Università degli Studi di Torino
Scuola di Amministrazione Aziendale

Corso di laurea in
Management dell'informazione e della comunicazione aziendale

Corso di
Informatica generale (parte teorica)

Diego Magro

ARGOMENTI DI QUESTO GRUPPO DI LUCIDI

- **Codifica dell'informazione**
  - informazione e codifica
  - codifica dei caratteri
  - codifica dei numeri
  - codifica delle immagini
  - codifica dei filmati (cenni)
  - codifica dei suoni

- **I file come base per l'archiviazione dell'informazione (cenni)**

CODIFICA DELL'INFORMAZIONE

INFORMAZIONE

Difficile definire il concetto di informazione

In questo ambito, intendiamo l'informazione come l'insieme dei dati (potenziali oggetti di elaborazione) e delle istruzioni che specificano quale elaborazione effettuare sui dati

Possono essere informazioni: i numeri di telefono degli abitanti di una città, il valore di π, lo spartito di una sinfonia, il filmato delle vacanze, il canto di alcune cicale, una ricetta di cucina, la descrizione del funzionamento di un processo aziendale, le istruzioni per trovare le soluzioni di un'equazione di secondo grado,…

Come detto, l'informatica ha come oggetto l'*informazione* e il calcolatore, che è uno strumento per la gestione (elaborazione, memorizzazione, supporto alla condivisione) automatica dell'informazione

TIPI DI INFORMAZIONE (1)

In questo contesto, possiamo individuare i seguenti 5 tipi di informazione:

Testuale:

“C’era un gran rumore negli universi. Generazioni di stelle nascevano e morivano sotto lo sguardo di telescopi assuefatti, fortune elettromagetiche venivano dissipate in un attimo, sorgevano imperi d’elio e svanivano civiltà molecolari, gang di gas sovreccitati seminavano il panico, le galassie fuggivano rombando dal loro luogo d’origine, i buchi neri tracannavano energia e da bolle frattali nascevano universi dissidenti, ognuno con legislazione fisica autonoma. ”

[Da *Stefano Benni, Elianto, Feltrinelli, 1996*]

TIPI DI INFORMAZIONE (2)

Numerica:

2 * 3,14 * 4 = 25,12

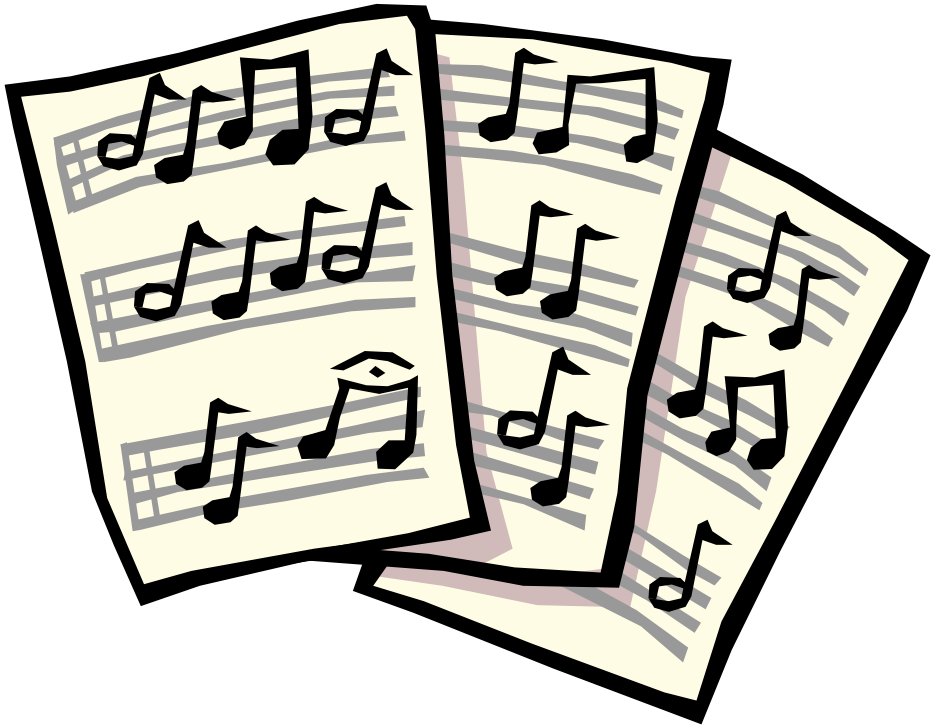
TIPI DI INFORMAZIONE (3)

Visiva (grafica, immagini, video):

[© ClipArt Microsoft Powerpoint]

TIPI DI INFORMAZIONE (4)

Audio:



[© ClipArt Microsoft Powerpoint]

TIPI DI INFORMAZIONE (5)

Istruzioni per il computer (tratteremo queste nel capitolo “Architettura harware degli elaboratori”) ed altre informazioni specifiche legate ai calcolatori (es. indirizzi di computer in una rete di calcolatori…anche di questo parleremo in altri capitoli)

“*copia nel registro R1 il contenuto della cella di memoria 22430*”

“invia il messaggio “XYZ” al calcolatore indentificato dall’indirizzo 128.234.116.39”

INFORMAZIONE E SUA RAPPRESENTAZIONE (O CODIFICA) (1)

- **Per poter essere elaborata, memorizzata o condivisa per mezzo di calcolatori, l'informazione deve in qualche modo essere rappresentata (o, equivalentemente, codificata) all'interno dei calcolatori**

- **E' bene tener presente che:**

l'informazione e la sua rappresentazione (o codifica) sono due cose distinte!

INFORMAZIONE E SUA RAPPRESENTAZIONE (O CODIFICA) (2)

“Ceci n’est pas une pipe” (Questo non è una pipa)

INFORMAZIONE E SUA RAPPRESENTAZIONE (O CODIFICA) (3)

…così come l'immagine di una pipa non è la pipa, allo stesso modo, la codifica di un'immagine non è l'immagine!

…questo vale per ogni altro tipo di informazione

la stessa informazione può essere rappresentata in modi diversi, es. il numero *sedici* può essere rappresentato

in notazione decimale come 16

in numerazione romana come XVI

in notazione binaria come 10000

in notazione esadecimale come 10

…ma esso resta pur sempre il numero *sedici*!

INFORMAZIONE E SUA RAPPRESENTAZIONE (O CODIFICA) (4)

- **In questa prima parte, vedremo alcuni modi in cui l'informazione può essere opportunamente codificata all'interno di un calcolatore**

- **Problema: come può un'informazione (di qualunque natura) essere rappresentata all'interno di un calcolatore affinché le opportune applicazioni software siano in grado di decodificarla, presentarla, elaborarla, condividerla (o supportarne la condivisione)?**
- **Es. come possiamo codificare un testo affinché un'elaboratore di testi possa visualizzarlo o consentire ad un utente di modificarlo e di condividerlo con altri utenti? Come possiamo rappresentare le dimensioni di un parallelepipedo affinché un'applicazione di calcolo possa determinarne il volume? Come possiamo codificare un filmato o una sinfonia affinché appositi software possano riprodurli? Come possiamo rappresentare un'istruzione affinché il computer la possa eseguire [ma di questo ci occuperemo in un altro capitolo]?…**

INFORMAZIONE E SUA RAPPRESENTAZIONE (O CODIFICA) (5)

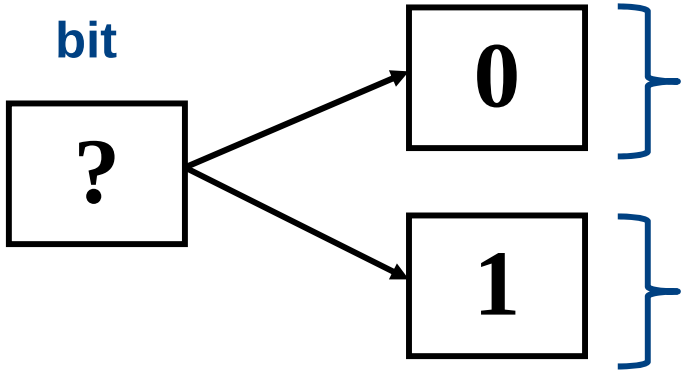
- **I “mattoncini” di base con cui TUTTA l’informazione è codificata all’interno di un calcolatore sono due simboli, cui convenzionalmente si attribuiscono i nomi ‘0’ (zero) e ‘1’ (uno)**

- **In un calcolatore, questi simboli sono realizzati da elementi fisici che in ogni istante rappresentano uno (ed uno solo) dei due simboli di base. Tali elementi fisici sono chiamati**

BIT

- **Bit = BI(nary) (digi)T o cifra binaria**
- **Da un punto di vista logico, un bit è una variabile che in ogni istante assume un valore (ed uno solo) tra due possibili**
- **Talvolta, con il termine ‘bit’ ci si riferisce anche ai valori che l’elemento fisico rappresenta (oltre che all’elemento fisico stesso)**
- **Attenzione! In questo contesto ‘0’ e ‘1’ sono solo due nomi convenzionalmente attribuiti ai valori che un bit può assumere: non significano necessariamente il numero 0 e il numero 1**

INFORMAZIONE E SUA RAPPRESENTAZIONE (O CODIFICA) (5)



bit che contiene (rappresenta) il simbolo ‘0’

bit che contiene (rappresenta) il simbolo ‘1’

- **In un calcolatore, TUTTA l’informazione (testi, numeri, immagini, filmati, suoni, istruzioni per il calcolatore e dati specifici relativi ai computer) è rappresentata da sequenze di valori di bit,**

  es. 1001000101111101 (rappresentazione digitale)

- **…con buona pace di poeti, matematici, pittori, registi, musicisti e… informatici!**

REALIZZAZIONE FISICA DEL BIT (1)

- **Come si può realizzare fisicamente un bit?**
- **E' sufficiente avere un sistema costituito da un 'osservatore' (O) e da un 'dispositivo osservato' (D), tale che, ogni volta che O 'osserva' D, quest'ultimo si trovi (dal punto di vista di O) in uno (ed uno solo) di due possibili stati distinti (e distinguibili da parte di O)**

- **Esempi:**
  - una persona (D) per la quale si considerino i due stati "seduto" e "non seduto" può essere una realizzazione fisica di un bit per un osservatore umano (O): infatti, ogni volta che osserviamo una persona, questa risulta essere o "seduta" oppure "non seduta" e non può risultare entrambe le cose né può risultare essere in una situazione diversa da queste due; naturalmente, occorre che la persona osservatrice O sia in grado distinguere se D è "seduta" o "non seduta"
  - una lampadina (D) per la quale si considerino gli stati "spenta" e "accesa" e una persona (O) che osserva la lampadina (e che sia in grado di discernere i due possibili stati)
  - …

REALIZZAZIONE FISICA DEL BIT (2)

- **Nei calcolatori, la realizzazione fisica dei bit non è attuata né da persone né da lampadine, ma si usano altri mezzi:**
  - presenza/assenza di carica elettrica
  - passaggio/non passaggio di corrente elettrica
  - tensione elettrica sopra/sotto una certa soglia
  - passaggio/non passaggio di luce
  - due stati di polarizzazione di particelle magnetiche
  - …
- **…ovviamente opportunamente accoppiati con i rispettivi meccanismi in grado di "osservare" e riconoscere lo stato del dispositivo**

SIGNIFICATO DEL BIT (1)

- **Che significato hanno i due valori che un bit può assumere? Cioè: se, in un certo istante, un bit ha valore ‘0’ (o, analogamente, ‘1’), cosa significa? Quale informazione rappresenta?**

- **Di per sé, il valore di un bit non ha alcun significato intrinseco!**

- **Affinché si possa attribuire significato al valore di un bit (o alla configurazione di valori di una sequenza di bit), occorre stabilire una qualche convenzione**

SIGNIFICATO DEL BIT (2)

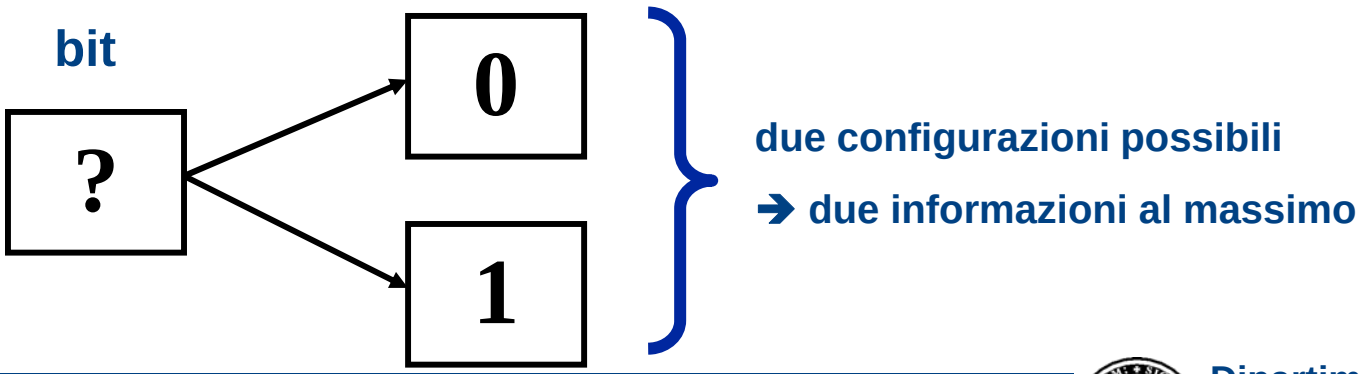
- **Es, ‘1’ non significa nulla, a priori, ma, se si stabilisce che con ‘1’ rappresentiamo il concetto di “bello” e con “0” quello di brutto, allora, nel contesto di questa convenzione, ‘1’ significa *bello***

- **Altri esempi:**
  - vero ➔ 1, falso ➔0
  - alto ➔ 1, basso ➔0
  - alto ➔0, basso ➔1
  - rosso ➔ 0, giallo ➔1
  - …

- **‘1110011001’ di per sé non significa nulla. Può avere significato solo all’interno di una convenzione (ovviamente, la stessa sequenza di bit può avere significati diversi nel contesto di due diverse convenzioni)**

NUMERO DI BIT E NUMERO MASSIMO DI INFORMAZIONI (1)

- **Quante informazioni è possibile rappresentare avendo a disposizione un solo bit?**
- **Siccome i possibili valori per un singolo bit sono due, esso ci consente di rappresentare soltanto due informazioni**
- **In altre parole, con un solo bit, è possibile avere due sole configurazioni e, nella stessa convenzione, ad ogni configurazione possiamo attribuire non più di un significato**

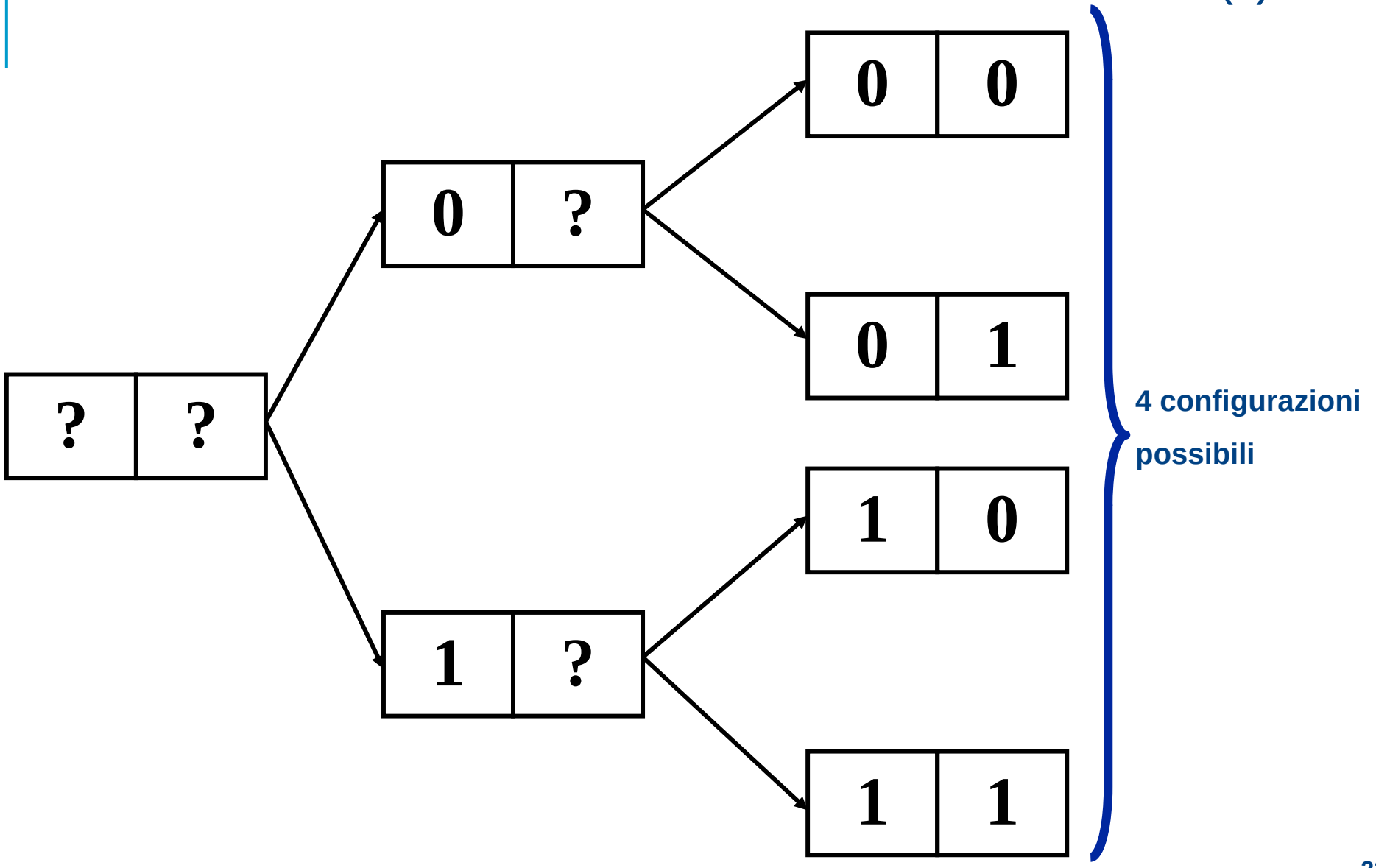


NUMERO DI BIT E NUMERO MASSIMO DI INFORMAZIONI (2)

- **Come fare, se si vogliono rappresentare, ad esempio, 4 informazioni (es. *bianco*, *nero*, *rosso*, *blu*)?**

- **Occorrono almeno 2, bit, infatti due bit consentono 4 configurazioni:**

NUMERO DI BIT E NUMERO MASSIMO DI INFORMAZIONI (3)



NUMERO DI BIT E NUMERO MASSIMO DI INFORMAZIONI (4)

- **Una possibile convenzione:**

  bianco **➔00,** ***nero*** **➔01,** ***rosso*** **➔10,** ***blu*** **➔11**

- **Quindi:**

  con 1 bit, 2 (= 2^1) configurazioni possibili, al massimo 2 informazioni rappresentabili

  con 2 bit, 4 (= 2^2) configurazioni possibili, al massimo 4 informazioni rappresentabili

  con 3 bit, 8 (= 2^3) configurazioni possibili, al massimo 8 informazioni rappresentabili

  …

  con *n* bit, 2^n configurazioni possibili, al massimo 2^n informazioni rappresentabili

NUMERO DI BIT E NUMERO MASSIMO DI INFORMAZIONI (5)

- **Per rappresentare k informazioni, sono necessari almeno *n* bit tale che *n* è il più piccolo numero naturale (0,1,2,3,…) tale che $2^n \geq k$**

- **es, per rappresentare 9 informazioni, 3 bit non sono sufficienti (infatti $2^3 = 8 < 9$) e ne occorrono almeno 4 (infatti $2^4 = 16 > 9$)…**

  …con 4 bit e 9 informazioni soltanto da rappresentare, 7 configurazioni delle 16 possibili non sarebbero necessarie…pazienza! Non ci si può far nulla…

IL BYTE

- **Byte: sequenza di 8 bit**

- **Il byte ha assunto particolare importanza in informatica, in quanto in molti computer la più piccola sequenza di bit che è manipolabile come un'unità è costituita da 8 bit**

- **…ciò significa, ad esempio, che se il valore un singolo bit di una sequenza deve essere cambiato (es. da '0' a '1'), all'interno del calcolatore questa operazione è fatta "coinvolgendo" (es. copiando e riscrivendo) una sequenza di 8 bit (byte, appunto) contenente il bit da modificare**

MULTIPLI DEL BYTE (e del bit)

- **Il byte è usato come unità di misura di varie grandezze: dimensione della memoria principale o secondaria di un calcolatore, velocità di trasferimento dati in una rete di calcolatori, dimensione del codice di un programma,…**

- **Come nel caso di altre unità di misura (es. il metro), hanno interesse i multipli del byte. Quelli usati sono:**
  - kilobyte (KB) = 2^{10} = 1024 byte
  - megabyte (MB) = 2^{20} = 1.048.576 (circa un milione di) byte
  - gigabyte (GB) = 2^{30} = 1.073.741.824 (circa un miliardo di) byte
  - terabyte (TB) = 2^{40} = 1.099.511.627.776 (circa mille miliardi di) byte
  - petabyte (PB) = 2^{50} = 1.125.899.906.842.624 (circa un milione di miliardi di) byte

- **A differenza di altre unità di misura (es. il metro), non hanno interesse i sottomultipli del byte (es.: non esite il *decibyte*)**
- **Analogamente, anche il bit e i suoi multipli sono usati come unità di misura:** kilobit (Kb, 1024 bit), megabit (Mb, 1.048.576 bit), ecc.

CODIFICA DEI CARATTERI (1)

- **Nei sistemi di scrittura umani, la forma scritta del linguaggio naturale si fonda sui *caratteri***
- **La codifica dell'informazione testuale implica, quindi, una codifica dei caratteri**
- **Non è banale definire il concetto di carattere: in prima approssimazione possiamo dire che un carattere è l'unità grafica minima (*grafema*) di un sistema di scrittura**
- **Ad esempio, le lettere dell'alfabeto anglosassone ("A", "a", "s", …), i segni di interpunzione (". " [punto], ","[virgola], …), ecc.**

CODIFICA DEI CARATTERI (2)

- **La definizione di carattere data sopra è approssimata (es.: “è”[la lettera “e” con sopra l’accento grave] è un carattere o è una combinazione di due caratteri? Un carattere cinese è un singolo carattere anche se è composto da più segni di base?), ma, per i nostri scopi, è sufficiente a suggerire che cosa sia un carattere**

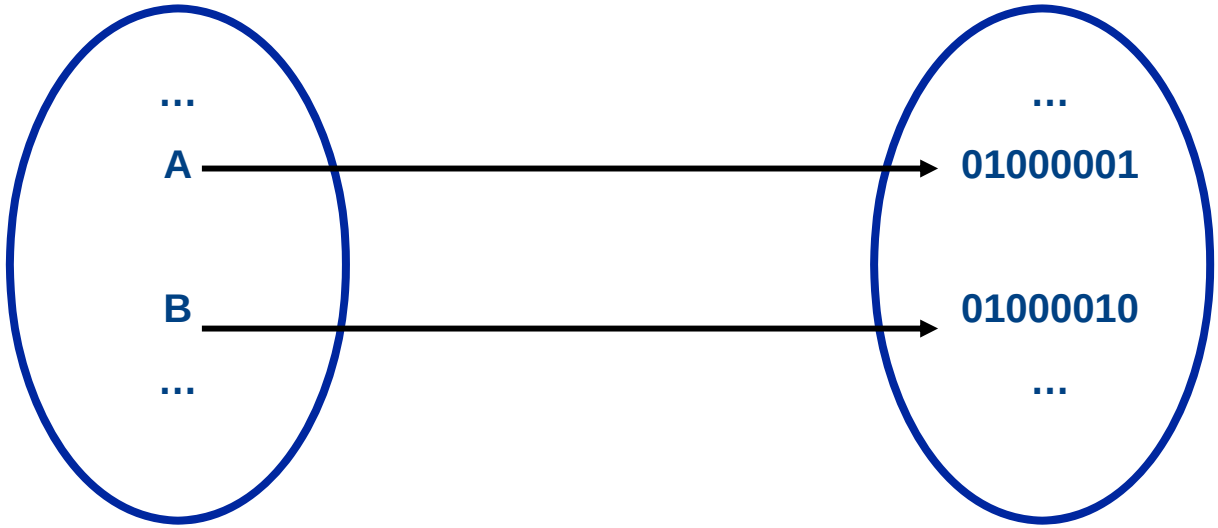
- **Inoltre, in informatica, sono considerati caratteri anche certe entità che hanno (o avevano) una funzione di controllo dei dispositivi di visualizzazione dei testi (es. stampanti): sono i cosiddetti caratteri speciali o caratteri di controllo**

- **Infine, occorre distinguere fra il carattere e la forma che esso assume nella visualizzazione (es il carattere “c minuscolo” può assumere varie forme nella visualizzazione: c, *c*, $\mathcal{C}$, **c**, $\mathfrak{c}$, …)**

CODIFICA DEI CARATTERI (3)

- **Problema (in prima approssimazione): stabilire una convenzione che associ ad ogni carattere una sequenza di valori di bit**

Es.:



CODIFICA DEI CARATTERI (4)

- **Ciò che sarebbe auspicabile: avere un'UNICA CONVENZIONE condivisa da tutti, cioè uno STANDARD**
  - questo consentirebbe ad un testo di essere codificato (e decodificato) nello stesso modo su ogni computer (e da ogni applicazione informatica) del mondo ➔ l'obiettivo del computer di essere supporto automatico per la condivisione di informazione sarebbe pienamente centrato almeno nel caso dell'informazione testuale

- **Ciò che si ha in realtà: numerose convenzioni, per molte delle quali si dice che sono degli standard!**

- **Perché questo fenomeno? Perché il processo di diffusione dei computer è stato più rapido del processo di standardizzazione**

CODIFICA DEI CARATTERI (5)

- **Altra cosa da tener presente per capire il fenomeno “codifica dei caratteri”: all’inizio, la diffusione dei computer ha interessato prevalentemente gli Stati Uniti e l’Europa occidentale**

- **Vi sono vari “standard” e modi per codificare i caratteri, noi accenneremo ai seguenti:**

  1. ASCII
  2. Estensioni ASCII
  3. Unicode + UTF-8

ASCII (1)

- **ASCII = American Standard Code for Information Interchange**
- **Proposto negli anni ’60, aggiornato fino agli anni ’80, è attualmente la convenzione di codifica più diffusa**
- **Caratteristiche principali:**
  1. E’ basato sui caratteri utilizzati nell’inglese
  2. Codifica 128 caratteri
  3. usa configurazioni di 7 bit

  } $2^7 = 128$

  4. codifica 33 caratteri di controllo e 95 caratteri stampabili (le 26 lettere maiuscole e le 26 minuscole dell’alfabeto anglosassone, le 10 cifre decimali, i segni di punteggiatura, i simboli delle operazioni aritmetiche, ed altri caratteri come “%”, “&”, “{”, ecc.)

ASCII (2)

Generalmente, la codifica ASCII richiede un byte (8 bit) per ogni carattere, anche se utilizza effettivamente solo 7 bit ➔nella tabella il primo bit di ogni byte che codifica un carattere è sempre '0'

ASCII	Simb.	ASCII	Simb.	ASCII	Simb.
00000000	NUL	00001110	SO	00011100	FS
00000001	SOH	00001111	SI	00011101	GS
00000010	STX	00010000	DLE	00011110	RS
00000011	ETX	00010001	DC1	00011111	US
00000100	EOT	00010010	DC2	00100000	SP
00000101	ENQ	00010011	DC3	00100001	!
00000110	ACK	00010011	DC4	00100010	"
00000111	BEL	00010101	NAK	00100011	#
00001000	BS	00010110	SYN	00100100	$
00001001	HT	00010111	ETB	00100101	%
00001010	LF	00011000	CAN	00100110	&
00001011	VT	00011001	EM	00100111	‘
00001100	FF	00011010	SUB	00101000	(
00001101	CR	00011011	ESC	00101001	)

ASCII (3)

ASCII	Simb.	ASCII	Simb.	ASCII	Simb.
00101010	*	00111001	9	01000111	G
00101011	+	00111010	:	01001000	H
00101100	,	00111011	;	01001001	I
00101101	-	00111100	<	01001010	J
00101110	.	00111101	=	01001011	K
00101111	/	00111110	>	01001100	L
00110000	0	00111111	?	01001101	M
00110001	1	01000000	@	01001110	N
00110010	2	01000001	A	01001111	O
00110011	3	01000010	B	01010000	P
00110100	4	01000011	C	01010001	Q
00110101	5	01000100	D	01010010	R
00110110	6	01000101	E	01010011	S
00110111	7	01000110	F	01010100	T
00111000	8				

ASCII (4)

ASCII	Simb.	ASCII	Simb.	ASCII	Simb.
01010101	U	01100011	c	01110001	q
01010110	V	01100100	d	01110010	r
01010111	W	01100101	e	01110011	s
01011000	X	01100110	f	01110100	t
01011001	Y	01100111	g	01110101	u
01011010	Z	01101000	h	01110110	v
01011011	[	01101001	i	01110111	w
01011100	\	01101010	j	01111000	x
01011101	]	01101011	k	01111001	y
01011110	^	01101100	l	01111010	z
01011111	-	01101101	m	01111011	{
01100000	`	01101110	n	01111100	|
01100001	a	01101111	o	01111101	}
01100010	b	01110000	p	01111110	~
				01111111	DEL

ESTENSIONI ASCII (1)

- **l'ASCII non codifica molti dei caratteri utilizzati nella forma scritta di molti linguaggi naturali correnti!**
- **Es.: non codifica le lettere accentate (es.: 'è'), non codifica i caratteri dell'alfabeto cirillico o quelle dell'alfabeto greco, non codifica i caratteri giapponesi, cinesi o coreani, …**
- **➔ sono nate altre codifiche di caratteri per rispondere a queste esigenze: quelle cui accenniamo qui sono le cosiddette estensioni ASCII**
- **(si noti il plurale: ne esistono più di una, ciascuna estende l'ASCII in direzioni diverse)**

ESTENSIONI ASCII (2)

- **Caratteristiche principali:**
  1. utilizzano 8 bit (l'intero byte) ➔ hanno a disposizione 2^8 = 256 configurazioni di bit
  2. sono compatibili con l'ASCII: tutte le configurazioni di bit in cui il primo bit è '0' hanno lo stesso significato (cioè rappresentano lo stesso carattere) dell'ASCII
  3. i "nuovi" caratteri sono codificati con le configurazioni in cui il primo bit del byte è '1'

- **Es.: ISO 8859-1 (detta anche ISO Latin 1)[1] codifica i caratteri della maggior parte degli alfabeti dell'Europa occidentale. Altre estensioni ASCII codificano (oltre ai caratteri dell'alfabeto anglosassone) i caratteri di alfabeti dell'Europa orientale, ecc.**

1 ISO Latin 9, introdotta nel 1999, modifica leggermente ISO Latin 1 (e introduce la codifica per il simbolo dell'euro €)

ESTENSIONI ASCII (3)

- **➔ usando, ad esempio, ISO Latin1, si possono rappresentare tutti i testi in inglese (ovvio: è un'estensione di ASCII), ma anche tutti i testi in italiano (in cui possono comparire lettere accentate)**
- **usando un'altra estensione ASCII (ISO 8859-5) si possono rappresentare tutti i testi in inglese (ovvio), ma anche tutti i testi in un alfabeto cirillico e anche tutti i testi in cui compaiono caratteri dell'alfabeto anglosassone e anche caratteri cirillici**
- **Problemi non risolti da questo approccio:**
  1. possibilità di rappresentare testi che mischiano caratteri codificati in una certa estensione (es Latin1) con caratteri non codificati in quella estensione, anche se codificati in altre estensioni (es. italiano e russo)
  2. condivisione di un testo qualunque fra applicazioni che usano estensioni diverse (es. Latin1 e ISO 8859-5)
  3. rappresentazione di testi in cinese, giapponese o coreano
  4. …

UNICODE (1)

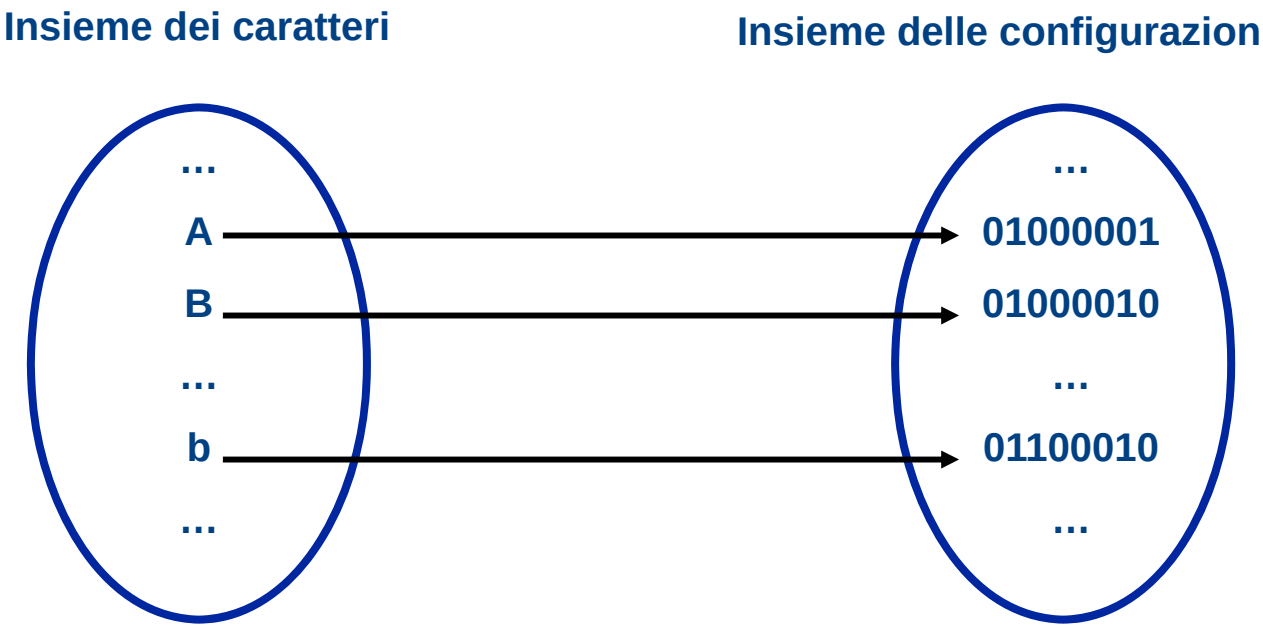
- **Per ovviare a questi problemi sono in atto sforzi di standardizzazione**
- **Uno dei più importanti è UNICODE**
- **Attualmente, UNICODE "copre" tutti i caratteri in uso in tutti gli alfabeti correnti: anglosassone, dell'Europa (occidentale e orientale), arabo, cinese, giapponese, coreano, …, alcuni caratteri di alfabeti di lingue morte (es. fenicio), simboli matematici e musicali, ecc.**
- **E' tuttora in evoluzione**
- **E' attualmente il più serio candidato a diventare uno standard internazionale (e a soppiantare anche l'ancora diffusissimo ASCII e le sue estensioni)**

UNICODE (2)

- **Per capire meglio le caratteristiche di Unicode è utile confrontarlo con ASCII e le sue estensioni ISO 8859**
- **ASCII (ogni sua estensione ISO 8859):**
  1. individuano un insieme di caratteri che si vuole rappresentare
  2. stabiliscono il numero di bit necessari a rappresentare ogni singolo carattere (7 per ASCII, 8 per le estensioni ISO 8859)
  3. stabiliscono un'associazione fra ogni carattere dell'insieme scelto e una ed una sola configurazione di bit della lunghezza stabilita (la quale non può essere associata a più di un carattere)

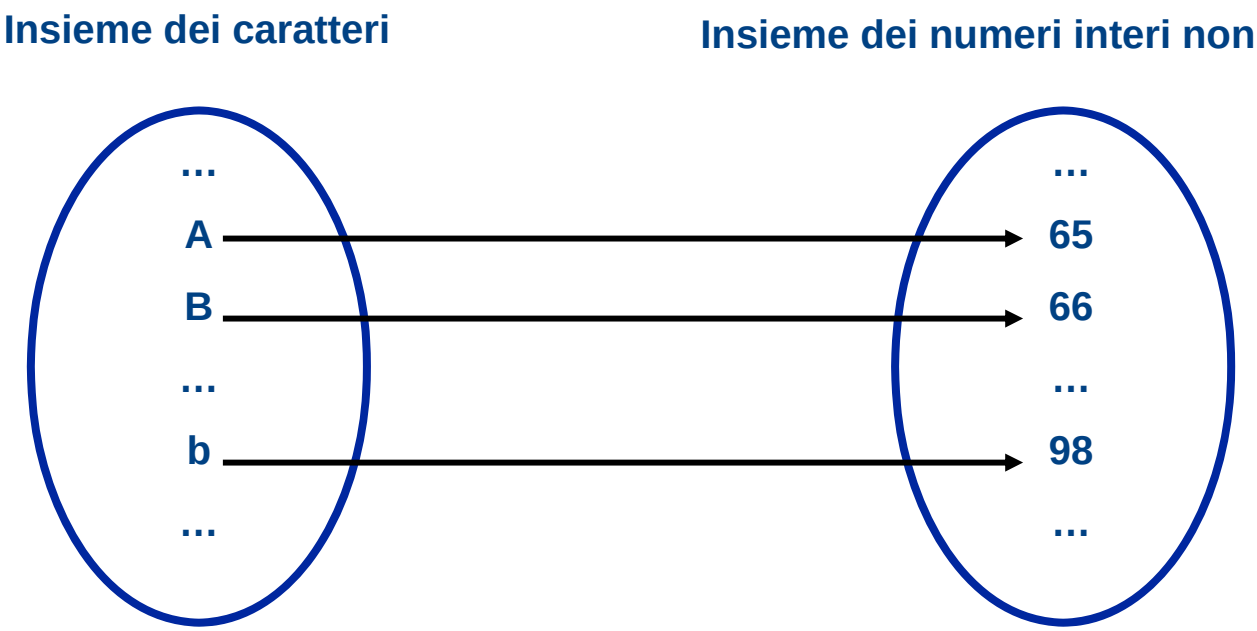
UNICODE (3)

ASCII



UNICODE (4)

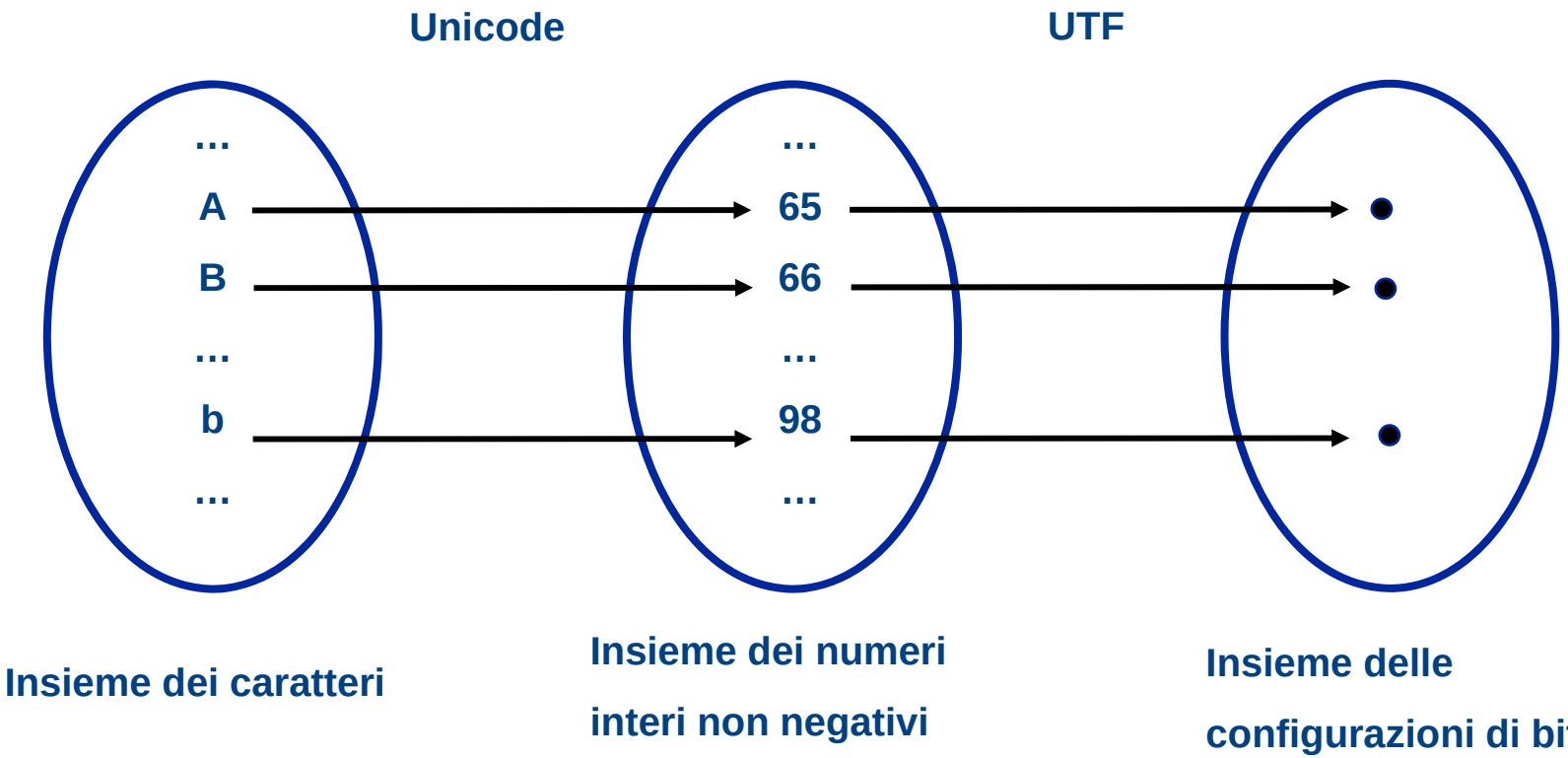
- **Unicode ha un approccio diverso (più flessibile), la cui "filosofia" può essere così schematizzata:**
  1. individua l'insieme di caratteri che si vuole rappresentare
  2. assegna ad ogni carattere un numero intero non negativo



UNICODE E UTF (1)

- **Unicode non specifica quali configurazioni di bit sono associate ai vari caratteri (lascia libertà), in questo modo è più flessibile e consente diverse codifiche, in termini di configurazioni di bit**
- **"…però un computer può codificare l'informazione solo attraverso configurazioni di bit ➔ qualche convenzione deve pur esistere che specifichi le regole per rappresentare i caratteri Unicode con configurazioni di bit!"**
- **Attualmente questo compito è principalmente assolto dalle regole specificate nelle varie versioni di UTF (Unicode Transformation Format), anche se altri meccanismi sono possibili: UFT sono dei formati di codifica che specificano come rappresentare ogni singolo numero intero, associato da Unicode ad un carattere, con una sequenza di bit**
- **Vi è più di un formato UTF, fra questi, quelli più diffusi vanno sotto i nomi di UTF-8, UTF-16, UTF-32**

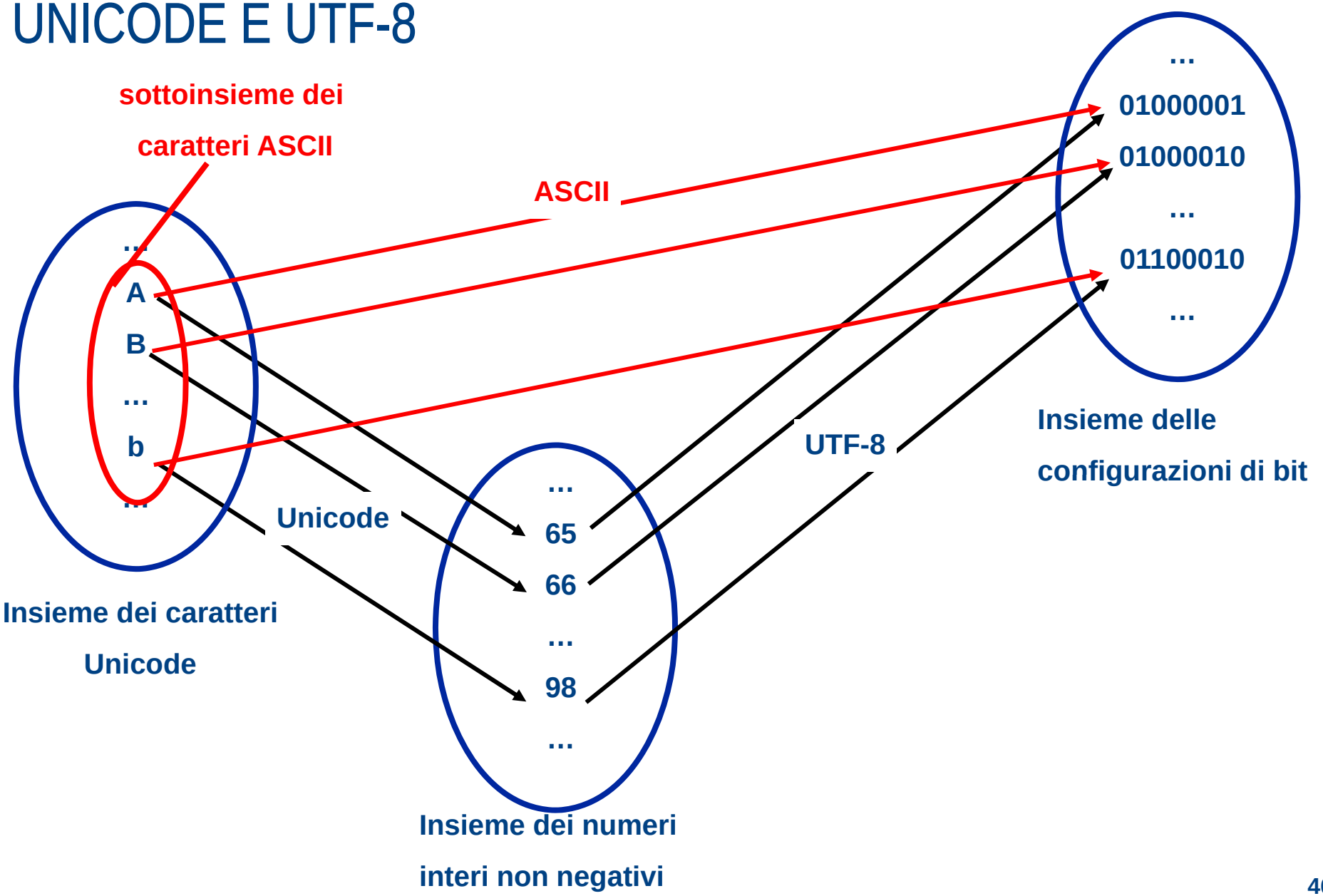
UNICODE E UTF (2)



UNICODE E UTF (3)

- **Non entriamo nei dettagli dei formati UTF e delle varie differenze fra UTF-8, UTF-16 e UTF-32**
- **Solo due parole su UFT-8:**
  - per il modo in cui Unicode assegna ai caratteri i corrispondenti numeri interi e per il modo in cui UTF-8 specifica la rappresentazione di tali numeri interi con sequenze di bit, Unicode+UFT-8 è compatibile con ASCII (cioè i 128 caratteri ASCII sono codificati nello stesso modo, con le stesse sequenze di bit, da ASCII e da Unicode+UFT-8)
  - il numero di bit necessari per codificare ogni singolo carattere in Unicode+UFT-8 non è fisso, ma può variare da carattere a carattere (si va da uno a quattro bytes per carattere)

UNICODE E UTF-8



CODIFICA DELLE PAROLE

Sono sequenze di caratteri

- **Esempio: “INFORMATICA” con ASCII**

01001001	01001110	01000110	01001111	01010010	01001101	01000001
I	N	F	O	R	M	A

01010100	01001001	01000011	01000001
T	I	C	A

CODIFICA TESTI STRUTTURATI

- **sono sequenze di parole separate da *caratteri speciali* (spazi bianchi, caratteri di fine riga, ecc)**
- **I *caratteri speciali* sono anch'essi codificati (es. in ASCII lo spazio è 00100000)**
- **quindi anche un testo strutturato è rappresentato con una sequenza di valori di bit**
- **(la sequenza che rappresenta il testo è costituita da una giustapposizione di sottosequenze, ciascuna delle quali rappresenta un carattere)**

DECODIFICA DI TESTI (1)

- **Problema: dalla sequenza di bit al testo rappresentato**
- **All'interno della sequenza di valori di bit che rappresenta il testo, si individuano le sottosequenze che rappresentano i caratteri e si determina il carattere corrispondente ad ogni sottosequenza**
- **Es. Se il testo è stato codificato con ASCII o qualche sua estensione ISO 8859, si individuano le sottosequenze di 8 bit e si determina il carattere corispondente ad ogni byte**
- **(Ovviamente per poter correttamente decodificare un testo occorre sapere con quale codifica esso è rappresentato)**

DECODIFICA DI TESTI (2)

Es., data la seguente sequenza di bit:

01000110011010010110110001100001011100110111010001110010011011110110001101100011011000010000101000001101011000110110111101110010011101000110000100100000011001010010000001100111011000010110100101100000100101100

se sappiamo che essa è la rappresentazione di un testo secondo la codifica ASCII, per decodificarla, possiamo operare nel seguente modo (analogo a quello che utilizzerebbe un programma per la visualizzazione di testi):

DECODIFICA DI TESTI (3)

1. **suddividiamo la sequenza di valori di bit in sottosequenze di 8 valori ciascuna:**

01000110 01101001 01101100 01100001 01110011 01110100
01110010 01101111 01100011 01100011 01100001 00001010
00001101 01100011 01101111 01110010 01110100 01100001
00100000 01100101 00100000 01100111 01100001 01101001
01100001 00101100

DECODIFICA DI TESTI (4)

2. **determiniamo il carattere corrispondente ad ogni sottosequenza:**

01000110 (F) 01101001 (i) 01101100 (l) 01100001 (a) 01110011 (s)
01110100 (t) 01110010 (r) 01101111 (o) 01100011 (c) 01100011 (c)
01100001 (a)
00001010 (Line feed)
00001101 (Carriage return)

} **assieme, rappresentano il comando di scrivere su una nuova riga**

01100011 (c) 01101111 (o) 01110010 (r) 01110100 (t) 01100001 (a)
00100000 (blank) 01100101 (e) 00100000 (blank) 01100111 (g)
01100001 (a) 01101001 (i) 01100001 (a) 00101100 (,)

DECODIFICA DI TESTI (5)

Il testo rappresentato dalla precedente sequenza di valori di bit è così visualizzato:

Filastrocca

corta e gaia,

…e scopriamo, quindi, che è l'inizio di una filastrocca di Gianni Rodari

CODIFICA DEI NUMERI (1)

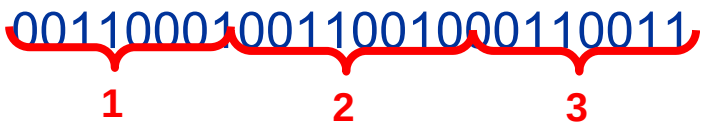
Vedremo codifica di:

- **numeri interi senza segno**
- **numeri interi con segno**
- **numeri con la virgola**

CODIFICA DEI NUMERI (2)

Le codifiche dei caratteri consentono di codificare le cifre decimali da“0” a “9” ➔ consentono una rappresentazione dei numeri

Es.: usando il codice ASCII, possiamo rappresentare il numero 123 in questo modo:



N.B.: questa è la rappresentazione simbolica delle cifre decimali che compongono un numero, ma NON rappresenta il valore del numero

Questa rappresentazione è adatta nell’interazione con l’utente ➔ è usata nell’acquisizione di dati (input) e nella presentazione di dati (output) da e ad un utente (solitamente) umano

Non può essere utilizzata da un calcolatore per svolgere i calcoli: a questo scopo serve rappresentare il VALORE del numero

NOTAZIONE DECIMALE (1)

- **Per capire come sono rappresentati i valori numerici all'interno di un calcolatore, occorre capire (solo un po'!) la notazione binaria e, per fare questo, è opportuno riprendere la comune notazione decimale**

- **NOTAZIONE DECIMALE:**
  1. E' quella a cui siamo abituati
  2. utilizza dieci simboli (0, 1, 2, …, 9) che denotano i valori *zero*, *uno*, *due*,…,*nove*
  3. è in base dieci
  4. è di tipo posizionale (cioè ogni cifra di un numerale[1] rappresenta un valore che dipende dalla posizione di tale cifra all'interno della sequenza di cifre che costituiscono il numerale)

1 Il numerale è la sequenza di simboli usata, in un sistema di numerazione, per denotare il numero: ad esempio, 'XIV' e '14' sono due diversi numerali (il primo nel sistema Romano, il secondo in quello decimale) che denotano entrambi il (medesimo) numero *quattordici*

NOTAZIONE DECIMALE (2)

- **Es.: il valore del numero centoventitré, in notazione decimale, si scrive**

  123

il valore è così determinato: tre unità + due decine + un centinaio, cioè:

123 = 3*1 + 2*10 + 1*100 =

= 3* 10^0 + 2*10^1 + 1*10^2

➔**ogni cifra rappresenta il valore ottenuto moltiplicando il valore della cifra per quello di un'opportuna potenza di dieci (la base!) e tale potenza dipende dalla posizione che la cifra occupa all'interno del numero.**

➔**il valore del numero si ottiene sommando i valori rappresentati da ogni singola cifra**

NOTAZIONE BINARIA (1)

- **La notazione decimale non è l'unico modo di rappresentare i valori numerici**
- **L'uso della base dieci può apparirci ovvio, ma un qualunque altro numero intero positivo potrebbe essere scelto come base:**

in informatica, risulta particolarmente comoda la base due, cioè la notazione binaria (questo non sorprende, visto che un bit può rappresentare solo sue simboli)

ATTENZIONE: ciò non significa che sempre e in ogni caso il valore di un numero sia rappresentato all'interno di un calcolatore con la sua notazione binaria! Significa soltanto che essa costituisce spesso il fondamento della rappresentazione

NOTAZIONE BINARIA (2)

- **La notazione binaria è analoga a quella decimale, soltanto che usa due simboli e la base due**
- **Usando solo due simboli è adatta all'uso all'interno di un calcolatore (dove si hanno a disposizione solo i bit)**

NOTAZIONE BINARIA:

1. utilizza due simboli ('0', '1') che denotano i valori *zero* e *uno*
2. è in base due
3. è di tipo posizionale (cioè ogni cifra di un numero rappresenta un valore che dipende dalla sua posizione)

Es.: $1101_{bin} = 1*2^0 + 0*2^1 + 1*2^2 + 1*2^3 =$

$= 1*1 + 0*2 + 1*4 + 1*8 =$ **tredici** (13_{dec})

$100110_{bin} = 0*2^0 + 1*2^1 + 1*2^2 + 0*2^3 + 0*2^4 + 1*2^5 =$

$=0*1 + 1*2 + 1*4 + 0*8 + 0*16 + 1*32 =$ **trentotto** (38_{dec})

NOTAZIONE BINARIA (3)

CONVERSIONE DI BASE

Da base 2 a base 10:

se c_i è o 0 o 1: $c_n c_{n-1} \ldots c_1 c_{0\ bin} = (c_0 * 2^0 + c_1 * 2^1 + \ldots + c_{n-1} * 2^{n-1} + c_n * 2^n)_{dec}$

Es.: $1110_{bin} = (0*2^0 + 1*2^1 + 1*2^2 + 1*2^3)_{dec} = 14_{dec}$

Da base 10 a base 2:

leggere sul libro, se curiosi

NOTAZIONE BINARIA (4)

ADDIZIONE CON NUMERI IN NOTAZIONE BINARIA

Procedimento analogo a quello usato nel caso della notazione decimale (somma di cifre corrispondenti e riporti)

Es.:

101011_{bin} +	43_{dec} +
1001_{bin} =	9_{dec} =
----------	----
110100_{bin}	52_{dec}

CODIFICA DEI NUMERI INTERI SENZA SEGNO (1)

- **Sono quelli che i matematici chiamano *numeri naturali*: 0, 1, 2, …**
- **siccome, fra di essi non vi sono numeri negativi, non è necessario specificare il segno**
- **Il valore di un numero senza segno è codificato all'interno di un calcolatore con la notazione binaria**
- **Es.:**
  - il valore del numerosenza segno *tredici* è codificato dalla configurazione di bit 1101, che è proprio il numero *tredici* scritto in notazione binaria
  - il valore del numero senza segno *trentanove* è codificato dalla configurazione di bit 100110, che è proprio il numero *trentanove* scritto in notazione binaria

CODIFICA DEI NUMERI INTERI SENZA SEGNO (2)

- **ATTENZIONE!**

  I calcolatori sono dispositivi fisici finiti, quindi, all'interno dei calcolatori, lo "spazio" riservato alla rappresentazione dei numeri è finito…ma vi sono infiniti numeri: come è possibile rappresentarli tutti?

  NON E' POSSIBILE RAPPRESENTARE TUTTI I NUMERI IN UN CALCOLATORE!

  Non è possibile rappresentare nemmeno tutti i numeri interi senza segno: se ne rappresenta solo una parte, gli altri si approssimano (come accennato in seguito)

CODIFICA DEI NUMERI INTERI SENZA SEGNO (3)

- **Es.: con tre bit a disposizione, si possono rappresentare i seguenti interi senza segno:**

  $000_{bin} = 0_{dec}$

  $001_{bin} = 1_{dec}$

  $010_{bin} = 2_{dec}$

  $011_{bin} = 3_{dec}$

  $100_{bin} = 4_{dec}$

  $101_{bin} = 5_{dec}$

  $110_{bin} = 6_{dec}$

  $111_{bin} = 7_{dec}$

Cioè: si possono rappresentare tanti numeri interi senza segno quante sono le possibili configurazioni dei bit a disposizione

CODIFICA DEI NUMERI INTERI SENZA SEGNO (4)

- **In generale, con *n* bit, è possibile rappresentare 2^n numeri interi senza segno, vale a dire i numeri da 0 a 2^n - 1**

- **Es.: con 3 bit si rappresentano 2^3 numeri interi senza segno, vale a dire i numeri da 0 a 2^3 - 1(da 0 a 7)**

- **Il numero di bit effettivamente disponibili in un calcolatore per rappresentare gli interi senza segno può variare; tipicamente di usano:**

  16 bit (consente di rappresentare i numeri da 0_{dec} a 65535_{dec})

  32 bit (da 0_{dec} a 4294967295_{dec})

  64 bit (da 0_{dec} a $18446744073709551615_{dec}$)

CODIFICA DEI NUMERI INTERI SENZA SEGNO (5)

Per convincersi che la codifica ASCII non è adatta alla rappresentazione dei valori dei numeri, si consideri questo esempio:

Es.:

00110011 + (codifica ASCII di “3”)
00111000 = (codifica ASCII di “8”)

01101011 (codifica ASCII di “k”)

➔ la codifica ASCII dei numeri non consente operazioni aritmetiche!

CODIFICA DEI NUMERI INTERI CON SEGNO (1)

Finora abbiamo visto solo la codifica dei numeri interi non negativi...come si rappresentano gli interi negativi? ...più in generale, come si rappresentano i numeri interi con segno?

Supponiamo di avere a disposizione N bit

Idea! **Usiamo il primo bit per codificare il segno (es: 0 per + e 1 per -) e i restanti N-1 per il valore assoluto**

...come vedremo, non è una buona idea!

CODIFICA DEI NUMERI INTERI CON SEGNO (2)

- **Es.: con N=4, potremmo codificare i numeri così:**

Positivi	**Negativi**
$0000 = +0_{dec}$	$1000 = -0_{dec}$
$0001 = +1_{dec}$	$1001 = -1_{dec}$
$0010 = +2_{dec}$	$1010 = -2_{dec}$
$0011 = +3_{dec}$	$1011 = -3_{dec}$
$0100 = +4_{dec}$	$1100 = -4_{dec}$
$0101 = +5_{dec}$	$1101 = -5_{dec}$
$0110 = +6_{dec}$	$1110 = -6_{dec}$
$0111 = +7_{dec}$	$1111 = -7_{dec}$

CODIFICA DEI NUMERI INTERI CON SEGNO (3)

Questa idea non funziona!

Infatti:

1. **vi sono due diverse rappresentazioni per 0 (+0 e –0)**
2. **non valgono le usuali regole dell'addizione operazioni aritmetiche, es.:**

$$0010 + (+2_{dec})$$
$$1011 = (-3_{dec})$$
$$-------$$
$$1101 \quad (-5_{dec} \text{ ERRATO!})$$

➔ **la somma di numeri con segno richiederebbe, in un calcolatore, circuiti elettronici diversi da quelli utilizzati per sommare due numeri senza segno**

CODIFICA DEI NUMERI INTERI CON SEGNO (4)

- **Per ovviare a questi inconvenienti, si possono usare due diverse codifiche (entrambe in uso):**

  1. la codifica "in **complemento a due**" (non duplica la rappresentazione dello zero e consente di sommare due numeri con segno nello stesso modo [➔ con gli stessi circuiti elettronici] in cui si sommano due numeri senza segno)
  2. la codifica "in eccesso" (non duplica la rappresentazione dello zero, ma la somma di due numeri con segno deve essere effettuata in modo diverso da quella di due numeri senza segno)

- **esiste anche il complemento a uno (che non è più utilizzata e che non vedremo)**

CODIFICA DEI NUMERI INTERI CON SEGNO (COMPLEMENTO A DUE) (1)

CODIFICA IN COMPLEMENTO A DUE (non importa ricordare come funziona...il funzionalmento viene qui presentato solo per completezza)

- **Si fissa un'ampiezza di *N* bit**
- **Il bit piu' significativo (a sinistra) determina il segno: 0 corrisponde a +, 1 a -**
- **Se il numero è positivo gli N-1 bit restanti sono la codifica binaria (vista in precedenza)**
- **Se il num. e' negativo:**
- **si considera la rappresentazione in complemento a due del numero positivo con lo stesso valore assoluto**
- **si invertono tutti i bit di tale rappresentazione (gli "0" diventano "1" e viceversa)**
- **si somma 1 al risultato ottenuto al passo precedente**

CODIFICA DEI NUMERI INTERI CON SEGNO (COMPLEMENTO A DUE) (2)

- **Es.: supponiamo N=4. Il numero $+5_{dec}$ è così codificato:**

  0**101**

Il numero -5_{dec} è così codificato:

1. **si considera la codifica di $+5_{dec}$: 0101**
2. **si invertono tutti i bit: 1010**
3. **si somma 1: 1010 +**

1 =

1011

1**011 è la rappresentazione in complemento a due di -5_{dec}**

N.B. Il primo bit è sempre riservato al segno!

CODIFICA DEI NUMERI INTERI CON SEGNO (COMPLEMENTO A DUE) (3)

- **In questo modo, con N bit si rappresentano i numeri da**

  -2^{N-1} a $+2^{N-1}-1$

- **Es.: con N=4, si codificano i seguenti numeri:**

Non negativi	Negativi
$\mathbf{0}000 = +0_{dec}$	$1000 = -8_{dec}$
$\mathbf{0}001 = +1_{dec}$	$1001 = -7_{dec}$
$\mathbf{0}010 = +2_{dec}$	$1010 = -6_{dec}$
$\mathbf{0}011 = +3_{dec}$	$1011 = -5_{dec}$
$\mathbf{0}100 = +4_{dec}$	$1100 = -4_{dec}$
$\mathbf{0}101 = +5_{dec}$	$1101 = -3_{dec}$
$\mathbf{0}110 = +6_{dec}$	$1110 = -2_{dec}$
$\mathbf{0}111 = +7_{dec}$	$1111 = -1_{dec}$

CODIFICA DEI NUMERI INTERI CON SEGNO (COMPLEMENTO A DUE) (4)

Si noti (come già detto):

1. **nella codifica in complemento a due c'è una sola rappresentazione dello zero;**
2. **la somma può essere effettuata nello stesso modo usato per sommare due numeri senza segno ; es.:**

   0010 + ($+2_{dec}$)

   1101 = (-3_{dec})

   1111 (-1_{dec} CORRETTO!)

CODIFICA DEI NUMERI INTERI CON SEGNO (IN ECCESSO) (1)

- **Con n bit, si ha una codifica in eccesso 2^{n-1}**
- **Es.: con 8 bit si ha una codifica in eccesso 128 (cioè 2^{8-1})**
- **Il "trucco" utilizzato è il seguente:**
  - si usano gli n bit per rappresentare gli interi da -2^{n-1} a $+ 2^{n-1}-1$ (es., con 8 bit, si rappresentano gli interi da –128 a +127)
  - anziché codificare direttamente il numero, si codifica in notazione binaria il numero più l'eccesso (con 8 bit, si codifica in notazione binaria il numero più il valore 128)
- **Si noti che si può usare la notazione binaria per codificare direttamente la somma fra il numero da rappresentare più l'eccesso perché il risultato di tale somma non è mai negativo!**

CODIFICA DEI NUMERI INTERI CON SEGNO (IN ECCESSO) (2)

Es. (supponendo di avere a disposizione 8 bit ➔ eccesso 128):

- **per rappresentare -8_{dec} , prima si calcola la somma**

  $-8_{dec} + 128_{dec} = 120_{dec}$ e poi si codifica tale risultato in notazione binaria, ottenendo la seguente configurazione di bit: 01111000

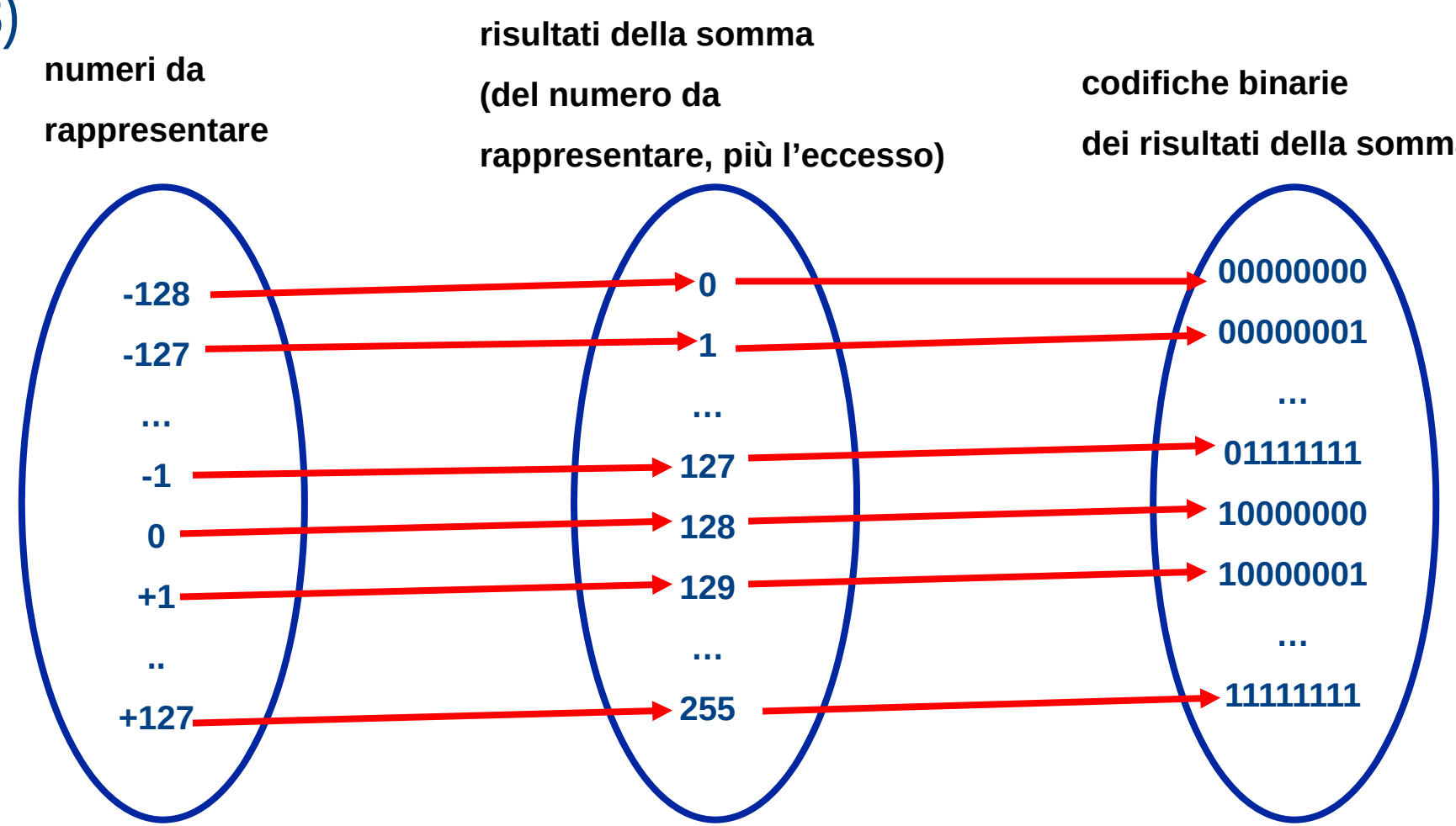
- **per rappresentare + 8_{dec}, prima si calcola la somma**

  $+8_{dec} + 128_{dec} = 136_{dec}$ e poi si codifica tale risultato in notazione binaria, ottenendo la seguente configurazione di bit: 10001000

- **per rappresentare , 0_{dec} prima si calcola la somma**

  $0_{dec} + 128_{dec} = 128_{dec}$ e poi si codifica tale risultato in notazione binaria, ottenendo la seguente configurazione di bit: 10000000

CODIFICA DEI NUMERI INTERI CON SEGNO (IN ECCESSO) (3)



CODIFICA DEI NUMERI INTERI CON SEGNO (IN ECCESSO) (4)

- **Si noti (come già detto):**

  1. non c'è duplicazione della rappresentazione dello zero

  2. la somma di due numeri con segno deve essere effettuata in modo diverso da quella di due numeri senza segno

- **Nonostante il punto 2, la codifica in eccesso trova applicazione nella codifica dei numeri con la virgola (o con valore assoluto molto grande), come vedremo**

CODIFICA DEI NUMERI CON LA VIRGOLA (1)

Sappiamo rappresentare numeri interi (sia senza segno che con segno)...e i numeri con la virgola?

...vedremo i cosiddetti numeri a virgola mobile (floating point)

Per semplicità, per presentare l'idea di base, consideriamo inizialmente il sistema decimale

Osservazione: un numero con la virgola (con un numero FINITO di cifre non nulle dopo la virgola) è sempre esprimibile come una frazione con una potenza di dieci come denominatore, es.

$+37,12 = +3712/100 = +3712/ 10^2$

o, analogamente: $+37,12 = +3712*10^{-2}$

CODIFICA DEI NUMERI CON LA VIRGOLA (2)

+12,52 = +1252/100 = +1252 * 10^{-2}

Fissata la base (in questo esempio, '10'), un numero con la virgola (con un numero finito di cifre non nulle dopo la virgola) è quindi esprimibile come una coppia: <mantissa, esponente>, dove sia la mantissa che l'esponente hanno un segno, ad es.:

+12,52 = <+1252; -2>

+0,128 = +128/1000 = +128 * 10^{-3} è esprimibile con la coppia <+128; -3>

CODIFICA DEI NUMERI CON LA VIRGOLA (3)

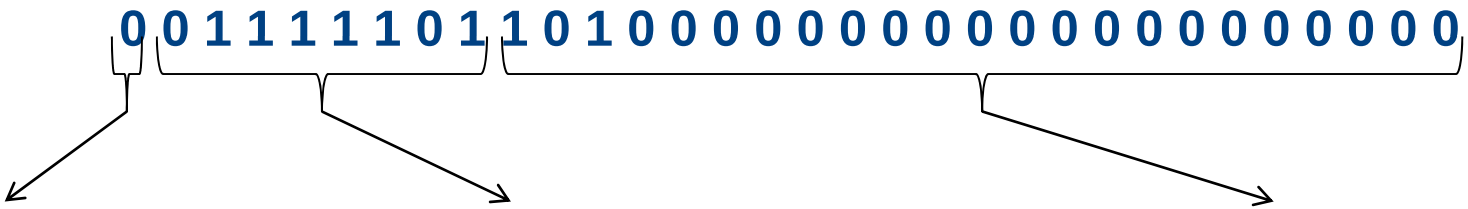
Il principio su cui si basa la codifica dei numeri con la virgola è analogo, con l'importante differenza che la base è 2 anziché 10

Generalmente, tali numeri (detti *floating point*) sono codificati in un calcolatore utilizzando 32 bit (nei numeri detti 'a precisione singola') o 64 bit (nei numeri detti 'a precisione doppia'), in cui:

1. il primo bit è usato per rappresentare il segno ('0' positivo e '1' negativo) del numero
2. un certo numero di bit sono utilizzati per la codifica in eccesso dell'esponente (8 bit nella precisione singola e 11 bit nella precisione doppia)
3. i restanti bit sono utilizzati per la codifica in notazione binaria del valore assoluto della mantissa (23 bit nella precisione singola e 52 bit nella precisione doppia)
4. la mantissa rappresenta solo la sequenza di cifre (binarie!) dopo la virgola: il numero prima della virgola si assume sempre uguale a 1

CODIFICA DEI NUMERI CON LA VIRGOLA (4)

Esempio, consideriamo il seguente numero floating point a precisione singola:



segno: +

esponente (8 bit, ➔ eccesso 2^7=128):

01111101_{bin}= 125_{dec}

➔ $125_{dec} - 128_{dec} = -3_{dec}$

mantissa:

$1{,}101_{bin}$= $(1+1*2^{-1} + 0*2^{-2} + 1*2^{-3})_{dec}$ =

$1{,}625_{dec}$

$+(1{,}625*2^{-3})_{dec} = +0{,}203125_{dec}$

CODIFICA DEI NUMERI (CONSIDERAZIONI CONCLUSIVE)

- **Interi con e senza segno: siccome il calcolatore dispone di risorse limitate, solo un sottoinsieme finito dei numeri interi è effettivamente rappresentabile...gli altri, cioè quelli con valore assoluto "molto grande" (nel senso che la rappresentazione del loro valore esatto richiederebbe più bit di quelli effettivamente disponibili), si possono rappresentare solo in forma approssimata (ad esempio, come floating point)**

- **discorso analogo vale per i numeri decimali finiti**

- **...e che dire del valore dei decimali periodici (es. $1{,}3333_{...dec}$) e degli irrazionali (es. $\sqrt{2}_{dec}$)? Non si rappresenta! ...o meglio: anche per questi, si può rappresentare solo un'approssimazione del loro valore.**

CODIFICA DELLE IMMAGINI (1)

- **Vi sono due approcci alla rappresentazioni delle immagini in un calcolatore:**

1. grafica raster (o bitmap): c'è una corrispondenza (più o meno diretta, a seconda degli approcci effettivamente adottati nell'organizzazione interna dei dati...ma noi non entreremo in questi dettagli) fra la visualizzazione delle immagini per mezzo di periferiche raster (es. monitor) e l'organizzazione dei bit che ne costituiscono la codifica

2. grafica vettoriale: le immagini sono descritte (in un linguaggio testuale) a partire da elementi geometrici di base e loro proprietà, specificate in termini matematici

CODIFICA DELLE IMMAGINI (2)

Vedremo:

1. grafica raster:
   a. generalità
   b. immagini in due soli colori (es., bianco e nero)
   c. immagini ‘a livelli di grigio’
   d. immagini a colori:
      i. true color
      ii. palette
   e. compressione dell’informazione (senza e con perdita di informazione)

2. grafica vettoriale: cenni

3. principali vantaggi e svantaggi dell’uno e dell’altro approccio e breve confronto fra i due
4. Codifica delle immagini in movimento (filmati): cenni

GRAFICA RASTER: GENERALITA' (1)

1. **L'immagine viene suddivisa in una griglia di quadratini tutti della stessa dimensione**
2. **Il colore (prevalente o medio) di ogni quadratino è codificato per mezzo di una sequenza di bit (la lunghezza della sequenza varia a seconda dei casi)**
3. **viene stabilita una convenzione che specifica in quale ordine sono codificati i quadratini (ad es., dal basso verso l'alto e da sinistra verso destra)**

GRAFICA RASTER: GENERALITA' (2)

- **Ogni quadratino che compone l'immagine è detto pixel (da 'picture element', elemento d'immagine)**

- **Siccome la codifica raster specifica un solo colore per ogni pixel, l'immagine effettivamente codificata è un'approssimazione di quella "reale". Nell'esempio seguente, un pixel in cui prevale il nero è considerato interamente nero e un pixel in cui prevale il bianco è considerato interamente bianco:**

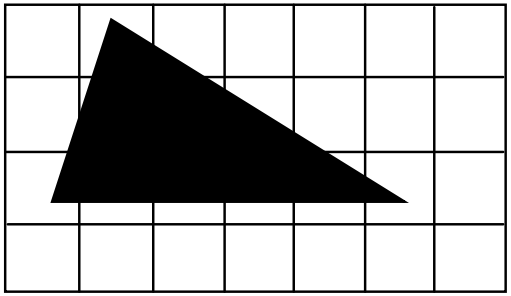


immagine "reale"

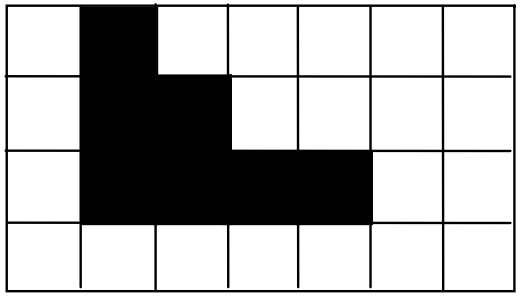
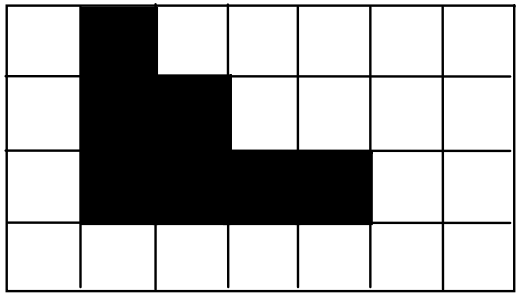


immagine codificata

GRAFICA RASTER: GENERALITA' (3)

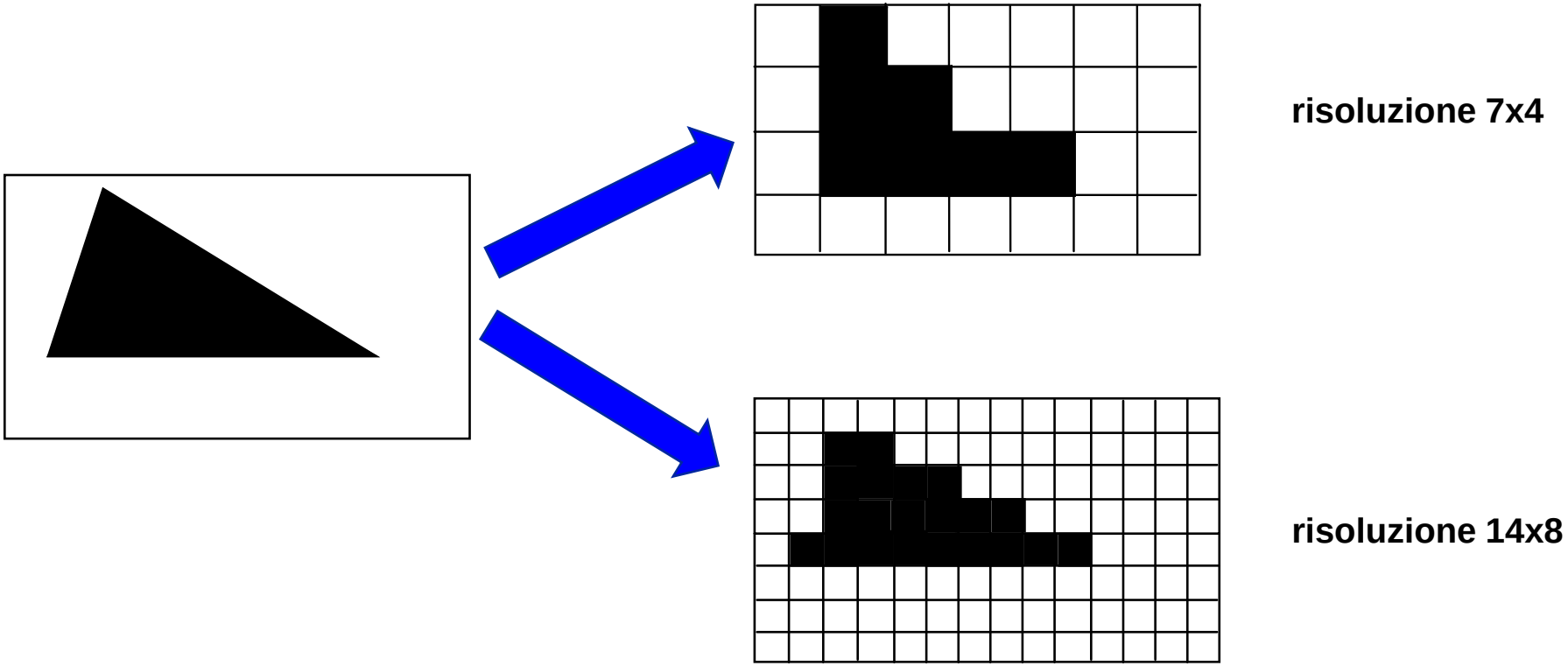
- **Il numero di pixel che compongono l'immagine è detto risoluzione**

- **La risoluzione di un'immagine è specificata indicando esplicitamente il prodotto del numero di pixel sul lato base per il numero di pixel sul lato altezza (immagini rettangolari), anziché il risultato di tale prodotto, ad es., la risoluzione dell'immagine seguente è 7x4:**



GRAFICA RASTER: GENERALITA' (4)

- **La qualità dell'immagine migliora se, a parità di dimensioni della superficie dell'immagine, aumenta la risoluzione (cioè se la griglia di pixel in cui l'immagine è suddivisa si infittisce):**



GRAFICA RASTER: GENERALITA’ (5)

L’ingrandimento di un dettaglio

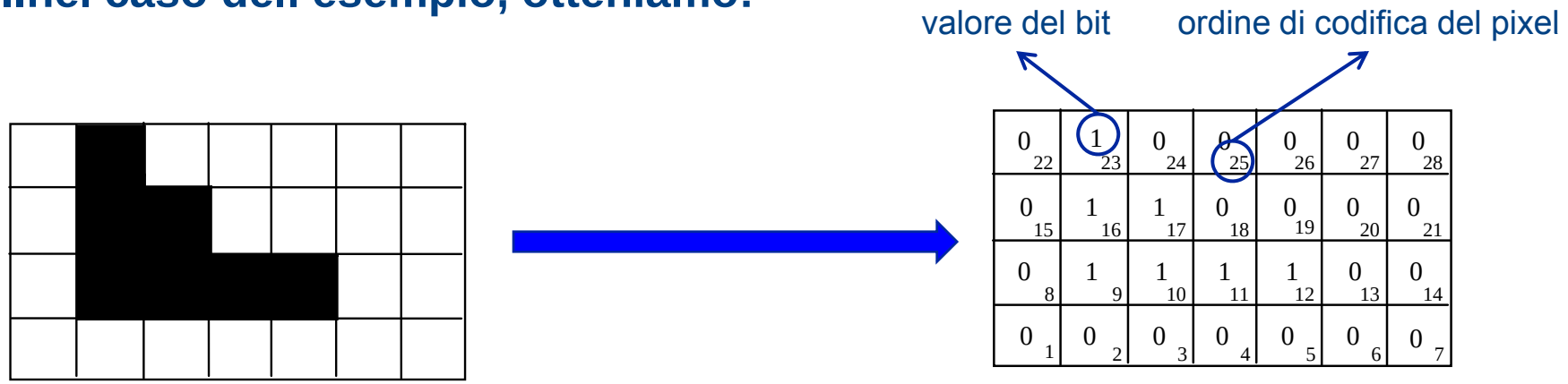
può mettere in evidenza i pixel

CODIFICA DELLE IMMAGINI A DUE COLORI (BIANCO E NERO) (1)

- **Ogni pixel può essere bianco oppure nero (un solo colore di due possibili)**
- **Immagini di questo tipo sono utilizzate, per esempio, per codificare una pagina di testo (caratteri neri su sfondo bianco), in cui il testo è trattato come un'immagine (➔ rappresentazione dei pixel che compongono i caratteri, non codifica dei caratteri stessi...)**
- **Per ogni pixel, è sufficiente un bit per codificarne il colore**
- **E' necessaria una convenzione che associ ad ogni colore un differente valore di bit (ed uno solo), ad es.:**
  - bianco → 0
  - nero → 1
- **Supponendo di attenersi alla convenzione di codificare i pixel seguendo l'ordine dal basso verso l'alto e da sinistra verso destra...**

CODIFICA DELLE IMMAGINI A DUE COLORI (BIANCO E NERO) (2)

- **...nel caso dell'esempio, otteniamo:**



CODIFICA DELL'IMMAGINE COME SEQUENZA DI VALORI DI BIT:

0000000 0111100 0110000 0100000

prima riga ... quarta riga

CODIFICA DELLE IMMAGINI A DUE COLORI (BIANCO E NERO) (3)

- **Occupazione di memoria: es., immagine con risoluzione 3072 x 2304 a due soli colori**
  - 1 bit * 3072 * 2304 = 7.077.888 bit = 884.736 bytes = 864 KB

CODIFICA DELLE IMMAGINI A LIVELLI DI GRIGIO (1)

- **Ogni pixel assume una sfumatura di grigio fra quelle possibili, nello spettro che va dal bianco al nero**
- **Immagini di questo tipo sono utilizzate nelle occasioni in cui si vogliono immagini dette “in bianco e nero” (da non confondere con il caso precedende!), come ad esempio in alcune stampe**
- **Per ogni pixel, si rappresenta il suo livello medio di grigio dell’immagine “reale”**
- **Il numero di livelli di grigio effettivamente rappresentabili dipende dal numero di bit utilizzati per codificare il livello di grigio di ogni singolo pixel**

CODIFICA DELLE IMMAGINI A LIVELLI DI GRIGIO (2)

- **Solitamente, si usa un byte (8 bit) per il livello di grigio di ogni singolo pixel ➔ una convenzione associa ad ogni livello di grigio una differente configurazione di 8 bit (ed una sola), ad es.:**
  - bianco → 00000000
  - ...
  - grigio di media intensità → 01111111
  - ...
  - nero → 11111111

- **In questo modo, è possibile rappresentare 2^8 = 256 livelli di grigio differenti**

CODIFICA DELLE IMMAGINI A LIVELLI DI GRIGIO (3)

- **Occupazione di memoria: es., immagine con risoluzione 3072 x 2304 a livelli di grigio (1 byte per ogni pixel):**
  - 1 byte* 3072 * 2304 = 7.077.888 bytes = 6912 KB = 6,75 MB

CODIFICA “TRUE COLOR” DELLE IMMAGINI A COLORI (1)

- **Analogo ai livelli di grigio, ma per riprodurre i colori si usano i livelli di 3 colori base[1]**

- **Due possibili codifiche: RGB (Red, Green, Blue) e CMY (Cyan, Magenta, Yellow)[2] ; ogni colore è ottenuto come la combinazione delle intensità di ciascuno dei 3 colori base ➔ ogni colore è rappresentato con 3 configurazioni di bit <x,y,z> che rappresentano ciascuno il livello di un colore base**

1 Per ogni pixel, oltre al colore, possono essere codificate altre informazioni, come ad esempio la trasparenza: qui considereremo solo il colore

2 In molte stampanti, il nero è un colore base: CMYK (Cyan, Magenta, Yellow, blacK)

CODIFICA “TRUE COLOR” DELLE IMMAGINI A COLORI (2)

- **Solitamente, si usa un byte per ogni colore base (cioè 256 livelli per ogni colore base), in questo modo è possibile rappresentare**

  $2^8 * 2^8 * 2^8 = 2^{24} = 16.777.216$ colori diversi

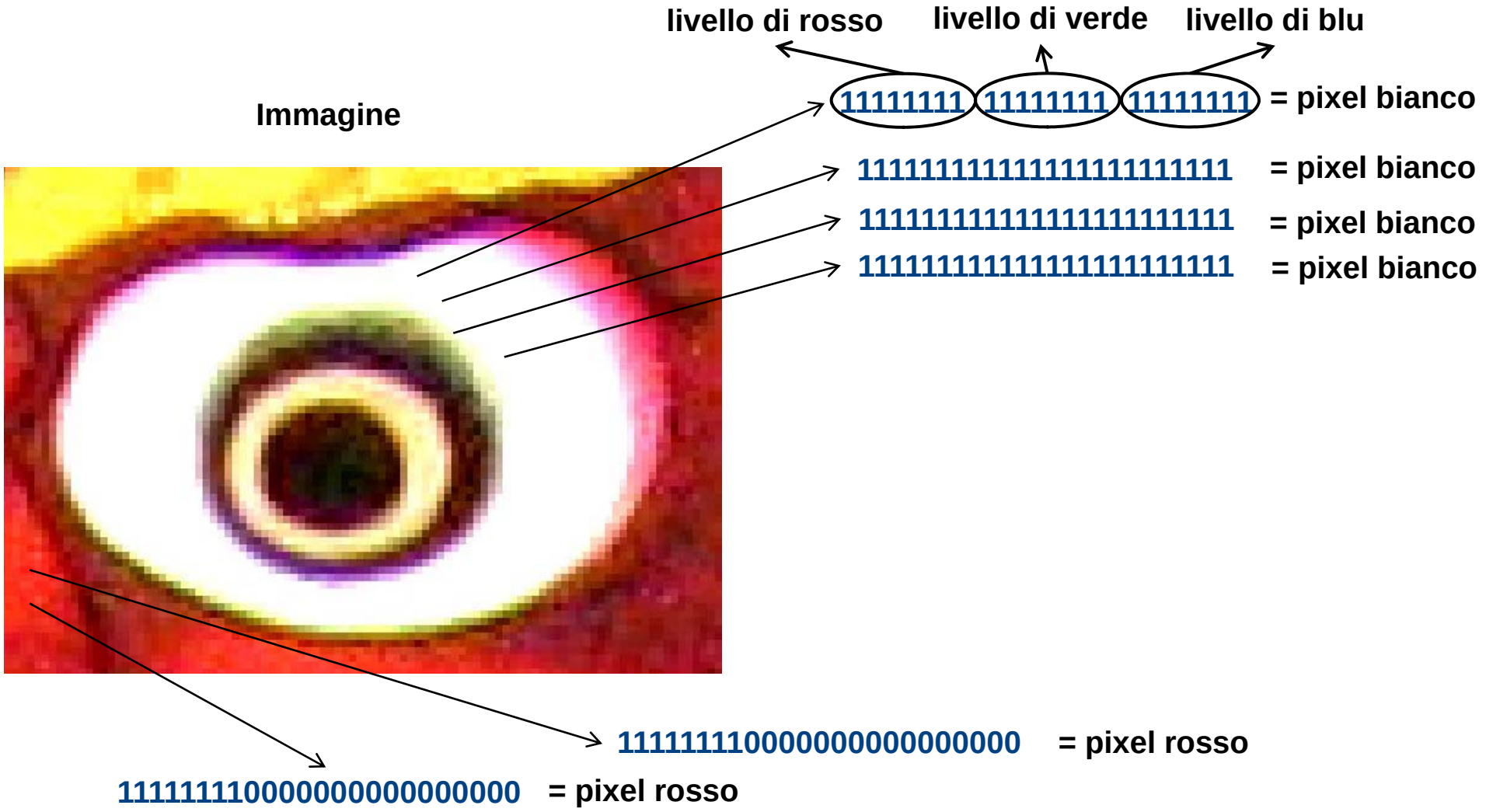
- **<u>Una convenzione associa ad ogni livello di ciascun colore base una differente configurazione di 8 bit (ed una sola)</u>**

CODIFICA “TRUE COLOR” DELLE IMMAGINI A COLORI (3)

- **Per ogni pixel, il colore è specificato con tre configurazioni di bit (una per ogni colore base), ad esempio, nel caso RGB:**
  -

Colore	R	G	B
Nero	00000000	00000000	00000000
...	...	...	...
Rosso base	11111111	00000000	00000000
...	...	...	...
Verde base	00000000	11111111	00000000
...	...	...	...
Blu base	00000000	00000000	11111111
...	...	...	...
Grigio di media intensità	10000000	10000000	10000000
...	...	...	...
Bianco	11111111	11111111	11111111

CODIFICA “TRUE COLOR” DELLE IMMAGINI A COLORI (4)



CODIFICA “TRUE COLOR” DELLE IMMAGINI A COLORI (5)

- **Occupazione di memoria: es., immagine con risoluzione 3072 x 2304 a colori, con codifica “true color” (3 bytes per ogni pixel)[1]:**
  - 3 bytes* 3072 * 2304 = 21.233.664 bytes = 20.736 KB = 20,25 MB

1 Come nei casi precedenti, supponiamo che per ogni pixel, sia rappresentato solo il colore

CODIFICA “CON PALETTE” DELLE IMMAGINI A COLORI (1)

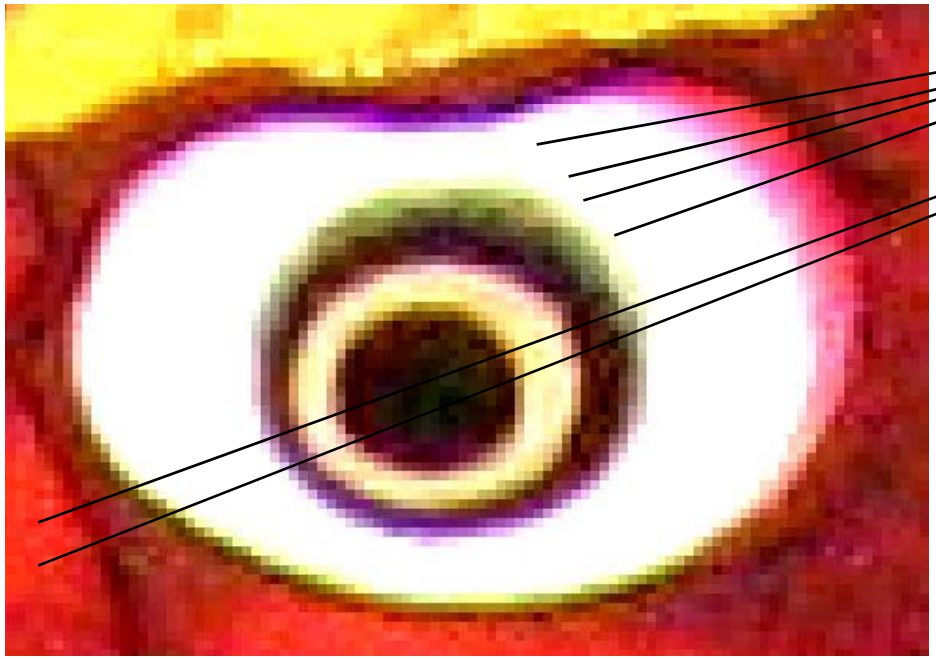
- **Nella codifica “true color”, si assume che, in ogni immagine, un pixel possa avere un qualunque colore fra quelli rappresentabili attraverso i livelli disponibili per i colori base**
- **Nella realtà, è piuttosto frequente che ogni immagine usi solo una piccola parte dei colori disponibili ➔ IDEA (che permette di risparmiare memoria):**
- **Per ogni immagine,**
  1. individuare l’insieme dei colori effettivamente utilizzati nell’immagine
  2. costruire una tabella (detta “**palette**” in inglese o “**tavolozza**” in italiano) contenente tali colori (ciascuno codificato, come nel caso precedente, con tre configurazioni di bit ciascuna dei quali specifica un livello per un colore base)
  3. attribuire un numero univoco a ciascuno di questi colori
  4. per ogni pixel, specificare il suddetto numero che ne identifica il colore nella tavolozza
- **...naturalmente, immagini diverse possono far riferimento a tavolozze diverse**

CODIFICA “CON PALETTE” DELLE IMMAGINI A COLORI (2)

- **Solitamente, si usano tavolozze di 256 (= 2^8) colori, in questo modo, è sufficiente un byte (cioè 8 bit) per identificare ciascun colore presente nella tavolozza**
- **...se la figura contiene più di 256 colori, con opportuni criteri, vengono inseriti nella tavolozza i colori più appropriati, in modo da ottenere una codifica dell’immagine quanto più possibile accettabile, anche se approssimata**
- **Questo approccio è poco adatto alle immagini fotografiche (che spesso utilizzano molti colori), per le quali si preferisce una codifica “true color”**

CODIFICA “CON PALETTE” DELLE IMMAGINI A COLORI (3)

Immagine



Numero colore

00000000

00000001

…

Tavolozza

R	G	B
11111111	11111111	11111111
11111111	00000000	00000000
…	…	…

CODIFICA “CON PALETTE” DELLE IMMAGINI A COLORI (4)

- **<u>Occupazione di memoria</u>: es., immagine con risoluzione 3072 x 2304 a colori, con codifica “con palette” con 256 colori[1]. Per codificare l’immagine assieme alla propria tavolozza, sono necessari:**
  - 1 byte* 3072 * 2304 + 3 bytes * 256 = 7.079.680 bytes = 6.913,75 KB ≈ 6,75 MB

codifica dell’immagine vera e propria — **codifica della tavolozza**

- **Si noti che un’analoga immagine codificata in “true color” richiede 20,25 MB**

1 Come nei casi precedenti, supponiamo che, per ogni pixel, sia rappresentato solo il colore

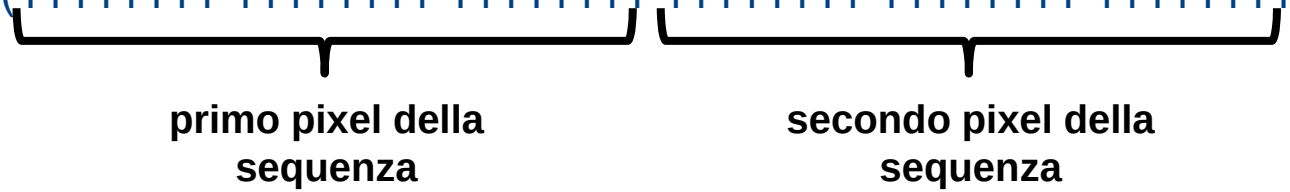
COMPRESSIONE DELL'INFORMAZIONE (1)

- **Le codifiche delle immagini richiedono molta memoria (specialmente quelle in "true color")**
- **➔ esistono tecniche che consentono di codificare le immagini risparmiando memoria...tali tecniche sono dette di compressione dell'informazione**
- **Gli applicativi che visualizzano le immagini, per visualizzare immagini codificate con tecniche di compressione, prima "decomprimono" la codifica, ottenendo una codifica decompressa che può essere immediatamente visualizzata**

COMPRESSIONE DELL'INFORMAZIONE (2)

- **Due tipi di compressione:**
  1. compressione senza perdita (di informazione): tecniche di compressione di questo tipo consentono di codificare un'immagine in un formato compresso (quindi, generalmente, risparmiando memoria), tale che la decompressione della codifica compressa consente di riprodurre esattamente l'immagine di partenza (cioè la codifica compressa, contiene tutte le informazioni necessarie a "ricostruire" l'immagine di partenza anche se rappresentate "in maniera sintetica")...questo è il principale vantaggio di questo tipo di compressione.

     Ad esempio: una sequenza di 1000 pixel bianchi consecutivi in un'immagine a colori, codificata in un formato "true color" non compresso con 3 bytes per pixel, sarebbe rappresentata specificando esplicitamente il colore di ciascuno dei 1000 pixel (11111111 11111111 11111111 11111111 11111111 11111111 ...) e



     richiederebbe 3000 bytes

COMPRESSIONE DELL'INFORMAZIONE (3)

Un modo ovvio (ed una "filosofia" adottata in alcuni meccanismi di compressione) per comprimere questa informazione è quello di codificarla ad esempio come "1000 X 11111111 11111111 11111111", anziché ripetere 1000 volte lo stesso dato.

Uno dei principali svantaggi di questo approccio è che non sempre esso consente un sufficiente risparmio di memoria

2. compressione con perdita (di informazione): tecniche di compressione di questo tipo consentono un maggior risparmio di memoria, a prezzo della perdita di un po' di informazione. In questo caso, la decompressione della codifica compressa non con consente di riprodurre esattamente l'immagine di partenza, ma solo una sua approssimazione (cioè la codifica compressa, manca di alcune informazioni necessarie a "ricostruire" l'immagine di partenza). Questi meccanismi di compressione sono piuttosto sofisticati e mirano ad una codifica in cui l'informazione persa sia poco rilevante per un osservatore umano (ad esempio, l'occhio umano non è generalmente in grado di apprezzare sottili differenze di colore in pixel adiancenti...), in modo che l'immagine riprodotta, pur non essendo fedele all'originale, sia comunque di buona qualità per un osservatore umano

ALCUNI FORMATI DI CODIFICA RASTER PER IMMAGINI A COLORI

- **JPEG (Joint Photographic Experts Group)**
- **GIF (Graphic Interchange Format)**
- **BPM (BitMaP)**
- **TIFF (Tagged Image File Format)**
- **PNG (Portable Network Graphic)**
- **...**

GRAFICA VETTORIALE

- **Diversamente da quanto avviene nella grafica raster, in quella vettoriale l'immagine non è descritta pixel per pixel, ma è descritta (in un linguaggio testuale) a partire da elementi geometrici di base (linee, rettangoli, curve geometriche, ecc.) e loro proprietà (colore interno, colore del contorno, ecc.), specificate in termini matematici**

- **Gli elementi di un'immagine possono essere ruotati, dilatati, contratti, traslati, ecc.**

ALCUNI FORMATI DI CODIFICA VETTORIALE E IBRIDI

- **Fra i formati di codifica vettoriale, troviamo CDR (CorelDRAW), SVG (Scalable Vector Graphic), DWG (DraWinG), ecc.**

- **Vi sono formati ibridi (raster + vettoriale), come Postscript, PDF (Portable Document Format), WMF (Windows MetaFile), EMF (Enhanced MetaFile)**

RASTER vs VETTORIALE (1)

- **La grafica raster è adatta a rappresentare immagini “irregolari” con molti colori (ad esempio, quelle fotografiche), per le quali, invece, la grafica vettoriale risulta inadatta**
- **La grafica vettoriale è adatta in applicazioni tipo CAD (Computer Aided Design)**
- **La grafica raster richiede generalmente più memoria, ma quella vettoriale richiede solitamente una maggiore eleborazione della codifica per visualizzare l’immagine**
- **L’ingrandimento di un’immagine raster “oltre un certo limite” produce sgranature (i pixel diventano più o meno evidenti), mentre un’immagine vettoriale può essere ingrandita a piacere senza perdere qualità (...i font dei caratteri negli elaboratori di testi, ad esempio, sono solitamente rappresentati secondo una codifica vettoriale, così da poterli ingrandire senza perdere qualità)**

RASTER vs VETTORIALE (2)

- **Monitor e stampante sono due esempi di periferiche di output raster (cioè che richiedono una codifica raster delle immagini per poterle visualizzare): la visualizzazione di un'immagine vettoriale tramite queste periferiche richiede un processo di rasterizzazione dell'immagine (processo piuttosto semplice)**

- **Il plotter è invece un esempio di periferica di output vettoriale (quindi, un'immagine vettoriale non necessita di rasterizzazione per essere stampata su un plotter)**

- **La vettorializzazione di un'immagine raster è un processo complicato**

CODIFICA DELLE IMMAGINI IN MOVIMENTO (FILMATI)

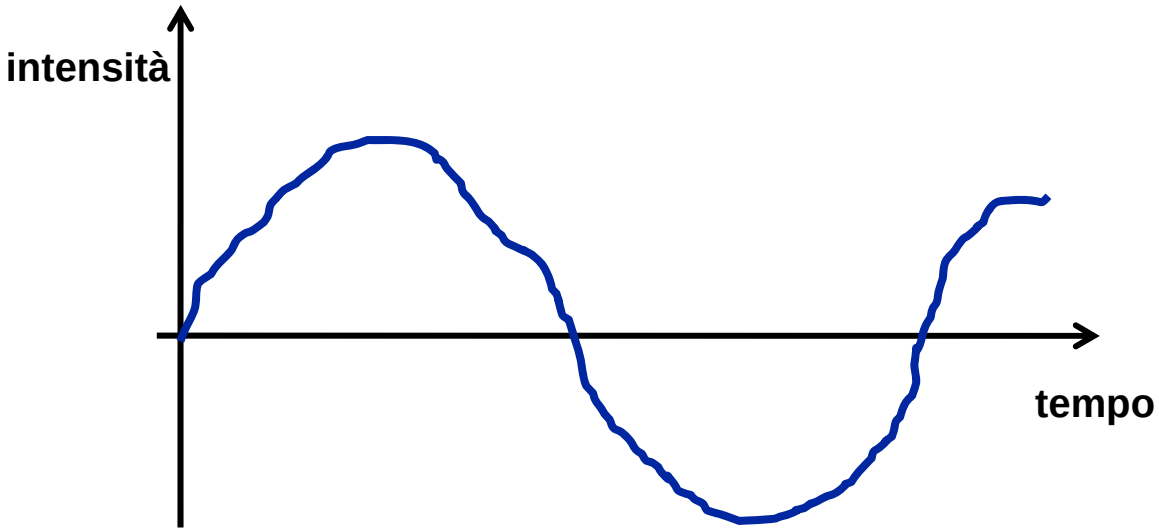
- **Un filmato (nella sua parte video) è costituito da una sequenza di immagini (fotogrammi)**
- **La riproduzione di un filmato è realizzata da applicazioni software che riproducono in rapida sequenza i fotogrammi che costituiscono il filmato, dando così l'illusione del movimento**
- **Per i filmati, è fondamentale contenere la quantità di memoria necessaria a codificarli ➔ si usano tecniche di compressione dell'informazione che comprimono sia la codifica di ogni singolo fotogramma (analogamente a quanto fatto per le immagini "statiche"), sia le sequenze di fotogrammi (ad esempio, si codificano interamente solo alcuni fotogrammi e i fotogrammi compresi fra due codificati interamente sono specificati solo codificando le differenza fra uno e quello che lo precede)**
- **Alcuni formati video: MPEG (Moving Picture Experts Group), AVI (Audio Video Interleave), ecc.**

CODIFICA DEI SUONI (1)

- **Il suono è un'onda meccanica che si propaga in un mezzo e che varia nel tempo con continuità**
- **Analogamente al caso delle immagini, la codifica digitale di un suono consente una rappresentazione approssimata del suono "reale"**
- **L'approssimazione deriva principalmente da due processi necessari alla codifica digitale del suono: campionamento e quantizzazione**

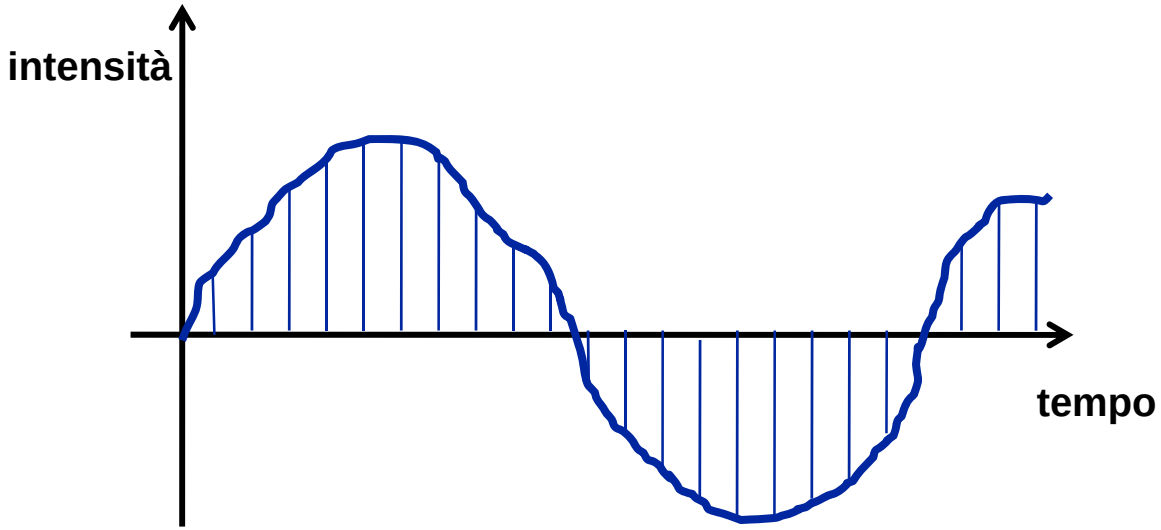
CODIFICA DEI SUONI (2)

Rappresentazione cartesiana della variazione nel tempo dell'intensità del suono



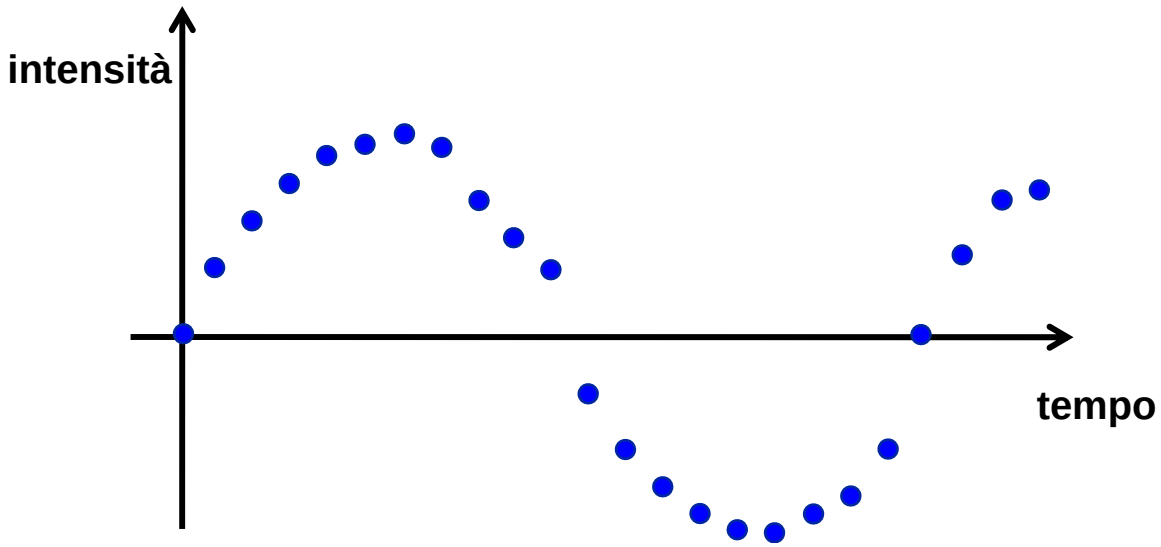
CODIFICA DEI SUONI (3)

Campionamento: misurazione dell'intensità del suono ad intervalli di tempo regolari...più frequenti sono le misurazioni, maggiore è la fedeltà del suono codificato a quello "reale" e maggiore è la quantità di memoria occupata dalla codifica (es. nei CD musicali la frequenza di campionamento è di 44.100 misurazioni al secondo)



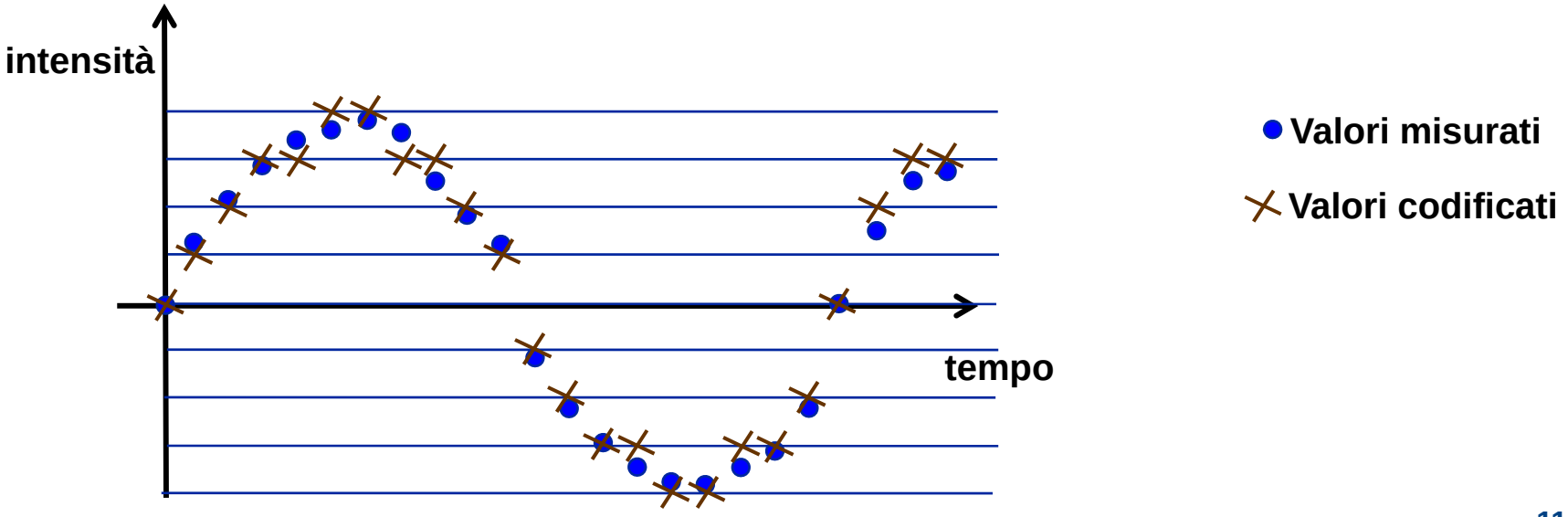
CODIFICA DEI SUONI (4)

Risultato del campionamento: sequenza finita di valori dell'intensità negli istanti di misurazione



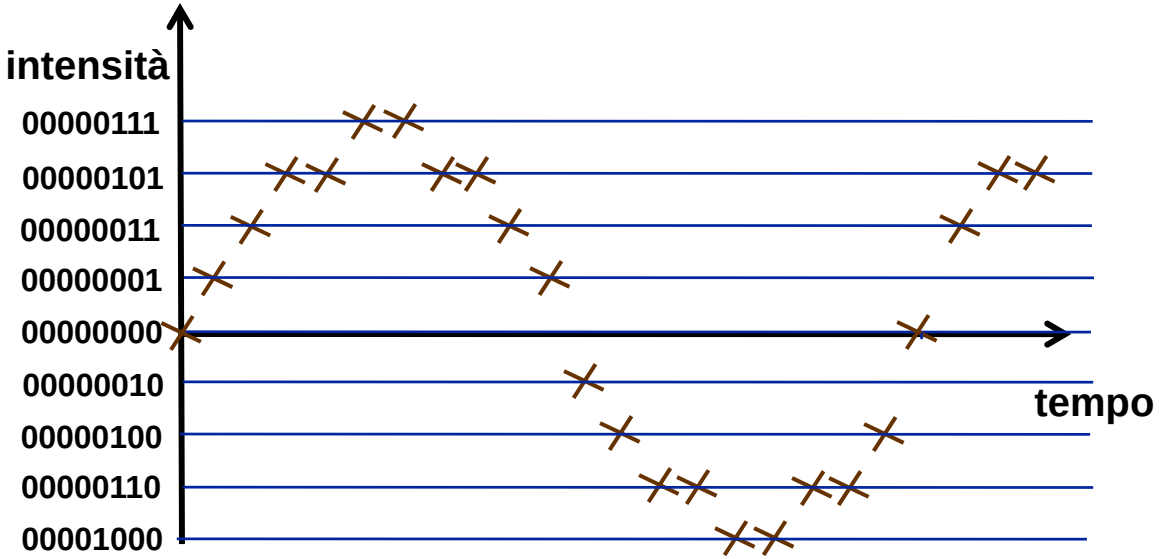
CODIFICA DEI SUONI (5)

Quantizzazione: l'intervallo di variazione dell'intensità viene suddiviso in una sequenza di intervalli tutti della stessa dimensione. Gli estremi degli intervalli sono le intensità discretizzate. La codifica non rappresenterà i valori esatti dell'intensità negli istanti di misurazione, ma approssimerà ciascuno di questi valori con la più vicina intensità discretizzata....più fine è la suddivisione in intervalli, maggiore è la fedeltà del suono codificato a quello "reale" e maggiore è la quantità di memoria occupata dalla codifica



CODIFICA DEI SUONI (6)

Ogni valore discretizzato è identificato da una sequenza di valori di bit (nell'esempio sono stati utilizzati 8 bit, ma nella realtà se ne possono usare anche di più, ad es., nei CD musicali si usano 16 o 32 bit)



Tralasciando alcuni dettagli, il suono dell'esempio potrebbe essere codificato nel seguente modo:

00000000 00000001 00000011 00000101 00000101 00000111 00000111 ...

CODIFICA DEI SUONI (7)

- **Anche nel caso dei suoni, si utilizzato tecniche di compressione dell'informazione che consentono di risparmiare memoria nella codifica**

- **Alcuni formati di codifica dei suoni: MP3 (MPEG Layer 3), MIDI (Musical Instrument Digital Interface), AAC (Advanced Audio Coding), ecc.**

I FILE COME BASE PER L'ARCHIVIAZIONE DELL'INFORMAZIONE

I FILE

- **Le informazioni digitalizzate (quindi rappresentate da sequenze di valori di bit) possono essere archiviate (memorizzate in maniera persistente) su supporti di memoria di massa, es. floppy, hard disk, cd, nastri, ecc.**
- **Una sequenza di valori di bit memorizzata come una singola unità si chiama “file”**
- **Per l’utente finale, ogni file ha un nome, spesso composto dal nome vero e proprio, più la cosiddetta estensione (separati da un punto ‘.’), es. “IntroInfCapp1-2.pptx”**
- **Solitamente l’estensione indica il tipo di file (ma non è detto che sia sempre così)**