Software

L'hardware da solo non è sufficiente per il funzionamento dell'elaboratore ma è necessario introdurre il software

- Una programmazione diretta della macchina hardware da parte degli utenti è davvero difficile l'utente dovrebbe conoscere l'organizzazione fisica dell'elaboratore e il suo linguaggio macchina.
- Ogni programma dovrebbe essere scritto utilizzando delle sequenze di bit ed ogni piccola differenza hardware comporterebbe una riscrittura del programma stesso

Software

Il software può essere pensato come un insieme di istruzioni che ci lascia interagire con il computer.
Ci sono tipi diversi di software, ognuno con una funzione diversa da assolvere:

- Sistema Operativo e Software di Base: mette il computer in grado di lavorare
- Software Applicativo: consente all'utente di portare avanti compiti specifici (ad es. programmi per video-scrittura, gestione database, etc)
- Software di Utilità: viene usato per l'ottimizzazione e la manutenzione del computer (ad es. anti-virus)

Sistema Operativo

I compiti del sistema operativo sono principalmente i seguenti:

- Avvio dell'elaboratore
- Gestione dei servizi
- Gestione dei dispositivi
- Interazione con l'utente (Interfaccia)

Esempi di sistemi operativi sono i popolari Microsoft Windows, Mac OS, Linux.

Rappresentazione binaria 1

There are only 10 types of people in the world: those who understand binary and those who don't.

Rappresentazione binaria 2

All'interno di un calcolatore, tutte le informazioni sono rappresentabili in forma binaria.

- Per informazione intendiamo:
  - numeri (naturali, interi, reali, ...)
  - caratteri
  - immagini
  - suoni
  - programmi
  - ...
- La più piccola unità di informazione memorizzabile o elaborabile da un calcolatore, il bit (binary digit), corrisponde allo stato di un dispositivo fisico che viene interpretato come 1 o 0.
- In un calcolatore tutte le informazioni sono rappresentate in forma binaria, come sequenze di 0 e 1.
- Per motivi tecnologici, ovvero di affidabilità: distinguere tra due valori di una grandezza fisica è più semplice che non ad esempio tra dieci valori. I dispositivi facilmente presentano due stati stabili.

Rappresentazione di numeri naturali

- Un numero naturale è un oggetto matematico, che può essere rappresentato mediante una sequenza di simboli di un alfabeto fissato.
- È importante distinguere tra numero e sua rappresentazione: il numerale "234" è la rappresentazione del numero 234.
- Si distinguono due tipi di rappresentazione:

  additiva: ad es. le cifre romane

  posizionale: una cifra contribuisce con un valore diverso al numero a seconda della posizione in cui si trova

Sistema di numerazione romano

I	V	X	L	C	D	M
1	5	10	50	100	500	1000

Ad esempio 2014 si scrive come MMXIV

- $XVII + VI = X\underline{VV}\ III = XXII$ $(17 + 6 = 23)$ addizione
- $X\underline{VI}I - VI = XI$ $(17 - 6 = 11)$ sottrazione
- $XVIIxII = XX\underline{VV}\ \underline{IIII} = XXXIV$ $(17 \times 2 = 34)$ moltiplicazione
- $XXXIV : II = XXX\ \underline{IIII} : 2 = XVII$ $(34 : 2 = 17)$ divisione

Rappresentazione posizionale

- Un numero è rappresentato da una sequenza finita di cifre di un certo alfabeto:

$$c_{n-1}c_{n-2}\cdots c_1 c_0 \ = \ N_b$$

  c_0 viene detta cifra meno significativa

  c_{n-1} viene detta cifra più significativa
- Il numero b di cifre diverse (dimensione dell'alfabeto) è detto base del sistema di numerazione.
- Ad ogni cifra è associato un valore compreso tra 0 e $b-1$.

Base	Alfabeto	Sistema
2	0, 1	binario
8	0, . . . , 7	ottale
10	0, . . . , 9	decimale
16	0, . . . , 9, A, . . . , F	esadecimale

- Il significato di una sequenza di cifre (il numero N che essa rappresenta) dipende dalla base b:

$$c_{n-1} \cdot b^{n-1} + c_{n-2} \cdot b^{n-2} + \cdots + c_1 \cdot b^1 + c_0 \cdot b^0 \quad = \quad \sum_{i=0}^{n-1} c_i \cdot b^i \quad = \quad N$$

Esempio: Il numerale 101 rappresenta numeri diversi a seconda del sistema usato:

Sistema	Base b	101_b	Valore$_{10}$
decimale	10	$(101)_{10}$	$101 = 1 \cdot 10^2 + 0 \cdot 10^1 + 1 \cdot 10^0$
binario	2	$(101)_2$	$5 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$
ottale	8	$(101)_8$	$65 = 1 \cdot 8^2 + 0 \cdot 8^1 + 1 \cdot 8^0$
esadecimale	16	$(101)_{16}$	$257 = 1 \cdot 16^2 + 0 \cdot 16^1 + 1 \cdot 16^0$

Intervallo di rappresentazione

- I numeri rappresentabili in base b con n posizioni (cifre) vanno da 0 a $b^n - 1$.

3 cifre in base 10 :	da	0	a	$999 = 10^3 - 1$
8 cifre in base 2 :	da	0	a	$255 = 2^8 - 1$
16 cifre in base 2 :	da	0	a	$65\,535 = 2^{16} - 1$
32 cifre in base 2 :	da	0	a	$4\,294\,967\,296 = 2^{32} - 1$
2 cifre in base 16 :	da	0	a	$255 = 16^2 - 1$
8 cifre in base 16 :	da	0	a	$4\,294\,967\,296 = 16^8 - 1$

Conversioni di base: da base b a base 10

- Usando direttamente

$$c_{n-1} \cdot b^{n-1} + c_{n-2} \cdot b^{n-2} + \cdots + c_1 \cdot b^1 + c_0 \cdot b^0 \quad = \quad \sum_{i=0}^{n-1} c_i \cdot b^i \quad = \quad N$$

esprimendo le cifre e b in base 10 (e facendo i conti in base 10)

Esercizio
(domani) Scrivere l'algoritmo di conversione da base b a base 10.

Conversioni di base: da base 10 a base b

$$\begin{aligned} N &= c_0 + c_1 \cdot b^1 + c_2 \cdot b^2 + \cdots + c_{k-1} \cdot b^{k-1} \\ &= c_0 + b \cdot (c_1 + b \cdot (c_2 + \cdots + b \cdot c_{k-1}) \cdots) \end{aligned}$$

- Vogliamo determinare le cifre $c_0, c_1, \ldots, c_{k-1}$
- Consideriamo la divisione di N per b:

$$\begin{aligned} N &= R + b \cdot Q \qquad\qquad (0 \leq R < b) \\ &= c_0 + b \cdot (c_1 + b \cdot (\cdots)) \end{aligned}$$

$$\Downarrow$$

$R = c_0$ ovvero, il resto R della divisione di N per b dà c_0 (cifra meno significativa)

$$Q = c_1 + b \cdot (\cdots)$$

- A partire dal quoziente Q si può iterare il procedimento per ottenere le cifre successive (fino a che Q diventa 0).

Conversione da base 10 a base b

```
i = 0;
while (num != 0) {
    c[i] = num % b;
    num = num / b;
    i = i+1; }
```

N.B. Le cifre vengono determinate dalla meno significativa alla più significativa.

Esempio: $(25)_{10} = (???)_2 = (11001)_2$

$N : b$	Q	R	cifra	
25 : 2	12	1	c_0	↑
12 : 2	6	0	c_1	↑
6 : 2	3	0	c_2	↑
3 : 2	1	1	c_3	↑
1 : 2	0	1	c_4	↑

N.B. servono 5 bit (con cui possiamo rappresentare i numeri da 0 a 31)

Conversione tra le basi 2, 8 e 16

Basta prendere i bit a gruppi di 3 (oppure 4) e trovare il numero ottale (oppure esadecimale) corrispondente.

0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

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

Ad esempio

$347_8 = 011\ 011\ 111_2$

$F3A_{16} = 1111\ 0011\ 1010_2$

Rappresentazione di numeri interi

- Dobbiamo rappresentare anche il segno: si usa uno dei bit (quello più significativo)

Rappresentazione tramite modulo e segno

- il bit più significativo rappresenta il segno
- le altre $n-1$ cifre rappresentano il valore assoluto
- problemi:
  - doppia rappresentazione per lo zero ($00\cdots00$ e $10\cdots00$)
  - le operazioni aritmetiche sono complicate: somma trattata diversamente dalla sottrazione (analisi per casi)

$\Longrightarrow$ invece della rappresentazione tramite modulo e segno si usa una rappresentazione in complemento alla base

Rappresentazione in complemento alla base

In quanto segue b indica la base e n indica il numero complessivo di cifre.

- Con base b e n cifre, abbiamo a disposizione b^n configurazioni distinte.
- Utilizziamo metà delle configurazioni per rappresentare numeri positivi e l'altra metà per rappresentare numeri negativi.

$$||X|| = \begin{cases} X & \text{se } X \geq 0 \\ b^n - |X| & \text{se } X < 0 \end{cases}$$

- in questo modo si rappresentano gli interi relativi nell'intervallo $[-b^n/2,\ b^n/2)$
  - se $X \geq 0$: $||X||$ è compresa in $[0,\ b^n/2)$
  - se $X < 0$: $||X||$ è compresa in $[b^n/2,\ b^n)$
- lo 0 ha una sola rappresentazione

Ad esempio, in base 10, con $n = 4$, $X = 24$ risulta 9976.

Rappresentazione in complemento alla base

N	$b = 10$ e $n = 1$	$b = 2$ e $n = 3$
-5	5	
-4	6	100
-3	7	101
-2	8	110
-1	9	111
0	0	000
1	1	001
2	2	010
3	3	011
4	4	

- se $b = 2 \Longrightarrow$ rappresentazione in complemento a 2
  - rappresentazione degli interi relativi nell'intervallo $[-2^{n-1}, 2^{n-1})$
  - positivi: la cifra più significativa è 0 (rappresentati nella parte inferiore dell'intervallo)
  - negativi: la cifra più significativa è 1 (rappresentati nella parte superiore dell'intervallo)

Operazione di complementazione

Vogliamo determinare un algoritmo per determinare la rappresentazione in complemento alla base di $-X$, data quella di X. Indipendentemente dal segno di X, abbiamo:

$$||X|| + ||-X|| = |X| + b^n - |X| = b^n$$

per cui

$$||-X|| = b^n - ||X||$$

o equivalentemente

$$||-X|| = b^n - 1 - ||X|| + 1$$

Operazione di complementazione

- Supponiamo:

$$||X|| = \sum_{i=0}^{n-1} c_i \cdot b^i$$

e ricordiamo che la rappresentazione di $b^n - 1$ è

$$\sum_{i=0}^{n-1} (b-1) \cdot b^i$$

- Otteniamo:

$$\begin{aligned} ||-X|| &= b^n - 1 - ||X|| + 1 \\ &= (\sum_{i=0}^{n-1} (b-1) \cdot b^i) - (\sum_{i=0}^{n-1} c_i \cdot b^i) + 1 \end{aligned}$$

Operazione di complementazione

- Sia ora k la prima posizione significativa di $||X||$, ovvero la prima cifra (a partire da destra) diversa da 0. Abbiamo allora:

$$\begin{aligned} ||-X|| &= (\sum_{i=0}^{n-1} (b-1) \cdot b^i) - (\sum_{i=0}^{n-1} c_i \cdot b^i) + 1 \\ &= (\sum_{i=0}^{n-1} (b-1) \cdot b^i) - (\sum_{i=k}^{n-1} c_i \cdot b^i) + 1 \\ &= (\sum_{i=k+1}^{n-1} ((b-1) - c_i) \cdot b^i) + ((b-1) - c_k) \cdot b^k + \\ &\quad (\sum_{i=0}^{k-1} (b-1) \cdot b^i) + 1 \end{aligned}$$

- Osserviamo ora che $(\sum_{i=0}^{k-1} (b-1) \cdot b^i) + 1 = b^k$.

Operazione di complementazione

- Sia ora k la prima posizione significativa di $||X||$, ovvero la prima cifra (a partire da destra) diversa da 0. Abbiamo allora:

$$\begin{aligned} ||-X|| &= (\sum_{i=0}^{n-1}(b-1)\cdot b^i) - (\sum_{i=0}^{n-1} c_i \cdot b^i) + 1 \\ &= (\sum_{i=0}^{n-1}(b-1)\cdot b^i) - (\sum_{i=k}^{n-1} c_i \cdot b^i) + 1 \\ &= (\sum_{i=k+1}^{n-1}((b-1)-c_i)\cdot b^i) + ((b-1)-c_k)\cdot b^k + \quad b^k \end{aligned}$$

Operazione di complementazione

$$\begin{aligned} ||-X|| &= (\sum_{i=k+1}^{n-1}((b-1)-c_i)\cdot b^i) + ((b-1)-c_k)\cdot b^k + b^k \\ &= (\sum_{i=k+1}^{n-1}((b-1)-c_i)\cdot b^i) + (b-c_k)\cdot b^k \end{aligned}$$

- L'ultimo addendo è 0, poichè $c_i = 0$, per ogni $i = 0, \ldots, k-1$.
- Come possiamo leggere quanto ottenuto?

  La rappresentazione di $-X$ si ottiene da quella di X:

  1. ricopiando gli zeri meno significativi
  2. complementando alla base la prima cifra significativa
  3. complementando alla base meno uno le rimanenti cifre

Operazione di complementazione

$$||-X|| = (\sum_{i=k+1}^{n-1}((b-1)-c_i)\cdot b^i) + ((b-1)-c_k)\cdot b^k + b^k$$
$$= (\sum_{i=k+1}^{n-1}((b-1)-c_i)\cdot b^i) + (b-c_k)\cdot b^k + (\sum_{i=0}^{k-1} c_i \cdot b^i)$$

- L'ultimo addendo è 0, poichè $c_i = 0$, per ogni $i = 0, \ldots, k-1$.
- Come possiamo leggere quanto ottenuto?

  La rappresentazione di $-X$ si ottiene da quella di X:

  1. ricopiando gli zeri meno significativi
  2. complementando alla base la prima cifra significativa
  3. complementando alla base meno uno le rimanenti cifre

Operazione di complementazione

Esempio: $b = 3$, $n = 4$, $||X|| = 0210$ (dunque $X = (21)_{10}$)

$$||-X|| = 2020$$

Verifichiamo:

- $2020 = (60)_{10}$
- $3^4 - 60 = 81 - 60 = 21 = |-X|$

Complemento a 2

Nel caso del complemento a 2 abbiamo più semplicemente:

- partendo da destra, si lasciano inalterate tutte le cifre fino al primo 1 compreso
- si invertono le rimanenti cifre

Esempio: Rappresentazione di -14 e di -298 in complemento a 2:

- $14 = 8 + 4 + 2$
- $||14|| = 1110 \quad = \quad 00001110$
- $||-14|| = 11110010$

- $298 = 256 + 32 + 8 + 2$
- $||298|| = 100101010 \quad = \quad 0100101010$
- $||-298|| = 1011010110$

Operazioni su interi relativi in complemento a 2

Somma di due numeri

- si effettua bit a bit
- non è necessario preoccuparsi dei segni
- il risultato sarà corretto (in complemento a 2 se negativo)
- può verificarsi trabocco (overflow) $\Longrightarrow$ il risultato non è corretto

  Si verifica quando il numero di bit a disposizione non è sufficiente per rappresentare il risultato.

Operazioni su interi relativi in complemento a 2

Esempio: $n = 5$, $\pm 9 \pm 3$, $\pm 9 \pm 8$

intervallo di rappresentazione: da -2^4 a $2^4 - 1$ (da -16 a 15)

rip.	00011		11001		00111		11111
+9	01001	+9	01001	-9	10111	-9	10111
+3	00011	-3	11101	+3	00011	-3	11101
+12	01100	+6	00110	-6	11010	-12	10100

In questi casi non si ha trabocco.

rip.	01000		10000
+9	01001	-9	10111
+8	01000	-8	11000
-15	10001	+15	01111
(e non +17)		(e non −17)	

Si ha trabocco quando il riporto sul bit di segno è diverso dall'ultimo riporto.

Operazioni su interi relativi in complemento a 2

Differenza tra due numeri: si somma al primo il complemento del secondo

Esempio: $n = 5$, intervallo di rappresentazione: da -16 a 15

rip.		11111
	+9	01001
	-9	10111
	0	00000

	11001
+9	01001
-3	11101
+6	00110

Numeri frazionari

- Numeri reali compresi tra 0 e 1: si rappresentano comunemente come

$$N = 0.c_{-1}c_{-2}\ldots c_{-n}$$

- Il peso delle cifre dipende, al solito, dalla loro posizione e dalla base prescelta

$$N_b = c_{-1}\cdot b^{-1} + c_{-2}\cdot b^{-2} + \cdots + c_{-n}\cdot b^{-n} = \sum_{i=-n}^{-1} c_i \cdot b^i$$

Esempio: Consideriamo $b = 10$ ed il numero 0.587

$$0.587_{10} = 5\cdot 10^{-1} + 8\cdot 10^{-2} + 7\cdot 10^{-3}$$

Numeri frazionari

$$N_b = \sum_{i=-n}^{-1} c_i \cdot b^i \qquad (\bullet)$$

- Nel caso di un numero frazionario in binario, possiamo usare la $(\bullet)$ per convertirlo in base 10

  Esempio: Convertiamo in base 10 il numero frazionario binario 0.1011_2

$$0.1011_2 = 1\cdot 2^{-1} + 1\cdot 2^{-3} + 1\cdot 2^{-4} = 0.6875_{10}$$

- La rappresentazione dei numeri frazionari può introdurre approssimazioni dovute alla limitatezza delle cifre dopo la virgola.
- L'approssimazione è comunque inferiore a b^{-n} dove n è il numero di cifre utilizzate.

Conversione di un numero frazionario da base 10 a base 2

- Il metodo più semplice consiste nell'effettuare una sequenza di moltiplicazioni per 2 prendendo ad ogni passo la parte intera del risultato come cifra binaria della rappresentazione
- **Esempio:** Convertiamo 0.125 in base 2

$$\begin{array}{rcr|l} 0.125 \times 2 & = & 0.25 & 0 \\ 0.25 \times 2 & = & 0.5 & 0 \\ 0.5 \times 2 & = & 1.0 & 1 \end{array}$$

- In questo caso abbiamo una rappresentazione esatta su 3 cifre ($0.125 = 1/8$)

$$0.125_{10} = 0.001_2$$

Conversione di un numero frazionario da base 10 a base 2

- **Esempio:** Convertiamo 0.587_{10} in base 2

$$\begin{array}{rcr|l} 0.587 \times 2 & = & 1.174 & 1 \\ 0.174 \times 2 & = & 0.348 & 0 \\ 0.348 \times 2 & = & 0.696 & 0 \\ 0.696 \times 2 & = & 1.392 & 1 \\ 0.392 \times 2 & = & 0.784 & 0 \\ 0.784 \times 2 & = & 1.568 & 1 \\ 0.568 \times 2 & = & \ldots & \end{array}$$

- Quindi la rappresentazione di 0.587_{10} in base 2 è:
  - 0.1001_2 con 4 cifre (approssimazione accurata entro 2^{-4})
  - 0.100101_2 con 6 cifre (approssimazione accurata entro 2^{-6})

L'aritmetica reale

- L'insieme dei reali (e dei razionali) è infinito $\Longrightarrow$ non è possibile rapprentarlo tutto

Rappresentazione in virgola fissa

Si rappresentano separatamente, usando un numero fissato di cifre
- parte intera e,
- parte frazionaria

(si usa una virgola per separare le due parti)

$$N_b = c_{n-1}\ c_{n-2}\ \cdots\ c_1\ c_0\,,c_{-1}\ c_{-2}\ \cdots\ c_{-m}$$

rappresenta il numero

$$N = c_{n-1} \cdot b^{n-1} + \cdots + c_0 \cdot b^0 + c_{-1} \cdot b^{-1} + \cdots + c_{-m} \cdot b^{-m}$$

L'aritmetica reale

- Limitazioni della rappresentazione:
  - k bit per la parte intera $\Longrightarrow$ $(-2^k,\ 2^k)$
  - m bit per la parte frazionaria $\Longrightarrow$ precisione $\leq 2^{-m}$

Rappresentazione in virgola mobile (floating point)

Utilizza la notazione esponenziale. Si esprime il numero come prodotto di due parti

$$X = m \cdot b^e$$

Esempio:

$$1150 = 1.15 \times 10^3$$

ma anche

$$1150 = 0.115 \times 10^4$$

Rappresentazione in virgola mobile

Rappresentazione in forma normalizzata in base b

$$X = m \cdot b^e$$

- e è la caratteristica in base b di X: intero relativo
- m è la mantissa in base b di X: numero frazionario tale che $1/b \leq |m| < 1$
- **Esempio:**

$$1150 = \underbrace{0.115}_{\text{mantissa}} \times \underbrace{10^4}_{\text{caratteristica}}$$

Rappresentazione in virgola mobile

- Se la caratteristica è rappresentata dalla sequenza di cifre

$$c_1 \; c_2 \; c_3 \; \cdots$$

allora rappresenta il valore

$$c_1 \cdot b^{-1} + c_2 b^{-2} + \cdots$$

Esempio: $X = (5)_{10} = (101)_2$ che normalizzato diventa:

$$\begin{aligned} m &= |m| = (0.101 \cdots 0000)_2 \\ e &= (11)_2 \end{aligned}$$

Rappresentazione in virgola mobile

- Fissati:
  - k bit per mantissa
  - h bit per caratteristica
  - 1 bit per il segno

  l'insieme di reali rappresentabili è fissato (e limitato)

$$(0.1) \quad 1/2 \quad \leq \quad |m| \leq \sum_{i=1}^{k} 2^{-i} \quad (0.11\ldots1)$$

$$|e| \quad \leq \quad 2^{h-1} - 1$$

- Questo fissa anche massimo e minimo (in valore assoluto) numero rappresentabile.
- Ipotesi realistica: reali rappresentati con 32 bit:
  - 24 bit per la mantissa
  - 7 bit per la caratteristica (in complemento)
  - 1 bit per il segno della mantissa (0 positivo, 1 negativo)

Rappresentazione in virgola mobile

Insieme $\mathcal{F}$ dei numeri rappresentabili in virgola mobile

- sottoinsieme finito dei numeri razionali rappresentabili (con n bit)
- simmetrico rispetto allo 0
- gli elementi non sono uniformemente distribuiti sull'asse reale
  - densi intorno allo 0
  - radi intorno al massimo rappresentabile

    $m_1 = 0.10, m_2 = 0.11, e_1 = 5$
    $X_1 = 0.10 \times 10^5 = 10000$
    $X_2 = 0.11 \times 10^5 = 11000$
- molti razionali non appartengono ad $\mathcal{F}$ (ed es. $1/3$, $1/5$, ...)
- non è chiuso rispetto ad addizioni e moltiplicazioni
- per rappresentare un reale X si sceglie l'elemento di $\mathcal{F}$ più vicino ad X
- la funzione che associa ad un reale X l'elemento di $\mathcal{F}$ più vicino ad X è detta funzione di arrotondamento

Limitazioni aritmetiche

Dovute al fatto che il numero di bit usati per rappresentare un numero è limitato

- perdita di precisione
- arrotondamento: mantissa non sufficiente a rappresentare tutte le cifre significative del numero
- errore di overflow: caratteristica non sufficiente (numero troppo grande)
- errore di underflow: numero troppo piccolo viene rappresentato come 0

Formati standard proposti da IEEE (Institute of Electrical and Electronics Engineers)

- singola precisione: 32 bit
- doppia precisione: 64 bit
- quadrupla precisione: 128 bit

Rappresentazione dei caratteri

- Le lettere sono appunto 26. Tuttavia quando scriviamo usiamo lettere minuscole, maiuscole, lettere accentate, simboli di punteggiatura, etc.
- Con 8 bit (e quindi $28 = 256$ possibilità) possiamo ragionevolmente rappresentare tutti questi simboli.

Codici

- ASCII (American Standard Code for Information Interchange)
  - 7 bit Standard ASCII
  - 8 bit Extended ASCII
- 16 bit UNICODE che assegna un numero univoco a ogni carattere, indipendentemente dalla piattaforma, dall'applicazione, e dalla lingua. UNICODE permette la rappresentazione dei caratteri di molti alfabeti non latini: arabo, ebraico,cinese, giapponese, coreano, tailandese, ecc, e anche di caratteri speciali (matematici, tecnici, frecce, elementi grafici, etc.).

Codifica delle immagini e dei filmati

- Ogni immagine viene rappresentata con sequenze di PIXEL (Picture Element)
- Ad ogni pixel viene assegnato una sequenza di bit. Ad es., con 4 bit possiamo rappresentare $2^4 = 16$ colori diversi, mentre con 8 bit ne abbiamo $2^8 = 256$.
- I filmati sono memorizzati come sequenze di immagini o fotogrammi, solitamente compresse per risparmiare spazio (si possono ad es. memorizzare solo le variazioni tra un fotogramma e l'altro).

Codifica dei suoni

- Fisicamente un suono è rappresentato come un'onda che descrive la variazione della pressione dell'aria nel tempo (onda sonora)
- Misurando il valore dell'onda ad intervalli di tempo costanti si ottengono dei valori numerici facili da codificare.
- Quanto più frequentemente il valore dell'onda viene campionato, tanto più precisa sarà la sua rappresentazione.
- Il numero di campioni raccolti per ogni secondo definisce la frequenza di campionamento che si misura in Hertz (Hz)

Gnocchi in brodo (Pellegrino Artusi, 1891)

... è una minestra da farsene onore; ma se non volete consumare appositamente per lei un petto di pollastra o di cappone, aspettate che vi capiti d'occasione. Cuocete nell'acqua, o meglio a vapore, grammi 200 di patate grosse e farinacee e passatele per istaccio. A queste unite il petto di pollo lesso tritato finissimo colla lunetta, grammi 40 di parmigiano grattato, due rossi d'uovo, sale quanto basta e odore di noce moscata. Mescolate e versate il composto sulla spianatoia sopra a grammi 30 o 40 (che tanti devono bastare) di farina per legarlo, e poterlo tirare a bastoncini grossi quanto il dito mignolo. Tagliate questi a tocchetti e gettateli nel brodo bollente ove una cottura di cinque o sei minuti sarà sufficiente. Questa dose potrà bastare per sette od otto persone. Se il petto di pollo è grosso, due soli rossi non saranno sufficienti...

La ricetta è piacevole da leggere, ma non è facile da seguire.