RAPPRESENTAZIONE DELL'INFORMAZIONE

Prof. Ing. Donato Impedovo

Rappresentazione digitale Dell'informazione

codifica

Informazione /dati

rappresentazione digitale

decodifica

Mondo esterno

"Reale"

Computer: memorizzazione, elaborazione

Il BIT

Alfabeto del calcolatore costituito da due simboli: 0,1.

BIT (binary digit):

- Unità elementare di informazione.
- La cifra binaria può assumere solo due valori alternativi: 0 oppure 1.

Presenza	Assenza
Vero	Falso
1	0
Acceso	Spento
+	–
Sì	No
Favorevole	Contrario
Yang	Yin
Lisa	Bart

- Archiviato da un dispositivo digitale o un sistema fisico che esiste in uno di due possibili stati distinti.

  Es.:
  - i due stati stabili di un flip-flop,
  - due posizioni di un interruttore elettrico,
  - due distinte tensione o gli attuali livelli consentiti da un circuito,
  - due distinti livelli di intensità della luce,
  - due direzioni di magnetizzazione o di polarizzazione, ecc

Sequenze di BIT

Per poter rappresentare un numero maggiore di informazione si usano ***sequenze*** di bit

Il processo che fa corrispondere ad un dato reale una sequenze di bit prende il nome ***codifica dell'informazione***

Es.1: un esame può avere quattro possibili esiti: *ottimo, discreto, sufficiente, insufficiente.* Quanti bit sono necessari per codificare tale informazione?

due bit:

ottimo	00
discreto	01
sufficiente	10
insufficiente	11

Es.2: otto colori: *nero, rosso, blu, giallo, verde, viola, grigio, arancione*

tre bit:

nero	000
rosso	001
blu	010
giallo	011
verde	100
viola	101
grigio	110
arancione	111

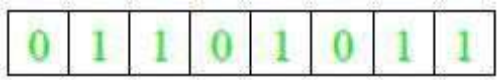
bit, Byte e word

- Con N bit si possono codificare 2^N stati differenti
- Per rappresentare N stati, devo usare almeno $\log_2(N)$ bit

Vogliamo codificare 19 colori… quanti bit servono??
ceil($\log_2(N)$) – arrotonda all'intero più grande

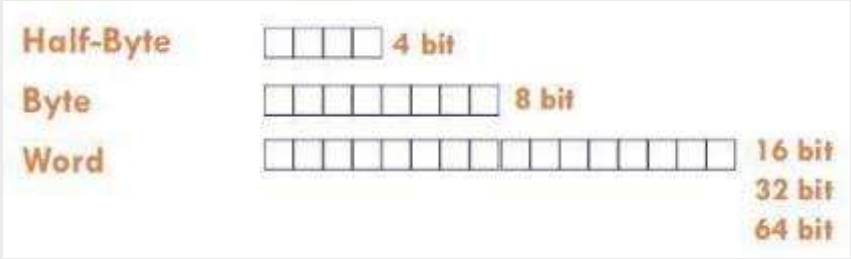
I sistemi moderni memorizzano e manipolano miliardi di bit: necessità di multipli.
Le informazioni sono rappresentate mediante stringhe di bit

- 8 bit (**b**) = 1 Byte (**B**)



OSSERVAZIONE: con la lettera *b*, minuscolo, si indicano i bit, con la lettera *B*, maiuscolo, si indicano i byte

Strutture logiche:



bit, Byte e multipli

I moderni sistemi memorizzano e manipolano molti bit e byte: necessità di multipli

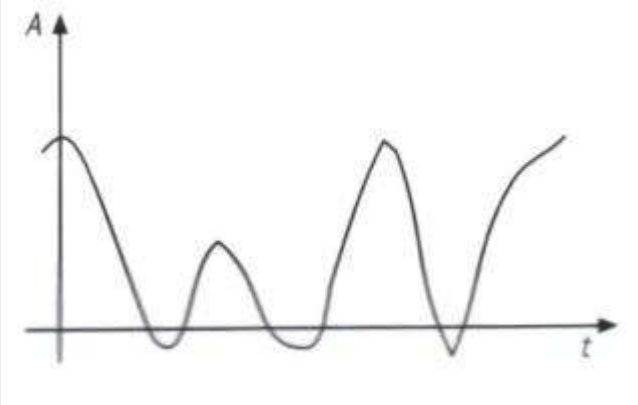
K – chilo (mille)
M – Mega (milioni)
G – Giga (miliardi)
T – Tera (migliaia di miliardi)

Nome	Sigla	In bit	In byte	In potenze di 2
Bit	Bit	1 bit	1/8	2^1=2 stati
Byte	Byte	8	1	2^8=256 stati
KiloByte	KB	8.192	1.024	2^{10} byte
MegaByte	MB	8.388.608	1.048.576	2^{20} byte
GigaByte	GB	8.589.934.592	1.073.741.824	2^{30} byte
TeraByte	TB	8.796.093.022.208	1.099.511.627.776	2^{40} byte

Esempio: un hard disk (HD) da 500GB contiene esattamente:
$500x2^{30} = 500x2^{10}x2^{10}x2^{10} = 500x1024x1024x1024 = 536.870.912.000$byte
E in termini di bit?

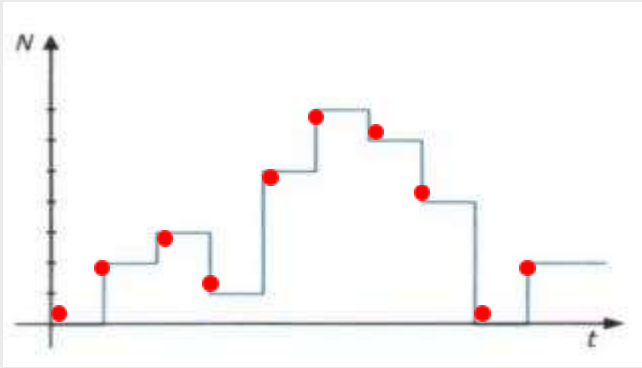
Rappresentazione digitale dei segnali

SEGNALE ANALOGICO: segnale tempo-continuo che può assumere tutti gli infiniti valori della grandezza fisica osservabile che rappresenta contenuti all'interno di un determinato range. I valori utili che lo rappresentano sono continui (infiniti) in un intervallo e non numerabili.



SEGNALE DIGITALE: segnale tempo-discreto che all'interno di un determinato range può assumere solo un numero discreto (numerabile finito) di valori.

Es.: onda quadra che assume i valori logici ALTO (1) e BASSO (0).



Vantaggi:

- utilizzo di un unico linguaggio (binario)
- costanza di qualità nel tempo
- possibilità di compressione

Rappresentazione digitale di segnali

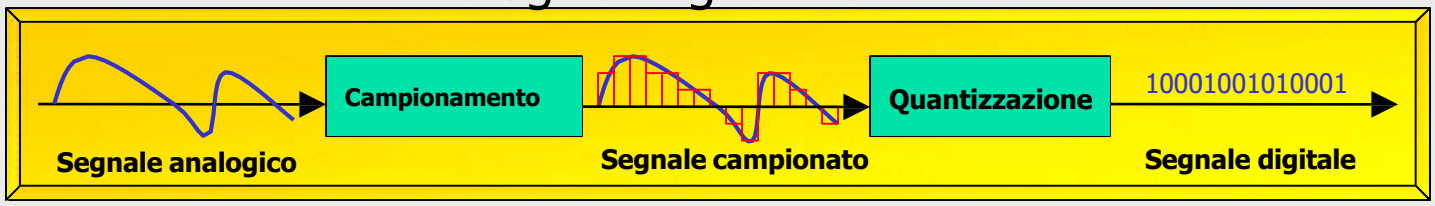
CAMPIONAMENTO:

- processo di conversione di un segnale tempo-continuo in un segnale tempo-discreto,
- L'ampiezza del segnale continuo viene considerata a intervalli di tempo regolari (T - periodo di campionamento).

QUANTIZZAZIONE:

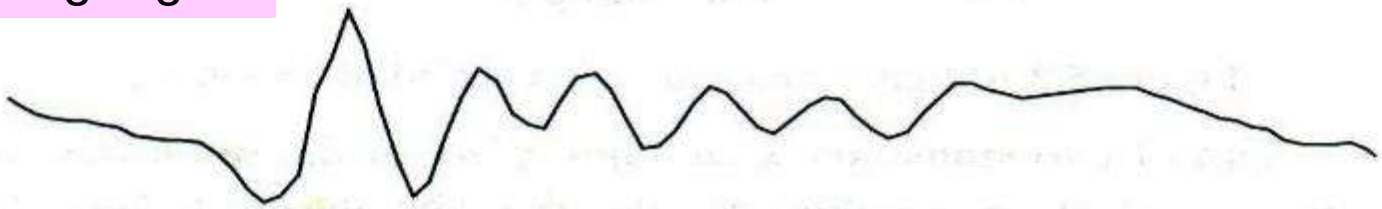
- processo di conversione di un segnale a valori continui in uno a valori discreti.
- Più è alto il numero di bit utilizzati nella quantizzazione e minore è l'errore che si commette (errore di quantizzazione), cioè si riduce la distanza media tra il valore campionato (continuo) e il corrispondente valore quantizzato

Analog to **D**igital **C**onverter



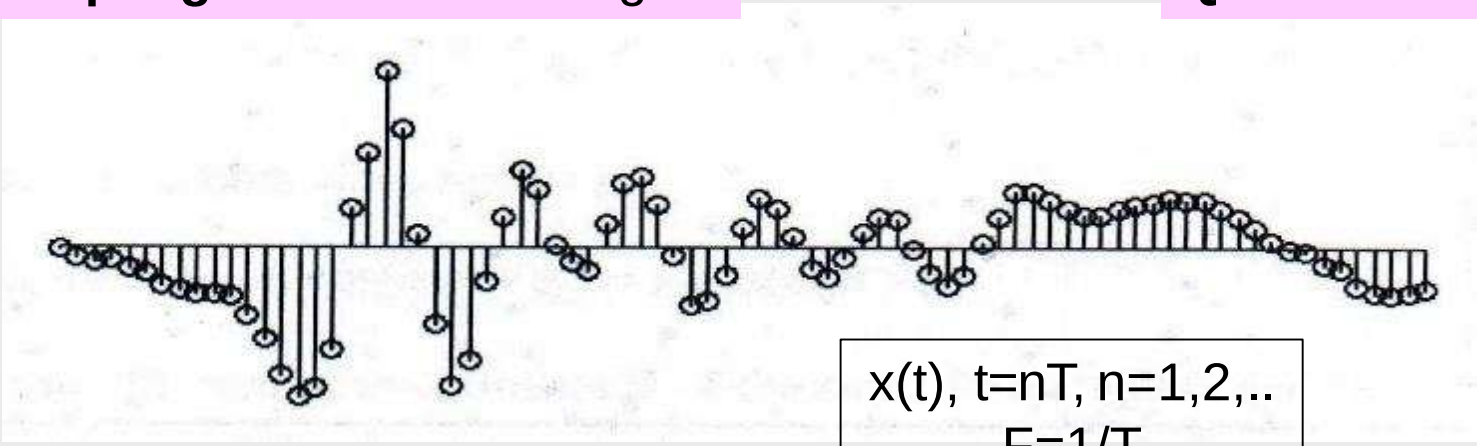
Rappresentazione digitale di segnali

Analog Signal



Sampling: Discrete-time Signal

Quantization



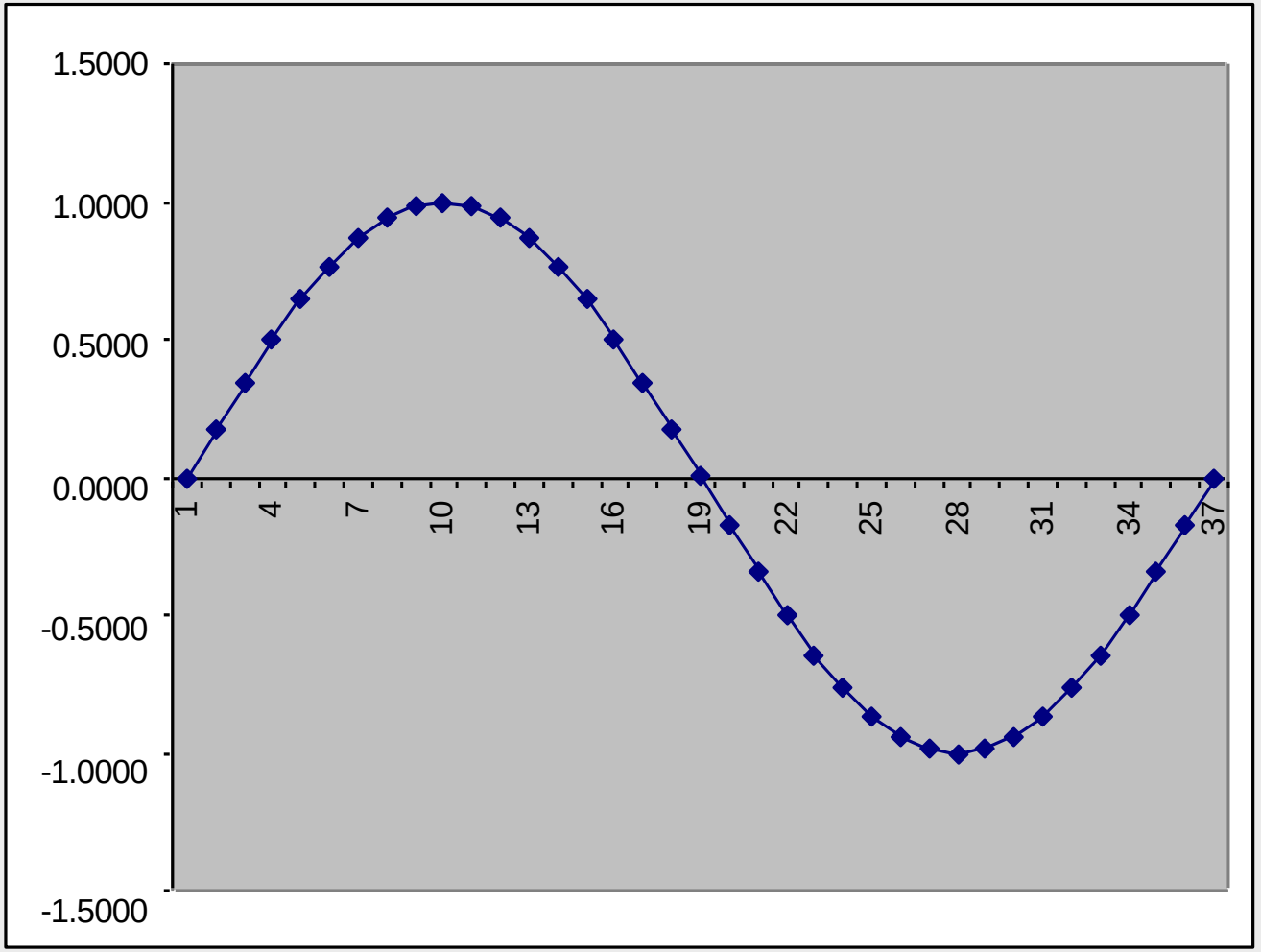
x(t), t=nT, n=1,2,..
F=1/T

Problemi legati al campionamento

Segnale sinusoidale a frequenza f_s=27 [Hz]

fc=1KHz

fc=37 fs



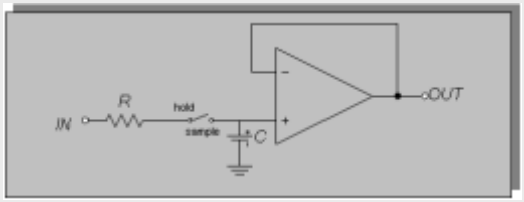
Problemi legati al campionamento

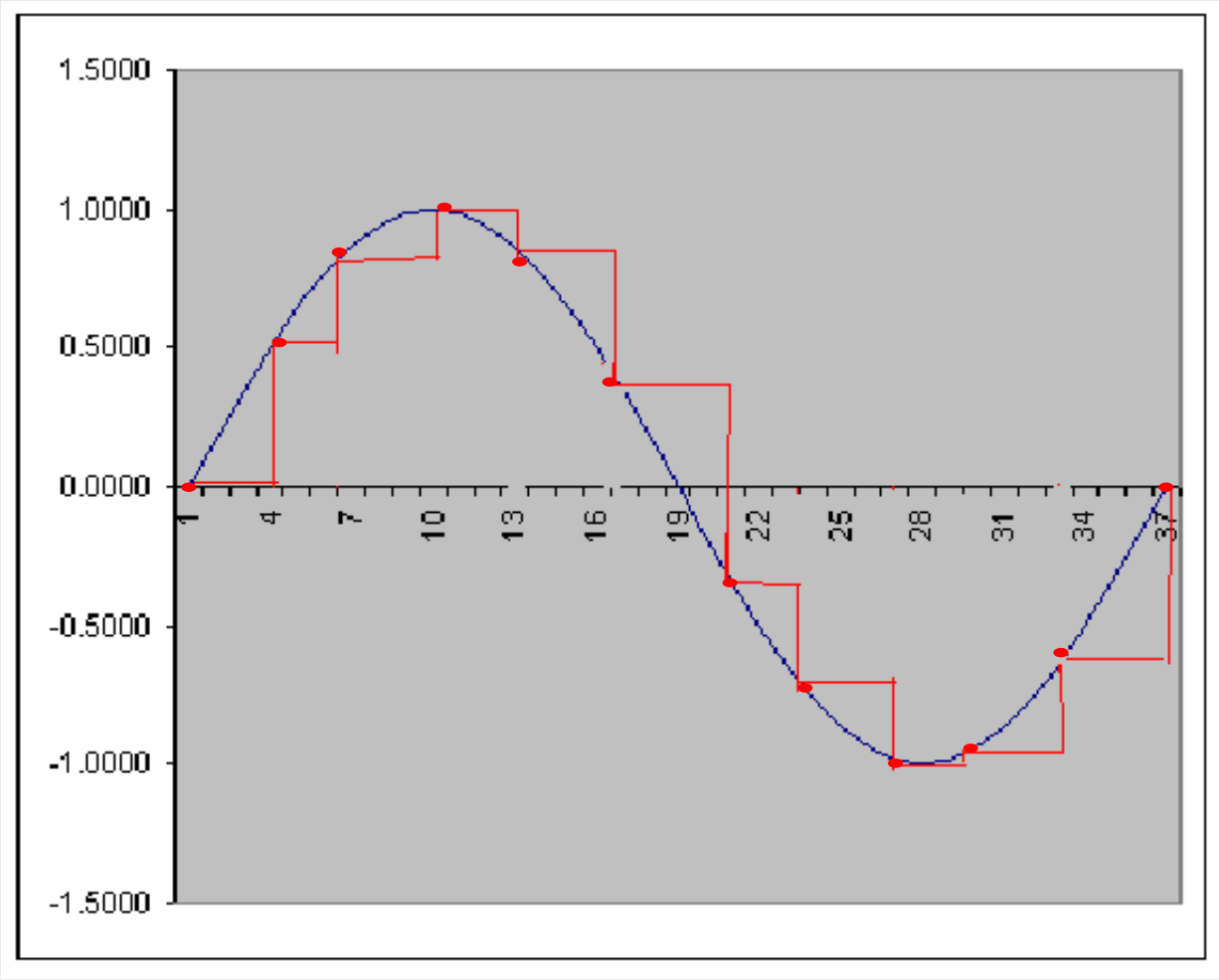
Segnale sinusoidale a frequenza f_s=27 [Hz]

fc=333 Hz

fc=12,33 fs

Sample and hold



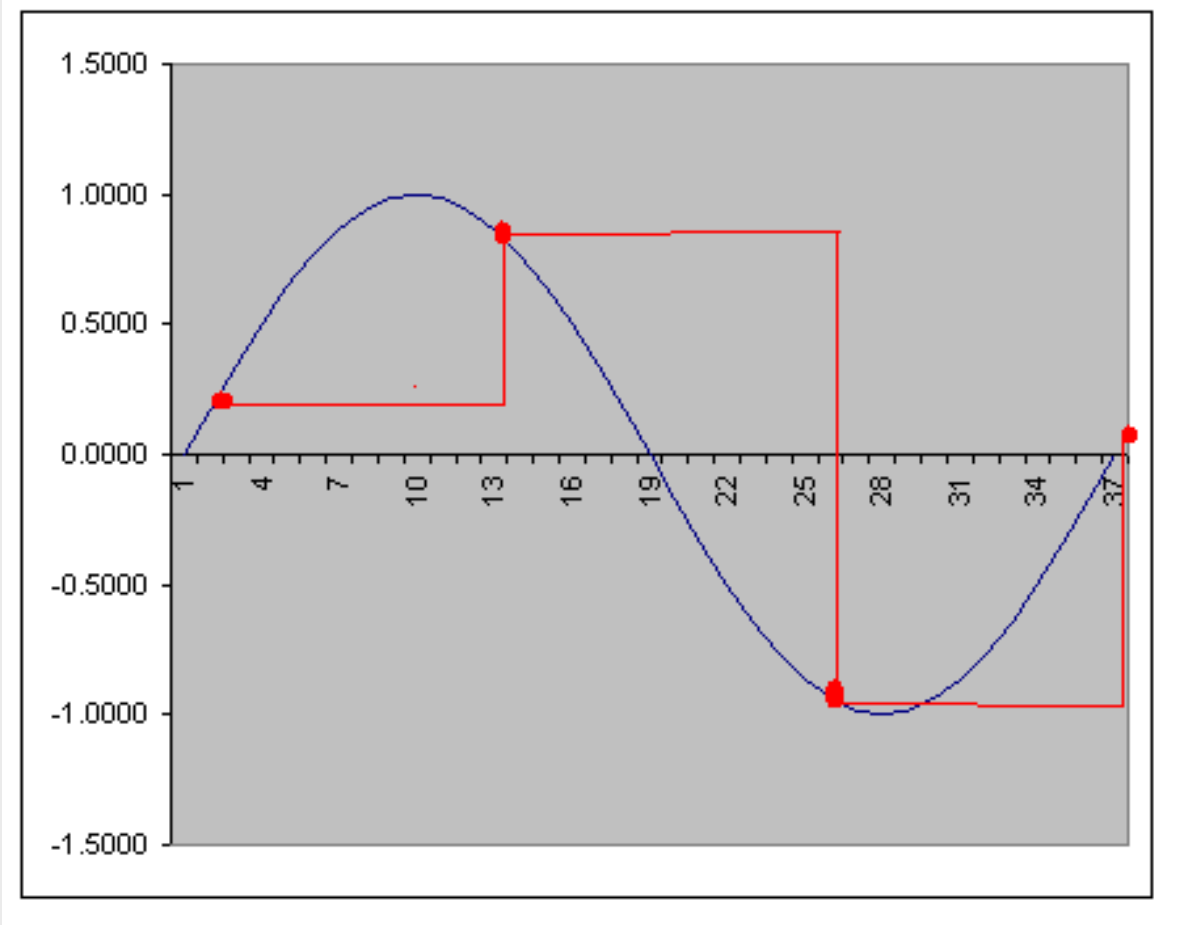


Problemi legati al campionamento

Segnale sinusoidale a frequenza f_s=27 [Hz]

fc=110 Hz

fc= 4 fs

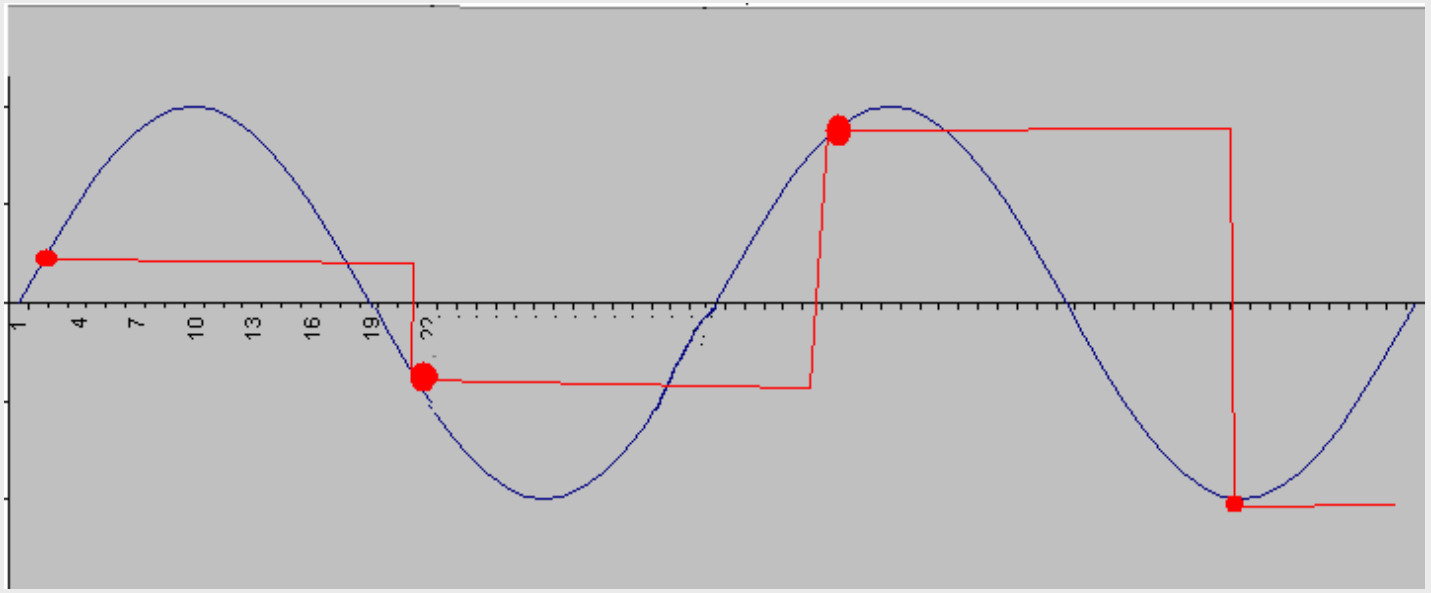


Problemi legati al campionamento

Segnale sinusoidale a frequenza f_s=27 [Hz]

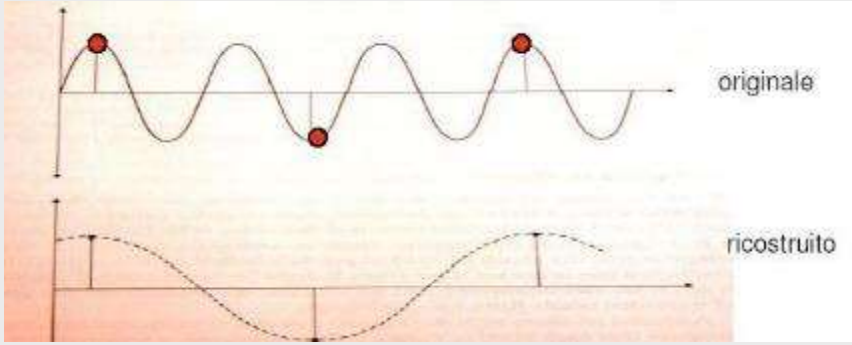
fc=50 Hz

fc=1,8 fs

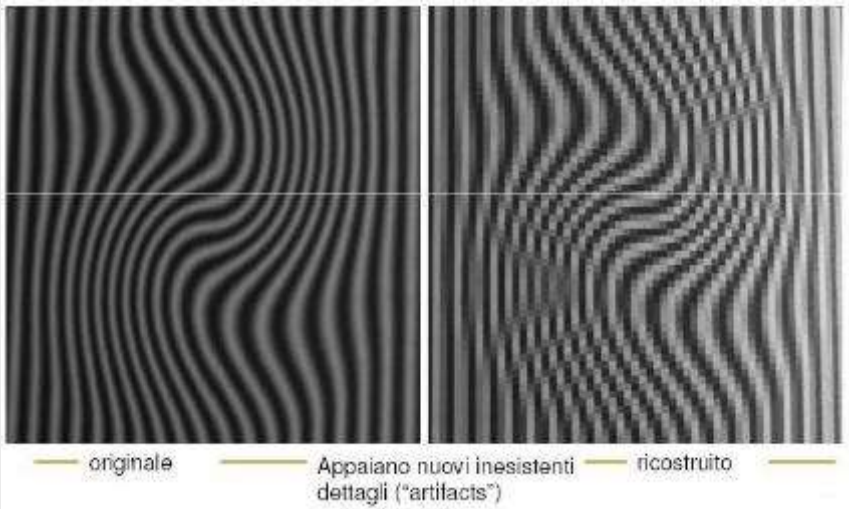


Problemi del campionamento

1. Perdita di informazioni

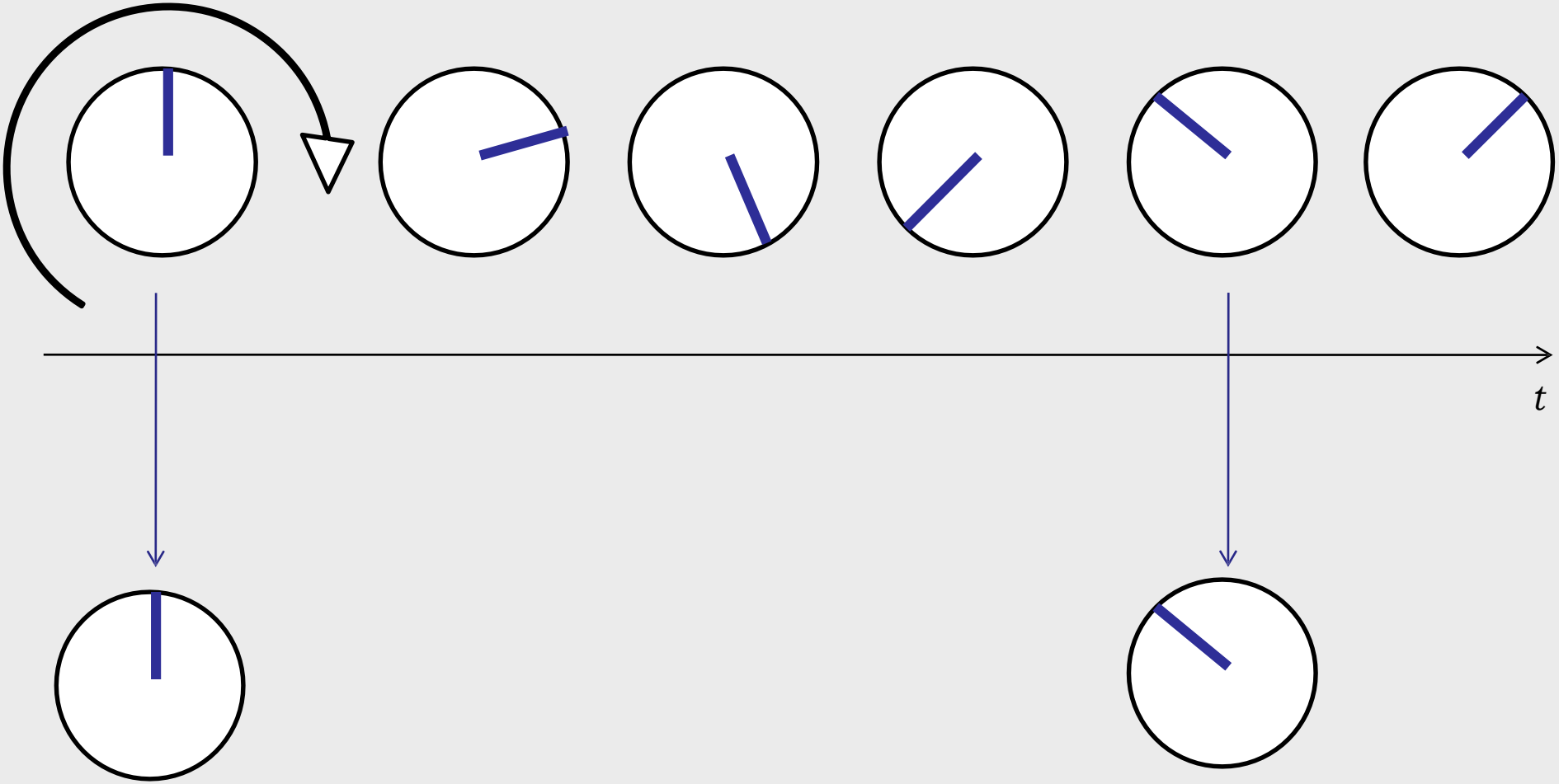


2. Comparsa di elementi NON PRESENTI nell'originale. **ALIASING**



ES.: ruota dell'auto
Periodo di campionamento dell'occhio = *100 ms* (10Hz)

Problemi legati al campionamento



Teorema del campionamento di Nyquist-Shannon

Dato un segnale, con larghezza di banda *B* finita e nota, la frequenza minima di campionamento, per poter ricostruire correttamente tale segnale, deve essere pari ad almeno *2B*.

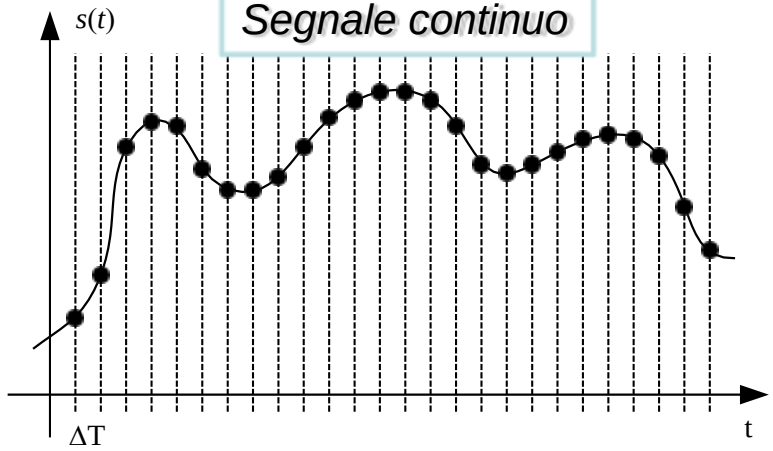
Un segnale campionato con frequenza pari a f_c avrà massima frequenza risolvibile pari a $f_c/2$

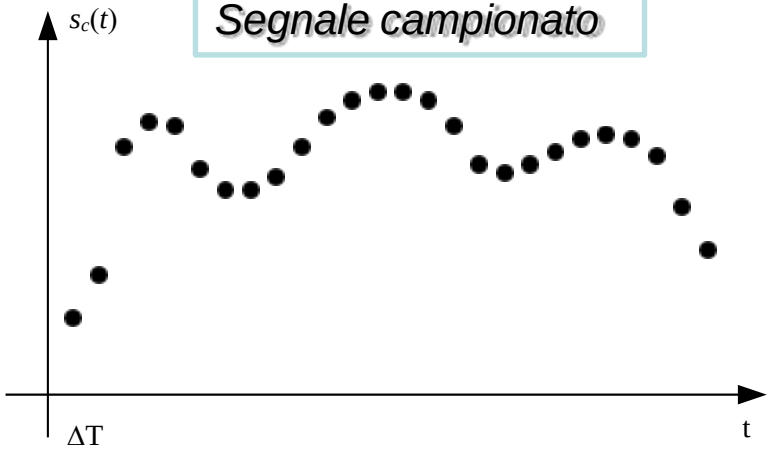
Es.: i CD audio impiegano una frequenza di campionamento di 44.100 Hz (formato wave), la massima frequenza risolvibile è 22.050 Hz, appena sopra il limite percepibile dall'udito umano di 20.000 Hz.

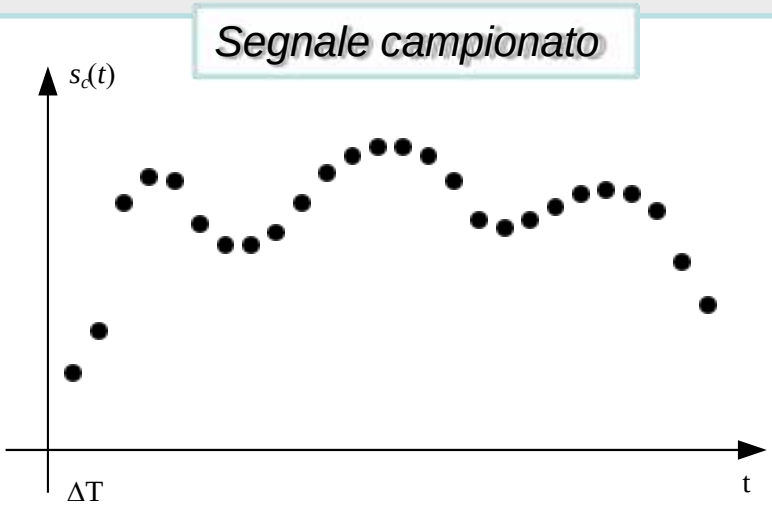
Speech Production: *up to 6-8 KHz*

Speech Perception: *up to 20KHz*

Rappresentazione digitale di segnali

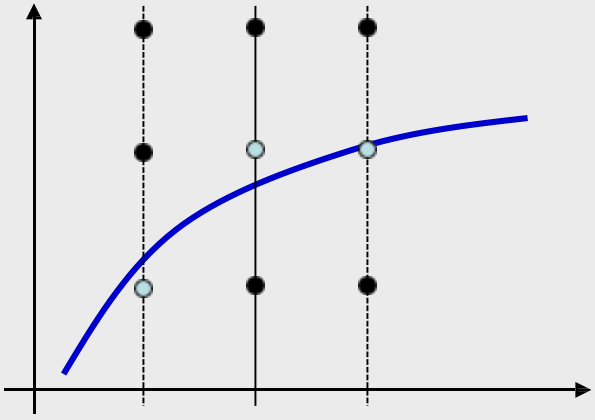


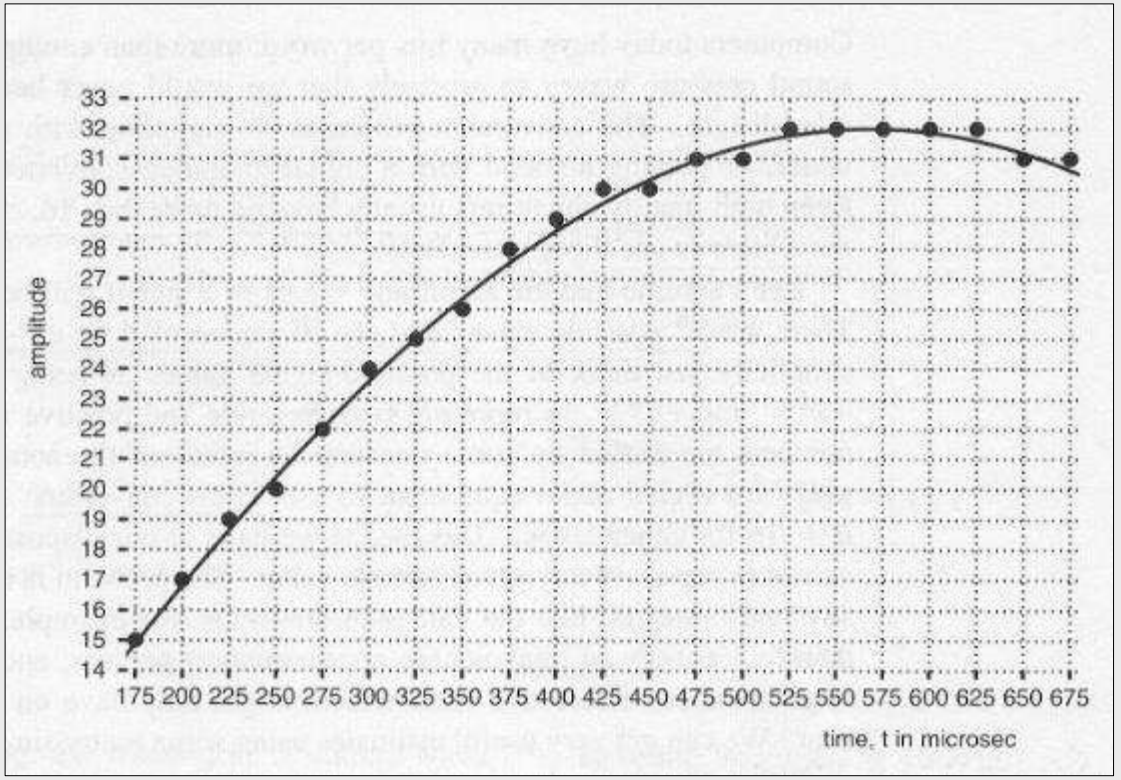




Quantizzazione

- La curva continua rappresenta la forma d'onda originaria, i punti rappresentano i campioni quantizzati al numero intero più vicino
- Il processo introduce delle imprecisioni dovute all'approssimazione del segnale reale (variabile in maniera continua) tramite un insieme finito di numeri.





Codifica dei dati: SISTEMA NUMERICO

Un sistema numerico viene determinato per mezzo di:

- Un insieme limitato di simboli (le cifre), che rappresentano quantità prestabilite

  Es.: sistema decimale: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
  sistema binario: 0, 1
  sistema romano: I, V, X, L, C, D, M, …

- Regole per costruire i numeri:
  1. Sistemi non posizionali, es. Romano, sistema additivo: le cifre sono sommate o sottratte sulla base dell'ordine in cui compaiono

     Es: XII = 10+1+1 = 12, IX=10-1=9

  2. Sistemi posizionali (es. decimale, binario, ecc.): il valore delle cifre dipende dalla loro posizione all'interno del numero

     BASE 10: $2562 = 2*10^3 + 5*10^2 + 6*10^1 + 2*10^0$

     BASE 2: $101 = 1*2^2 + 0*2^1 + 1*2^0$

     BASE 8: $72 = 7*8^1 + 2*8^0$

- Regole che consentano di eseguire operazioni tra i numeri

Sistemi in base B

- La base definisce il numero di cifre diverse del sistema di numerazione
- La cifra di minor valore (in termini di valore assoluto) è sempre lo 0, le altre sono, nell'ordine, 1, 2,…,B-1
- Se B>10 occorre introdurre B-10 simboli in aggiunta alle cifre decimali
  Es: sistema esadecimale : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Un numero **intero** N si rappresenta con la scrittura $(c_n c_{n-1} \ldots c_2 c_1 c_0)_B$

$$N = c_n B^n + c_{n-1} B^{n-1} + \ldots + c_2 B^2 + c_1 B^1 + c_0 B^0$$

c_n è la ***cifra più significativa***, c_0 quella *meno significativa*

Un numero **frazionario** N' si rappresenta con la scrittura $(0,c_1 c_2 \ldots c_n)_B$

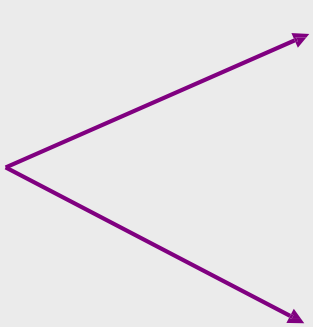
$$N' = c_1 B^{-1} + c_2 B^{-2} + \ldots + c_n B^{-n}$$

Numeri interi senza segno

Con *n* cifre, in base *B*, si rappresentano tutti i numeri interi positivi da *0* a B^n-1
(B^n numeri distinti)

Esempio: base 10

2 cifre: da 0 a $10^2-1 = 99$

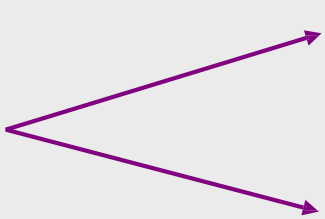


00
01
02
….
98
99

$10^2 = 100$ *valori (configurazioni)*

Esempio: base 2

2 cifre: da 0 a $2^2-1 = 3$



00
01
10
11

$2^2 = 4$ *valori*

Sistema in base 2 (sistema binario)

La base 2 è la più piccola per un sistema di numerazione

Cifre: 0 1 – bit (binary digit)

Esempi:

Forma polinomia

$(101101)_2 = 1\,2^5 + 0\,2^4 + 1\,2^3 + 1\,2^2 + 0\,2^1 + 1\,2^0 =$

$32 + 0 + 8 + 4 + 0 + 1 = (45)_{10}$

$(0,0101)_2 = 0\,2^{-1} + 1\,2^{-2} + 0\,2^{-3} + 1\,2^{-4} =$

$0 + 0,25 + 0 + 0,0625 = (0,3125)_{10}$

$(11,101)_2 = 1\,2^1 + 1\,2^0 + 1\,2^{-1} + 0\,2^{-2} + 1\,2^{-3} =$

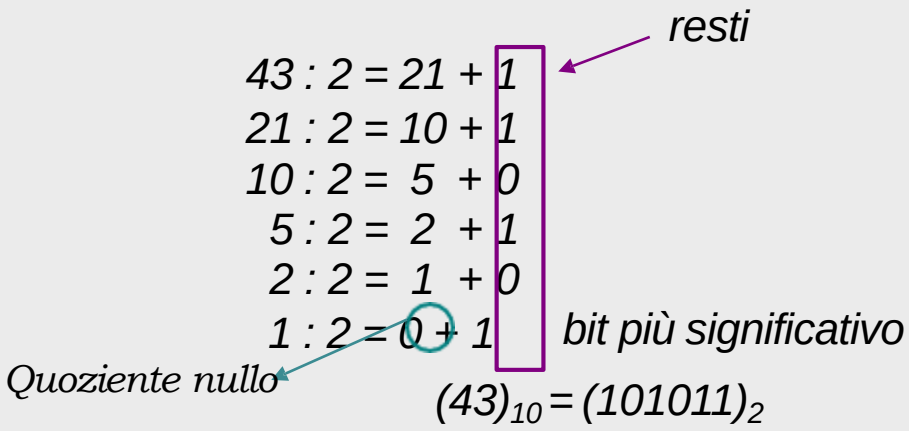
$2 + 1 + 0,5 + 0 + 0,125 = (3,625)_{10}$

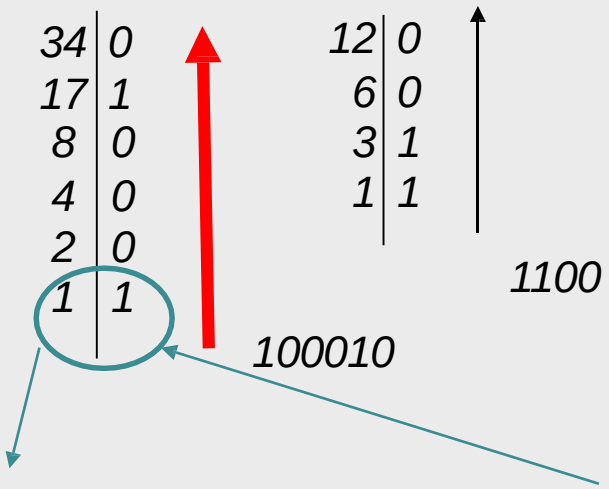
Conversioni di base: dec2bin

Si divide ripetutamente il numero **intero** decimale per 2 fino ad ottenere un quoziente nullo. Le cifre del numero binario sono i resti delle divisioni; la cifra più significativa è l'ultimo resto ottenuto

Esempi:

- Di quanti bit ho bisogno??...





Ottenuto il quoziente 1 sulla colonna di sx, lo si replica sulla colonna di dx

Esercizio

Si verifichi se le seguenti corrispondenze sono corrette:

$(110010)_2=(50)_{10}$ $(1110101)_2=(102)_{10}$ $(1111)_2=(17)_{10}$ $(10000000)_2=(128)_{10}$

$(11011)_2=(27)_{10}$ $(100001)_2=(39)_{10}$ $(11111111)_2=(255)_{10}$

Conversioni di base: dec2bin

Si moltiplica ripetutamente il numero **frazionario** decimale per 2, fino ad ottenere una parte decimale nulla o, dato che la condizione potrebbe non verificarsi mai, per un numero prefissato di volte. Le cifre del numero binario sono le parti intere dei prodotti successivi; la cifra più significativa è il risultato della prima moltiplicazione

Esempio: convertire in binario $(0.25)_{10}$, $(0.21875)_{10}$ *e* $(0.45)_{10}$

$.25 \times 2 = 0.50$
$.5 \times 2 = 1.0$

$(0.25)_{10} = (0.01)_2$

$.21875 \times 2 = 0.4375$
$.4375 \times 2 = 0.875$
$.875 \times 2 = 1.75$
$.75 \times 2 = 1.5$
$.5 \times 2 = 1.0$

$(0.21875)_{10} = (0.00111)_2$

$.45 \times 2 = 0.9$
$.90 \times 2 = 1.8$
$.80 \times 2 = 1.6$
$.60 \times 2 = 1.2$
$.20 \times 2 = 0.4$
$.40 \times 2 = 0.8$
$.80 \times 2 = 1.6$ etc.

$(0.45)_{10} \approx (0.0111001)_2$

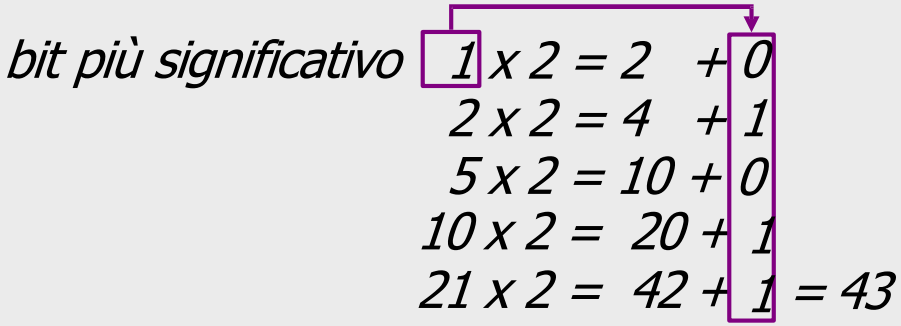
Conversioni di base: bin2dec

- Oltre all'espansione esplicita in potenze del 2 – forma polinomia…

$$(101011)_2 = 1\,2^5 + 0\,2^4 + 1\,2^3 + 0\,2^2 + 1\,2^1 + 1\,2^0 = (43)_{10}$$

- si può operare nel modo seguente:
  - si raddoppia il bit più significativo e si aggiunge al secondo bit;
  - si raddoppia la somma e si aggiunge al terzo bit…
  - .si continua fino al bit meno significativo

Esempio: convertire in decimale $(101011)_2$



$$(101011)_2 = (43)_{10}$$

Sistema esadecimale

Cifre: 0 1 2 3 4 5 6 7 8 9 A B C D E F

La corrispondenza in decimale delle cifre oltre il 9 è

$A = (10)_{10}$ $D = (13)_{10}$
$B = (11)_{10}$ $E = (14)_{10}$
$C = (12)_{10}$ $F = (15)_{10}$

Esempio:

$(3A2F)_{16} = 3\,16^3 + 10\,16^2 + 2\,16^1 + 15\,16^0 =$
$3\,4096 + 10\,256 + 2\,16 + 15 = (14895)_{10}$

- Una cifra esadecimale corrisponde a 4 bit

0 corrisponde a 4 bit a 0

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

F corrisponde a 4 bit a 1

- Si possono rappresentare numeri binari lunghi con poche cifre (1/4).
- BIN2HEX: raggruppando le cifre binarie in gruppi di 4 e sostituendole con le cifre esadecimali secondo la tabella seguente

Codifica dei dati: SISTEMA NUMERICO

Decimale	Binario	Ottale	Esadecimale
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Sistema esadecimale

HEX2BIN

- Un numero binario di 4n bit corrisponde a un numero esadecimale di n cifre

Esempio: 32 bit corrispondono a 8 cifre esadecimali

1101 1001 0001 1011 0100 0011 0111 1111
D 9 1 B 4 3 7 F

$(D91B437F)_{16}$

Esempio: 16 bit corrispondono a 4 cifre esadecimali

0000 0000 1111 1111
0 0 F F

$(00FF)_{16}$

Codifica dei dati: Numeri interi relativi

Per rappresentare numeri con segno in binario, occorre utilizzare 1 bit per definire il segno del numero

Si possono usare 3 tecniche di codifica:

- *Modulo e segno*
- *Complemento a 1 (poco usato)*
- *Complemento a 2*

- Rappresentazione **modulo e segno**

  n bit a disposizione per rappresentare un numero intero relativo X

  il primo bit (bit più significativo) viene usato per indicare il segno:

  Primo bit = 0 numero positivo

  Primo bit = 1 numero negativo

  Con n cifre si possono rappresentare i numeri: $-(2^{n-1}-1) \leq X \leq +(2^{n-1}-1)$

  "0" positivo e "0" negativo…

Codifica dei dati: Numeri interi relativi

- Il bit più significativo rappresenta il segno: 0 per i numeri positivi, 1 per quelli negativi
- Esiste uno zero positivo (00…0) e uno zero negativo (10…0)
- Se si utilizzano n bit si rappresentano tutti i numeri compresi fra $-(2^{n-1}-1)$ e $+2^{n-1}-1$

Esempio: con 4 bit si rappresentano i numeri fra –7 ($-(2^3-1)$) e +7 (2^3-1)

positivi		negativi	
0000	+0	1000	-0
0001	+1	1001	-1
0010	+2	1010	-2
0011	+3	1011	-3
0100	+4	1100	-4
0101	+5	1101	-5
0110	+6	1110	-6
0111	+7	1111	-7

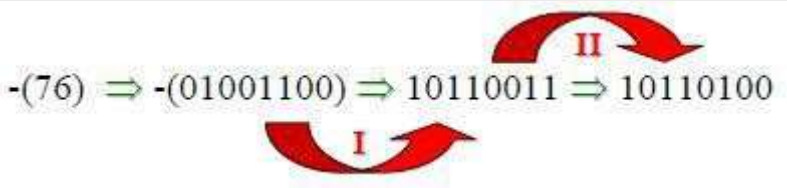
Codifica dei dati: Numeri interi relativi

- Rappresentazione in **complemento a 2**
  - Il complemento a 2 di un numero binario $(N)_2$ a n cifre è il numero $2^n - (N)_2$

  - il bit più a sinistra indica il segno,
  - un numero positivo è individuato dal primo bit uguale a 0 (rappr. modulo e segno)
  - per ottenere un numero negativo partendo dalla sua versione positiva, si procede in due passi:

    I. Si sostituiscono tutti gli uno con degli zero e viceversa (complemento a 1)

    II. Si aggiunge 1 al risultato (somma binaria simile a quella decimale, si riporta 1 se la somma è maggiore di 1) *(…vedi slide successiva)*



ES: 4= 0100
1) 1011
2) 1
1100 =-4

ES: 6= 0110
1) 1001
2) 1
1010 =-6

- Il campo dei numeri rappresentabili è da -2^{n-1} a $+2^{n-1}-1$

Nota: 0111 +7
1000 -8

Addizione Binaria

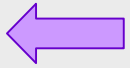
- Consideriamo le 4 possibilità di addizione tra due bit:

  0 + 0 = 0
  0 + 1 = 1
  1 + 0 = 1
  1 + 1 = 0 con riporto (carry) di 1

- Osservazione…. $(1)_2+(1)_2 = (10)_2$ … $(1+1=2)_{10}$!!

Esempio

```
 1 11 1      riporti
 01011011+
  01011010
 10110101
```



```
 91+
 90
 ---
181
```

Codifica dei dati: Numeri interi relativi

- I numeri positivi sono rappresentati in modulo e segno
- I numeri negativi hanno un 1 nella posizione più significativa e sono rappresentati in complemento a 2
- Lo zero è rappresentato come numero positivo (con una sequenza di n zeri)
- Il campo dei numeri rappresentabili è da -2^{n-1} a $+2^{n-1}-1$

Ad esempio: numeri a 4 cifre

0000	*+0*	*1000*	*-8*
0001	*+1*	*1001*	*-7*
0010	*+2*	*1010*	*-6*
0011	*+3*	*1011*	*-5*
0100	*+4*	*1100*	*-4*
0101	*+5*	*1101*	*-3*
0110	*+6*	*1110*	*-2*
0111	*+7*	*1111*	*-1*

Nota: 0111 +7
1000 -8

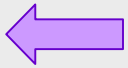
Sottrazione Binaria

- Consideriamo le 4 possibilità di sottrazione tra due bit:

  0 – 0 = 0
  1 – 0 = 1
  1 – 1 = 0
  0 – 1 = 1 con prestito (borrow) di 1 dalla cifra precedente a sinistra

Esempio

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



$$\begin{array}{r} 25 - \\ 5 \\ \hline 20 \end{array}$$

- Il calcolo della sottrazione può divenire complicato: quando si ha una richiesta sulla cifra precedente a sinistra, che è uno 0, l'operazione si propaga a sinistra fino alla prima cifra ad 1 del sottraendo

Utilizzando la rappresentazione in complemento a 2, addizione e sottrazione sono trattate come una unica operazione

Sottrazione Binaria

- Utilizzando la rappresentazione in complemento a 2, addizione e sottrazione sono trattate come una unica operazione

$N_1 - N_2 = N_1 + (2^n - N_2) - 2^n$ ← si trascura il bit n +1

$(2^n - N_2)$: complemento a 2 di N_2 ($-N_2$)

- Si calcola il complemento a 2 di N_2
- Si somma N_1 con il complemento a 2 di N_2
- Si trascura il bit più significativo del risultato

Esempio: $(010001)_2 - (000101)_2 = (17)_{10} - (5)_{10}$

```
  010001 +
  111011
 ------
1001100   →  (12)10
```

OVERFLOW

Si considerino due numeri binari ad n bit

Si ha **overflow** quando il risultato di un'operazione tra i due numeri non è rappresentabile correttamente con n bit (eccede il limite superiore o inferiore di rappresentabilità con gli n bit)

Esempio: 5 bit — [-16,+15], rappresentazione compl a 2

14 +	01110 +		-8 +	11000 +	
10	01010		-10	10110	
24	11000	→ -8	-18	101110	→ +14

- Per evitare l'overflow occorre aumentare il numero di bit utilizzati per rappresentare gli operandi (operazione non possibile se il calcolatore è già stato progettato)

- In caso di overflow la circuteria hardware lo rileva e segnala un errore (bit di overflow nella PSW del processore)

Numeri razionali
Notazione in virgola mobile

- Ogni numero può essere rappresentato come prodotto delle sue cifre significative per una potenza della base il cui valore dipende dalla posizione della virgola nel numero di partenza

  Es. $0.0012 = 12 \times 10^{-4}$

  $35.5 = 355 \times 10^{-1}$

- La IEEE ha previsto uno standard (IEEE 754) per la rappresentazione in virgola mobile
  - *singola precisione (32 bit = 4 byte)*
  - *doppia precisione (64 bit = 8 byte)*

Un qualsiasi numero può essere espresso come: $N = \pm M \times r^E$

Dove M = Mantissa – CIFRE RAPPRESENTATIVE DEL NUMERO
r = Base del sistema di numerazione
E = Esponente della base

Oss.: rappresentazione non univoca: $0.0012 = 0.12 \times 10^{-2} = 12 \times 10^{-4}$

NB: la mantissa contiene sempre solo le cifre significative del numero

Numeri razionali
Notazione in virgola mobile

Un qualsiasi numero può essere espresso come:

$$N = \pm M \times r^{E}$$

La base può essere omessa dato che nel calcolatore è sempre 2

Considerazioni sulla mantissa:

- M intero
- 1/r<=M<=1 MANTISSA NORMALIZZATA

Considerazioni sull'esponente:

- Numero intero relativo in complemento a 2

Numeri razionali
Notazione in virgola mobile

Un qualsiasi numero può essere espresso come:

$$N = \pm M \times r^{E}$$

Esempio

10111010

- Segno=1 -> numero negativo
- Esponente=011 in complemento a 2 = +3
- Mantissa normalizzata=0.1010 = $2^{-1}+2^{-3}$ =1/2+1/8=5/8

N=-5/8x 2^{3} =-5

Esercizio:

- Determinare il più grande numero rappresentabile
- Determinare il più piccolo numero (vicino allo zero) rappresentabile

Numeri razionali
Notazione in virgola mobile

(+7) in base 10
•numero positivo -> Segno=0
•7 mantissa su 4 bit = $0111x2^0 = 0.111x2^3$, mantissa=1110
•Esponente=011 in complemento a 2 = +3
N=0 011 1110

Numero più piccolo (più vicino allo 0)
•numero positivo -> Segno=0
•Esponente = -4 -> 100
•Mantissa = 0.0001
N=0 100 0001

Secondo numero più piccolo
•numero positivo -> Segno=0
•Esponente = -4 -> 100
•Mantissa = 0.0010
N=0 100 0010

01000001
- Segno=0 -> numero positivo
- Esponente = -4
- Mantissa normalizzata=0.0001 = 2^{-4} =1/16

N=+1/16x 2^{-4} = + 1/256

01000010
- Segno=0 -> numero positivo
- Esponente = -4
- Mantissa normalizzata=0.0010 = 2^{-3} =1/8

N=+1/8x 2^{-4} = + 1/128

Distanza=1/128=0.0078125

Numeri razionali
Notazione in virgola mobile

Numero più grande positivo
•numero positivo -> Segno=0
•Esponente = +3 -> 011
•Mantissa = 1111
N=0 011 1111

00111111
- Segno=0 -> numero positivo
- Esponente=011 in complemento a 2 = +3
- Mantissa normalizzata=1111 = $2^{-1}+2^{-2}+2^{-3}+2^{-4}$ =15/16

N=+15/16x 2^3 = 7,5 (si provi a rappresentare 8!!)

Secondo numero più grande positivo
•numero positivo -> Segno=0
•Esponente = +3 -> 011
•**Mantissa = 1110**
N=0 011 1110

00111110
- Segno=0 -> numero positivo
- Esponente=011 in complemento a 2 = +3
- Mantissa normalizzata=1110 = $2^{-1}+2^{-2}+2^{-3}$ =7/8

N=+7/8x 2^3 = 7 (si provi a rappresentare 7.25!!)

Distanza=0.5

Numeri razionali
Notazione in virgola mobile

(-5.5) in base 10

•numero negativo -> Segno=1

•5.5 mantissa su 4 bit = $101.1x2^0 = 0.1011x2^3$, mantissa=1011

$.5 \times 2 = 1.0$

•Esponente=011 in complemento a 2 = +3

N=1 011 1011

(-5.75) in base 10

- numero negativo -> Segno=1
- 5.5 = $101.11x2^0 = 0.10111x2^3$, mantissa=1011 (mantissa su 4 bit!)
- Esponente=011 in complemento a 2 = +3

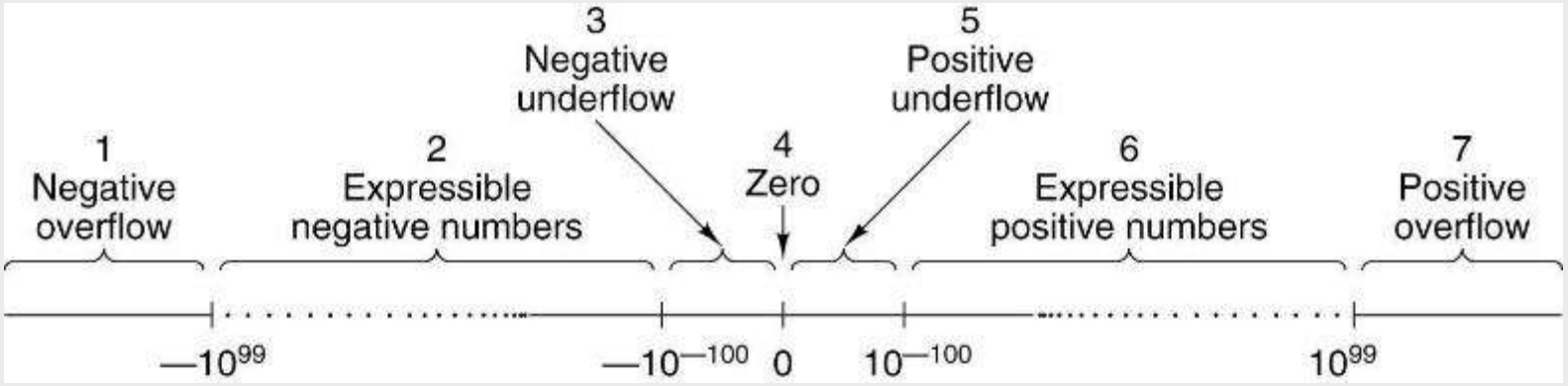
N=1 011 1011 = -5.5!!! **ERRORE DI APPROSSIMAZIONE**

$.75 \times 2 = 1.50$

$.5 \times 2 = 1.0$

Numeri razionali
Notazione in virgola mobile

Characteristics of IEEE floating-point numbers.



Il valore è

$(-1)^S$ 1.M 2^{E-127} se E≠0
$(-1)^S$ 0.M 2^{-127} se E=0

Normalized	±	0 < Exp < Max	Any bit pattern
Denormalized	±	0	Any nonzero bit pattern
Zero	±	0	0
Infinity	±	1 1 1…1	0
Not a number	±	1 1 1…1	Any nonzero bit pattern

Sign bit

Numeri razionali
Notazione in virgola mobile

Characteristics of IEEE floating-point numbers.

Item	Single precision	Double precision
Bits in sign	1	1
Bits in exponent	8	11
Bits in fraction	23	52
Bits, total	32	64
Exponent system	Excess 127	Excess 1023
Exponent range	−126 to +127	−1022 to +1023
Smallest normalized number	2^{-126}	2^{-1022}
Largest normalized number	approx. 2^{128}	approx. 2^{1024}
Decimal range	approx. 10^{-38} to 10^{38}	approx. 10^{-308} to 10^{308}
Smallest denormalized number	approx. 10^{-45}	approx. 10^{-324}

Aritmetica degli elaboratori

- Rango finito dei numeri rappresentabili
  - Qualunque sia la codifica, esistono sempre il più grande ed il più piccolo numero
  - I limiti inferiore e superiore dipendono sia dal tipo di codifica, sia dal numero di bit utilizzati
  - Se il risultato di un'operazione non appartiene al rango dei numeri rappresentabili, si dice che si è verificato un overflow (un underflow, più precisamente, se il risultato è più piccolo del più piccolo numero rappresentabile)

- Precisione finita dei numeri
  - La precisione della rappresentazione di un numero frazionario è una misura di quanto essa corrisponda al numero che deve essere rappresentato
  - Negli elaboratori, i numeri frazionari sono rappresentati in virgola mobile (floating point), utilizzando un numero finito di bit
  - È plausibile che un numero reale non ammetta una rappresentazione finita,
  - Negli elaboratori si rappresentano soltanto numeri razionali (fino ad una data precisione)

Codifica dei dati: Rappresentazione dei caratteri

Ciascun carattere alfanumerico, di punteggiatura o di controllo che compone il testo deve essere rappresentato nei termini di codice binario

CODICE ASCII *(American Standard Code for Information Interchange) o* **ISO-646**
utilizza 8bit di cui uno di controllo (di parità): 7bit=128 configurazioni

i caratteri sono distinti in:

- **caratteri di comando (codici di trasmissione o di controllo della stampa)**
  Line Feed (LF) 00001010 Escape (Esc) 00011011

- **caratteri alfanumerici: lettere dell'alfabeto maiuscole e minuscole e cifre numeriche**
  A 01000001 B 01000010 C 01000011
  a 01100001 b 01100010 c 01100011

- **Segni: simboli di punteggiatura e operatori aritmetico-logici**
  ; 00111011 " 00100010 [01011011
  * 00101010 / 00101111 > 00111110

Codifica dei dati: Rappresentazione dei caratteri

CODICE ASCII

NUL	00000000	blank	00100000	@	01000000	‘	01100000
SOH	00000001	¡	00100001	A	01000001	a	01100001
STX	00000010	"	00100010	B	01000010	b	01100010
ETX	00000011	#	00100011	C	01000011	c	01100011
EOT	00000100	$	00100100	D	01000100	d	01100100
ENQ	00000101	%	00100101	E	01000101	e	01100101
ACK	00000110	&	00100110	F	01000110	f	01100110
BEL	00000111	'	00100111	G	01000111	g	01100111
BS	00001000	(	00101000	H	01001000	h	01101000
HT	00001001	)	00101001	I	01001001	i	01101001
LF	00001010	*	00101010	J	01001010	l	01101010
VT	00001011	+	00101011	K	01001011	j	01101011
FF	00001100	,	00101100	L	01001100	k	01101100
CR	00001101	-	00101101	M	01001101	m	01101101
SO	00001110	.	00101110	N	01001110	n	01101110
SI	00001111	/	00101111	O	01001111	o	01101111
DLE	00010000	0	00110000	P	01010000	p	01110000
DC1	00010001	1	00110001	Q	01010001	q	01110001
DC2	00010010	2	00110010	R	01010010	r	01110010
DC3	00010011	3	00110011	S	01010011	s	01110011
DC4	00010100	4	00110100	T	01010100	t	01110100
NAK	00010101	5	00110101	U	01010101	u	01110101
SYN	00010110	6	00110110	V	01010110	v	01110110
ETB	00010111	7	00110111	W	01010111	w	01110111
CAN	00011000	8	00111000	X	01011000	x	01111000
EM	00011001	9	00111001	Y	01011001	y	01111001
SUB	00011010	:	00111010	Z	01011010	z	01111010
ESC	00011011	;	00111011	[	01011011	{	01111011
FS	00011100	<	00111100	\	01011100	¡	01111100
GS	00011101	=	00111101	]	01011101	}	01111101
RS	00011110	>	00111110	←	01011110	≈	01111110
US	00011111	?	00111111	↑	01011111	DEL	01111111

01000011 01101001 01100001 01101111
C i a o

Codifica dei dati: Rappresentazione dei caratteri

ASCII sviluppato per lingue anglosassoni, quindi non contiene codici per molti caratteri di altre lingue

UNICODE:sistema di codifica che assegna un numero univoco ad ogni carattere usato per la scrittura di testi, in maniera indipendente dalla lingua, dalla piattaforma informatica e dal programma utilizzato.

- Codice a 16 bit che può essere usato per codificare 2^{16} simboli diversi (codificando i caratteri usati in quasi tutte le lingue vive e in alcune lingue morte, nonché simboli matematici e chimici, cartografici, l'alfabeto Braille, ecc.)
- PRO:
  - Universalità
- Problemi:
  - Non tutti gli editori lo trattano
  - Dimensioni dei file raddoppiano rispetto all'ASCII

L'ultima versione dello standard definisce 3 tipi diversi di codifica che permettono agli stessi dati di essere trasmessi in byte (8 bit – UTF-8), word (16 bit – UTF-16) o double word (32 bit – UTF-32).

Rappresentazione digitale dei suoni

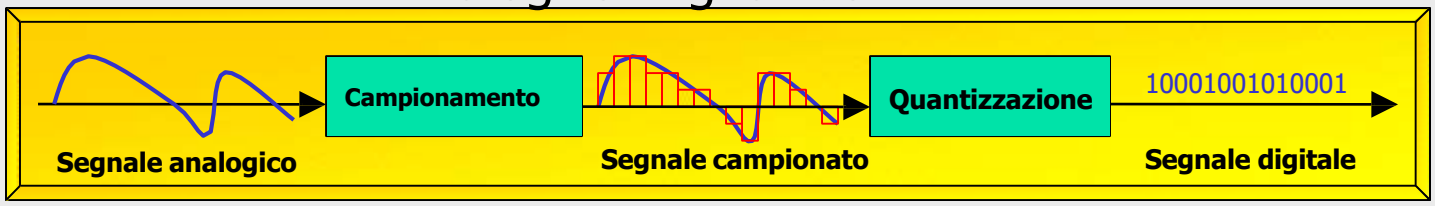
CAMPIONAMENTO:

- processo di conversione di un segnale tempo-continuo in un segnale tempo-discreto,
- L'ampiezza del segnale continuo viene considerata a intervalli di tempo regolari (T - periodo di campionamento).

QUANTIZZAZIONE:

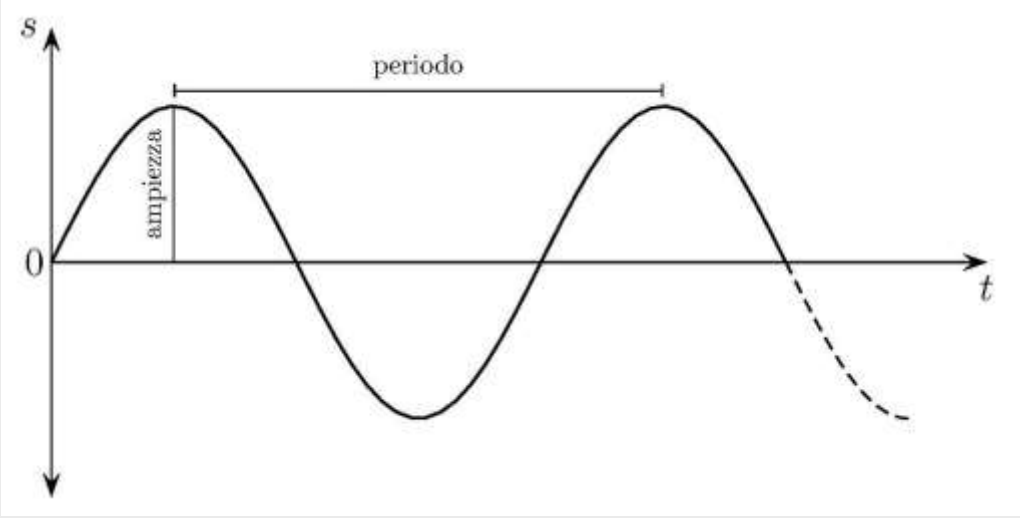
- processo di conversione di un segnale a valori continui in uno a valori discreti.
- Più è alto il numero di bit utilizzati nella quantizzazione e minore è l'errore che si commette (errore di quantizzazione), cioè si riduce la distanza media tra il valore campionato (continuo) e il corrispondente valore quantizzato

Analog to **D**igital **C**onverter



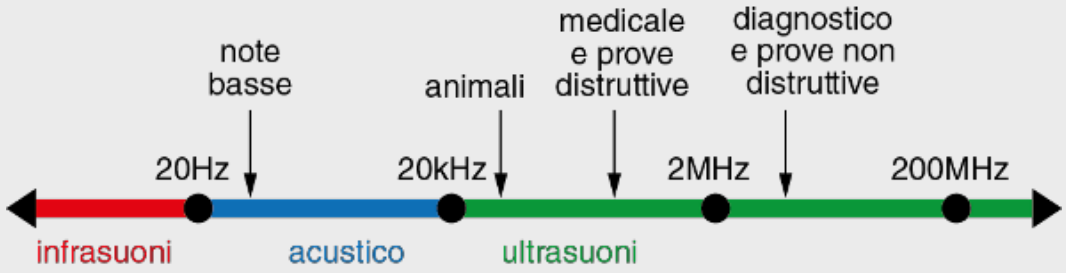
Codifica audio (1/2)

- Il **suono** è definito come una rapida variazione di pressione, prodotta da una sorgente, in un mezzo elastico → tale variazione produce oscillazioni nelle particelle del mezzo
- Le oscillazioni sono spostamenti delle particelle intorno alla posizione di riposo e nella direzione di propagazione del suono
- Tali oscillazioni sono rappresentate da **onde sonore**



Codifica audio (2/2)

- La soglia dell'udibile è fra i 20 e i 20 kHz



- Formato WAVE (WAVEform audio file format):
  - Per il Teorema di Nyquist-Shannon, la frequenza di campionamento è 44100 Hz (≈ 20000 x 2)
  - Campioni per 2 canali (sinistro/destro)
  - 16 bit per campione per canale
  - 44100 x 16 x 2 = 1 411 200 bit per secondo (≈ 0,17 MB)
  - Quanto spazio occupa una canzone di 3 minuti?

Algoritmi di compressione per file audio

Codifica della musica:

MP3 (compressione lossy): Sfrutta conoscenza dei limiti dell'udito umano per ridurre la quantita' di informazione da memorizzare:

- Esclusi suoni che l'orecchio percepisce poco (un suono ad una frequenza viene percepito meglio di suoni a frequenze adiacenti a più basso volume)
- Quando c'e' un suono particolarmente rumoroso, non registrare gli altri suoni
- Fattore di riduzione: anche 10 volte (= 3-4M per canzone)

Codifica della voce:

Pulse Code Modulation (PCM)

- Trasmissioni telefoniche
- Frequenza campionamento: 8KHz
- Quantizzazione su 8 bit
- Velocità sul canale trasmissivo >= 64Kbps

Codifica delle immagini

- Le operazioni di campionamento e quantizzazione si applicano nello spazio (2D) invece che nel tempo.

- Il campionamento consiste nel dividere l'immagine in sottoinsiemi regolari (pixel = picture element), per ognuno dei quali si dovrà prelevare un campione che si considera rappresentativo di tutto il sottoinsieme.

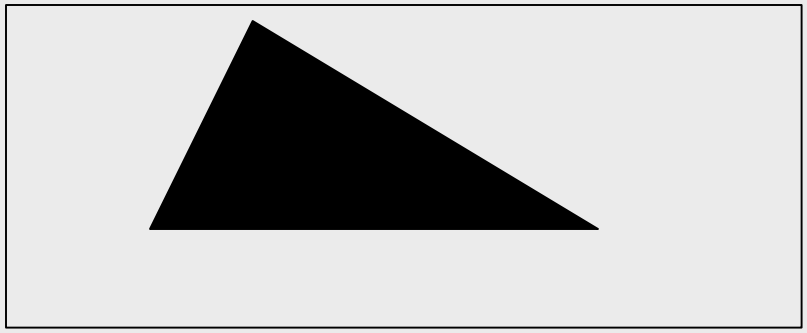
- La quantizzazione è la codifica del colore associato a ogni pixel: i più recenti formati utilizzano 32 bit (4 byte) per pixel di cui:
  - 8 bit per ognuna delle tre componenti fondamentali (RGB: red, green, blue)
  - altri 8 per gestire le trasparenze.
  - L'immagine è una mappa di bit (bitmap .bmp)

    Esempio di dimensioni:

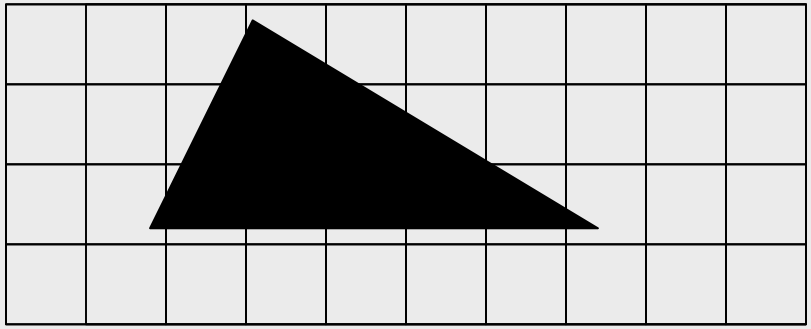
    Macchina fotografica a 10Mpixel: 3872 x 2592 pixel

    3872 x 2592 x4B=38,29MB

Codifica delle immagini

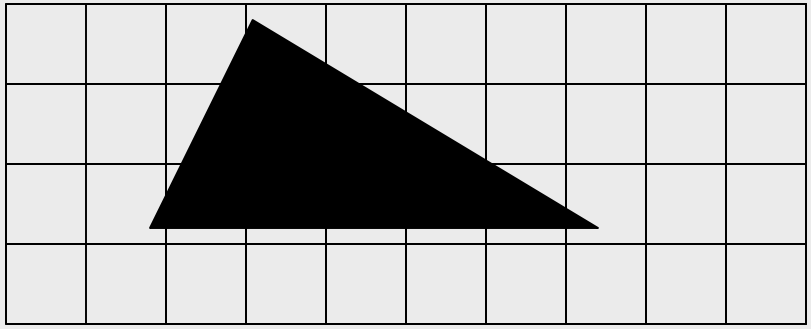


Consideriamo una griglia formata da righe orizzontali e verticali a distanza costante



- Ogni quadratino prende il nome di ***pixel*** (picture element) e può essere codificato in binario secondo la seguente convenzione:
  - “0” viene utilizzato per la codifica di un pixel corrispondente ad un quadratino in cui il bianco è predominante
  - “1” viene utilizzato per la codifica di un pixel corrispondente ad un quadratino in cui il nero è predominante

Codifica delle immagini



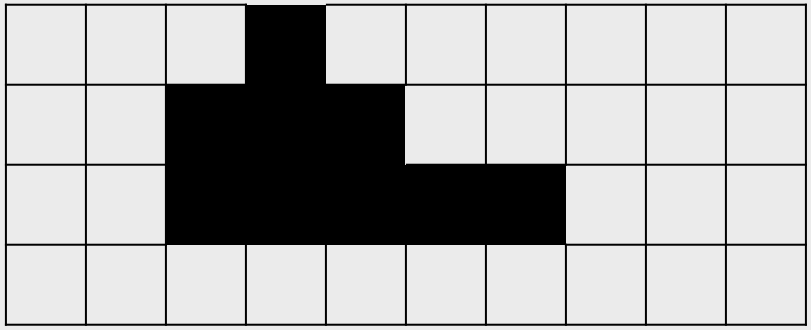
0	0	0	1	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0
0	0	1	1	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0

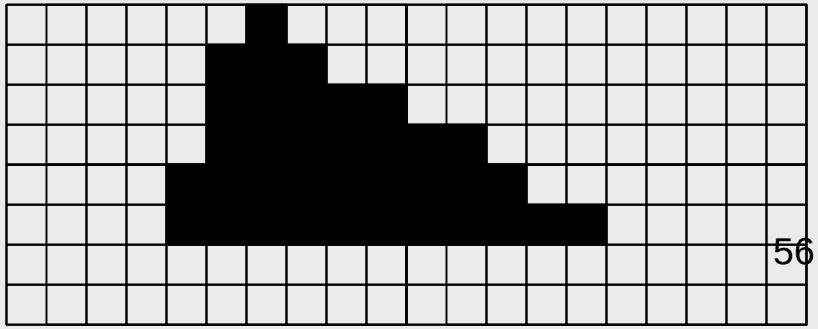
Codifica delle immagini

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

Poiché una sequenza di bit è lineare, è necessario definire convenzioni per ordinare la griglia dei pixel in una sequenza.
Non sempre il cortorno della figura coincide con le linee della griglia. Quella che si ottiene nella codifica è un'approssimazione della figura originaria

La rappresentazione sarà più fedele all'aumentare del numero di pixel, ossia al diminuire delle dimensioni dei quadratini della griglia in cui è suddivisa l'immagine





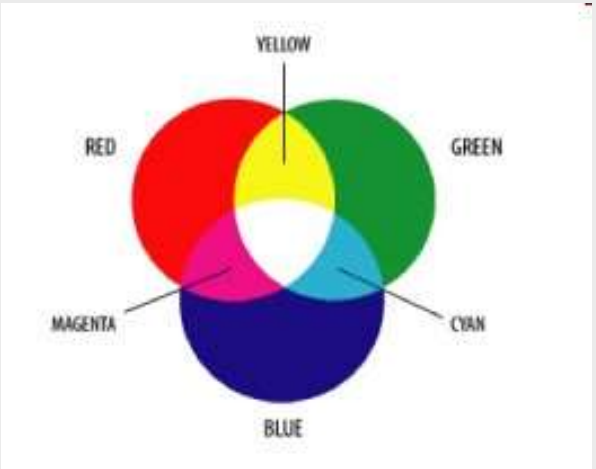
La risoluzione

- dpi= dot per inch
  - Numero di pixel presenti su una linea lunga 2,54 cm

Spazio dei colori

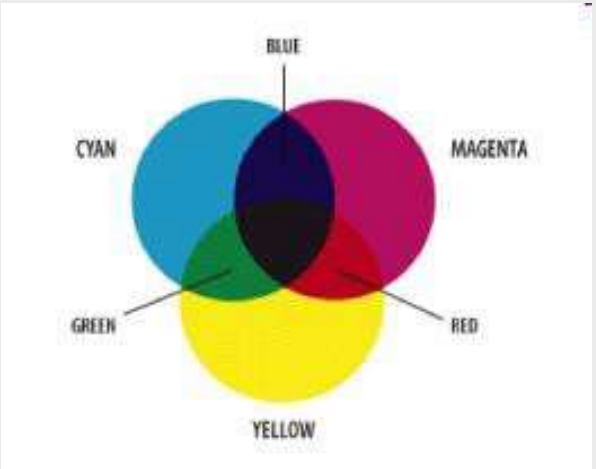
RGB

Sistema additivo: i colori ROSSO, VERDE e BLU (RGB) si mescolano tra loro creando colori di luminosità maggiore. Viene utilizzato quando si vogliono mischiare fasci di luce colorata, ad esempio sugli schermi e sui monitor.



CMY

Sistema sottrattivo: CIANO, MAGENTA e GIALLO (CMY). I colori che vengono generati hanno meno luminosità. Questo sistema si usa nel caso in cui si mischiano sostanze colorate (che riflettono la luce): inchiostri, tempere etc.



Rappresentazione dei colori

- Se si fa corrispondere a ogni pixel un byte, potremo differenziare 256 colori
- Necessità di una tabella di codifica dei colori…

  Ad es: 00101101 →

- L'immagine viene fatta corrispondere a una matrice
- Ogni pixel dell'immagine viene codificato dalla sequenza di '0' e '1' associato al suo colore dalla tabella di codifica dei colori utilizzata

La **dimensione** dell'immagine è data dal numero di pixel che costituiscono la base per il numero di pixel che costituiscono l'altezza.

La **profondità** dell'immagine è il numero di bit che utiliziamo per rappresentare il colore di un singolo pixel dell'immagine (numero di colori disponibili)

Numero di bit per immagine = dimensione x profondità

Algoritmi di compressione delle immagini

Formato BITMAP (.bmp)

Esempio di dimensioni:
Macchina fotografica a 10Mpixel: 3872 x 2592 pixel
3872 x 2592 x4B=38,29MB

Per l'occhio umano lievi cambiamenti di colore in punti contigui sono poco visibili e quindi possono essere eliminati (su questo principio si basano, per esempio, i formati GIF e JPEG);

Formato GIF: riduce l'occupazione di un'immagine **limitando il numero di colori** che compaiono in essa -> vengono scelti quelli più frequenti, alcune sfumature vengono perse e sostituite dalle sfumature più vicine fra quelle mantenute
Più si limita il numero di colori più l'immagine occuperà meno spazio; il numero può andare da un minimo di 2 ad un massimo di 256.
Adatto ad immagini geometriche.

Formato JPEG riduce l'occupazione di un'immagine diminuendo la qualità di visualizzazione -> consente di usare tutta la gamma RGB.

1. Si sfrutta la diversa sensibilità dell'occhio a diversi livelli di luminosità
2. In aree 8x8 pixel si individuano solo punti significativi per la percezione umana

Adatto ad immagini di tipo fotografico o in generale ad immagini dove un limite sul numero di colori produrrebbe differenze troppo significative

Informazione Multimediale

- **Filmato:** successione di fotogrammi (frame) accompagnata da una flusso audio

  Il movimento è rappresentato in modo discreto: con un numero sufficientemente alto di fotogrammi fissi (24-30 al secondo) l'occhio umano percepisce il movimento come un continuo (l'occhio umano "scatta" 10 foto al secondo)

  frame per second fps

- Come codificare un video?

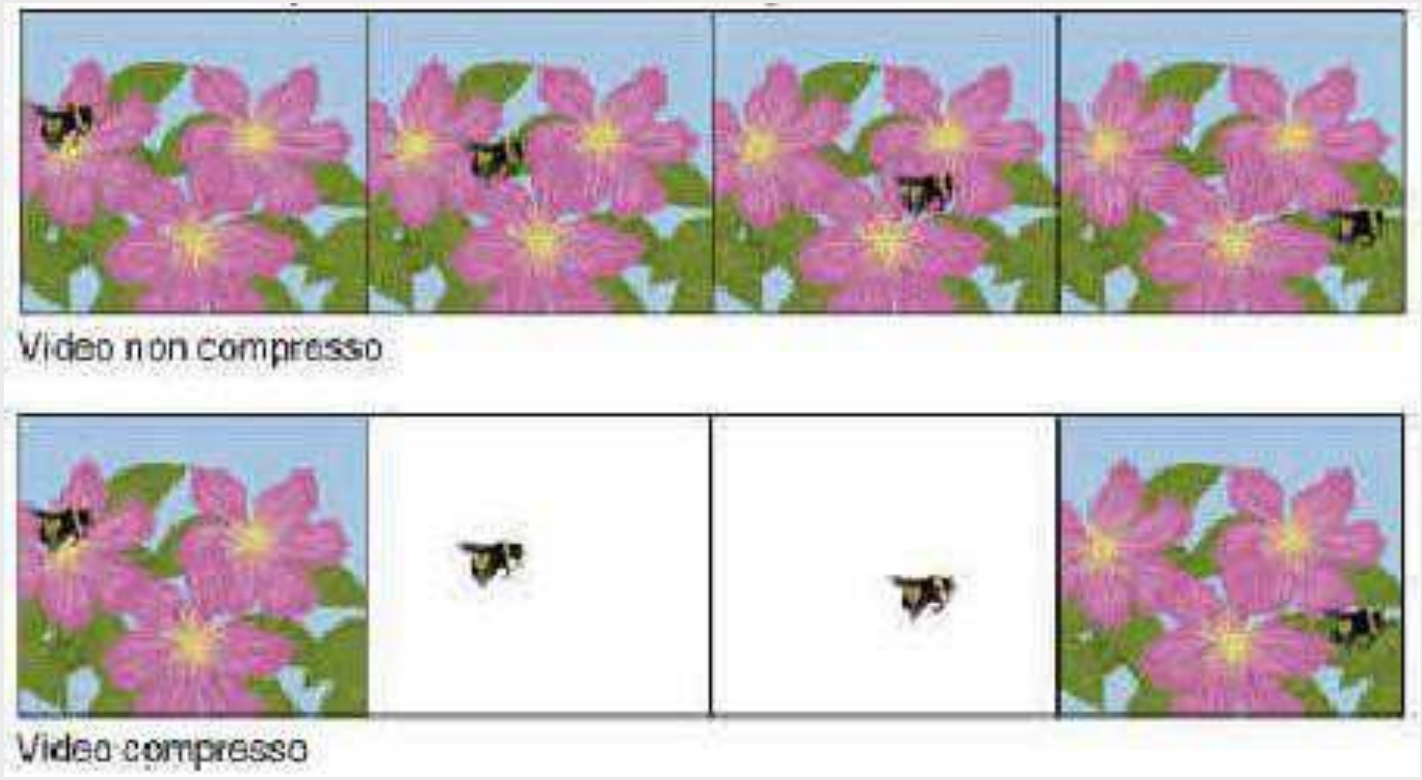
  IDEA..
  - Ogni fotogramma è un'immagine: codifica delle immagini
  - Audio: codifica dell'audio.

  PROBLEMA: per codificare un breve filmato servono moltissimi bit!
  - Es: codificando separatamente ogni fotogramma come immagine fissa, lo spazio di memoria richiesto sarebbe enorme (650 MB per un minuto di flusso video)

- SOLUZIONE: codificare solo delle 'differenze' fra un fotogramma e l'altro (MPEG)

Algoritmi di compressione video

MPEG: immagini successive hanno parti uguali e quindi di ogni immagine possono essere memorizzate solo le differenze con l'immagine precedente (come fanno, per esempio, vari standard MPEG)



Informazione Multimediale

La dimensione del filmato dipende da almeno cinque fattori:

- **Lunghezza del filmato** (in termini di [s])

- **Risoluzione grafica** (quanto più fitta è la griglia che usiamo per digitalizzare i singoli fotogrammi)
  - Una bassa risoluzione rende il filmato “quadrettato e indistinto”

- **L’ampiezza della ‘palette’** di colori utilizzata (ossia il numero dei colori)
  - Una palette troppo ristretta rende i colori poco realistici

- Il **numero di fotogrammi** (o frame) per secondo
  - Un numero limitato di fotogrammi fa apparire la sequenza di immagini poco fluida

- La **qualità dell’audio**

Algoritmi di compressione video

Quicktime (Apple)

Avi (Microsoft)

Real Video: usato in applicazioni di streaming video per la fruizione in tempo reale di un video

- i pacchetti audio-video che rappresentano il filmato vengono trasmessi continuamente l'uno dopo l'altro e vengono visualizzati man mano che arrivano nell'ordine;
- poi vengono buttati via

streaming video vs. classico download