

Informazione binaria

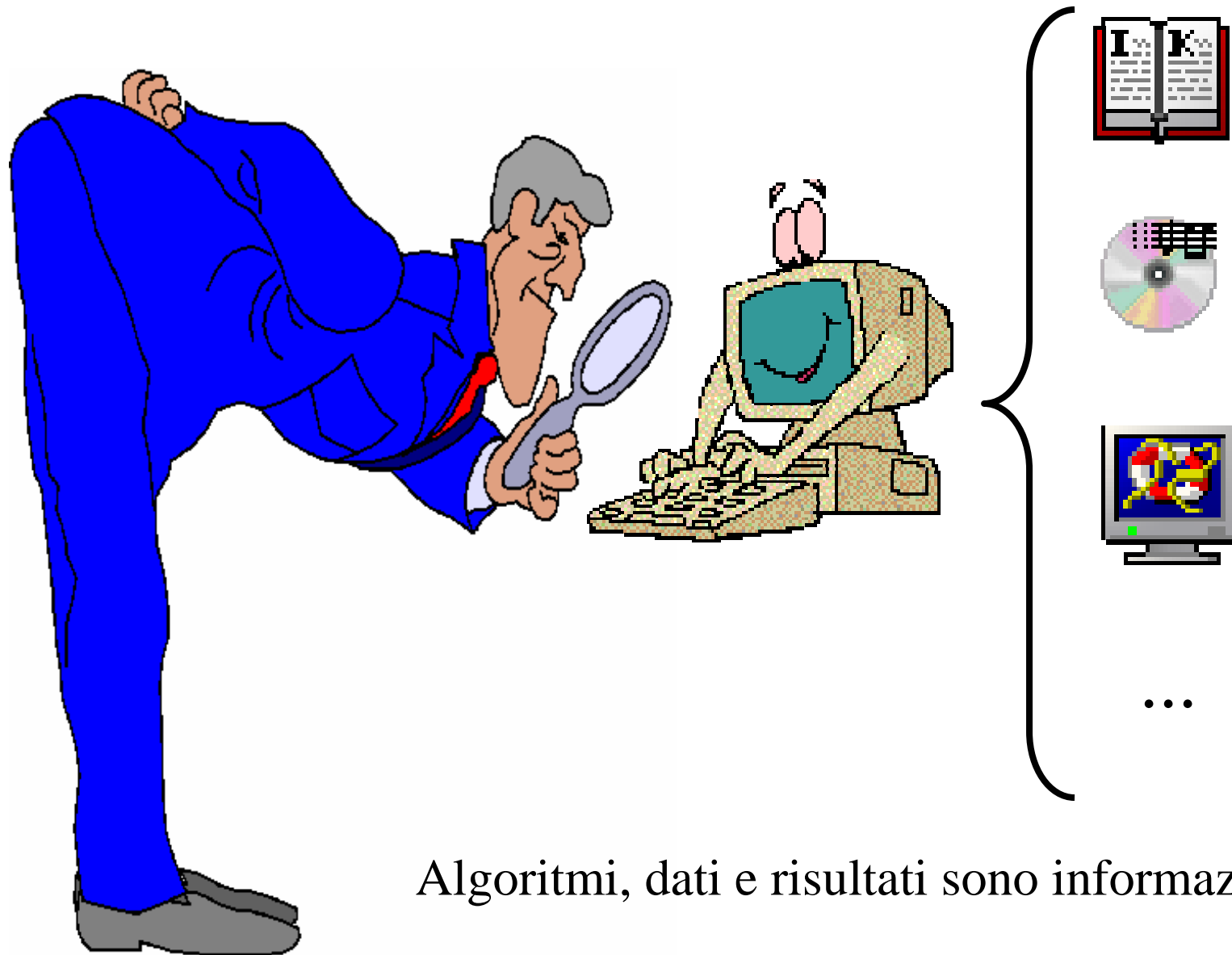
- prima parte -

Fondamenti di Informatica A

Percorso di Preparazione agli Studi di Ingegneria

Università degli Studi di Brescia

Docente: Massimiliano Giacomini



Algoritmi, dati e risultati sono informazioni...

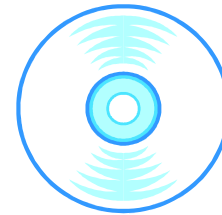
LA RAPPRESENTAZIONE DELL'INFORMAZIONE

Il concetto di informazione e supporto

- **Informazione**: entità che può essere comunicata
- Non può esistere informazione senza **supporto** fisico: mezzo su cui l'informazione può essere **memorizzata** e attraverso cui può essere **trasmessa**





Un brano musicale



Il CD in cui è memorizzato

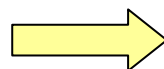
*L'aria attraverso cui viene
trasmesso*

...

- Il **supporto** deve poter assumere *configurazioni differenti* altrimenti non è in grado di portare informazione
- Ad ogni configurazione viene associato un certo *elemento di informazione*:
 - elemento di informazione *rappresentato* dalla configurazione del supporto
Es. soccorso sanitario: 
 - Associazione simboli-significati: convenzione semantica
Es. di convenzione semantica alternativa
Soccorso sanitario: 

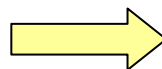
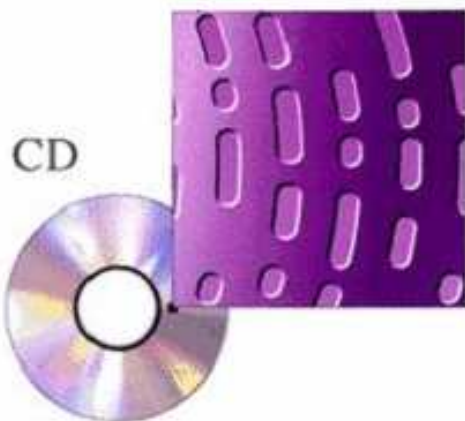
Mondo analogico e mondo digitale

- Come è rappresentato un brano musicale su un vecchio disco di vinile?



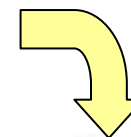
rappresentazione analogica (la forma del solco descrive per analogia il suono da produrre)

- Come è rappresentato un brano musicale su un CD?



0101 0011 0011

rappresentazione digitale (il supporto dà una informazione che va decodificata)

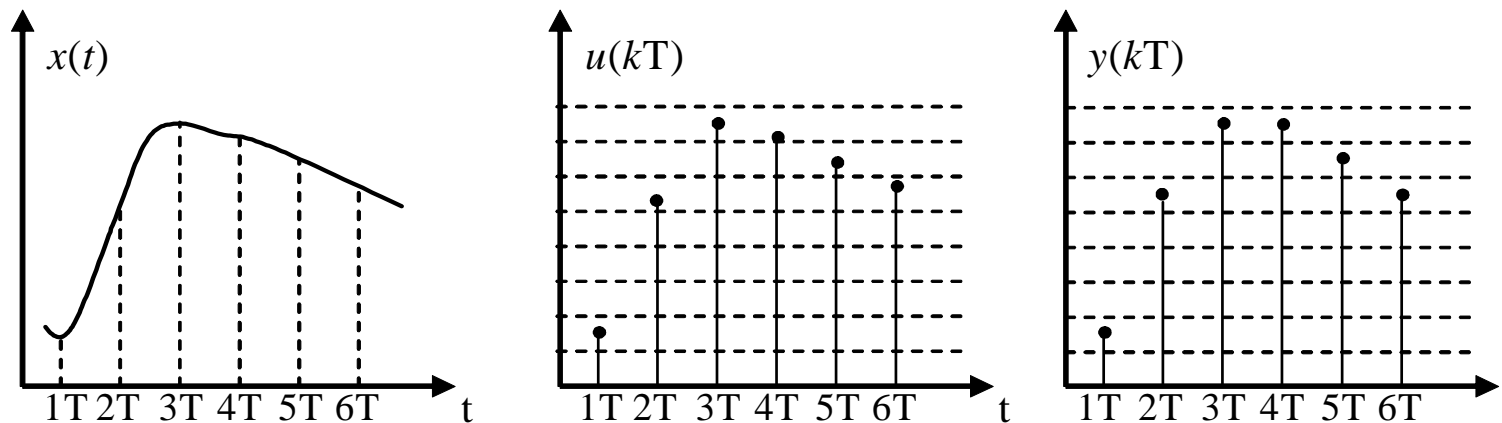
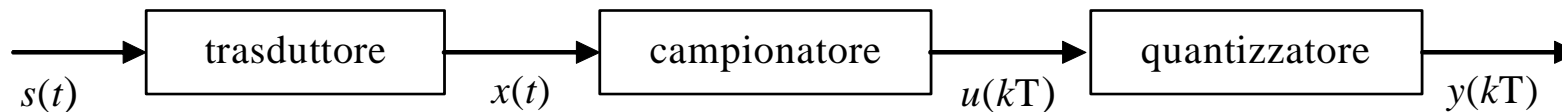


Rappresentazione analogica e digitale

- **Rappresentazione analogica:** le configurazioni assunte dal supporto possono variare in modo continuo e riflettono secondo una analogia diretta la struttura dell'informazione
- **Rappresentazione digitale:** l'insieme delle configurazioni è finito (e fissato a priori) e la struttura dell'informazione si può dedurre dalle configurazioni solo attraverso una specifica regola di codifica.
Conseguenza: si può rappresentare un insieme di elementi di informazione finito e predefinito (se emerge l'esigenza di codificare un nuovo elemento, necessario rivedere o estendere la codifica)

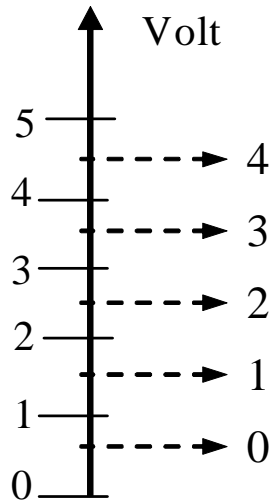
Perché la rappresentazione digitale? E perché proprio quella binaria?

- Una qualunque grandezza fisica può essere convertita in forma digitale (conversione analogico-digitale o digitalizzazione)

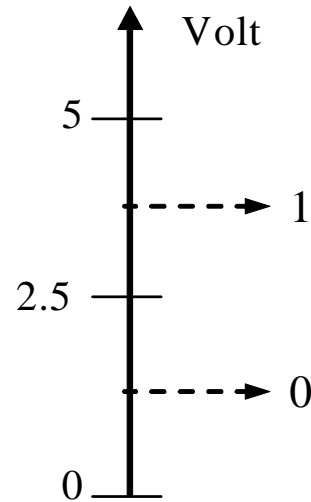


- ❖ Frequenza di campionamento: numero di campioni acquisiti nell'unità di tempo (Hz: al secondo)
- ❖ Intervalli di quantizzazione: sottointervalli in cui si suddivide il codominio di u . Ad ognuno di essi è associato un valore detto livello di quantizzazione
- ❖ La conversione analogico-digitale comporta perdita di informazione, ma...
 - Campionamento è (almeno in linea teorica) reversibile [no perdita di informazione] se la frequenza di campionamento è sufficientemente alta
 - Quantizzazione: perdita di informazione arbitrariamente piccola all'aumentare del numero di livelli di quantizzazione + valori dei segnali analogici non distinguibili (rumore, imprecisione strumento di misura, ecc.)

- Immunità al rumore (affidabilità):

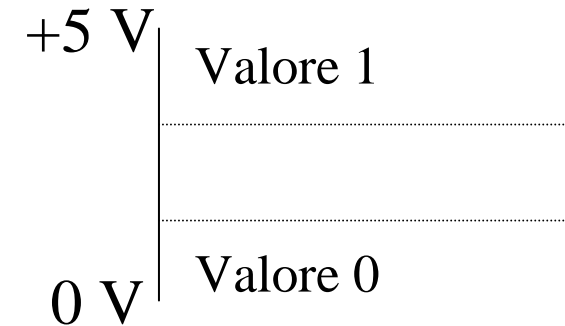


(a)



(b)

In realtà:



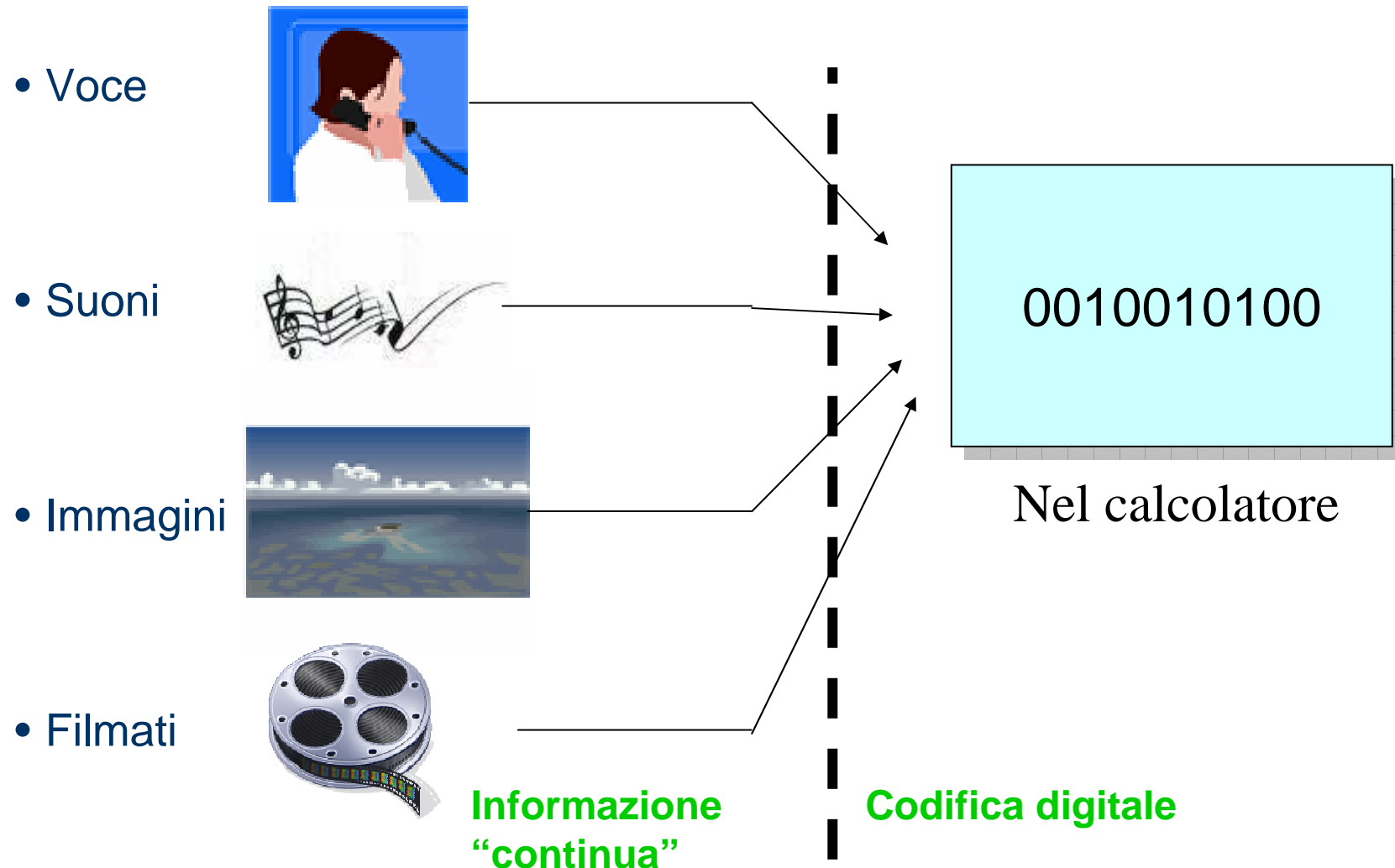
- Dal punto di vista tecnologico, realizzabilità mediante dispositivi a due stati (livelli di tensione, magnetizzazione, ecc.)



Tutti i calcolatori elettronici e i dispositivi di memorizzazione impiegati utilizzano la codifica binaria

❖ Come posso limitarmi a due soli simboli 0 e 1?

- ❖ Utilizzo sequenze di bit [ad esempio, associo agli 8 livelli di quantizzazione le sequenze 000, 001, 010, 011, ...]



Altre ragioni del “predominio del digitale”

- Informazioni di natura “prettamente simbolica”
- Uniformità nella rappresentazione: una varietà ristretta di dispositivi può realizzare svariate funzioni
- Tecniche di codifica e trasmissione uniformi su tipologie di informazioni diverse
- Elaborazione simbolica (numerica) dei segnali digitali

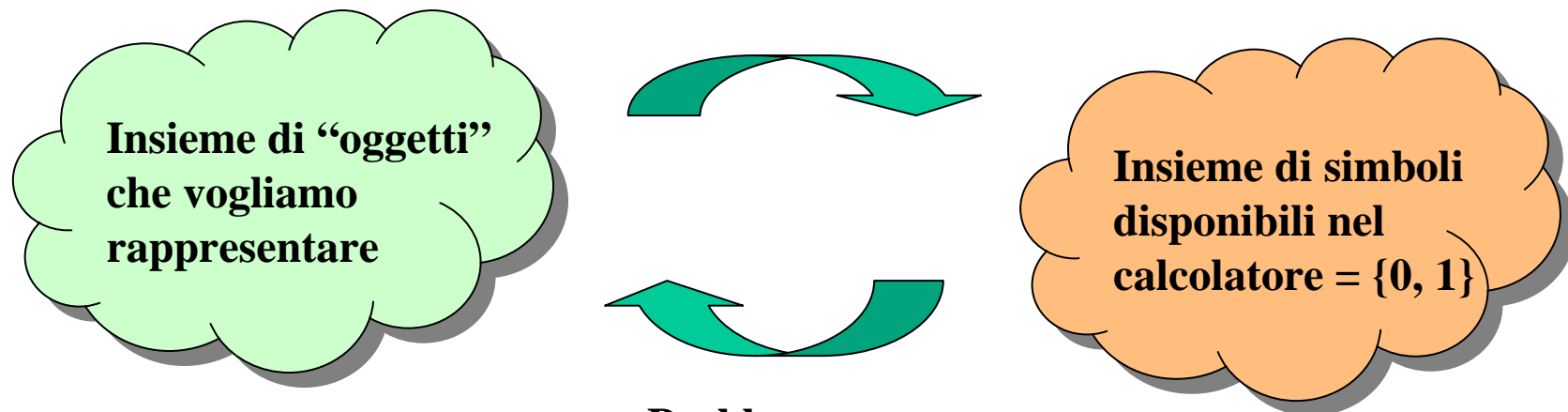
LA CODIFICA BINARIA

Il problema della rappresentazione

- Abbiamo a disposizione due simboli:

Alfabeto binario = {0, 1}

dove 0 e 1 sono dette cifre binarie o BIT (*Binary digIT*)



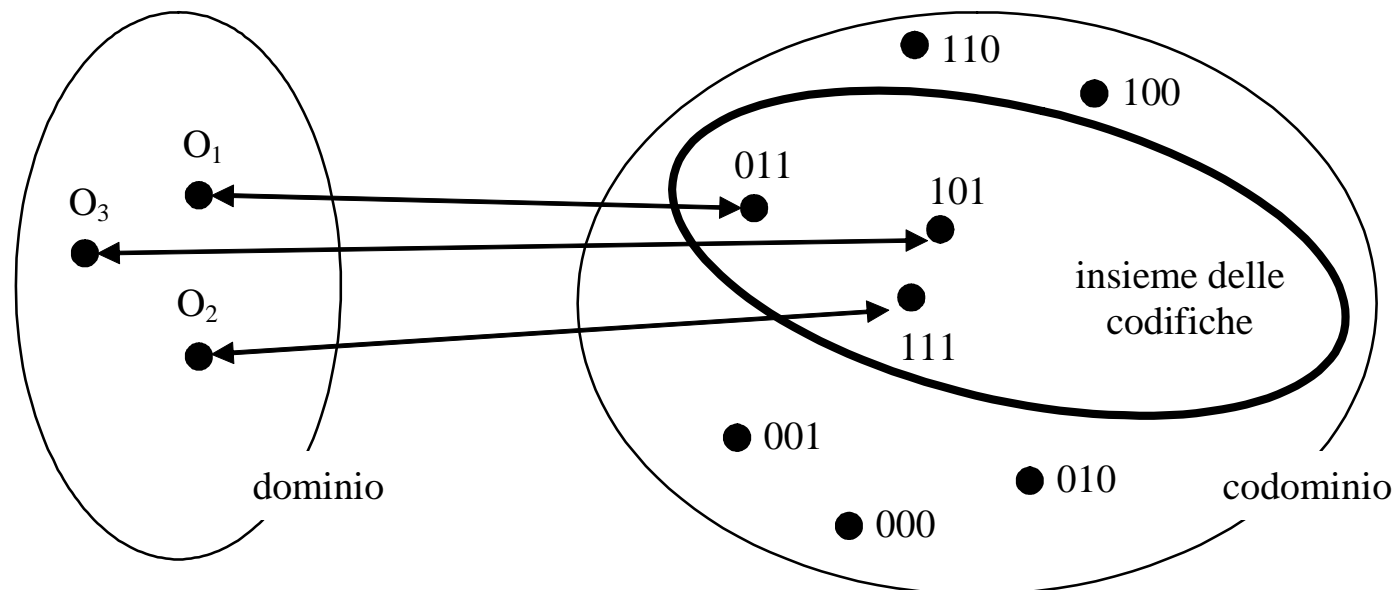
Problema:

*assegnare un codice univoco a tutti
gli oggetti compresi in un insieme*

- Altro problema: il numero di bit disponibili per la rappresentazione nel calcolatore può essere limitato (es. registri CPU)

Codice binario a n bit

- Funzione:
 - **dominio** (insieme di oggetti da rappresentare)
 - **codominio**: insieme di tutte le possibili sequenze di n bit
- Funzione biunivoca tra il dominio e la sua immagine, detta **insieme delle codifiche**
- Esempio di codice binario a 3 bit:



- Ho 2 simboli ed n oggetti da codificare: qual è la lunghezza k delle sequenze?
- Oppure: dispongo di sequenze di lunghezza k di simboli 0 e 1, qual è il numero n di oggetti che posso codificare?
- Se $k = 1$
 - Posso codificare due oggetti ($n=2$): al primo assegno la codifica '0' e al secondo assegno la codifica '1'
- Se $k = 2$
 - Posso codificare $n=4$ oggetti: 00, 01, 10, 11
- Se $k = 3$
 - Posso codificare $n=8$ oggetti: 000, 001, 010, 011, 100, 101, 110, 111
- Qual è la **regola**????

$$n = 2^k \qquad k = \lceil \log_2 n \rceil$$

- Se ho a disposizione sequenze di $k = 5$ bit, quanti elementi posso codificare?
 - $n = 2^5 = 32$ elementi
- Se $n = 128$, di quanti bit ho bisogno (k) per codificarli tutti?
 - $k = \lceil \log_2 128 \rceil = 7$
- ...e se $n = 129$???
- Allora ho bisogno di 1 bit in più! Ottengo uno spreco di configurazioni, perché con 8 bit posso codificare fino a 256 elementi

Esempio: i mesi dell'anno

Gennaio Febbraio
Marzo Aprile
Maggio Giugno
Luglio Agosto
Settembre Ottobre
Novembre Dicembre

1 bit → 2 gruppi

Gennaio	Febbraio	0
Marzo	Aprile	
Maggio	Giugno	
Luglio	Agosto	1
Settembre	Ottobre	
Novembre	Dicembre	

Gennaio 000	Febbraio 010
Marzo 001	Aprile 011
Maggio	Giugno
Luglio 100	Agosto 110
Settembre 101	Ottobre 111
Novembre	Dicembre

3 bit → 8 gruppi

2 bit → 4 gruppi

Gennaio	Febbraio	00	01
Marzo	Aprile		
Maggio	Giugno		
Luglio 10	Agosto 11	10	11
Settembre	Ottobre		
Novembre	Dicembre		

Gennaio 0000	Febbraio 0100
Marzo 0010	Aprile 0110
Maggio 0011	Giugno 0111
Luglio 1000	Agosto 1100
Settembre 1010	Ottobre 1110
Novembre 1011	Dicembre 1111

4 bit → 16 gruppi... mancano 4 configurazioni!

Esempio: codifica BCD

Cifra decimale rappresentata	Codifica binaria			
	b_3	b_2	b_1	b_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
	1	0	1	0
	1	0	1	1
	1	1	0	0
	1	1	0	1
	1	1	1	0
	1	1	1	1

← codifiche
non usate

Tipologie di codici

Nel seguito vedremo tipologie di rappresentazioni diverse:

- Senza assumere limitazioni sul numero di bit a disposizione:
per numeri [notazione binaria, ovvero posizionale con base 2]
- Disponendo di un numero di bit limitato:
 - numeri naturali
 - interi relativi [valore assoluto e segno, complemento a due]
 - “reali” [virgola fissa e virgola mobile]
 - valori logici, caratteri alfabetici, testi
 - suoni, immagini e sequenze video
 - codici per la rilevazione e correzione di errori
- Codici di compressione (senza | con perdita)

Tipologie di codici

Nel seguito vedremo tipologie di rappresentazioni diverse:

- Senza assumere limitazioni sul numero di bit a disposizione:
per numeri [notazione binaria, ovvero posizionale con base 2]
- Disponendo di un numero di bit limitato:
 - numeri naturali
 - interi relativi [valore assoluto e segno, complemento a due]
 - “reali” [virgola fissa e virgola mobile]
 - valori logici, caratteri alfabetici, testi
 - suoni, immagini e sequenze video
 - codici per la rilevazione e correzione di errori
- Codici di compressione (senza | con perdita)

Sistema di numerazione posizionale

base = b insieme di **cifre** usate: $\{0, \dots, b-1\}$

Rappresentazione:

$$a_k a_{k-1} \dots a_0 \cdot a_{-1} a_{-2} \dots a_{-h}$$

Numero rappresentato:

$$\sum_{i=-h}^{+k} a_i b^i = a_k * b^k + a_{k-1} * b^{k-1} + \dots + a_0 * b^0 + a_{-1} * b^{-1} + \dots + a_{-h} * b^{-h}$$

Ad ogni cifra è attribuito un **peso** che dipende dalla sua **posizione**

Esempio (con base $b=10$) $N = 4027.234_{10}$

$$N = 4 \cdot 10^3 + 0 \cdot 10^2 + 2 \cdot 10^1 + 7 \cdot 10^0 + 2 \cdot 10^{-1} + 3 \cdot 10^{-2} + 4 \cdot 10^{-3}$$

Le basi più comuni

- Se la base è b , allora le *cifre* che possono essere utilizzate per comporre un numero vanno
da 0 a $b-1$
 - Esempio: $b = 10$, cifre possibili: $[0,1,2,3,4,5,6,7,8,9]$
 - Esempio: $b = 2$, cifre possibili: $[0,1]$
 - Esempio: $b = 8$, cifre possibili: $[0,1,2,3,4,5,6,7]$
 - Esempio: $b = 16$, cifre possibili: $[0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F]$

Notazione binaria

base = 2

cifre: 0, 1

- Numeri espressi nella forma

$$(a_n a_{n-1} \dots a_1 a_0 . a_{-1} a_{-2} \dots)_2 \quad [a_i \in \{0,1\}]$$

il cui “valore” è

$$(a_n * 2^n + a_{n-1} * 2^{n-1} + \dots + a_0 + a_{-1} * 2^{-1} + a_{-2} * 2^{-2} \dots)$$

ESEMPIO

$$N = 101011.1011_2$$

$$N = 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$$

$$+ 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} + 1 \cdot 2^{-4} =$$

$$= 43.6875_{10}$$

Conversione binario \Rightarrow decimale

Come visto, la conversione si ottiene direttamente dalla definizione stessa di numero binario

- Scriviamo i numeri denotando la base attraverso il pedice: es.

1101.1_{due}

- E' facile convertirlo in un numero decimale facendo:

$$\begin{aligned} 1101.1_{\text{due}} &= 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} = \\ &= 8_{\text{dieci}} + 4_{\text{dieci}} + 0_{\text{dieci}} + 1_{\text{dieci}} + 0.5_{\text{dieci}} = 13.5_{\text{dieci}} \end{aligned}$$

- Altri esempi:

$$\begin{aligned} 10101.01_{\text{due}} &= 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} \\ &= 16 + 4 + 1 + 0.25 \\ &= 21.25_{\text{dieci}} \end{aligned}$$

$$\begin{aligned} 110010.001_{\text{due}} &= 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + \dots + 1 \times 2^{-3} = \\ &= 32 + 16 + 2 + 0.125 \\ &= 50.125_{\text{dieci}} \end{aligned}$$

$$\begin{aligned} 1000001_{\text{due}} &= 1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\ &= 64 + 1 \\ &= 65_{\text{dieci}} \end{aligned}$$

ESERCIZIETTO

- Il numero binario 101001011_{due} è pari o dispari?
- A quale numero decimale corrisponde?

$$101001011_{\text{due}} = (1 \times 2^8 + 0 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0)_{\text{dieci}} = (256 + 64 + 8 + 2 + 1)_{\text{dieci}} = 331_{\text{dieci}}$$

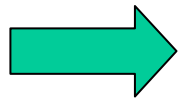
E' evidente che è possibile verificare direttamente se un numero binario è pari o dispari, senza passare per il decimale...

Conversione decimale \Rightarrow binario

- Usare la definizione è complesso!

- Esempio: $345_{\text{dieci}} = \overset{3_{\text{dieci}}}{11} \times \overset{10_{\text{dieci}}}{1010} \overset{2_{\text{dieci}}}{10} + 100 \times 1010^{01} + 101 \times 1010^0 = \dots$

... e poi bisogna fare le moltiplicazioni e l'elevamento a potenza in base 2 e sommarne i risultati in base 2



Useremo un “metodo pratico”.

Dato un numero decimale N , è innanzitutto necessario distinguere la parte intera e la parte frazionaria: $N = I.F$

ES: con 56.5_{dieci} , convertiamo separatamente 56 e 0.5

REGOLA PRATICA PER CONVERTIRE LA PARTE INTERA

$$I = \overset{?}{C_n} * 2^n + \overset{?}{C_{n-1}} * 2^{n-1} + \dots + \overset{?}{C_1} * 2^1 + \overset{?}{C_0}$$

$$I/2 = C_n * 2^{n-1} + C_{n-1} * 2^{n-2} + \dots + C_2 * 2^1 + C_1 \quad \text{con resto } C_0$$

$$I/4 = C_n * 2^{n-2} + C_{n-1} * 2^{n-3} + \dots + C_2 \quad \text{con resto } C_1$$

.....

Esempio

$$21_{\text{dieci}} = 10101_{\text{due}} = (1x2^4 + 0x2^3 + 1x2^2 + 0x2^1) + 1x2^0$$

$$/2 : (1x2^3 + 0x2^2 + 1x2^1) + 0x2^0 \quad \text{con resto } 1$$

$$/2 : (1x2^2 + 0x2^1) + 1x2^0 \quad \text{con resto } 0$$

$$/2 : (1x2^1) + 0x2^0 \quad \text{con resto } 1$$

$$/2 : 1x2^0 \quad \text{con resto } 0$$

$$/2 : 0 \quad \text{con resto } 1$$

- In pratica, per convertire un numero decimale in un numero binario basta fare una sequenza di divisioni (operazione **div**) per la **base 2** e prendere il resto (operazione **mod**)
- Alla fine, quando il numero è diventato 0, si leggono i resti dall'ultimo al primo e si ottiene di nuovo il numero
- Esempio:
 - $56 \text{ div } 2 = 28 \ \& \ 56 \text{ mod } 2 = \mathbf{0}$ (**cifra meno significativa** del numero bin)
 - $28 \text{ div } 2 = 14 \ \& \ 28 \text{ mod } 2 = \mathbf{0}$
 - $14 \text{ div } 2 = 7 \ \& \ 14 \text{ mod } 2 = \mathbf{0}$
 - $7 \text{ div } 2 = 3 \ \& \ 7 \text{ mod } 2 = \mathbf{1}$
 - $3 \text{ div } 2 = 1 \ \& \ 3 \text{ mod } 2 = \mathbf{1}$
 - $1 \text{ div } 2 = 0 \ \& \ 1 \text{ mod } 2 = \mathbf{1}$ (**cifra più significativa** del numero bin)

Si ottiene $\mathbf{111000}_{\text{due}} = 32_{\text{dieci}} + 16_{\text{dieci}} + 8_{\text{dieci}} + 0 + 0 + 0 = 56_{\text{dieci}}$

Esempio di conversione da decimale a binario

57_{10}

57	1
28	0
14	0
7	1
3	1
1	1
0	

*Si legge dal
basso verso
l'alto !!!*


Risultato = **111001**_{due}

Esercizio: riconvertire il risultato in decimale

ERRORE TIPICO:

considerare la prima cifra ottenuta come la più significativa:

81	1
40	0
20	0
10	0
5	1
2	0
1	1
0	



otterrei **1000101** che vale 69!

NB: se si è colti dal dubbio, ragionare:

**se continuassi il procedimento di divisioni successive aggiungerei zeri;
questi “non pesano” solo se corrispondono alle posizioni più significative
(0...0xyz) !**

Metodo “più pratico” di conversione da decimale intero a binario

128	64	32	16	8	4	2	1	
7	6	5	4	3	2	1	0	

ES: convertire in binario 137

$$\begin{aligned} 137 &= 128 + 8 + 1 & [= 2^7 + 2^3 + 2^0] \\ &= 10001001_2 \end{aligned}$$

**NB: almeno per valori non troppo elevati,
conviene abituarsi ad utilizzare questo metodo.**

REGOLA PRATICA PER CONVERTIRE LA PARTE FRAZIONARIA

$$F = C_{-1} * 2^{-1} + C_{-2} * 2^{-2} + + C_{-n} * 2^{-n}$$

$$F * 2 = C_{-1} + C_{-2} * 2^{-1} + + C_{-n} * 2^{-(n-1)}$$

la parte intera è C_{-1}

$$(F * 2 - C_{-1}) * 2 = C_{-2} + + C_{-n} * 2^{-(n-2)}$$

la parte intera è C_{-2}

.....

- Basta fare una *sequenza di moltiplicazioni per 2* e prendere la parte intera di ciascun prodotto dalla cifra più significativa a quella meno significativa
- Esempio: $0.587_{\text{dieci}} \rightarrow \text{binario?}$
 - $0.587 \times 2 = 1.174$: p.f. 0.174, parte intera **1** (cifra più significativa)
 - $0.174 \times 2 = 0.348$: p.f. 0.348, parte intera **0**
 - $0.348 \times 2 = 0.696$: p.f. 0.696, parte intera **0**
 - $0.696 \times 2 = 1.392$: p.f. 0.392, parte intera **1**
 - $0.392 \times 2 = 0.784$: p.f. 0.784, parte intera **0**
 - $0.784 \times 2 = 1.568$: p.f. 0.568, parte intera **1**
 -

Si ottiene 0.10010_{due} con 5 cifre binarie dopo la virgola, oppure 0.100101_{due} con 6 cifre binarie dopo la virgola, oppure...

➔ In ogni caso c'è un' approssimazione

Esempio: convertire 43.6875_{10} in binario


43		1	↑
21		1	
10		0	
5		1	
2		0	
1		1	
0			

0.6875		1	
.375		0	
.75		1	
.5		1	↓
0			

$\Rightarrow 43.6875_{10} = 101011.1011_2$

ERRORE TIPICO:

considerare la prima cifra ottenuta come la meno significativa:

0.6875		1	
0.375		0	
0.75		1	
0.5		1	
0			

$$0.1101 = 0.5 + 0.25 + \dots > 0.75$$

NB: se continuassi il procedimento di moltiplicazioni successive aggiungerei zeri; questi “non pesano” solo se corrispondono alle posizioni meno significative (0.xyz0...0) !

Operazioni aritmetiche

- Per le operazioni in base 2 valgono le *stesse regole e proprietà delle operazioni in base 10*

ARITMETICA BINARIA: ADDIZIONE

+	0	1
0	0	1
1	1	(1) 0

Riporto: $1 + 1 = 2_{\text{dieci}} = 10_{\text{due}}$

Esempio:

1 1 0 + 6

1 0 = 2

1 0 0 0

ARITMETICA BINARIA: SOTTRAZIONE

-	0	1
0	0	(1) 1
1	1	0

Prestito (borrow): $10_2 - 1_2 (= 2_{10} - 1_{10}) = 01_2$

Esempio:

1 1 1 0 – 14

1 1 = 3

1 0 1 1 11

ARITMETICA BINARIA: MOLTIPLICAZIONE

*	0	1
0	0	0
1	0	1

Esempio:

$$\begin{array}{r} 111010 * 58 \\ 1011 = 11 \\ \hline 111010 \\ 111010 \\ 000000 \\ 111010 \\ \hline 100111110 \end{array} \quad \begin{array}{l} 638 \end{array}$$

ALTRI ESEMPI

addizione

$$\begin{array}{r} 1\ 0\ 1\ 1\ + \\ 0\ 1\ 1\ 1\ = \\ \hline (1)\ 0\ 0\ 1\ 0 \end{array}$$

sottrazione

$$\begin{array}{r} 1\ 1\ 0\ 0\ - \\ 0\ 0\ 1\ 1\ = \\ \hline 1\ 0\ 0\ 1 \end{array}$$

moltiplicazione

$$\begin{array}{r} 1\ 1\ 0\ 1\ \times \\ 1\ 0\ 1\ 1\ = \\ \hline 1\ 1\ 0\ 1\ + \\ 1\ 1\ 0\ 1\ + \\ 0\ 0\ 0\ 0\ + \\ 1\ 1\ 0\ 1\ + \\ \hline 1\ 0\ 0\ 0\ 1\ 1\ 1\ 1 \end{array}$$

Esercizio: trasformando in decimale controllare se i risultati sono corretti

Numeri in base 8 (ottali)

- Le cifre: [0, 1, 2, 3, 4, 5, 6, 7]

- **17**_{otto} = ?_{dieci}

$$1_{\text{otto}} = 1_{\text{dieci}}$$

$$7_{\text{otto}} = 7_{\text{dieci}}$$

$$17_{\text{otto}} = (1 \times 8^1 + 7 \times 8^0)_{\text{dieci}} = (8 + 7)_{\text{dieci}} = 15_{\text{dieci}}$$

- **372**_{otto} = ?_{dieci}

$$372_{\text{otto}} = (3 \times 8^2 + 7 \times 8^1 + 2 \times 8^0)_{\text{dieci}} = (3 \times 64 + 56 + 2)_{\text{dieci}} = 250_{\text{dieci}}$$

Numeri in base 16 (esadecimale)

- Le cifre: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F]

- **7D2_{sedici}** = ?_{dieci}

$$7_{\text{sedici}} = 7_{\text{dieci}} \quad D_{\text{sedici}} = 13_{\text{dieci}} \quad 2_{\text{sedici}} = 2_{\text{dieci}}$$

$$\mathbf{7D2_{\text{sedici}}} = (7 \times 16^2 + 13 \times 16^1 + 2 \times 16^0)_{\text{dieci}} = (7 \times 256 + 208 + 2)_{\text{dieci}} = (1792 + 208 + 2)_{\text{dieci}} = \mathbf{2002_{\text{dieci}}}$$

- **FA_{sedici}** = ?_{dieci}

$$F_{\text{sedici}} = 15_{\text{dieci}} \quad A_{\text{sedici}} = 10_{\text{dieci}}$$

$$\mathbf{FA_{\text{sedici}}} = (15 \times 16^1 + 10 \times 16^0)_{\text{dieci}} = (240 + 10)_{\text{dieci}} = \mathbf{250_{\text{dieci}}}$$

I primi 16 numeri in base 10, 2, 8, e 16

Sistema di numerazione			
decimale	binario	ottale	esadecimale
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Perché le basi 2, 8 e 16?

- Come già visto, la rappresentazione **binaria** ha *motivazioni di tipo tecnologico*
- Le rappresentazioni **ottali** ed **esadecimali** sono utili per rappresentare sinteticamente i valori binari
- E' facile convertire un numero in base 2 in un numero in base 8 o 16
- Le cifre binarie si possono raggruppare **a 3 a 3** e poi codificare con **numeri ottali**
- Le cifre binarie si possono raggruppare **a 4 a 4** e poi codificare con **numeri esadecimali**

Conversione binario \Rightarrow ottale

Tabella di conversione

000 _{due}	0 _{otto}
001 _{due}	1 _{otto}
010 _{due}	2 _{otto}
011 _{due}	3 _{otto}
100 _{due}	4 _{otto}
101 _{due}	5 _{otto}
110 _{due}	6 _{otto}
111 _{due}	7 _{otto}

11 110 110 100.001_{due} = 3 6 6 4.1_{otto}



Separazione a gruppi di tre cifre binarie a partire dalla meno significativa per la parte intera, dalla più significativa per la parte frazionaria [dalla virgola!]

Nel gruppo “più significativo” della parte intera si possono aggiungere degli zeri a sinistra, nel “meno significativo” della frazionaria zeri a destra

Conversione binario \Rightarrow esadecimale

Tabella di conversione

0000 _{due}	0 _{sedici}
0001 _{due}	1 _{sedici}
0010 _{due}	2 _{sedici}
0011 _{due}	3 _{sedici}
0100 _{due}	4 _{sedici}
0101 _{due}	5 _{sedici}
0110 _{due}	6 _{sedici}
0111 _{due}	7 _{sedici}
1000 _{due}	8 _{sedici}
1001 _{due}	9 _{sedici}
1010 _{due}	A _{sedici}
1011 _{due}	B _{sedici}
1100 _{due}	C _{sedici}
1101 _{due}	D _{sedici}
1110 _{due}	E _{sedici}
1111 _{due}	F _{sedici}

$$111 \text{ } 1011 \text{ } 0100_{\text{due}} = 7 \text{ B } 4_{\text{sedici}}$$



*Si procede nello stesso modo,
ma separando le cifre a gruppi
di 4 anziché di 3*

ERRORE TIPICO:

Convertire in notazione ottale il numero binario 10111010.11

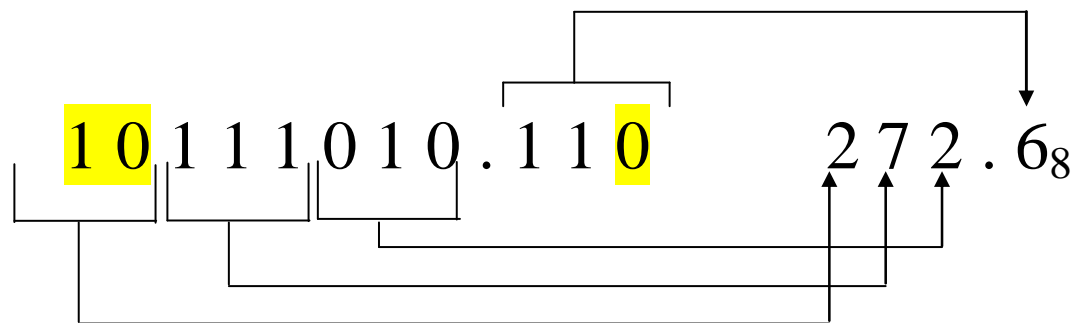
10111010.11
5 6 2 . 3

Invece $562.3_8 = 5 \cdot 64 + 6 \cdot 8 + 2 + 3/8 = 370.375$ che sicuramente non può essere rappresentato con una parte intera di soli 8 bit!!!

PARTIRE SEMPRE DAL PUNTO DI RADICE, EVENTUALMENTE COMPLETANDO LE CIFRE CON DEGLI ZERI PER OTTENERE LE TERNE:

xxx xxx . yyy yyy ...

L'esercizio quindi va risolto così:

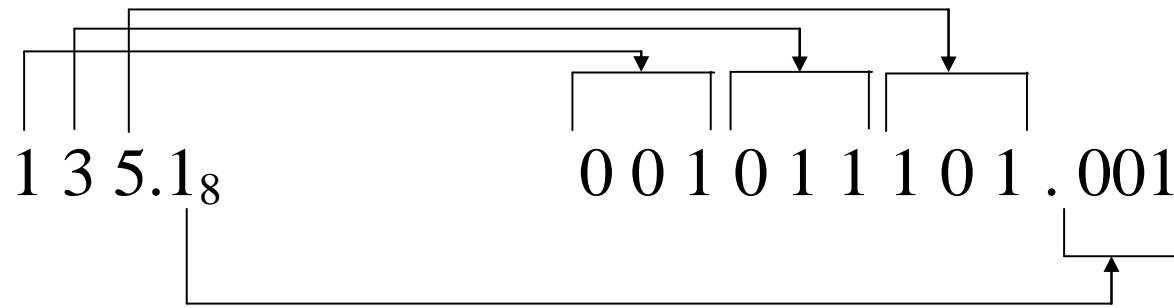


infatti risulta $272.6_8 = 2 \cdot 64 + 7 \cdot 8 + 2 + 6/8 = 186.75$

e $10111010.11_2 = 128 + 32 + 16 + 8 + 2 + 0.5 + 0.25 = 186.75$

ESERCIZIO

Convertire in binario il numero in notazione ottale 135.1_8

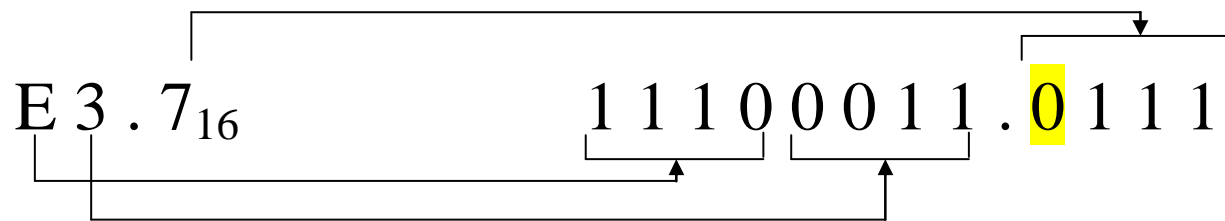
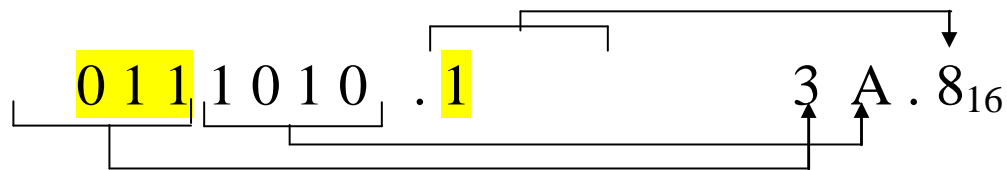


ERRORE TIPICO: CONVERTIRE IN

$001\ 011\ 101\ .\ 1$

infatti $0.1_8 = 1/8 = 0.125$ mentre $0.1_2 = 1/2 = 0.5$

Altri esempi di conversione esadecimale-binario



ATTENZIONE!!! PARTIRE SEMPRE DAL PUNTO DECIMALE!!!
XXXX . XXXX XXXX

Esercizio proposto

- Dato il numero binario $001010110111_{\text{due}}$ convertirlo in un numero ottale e poi in un numero esadecimale
- Convertire il numero ottale in numero decimale
- **Numero ottale:** $001\ 010\ 110\ 111 \rightarrow 1267_{\text{otto}}$
- **Numero esadecimale:** $0010\ 1011\ 0111 \rightarrow 2B7_{16}$
- **Numero decimale:** $1267_{\text{otto}} = (1 \times 8^3 + 2 \times 8^2 + 6 \times 8^1 + 7 \times 8^0)_{\text{dieci}} = (512 + 128 + 48 + 7)_{\text{dieci}} = 695_{\text{dieci}}$

Esercizi

- Se la base considerata è $b = 4$, quali sono le cifre utilizzate per comporre i numeri?
- $[0,1,2,3]$
- Convertire il numero $(1320)_{\text{quattro}}$ nel corrispondente numero in base 10
- $1320_{\text{quattro}} = (1 \times 4^3 + 3 \times 4^2 + 2 \times 4^1 + 0 \times 4^0)_{\text{dieci}} = (64 + 48 + 8)_{\text{dieci}} = 120_{\text{dieci}}$
- Qual è il numero massimo rappresentabile in base 3 con quattro cifre (espresso in base 3)?
- 2222_{tre}

Esercizi proposti

- Convertire in formato *decimale* i seguenti numeri *binari*:
 - 11, 101011, 1100, 111111, 10101010
- Convertire in formato *decimale* i seguenti numeri *ottali*:
 - 12, 23, 345, 333, 560
- Convertire in formato *decimale* i seguenti numeri *esadecimali*:
 - 12, DAB, 15D, FFFF, 51A
- Convertire in *binario* i seguenti numeri *decimali*:
 - 45, 234, 67, 83, 972
- Convertire in *ottale* e in *esadecimale* i *numeri binari* ottenuti dalla conversione dei numeri decimali di cui al punto precedente