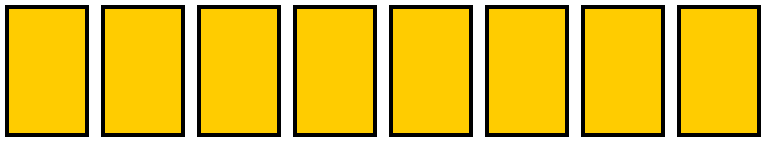
Breve ripasso di aritmetica binaria (corso di reti di calcolatori)

Luciano Bononi

Email: luciano.bononi@unibo.it

La rappresentazione dei dati sul calcolatore

- **i bit possono essere considerati in sequenza (in memoria)**
  - sequenza di 8 bit = 1 Byte



0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1
0 0 0 0 0 0 1 0
0 0 0 0 0 0 1 1

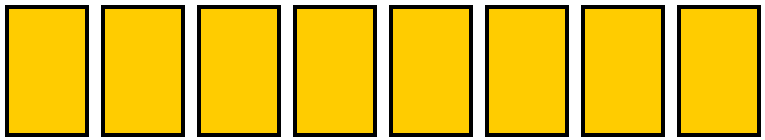
1 1 1 1 1 1 1 0
1 1 1 1 1 1 1 1

2^n possibili sequenze diverse da n bit

n = numero di bit considerati

La rappresentazione dei dati sul calcolatore

- **Dato un byte come possiamo associarvi i simboli o valori?**
  - sequenza di 8 bit = 1 Byte (primo esempio poco utile)

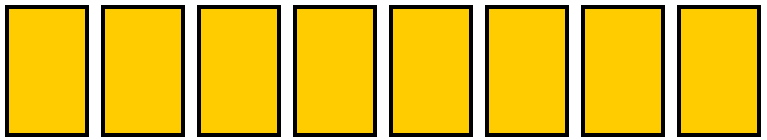


0	0	0	0	0	0	0	1	= valore 1
0	0	0	0	0	0	1	0	= valore 2
0	0	0	0	0	1	0	0	= valore 3
0	0	0	0	1	0	0	0	= valore 4
0	0	0	1	0	0	0	0	= valore 5
0	0	1	0	0	0	0	0	= valore 6
0	1	0	0	0	0	0	0	= valore 7
1	0	0	0	0	0	0	0	= valore 8

Solo valori da 1 a 8 ?
non sfrutto tutte le
possibili combinazioni!!!

La rappresentazione dei dati sul calcolatore

- **Dato un byte come possiamo associarvi i simboli o valori?**
  - sequenza di 8 bit = 1 Byte

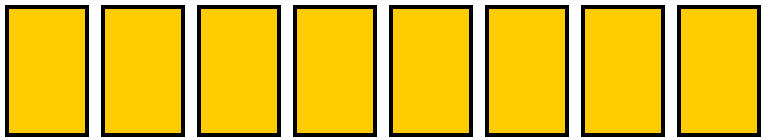


Idea: sfrutto tutte le possibili combinazioni diverse di 8 bit
256 valori [0..255]

0	0	0	0	0	0	0	0	= valore 0
0	0	0	0	0	0	0	1	= valore 1
0	0	0	0	0	0	1	0	= valore 2
0	0	0	0	0	0	1	1	= valore 3
0	0	0	0	0	1	0	0	= valore 4
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	
1	1	1	1	1	1	1	0	= valore 254
1	1	1	1	1	1	1	1	= valore 255

La rappresentazione dei dati sul calcolatore

- **Dato un byte come possiamo associarvi i simboli o valori?**
  - sequenza di 8 bit = 1 Byte



Idea: sfrutto tutte le possibili combinazioni diverse di 8 bit
256 simboli

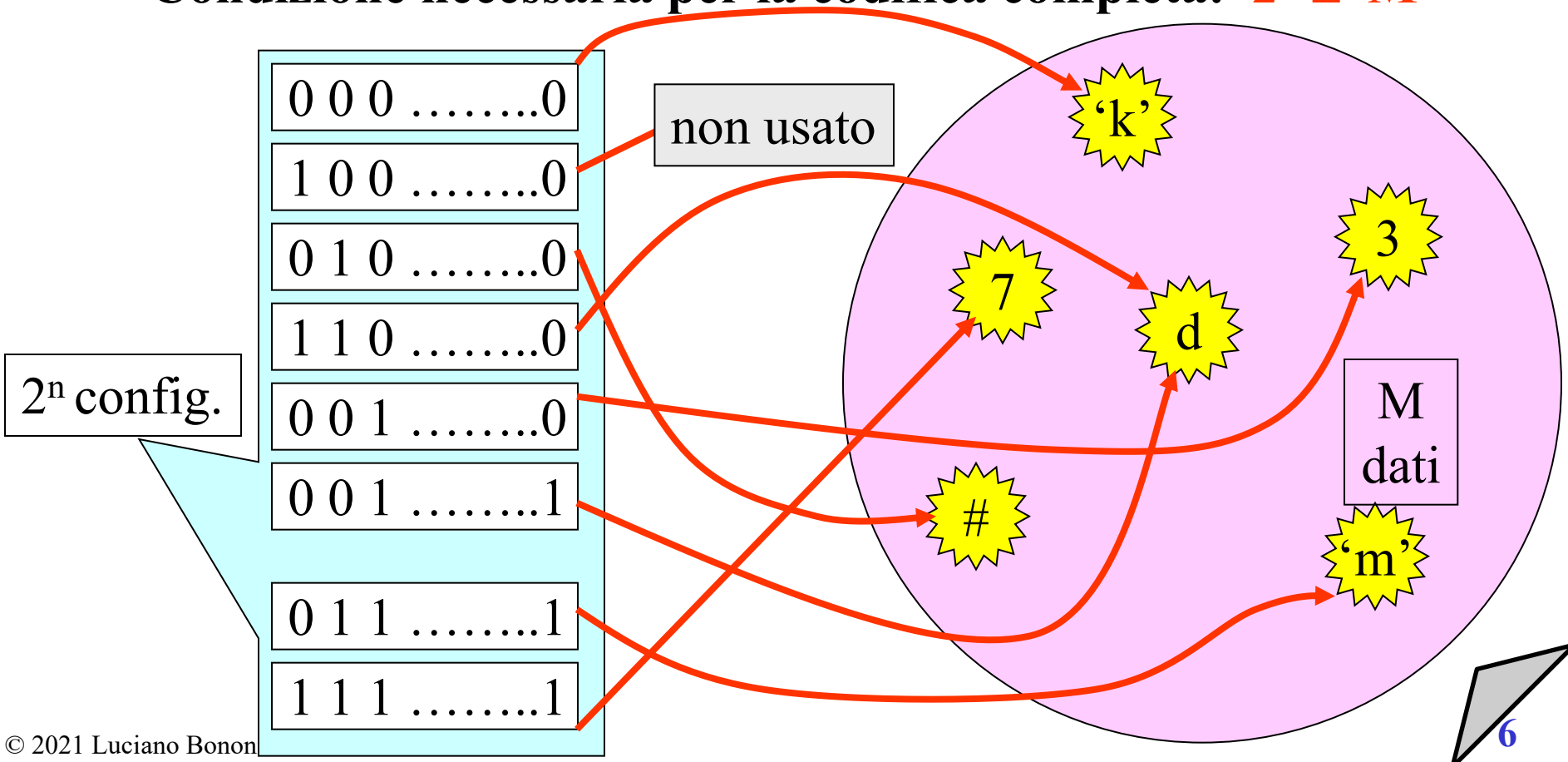
0	0	0	0	0	0	0	0	= simbolo *A*
0	0	0	0	0	0	0	1	= simbolo B
0	0	0	0	0	0	1	0	= simbolo *C*
0	0	0	0	0	0	1	1	= simbolo E
0	0	0	0	0	1	0	0	= simbolo F
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	
1	1	1	1	1	1	1	0	= simbolo %
1	1	1	1	1	1	1	1	= simbolo $

Codici binari: convenzioni per rappresentare info.

Funzioni dall'insieme delle 2^n configurazioni di n bit ad un insieme di M informazioni o dati

(*valori, simboli, istruzioni*, ecc.).

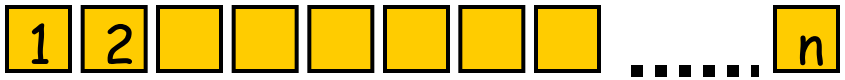
Condizione necessaria per la codifica completa: $2^n \geq M$



Codici binari: quanti sono?

La scelta di un codice è condivisa da sorgente e destinazione ed ha due gradi di libertà:

- **il numero di bit n (qualsiasi, a patto che sia $2^n \geq M$)**



- **l'associazione tra configurazioni e informazioni; a parità di n e di M le associazioni possibili sono**

$$C = 2^n! / (2^n - M)!$$

n = 1, M = 2	C = 2 (logica pos. e neg.)
n = 2, M = 4	C = 24
n = 3, M = 8	C = 64.320
n = 4, M = 10	C = 29.000.000.000

Sistemi di numerazione

Posizionali

• il valore di un simbolo dipende dalla posizione che esso occupa all'interno della configurazione, seguendo una legge nota. I vari sistemi di numerazione posizionale differiscono per la scelta della base B. La base B indica il numero di simboli usati.

- **decimale B=10**, **binario B=2**, **ottale B=8**, **esadecimale B=16**

Non posizionali

Il valore di un simbolo non dipende dalla posizione che esso occupa all'interno della configurazione. (es: Numeri Romani)

Interpretazione (funzione valore)

• Nei sistemi posizionali, i simboli di una configurazione possono essere interpretati come i coefficienti del seguente polinomio [1] (detto funzione Valore)

$$V = \sum_{i=-m}^{n-1} d_i \cdot B^i$$

B = base

d_i = i-esima cifra $\in$ [0..B-1]

n = numero di cifre parte intera

m = numero di cifre parte frazionaria

La virgola e' posta tra le cifre di posizione 0 e –1.

Interpretazione (funzione valore)

Esempio: sistema decimale

Il numero 245.6 decimale può essere rappresentato come segue:

B = 10	base
n=3	numero cifre parte intera
m=1	numero cifre parte frazionaria
d = {2,4,5,6}	insieme delle cifre

cifra	2	4	5	6
posizione	2	1	0	-1
peso	10^{2}	10^{1}	10^{0}	10^{-1}

$$V = \sum_{i=-m}^{n-1} d_i \cdot B^i \qquad = 2 \cdot 10^{2} + 4 \cdot 10^{1} + 5 \cdot 10^{0} + 6 \cdot 10^{-1} = 245.6$$

Esempio di interpretazione valore binario naturale

Esempio: Quale è il valore decimale corrispondente al numero binario 1101.010_2 ?

$cifra_2$	1	1	0	1	.	0	1	0...
peso	2^3	2^2	2^1	2^0	.	2^{-1}	2^{-2}	2^{-3}
valore	1•8	1•4	0•2	1•1	.	0•1/2	1•1/4	0•1/8

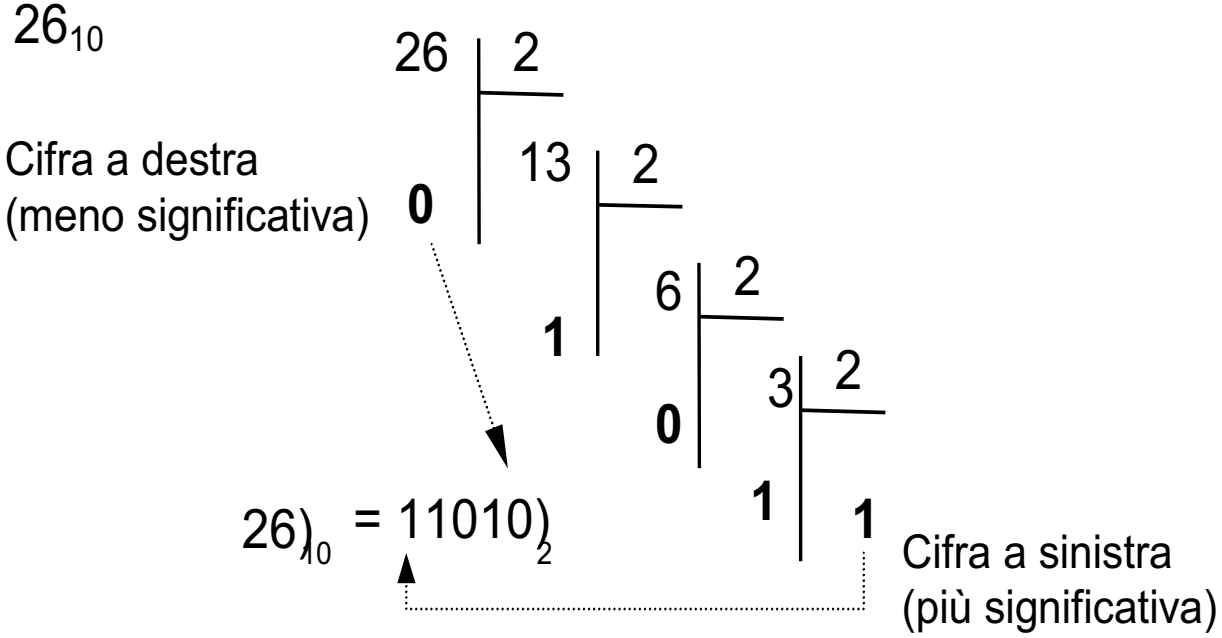
$$1101.010_2 = 1\bullet 2^3 + 1\bullet 2^2 + 0\bullet 2^1 + 1\bullet 2^0 + 0\bullet 2^0 + 1\bullet 2^0 = 13.25_{10}$$

Metodo della divisione: Base 10 -> Base B

Es. base 2:

Per valori interi positivi:

Per ottenere il valore in base B, di un numero intero codificato nel sistema decimale, si procede utilizzando un metodo iterativo di successive divisioni per la base: al termine del procedimento i resti delle divisioni, dall'ultimo al primo, rappresentano il valore iniziale in base B.



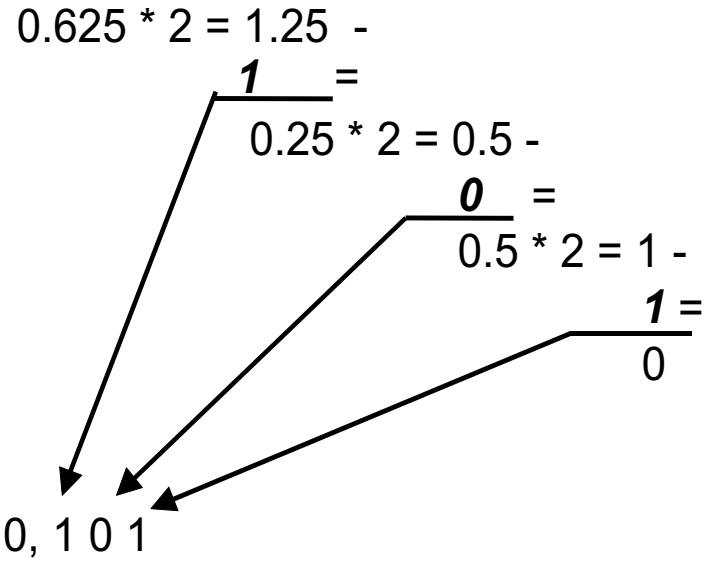
Metodo della divisione: Base 10 -> binario naturale

Es. base 2

Per valori con cifre decimali: si separa la parte intera da quella frazionaria, La parte intera si calcola come nel caso precedente La parte frazionaria si ottiene come segue:

1. Si moltiplica la parte frazionaria per 2
2. Se il numero ottenuto è maggiore di 1, si sottrae 1 e si considera come prima cifra dopo la virgola un ‘1’.
3. Se invece il numero è nella forma 0,….. => la cifra da inserire è uno ‘0’.
4. Si ripete dal passo 1 fino a che il numero di partenza non è zero.

Esempio: $0.625_{10} = 0.101_{2}$



Metodo veloce (pratico): Base 10 -> binario naturale

Con la pratica, solo dopo avere bene compreso la metodologia è possibile usare un metodo alternativo veloce.

1. Dato il numero N in base 10 iniziale (es. 349), si deve cercare a mente velocemente la più alta potenza del 2 inferiore al numero stesso (sia essa 2^k = 256, k=8)
2. Si sottrae 2^k da N ottenendo 349 – 256 = 92
3. Se il risultato è zero abbiamo finito, altrimenti iteriamo il procedimento fino a che non otteniamo zero dalla sottrazione.
4. Inseriamo un bit a UNO nella notazione binaria di N in posizione k. (considerare che le posizioni partono da 0 a n-1, con n corrispondente al numero di bit totali della rappresentazione binaria usata e consentita).

349 – 256 = 93 – 64 = 29 – 16 = 13 – 8 = 5 - 4 = 1 - 1 = 0

	2^8		2^6		2^4	2^3	2^2		2^0
349 in binario =	1	0	1	0	1	1	1	0	1

Infatti, $V = \sum_{i=-m}^{n-1} d_i \cdot B^i$ = 1*2^8 + 0*2^7+ 1*2^6 + 0*2^5 + 1*2^4....

1*256 + 0 + 1*64 + 0 + 1*16 +1*8 +1*4 +1*1

Somma di valori in binario naturale

Assumiamo di avere solo n=3 bit a disposizione per rappresentare i valori e il risultato, quindi posso rappresentare B^n = 8 valori diversi interi positivi (da 0 a 7).

$5_{10}+1_{10}=6_{10}$

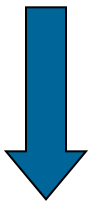
```
  1
1 0 1 +
0 0 1 =
-------
1 1 0
```

$2_{10}+3_{10}=5_{10}$

```
  1
0 1 0 +
0 1 1 =
-------
1 0 1
```

$5_{10}+3_{10}=8_{10}$

```
1 1 1
  1 0 1 +
  0 1 1 =
---------
1 0 0 0
```



Overflow!!

Per rappresentare il risultato della somma di 5_{10} + 3_{10} sono necessari 4 bit !

Altre basi: ottale e esadecimale

Quando per la rappresentazione di un numero si utilizzano molte cifre binarie può convenire usare altri sistemi di numerazione.

I sistemi ottale ed esadecimale sono utilizzati principalmente per rappresentare in modo più compatto i numeri binari.

L'algoritmo della divisione per passare da base 10 a ottale o esadecimale vale anche in questo caso. Provare.

I simboli del sistema **Ottale** sono 8 (da 0 a 7):
{ 0, 1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15, 16, 17, 20, ...}

I simboli del sistema **Esadecimale** sono 16 (da 0 a F):
{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10, 11, 12, 13, 14, ...}

Cambiamenti di base (metodo veloce)

Esiste un metodo veloce quando base di partenza BP e di arrivo BA sono esprimibili come BA=BP^k.

BP:Binario -> BA:Ottale , K=3, (8=2^3)

Per passare dalla codifica Binaria a quella Ottale, si raggruppano le cifre binarie a gruppi di 3 (**a partire da destra, allineandosi alla virgola**) e le si sostituiscono con una cifra del sistema ottale.

Esempio : $111001010_2 = 712_8$

Ottale -> Binario

Per passare dalla codifica Ottale a quella Binaria, si sostituisce ad ogni cifra ottale la corrispondente codifica binaria (composta da 3 cifre).

Esempio : $302_8 = 011000010_2$

Cambiamenti di base (metodo veloce)

BP: Binario -> BA:esadecimale , K=4, (16=2^4)

Per passare dal codice Binario a quello Esadecimale, si raggruppano le cifre a gruppi di 4 (a partire da destra) e le si sostituiscono con una cifra del sistema esadecimale.

Esempio : $1001000111111_2 = 91F_{16}$

Esadecimale -> Binario

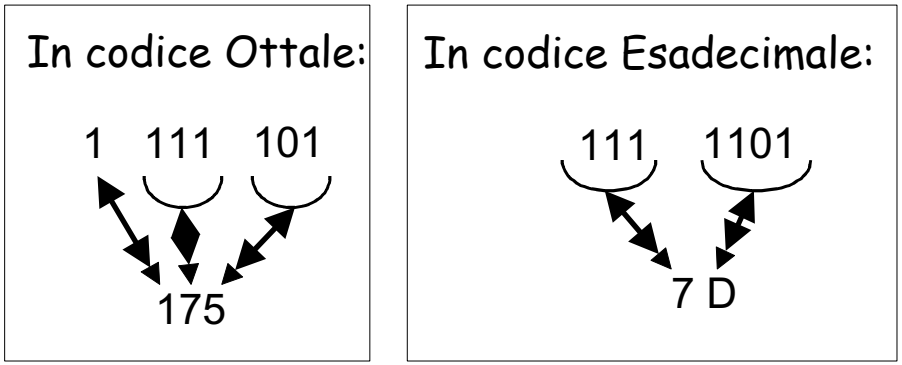
Per passare dal codice Esadecimale a quello Binario, si sostituisce ad ogni cifra esadecimale la corrispondente configurazione binaria (composta da 4 cifre).

Esempio : $A7F_{16} = 101001111111_2$

Esempi

Esempio 1

Codifica del numero $125_{10} = 1111101_2$



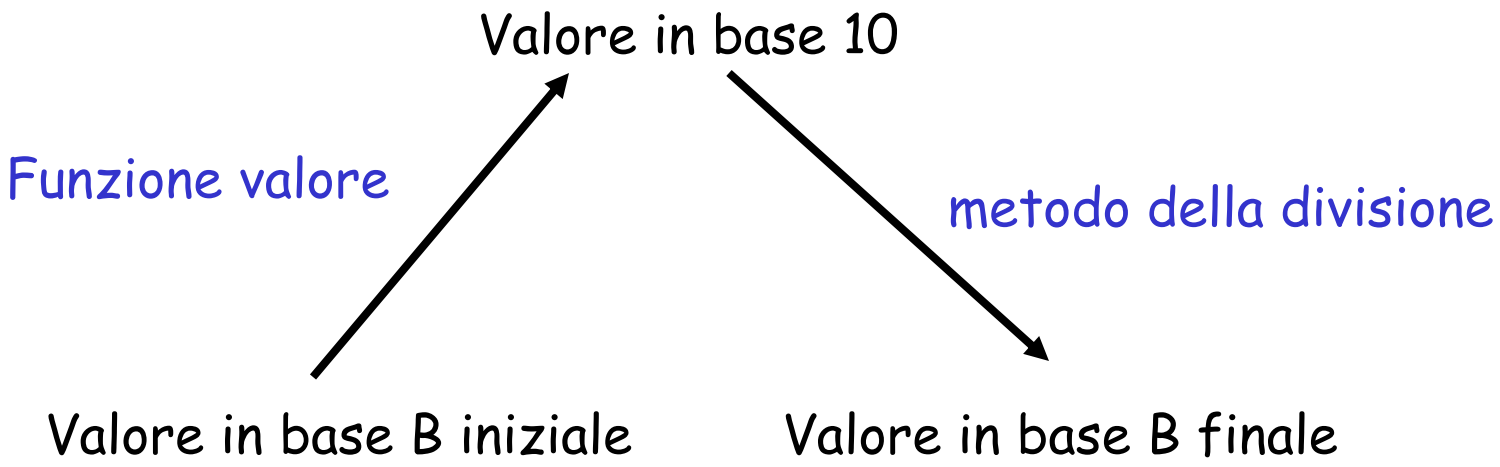
Esempio 2

Decimale	Binario	Ottale	Esadecimale
5	101	5	5
12	1100	14	C
78	1001110	116	4E
149	10010101	225	95

Esempi

Decimale	Binario (dcba)	Ottale	Esadecimale
0	0000	00	0
1	0001	01	1
2	0010	02	2
3	0011	03	3
4	0100	04	4
5	0101	05	5
6	0110	06	6
7	0111	07	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

Riassunto: passaggi di base generici



Codifica dei caratteri ASCII (7 bit -> 128 caratteri)

Il codice ASCII è non ridondante, perchè i simboli che vengono codificati sono in numero pari alle configurazioni ottenibili con 7 cifre binarie.

	000	001	010	011	100	101	110	111
0000	NUL	DLE		0	@	P	°	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	‘	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

MSB

LSB