Corso di Fondamenti di Informatica

Lezione 3

Nicola Capuano

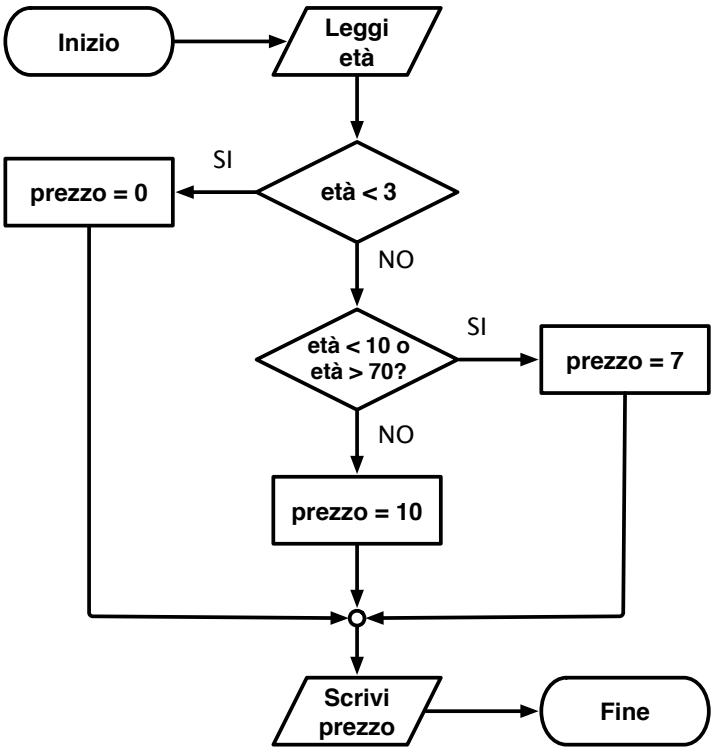
Dipartimento di Scienze Aziendali, Management & Innovation Systems

ncapuano@unisa.it

Esercizio per casa (soluzione 1)

Descrivere un algoritmo che calcola il prezzo del biglietto del cinema come segue:

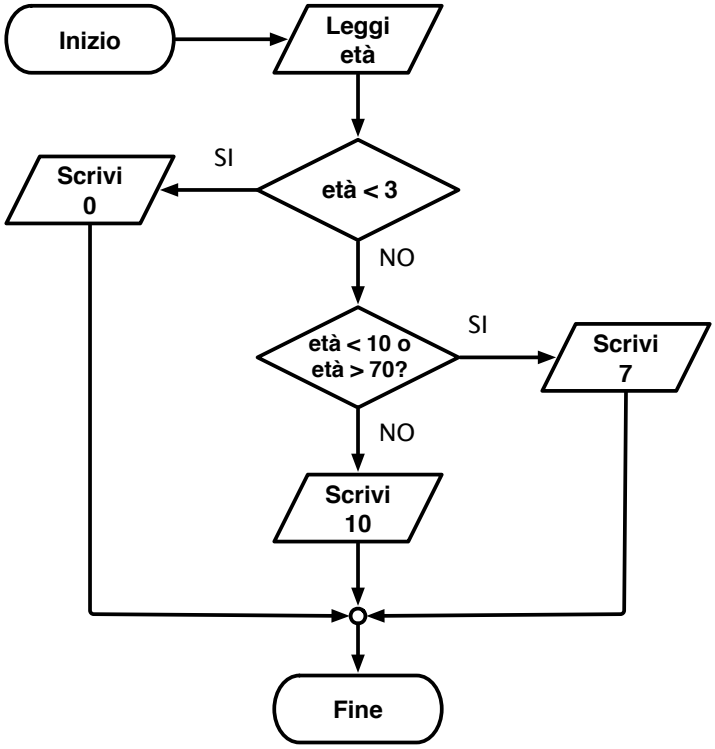
- Se lo spettatore ha meno di 3 anni l'ingresso è gratuito
- Altrimenti, se lo spettatore ha meno di 10 anni o più di 70 anni il prezzo è 7 €
- Altrimenti il prezzo è 10 €



Esercizio per casa (soluzione 2)

Descrivere un algoritmo che calcola il prezzo del biglietto del cinema come segue:

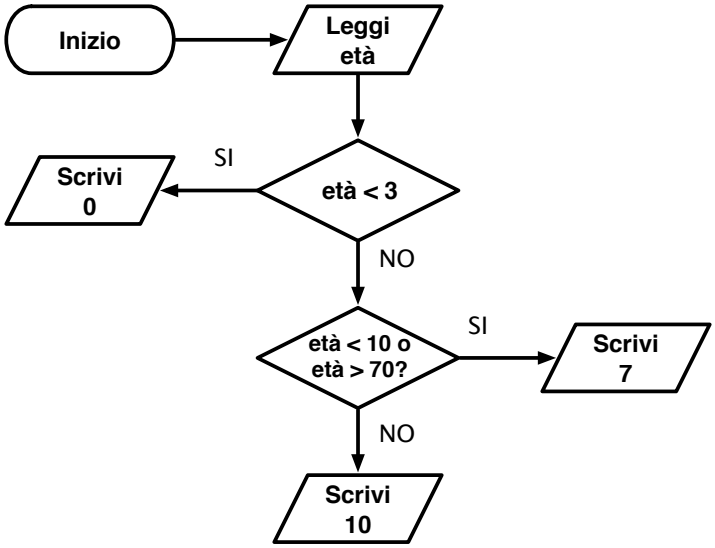
- Se lo spettatore ha meno di 3 anni l'ingresso è gratuito
- Altrimenti, se lo spettatore ha meno di 10 anni o più di 70 anni il prezzo è 7 €
- Altrimenti il prezzo è 10 €



Esercizio per casa (errore 1)

Descrivere un algoritmo che calcola il prezzo del biglietto del cinema come segue:

- Se lo spettatore ha meno di 3 anni l'ingresso è gratuito
- Altrimenti, se lo spettatore ha meno di 10 anni o più di 70 anni il prezzo è 7 €
- Altrimenti il prezzo è 10 €

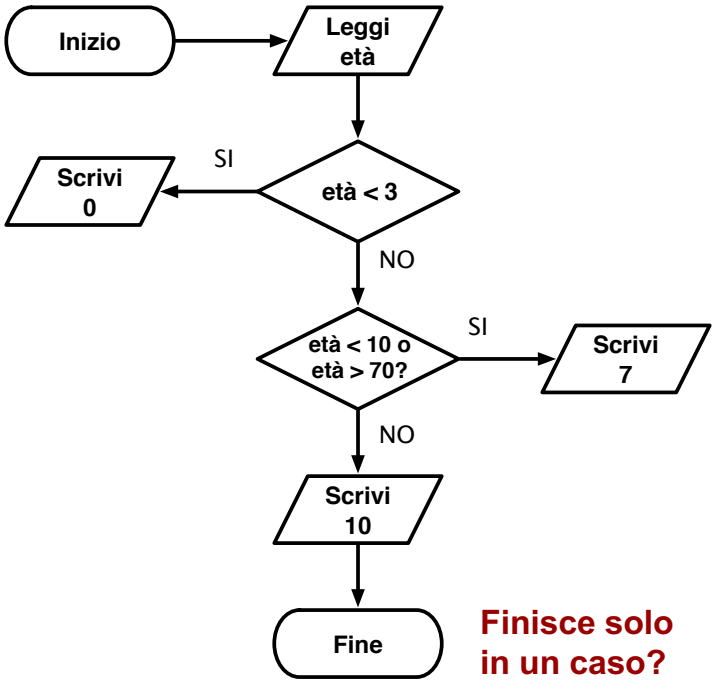


Quando finisce?

Esercizio per casa (errore 2)

Descrivere un algoritmo che calcola il prezzo del biglietto del cinema come segue:

- Se lo spettatore ha meno di 3 anni l'ingresso è gratuito
- Altrimenti, se lo spettatore ha meno di 10 anni o più di 70 anni il prezzo è 7 €
- Altrimenti il prezzo è 10 €

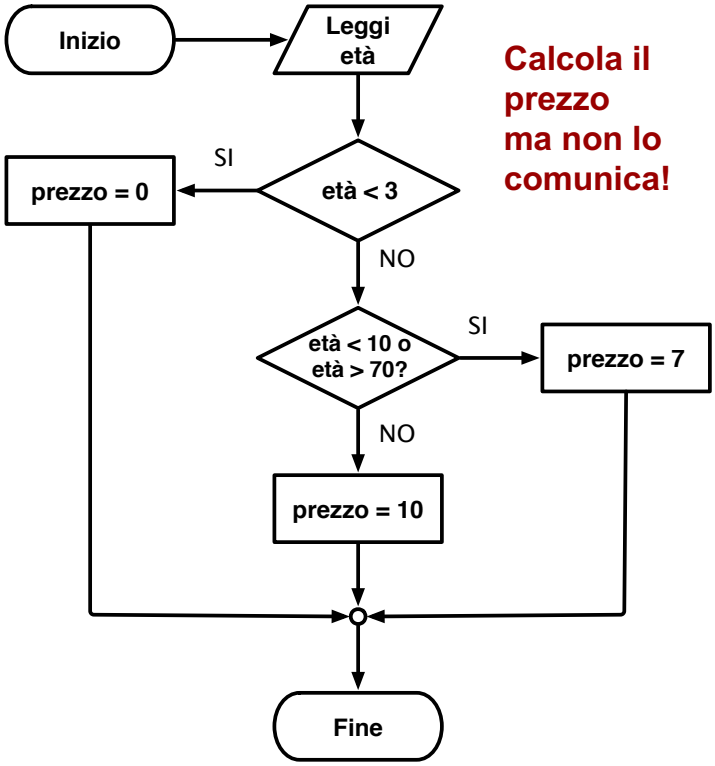


Finisce solo in un caso?

Esercizio per casa (errore 3)

Descrivere un algoritmo che calcola il prezzo del biglietto del cinema come segue:

- Se lo spettatore ha meno di 3 anni l'ingresso è gratuito
- Altrimenti, se lo spettatore ha meno di 10 anni o più di 70 anni il prezzo è 7 €
- Altrimenti il prezzo è 10 €

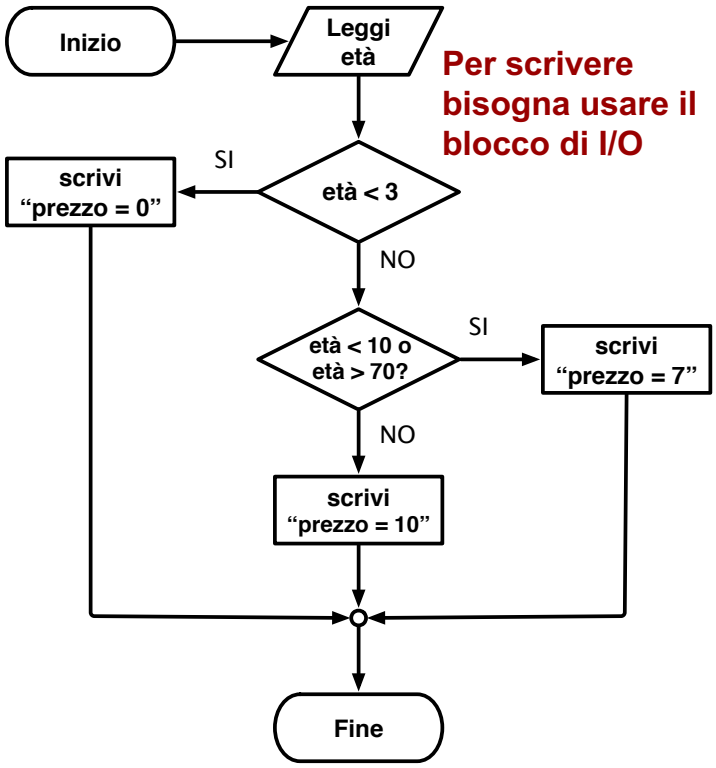


Calcola il prezzo ma non lo comunica!

Esercizio per casa (errore 4)

Descrivere un algoritmo che calcola il prezzo del biglietto del cinema come segue:

- Se lo spettatore ha meno di 3 anni l'ingresso è gratuito
- Altrimenti, se lo spettatore ha meno di 10 anni o più di 70 anni il prezzo è 7 €
- Altrimenti il prezzo è 10 €

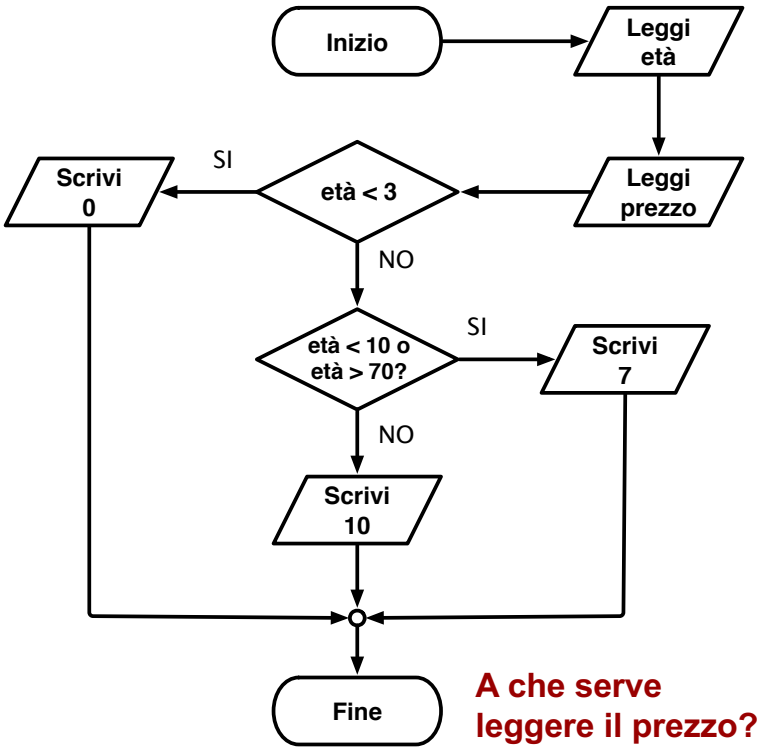


Per scrivere bisogna usare il blocco di I/O

Esercizio per casa (errore 5)

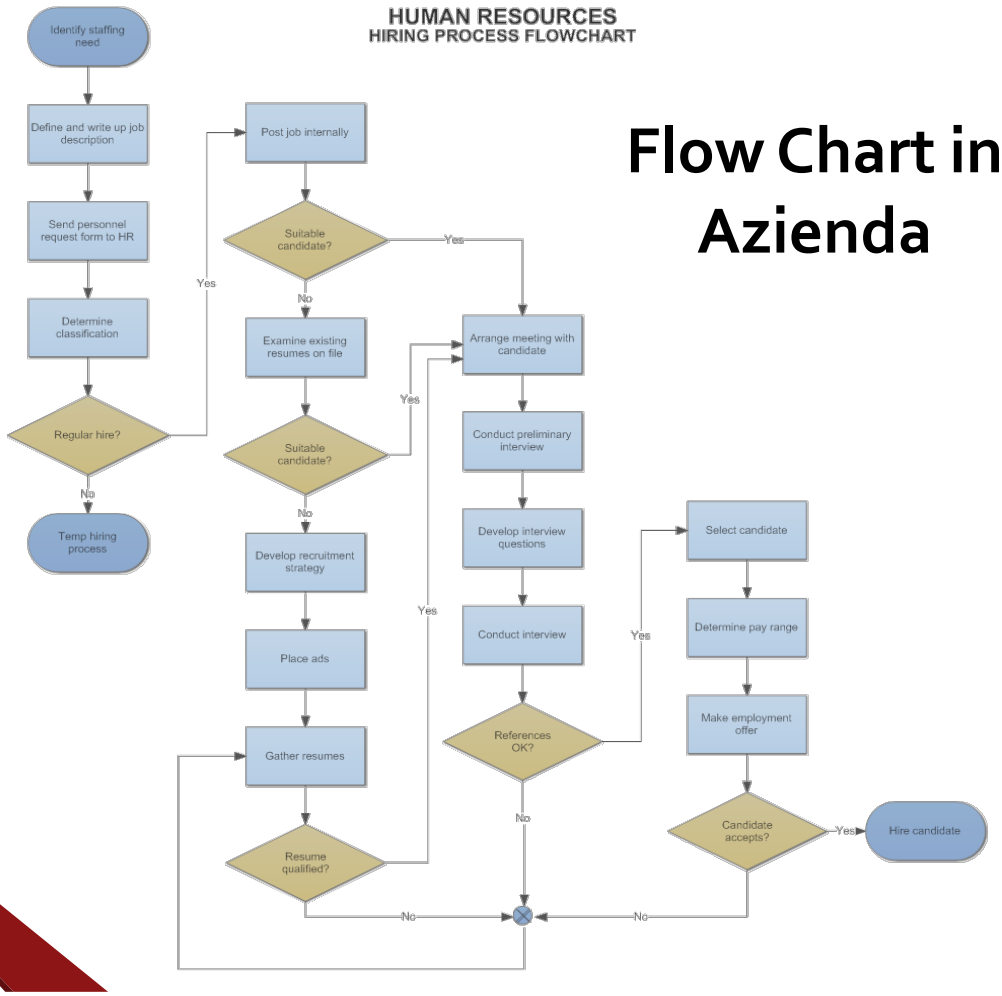
Descrivere un algoritmo che calcola il prezzo del biglietto del cinema come segue:

- Se lo spettatore ha meno di 3 anni l'ingresso è gratuito
- Altrimenti, se lo spettatore ha meno di 10 anni o più di 70 anni il prezzo è 7 €
- Altrimenti il prezzo è 10 €

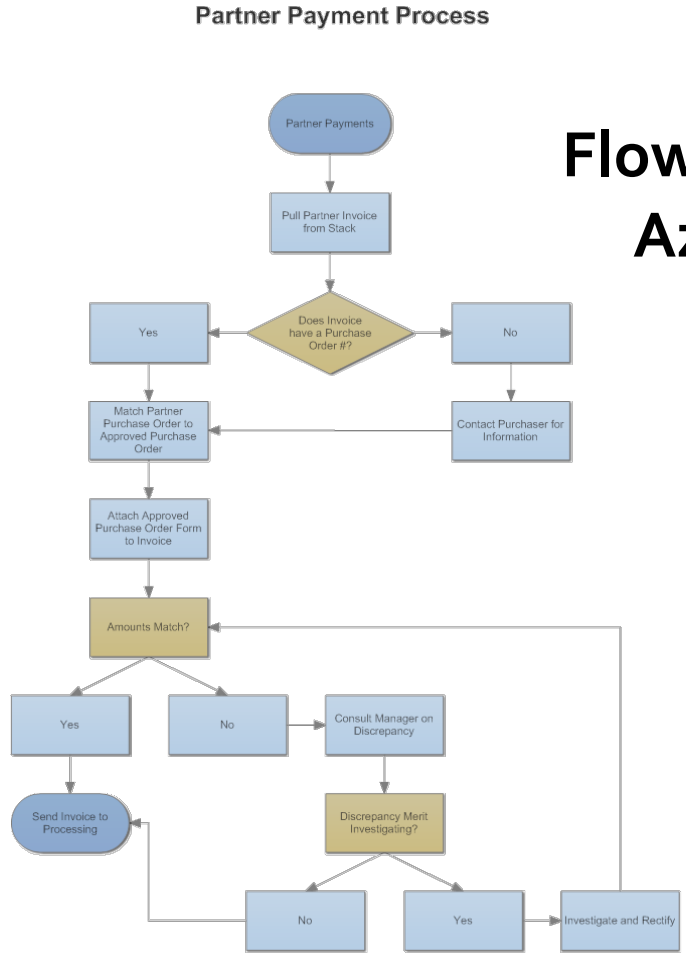


A che serve leggere il prezzo?

Flow Chart in Azienda



Flow Chart in Azienda

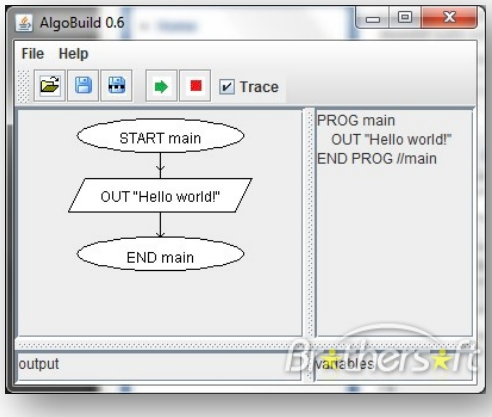


AlgoBuild

AlgoBuild è un software didattico per progettare algoritmi

- Presenta un ambiente visuale in cui disegnare flow-chart
- L'algoritmo viene visualizzato anche in forma di pseudocodice
- L'algoritmo può essere eseguito

https://algobuild.com/



Programma del Corso

Modulo 1 - Tecnologie dell'informazione e della comunicazione

- Introduzione alle ICT
- Rappresentazione Digitale dell'Informazione
- Rappresentazione Digitale dei Dati Multimediali
- Architettura Hardware di un Computer
- Software e Sistemi Operativi
- Reti di computer

Rappresentazione Digitale dell'Informazione

Parte 2: Codifica dell'Informazione

Bibliografia

- Par 2.4: Il Sistema Binario
- Par. 2.5: Bit e Byte
- Approfondimenti su queste slide

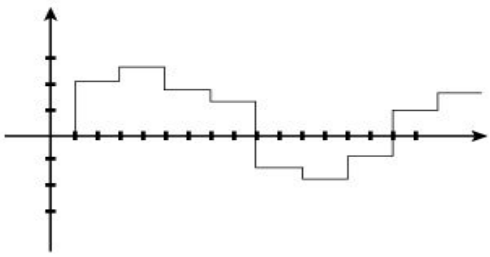
Segnali Digitali

I segnali digitali possono assumere solo un numero discreto di valori

- Possono essere **elaborati più facilmente** dei segnali analogici
- Possono essere **registrati in maniera più fedele e stabile** dei segnali analogici

Sono poco sensibili alle interferenze

- È più semplice identificare, senza commettere errore, uno tra un numero finito di valori che uno tra un numero infinito di valori



Digitalizzazione

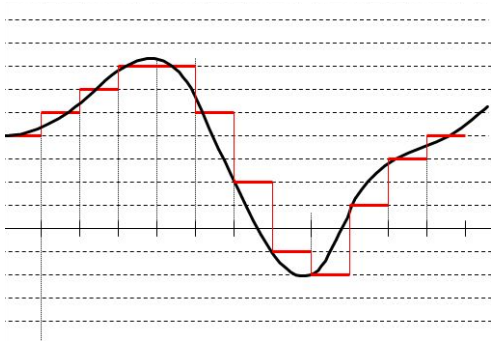
Il nostro è un mondo **ANALOGICO**

Il computer **elabora informazioni DIGITALI**

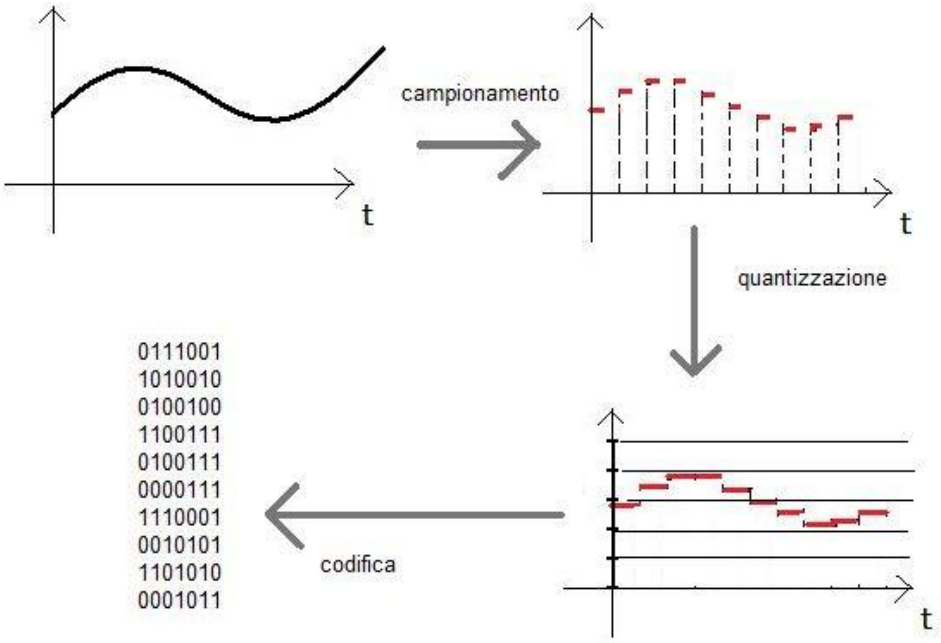
Tutti i dati (musica, film, foto, ecc.), per poter essere **trattati da un computer**, devono essere **TRASFORMATI**

- dal formato **ANALOGICO**
- a quello **DIGITALE**

Questa operazione è detta **DIGITALIZZAZIONE**



Digitalizzazione

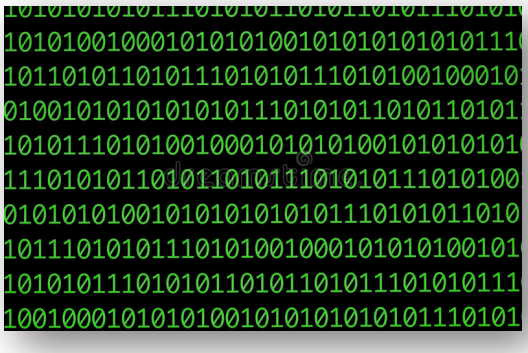


Informazione Binaria

In un computer i dispositivi elettrici, elettronici o meccanici, di norma, assumono solo uno tra **due stati**:

- Interruttore: aperto/chiuso
- Corrente in un filo: passa/non passa
- Polarità di un punto su un supporto magnetico: +/-
- Punto su un CD o DVD: riflette/non riflette la luce

Un **computer** è quindi in grado di trattare, in modo **diretto**, solo informazioni rappresentate in **binario.**



Informazione Binaria

Per rappresentare un'informazione binaria è sufficiente disporre di un alfabeto con **due simboli** a cui si associano i due valori dell'informazione binaria.

L'unità minima del linguaggio digitale è il **Bit**, termine che deriva dalla contrazione di (**BI**nary digi**T**)

Un bit può assumere solo due stati:

- Acceso (1)
- Spento (0)

Informazione Binaria

Esempio: si vuole "**rappresentare**" lo **stato di luminosità di una stanza ogni 10 minuti**, a partire dalle ore 15:00 e fino alle ore 15:30.

- La luce in una stanza può essere in uno solo di due possibili stati: **Spenta (0)** o **Accesa (1)**
- Così, se alla fine avremo scritto, **0111**, significa che:
  - Alle ore 15:00: la luce era spenta (0)
  - Alle ore 15:10: la luce era accesa (1)
  - Alle ore 15:20: la luce era accesa (1)
  - Alle ore 15:30: la luce era accesa (1)

Informazioni più "Complesse"

Quando l'informazione è più complessa della binaria?
Cioè, quando la scelta è **fra più di due alternative?**

- non è possibile usare altri simboli (2, 3, …) oltre 0 e 1 perché **il computer conosce solo l'alfabeto binario**!
- **Ma è possibile costruire "parole" concatenando più bit**

Esempio: rappresentare l'esito di una partita tra A e B

- se non può terminare con un pareggio (es eliminatoria diretta):
  0 vince A; 1 vince B
- se può terminare con un pareggio:
  00 pareggio; 10 vince A; 01 vince B

Informazioni più "Complesse"

Codificare l'informazione sul colore di un Semaforo

Codificare con 3 bit

- **100** → ROSSO ACCESO, Giallo spento, Verde spento
- **010** → Rosso spento, GIALLO ACCESO, Verde spento
- **001** → Rosso spento, Giallo spento, VERDE ACCESO

Codificare con 2 bits (più efficiente)

- **00** → ROSSO ACCESO, Giallo spento, Verde spento
- **01** → Rosso spento, GIALLO ACCESO, Verde spento
- **10** → Rosso spento, Giallo spento, VERDE ACCESO
- **11** → SEMAFORO GUASTO (messaggio "aggiuntivo")

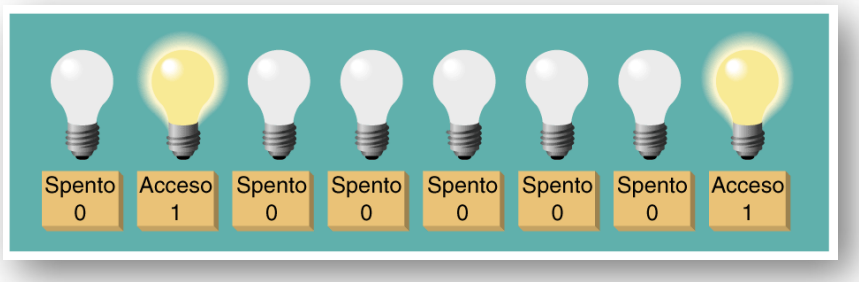
Informazioni più "Complesse"

Per trasmettere una **maggiore quantità di dati** si ricorre ad un'unità più grande detta **Byte**

Il Byte è costituito da 8 bit

- Con 1 byte si rappresentano **256 messaggi differenti.**
- Ogni messaggio è rappresentato da una **sequenza**

da **00000000** a **11111111**



Sequenze di Bit

Quanti messaggi (parole) si possono costruire con *k* bit?

- Passando **da *k*-1 bit a *k* bit** il numero di **parole raddoppia**
- Per costruire tutte le parole composte da *k* bit, basta aggiungere prima o e poi 1 a tutte le parole lunghe *k*-1 bit.
  - k=1 → 0, 1
  - k=2 → 0**0**, 0**1**, 1**0**, 1**1**
  - k=3 → 00**0**, 00**1**, 01**0**, 01**1**, 10**0**, 10**1**, 11**0**, 11**1**

Per cui, con *k* bit si possono costruire 2^k parole

- con **8** bit (1 byte) → **256** parole
- con **16** bit (2 bytes) → **65.536** parole
- con **32** bit (4 bytes) → **4.294.967.296** parole
- con **64** bit (8 bytes) → **18.446.744.073.709.552.000** parole

Unità di misura

Il byte, e i suoi multipli, sono utilizzati come di misura delle **capacità di memoria** di un computer.

Valore	Nome	Abbreviazione	Potenza
1	Byte o bit	1	2^0
1024	Kilobyte o kilobit	1kB o 1kb	2^{10}
1 048 576	Megabyte o megabit	1MB o 1Mb	2^{20}
1 073 741 824	Gigabyte o gigabit	1GB o 1Gb	2^{30}
1 099 511 627 776	Terabyte o terabit	1TB o 1Tb	2^{40}

Codifica di un Testo

Un Testo è una successione di caratteri

- **Ogni carattere appartiene a un insieme finito e ristretto di simboli** (lingue basate sull'alfabeto latino)

In un testo non ci sono solo le lettere dell'alfabeto, ma anche altri simboli quali:

- Simboli numerici
- Simboli di interpunzione
- Simboli speciali (**+ - & ^** …)
- Segnali di comando (“**Vai a capo**”, “**TAB**”, ecc..)

Codifica di un Testo

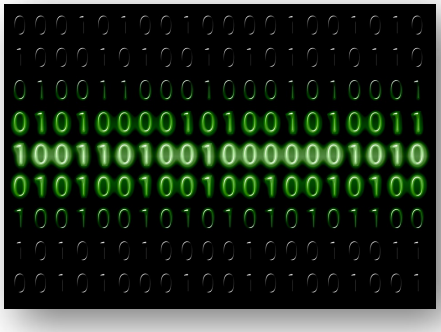
Per codificare un testo applichiamo il **metodo generale**

- **oggetto da codificare → successione di bit**

Per codificare i testi faremo corrispondere:

- **ad ogni carattere una diversa parola binaria**

Le parole binarie devono contenere un numero di bits sufficienti a coprire tutti i principali caratteri



Codifica ASCII

La **codifica più diffusa** nei computer è detta **ASCII** (si legge 'aski', American Standard Code for Information Interchange)

Inizialmente essa era **definita su 7 bit** e permetteva quindi di codificare 128 caratteri diversi (2^7=128)

Esempio:

1000001 (65) → A

1000010 (66) → B

1000011 (67) → C

Byte	Cod.	Char	Byte	Cod.	Char	Byte	Cod.	Char	Byte	Cod.	Char
00000000	0	Null	00100000	32	Spc	01000000	64	@	01100000	96	`
00000001	1	Start of heading	00100001	33	!	01000001	65	A	01100001	97	a
00000010	2	Start of text	00100010	34	"	01000010	66	B	01100010	98	b
00000011	3	End of text	00100011	35	#	01000011	67	C	01100011	99	c
00000100	4	End of transmit	00100100	36	$	01000100	68	D	01100100	100	d
00000101	5	Enquiry	00100101	37	%	01000101	69	E	01100101	101	e
00000110	6	Acknowledge	00100110	38	&	01000110	70	F	01100110	102	f
00000111	7	Audible bell	00100111	39	'	01000111	71	G	01100111	103	g
00001000	8	Backspace	00101000	40	(	01001000	72	H	01101000	104	h
00001001	9	Horizontal tab	00101001	41	)	01001001	73	I	01101001	105	i
00001010	10	Line feed	00101010	42	*	01001010	74	J	01101010	106	j
00001011	11	Vertical tab	00101011	43	+	01001011	75	K	01101011	107	k
00001100	12	Form Feed	00101100	44	,	01001100	76	L	01101100	108	l
00001101	13	Carriage return	00101101	45	-	01001101	77	M	01101101	109	m
00001110	14	Shift out	00101110	46	.	01001110	78	N	01101110	110	n
00001111	15	Shift in	00101111	47	/	01001111	79	O	01101111	111	o
00010000	16	Data link escape	00110000	48	0	01010000	80	P	01110000	112	p
00010001	17	Device control 1	00110001	49	1	01010001	81	Q	01110001	113	q
00010010	18	Device control 2	00110010	50	2	01010010	82	R	01110010	114	r
00010011	19	Device control 3	00110011	51	3	01010011	83	S	01110011	115	s
00010100	20	Device control 4	00110100	52	4	01010100	84	T	01110100	116	t
00010101	21	Neg. acknowledge	00110101	53	5	01010101	85	U	01110101	117	u
00010110	22	Synchronous idle	00110110	54	6	01010110	86	V	01110110	118	v
00010111	23	End trans. block	00110111	55	7	01010111	87	W	01110111	119	w
00011000	24	Cancel	00111000	56	8	01011000	88	X	01111000	120	x
00011001	25	End of medium	00111001	57	9	01011001	89	Y	01111001	121	y
00011010	26	Substitution	00111010	58	:	01011010	90	Z	01111010	122	z
00011011	27	Escape	00111011	59	;	01011011	91	[	01111011	123	{
00011100	28	File separator	00111100	60	<	01011100	92	\	01111100	124	|
00011101	29	Group separator	00111101	61	=	01011101	93	]	01111101	125	}
00011110	30	Record Separator	00111110	62	>	01011110	94	^	01111110	126	~
00011111	31	Unit separator	00111111	63	?	01011111	95	_	01111111	127	Del

Codifica ASCII

La codifica attualmente usata, **ISO Latin 1** (detta anche ASCII estesa), ha **aggiunto un bit 1 in prima posizione** alla ASCII su 7 bit e fa corrispondere

- **ad ogni carattere un numero binario di 8 bits**
- Con **8 bit, è possibile rappresentare 256 caratteri diversi** (2^8=256)

È stato possibile in questo modo includere vocali accentate, simboli semigrafici e altri simboli di uso meno comune

Byte	Cod.	Char	Byte	Cod.	Char	Byte	Cod.	Char	Byte	Cod.	Char
10000000	128	Ç	10100000	160	á	11000000	192	+	11100000	224	Ó
10000001	129	ü	10100001	161	í	11000001	193	-	11100001	225	ß
10000010	130	é	10100010	162	ó	11000010	194	-	11100010	226	Ô
10000011	131	â	10100011	163	ú	11000011	195	+	11100011	227	Ò
10000100	132	ä	10100100	164	ñ	11000100	196	-	11100100	228	õ
10000101	133	à	10100101	165	Ñ	11000101	197	+	11100101	229	Õ
10000110	134	å	10100110	166	ª	11000110	198	ã	11100110	230	µ
10000111	135	ç	10100111	167	º	11000111	199	Ã	11100111	231	þ
10001000	136	ê	10101000	168	¿	11001000	200	+	11101000	232	Þ
10001001	137	ë	10101001	169	®	11001001	201	+	11101001	233	Ú
10001010	138	è	10101010	170	¬	11001010	202	-	11101010	234	Û
10001011	139	ï	10101011	171	½	11001011	203	-	11101011	235	Ù
10001100	140	î	10101100	172	¼	11001100	204	¦	11101100	236	ý
10001101	141	ì	10101101	173	¡	11001101	205	-	11101101	237	Ý
10001110	142	Ä	10101110	174	«	11001110	206	+	11101110	238	¯
10001111	143	Å	10101111	175	»	11001111	207	¤	11101111	239	´
10010000	144	É	10110000	176	_	11010000	208	ð	11110000	240	-
10010001	145	æ	10110001	177	_	11010001	209	Ð	11110001	241	±
10010010	146	Æ	10110010	178	_	11010010	210	Ê	11110010	242	‗
10010011	147	ô	10110011	179	¦	11010011	211	Ë	11110011	243	¾
10010100	148	ö	10110100	180	¦	11010100	212	È	11110100	244	¶
10010101	149	ò	10110101	181	Á	11010101	213	i	11110101	245	§
10010110	150	û	10110110	182	Â	11010110	214	Í	11110110	246	÷
10010111	151	ù	10110111	183	À	11010111	215	Î	11110111	247	¸
10011000	152	ÿ	10111000	184	©	11011000	216	Ï	11111000	248	º
10011001	153	Ö	10111001	185	¦	11011001	217	+	11111001	249	¨
10011010	154	Ü	10111010	186	¦	11011010	218	+	11111010	250	·
10011011	155	ø	10111011	187	+	11011011	219	_	11111011	251	¹
10011100	156	£	10111100	188	+	11011100	220	_	11111100	252	³
10011101	157	Ø	10111101	189	¢	11011101	221	¦	11111101	253	²
10011110	158	×	10111110	190	¥	11011110	222	Ì	11111110	254	■
10011111	159	ƒ	10111111	191	+	11011111	223	_	11111111	255	

Codifica/Decodifica

(**Codifica**) Una stringa di caratteri è rappresentata da una successione di 8 bit:

C	a	r	o		A	l	d	o	,
01000011	01100001	01110010	01101111	00100000	01000001	01101100	01100100	01101111	00101100

(**Decodifica**) Data una sequenza di bits, il testo che essa codifica può essere così ricostruito:

- Si divide la sequenza in gruppi di 8 bits
- Si determina il carattere corrispondente ad ogni byte

Codifica UNICODE

ISO Latin 1 su 8 bit non è veramente 'universale'

In via di definizione la codifica **UNICODE**

- Incorpora i caratteri usati in quasi tutte le lingue vive e in alcune lingue morte, nonché:
  - simboli matematici e chimici
  - cartografici
  - l'alfabeto Braille
  - Ideogrammi, ecc.
- **Unicode originariamente era codifica a 16 bit**, capace di codificare **65.536 caratteri**
- **Attualmente Unicode è una codifica a 21 bit**, capace di codificare circa **un milione di caratteri**

Codifica dei Numeri

Tavola ASCII presenta tutte le 10 cifre (da 0 a 9).

Es: **0** → 00110**000**; **1**→00110**001**; **2**→ 00110**010**

Il numero 324 viene così rappresentato da 3 bytes:

00110011	0110010	00110100
3	2	4

Questa rappresentazione:

- **Non è efficiente**
- **Non è adatta per eseguire le operazioni aritmetiche**

Per i numeri si utilizza una codifica differente!

Sistemi di Numerazione Posizionali

Un sistema di numerazione si dice **posizionale** se **le cifre assumono valori diversi a seconda della loro posizione**

Il **sistema decimale** è un sistema posizionale

- Es. in **31** la cifra **1** rappresenta una **unità** **(valore = 1)**
- Es. in **13** la cifra **1** rappresenta una **decina** **(valore = 10)**

I sistemi posizionali consentono:

- di **rappresentare numeri grandi** con poche cifre
- di **svolgere calcoli** in modo semplice ed efficiente

Sistemi di Numerazione Posizionali

I sistemi di numerazione posizionale sono caratterizzati da una **base b** e un **alfabeto a**

Base (b)

- numero degli elementi dell'alfabeto
- **sistema decimale: b = 10.**

Alfabeto (a)

- insieme dei "simboli" (detti cifre) disponibili,
- A ogni cifra corrisponde un valore compreso tra o e (b-1)
- **sistema decimale: a = {0,1,2,3,4,5,6,7,8,9}**

Esempi:

(ottale) b = 8; a = {0,1,2,3,4,5,6,7}

(esadecimale) b = 16; a = {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}

Sistema di Numerazione Decimale

Rappresentazione posizionale in base 10:

- Il valore di un numero è pari alla somma di **potenze del 10** moltiplicato il valore del simbolo corrispondente

$$434 = 4 \cdot 10^2 + 3 \cdot 10^1 + 4 \cdot 10^0$$

- La parte frazionaria (a destra del simbolo separatore) si valuta con potenze negative:

$$4,34 = 4 \cdot 10^0 + 3 \cdot 10^{-1} + 4 \cdot 10^{-2}$$

La notazione posizionale può essere usata in qualunque altro sistema di numerazione (cioè con base diversa da 10)

In un **computer** sono diffuse le basi 2, 8 e 16.

Sistema di Numerazione Binario

Rappresentazione posizionale in base 2:

- Il valore di un numero è pari alla somma di **potenze del 2** moltiplicato il valore del simbolo corrispondente

$$(1101)_2 = (1\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0)_{10} = (13)_{10}$$

1 x 8 + 1 x 4 + 0 x 2 + 1 x 1 = 8 + 4 + 1 = 13

Per evitare ambiguità si usa il pedice per indicare la base

- La parte frazionaria (a destra del simbolo separatore) si valuta con potenze negative:

$$(1,101)_2 = (1\cdot 2^0 + 1\cdot 2^{-1} + 0\cdot 2^{-2} + 1\cdot 2^{-3})_{10} = (1,625)_{10}$$

1 x 1 + 1 x 0.5 + 0 x 0.25 + 1 x 0.125 = 1 + 0.5 + 0.125 = 1.625

Confronto tra Basi Diverse

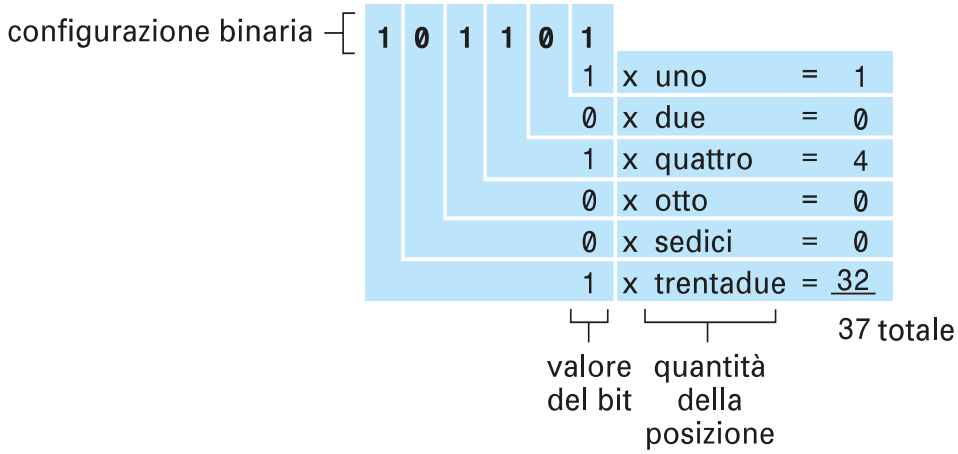
Base			
10	2	16	8
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	8	10
9	1001	9	11

Base			
10	2	16	8
10	1010	A	12
11	1011	B	13
12	1100	C	14
13	1101	D	15
14	1110	E	16
15	1111	F	17
16	10000	10	20
17	10001	11	21
18	10010	12	22
19	10011	13	23

Conversione Binario → Decimale

Moltiplicare ogni cifra del numero binario per un'opportuna potenza di 2 e sommare i prodotti

- per semplicità consideriamo solo numero interi



Conversione Binario → Decimale

Moltiplicare ogni cifra del numero binario per un'opportuna potenza di 2 e sommare i prodotti

- per semplicità consideriamo solo numero interi

$1001_2 = ?_{10}$

$= 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 =$

$= 1 \times 8 + 1 \times 1 = 8 + 1 = 9_{10}$

$101010_2 = ?_{10}$

$= 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 =$

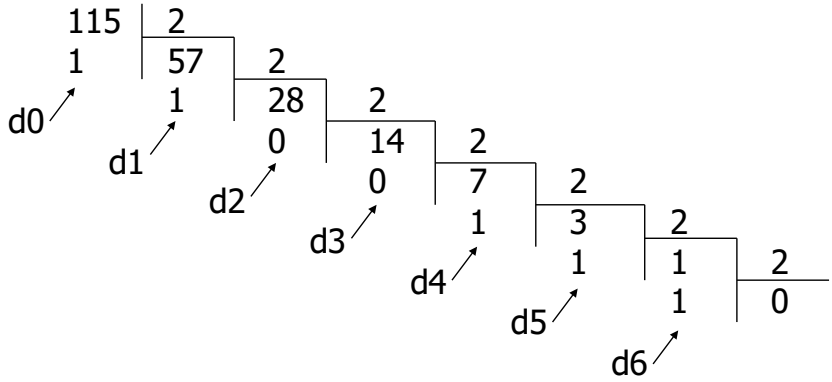
$= 1 \times 32 + 1 \times 8 + 1 \times 2 = 32 + 8 + 2 = 42_{10}$

Conversione Decimale → Binario

DECIMALE → BINARIO

- dividere per 2 il numero decimale e ripetere la divisione tra il quoziente e 2 fino a che il quoziente non diventa 0.
- I resti in ordine inverso corrispondono al numero trasformato

Esempio: $115_{10} = 1110011_2$

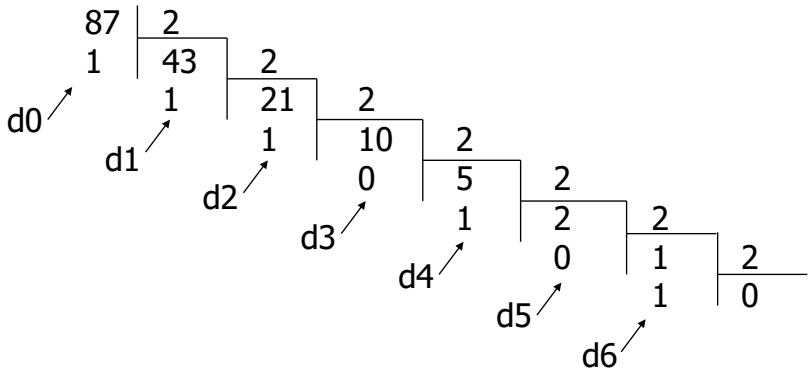


Conversione Decimale → Binario

DECIMALE → BINARIO

- dividere per 2 il numero decimale e ripetere la divisione tra il quoziente e 2 fino a che il quoziente non diventa 0.
- I resti in ordine inverso corrispondono al numero trasformato

$87_{10} = ?_2$



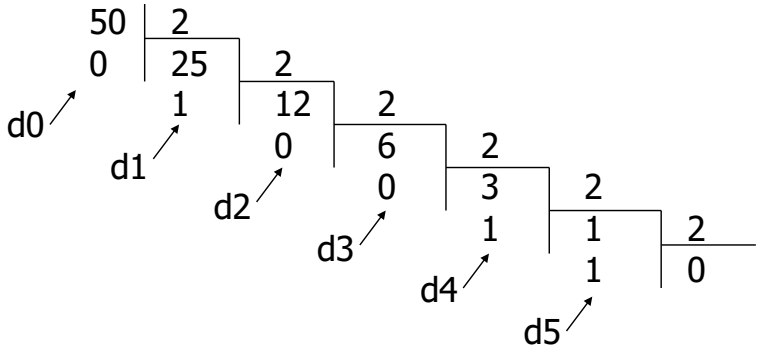
$= 1010111_2$

Conversione Decimale → Binario

DECIMALE → BINARIO

- **dividere per 2 il numero decimale e ripetere la divisione tra il quoziente e 2 fino a che il quoziente non diventa 0.**
- **I resti in ordine inverso corrispondono al numero trasformato**

50_{10} = $?_2$



= 110010_2

Somma tra Numeri Binari

Per eseguire la somma tra due numeri binari

- Incolonnare i numeri posizionandoli uno sotto l'altro partendo dalla prima cifra a destra
- Sommare i due numeri ricordando che:
  - **0 + 0 = 0**
  - **1 + 0 = 1**
  - **0 + 1 = 1**
  - **1 + 1 = 0 (con riporto di 1)**

1					riporti
1	0	1	0	0	+
		1	1	1	=
1	1	0	1	1	

Verifica:

10100_2 → 20_{10}

111_2 → 7_{10}

11011_2 → 27_{10}

Somma tra Numeri Binari

$110010_2 + 10001_2 = ?$

Verifica:

$50_{10} + 17_{10} = 67_{10}$

```
  11
 110010 +
  10001 =
-------
1000011
```

$111_2 + 101011_2 = ?$

Verifica:

$7_{10} + 43_{10} = 50_{10}$

```
   1111
    111 +
 101011 =
--------
 110010
```

Differenza tra Numeri Binari

Per eseguire la differenza tra due numeri binari

- Incolonnare i numeri posizionandoli uno sotto l'altro partendo dalla prima cifra a destra
- Sottrarre i due numeri ricordando che:
  - **0 – 0 = 0**
  - **1 – 1 = 0**
  - **1 – 0 = 1**
  - **0 – 1 = 1** (con il prestito di un **2** dalla colonna a sinistra)

```
1 1 1 1 0 –
  1 0 1 0 =
---------
1 0 1 0 0
```

Verifica:

10010_2 → 18_{10}

1101_2 → 13_{10}

101_2 → 5_{10}

Differenza tra Numeri Binari

$110010_2 - 10001_2 = ?$

Verifica:

$50_{10} - 17_{10} = 33_{10}$

```
   02
110010 -
 10001 =
------
100001
```

$101011_2 - 1111_2 = ?$

Verifica:

$43_{10} - 15_{10} = 28_{10}$

```
  12
0202
101011 -
  1111 =
------
011100
```

Esercizi per casa

Convertire in decimale i seguenti numeri binari:

1100_2 11111111_2

Convertire in binario i seguenti numero decimali:

37_{10} 67_{10}

Eseguire la seguente operazioni tra numeri binari:

- **$1010_2 + 11101_2$**
- **$10011110_2 + 11011011_2$**
- **$101010_2 - 100_2$**
- **$11011011_2 - 10011110_2$**

Bibliografia

- Par 2.4: Il Sistema Binario
- Par. 2.5: Bit e Byte
- Approfondimenti su queste slide