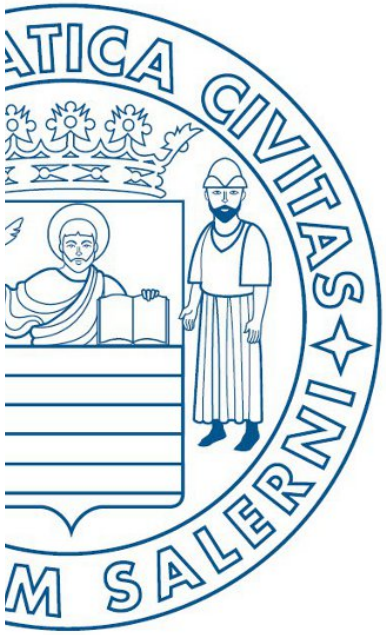


UNIVERSITÀ DEGLI STUDI DI SALERNO

Università di Salerno
Dipartimento di
Ingegneria Industriale

Fondamenti di Informatica

Linguaggi, Codifica e Rappresentazione dell'Informazione

Prof. Christian Esposito

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

A.A. 2017/18

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

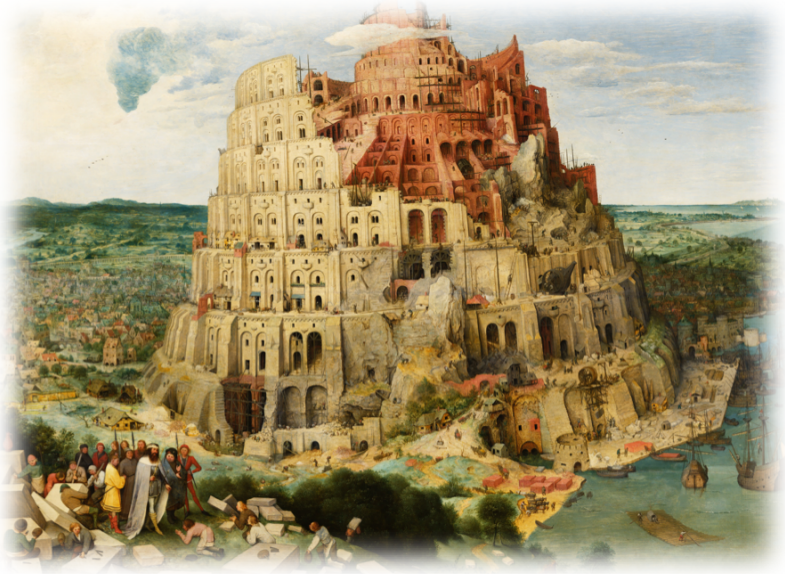
- **Alfabeto**
  - Collezione di simboli grafici, aventi di solito un ordine ben preciso, che servono a rappresentare le parole di una lingua
  - I simboli spesso (ma non sempre) sono lettere ovvero rappresentazione scritta di suoni linguistici
- **Vocabolario (o lessico)**
  - Insieme delle parole ammissibili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

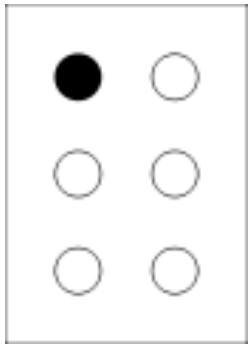
- **Accordo sui simboli**
  - a b c d e f g …
- **Accordo sul lessico**
  - casa, gatto, automobile, vado, …
- **Accordo sulla grammatica**
  - <soggetto verbo complemento>
- **Accordo sulla semantica**
  - La nonna chiude la porta (**OK**)
  - La porta chiude la nonna (**NO**)



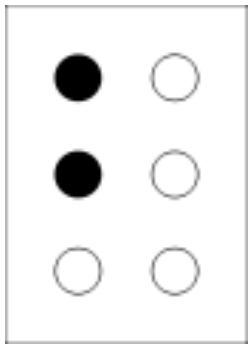
- **Accordo sulla codifica**
  - Regole per trasformare simboli, parole e frasi di un linguaggio in una nuova rappresentazione con possibilità di effettuare in maniera corretta anche l'operazione inversa
    - "a" in codice Morse (Samuel Morse, pittore e storico inglese) è " . – "
    - "b" in codice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

Codice Braille

- Lettera “a”



- Lettera “b”

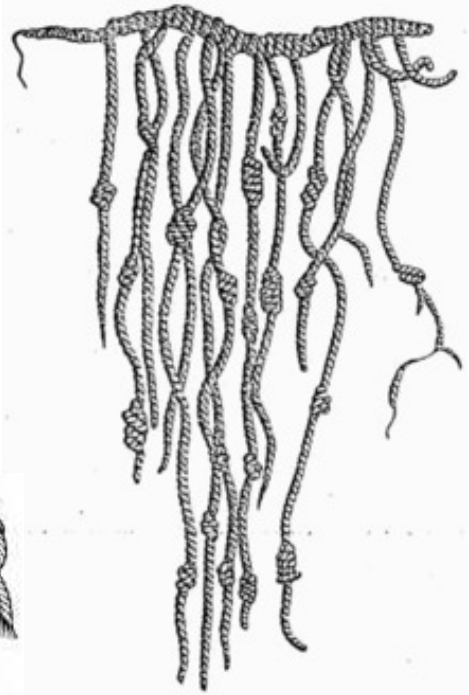


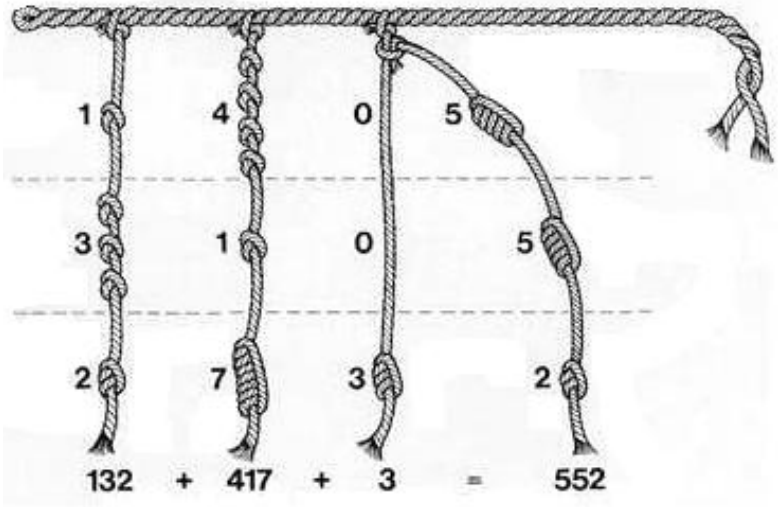
Codifica dei Numeri

- **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

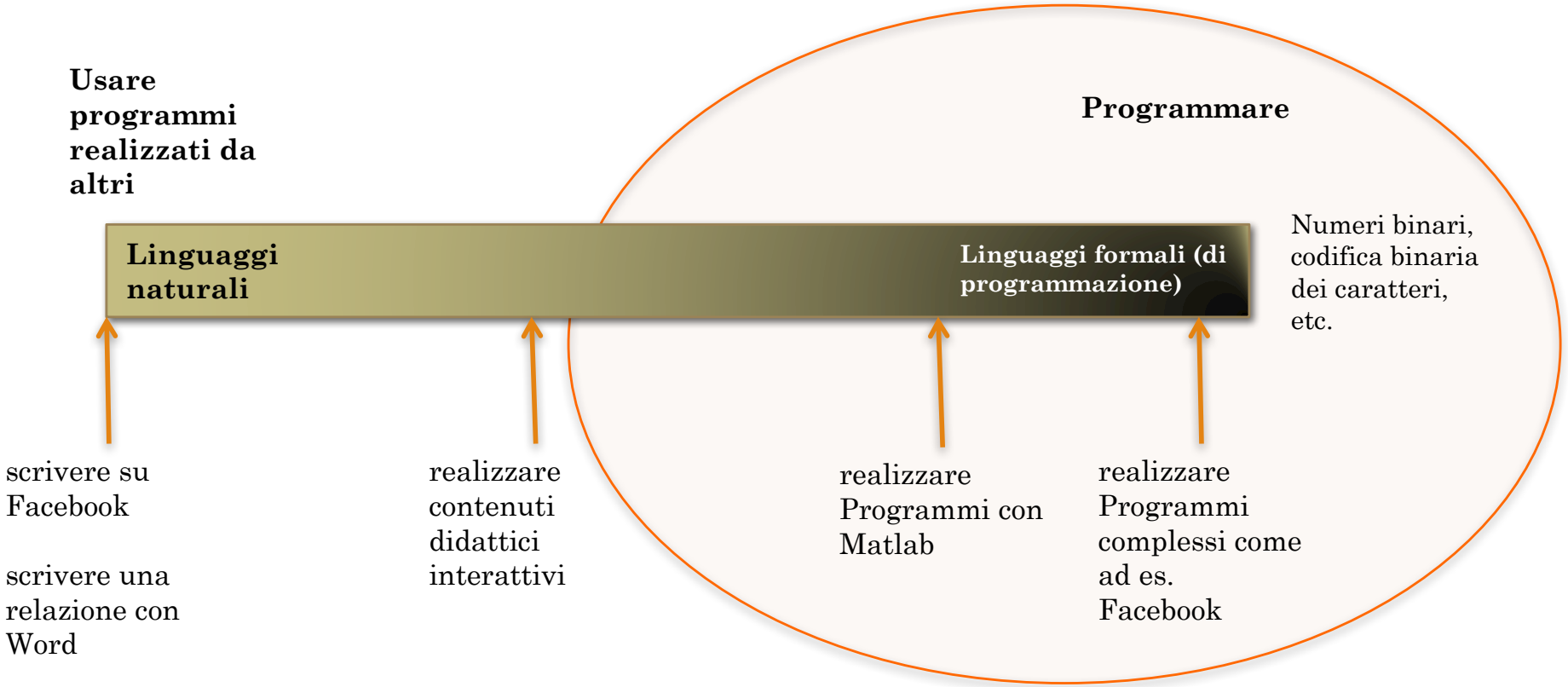
- Per comunicare con un elaboratore, l'**ambiguità** dei **linguaggi naturali** rappresenta un grosso problema
- Bisognerebbe fornire all'elaboratore anche il contesto e la conoscenza (le regole) per disambiguare le frasi
  - La rappresentazione (comprensibile al calcolatore) di contesto e regole è molto 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

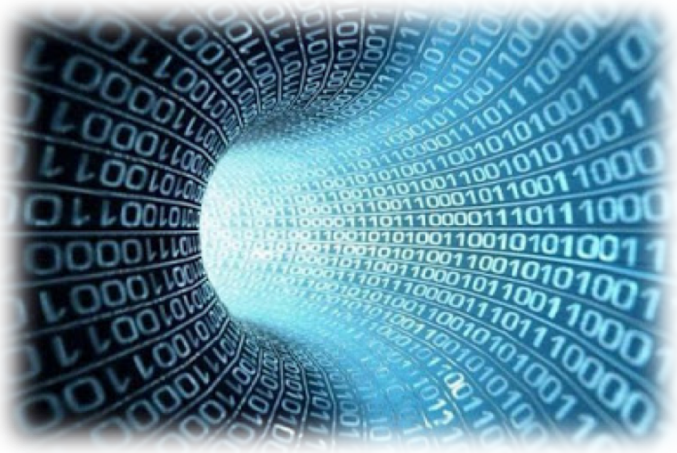


Rappresentazione dell'Informazione: Accordo sui Simboli

- 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
  - Millenovecentosessantatre -> rappresentazione testuale
  - …

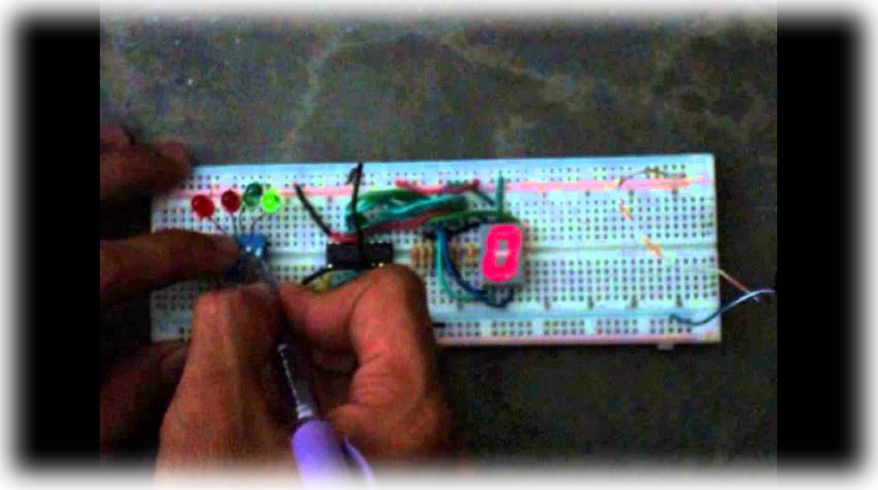
Rappresentazione dell'informazione nei calcolatori – 1/2

- 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

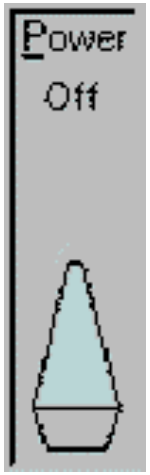
- **Alfabeto binario:** usiamo solo due cifre 0,1 (bit)
- **Problema:** assegnare una sequenza di bit univoca a tutti gli oggetti in un insieme predefinito
- **Quanti oggetti posso rappresentare con *k* bit?**
  - 1 bit => 2 stati (0, 1) => 2 oggetti
  - 2 bit => 4 stati (00, 01, 10, 11) => 4 oggetti
  - 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 $\leq 2^k$ => k $\geq log_2 N$ => 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, …	$32 = 2^5$
…	…	…

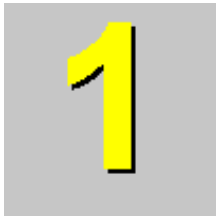
Un Primo Esempio di Codifica: Interruttore

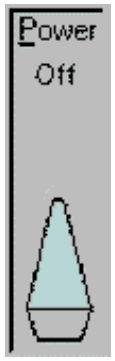
- Due sole possibilità (stati)
  - Spento
  - Acceso
- L'informazione sullo stato dell'interruttore corrisponde dunque alla scelta fra due sole alternative



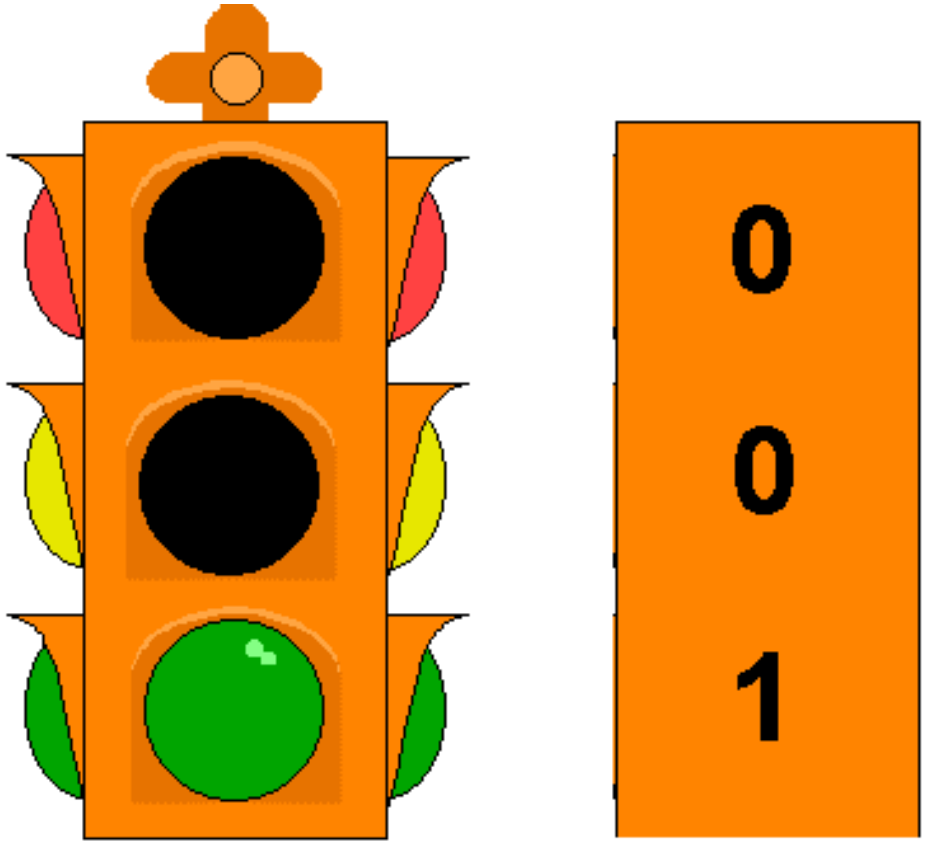
Un Primo Esempio di Codifica: Interruttore

- 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

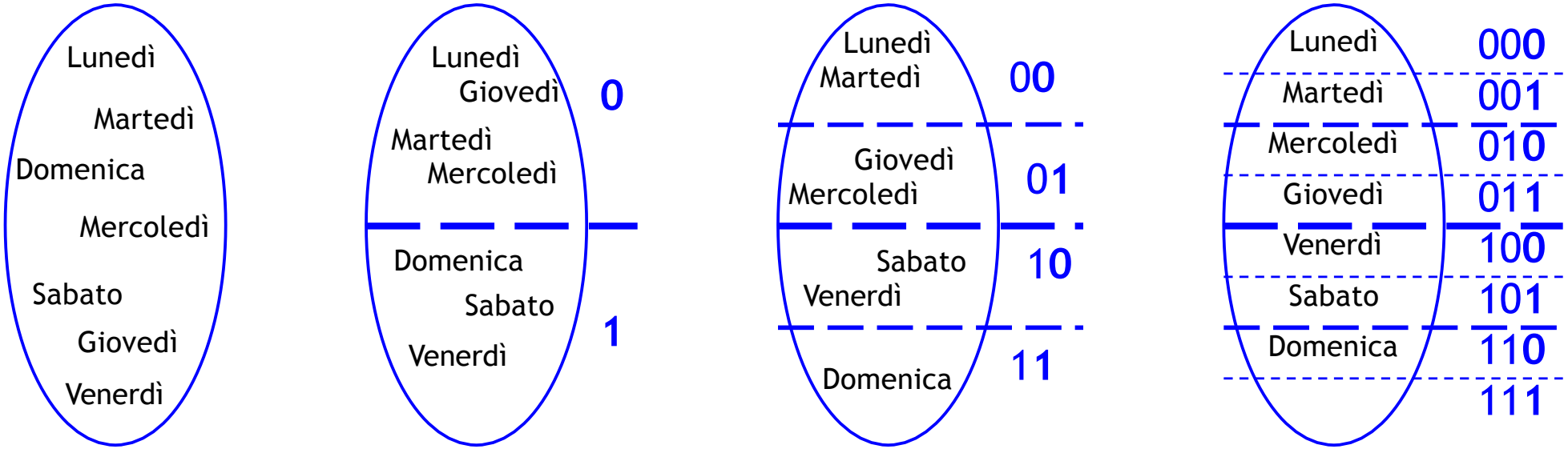
2 bit

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 => $k \geq log_2 7 => k = 3$
- 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

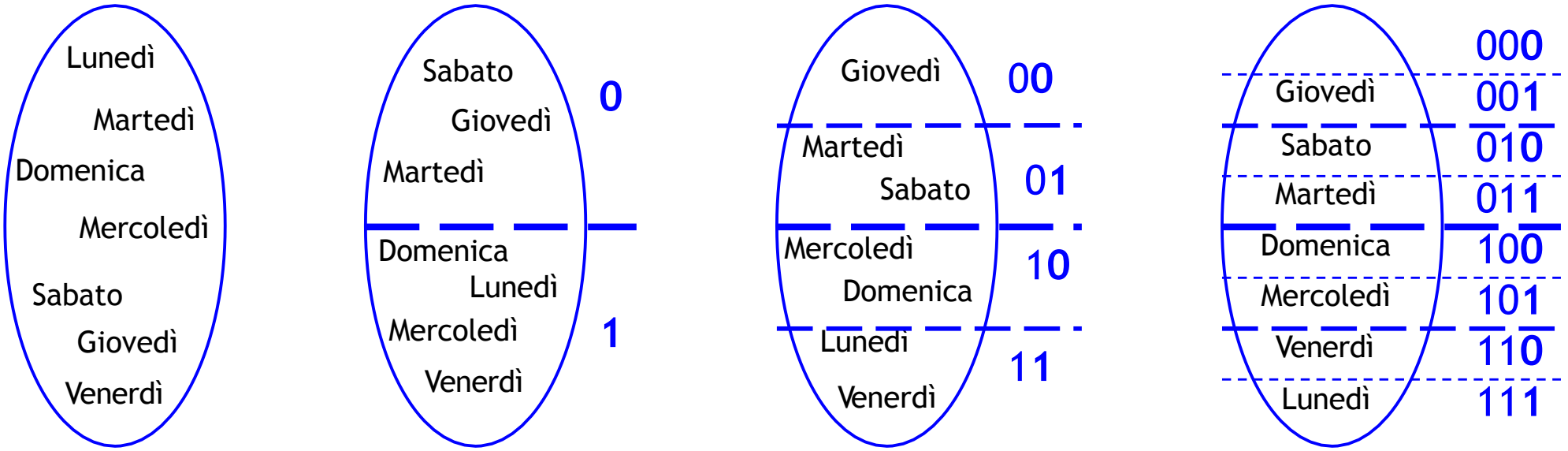


1 bit
2 “gruppi”

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 00000000… a 11111111
  - 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

- Sono stati proposti vari standard per la codifica dei caratteri, tra cui il più diffuso è quello ASCII (American Standard Code for Information Interchange) del 1968, che con 7 bit codifica un alfabeto di 128 caratteri.
- Lo standard che sta prendendo piede e che dovrebbe essere il successore di ASCII è UTF-8 del 1992, codifica principale di Unicode per Internet secondo il W3C.

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

La Codifica dei Numeri

- **Obiettivo**
  - Codifica in binario dei numeri per favorire l'elaborazione da parte dei calcolatori
- **Vincoli**
  - Codifica e decodifica devono essere definite in maniera tale da poter essere compiute in maniera automatica
- **Problema**
  - Deve essere possibile codificare tutti i numeri
    - 0, 1, 2, 3, …
    - -1, -2, -3, …
    - -12.4, -2.004, 0.56, 134.89, …
  - …in sequenze
    - 0000000, 000001, 000010, …

Notazione Posizionale

- 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

Valore di un Numero in Notazione Posizionale

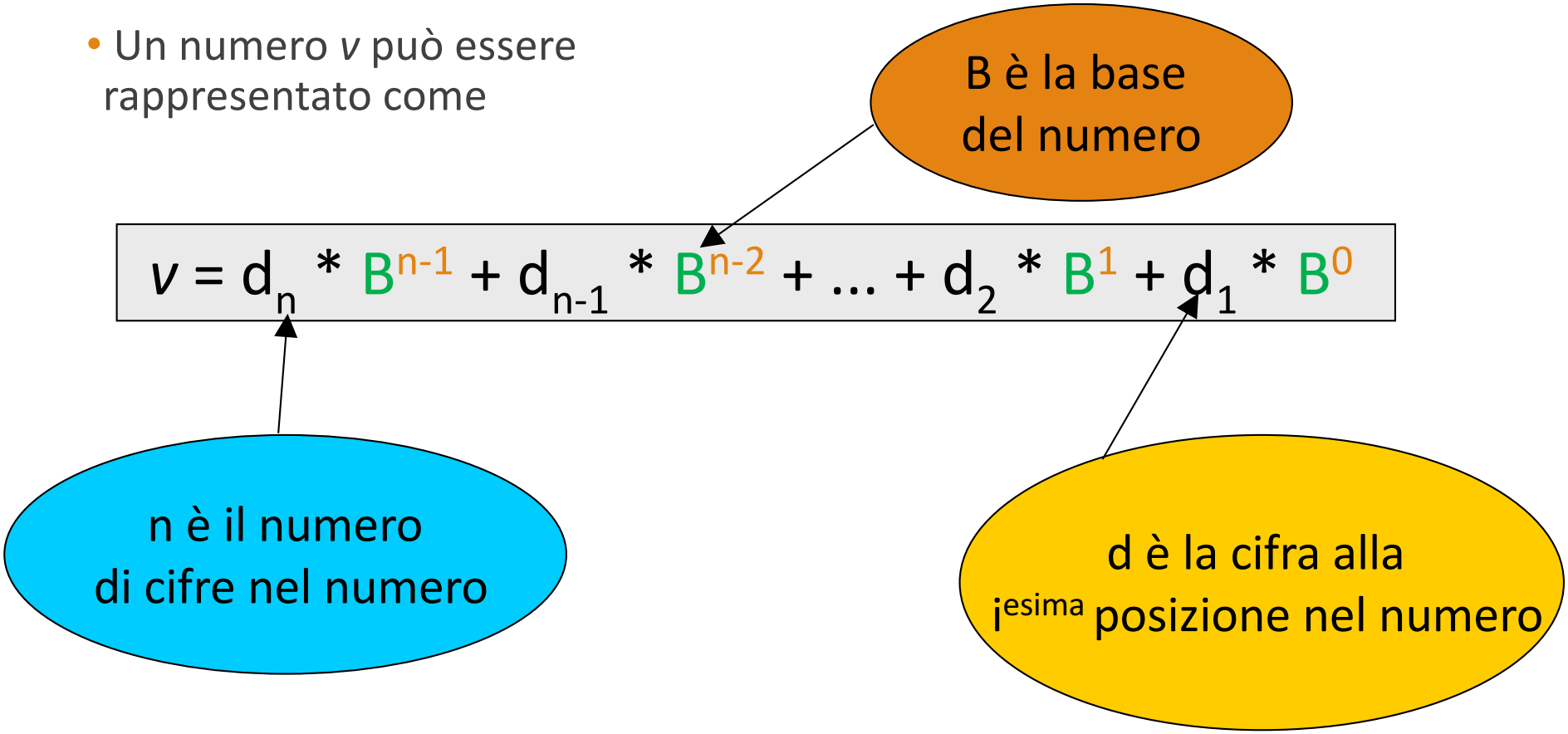
- Formalmente, il valore di un numero *v* espresso in questa notazione è dato dalla formula

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

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

Notazione Posizionale

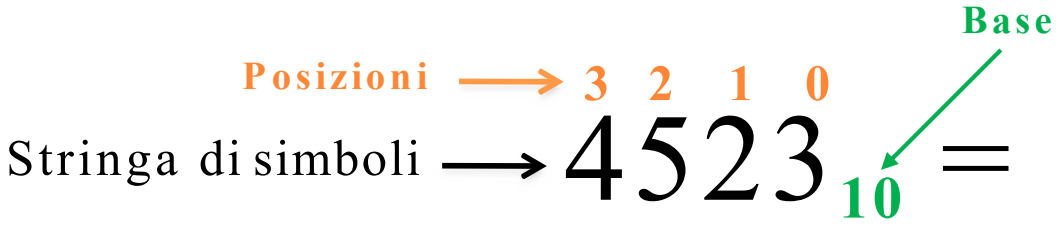
- Un numero *v* può essere rappresentato come



642 può essere scritto come $6_3 * 10^2 + 4_2 * 10^{+} 2_1$

Sistemi di Numerazione Posizionale

- 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, …, 9}
  - Non è l'unico sistema possibile
- Essendo posizionale, il valore di una "sequenza" di simboli viene calcolata assegnando dei "pesi" ad ogni simbolo a seconda della sua posizione



$$4*10^3 + 5*10^2 + 2*10^1 + 3*10^0$$

migliaia centinaia decine unità

Sistemi di Numerazione Posizionale

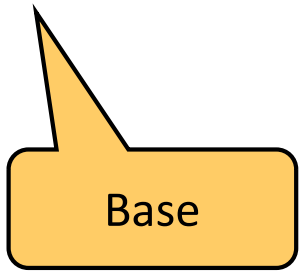
- **3251**
  - **1** unità , **5** decine, **2** centinaia , **3** unità di migliaia
- **745814763**
  - **3** unità, **6** decine, **7** centinaia, **4** unità di migliaia, **1** decina di migliaia, **8** centinaia di migliaia, **5** unità di milioni, **4** decine di milioni, **7** centinaia di milioni

Sistemi di Numerazione più Diffusi

Sistema	Base	Simboli	Usato dagli umani?	Usato dai computer?
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

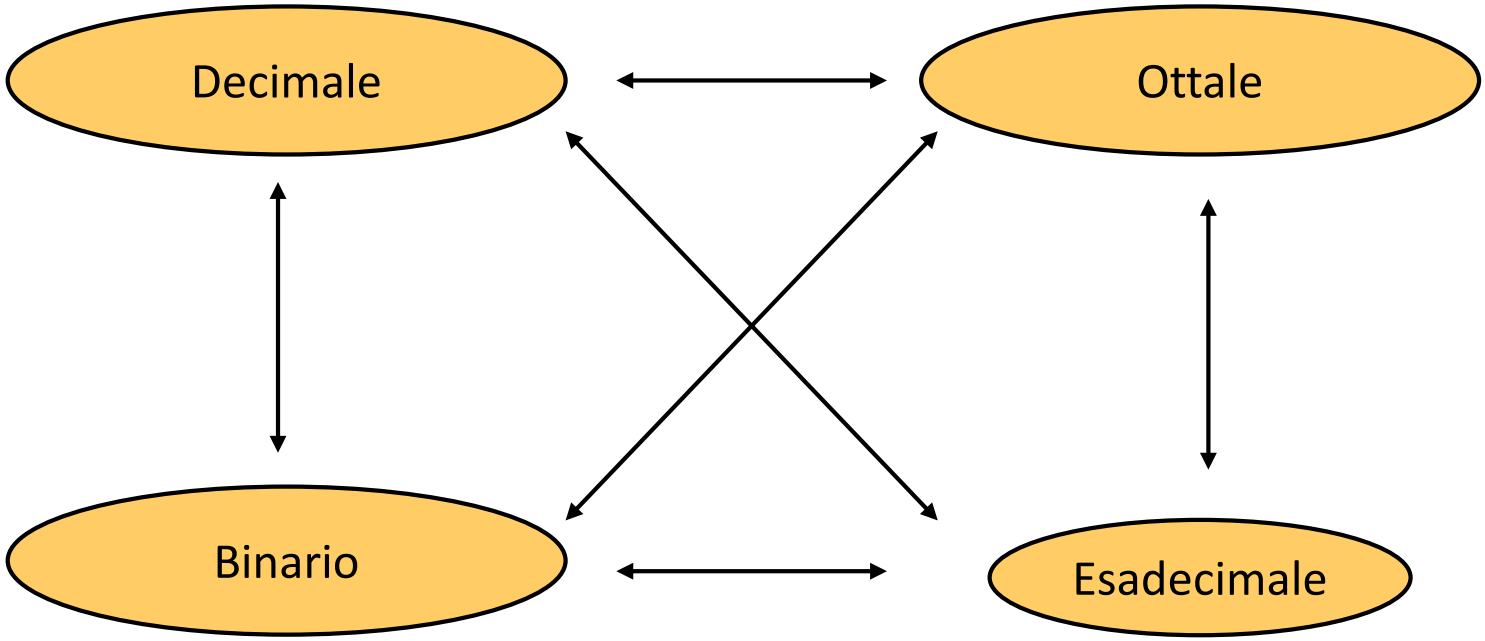
Esempio

$$25_{10} = 11001_{2} = 31_{8} = 19_{16}$$

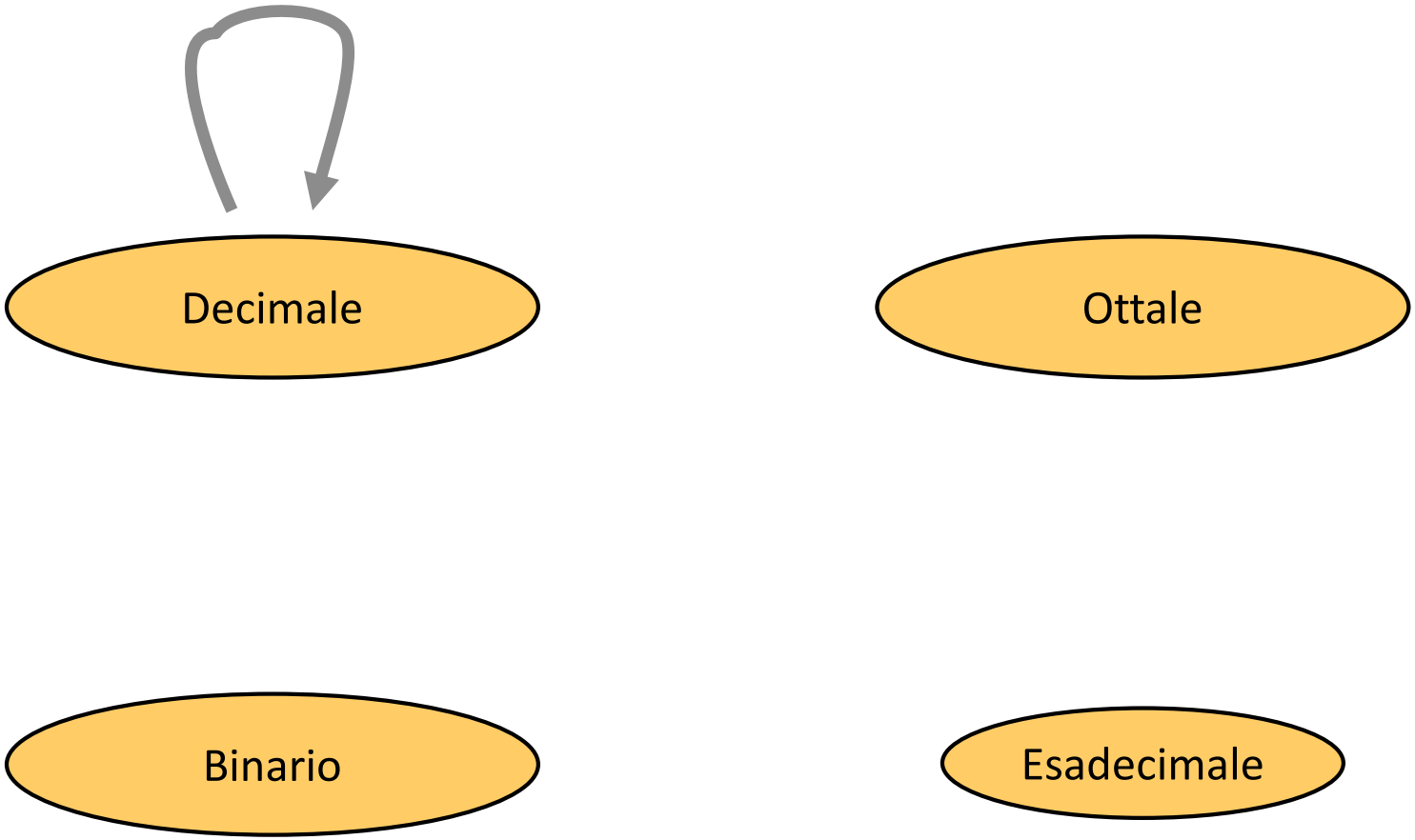


Conversioni tra Basi (più Diffuse)

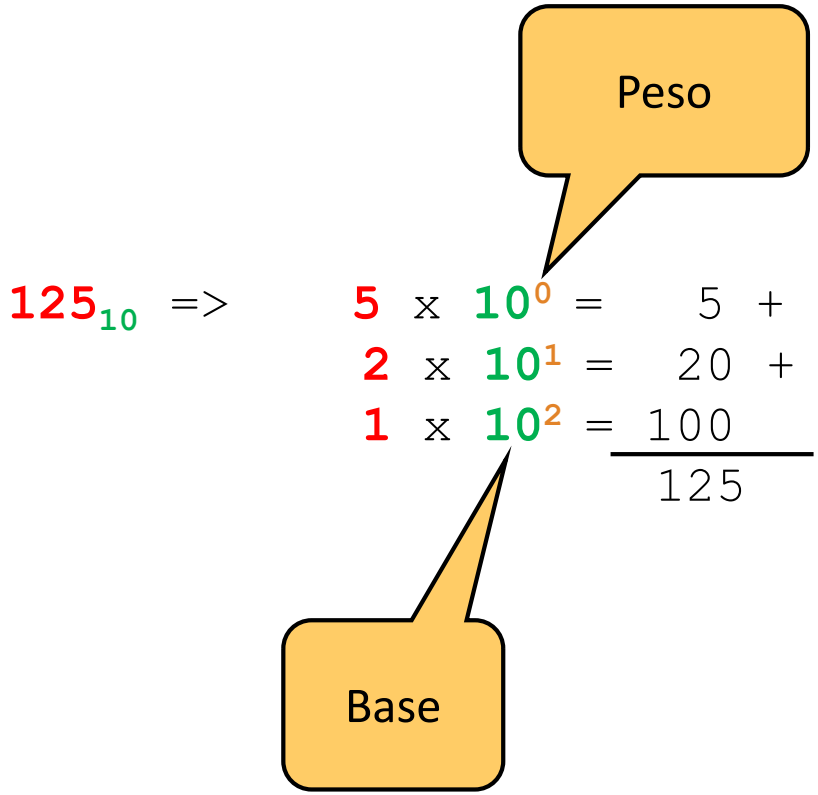
- Le possibilità



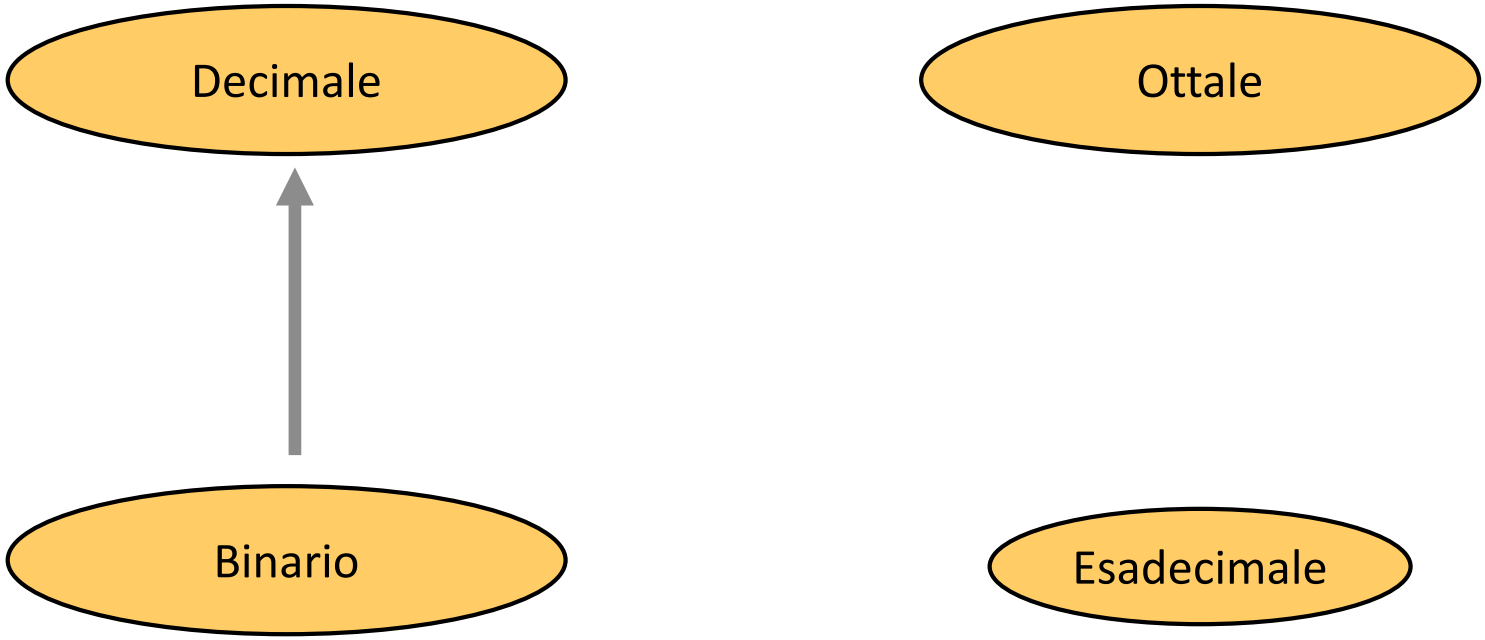
Da decimale a decimale



Da Decimale a Decimale



Da Binario a Decimale

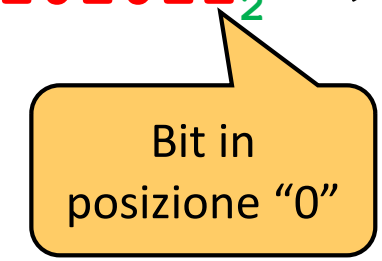


Da Binario a Decimale: Tecnica

- 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

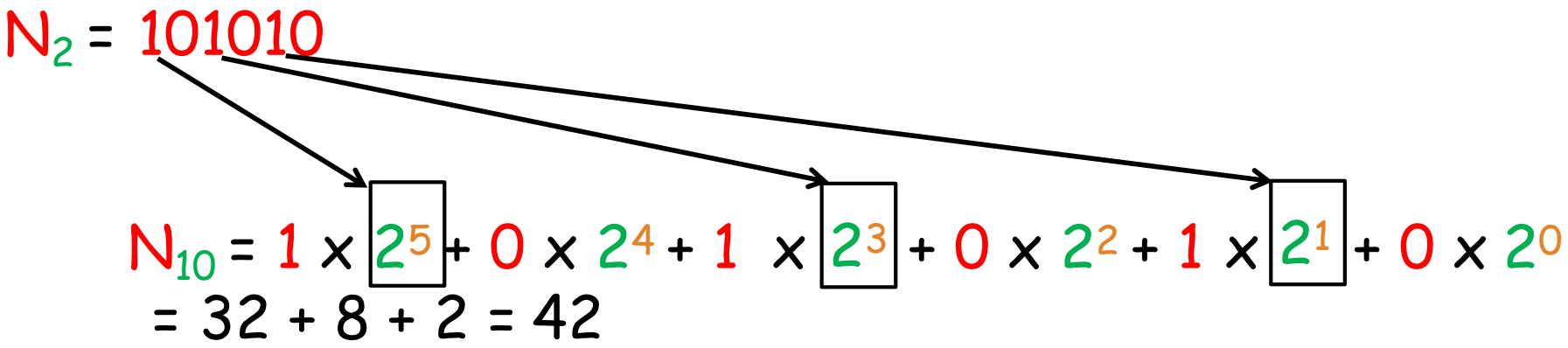
Da Binario a Decimale: Esempio

101011_2 =>



$$
\begin{aligned}
1 \times 2^0 &= 1 + \\
1 \times 2^1 &= 2 + \\
0 \times 2^2 &= 0 + \\
1 \times 2^3 &= 8 + \\
0 \times 2^4 &= 0 + \\
1 \times 2^5 &= 32 \\
\hline
&\quad 43_{10}
\end{aligned}
$$

Da Binario a Decimale: Esempio



$N_2 = 11011$

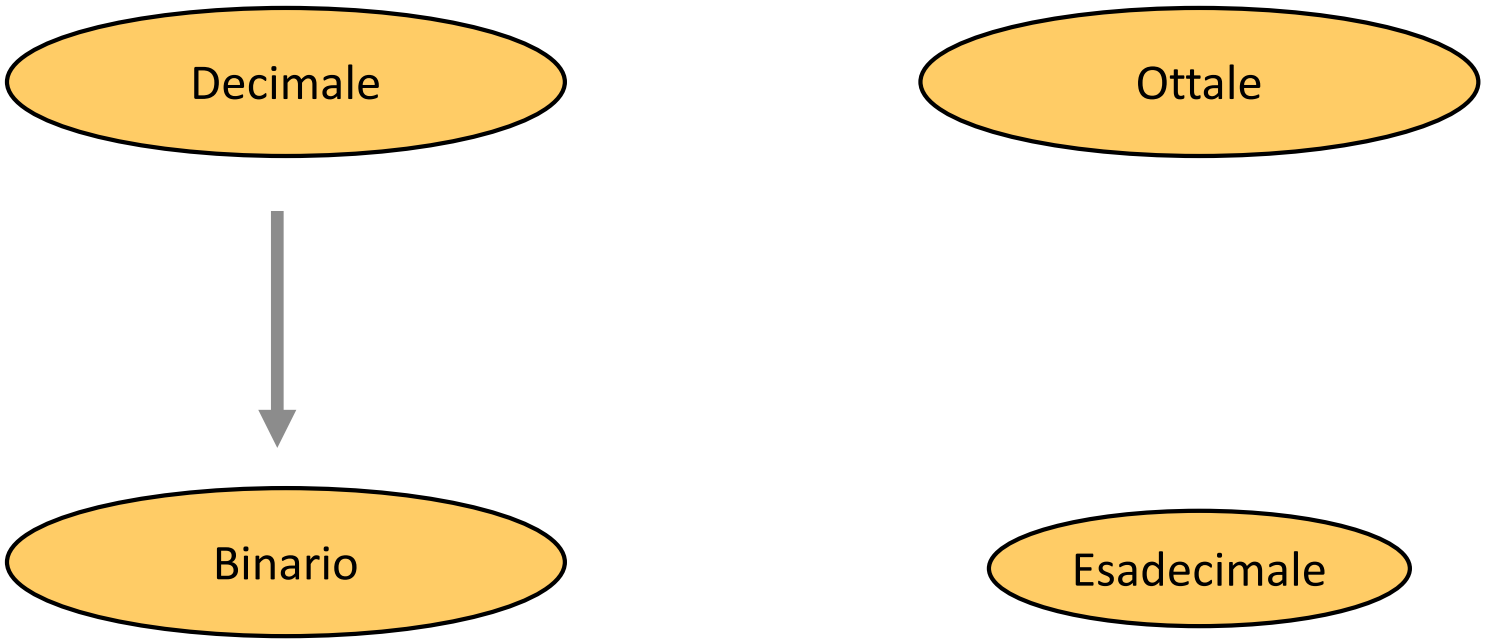
$N_{10} = 2^0 + 2^1 + 2^3 + 2^4 = 1 + 2 + 8 + 16 = 27$

Da Binario a Decimale: Altri Esempi

- $10011010 = 1*2^7 + 0*2^6 + 0*2^5 + 1*2^4 + 1*2^3 + 0*2^2 + 1*2^1 + 0*2^0$
  $= 2^7 + 2^4 + 2^3 + 2^1$
  $= 128 + 16 + 8 + 2$
  $= 154$

- $00101001 = 0*2^7 + 0*2^6 + 1*2^5 + 0*2^4 + 1*2^3 + 0*2^2 + 0*2^1 + 1*2^0$
  $= 2^5 + 2^3 + 2^0$
  $= 32 + 8 + 1$
  $= 41$

Da Decimale a Binario

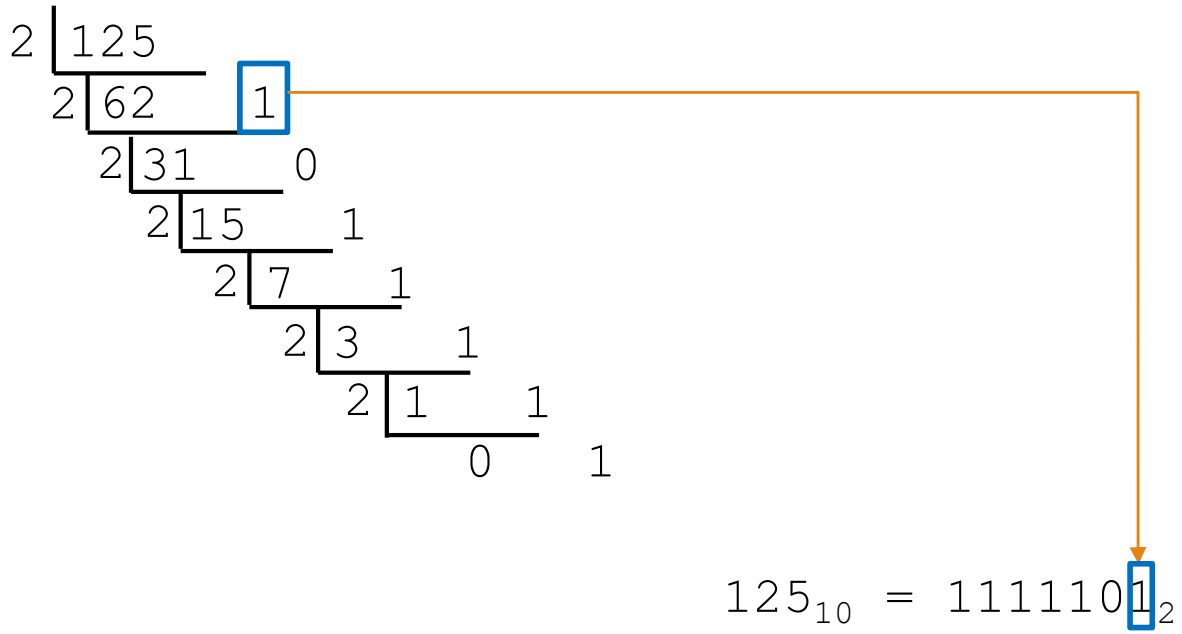


Da Decimale a Binario: Tecnica

- Dividi per due e tieni traccia del resto
- 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

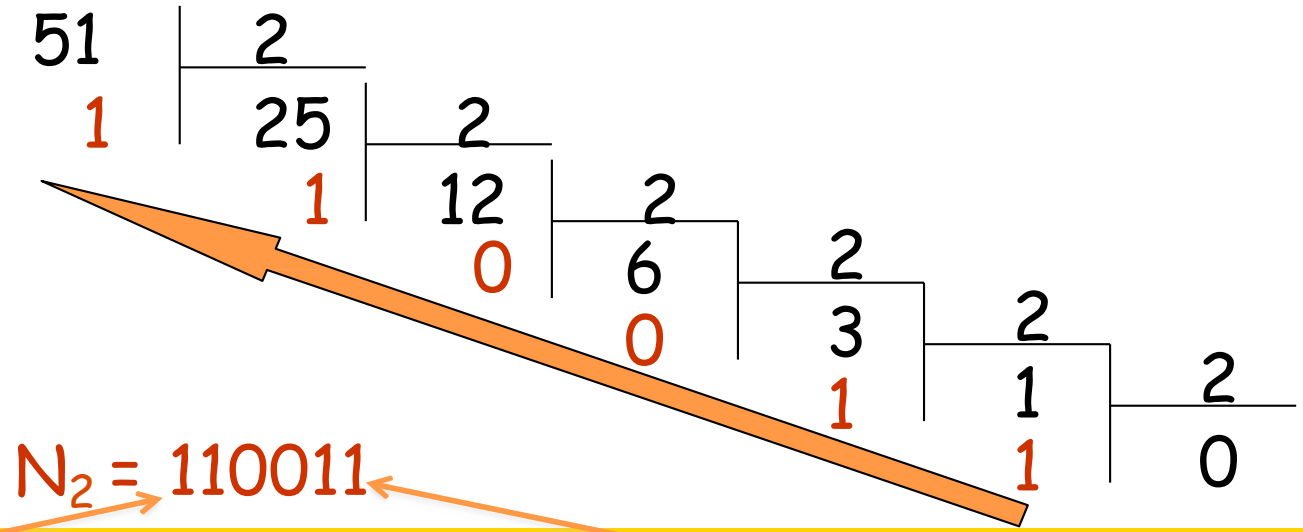
$125_{10} = ?_2$



Da Decimale a Binario: Esempio

(Da decimale a binario)

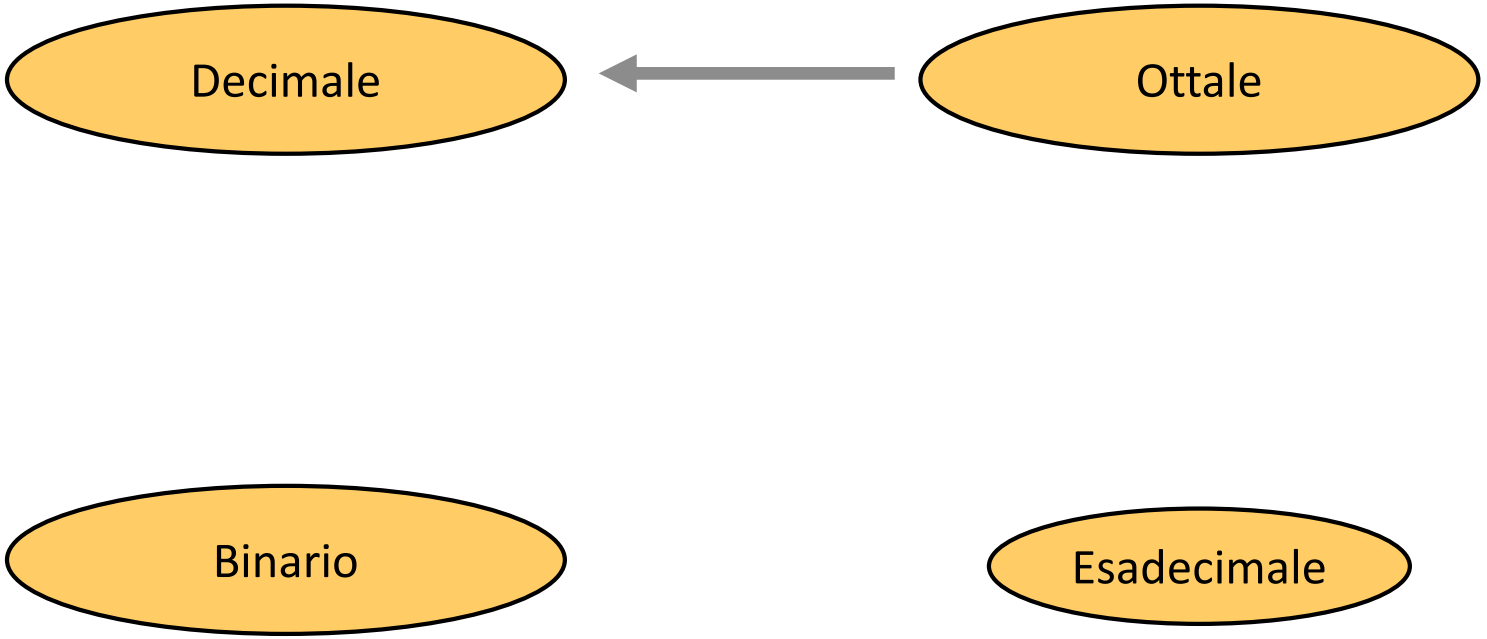
$N_{10} = 51$ $N_2 = ???$



$N_2 = 110011$

$51 = 1*2^5+1*2^4+0*2^3+0*2^2+1*2^1+1*2^0$

Da Ottale a Decimale



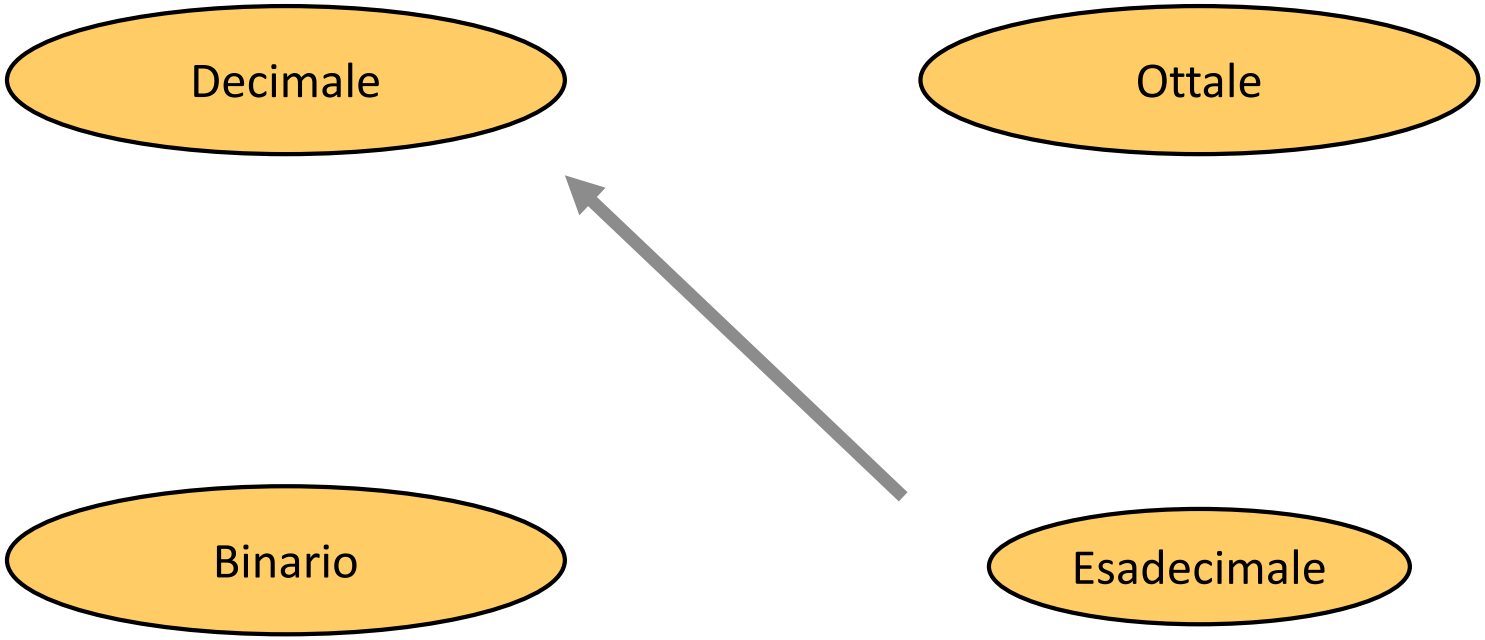
Da Ottale a Decimale: Tecnica

- Moltiplica ciascun bit per 8^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

Da Ottale a Decimale: Esempio

$$
\begin{array}{rrrrr}
724_8 \Rightarrow & 4 & \times\ 8^0 = & 4 & + \\
 & 2 & \times\ 8^1 = & 16 & + \\
 & 7 & \times\ 8^2 = & 448 & \\
\hline
 & & & 468_{10} &
\end{array}
$$

Da Esadecimale a Decimale



Da Esadecimale a Decimale: Tecnica

- Moltiplica ciascun bit per 16^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

Da Esadecimale a Decimale: Esempio

ABC_{16} =>

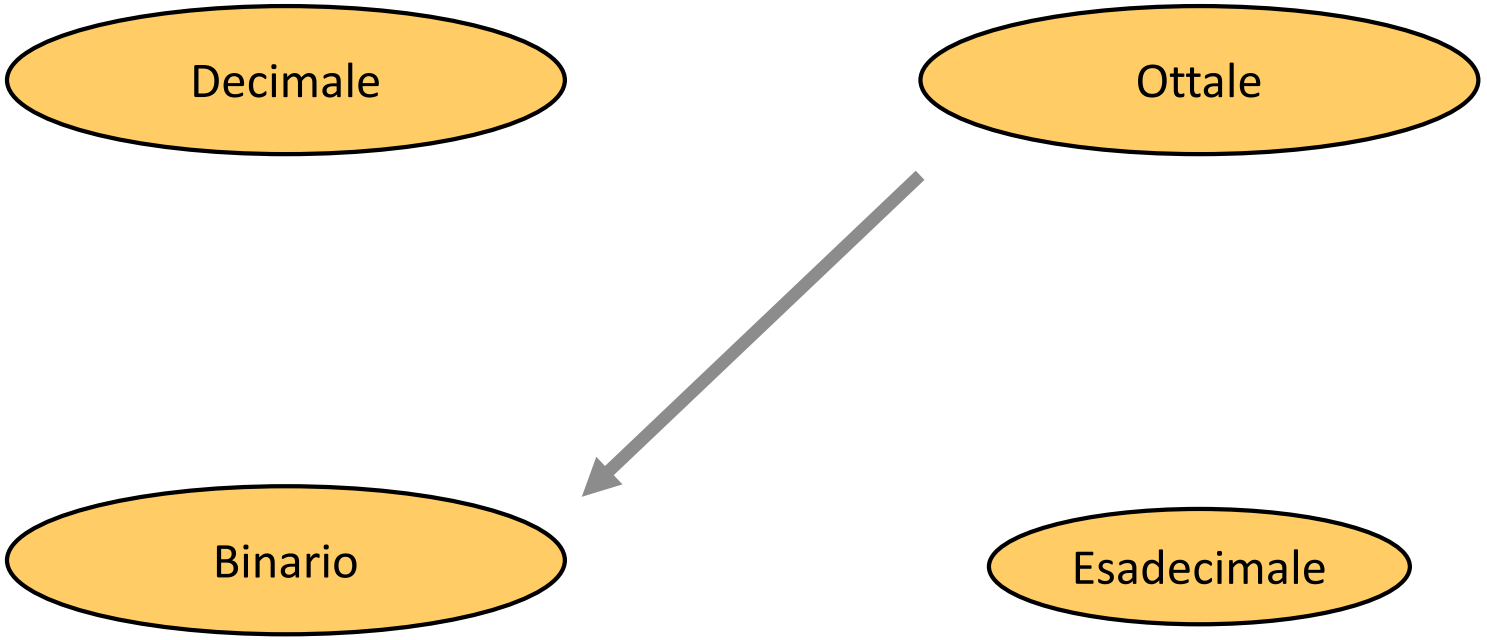
$C \times 16^0 = 12 \times 1 = 12 +$

$B \times 16^1 = 11 \times 16 = 176 +$

$A \times 16^2 = 10 \times 256 = 2560$

2748_{10}

Da Ottale a Binario

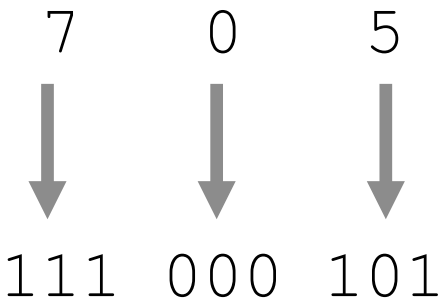


Da Ottale a Binario: Tecnica

- Converti ogni cifra ottale in una rappresentazione binaria equivalente a 3-bit

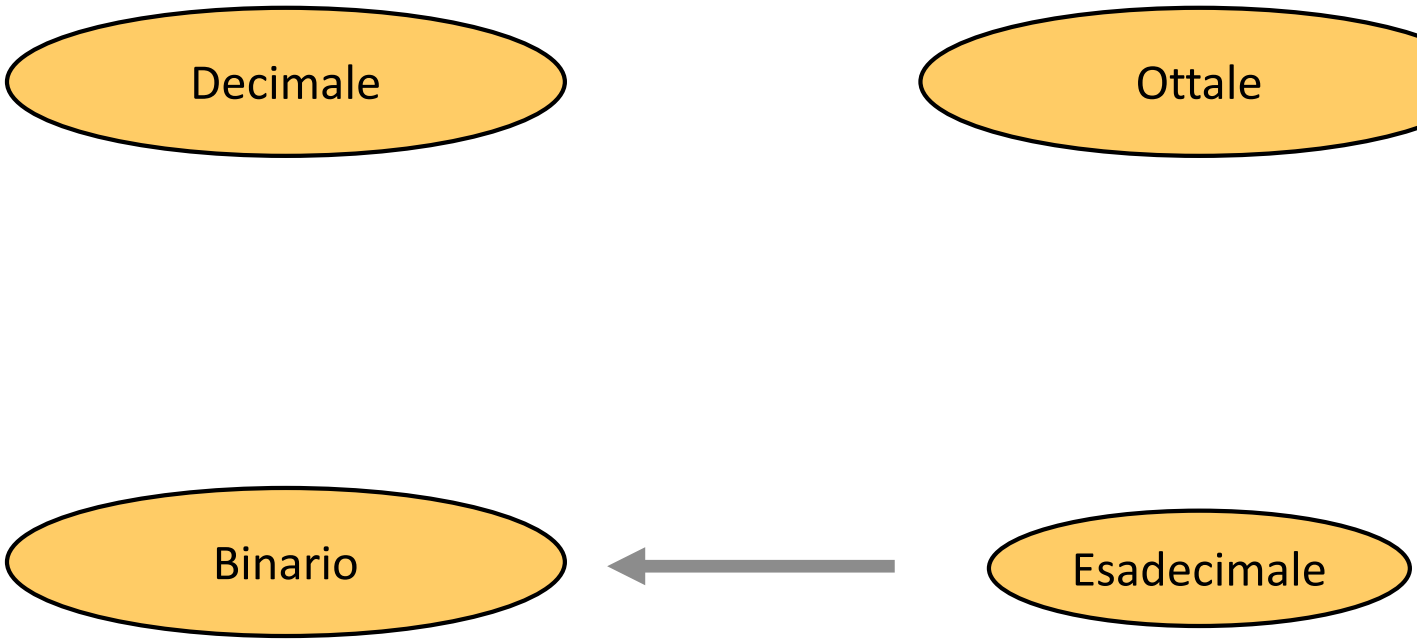
Da Ottale a Binario: Esempio

$705_8 = ?_2$



$705_8 = 111000101_2$

Da Esadecimale a Binario

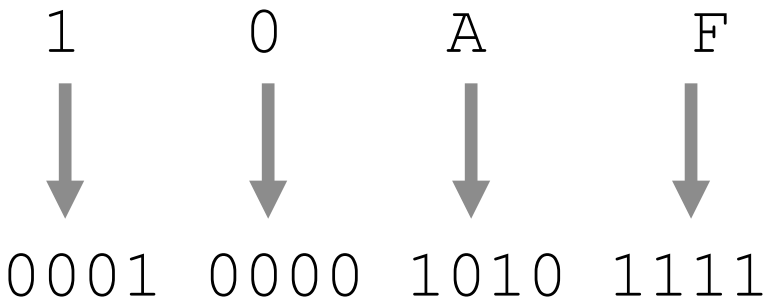


Da Esadecimale a Binario: Tecnica

- Converti ogni cifra esadecimale in una rappresentazione binaria equivalente a 4 bit

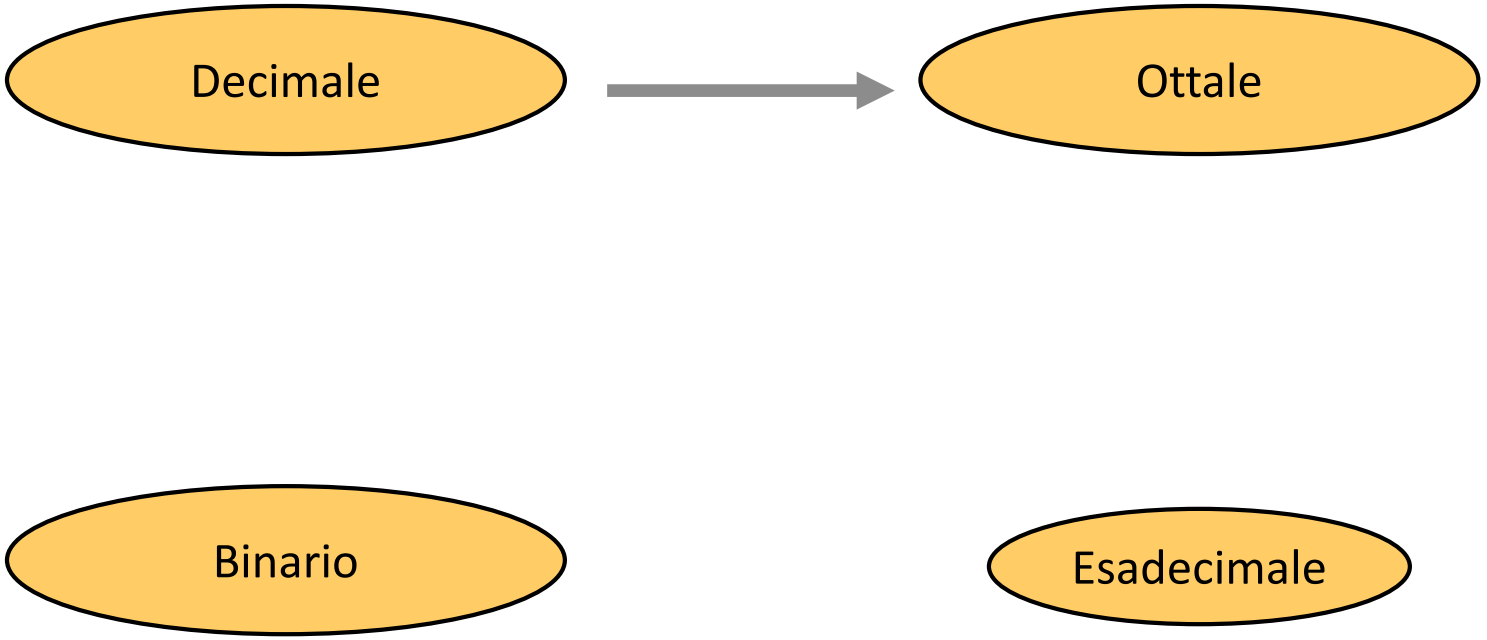
Da Esadecimale a Binario: Esempio

$10AF_{16} = ?_2$



$10AF_{16} = 0001000010101111_2$

Da Decimale a Ottale



Da Decimale a Ottale: Tecnica

- Dividi per 8
- Tieni traccia del resto

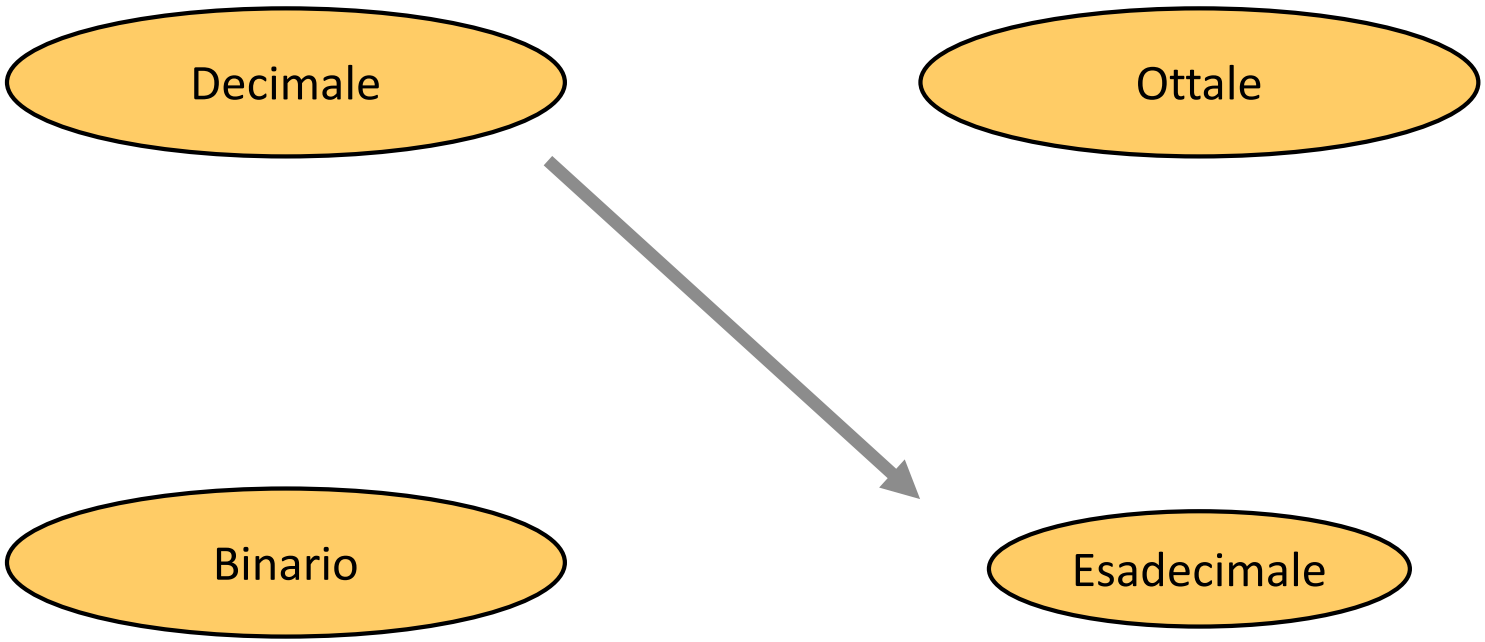
Da Decimale a Ottale: Esempio

$1234_{10} = ?_8$

8	1234	
8	154	2
8	19	2
8	2	3
	0	2

$1234_{10} = 2322_8$

Da Decimale ad Esadecimale



Da Decimale ad Esadecimale: Tecnica

- Dividi per 16
- Tieni traccia del resto

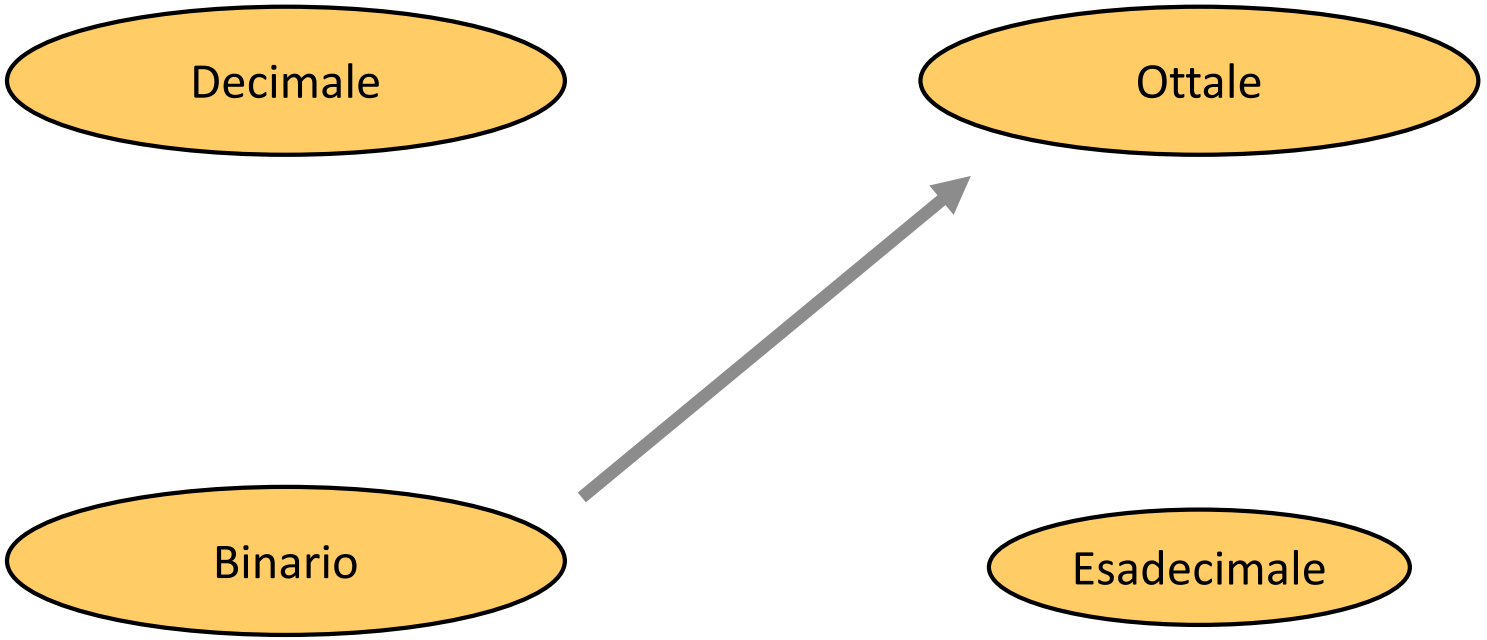
Da Decimale ad Esadecimale: Esempio

$1234_{10} = ?_{16}$

16	1234	
16	77	2
16	4	13 = D
	0	4

$1234_{10} = 4D2_{16}$

Da Binario ad Ottale



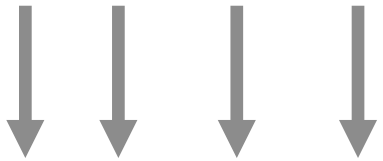
Da Binario ad Ottale

- Raggruppa i bit in gruppi di tre, partendo dalla destra
- Converti in cifre ottali

Da Binario ad Ottale: Esempio

$1011010111_2 = ?_8$

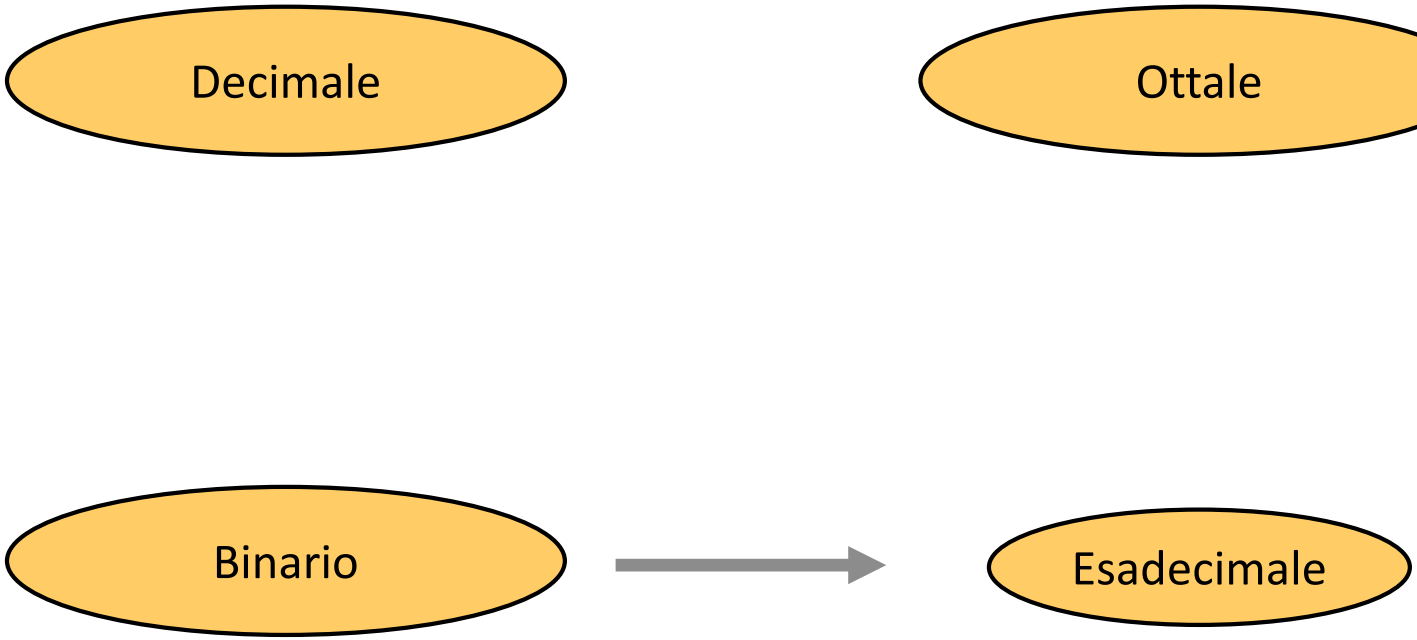
1 011 010 111



1 3 2 7

$1011010111_2 = 1327_8$

Da Binario ad Esadecimale

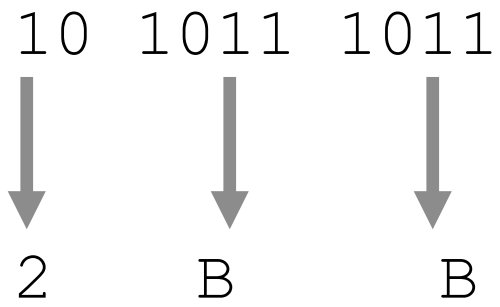


Da Binario ad Esadecimale: Tecnica

- Raggruppa i bit in gruppi di quattro, partendo dalla destra
- Converti in cifre esadecimali

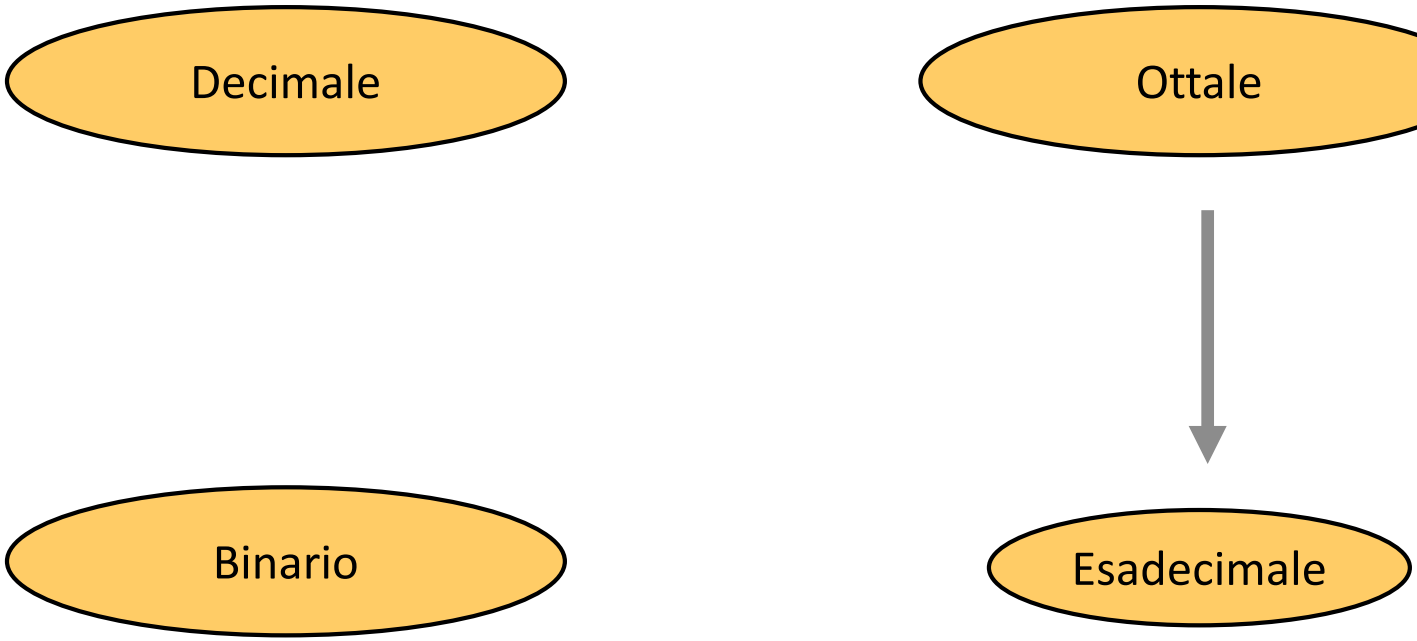
Da Binario ad Esadecimale: Esempio

$1010111011_2 = ?_{16}$



$1010111011_2 = 2BB_{16}$

Da Ottale ad Esadecimale

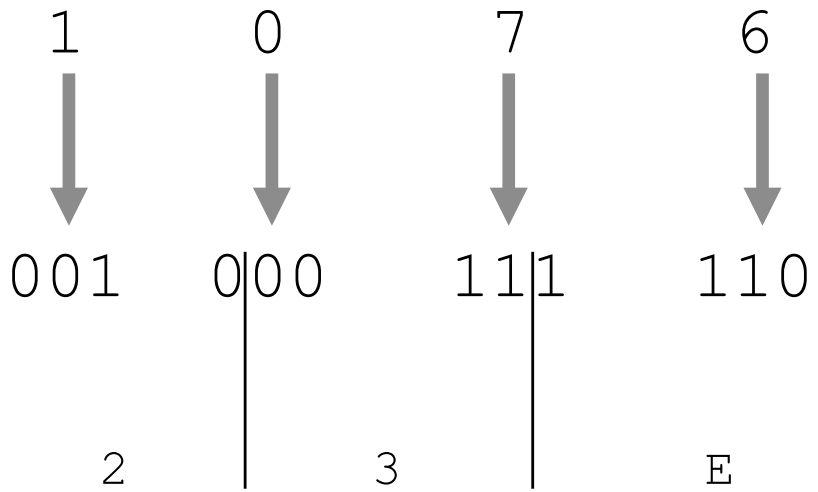


Da Ottale ad Esadecimale: Tecnica

- **Idea:** usare la codifica binaria come rappresentazione intermedia

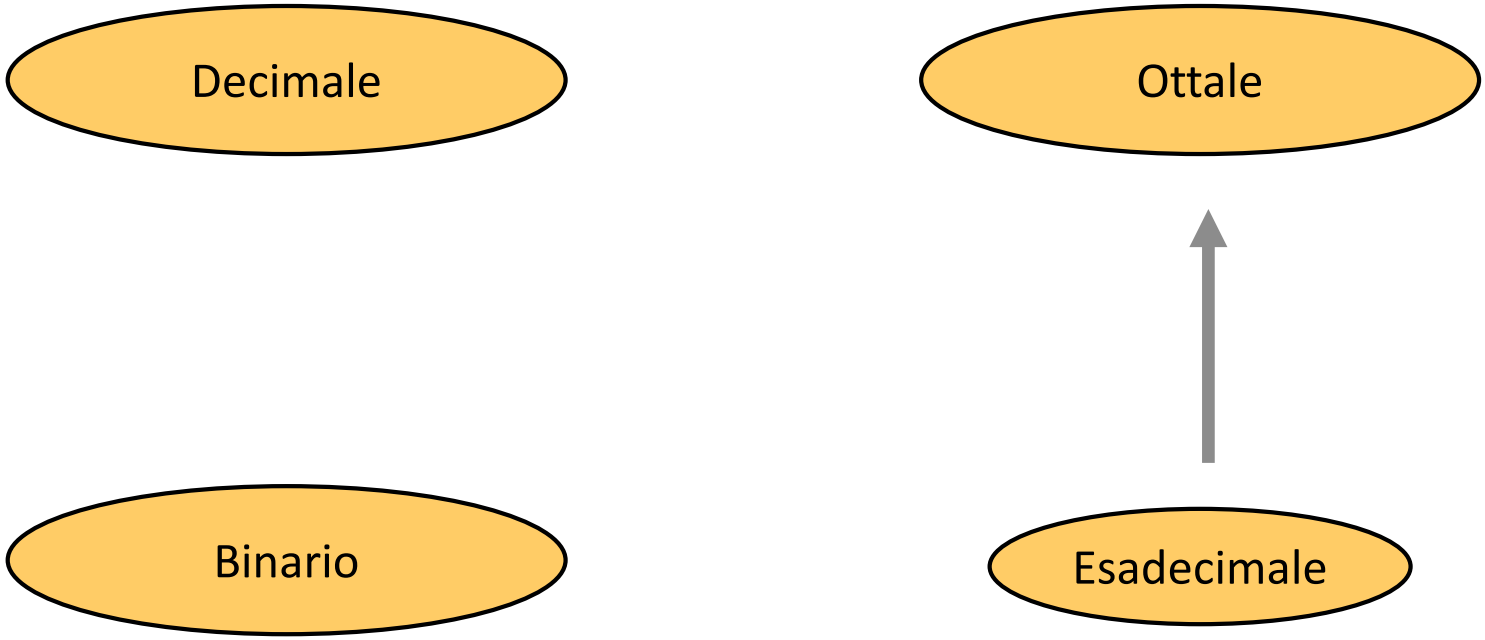
Da Ottale ad Esadecimale: Esempio

$1076_8 = ?_{16}$



$1076_8 = 23E_{16}$

Da Esadecimale ad Ottale

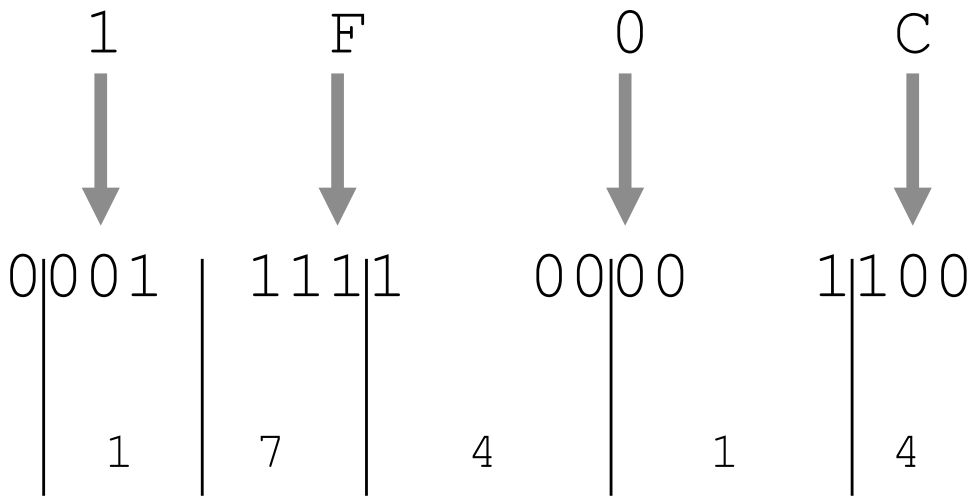


Da Esadecimale ad Ottale: Tecnica

- **Idea:** usare la codifica binaria come rappresentazione intermedia

Da Esadecimale ad Ottale: Esempio

$1F0C_{16} = ?_8$



$1F0C_{16} = 17414_8$

Più in generale…

- Per convertire un numero binario in un sistema che ha come base 2^z
  - Raggruppare le cifre in gruppi di z elementi
  - Convertire separatamente ciascun gruppo

Rappresentazione degli Interi: "Modulo e Segno"

- N = 0, +1, -1, +2, -2, +3, -3,…
- 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

$N_{10} = +14 \qquad N_{ms} = 01110$

$N_{10} = -14 \qquad N_{ms} = 11110$

Rappresentazione degli Interi: "Modulo e Segno"

- **Alfabeto binario**
  - Anche il segno è rappresentato da 0 o 1
  - 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: $+6_{10} = 0110_{ms}$ $-6_{10} = 1110_{ms}$
  - 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**
  - **Osservazione:** due rappresentazioni dello 0
    - Con **4 bit** sono $+0_{10} = 0000_{ms}$ $-0_{10} = 1000_{ms}$

Rappresentazione degli Interi: “Modulo e Segno”: Limiti sui Numeri Rappresentabili

- **4 bit** a disposizione
  - Possiamo rappresentare da 0000 a 0111 e da 1000 a 1111, in decimale da 0 a 7 e da 0 a -7
- **5 bit** a disposizione
  - Possiamo rappresentare da 00000 a 01111 e da 10000 a 11111, in decimale da 0 a 15 e da 0 a -15
- …
- Con **k bit** a disposizione possiamo rappresentare **numeri da 0 a $2^{k-1} - 1$** e **da $-(2^{k-1} - 1)$ a 0**

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, da 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 $+6_{10}$ => 0110_2 (N.B. ci vogliono 4 bit)
    2. Complemento di tutti i bit => 1001_{C2} (corrisponderebbe a -7_{10})
    3. Aggiungere 1 => 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

1 + 1 = 0 col riporto di 1

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

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

Codifica dei reali e dei frazionari (1/6)

- **Rappresentazione mediante interi**
  - L'insieme degli interi X può essere adoperato per rappresentare i reali x compresi nel medesi intervallo, approssimando il reale all'intero più prossimo.

$$X=r(x)=x \pm \varepsilon : 0 \leq \varepsilon \leq \frac{1}{2}$$

- **Rappresentazione dei frazionari**
  - Un numero frazionario rappresenta una classe di reali in modulo minore di 1, e per essi non è significativa la rappresentazione mediante interi.
  - Si applica un fattore di scala $M = b^n$, con n il numero di cifre e b la base di rappresentazione. Pertanto la rappresentazione di frazionari non negativi mediante interi diventa:

$$X = x \cdot M \pm \varepsilon : x \, \epsilon \, [0,1), |\varepsilon| \leq \frac{b^{-n}}{2}$$

  - Esempio: 0,500 con M = 10^3 viene rappresentato come 500, e 0,499 diventa 499.

Codifica dei reali e dei frazionari (2/6)

- **Conversione dei numeri frazionari**
  - MOLTIPLICHIAMO la parte frazionaria del numero dato per 2;
  - continuiamo a moltiplicare il RISULTATO ottenuto per 2 tenendo presente che, se il numero ottenuto è maggiore di 1 sottraiamo 1;
  - andiamo avanti fino a quando
    - non otteniamo un RISULTATO uguale a UNO oppure
    - fino a quando otteniamo un RISULTATO GIA' OTTENUTO IN PRECEDENZA.

Codifica dei reali e dei frazionari (3/6)

- **Virgola fissa e mobile**
  - Dicesi in virgola fissa una rappresentazione dei numeri con n cifre, in cui la posizione della virgola è predeterminata e non va esplicitamente espressa.
  - Esempio: il numero $.789x10^{-1}$ è rappresentato come 078900 in una rappresentazione a 6 cifre.

  - Dicesi in virgola mobile (o floating point) una rappresentazione in cui un numero reale è indicato dalla coppia (M,E), dove il primo termine prende il nome di mantissa e il secondo di esponente o caratteristica:

$$x = M \cdot b^E \pm \varepsilon$$

  - M ed E possono essere rappresentati in modi diversi.
  - Se $b|M|<1$, allora esiste un'altra rappresentazione (bM, E-1) che consente di ottenere una migliore approssimazione. Tra le possibili coppie che rappresentano x bisogna scegliere quella con l'approssimazione più alta, detta normalizzata, ovvero quando $b^{-1} \leq |M| < 1$.

Codifica dei reali e dei frazionari (4/6)

- **Aritmetica in virgola mobile**
  - Data una coppia (M,E) si può effettuare uno shift a destra della mantissa ed incremento dell'esponente, sempre che E+1 sia rappresentabile:

    Float shift right: $M' \cdot b^{E'} = M \cdot b^{E} = \frac{M}{b} \cdot b^{E+1}$
  - Uno shift a sinistra avviene come segue:

    Float shift left: $M' \cdot b^{E'} = M \cdot b^{E} = (Mb) \cdot b^{E-1}$
  - L'aritmetica in virgola mobile effettua le operazioni operando separatamente su mantissa ed esponente ed effettuando ove necessario operazioni di shift.
    - Addizione – Dati due numeri (A, Ea) e (B, Eb), se Ea è maggiore di Eb si effettua lo shift a destra fino ad ottenere Ea = Eb, altrimenti quello a sinistra. Successivamente, si sommano le mantisse.

      Esempio: Sommiamo A = (123×10^{0}) e B = (456×10^{-2}), effettuiamo due shift a destra su B fino ad ottenere B = $(4{,}56 \times 10^{0})$, sommiamo le mantisse e il risultato è C = $(127{,}56 \times 10^{0})$.
    - Moltiplicazione – Prodotto delle mantisse e somma degli esponenti con eventuale normalizzazione.

Codifica dei reali e dei frazionari (5/6)

- **Standard IEEE 754**
  - Per la rappresentazione dei numeri reali in macchina si adopera quella della virgola mobile binaria, come specificato nello standard IEEE 754. Per b=2, tale rappresentazione è la seguente:

$$\mathrm{x} = (-1)^{s} \cdot M \cdot 2^{E}$$

  - Essendo M normalizzata, ha il primo bit uguale a 1, ciò consente di ometterlo, così da avere un bit in più per la rappresentazione del numero. Questa tecnica prende il nome di bit nascosto:

$$\text{se } \mathrm{M} = \frac{1+f}{2}\text{, si ha che } \mathrm{x} = (-1)^{s} \cdot (1+f) \cdot 2^{E-1}$$

  - così che la rappresentazione è data dalla tripla (s, f, E).

  - Per l'esponente si ha la rappresentazione polarizzata: dato il numero di bit N_e, e l'esponente da rappresentare, E quello rappresentato e bias la costante di polarizzazione si ha E = e + bias, con l'intervallo degli esponenti effettivi pari a $e_{min} \leq e \leq e_{max}$.

Codifica dei reali e dei frazionari (6/6)

- Lo standard presenta vari formati di rappresentazione, tra i quali abbiamo quello a singola precisione su 32 bit e quella a doppia precisione su 64 bit. Nel primo caso, dati gli 8 bit per la rappresentazione dell'esponente, si ha che il suo intervallo di rappresentazione è pari a $0 < e < 255$. Pertanto la formula di rappresentazione diventa:

$$x = (-1)^s \cdot 2^{(e-127)} \cdot \underbrace{(1+f)}_{M}$$

con f pari alla parte frazionaria della mantissa, rappresentata con la tecnica del bit nascosto:

Esercizi 1

- Scrivere in binario semplice i seguenti numeri in base 10
  - 5310
  - 21110
- Scrivere in binario semplice su 7 bit il numero 13_{10}
- Scrivere in modulo e segno su 7 bit il numero 13_{10}
- Scrivere in modulo e segno su 7 bit il numero -13_{10}
- Scrivere in modulo e segno su 5 bit il numero 17_{10}

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}

Esercizi 3

- Un numero reale è rappresentato in virgola mobile secondo lo standard IEEE 754 su 32 bit nel seguente modo:

  s = 1
  E = 10000111
  f = 11011000000000000000000

- Ricavare il corrispondente valore decimale.

- Convertire i seguenti numeri decimali in virgola mobile in singola precisione secondo lo standard IEEE 754:
  1. -23.375_{10}
  2. -127.25_{10}
  3. $+131.5_{10}$
  4. -300.25_{10}
  5. -3.6_{10}

Esercizi Svolti

- Dato che e = 10000111_2 = 135_{10}.Si ha $N=(-1)^s \cdot 2^{(e-127)} \cdot 1.f = -1 \cdot 2^{135-127} \cdot 1.11011 = -1 \cdot 2^8 \cdot 1.11011 = -111011000_2 = -(2^8+2^7+2^6+2^4+2^3)_{10} = -472_{10}$

Esercizi Svolti

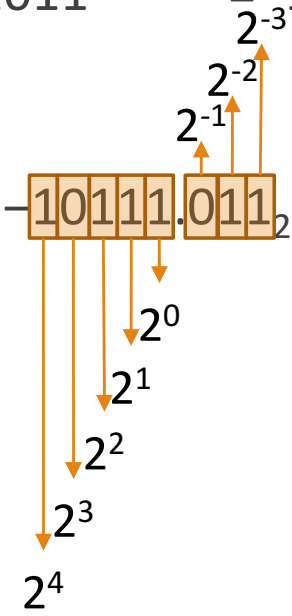
- Dato che e = 10000111_2 = 135_{10}.Si ha $N=(-1)^s \cdot 2^{(e-127)} \cdot 1.f = -1 \cdot 2^{135-127} \cdot 1.11011 = -1 \cdot 2^8 \cdot 1.11011 = -111011000_2 = -(2^8+2^7+2^6+2^4+2^3)_{10} = -472_{10}$

- $N_{10} = -23.375_{10} = -10111.011_2 =$

Esercizi Svolti

- Dato che e = 10000111_2 = 135_{10}.Si ha $N=(-1)^s \cdot 2^{(e-127)} \cdot 1.f = -1 \cdot 2^{135-127} \cdot 1.11011 = -1 \cdot 2^8 \cdot 1.11011 = -111011000_2 = -(2^8+2^7+2^6+2^4+2^3)_{10} = -472_{10}$

- $N_{10}=-23.375_{10}=-10111.011_2=-(2^4+2^2+2^1+2^0+2^{-2}+2^{-3})_{10}=$



$=-(16+4+2+1+0.25+0.125)_{10}=-23.375_{10}$

Esercizi Svolti

- Dato che e = 10000111_2 = 135_{10}.Si ha $N=(-1)^s \cdot 2^{(e-127)} \cdot 1.f = -1 \cdot 2^{135-127} \cdot 1.11011 = -1 \cdot 2^8 \cdot 1.11011 = -111011000_2 = -(2^8+2^7+2^6+2^4+2^3)_{10} = -472_{10}$

- $N_{10} = -23.375_{10} = -10111.011_2 = -1.0111011_2 \cdot 2^4$

  s= − =1, e=4, e_r= 4 + 127 = 131_{10}= 10000011_2

  m=1.0111011 = 0111011 (con hidden bit).

- N_{10}= 131.5_{10}= 10000011.1_2=$1.00000111_2 \cdot 2^7$

  s= + =0, e=7, e_r= 127 + 7 = 134 = 10000110_2

  m=1.00000111 = 00000111 (com hidden bit).

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}

$$111_2 = 1 * 2^2 + 1 * 2^1 + 1 * 2^0 = 7_{10}$$

Riferimenti

- **Libro di testo**
  - Capitolo 2