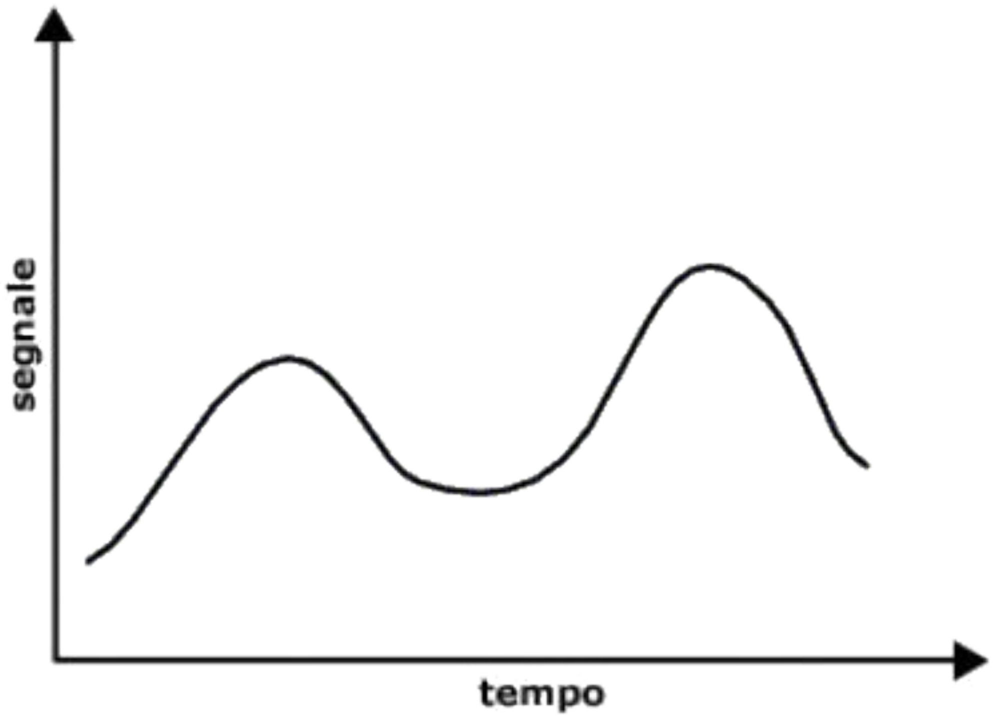
Architettura degli Elaboratori e Laboratorio

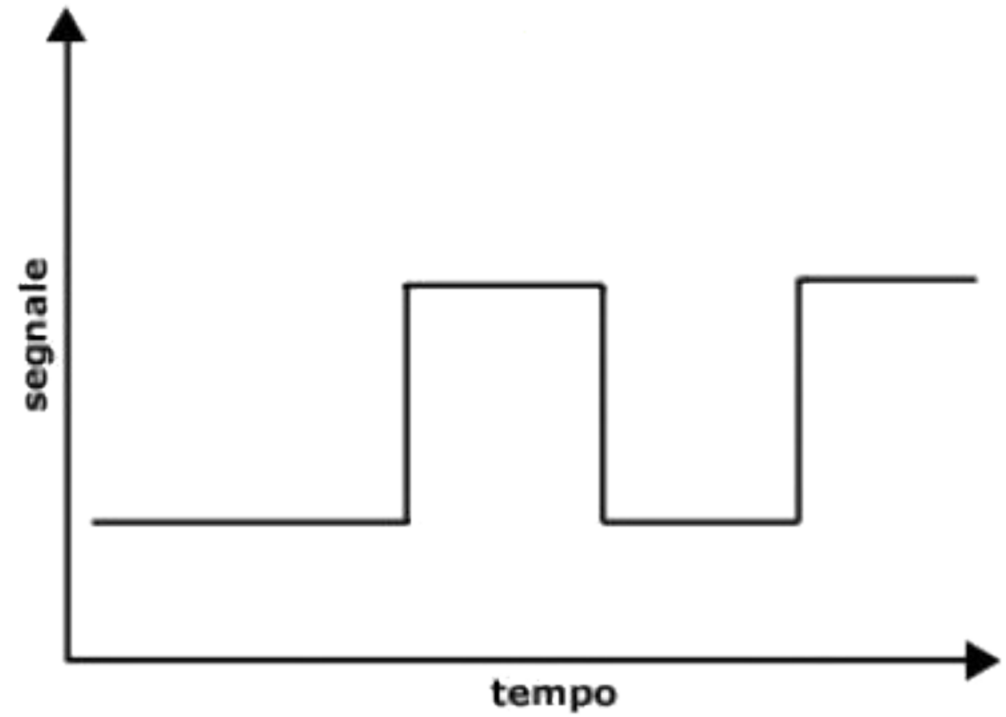
Matteo Manzali

Università degli Studi di Ferrara

Anno Accademico 2016 - 2017

Analogico vs digitale





Segnale analogico

Un segnale è analogico quando i valori utili che lo rappresentano sono continui (infiniti).

Segnale digitale

Un segnale è digitale quando i valori utili che lo rappresentano sono discreti e finiti.

I sistemi digitali

- In genere la moderna tecnologia si basa su segnali digitali (tecnologia digitale).
- I calcolatori utilizzano segnali digitali che possono assumere due stati logici:
  - le informazioni vengono rappresentate da sequenze binarie
  - alcune sequenze hanno nomi specifici

bit

01010100001110101011010101110100

byte

word

Aritmetica dei calcolatori

- L'aritmetica usata dai calcolatori è diversa da quella comunemente utilizzata dalle persone:
  - i numeri devono essere memorizzati entro un limitato spazio di memoria
  - la precisione con cui i numeri possono essere espressi è finita e predeterminata (dipende dall'architettura)
- Es.:

$$\pi = \boxed{3}\boxed{,}\boxed{1}\boxed{4}\boxed{1}\;5\;9\;2\;9\;\ldots$$

Numeri a precisione finita

- I numeri a precisione finita sono quelli rappresentati con un numero finito di cifre.
- Fissando le caratteristiche della precisione si determina anche l'insieme di valori rappresentabili.
- Es.:

  "Si possono rappresentare solo numeri naturali con al massimo due cifre"

  Insieme dei valori rappresentabili: 0 → 99

  Esempi di valori NON rappresentabili:

  999   0,123   -12   etc…

Operazioni a precisione finita

- Le operazioni con numeri a precisione finita causano errori nel caso in cui il risultato non appartiene all'insieme dei valori rappresentabili:
  - **underflow** → il risultato è minore del più piccolo valore rappresentabile
  - **overflow** → il risultato è maggiore del più grande valore rappresentabile
  - **non appartenenza all'insieme** → il risultato non è rappresentabile (pur non essendoci ne underflow ne overflow)

Operazioni a precisione finita

- Es.:

  “Si possono rappresentare solo numeri naturali con al massimo due cifre”

  20 - 90 = -70 (underflow)

  90 + 20 = 110 (overflow)

  5 / 2 = 2,5 (non appartenenza all’insieme)

Notazione posizionale

- La numerazione decimale che utilizziamo quotidianamente è una **notazione posizionale in base 10**.
- Nella notazione posizionale si associano alle cifre un diverso valore in base alla posizione che occupano nella notazione.

1234 è diverso da **4321**

- Base 10 significa che utilizziamo 10 cifre diverse per la numerazione (0, 1, 2, 3, 4, 5, 6, 7, 8, 9).

Notazione posizionale

- Scelta una **base** di rappresentazione B:
  - ogni numero è rappresentato da una sequenza di simboli (**cifre**) appartenente ad un alfabeto di **B simboli** distinti
  - ogni cifra rappresenta un valore compreso **fra 0 e B-1**
  - a ogni posizione corrisponde un **peso**, uguale ad una potenza della base crescente da destra a sinistra
  - valore del numero = somma dei prodotti di ciascuna cifra per il peso associato alla sua posizione
- Esempio di rappresentazione su N cifre:

$$d_{N-1}\, d_{N-2} \ldots d_1\, d_0 = d_{N-1} * B^{N-1} + d_{N-2} * B^{N-2} + \ldots + d_1 * B^1 + d_0 * B^0$$

Cifra più significativa

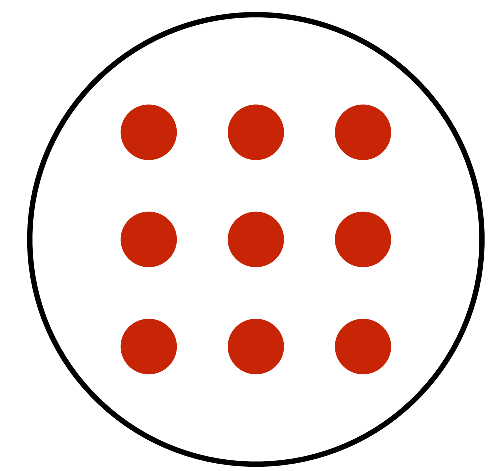
Cifra meno significativa

Sistemi con basi diverse

- Simboli ammessi con sistemi di basi diverse:
  - sistema **binario** (B=2): 0 1
  - sistema **decimale** (B=10): 0 1 2 3 4 5 6 7 8 9
  - sistema **esadecimale** (B=16): 0 1 2 3 4 5 6 7 8 9 A B C D E F
- I sistemi con base B > 10 richiedono dei simboli aggiuntivi rispetto al sistema decimale.
- Per convenzione si utilizzano le lettere dell’alfabeto:
  - A = 10
  - B = 11
  - etc…

Sistemi con basi diverse

- Ad ogni numero corrispondo notazioni diverse in basi diverse:
  - sistema **binario**:
    $1001 \rightarrow 1x2^3 + 0x2^2 + 0x2^1 + 1x2^0$
  - sistema **ottale**:
    $11 \rightarrow 1x8^1 + 1x8^0$
  - sistema **decimale**:
    $9 \rightarrow 9x10^0$



- Queste tre notazioni rappresentano lo stesso numero in basi diverse!
- 1001, 11, 9 … Come capire la base?

Sistemi con basi diverse

“Esistono 10 tipi di persone: quelle che capiscono la notazione binaria e quelli che non la capiscono.”

- Per evitare ambiguità se si utilizzano basi diverse bisogna specificare la base per ogni numero:
  - $(11111010001)_{\mathbf{due}} = (2001)_{\mathbf{dieci}} = (7D1)_{\mathbf{sedici}}$

base

  - Base in lettere, così è implicitamente decimale!

Conversione tra basi

- Convertire un numero da **base B a base 10** (somma dei pesi):
  - $(b)_B = (b_n \times B^n + b_{n-1} \times B^{n-1} + \ldots b_1 \times B^1 + b_0 \times B^0)_B = (c)_{dieci}$
  - si sommano i pesi delle singole cifre
  - è il metodo più intuitivo (ne esistono altri)
  - Esempio: *"convertire $(1011)_{cinque}$ in base 10"*

    $(\mathbf{1011})_{cinque} = \mathbf{1} \times 5^3 + \mathbf{0} \times 5^2 + \mathbf{1} \times 5^1 + \mathbf{1} \times 5^0 =$
    $125 + 0 + 5 + 1 = 131$

    $(1011)_{cinque} = (131)_{dieci}$

Conversione tra basi

- Convertire un numero da **base 10 a base B** (serie di divisioni):
  - $(c)_{dieci} = (c_n\ c_{n-1}\ \ldots\ c_1\ c_0)_{dieci} = (b_m\ b_{m-1}\ \ldots\ b_1\ b_0)_B$
  - se divido $(c)_{dieci}$ per B e prendo il resto ottengo b_0
  - se divido per B il quoziente della divisione precedente e prendo il resto ottengo b_1
  - continuo fino a quando il quoziente non è zero
  - Esempio: *"convertire $(131)_{dieci}$ in base 5"*

131 / 5 = 26 resto **1**
26 / 5 = 5 resto **1**
5 / 5 = 1 resto **0**
1 / 5 = 0 resto **1**

$(131)_{10} = (1011)_5$

I numeri binari

- Il sistema binario (**bin**) viene ampiamente utilizzato nella tecnologia digitale:
  - ogni cifra può essere rappresentata tramite un livello di tensione (es. 0 → 0V , 1 → 5V)
- Anche il sistema esadecimale (**hex**) è molto usato nei calcolatori
- I bit (le cifre di un numero binario) vengono spesso rappresentati raggruppati di 4 in 4:
  - facilità la conversione tra numeri binari ed esadecimali

    $(0010\ 1000\ 0000\ 1111)_{due}$
    $(\ 2\quad 8\quad 0\quad F\ )_{sedici}$

  - Alternativa per rappresentazione esadecimale: **0x**280F

I numeri interi

- In un sistema di calcolo i numeri interi sono tipicamente rappresentati con un numero di bit che è una potenza di 2:
  - 8 bit → char
  - 16 bit → short
  - 32 bit → int
  - 64 bit → long long int
- I numeri possono essere con segno (signed) o senza segno (unsigned):
  - char o unsigned char
  - int or unsigned int

Numeri senza segno

- Nei calcolatori i numeri naturali (**interi senza segno**) vengono rappresentati con la loro notazione in base 2.

- Vanno considerati sempre tutti i bit previsti dalla precisione scelta.

0000 0000 = 0
0000 0001 = 1
0000 0010 = 2
0000 0011 = 3
…
1111 1110 = 254
1111 1111 = 255 = $2^8 - 1$

Rappresentazione a 8 bit

Least Significant Bit (LSB)

Most Significant Bit (MSB)

- Con n bit si possono rappresentare 2^n numeri, da 0 a 2^n - 1.

Addizione

- Valgono le stesse regole dell'addizione con i numeri decimali (somma e riporto).
- Quando entrambi i bit hanno valore 1 si ha il riporto.
- Se la somma dei due bit più significativi genera riporto, si ha un overflow.
- Es.:

```
                          riporto
                 11                   1
1000 +          1001 +                 1000 +
0011 =          0011 =                 1000 =
------          ------                 ------
1011            1100                  10000

                                      overflow
```

Numeri con segno

- Per i numeri interi con segno si sono inventate diverse rappresentazioni nel tempo.
- Quella universalmente utilizzata ora è **“complemento a 2”**.
- Per calcolare l’opposto di un numero si procede come segue:
    - si negano tutti i bit del numero
    - si somma +1
- Es. *“convertire 3 in -3 (utilizzando 4 bit per la rappresentazione)”*

  `0011` → 3
  `1100` → nego i bit
  `1101` → sommo +1

Numeri con segno

- Riguardo al complemento a 2:
  - Il bit più significativo (MSB) rappresenta sempre il segno:
    - 0 → positivo, 1 → negativo
  - Rappresentazione dei numeri positivi identica a quella dei numeri senza segno.
  - Il complemento a 2 permette di implementare la sottrazione come somma dell'opposto:
    - (X - Y) = (X + (-Y))
    - semplifica la logica del processore

Numeri con segno

- Esempio di rappresentazione a 4 bit di numeri in complemento a 2:

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

bit del segno

- Dati n bit, il range dei valori rappresentabili con il complemento a 2 è $[-2^{n-1}, 2^{n-1} - 1]$.

Estensione del segno

- Dato un numero a n bit in complemento a 2, è possibile aumentare il numero di bit con cui il numero viene rappresentato estendendo il bit del segno:

`0011` (+3 a 4 bit) → `0000 0011` (+3 a 8 bit)

`1101` (-3 a 4 bit) → `1111 1101` (-3 a 8 bit)

Sottrazione

- Valgono le stesse regole della sottrazione con i numeri decimali (sottrazione e prestito).
- **Metodo alternativo**: grazie al complemento a 2 si può implementare la sottrazione come somma dell'opposto.
- Es.: *"calcolare 5 - 3 tramite rappresentazione binaria a 4 bit"*

$(5)_{dieci} = (0101)_{due}$ $(3)_{dieci} = (0011)_{due}$

```
0101 - 0011 = 0101 + (-0011) = 0101 + (1100+1) =
0101 + 1101 = 0010
```

complemento a 2

```
  0101 +
  1101 =
 -------
 10010
```

Moltiplicazione

- Prima un esempio in base dieci:

```
   123 x      // moltiplicando
    45 =      // moltiplicatore
 ----------
   615 +
  492
 ----------
  5535
```

- Si moltiplica il moltiplicando per ogni cifra del moltiplicatore a partire dalla meno significativa.

- Ad ogni iterazione si sposta a sinistra di una cifra il risultato parziale rispetto ai risultati parziali precedenti.

Moltiplicazione

- In binario il procedimento è simile:

```
   1110 x
   0110 =
   ------
   0000 +
  1110  +
 1110   +
0000    =
---------
1010100
```

- Ogni risultato parziale o è zero o è il moltiplicando spostato progressivamente a sinistra.
- La moltiplicazione si riduce ad una serie di somme e spostamenti a sinistra.