Macchina Digitale

> macchina digitale
>
> - sistema artificiale
>   - immagazzina, elabora e comunica informazioni
>   - impiega i segnali digitali (digitale: sinonimo di discreto)

Segnali Digitali ed Analogici

segnali analogici

> segnali analogici
>
> Le grandezze fisiche che noi percepiamo sono segnali analogici, ovvero variano in **modo continuo** all'interno di un intervallo di valori ammissibili.
>
> L'informazione è rappresentata da ogni possibile valore della grandezza fisica.
> Posso dunque rappresentare infinite informazioni con un unico segnale.
>
> Tuttavia anche un piccolo segnale mi distrugge l'informazione, e inoltre ho bisogno di dispositivi sofisticati per riconoscere il segnale.

segnali digitali

> segnali digitali
>
> Molto piè robusti, poiche qui è l'**intervallo** in cui si trova la grandezza a rappresnetare l'informazione.
>
> - meno informazioni
> - maggior robustezza
> - minor complessità
>
> Diminuendo gli intervalli si avrà maggior robustezza.
> Il minor numero di intervalli lo hanno i segnali binari.

segnali binari

> segnali binari
>
> - massima robustezza (**due intervalli**)

- informazione è data dal sapere se il segnale si trova sopra o sotto una data soglia
- per maggiore robustezza: **zona intermedia** dove segnale *non* definito
- i due livelli sono : ***H***igh e ***L***ow

Per alta robustezza dunque gestirò il segnale analogico sottostante.

Bit

Per descrivere a livello logico un segnale binario, uso una variabile binaria: **bit**.

> bit
>
> può assumere 0 o 1.
>
> - generalmente **non** sono numeri, ma **valori logici** (simboli), e indicano se il segnale è sopra o sotto la soglia.
> - generalmente si assume la **logica positiva**: con H rappresento 1.
>
> Efficacemente rappresentato da un **interruttore**.
>
> > interruttore
> >
> > essendo parte della realtà, è analogico, tuttavia attraversa velocemente i valori intermedi, e dunque può essere interpretato come digitale
> >
> > ## interruttore meccanico
> >
> > ## interruttore elettronico
> >
> > - transistor
> > - indipendenti dalla tecnologia
> > - vantaggio: usare l'uscita di un transistor per pilotarne altri, creando **reti di interruttori** che si influenzano a vicenda senza intervento umano

Codici binari

Per manetenere la robustezza del segnale binario ma rappresentare anche più informazioni, uso un codice binario.

> codice binario
>
> - una stringa (sequenza) di segnali binari.
> - le stringhe su cui chi genera e riceve l'informazione si sono accordati e formano un codice binario

#esempio
Per rappresentare 7 informazioni: $2^n \geq 7 \rightarrow n \geq [log_2 7] = [2.807] = 3$
Ovvero considero la prima potenza di 2 $\geq$ del numero richiesto.

- possono anche esistere configurazioni non valide

Definizione: Funzione dall'insieme delle 2^n configurazioni i n bit ad un insieme di M informazioni.
Condizione necessaria per la codifica: $2^n \geq M$ (se vi sono M simboli da codificare, occorrono almeno $2^n \geq M$ differenti configurazioni binarie)

Livelli di Astrazione

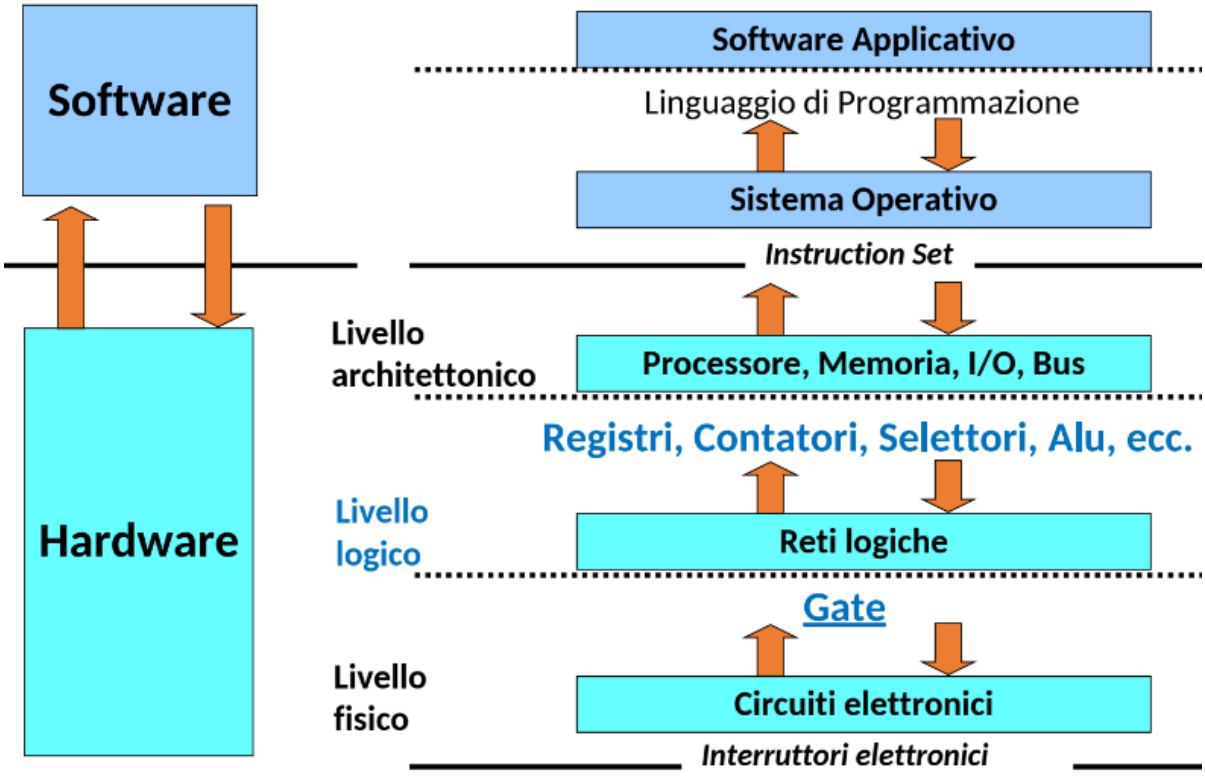
Un sistema complesso è articolati su più livelli, dunque è necessario **astrarre**.

astrazione

Ogni livello di un sistema complesso individua componenti primitivila cui struttura è definita nel livello sottostante e di cui ci interessa solo il comportamento

- dal livello alto al basso:
  - aumenta il numero di entità
  - diminuisce la complessità

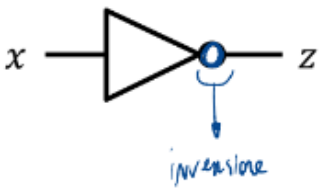
1. #esempio
   printf in C, la usiamo senza sapere come è fatta
2. #esempio



Reti Logiche

> rete logica
>
> Una rete logica è un'astrazione per una combinazione di interruttori che elaborano segnali binari.
>
> - per astrarre dalla complessità:
> si definiscono **componenti elementari**: *gate*, realizzati da interruttori elettronici.
> - ci interressa il loro **comportamento** (non la struttura → astrazione).
>
> cos'è:
> modello astratto che assume come primitive alcune semplici elaborazioni di segnali binari (gate) e permette di dedurre:
>
> 1. **struttura** di un dato **comportamento** (sintesi)
> 2. **comportamento** di una data **struttura**(analisi)

Gate

> gate
>
> # Gate con 1 ingresso
>
> - il numero di funzioni diverse di n ingressi binari con uscita binaria è
>
> $$2^{2^n}$$
>
> - i componenti elementari (funzioni) sono quindi 4.
> - tolte identità e costanti, rimane una funziona: l'operatore NOT
>
> 1. ## Gate NOT
>
> Tabella della Verità
>
> | x | z |
> |---|---|
> | 0 | 1 |
> | 1 | 0 |
>
> Simbolo
>
> 
>
>
> Espressione
>
> $z = \overline{x}$
> oppure
> $z = x'$
>
> # Gate con 2 ingressi
>
> 2^{2^n} con $n = 2$, dunque 16 possibili funzioni elementari

2. Gate AND

- astrazione di due interruttori in **serie**:

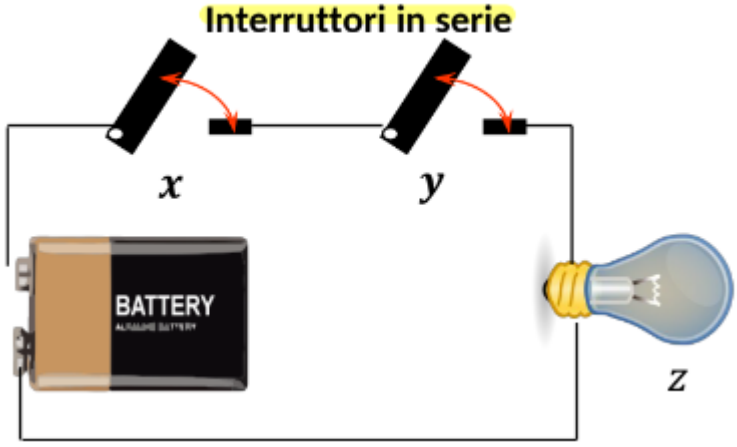


Tabella della Verità

x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

Simbolo

Espressione

Come un prodotto

$z = x \cdot y$

oppure

$z = xy$

Se è vero 'x' AND è vero 'y' ⇒ vero

-

3. Gate OR

- astrazione di due interruttori in **parallelo**:

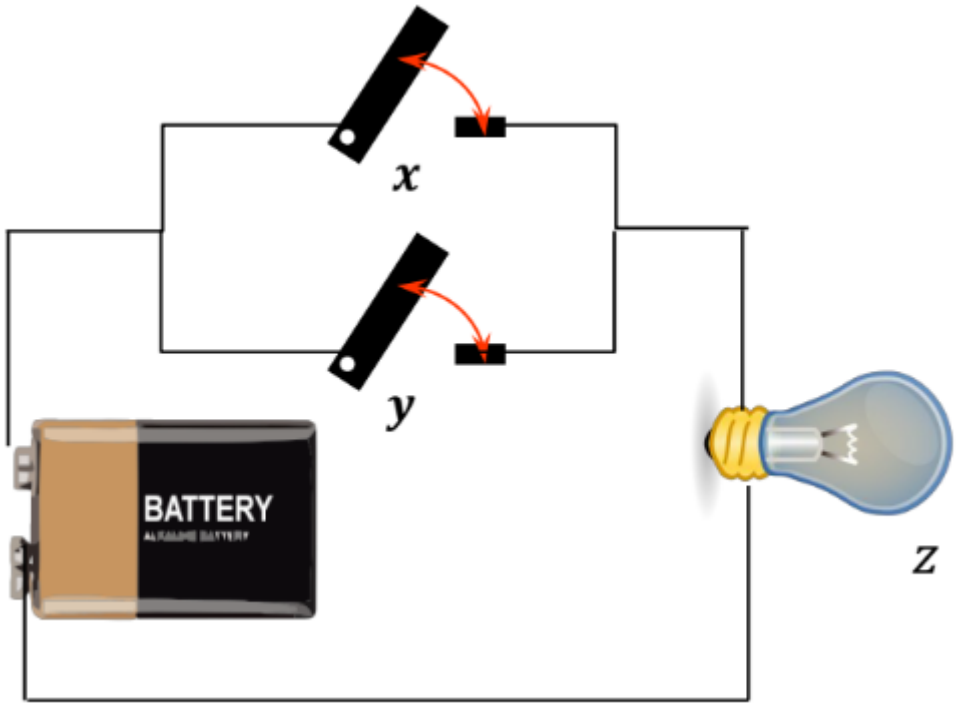
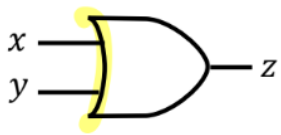


Tabella della Verità

x	y	z
0	0	0
0	1	1
1	0	1
1	1	1

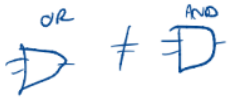
Simbolo



Espressione

Come la somma

$$z = x + y$$



Se c'è ALMENO un (1)

-

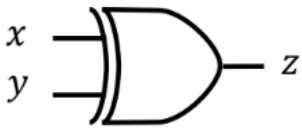
4. **Gate EXOR**

- EXCLUSIVE OR
- astrazione di due deviatori
- assume valore 1 se un ingresso ha valore 1 ma non entrambi

Tabella della Verità

x	y	z
0	0	0
0	1	1
1	0	1
1	1	0

Simbolo



Espressione

$$z = x \oplus y$$

Solo quando sono Diversi

-
- anche detto **somma a modulo 2** (output interpretato come risultato somma dei due bit)

Gate con più ingressi

- AND vale 1 se tutti gli ingressi hanno valore 1
- OR vale 1 se almeno un ingresso ha valore 1
- EXOR vale 1 se se e solo se il numero di '1' in ingresso è dispari

Gate negati:

1. NAND
2. NOR
3. EXNOR

Tabella della Verità	Simbolo	Espressione

x	y	z
0	0	1
0	1	1
1	0	1
1	1	0

NOT-AND → Contrario di AND

NAND

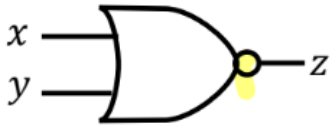
$z = x \uparrow y$

oppure

$z = \overline{xy}$

x	y	z
0	0	1
0	1	0
1	0	0
1	1	0

NOR



$z = x \downarrow y$

oppure

$z = \overline{x + y}$

x	y	z
0	0	1
0	1	0
1	0	0
1	1	1

EXNOR (o EQUIVALENCE)

$z = x \equiv y$

oppure

$z = \overline{x \oplus y}$

- EXNOR anche chiamato EQUIVALENCE perchè in **due ingressi** ha uscita 1 solo se entrambi gli ingressi sono uguali (non vale per più ingressi)

Tutti ottenibili da un NOT a cascata:

Diagrammi ad occhio

per rappresentare l'evoluzione di gruppi di segnali binari in forma compatta.
Dato che i segnali analogici non cambiano valore istantaneamente, vengono riportati anche i transitori alla successiva configurazione.

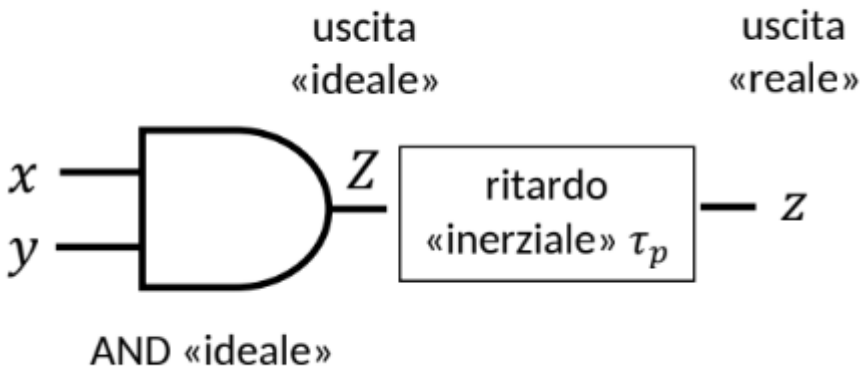
Bus di segnali

bus: insieme di segnali

Ritardi di propagazione

Dato che i gate sono componenti reali, esiste un ritardo di propagazione.
Quando cambia un ingresso di un gate, l'uscita non cambia istantaneamente, ma dopo un tempo t_p che dipende dalla tecnologia usata.

Questo ritardo è un ritardo "inerziale": un impulso di durata $< t_p$ su uno degli ingressi non appare in uscita.
Dunque esiste un limite superiore per la velocità di funzionamento di ogni gate.



Montaggi gate

Dati due gate, è possibile montarli:

1. in **serie**
2. in **parallelo**
3. in **retroazione** (uscita di un gate è ingresso dell'altro e viceversa)

Conversione A/D e D/A

Se input ed output sono segnali analogici, è possibile convertirli in binari e viceversa.
#esempio
scena reale, è possibile convertirla in un immagine digitale.

Informazione

> informazione
>
> - una macchina digitale elabora e comunica **informazione**.
> - Informazione: stringa di lunghezza finita di simboli appartenenti ad un alfabeto
>   - l'alfabeto definisce le informazioni "elementari"
>
> La macchina impiega un alfabeto binario, e dunque ogni informazione è rappresentata da una stringa di bit.
> Attraverso codici binari si trasformano informazioni di natura diversa in

> informazione sotto forma di stringhe di bit $\rightarrow$ **rappresentazione binaria dell'informazione**.
>
> In una stringa, il bit più significativo è quello tutto a sinistra.

Guarda definizione codice binario.

Proprietà di un codice

Un codice è una rappresentazione convenzionale dell'informazione.
La scelta di un codice deve essere condivisa da sorgente e destinazione, ed ha due gradi di libertà:

1. il numero di bit
2. l'associazione tra configurazioni ed informazioni

A parità di n ed M, le configurazioni possibili sono

$$C = 2^n!/(2^n - M)!$$

Codici ridondanti

Dato che la condizione per rendere necessaria la codifica è $2^n \geq M$, allora il numero minimo di bit è: $n_{min} = [log_2 M]$

- un codice che usa un numero $n > n_{min}$ è detto **codice ridonante**.

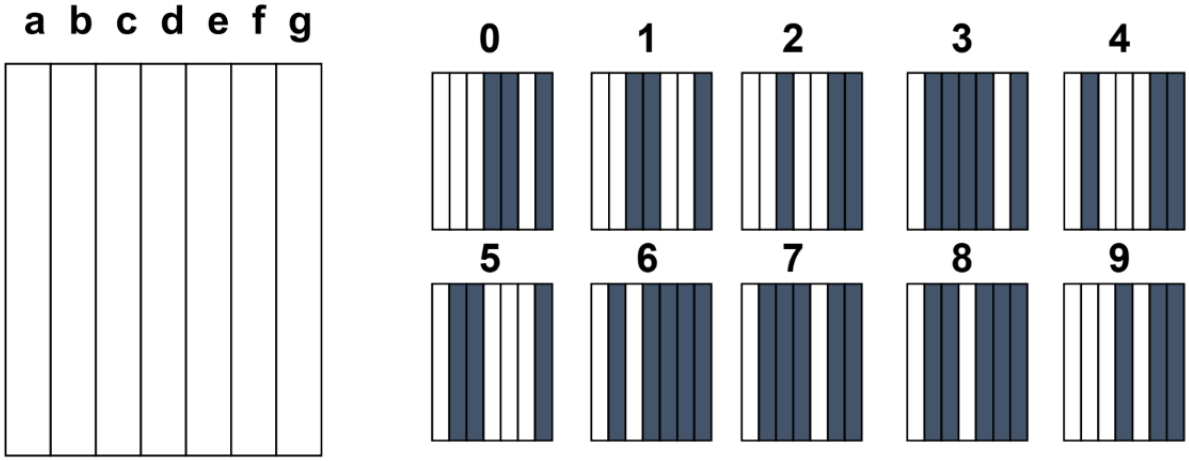
Gli umani preferiscono il ridondante, da maggiore robustezza.

Codice a 7 segmenti

Il codice per visualizzare a schermo i numeri decimali, è **ridondante**.
(0 acceso, 1 spento)

Universal Product Code

è un codice ridondante per associare un valore numerico ad un prodotto.



Per essere più leggibile dalla macchina inizia col bianco e finisce col nero sempre.

> rappresentazione dei numeri
>
> # Il sistema posizionale decimale
>
> Gli umani sono abituati ad una rappresentazione in **base 10**.
> #esempio
> $2048 = 2 \times 10^3 + 0 \times 10^2 + 4 \times 10^1 + 8 \times 10^0$
>
> Dato che le macchina digitale operano su segnali binari, il sistema posizionale usato è in **base 2**.
>
> ## Rappresentazione posizionale
>
> un numero in base $\beta \geq 2$ è rappresentato da $n + m$ cifre c_j (parte intera e parte frazionaria):
>
> $$(N)_\beta = (c_{n-1} \dots c_0, c_{-1} \dots c_{-m})_\beta$$
>
> In base β avrò β cifre possibili; per basi >10 si usano i caratteri come cifre aggiuntive.
>
> La **base 16** è spesso usata per rappresentare lunghe stringhe di numeri binari in modo compatto. (*indirizzi di memoria*)
>
> > cambio base
> >
> > # da base 2 a base 16:
> >
> > ad ogni gruppo di 4 cifre binarie corrisponde un simbolo esadecimale e viceversa.
> >
> > # Cambio base in generale
> >
> > in generale, per passare da una base β ad un'altra $\beta*$:

$$(N)_\beta = (I, F)_\beta = (?, ?)_{\beta*} = (c_{n-1} \dots c_0, c_{-1} \dots c_{-m})_\beta$$

base 2/16 $\rightarrow$ base 10

se $\beta = 2/16 \quad e \quad \beta* = 10$:
basta calcolare il valore di un numero intero tramite l'*espansione polinomiale*

base 10 $\rightarrow$ base 2/16

Si usa il **metodo di conversione iterativa**.

> metodo di conversione iterativa
>
> #esempio
>
> 1. convertire $(41,6875)_{10}$ in binario:
>
> **Cifre parte intera**
>
Quoziente intero	Resto	Cifra
41/2	20	1	$b_0 = 1$
20/2	10	0	$b_1 = 0$
10/2	5	0	$b_2 = 0$
5/2	2	1	$b_3 = 1$
2/2	1	0	$b_4 = 0$
1/2	0	1	$b_5 = 1$
>
> **Cifre parte frazionaria**
>
Risultato intero	Risultato frazionario	Cifra
0,6875*2	1	0,375	$b_{-1} = 1$
0,375*2	0	0,75	$b_{-2} = 0$
0,75*2	1	0,5	$b_{-3} = 1$
0,5*2	1	0	$b_{-4} = 1$
>
> N.B.: la conversione della parte frazionaria può richiedere un numero infinito di cifre. Devo fissare un numero di cifre significative dopo il quale la interrompo, se non termina prima.
>
> La rappresentazione cercata è quindi $(101001{,}1011)_2$
>
> 2. convertire $(41,6875)_{10}$ in esadecimale:
>
> **Cifre parte intera**
>
Quoziente intero	Resto	Cifra
41/16	2	9	$h_0 = 9$
2/16	0	2	$h_1 = 2$
>
> **Cifre parte frazionaria**
>
Risultato intero	Risultato frazionario	Cifra
0,6875*16	11	0	$h_{-1} = B$

base $\neq$ 10 $\rightarrow$ base $\neq$ 10

passare da $\beta \rightarrow 10 \rightarrow \beta*$

- per rappresentare un numero senza segno in una macchina digitale, si usa la rappresentazione in base 2.

Codice di Gray

> codice di Gray

- due configurazioni adiacenti differisconno per il valore di un solo bit (non accade nel codice binario).
- usato per codificare la posizione
  - ridurre sorgenti di errore nel converitre segnale analogico e digitale

Conversione di Codice: trascodifica

trascodifica

- il **codice interno** è NON ridondante per minimizzare numero bit da elaborare
- il **codice esterno** è
  - **ridondante**: semplificare generazione ed interpretazione delle informazioni
  - **standard**: per rendere possibile la connessione di macchine realizzate da costruttori diversi

Codice proprietario

Codice scelto da un costruttore per mettere in comunicazione macchine di sua produzione

Codice standard

Scelto da norme internazionali (**de iure**), o perchè tale macchina è molto utilizzata nel mercato (**de facto**)

#esempio

Calcolatrice

- codice ridondante 7 segmenti per visualizzare dati
- codice ridondante $1/N$ per introduzione dati e comandi
- codice binario per rappresentazione dei numeri interna

#esempio

Ascensore

- 4 piani, 5 tasti (anche quello dove sei già) $\implies$ ridondante

#finisci

Decoder

Encoder

Codice ASCII

> codice ASCII
>
> # ASCII a 7 bit
>
> - 128 caratteri (33 di controllo)
> - Primo srandare de iure per codifica binaria infromazioni
>
> # ASCII esteso: 8 e 16 biy
>
> - ascii 8 bit: includeva caratteri da lingue europee
> - Standard Unicode: 16 bit
>   scopo di codificare in binario con 2 byte simboli di tutte le lingue
>   (spoiler: non bastano)
> - Standard Unicode 2.0 : esteso, ora è a 21 bit

Unicode

> unicode
>
> - problema: architetture e linguaggi programmazione operano su gruppi di byte (8 bit).
>   Dunque 3 standard per mappare carattere unicode da 21 bit in una sequenza di byte **non ridondanti**.
>
> 1. UTF-32
>   usa 32 bit (**4 byte**) per ogni carattere → aggiunge 11 volte 0 a sinistra
> 2. UTF-16
>   usa **2 byte** per caratteri più comuni (63000) e **4 byte** per i restanti
> 3. UTF-8
>   **1 byte** per 128 ascii, **2 byte** per altri 1920, **3-4byte** per gli altri