rappresentazione degli Interi: “Modulo e Segno” – 1/7

---

- $N = 0, +1, -1, +2, -2, +3, -3, \dots$
- Come possiamo rappresentare il segno di un numero?
- **Idea:** Aggiungiamo un ulteriore bit che poniamo a
  - **1** se il numero è negativo
  - **0** altrimenti

Esempio

$$N_{10} = +14 \quad N_2 = 01110$$

$$N_{10} = -14 \quad N_2 = 11110$$

# Rappresentazione degli Interi: “Modulo e Segno” – 2/7

---

- **Alfabeto binario**
  - Anche il segno è rappresentato da 0 o 1
  - Indispensabile indicare il numero **k** di bit utilizzati
- **Modulo e segno** (rappresentazione con **k** bit)
  - **1 bit di segno (0 positivo, 1 negativo)**
  - Il primo bit è detto **bit più significativo** (o Most Significant Bit - **MSB**)
  - **k-1 bit di modulo**

# Rappresentazione degli Interi: “Modulo e Segno” – 3/7

---

- **Modulo e segno** (rappresentazione con  $k$  bit)
  - 1 bit di segno (**0 positivo, 1 negativo**)
  - Il primo bit è detto **bit più significativo** (o Most Significant Bit - **MSB**)
  - $k-1$  bit di modulo

## *Esempio*

- $k = 4$

1	1	1	0
---	---	---	---

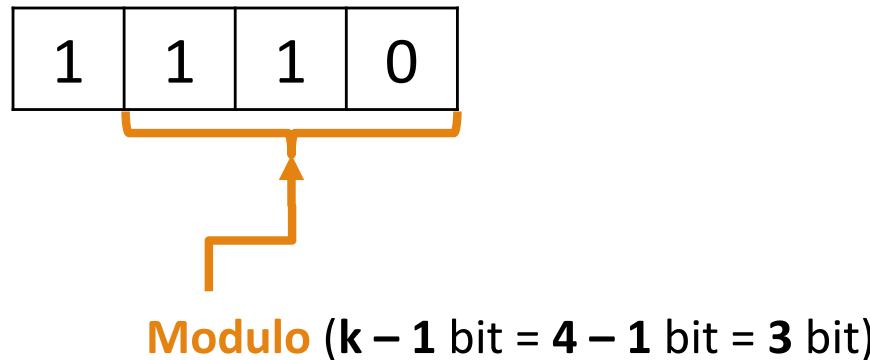
# Rappresentazione degli Interi: “Modulo e Segno” – 4/7

---

- **Modulo e segno** (rappresentazione con  $k$  bit)
  - 1 bit di segno (**0 positivo, 1 negativo**)
  - Il primo bit è detto **bit più significativo** (o Most Significant Bit - **MSB**)
  - $k-1$  bit di modulo

## *Esempio*

- $k = 4$

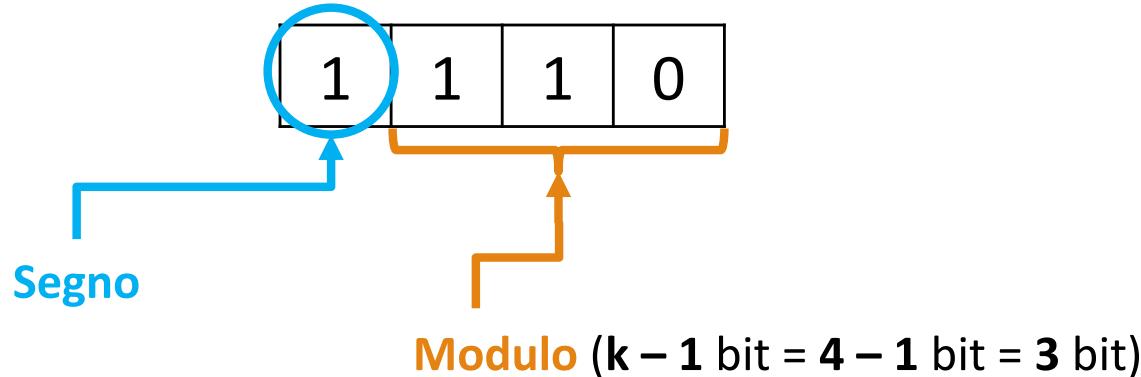


# Rappresentazione degli Interi: “Modulo e Segno” – 5/7

- **Modulo e segno** (rappresentazione con  $k$  bit)
  - 1 bit di segno (**0 positivo, 1 negativo**)
  - Il primo bit è detto **bit più significativo** (o Most Significant Bit - **MSB**)
  - $k-1$  bit di modulo

## *Esempio*

- $k = 4$

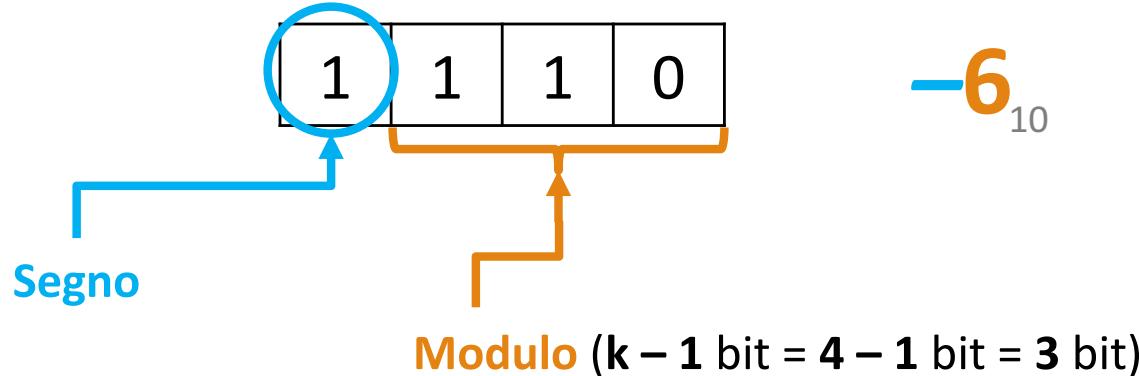


# Rappresentazione degli Interi: “Modulo e Segno” – 6/7

- **Modulo e segno** (rappresentazione con  $k$  bit)
  - 1 bit di segno (**0 positivo, 1 negativo**)
  - Il primo bit è detto **bit più significativo** (o Most Significant Bit - **MSB**)
  - $k-1$  bit di modulo

## Esempio

- $k = 4$



# Rappresentazione degli Interi: “Modulo e Segno” – 7/7

---

- Con **k bit** è possibile rappresentare i valori da  $-2^{k-1}+1$  a  $+2^{k-1}-1$ 
  - *Esempi*
    - **4 bit** → valori che vanno da **-7** a **+7**
    - **8 bit** → valori che vanno da **-127** a **+127**
- **Osservazione**
  - Due possibili rappresentazioni dello 0
    - Con **4 bit** sono  $+0_{10} = 0000_{ms}$        $-0_{10} = 1000_{ms}$

# Numeri Interi in Complemento a Due – 1/5

---

- **Idea:** l'interpretazione posizionale viene mantenuta e si modifica soltanto il peso del bit più significativo, invertendolo
- **Caratteristiche**
  - Lo zero ha una rappresentazione unica
  - Tutti i numeri che hanno il bit più significativo uguale a 1 sono negativi (come prima)
  - È sempre necessario specificare il numero di bit  $k$  che si vuole utilizzare per rappresentare un determinato numero
  - Si rappresentano i **valori da  $-2^{k-1}$  a  $+2^{k-1}-1$** 
    - Con **4 bit** i valori vanno da **-8** a **+7**
    - Con **8 bit** i valori vanno da **-128** a **+127**

# Numeri Interi in Complemento a Due – 2/5

---

- Consideriamo un generico numero binario su 8 bit
- Per stabilire la codifica di un generico numero negativo  $n < 0$ , sapendo che necessariamente il bit più significativo va posto a 1, è sufficiente riportare nei restanti bit il numero positivo che, sommato a  $-2^7$ , dà il valore  $n$
- Per esempio, proviamo a codificare  $-37$ . Essendo un numero negativo, il bit più significativo vale 1:

$-2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
1	?	?	?	?	?	?	?

- Nella parte restante della tabella dovremo inserire quel numero che sommato a  $-128$  da  $-37$ 
  - $-128 + x = -37 \Rightarrow x = 128 - 37 = 91$

# Numeri Interi in Complemento a Due – 3/5

---

- La codifica binaria di 91 è 1011011, che riportato nella tabella precedente fornisce la codifica desiderata:

$-2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
1	1	0	1	1	0	1	1

# Numeri Interi in Complemento a Due – 4/5

- Un metodo molto comodo per calcolare la rappresentazione di  $-X$  a partire da quella di  $+X$  è il seguente
  - Idea: effettuare il complemento di ogni bit di  $X$ , poi aggiungere 1
    1. Codifica binaria di  $+6_{10} \Rightarrow 0110_2$  (N.B. ci vogliono 4 bit)
    2. Complemento di tutti i bit  $\Rightarrow 1001_{C2}$  (corrisponderebbe a  $-7_{10}$ )
    3. Aggiungere 1  $\Rightarrow 1010_{C2}$  (che corrisponde a  $-6_{10}$ )  
$$-2^3 + 2^1 = -8 + 2 = -6$$

Il complemento di 1 è 0

Il complemento di 0 è 1

$$\begin{array}{r} 1001 \\ + 1 \\ \hline 1010 \end{array}$$

1 + 1 = 0 col riporto di 1

# Numeri Interi in Complemento a Due – 5/5

---

- **Estensione del “segno”**
  - I valori positivi iniziano con 0, quelli negativi con 1
  - Data la rappresentazione di un numero su  $k$  bit, la rappresentazione dello stesso numero su  $k+1$  bit si ottiene aggiungendo (a sinistra) un bit uguale al primo
- **Esempi**
  - Rappresentazione di -6 su 4 bit = 1010
  - Rappresentazione di -6 su 5 bit = 11010
  - Rappresentazione di -6 su 8 bit = 11111010

# Esercizi 1

---

- Scrivere in binario semplice i seguenti numeri in base 10
  - 53<sub>10</sub>
  - 21110
- Scrivere in binario semplice su 7 bit il numero 13<sub>10</sub>
- Scrivere in modulo e segno su 7 bit il numero 13<sub>10</sub>
- Scrivere in modulo e segno su 7 bit il numero -13<sub>10</sub>
- Scrivere in modulo e segno su 5 bit il numero 17<sub>10</sub>

# Soluzione Esercizi

---

- Scrivere in binario semplice su 7 bit il numero  $13_{10}$ 
  - 0001101
- Scrivere in modulo e segno su 7 bit il numero  $13_{10}$ 
  - 0001101
- Scrivere in modulo e segno su 7 bit il numero  $-13_{10}$ 
  - 1001101
- Scrivere in modulo e segno su 5 bit il numero  $17_{10}$ 
  - 10001, In modulo e segno è  $-1_{10}$
  - RISPOSTA: Non è possibile. Ho bisogno di almeno 6 bit (010001)

# Esercizi svolti

---

- $53 = 32 + 16 + 4 + 1$   
 $= 2^5 + 2^4 + 2^2 + 2^0$   
 $= 1*2^5 + 1*2^4 + 0*2^3 + 1*2^2 + 0*2^1 + 1*2^0$   
 $= 110101$  in binario
- $211 = 128 + 64 + 16 + 2 + 1$   
 $= 2^7 + 2^6 + 2^4 + 2^1 + 2^0$   
 $= 1*2^7 + 1*2^6 + 0*2^5 + 1*2^4 + 0*2^3 + 0*2^2 +$   
 $1*2^1 + 1*2^0$   
 $= 11010011$  in binario

# Esercizi 2

---

- Scrivere in binario semplice su 7 bit il numero  $11_{10}$
- Scrivere in modulo e segno su 8 bit il numero  $25_{10}$
- Scrivere in modulo e segno su 7 bit il numero  $-12_{10}$
- Scrivere in modulo e segno su 5 bit il numero  $20_{10}$

# Indovinello: come conta ET?

---

- Un Extra-Terrestre viene sulla Terra e ci dice che i re di Roma sono 13. Quante dita ha l'Extra-Terrestre?
  - Il 13 deve essere interpretato come una stringa di simboli
  - Non conosciamo la base della loro numerazione
  - Sappiamo che il loro sistema di numerazione è POSIZIONALE
  - Sappiamo che in decimale i re di Roma sono 7
- E se dicesse che i re di Roma sono 111?

# Indovinello: come conta ET?

- Un Extra-Terrestre viene sulla Terra e ci dice che i re di Roma sono 13. Quante dita ha l'Extra-Terrestre?
  - Il 13 deve essere interpretato come una stringa di simboli
  - Non conosciamo la base della loro numerazione
  - Sappiamo che il loro sistema di numerazione è POSIZIONALE
  - Sappiamo che in decimale i re di Roma sono 7

$$13_x = 1 * X^1 + 3 * X^0 = X + 3 = 7_{10}$$

$$\Rightarrow X = 7 - 3 = 4$$

$\Rightarrow$  L'Extra-Terrestre conta in base 4 per cui (sfruttando l'esperienza del sistema decimale) possiamo dire che ha 4 dita (2 per mano)

$\Rightarrow$  L'Extra-Terrestre usa l'alfabeto  $\{0, 1, 2, 3\}$

# Esercizi su sistemi di numerazione posizionale

---

- I re di Roma sono  $7_{10}$ 
  - Base 10, simboli {0, ..., 9}



$$7_{10} = 7 * 10^0 = 7_{10}$$

- I re di Roma sono  $13_4$ 
  - Base 4, simboli {0, 1, 2, 3}



$$13_4 = 1 * 4^1 + 3 * 4^0 = 7_{10}$$

- I re di Roma sono  $111_2$ 
  - Base 2, simboli {0, 1}

$$\begin{aligned} 111_2 = \\ 1 * 2^2 + 1 * 2^1 + 1 * 2^0 = 7_{10} \end{aligned}$$

# Riferimenti

---

- **Libro di testo**
  - Capitolo 2
    - Paragrafi 2, 2.1 [**NO approfondimento 2.4**], 2.2, 2.3, 2.4, 2.5 [**NO approfondimento 2.5**]