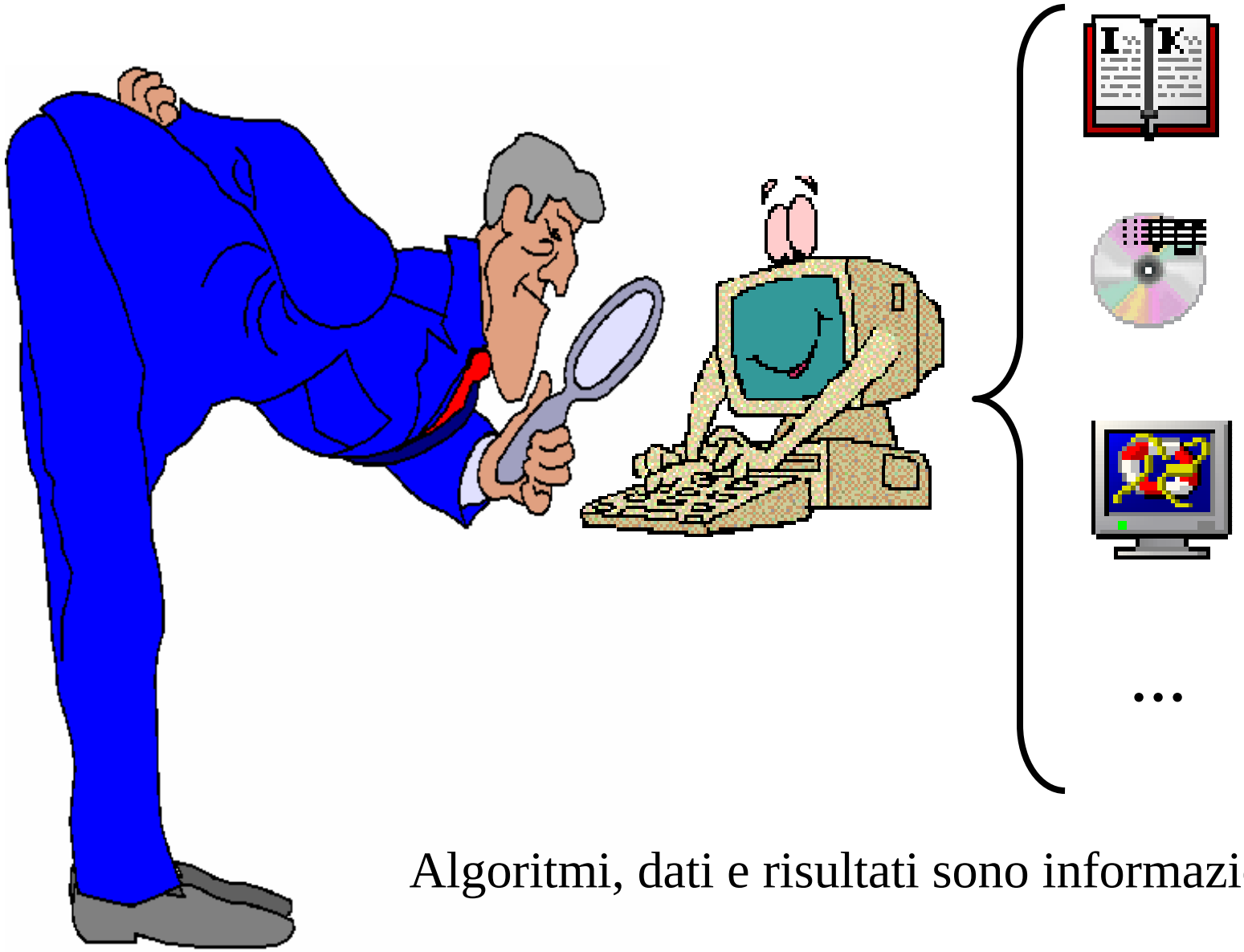
Informazione binaria:
- codici binari, notazione binaria/ottale/esadecimale -

Percorso di Preparazione agli Studi di Ingegneria

Università degli Studi di Brescia

Docente: Massimiliano Giacomin

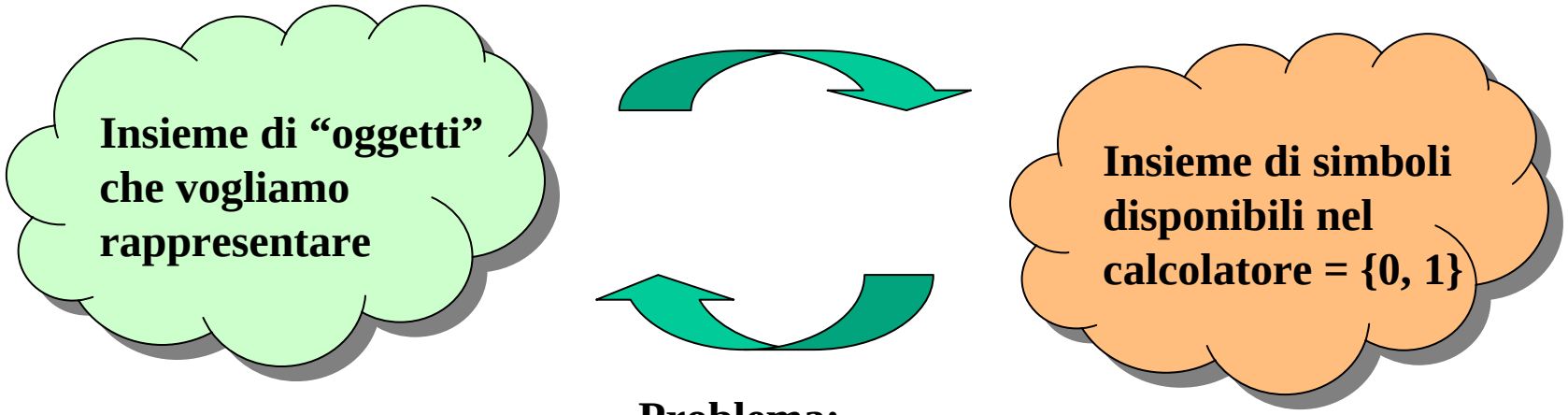


Algoritmi, dati e risultati sono informazioni…

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)

Esempio: una possibile codifica dei mesi dell'anno

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

1 bit ➔ 2 gruppi

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

2 bit ➔ 4 gruppi

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

3 bit ➔ 8 gruppi

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

4 bit ➔ 16 gruppi… mancano 4 configurazioni!

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

Esempio: altre possibili codifiche dei mesi dell'anno

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

Gennaio	**0000**	Febbraio	**0001**
Marzo	**0010**	Aprile	**0011**
Maggio	**0100**	Giugno	**0101**
Luglio	**0110**	Agosto	**0111**
Settembre	**1000**	Ottobre	**1001**
Novembre	**1010**	Dicembre	**1011**

Gennaio	**1011**	Febbraio	**0001**
Marzo	**0010**	Aprile	**0011**
Maggio	**0100**	Giugno	**0101**
Luglio	**0110**	Agosto	**0111**
Settembre	**1000**	Ottobre	**1001**
Novembre	**1010**	Dicembre	**0000**

e potremmo usare
più di 4 bit se volessimo

Un altro esempio: codifica BCD delle cifre decimali

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

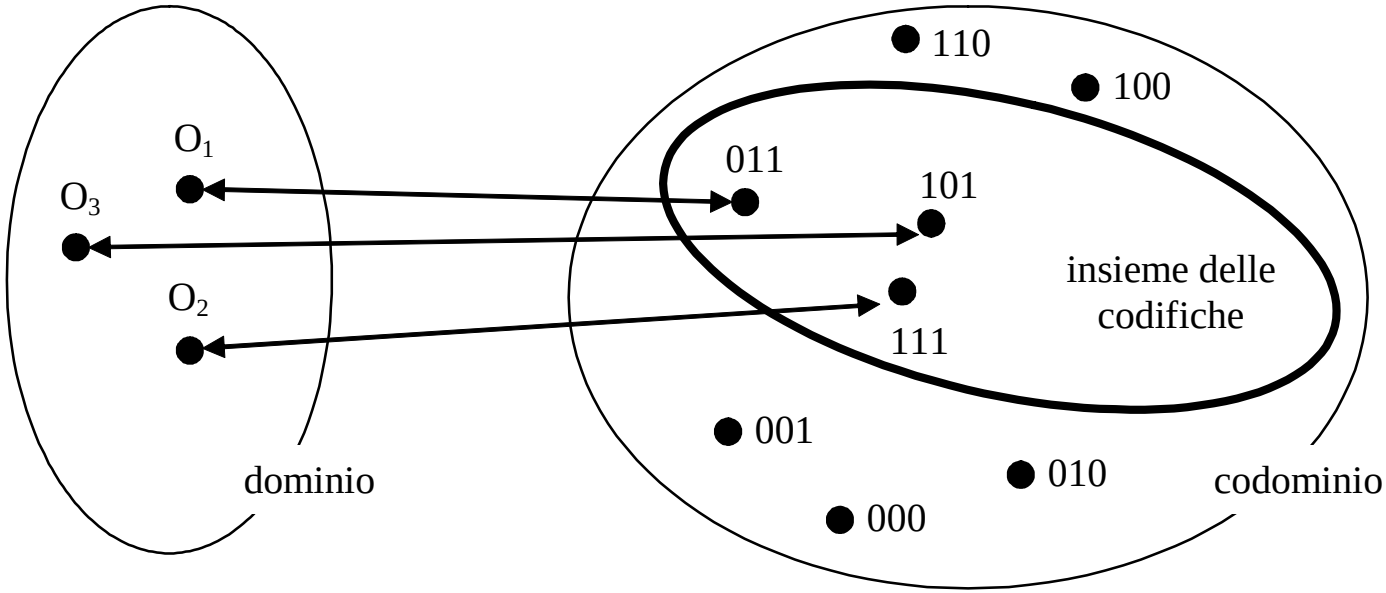
Codice binario a *n* bit

- Funzione:
  - dominio: un insieme di oggetti da rappresentare
  - codominio: insieme di tutte le possibili sequenze di n bit

  totale e iniettiva, ovvero biunivoca tra il dominio e la sua immagine

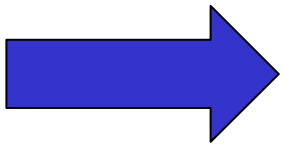
- L'immagine della funzione è detta insieme delle codifiche

- Esempio di codice binario a 3 bit:



Un po' di conti

- Se dispongo di sequenze di ***n*** bit,
  qual è il numero **N** di oggetti che posso codificare?
  - Se $n = 1$
    - Posso codificare due oggetti (N=2): ad esempio, al primo assegno la codifica '0' e al secondo assegno la codifica '1'
  - Se $n = 2$
    - Posso codificare N=4 oggetti: 00, 01, 10, 11
  - Se $n = 3$
    - Posso codificare N=8 oggetti: 000, 001, 010, 011, 100, 101, 110, 111



$$N = 2^n$$

- Se ho N oggetti da codificare (con due simboli 0 e 1): qual è il minimo numero *n* di bit necessario?

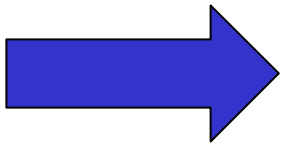
  - Se N = 128, ho bisogno di 7 bit perché $2^7=128$

    $n = \lceil \log_2 128 \rceil = 7$

  - Se N = 129 ho bisogno di 1 bit in più!

    $n = \lceil \log_2 129 \rceil = 8$

    Ottengo uno "spreco" di configurazioni, perché con 8 bit posso codificare fino a 256 elementi



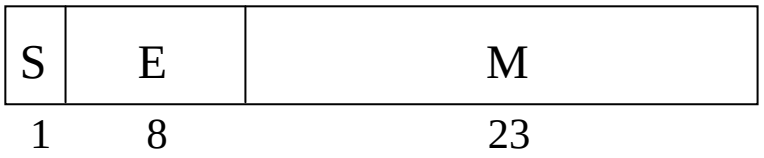
$$\mathbf{n = \lceil \log_2 N \rceil}$$

Formato

In molti casi, una rappresentazione è definita mediante un ***formato***

- il numero di bit *n* a disposizione
- i ***campi*** in cui sono suddivisi i bit:
  - quanti
  - in che ordine
  - quanto è lungo ciascun campo
  - cosa rappresenta ciascun campo

Esempio per la rappresentazione di numeri razionali (n=32)



Vedremo poi il significato…

Tipologie di codici

Nel seguito vedremo tipologie di rappresentazioni diverse:

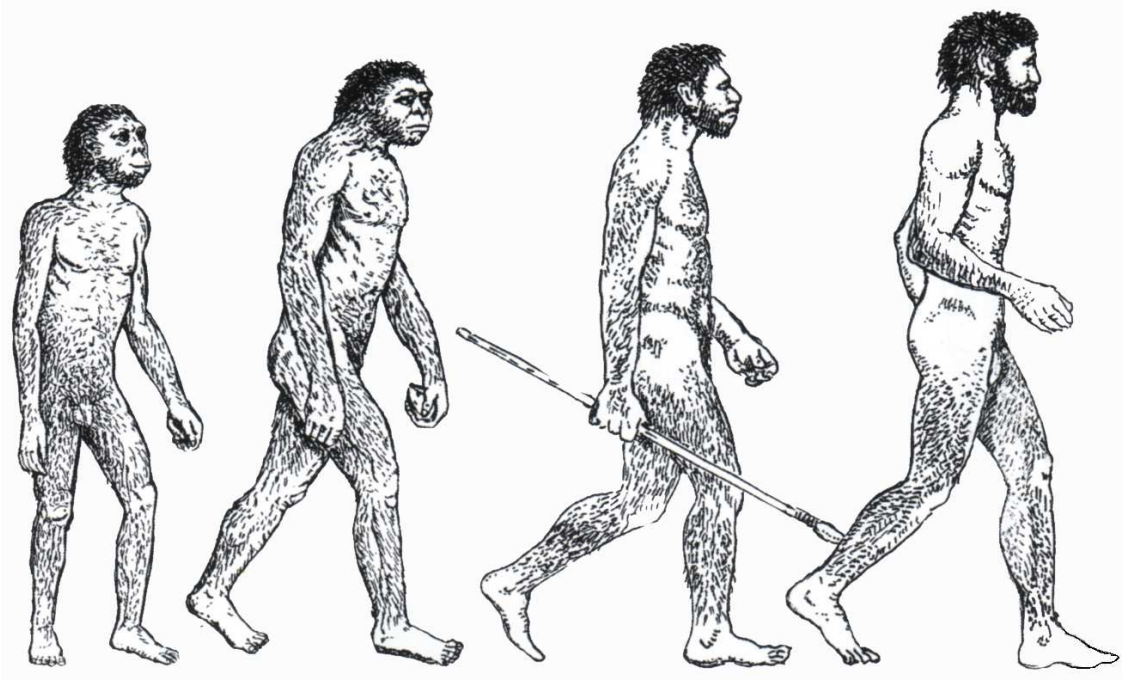
- Senza assumere limitazioni sul numero di bit a disposizione: per numeri [notazione binaria, ovvero posizionale con base 2]
- Disponendo di un numero di bit limitato:
  - numeri naturali
  - interi relativi [valore assoluto e segno, complemento a due]
  - "reali" [virgola fissa e virgola mobile]
  - 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)

Perché contiamo “in base 10”



- moltiplicare e dividere per 10
- e se avessimo due dita?

Sistema di numerazione posizionale

base = b insieme di cifre usate: {0, …, b-1}

Rappresentazione:

$a_k\ a_{k-1} \ldots a_0 \bullet a_{-1}\ a_{-2} \ldots a_{-h}$

Numero rappresentato:

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

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]

- La base si indica con un pedice, p.es. 10.1001_2

Notazione binaria

base = 2 cifre: 0, 1

- Numeri espressi nella forma

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

il cui "valore" è

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

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 ⇒ decimale

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

NOTA DI "PRATICA"

Conviene partire dall'ultima cifra della parte intera (il cui contributo eventuale vale 1), sapendo che spostandosi di una posizione verso sinistra il contributo raddoppia, spostandosi verso destra il contributo si dimezza

$$101011.1011_2$$

$$\downarrow$$

$$1 + 2 + 8 + 32 + 0.5 + 0.125 + 0.0625$$

$$= 43.6875_{10}$$

Esercizi individuali

- Esempio:

  $1101.1_{due} = 1x2^3 + 1x2^2 + 0x2^1 + 1x2^0 + 1x2^{-1} =$

  $= 8_{dieci} + 4_{dieci} + 0_{dieci} + 1_{dieci} + 0.5_{dieci} = 13.5_{dieci}$

- Altri esempi:

  $10101.01_{due} = 1x2^4 + 0x2^3 + 1x2^2 + 0x2^1 + 1x2^0 + 0x2^{-1} + 1x2^{-2}$

  $= 16 + 4 + 1 + 0.25$

  $= 21.25_{dieci}$

  $110010.001_{due} = 1x2^5 + 1x2^4 + 0x2^3 + 0x2^2 + 1x2^1 + 0x2^0 + \ldots + 1x2^{-3} =$

  $= 32 + 16 + 2 + 0.125$

  $= 50.125_{dieci}$

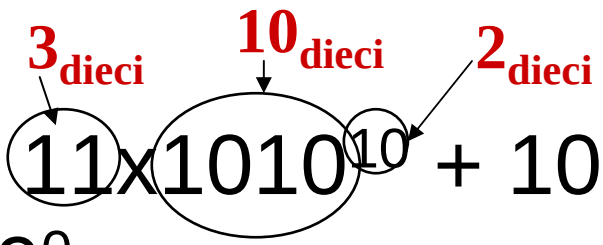
  $1000001_{due} = 1x2^6 + 0x2^5 + 0x2^4 + 0x2^3 + 0x2^2 + 0x2^1 + 1x2^0$

  $= 64 + 1$

  $= 65_{dieci}$

Conversione decimale ⇒ binario

- Usare la definizione è complesso!

- Esempio: 

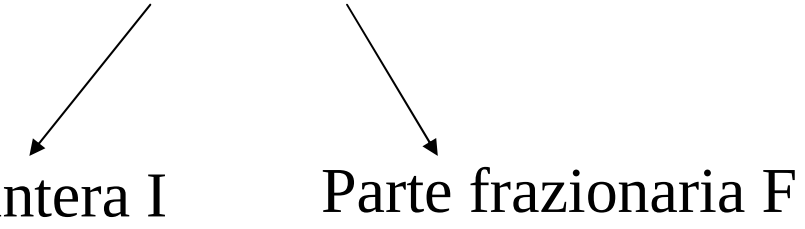
$345_{dieci} = 11x1010^{10} + 100x1010^{01} + 101x1010^{0} = \ldots$

… 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

ES: con 57.6875_{dieci}

convertiamo separatamente 57 e 0.6875



REGOLA PRATICA PER CONVERTIRE LA PARTE INTERA

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

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

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

.....

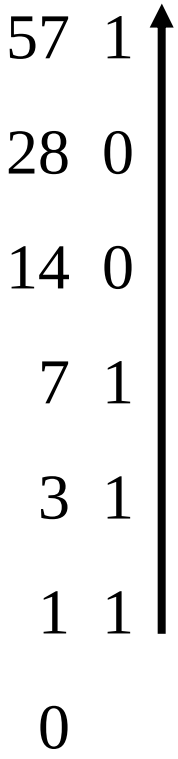
Esempio

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

/2 :	$(1x2^3 + 0x2^2 + 1x2^1) + 0x2^0$	con resto 1
/2 :	$(1x2^2 + 0x2^1) + 1x2^0$	con resto 0
/2 :	$(1x2^1) + 0x2^0$	con resto 1
/2:	$1x2^0$	con resto 0
/2:	0	con resto 1

Esempio di conversione da decimale a binario

57_{10}



Si legge dal basso verso l'alto !!!

Risultato = 111001_{due}

ERRORE TIPICO:

considerare la prima cifra ottenuta come la più significativa:

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

otterrei 100111 che vale 39!

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 57_{10}

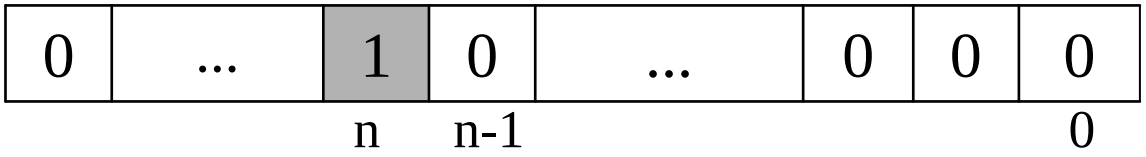
$57 = 111001_2$

$[32 + 16 + 8 + 1]$

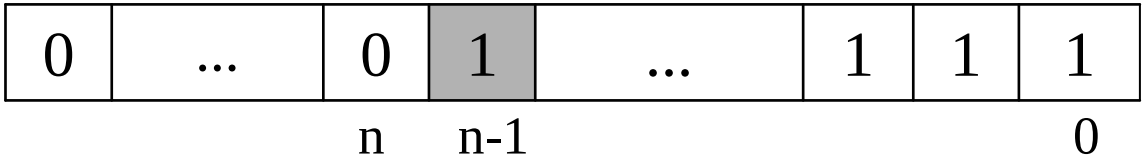
$[2^5 + 2^4 + 2^3 + 2^0]$

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

ESEMPI STANDARD



2^n



2^{n-1}

Esempi con 8 cifre:

1	0	0	0	0	0	0	0	128
0	1	1	1	1	1	1	1	127
1	1	1	1	1	1	1	1	255

REGOLA PRATICA PER CONVERTIRE LA PARTE FRAZIONARIA

$$F = \overset{?}{C}_{-1}*2^{-1} + \overset{?}{C}_{-2}*2^{-2} + \ldots\ldots + \overset{?}{C}_{-n}*2^{-n}$$

$$F*2 = C_{-1} + C_{-2}*2^{-1} + \ldots\ldots + C_{-n}*2^{-(n-1)}$$ la parte intera è C_{-1}

$$(F*2 - C_{-1}) * 2 = C_{-2} + \ldots\ldots + C_{-n}*2^{-(n-2)}$$ la parte intera è C_{-2}

……

- Esempio: 0.587_{dieci} ➔ binario?
  - 0.587 x 2 = 1.174: p.f. 0.174, parte intera **1 (cifra più significativa)**
  - 0.174 x 2 = 0.348: p.f. 0.348, parte intera **0**
  - 0.348 x 2 = 0.696: p.f. 0.696, parte intera **0**
  - 0.696 x 2 = 1.392: p.f. 0.392, parte intera **1**
  - 0.392 x 2 = 0.784: p.f. 0.784, parte intera **0**
  - 0.784 x 2 = 1.568: p.f. 0.568, parte intera **1**
  - ….

Si ottiene $\mathbf{0.10010_{due}}$ con 5 cifre binarie dopo la virgola, oppure $\mathbf{0.100101_{due}}$ con 6 cifre binarie dopo la virgola, oppure…

➔In questo caso c'è comunque un'approssimazione

TORNIAMO ALL'ESEMPIO: convertire 57.6875 in binario

Parte intera

$57 = 111001_2$ (vedi prima)

Parte frazionaria

0.6875	1
0.375	0
0.75	1
0.5	1
0	

$\Rightarrow 57.6875_{10} = 111001.1011_2$

ERRORE TIPICO:

considerare la prima cifra ottenuta come la meno significativa:

0.6875	1
0.375	0
0.75	1
0.5	1
0	

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

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**

MOLTIPLICAZIONE e DIVISIONE PER POTENZE DI 2

- moltiplicazione per 2^n:
  spostamento della virgola a destra di *n* posizioni
  (con eventuale aggiunta di zeri alla fine del numero)

- divisione per 2^n (o moltiplicazione per 2^{-n}):
  spostamento della virgola a sinistra di *n* posizioni

ARITMETICA BINARIA: ADDIZIONE

- è l'unica che ci servirà concretamente
- valgono le stesse regole e proprietà dell'addizione in base 10
- si può svolgere "in colonna": si sommano le cifre corrispondenti, se il numero binario ottenuto ha due cifre la più significativa è il riporto

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

NB: $1+1= 2_{10}=10_2$

ESEMPIO

$$\begin{array}{r} 1001101.10110 + \\ 1101100.11010 = \\ \hline 10111010.10000 \end{array}$$

Numeri in base 8 (ottali)

- Base: 8
- Le cifre: 0, 1, 2, 3, 4, 5, 6, 7

ES:

$573.671_8 =$

$5*8^2 + 7*8 + 3 + 6*8^{-1} + 7*8^{-2} + 1*8^{-3}$

NB: per convertire da decimale a ottale, tipicamente si passa attraverso i numeri binari (vedi poi)

Numeri in base 16 (esadecimali)

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

  10 11 12 13 14 15

ES: $97A.6E1_{16}$ =

$9*16^2 + 7*16 + 10 + 6*16^{-1} + 14*16^{-2} + 1*16^{-3}$

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

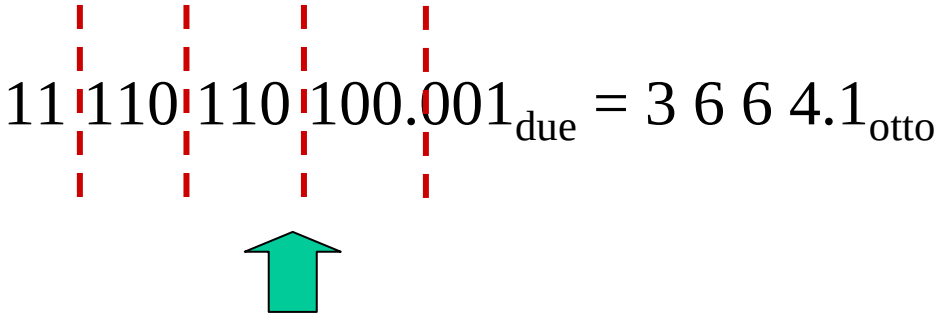
Perché le basi 2, 8 e 16?

- 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 ⇒ 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}



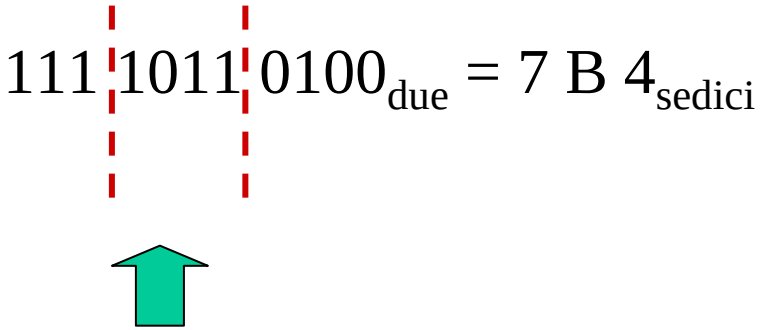
Separazione a gruppi di tre cifre binarie a partire dalla meno significativa per la parte intera, dalla più significativa per la parte frazionaria [dalla virgola!]

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

Conversione binario ⇒ 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}



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

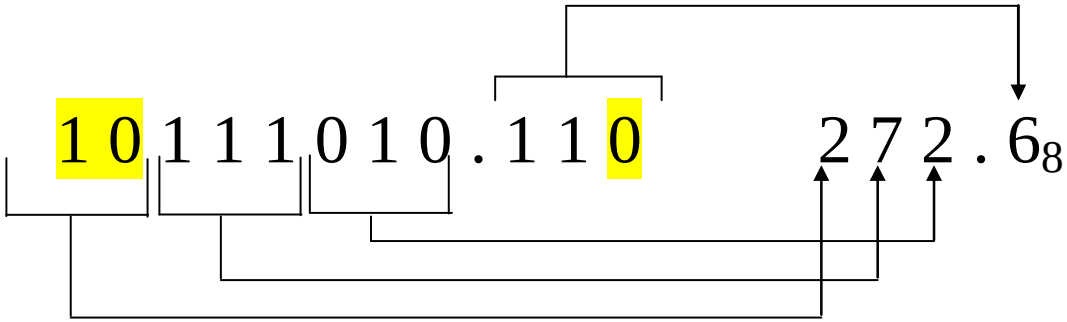
101	110	10.	11
5	6	2 .	3

Invece $562.3_8 = 5*64 + 6*8 + 2 + 3/8 = 370.375$ 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:

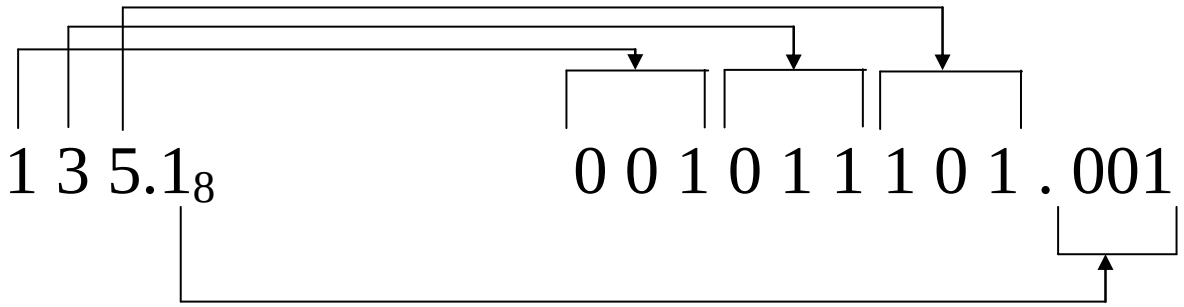


infatti risulta $272.6_8 = 2*64 + 7*8 + 2 + 6/8 = 186.75$

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

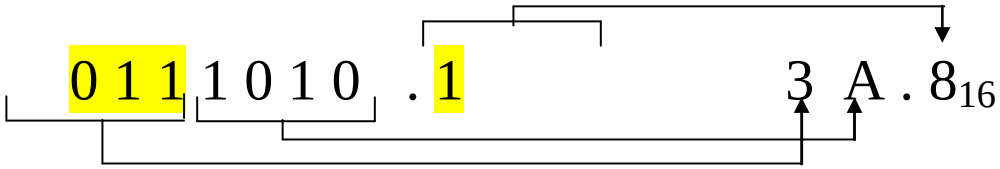


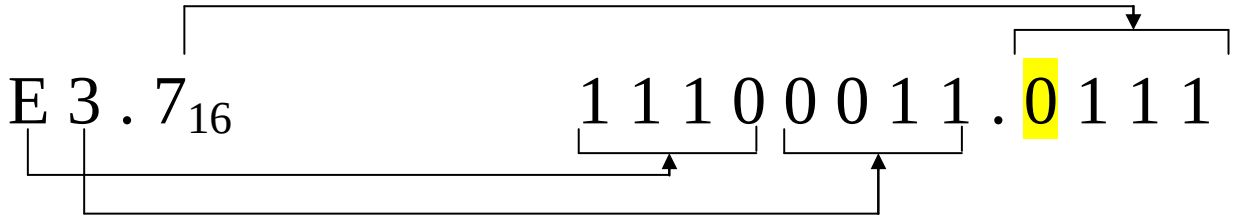
ERRORE TIPICO: CONVERTIRE IN

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

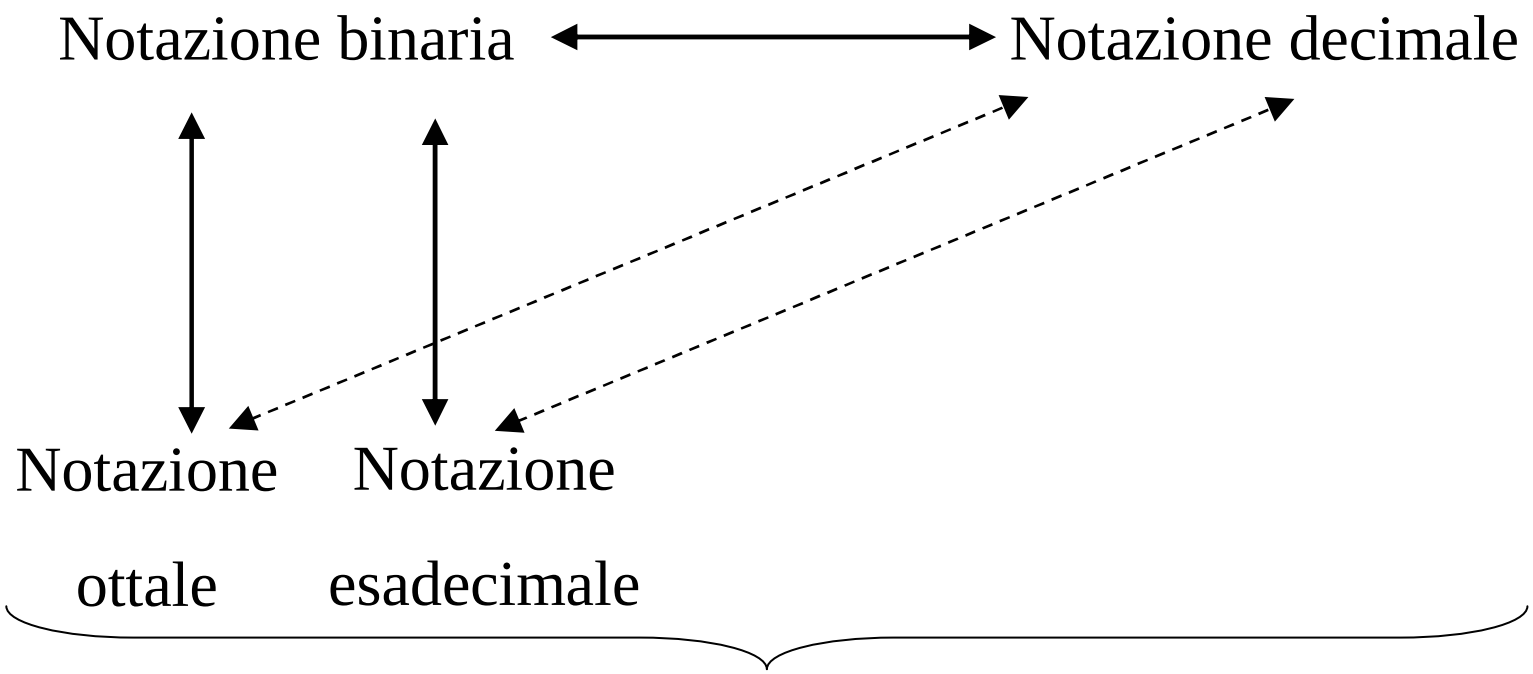




ATTENZIONE!!! PARTIRE SEMPRE DAL PUNTO DECIMALE!!!

xxxx . xxxx xxxx

CONVERSIONI ATTRAVERSO LA NOTAZIONE BINARIA



Consideriamo numeri interi e frazionari, (per ora)
senza limitazione sul numero di cifre a disposizione

Esercizio proposto con risposta

- Dato il numero binario 001010110111_{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 ➔ 1267_{otto}
- **Numero esadecimale**: 0010 1011 0111 ➔ $2B7_{16}$
- **Numero decimale**: $1267_{otto} = (1x8^3 + 2x8^2 + 6x8^1 + 7x8^0)_{dieci} = (512 + 128 + 48 + 7)_{dieci} = 695_{dieci}$

Esercizi proposti con risposte

- Se la base considerata è b = 4, quali sono le cifre utilizzate per comporre i numeri?
- [0,1,2,3]

- Convertire il numero $(1320)_{quattro}$ nel corrispondente numero in base 10
- $1320_{quattro} = (1x4^3 + 3x4^2 + 2x4^1 + 0x4^0)_{dieci} = (64 + 48 + 8)_{dieci} = 120_{dieci}$

- 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