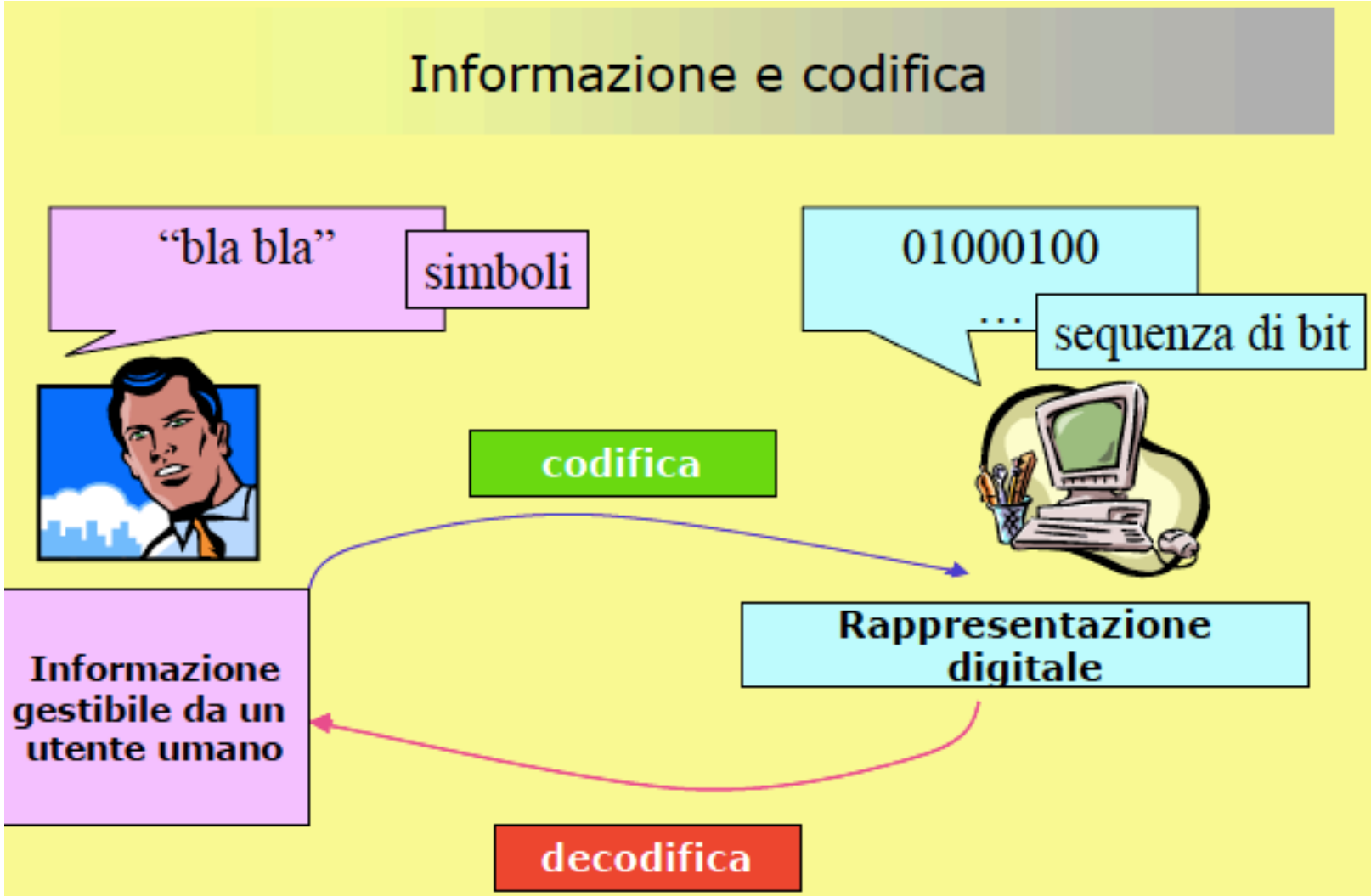
Informazione e codifica



Sommario

- Rappresentazione digitale o binaria
- Codifiche binarie di caratteri e simboli (dati non numerici)
- Codifica binaria dei dati numerici

Rappresentazione digitale o binaria

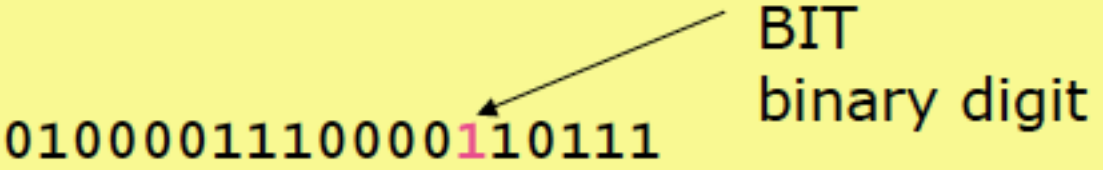
- All'interno dell'elaboratore ogni informazione è codificata usando la rappresentazione binaria o digitale, utilizzando cioè un alfabeto di due soli simboli: 0 e 1

Rappresentazione digitale: perché?

- **Scelta della rappresentazione: vincolata al tipo di operazione che devo fare su questa informazione**
- **Le ragioni della scelta di una rappresentazione binaria sono prevalentemente di carattere tecnologico, ossia dipende da come funzionano i dispositivi che costituiscono il computer**
- **questi dispositivi sono bistabili: assumono sempre uno stato fra due possibili**
- **Esempio:**
  - **due possibili stati di polarizzazione di una sostanza magnetizzabile**
  - **Passaggio/non passaggio di corrente attraverso un conduttore**
  - **Passaggio/non passaggio di luce attraverso una fibra ottica**
- **questi 2 possibili stati fisici possono essere naturalmente *tradotti* nei simboli numerici del sistema binario 1 e 0**

Bit

- L' entità minima di informazione codificabile attraverso la rappresentazione binaria è il **bit**
- Bit: da Binary Digit o cifra binaria
  - può assumere valori: 0 o 1



- Con una cifra binaria posso rappresentare soltanto due informazioni, ognuna delle quali viene fatta corrispondere convenzionalmente al simbolo 1 o 0

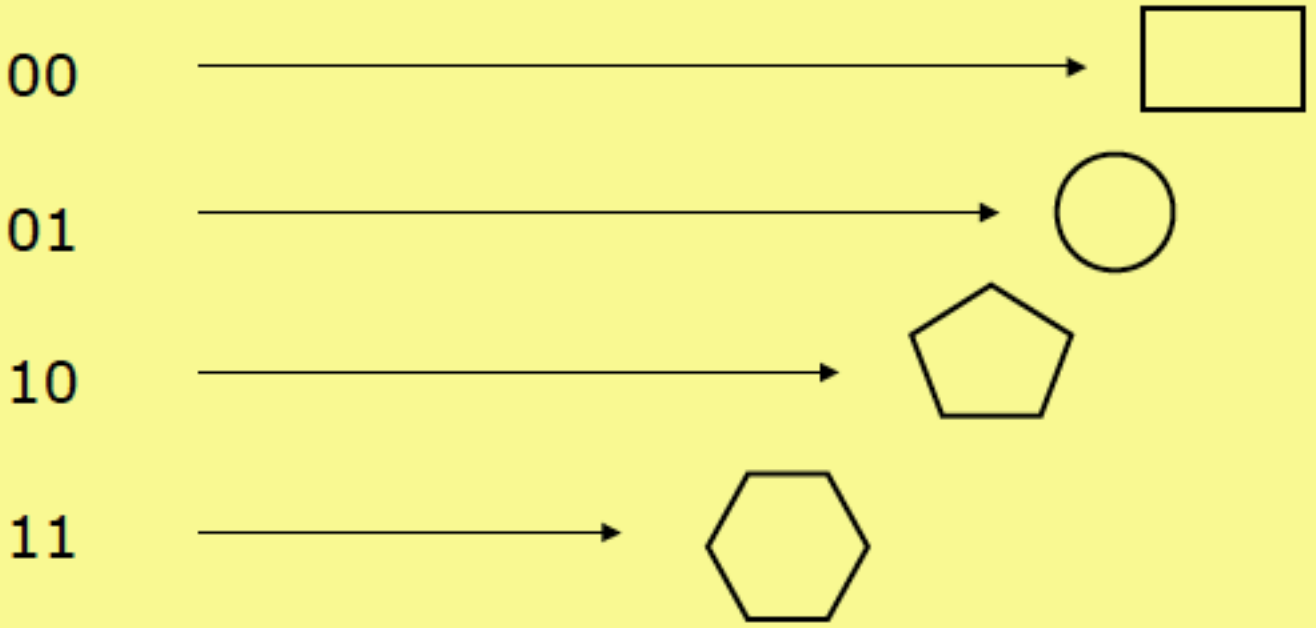
Codifica binaria dell'informazione

- Per poter rappresentare un numero maggiore di informazioni si usano sequenze di bit
- Ad esempio utilizzando 2 bit si possono codificare quattro informazioni diverse:

  00 01 10 11
- Il processo che fa corrispondere a una informazione una configurazione di bit prende il nome di codifica binaria dell'informazione
- L'associazione informazione/configurazione di bit è convenzionale: l'importante è che tutti quelli che devono condividere l'informazione usino la stessa convenzione

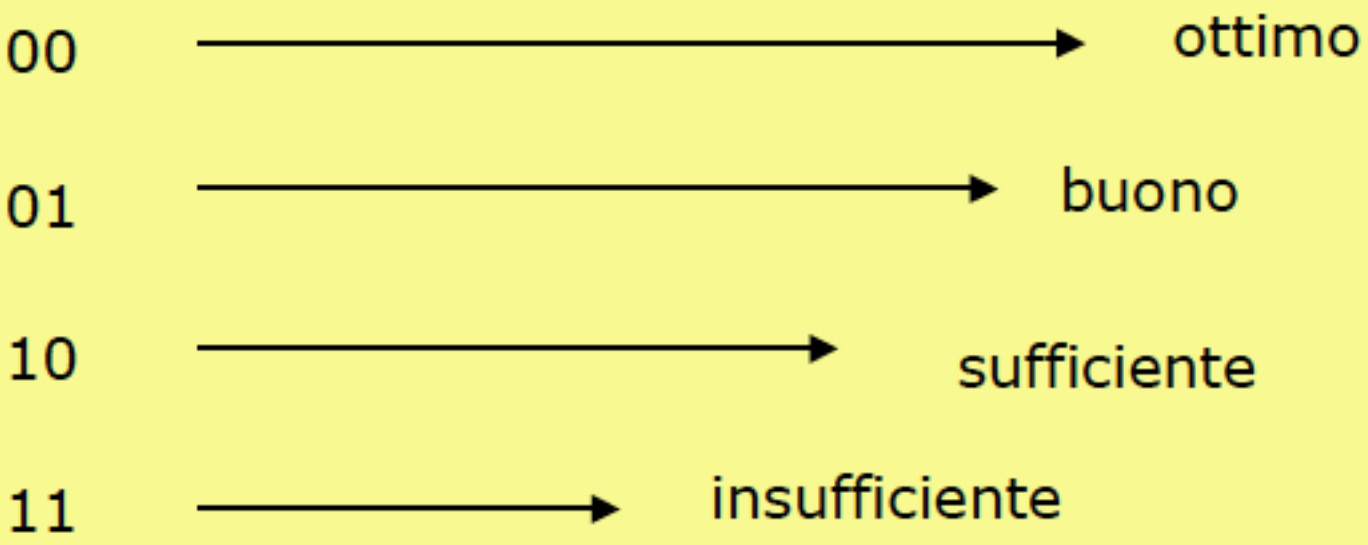
Esempio: simboli e codifiche

- Tramite 2 bit posso codificare 4 diverse figure geometriche:



Esempio: simboli e codifiche

- Tramite 2 bit posso codificare 4 diversi risultati di un esame



Sequenze di bit e codifiche

- Con 1 bit codifico 2 informazioni: 2^1
- Con 2 bit codifico 4 informazioni: 2^2
- Con 3 bit codifico 8 informazioni: 2^3

...

- Con N bit codifico 2^N informazioni

Sequenze di bit e codifiche

- Problema inverso: quanti bit ci vogliono per rappresentare M informazioni diverse (es. un alfabeto con M simboli)? Idea: seleziono il numero di bit N in modo che il numero delle possibili configurazioni di 0 e 1 sia max o uguale a M:

$$2^N >= M$$

-> c'è almeno una configurazione da far corrispondere a ogni simbolo dell'alfabeto

- Esempio:

Se devo codificare 220 informazioni diverse

dobbiamo avere almeno 8 bit:

$$2^8 = 256 \ (> 220)$$

7 bit non sarebbero sufficienti!

$$2^7 = 128 \ (< 220)$$

Sequenze di bit e codifiche

Numero di bit nella sequenza	Informazioni rappresentabili
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256

Byte

- Un gruppo di 8 bit viene denominato **byte**
- Corrisponde alla memorizzazione di un carattere
- Consente di codificare **2^8= 256 informazioni diverse**

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1
0	0	0	0	0	0	1	0
…							
1	1	1	1	1	1	1	1

8 biT= 1 byTe

Byte

- **Byte**: unità di misura della capacità di memorizzare informazione
- Si utilizzano i multipli dei byte
  - Kilo **KB** 2^{10} ~ un migliaio (**1024**)
  - Mega **MB** 2^{20} ~ un milione (**1KB*1024**)
  - Giga **GB** 2^{30} ~ un miliardo (**1MB*1024**)
  - Tera **TB** 2^{40} ~ mille miliardi (**1GB*1024**)
- Quanta memoria occupa un file: si misura in byte
- La capacità di memorizzazione di un dispositivo hardware si misura in byte (capacità di RAM, hard-disk)

Tipi di informazione e codifica binaria

- Vediamo nello specifico come vengono codificate mediante l'alfabeto binario tipi di informazione elementari che solitamente ci interessa elaborare:
  - caratteri utilizzati nella comunicazione scritta,
  - numeri,
  - immagini,
  - suoni,
  - video

Sommario

- Rappresentazione digitale o binaria
- Codifiche di caratteri e simboli
- Rappresentazione dei numeri

I caratteri usati nella comunicazione scritta

- Il primo problema che si presenta se vogliamo poter comunicare con il computer usando il nostro linguaggio è quello di poter rappresentare un alfabeto di caratteri
  - 52 lettere dell'alfabeto anglosassone (maiuscole + minuscole)
  - Segni di punteggiatura : , . ; ! ? " ' ...
  - Segni matematici + - { [> ...
  - Caratteri nazionali à è è ñ ç ...
  - Altri segni grafici: @ € © ...
- In totale si tratta di un alfabeto di 220 caratteri circa
- considerando che ho bisogno di associare a ogni carattere una configurazione di bit (codice numerico), ho bisogno di...

...almeno 8 bit (1 byte)

Codifica di dati alfabetici o caratteri

- codifiche standard: per rimediare all'apparente incomunicabilità fra linguaggio binario comprensibile dai computer e i dati alfabetici con cui di solito ci esprimiamo sono state definite alcune convenzioni (o **codici**) mediante le quali è possibile rappresentare in **modo univoco** un certo numero di caratteri attraverso configurazioni di bit prestabilite.
  - **ASCII** (American Standard Code for Information Interchange) standard: 7 bit per carattere, rappresenta 128 caratteri
  - **ASCII esteso**: 8 bit per carattere, rappresenta 256 caratteri
  - **UNICODE**: 16 bit per carattere: (ASCII + caratteri etnici), rappresenta circa 65.000 caratteri
- codifiche proprietarie:
  - MSWindows: 16 bit per carattere, simile a UNICODE

Unicode

- **UNICODE**: 16 bit per carattere -> in grado di codificare 2^{16} caratteri diversi (~ 65.000)
- ASCII + caratteri/simboli di altri alfabeti: greco, cirillico, cinese, arabo…
- http://www.unicode.org/charts/

Codice in forma esadecimale
1171_{16}

Binario:
0001 0001 0111 0001

Il codice ASCII

	Codice ASCII	Simbolo
caratteri di controllo →	0000 0000	NUL
	...	
segni di punteggiatura →	0010 0001	!
	...	
cifre decimali →	00110010	2
	00110000	0
	...	
I. Alfabetiche maiuscole →	0101 000	P
	...	
I. Alfabetiche minuscole →	0110 1001	i
	0110 1100	l
	0110 1111	o

Il codice ASCII

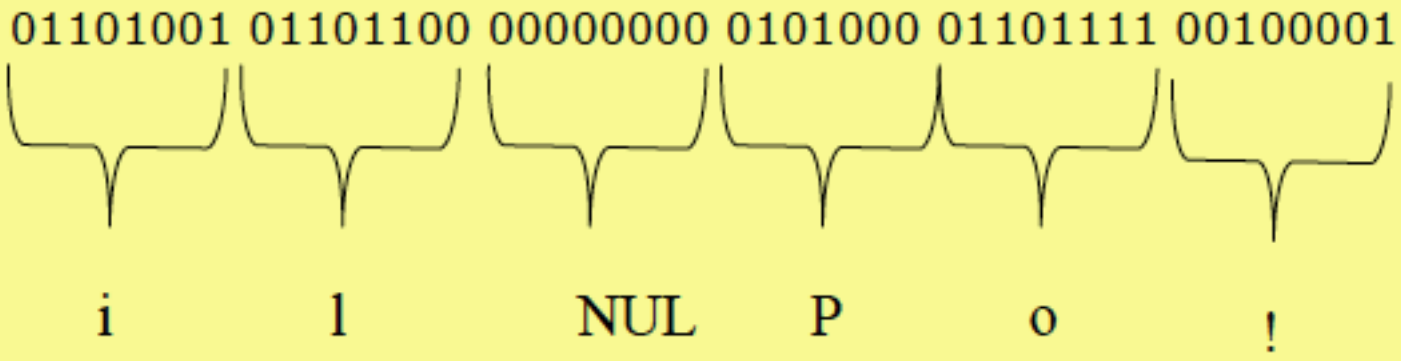
- **ASCII standard**: 7 bit per carattere -> in grado di codificare 2^{7}= 128 caratteri diversi
- Tabella: mancano le prime 32 configurazioni usate per la codifica dei caratteri speciali
- Legenda: nella prima colonna ho i primi 3 bit della codifica, nella prima riga gli altri quattro

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
010	sp	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
011	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
101	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
110	`	a	b	c	d	e	f	g	h	I	j	k	l	m	n	o
111	p	q	r	s	t	u	v	w	x	Y	z	{	|	}	~	canc

P -> 1010000

Sequenze di caratteri ASCII

- codifica di un singolo carattere -> codifica di una parola
- parola = sequenza di caratteri; codifica di una parola = sequenza delle codifiche ASCII dei caratteri che la costituiscono
- Codifica di un testo -> per la memorizzazione serviranno tanti byte quanti i caratteri che lo costituiscono (spazi bianchi e a capo compresi)
- Es. "il Po!"

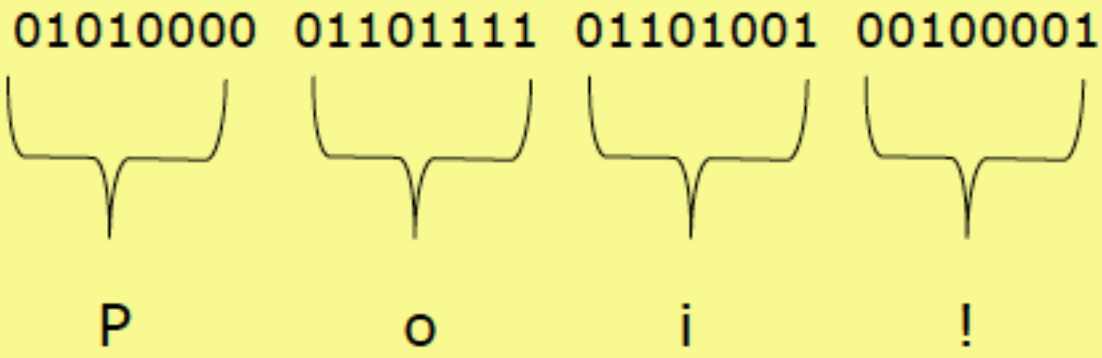


Sequenze di caratteri ASCII

- Il problema inverso: data una sequenza di bit, voglio ottenere il testo che codifica -> decodifica
- Dividendo la sequenza in gruppi di 8 bit è possibile risalire ai singoli caratteri che compongono la frase

Es.

- 01010000011011110110100100100001



Sommario

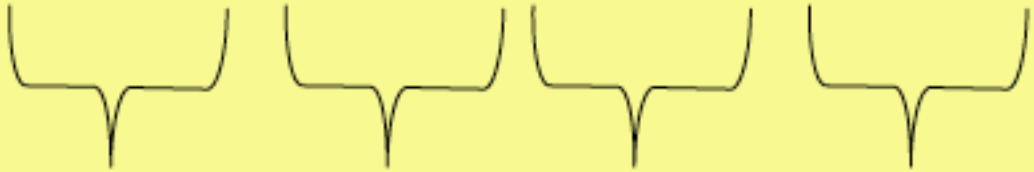
- Rappresentazione digitale o binaria
- Codifiche di caratteri e simboli
- Rappresentazione dei numeri

Numeri e codice ASCII

- Il codice ASCII permette di codificare cifre decimali da 0 a 9
- Usando il codice ASCII è possibile rappresentare numeri come sequenze di caratteri:

2002

00110100 00110000 00110000 00110100



2 0 0 2

- Problema: con questo tipo di rappresentazione non posso effettuare operazioni aritmetiche -> il modo in cui codificheremo i numeri in binario si rifà direttamente alle leggi di conversioni di numeri con basi differenti

Il sistema di numerazione posizionale decimale

- Partiamo dal modo in cui vengono rappresentati i numeri nel sistema di numerazione decimale a cui siamo abituati
- Notazione posizionale: ogni cifra del numero assume un certo valore in funzione della sua posizione

  365 notazione compatta, sta per

  $3 \times 100 + 6 \times 10 + 5 \times 1$ o meglio

  $3 \times 10^2 + 6 \times 10^1 + 5 \times 10^0$ in notazione esplicita

Notazione posizionale

- Ogni numero si esprime come la somma dei prodotti di ciascuna cifra per la base (nel nostro caso 10) elevata all'esponente che rappresenta la posizione della cifra (partendo da 0 a dx).

$$365 = (3 \times 10^2) + (6 \times 10^1) + (5 \times 10^0)$$

Notazione posizionale

- La notazione posizionale può essere usata con qualunque base creando così differenti sistemi di numerazione
- Per ogni base di numerazione utilizziamo un numero di cifre uguale alla base
- In informatica si utilizzano prevalentemente i **sistemi di numerazione binaria** (base 2), ottale (base 8), ed esadecimale (base 16)

Sistema di numerazione decimale

- La numerazione decimale (base 10) utilizza una notazione posizionale basata su 10 cifre (da 0 a 9) e sulle potenze di 10
  - Il numero 365 può essere rappresentato esplicitamente come

$$3 \times 10^2 + 6 \times 10^1 + 5 \times 10^0$$

MaT MeMo: data una base B,
$B^1 = B$, $B^0 = 1$

Sistema di numerazione binario

- Il sistema di numerazione **binario** (base 2) utilizza una notazione posizionale basata su 2 cifre (0 e 1) e sulle potenze di 2
  - Il numero 1001 può essere rappresentato esplicitamente come

$$1001_2 = 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 9$$

Sistema di numerazione ottale

- Il sistema di numerazione **ottale** (base 8) utilizza una notazione posizionale basata su 8 cifre (da 0 a 7) e sulle potenze di 8
  - Il numero 1001 può essere rappresentato esplicitamente come

    $1001_8 = 1 \times 8^3 + 0 \times 8^2 + 0 \times 8^1 + 1 \times 8^0 = 513$
- Esercizio: 534_8 in notazione esplicita
- Per evitare ambiguità si può esplicitamente indicare la base a cui si sta facendo riferimento:

- Esempio: $1001_2 \neq 1001_8 \neq 1001_{10}$

Sistema di numerazione esadecimale

- Il sistema di numerazione **esadecimale** (base 16) utilizza una notazione posizionale basata su 16 cifre (da 0 a 9 poi A,B,C,D,E,F) e sulle potenze di 16
  - Il numero B7FC può essere rappresentato esplicitamente come

    $(11) \times 16^3 + 7 \times 16^2 + (15) \times 16^1 + (12) \times 16^0 = 47100$

A	B	C	D	E	F
↓	↓	↓	↓	↓	↓
10	11	12	13	14	15

- Nelle pagine Web i colori vengono codificati utilizzando il sistema esadecimale.

Conversione da base n a base 10

- Per convertire un numero da una qualunque base alla base 10 è sufficiente rappresentarlo esplicitamente e fare i calcoli

  - $1001_2 = 1 \times 8^3 + 0 \times 8^2 + 0 \times 8^1 + 1 \times 8^0 = 513_{10}$
  - $B7FC_{16} = (11) \times 16^3 + 7 \times 16^2 + (15) \times 16^1 + (12) \times 16^0 = 47100_{10}$

Conversione da base 10 a base n

- Per convertire un numero decimale a una base n qualsiasi, occorre trovare tutti i resti delle successive divisioni del numero per la base n
- Esempio: voglio trovare il valore binario di 210; divido 210 per la base 2

Conversione da base 10 a base 2

Quoziente		Resto
210	2	
105	2	0
52	2	1
26	2	0
13	2	0
6	2	1
3	2	0
1	2	1
0		1

Lettura:
11010010

Conversione da base 10 a base 2

- Leggendo la sequenza dei resti dal basso verso l'alto si ottiene la codifica binaria:

  11010010_2

- Per verificare se avete fatto errori potete riconvertire il risultato in base 10

Rappresentazione dei numeri

- Tutti i numeri vengono distinti in 3 categorie:
  - Interi senza segno (positivi)
  - Interi con segno (positivi e negativi)
  - Reali (numeri positivi e negativi con virgola)
- Ognuna di queste categorie viene rappresentata in modo diverso
- Indipendentemente dalla rappresentazione scelta, essa sarà di tipo finito e consentirà di rappresentare solo un **sottoinsieme finito di numeri!**

Codici a lunghezza fissa

- All'interno del computer, a causa di vincoli tecnologici, per rappresentare qualsiasi tipo di numero occorre usare un **numero prefissato di cifre binarie (bit):** (16, 32…)
- Riflettiamo: supponiamo di avere 4 cifre a disposizione, qual è il più grande numero rappresentabile?

…dipende…

- In base 10: 9999
- In base 2: 1111 (15_{10})
- In base 16: FFFF (655535_{10})
- In base 8: 7777 (4095_{10})

Massimo numero rappresentabile

- In generale abbiamo che con n cifre a disposizione e base b, il più grande numero (intero positivo) rappresentabile si può esprimere come:

$$b^n - 1$$

- Es. con 4 cifre:
  - In base 10: $9999 = 10^4 - 1$
  - In base 2: $1111\ (15_{10}) = 2^4 - 1$
  - In base 16: $FFFF\ (655535_{10}) = 16^4 - 1$
  - In base 8: $7777\ (4095_{10}) = 8^4 - 1$

Numeri interi positivi

- Quindi nel caso di **numeri interi positivi** vale la seguente regola:

> Nella base di numerazione b, disponendo di un numero di cifre n, si possono rappresentare numeri:
>
> da 0 a $b^n - 1$

- Nella rappresentazione binaria di interi positivi a 16 bit i possibili valori saranno compresi fra 0 e 65.535 = $2^{16} - 1$ (base 2)
- Nella rappresentazione di interi positivi a 32 bit i possibili valori saranno compresi fra 0 e 4.294.967.295 = $2^{32} - 1$(base 2)

Numeri interi positivi e overflow

- Numeri più grandi del massimo numero rappresentabile creano problemi cosiddetti di **overflow**
- Esempio: supponiamo che il computer supporti una rappresentazione dei numeri a 8 bit e si voglia eseguire la seguente operazione fra numeri binari:

```
11111111 +    (255)
00000001 =    (1)
------------
********* -> errore di overflow
```

- Il risultato sarebbe 10000 0000 (9 cifre!), l'elaborazione si arresta e viene segnalato un errore di overflow