

```

+/-/*//%/*
abs(x)/round(x)/round(x,u)/min(x1,...,xn)/max(x1,xn)
from math import * /import math
(cos,sin,tan,acos,asin,atan,radians,degrees,exp,log,log(x1b),
log10,pow(x,u),sqrt(x))
assegnazione: =

```

assegnazione composta: i+=1	i = i + 1	j = 3	i = i/3
i*=2	i = i * 2	i**4 = 4	i = i ** 4
i-=1	i = i - 1	i % 2 = 2	i = i % 2

EDITOR → menu FILE → NEW file → salvataggio → Run file

INPUT() / INT() / FLOAT() / PRINT()

escape characters: \n, \t, \b, \\", \\", \"

↳ continuazione riga se si va a capo

nome variabile = 'stringa'

nome variabile[numeroposizione carattere] = 'carattere' s[0] = 's'

funzioni: len(), str(), ord(''), chr()

metodi: lower(), upper(), count(), find(), rfind(), replace()

slicing s[i:j], s[:j] ⇒ i=0, s[i:] ⇒ j = len(s), s[i:j] ⇒ si prende i carattere ogni n

PIPPO ie 1° carattere può avere indice 0 oppure indice -len(s)

>, <, >=, <=, ==, != operatori Booleani

not ha prec su and ha prec su or

in

metodi: isalpha(), isdecimal(), islower(), isupper()

IF if condizione:	IF/ELSE if cond:	ELIF if cond:
istruzione/i	ANNIDATE	ISTR
else:	if cond:	else:
istruzione/i	ISTR	if cond:

WHILE while condizione:

 istruzione 1

 ...

 istruzione n

scenaria di base (while):

1. inizializzazione variabile di controllo (i)

2. specifica della condizione

3. definizione dell'istruzione

4. incremento var di controllo (i+=1)

scenaria else con sentinella:

i = valore che permette l'escursione es. 0

while i != valore sentinella:

 istruzione/i i = input() } se vuo mettere un
 if i != valsentinella nuovo i

scenaria con accumulatore:

st = sentinella (diversa da 0)

a = accumulatore

while st != valore sentinella

 s = input()

 if condizione:

 a = accumulo

 print(a) → parziale nel ciclo

 print(a) → totale alla fine

⇒ if cond:
 ISTR
 elif cond:
 ISTR
 else:
 ISTR.
 stessa condiz.

else:
 ISTRUZ

 if cond:
 ISTRUZ

 else:
 ISTRUZ

funzione range(), range(u) = $0 \dots u$, range(x,y) = $x \dots y$
 puoi mettere solo int, solo stringhe
 si puo' creare un "range" con lista: $x = [val_1, val_2, 'stringa']$
 ma anche valori misti

clicci ammiciati: for i in range(x,u):
 for j in range(z,w):
 ISTRUZIONE
 ISTRUZIONE

istruzioni break e continue.
 conclude un'istruzione iterativa, ed è di norma usata all'interno di un'ISTRUZIONE CONDIZIONALE (if) modificata in un'ISTRUZIONE ITERATIVA

CONTINUE.
 conclude l'esecuzione di UNA SINGOLA ITERAZIONE, cioè interrompe l'esecuzione dell'iterazione in corso e sposta le controlli all'inizio del ciclo stesso

si possono usare allo stesso modo le while e le for

def nomefunzione(parametri): → la DEFINIZIONE non comporta direttamente l'ESECUZIONE
 istruzione/i ↗ POSSONO NON ESSERE PARAMETRI
 return risultato → PUO' NON ESSERE RETURN
 ○ POSSONO ESSERE + DI 1
 quando si esegue una funzione si usano variabili globali mentre lo programma principale non ha accesso alle variabili locali della funzione

è meglio che i dati di input della funzione siano tutti compresi nei parametri delle funzione.
 il risultato va restituito con return, non con print()

sarà funzione su file nella stessa directory del file programma
 le nome del file sarà le nome del modulo

funzione tester - file (from tester)

parametri optionali → devono essere gli ultimi nella lista dei parametri

→ se sono +di 1 e specifichi io penultimo, devi giustificare anche quello della sua dx (+ o -)

Modelli:

1. von Neumann
2. rappr. caratteri
3. rappr. positivi
4. rappr. complemento
5. rappr. reali
6. logica proposizionale

1. von neumann:
calcolatori elettronici
macchine meccaniche per il calcolo

1800 - Babbage: differential & analytic machine

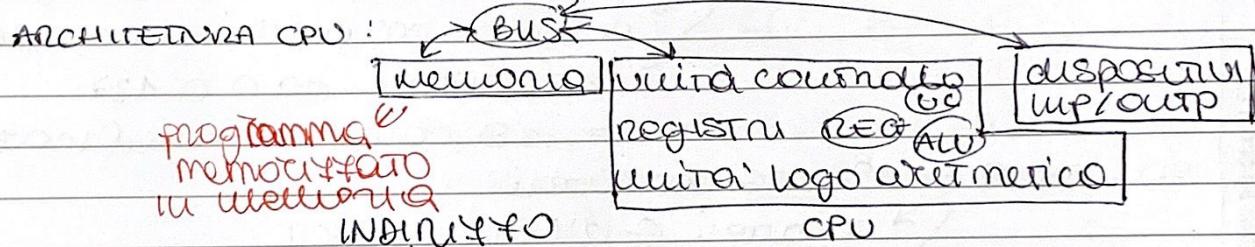
1800 - Ada Byron: interesse x scienze matematiche
e per macchine di Babbage

1940 ~ - nate macchine elettroniche
Zuse (Z1)

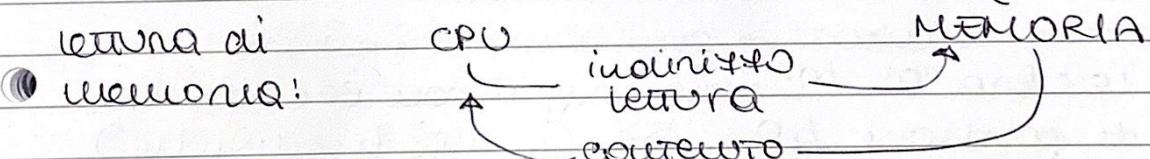
Turing (Colossus)

Eckert & Mauchly (ENIAC)

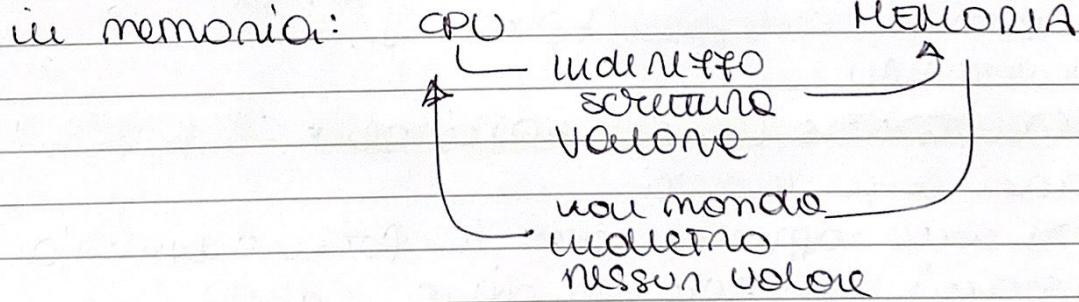
Von Neumann (architettura CPU)



MEMORIA: celle numerate, ogni cella può contenere un numero (+ HAR, MDR)
(in memoria → CONTENUTO)



scrittura



registri della CPU: IR - ISTRUZIONE register

PC - programma counter
memorizza indirizzo di inizio delle pross. istruz. da eseguire
registri dati - dati utilizzati x calcoli

ESECUZIONE ISTRUZIONE

1. Fetch: informaz in PC memorizzata in IR
2. Decode: unità di controllo decodifica l'ISTRUZ
3. Exec: si esegue con usi vari delle componenti
si aggiorna il PC x prox ISTRUZ
4. se prox ISTRUZ è HALT → si ferma.

L'evoluzione della macchina

legge di Moore: # Transistor per mm

2^ legge: costi ↑, produttori ↓

per proprietarie: + CPU su 1! chip

2. RAPPRESENTAZIONE CARATTERI, STRINGHE, ALTRI DATI

codifica ASCII (7 bit):
e' possibile usare 8 col 1°=0
ci sono caratteri di controllo
128 posizioni: da 0 a 127
= 128 caratteri codificati

ISO - 8859: $2^8 = 256$ caratteri
caratteri 0-127 come ASCII
128-159 non utilizzati

UNICODE UTF-8: È unita' da 8 bit
0-127 come ASCII

STRINGHE: seq. di caratteri → ogni carattere è un numero

CIFRE: le cifre '0'... '9' sono rapp. con le codifiche
di posizione 48...57 (in tutte le codifiche)

SUONI

COLORI: g. rosso, verde, blu } pixel } APPROX

IMMAGINI VETTORIALI

3. RAPPRESENTAZIONE NUMERI NATURALI (su piattaforma si ottiene rapp. positivi)

tutti i dati sono numeri

i numeri sono rappresentati in forma binaria

altro sistema: decimale: 10 cifre (0-9)

8 cifre (0-7)

esadecimale: 16 cifre (0-F) ~~0000~~

binario: 2 cifre (0,1)

RAPPRESENTAZ. POSIZIONALI

dato $b' = b$ e $b^0 = 1$ la regola generale e'

$$c_k c_{k-1} \dots c_1 c_0 = c_k \times b^k + c_{k-1} \times b^{k-1} + \dots + c_1 \times b^1 + c_0 \times b^0$$

in una base b qualsiasi

BINARIO → DECIMALE

$$(10110)_2 = \begin{array}{r} 1 \ 0 \ 1 \ 1 \ 0 \\ 16 \ 8 \ 4 \ 2 \ 1 \end{array}$$

$$\begin{aligned} &= 1 \times 16 + 0 \times 8 + 1 \times 4 + 1 \times 2 + 0 \times 1 \\ &= 16 + 4 + 2 \\ &= (22)_{10} \end{aligned}$$

OCTALE → DECIMALE

$$(42403)_8 = \begin{array}{r} 4 \ 2 \ 4 \ 0 \ 3 \\ 4096 \ 512 \ 64 \ 8 \ 1 \\ 8^4 \ 8^3 \ 8^2 \ 8^1 \ 8^0 \end{array}$$

$$\begin{aligned} &= 4 \times 4096 + 2 \times 512 + 4 \times 64 + 0 \times 8 + 3 \times 1 \\ &= 16384 + 1024 + 256 + 0 + 3 \\ &= (17859)_{10} \end{aligned}$$

DECIMALE → BINARIO (vole DECIMALE → base b)

$$\underbrace{(319)}_N {}_{10} \rightarrow \underbrace{(?)}_5 {}_2 \rightarrow \text{nuova base}$$

$$319/2 \quad 189 \quad r: 1 = c_0$$

$$159/2 \quad 49 \quad r: 1 = c_1$$

$$42/2 \quad 39 \quad r: 1 = c_2$$

$$39/2 \quad 19 \quad r: 1 = c_3$$

$$19/2 \quad 9 \quad r: 1 = c_4$$

$$9/2 \quad 4 \quad r: 1 = c_5$$

$$4/2 \quad 2 \quad r: 0 = c_6$$

$$2/2 \quad 1 \quad r: 0 = c_7$$

$$\boxed{1/2} \quad 0 \quad r: 1 = c_8$$

ricorda di farlo

$$(319)_{10} = (100111111)_2$$

si parla da destra!

DECINALE \rightarrow OTTALE

$$(319)_{10} = (?)_8$$

$$\begin{array}{r} 319/8 \quad 39 \quad \text{resto} = 7 \\ 39/8 \quad 4 \quad \text{resto} = 7 \\ 4/8 \quad 0 \quad \text{resto} = 4 \end{array}$$

$= (319)_{10} = (477)_8$

TI FERMI SOLO SE QUOTIENTE e' = 0 !!!

$\text{BASE}_1 \neq 10 \rightarrow \text{BASE}_2 \neq 10$

$$(u)_{B1} \rightarrow (u)_{10} \rightarrow (u)_{10} \rightarrow (u)_{B2}$$

BINARIO \rightarrow OTTALE

$$(11010111001)_2 = \underbrace{011}_{3} \underbrace{010}_{2} \underbrace{111}_{4} \underbrace{001}_{1}$$

$$= (3241)_8$$

AGGIUNGO ZERI DAVANTI PER OTTENERE TERZINE!

OTTALE \rightarrow BINARIO

$$(52142)_8 = \begin{matrix} 5 & 2 & 1 & 4 & 2 \\ || & || & || & || & || \end{matrix}$$

$$= 101 \ 010 \ 001 \ 111 \ 010$$

$$= (101010001111010)_2$$

BINARIO \rightarrow ESADECIMALE

$$(10101111001)_2 = \begin{matrix} 10101 \\ 0000 \times \text{ottenere 4 line} \\ 0101 \ 0111 \ 1001 \\ 5 \quad 4 \quad 9 \end{matrix}$$

$$= (5A9)_{16}$$

ESADECIMALE \rightarrow BINARIO

$$(5A9)_{16} = \begin{matrix} 5 & A & 9 \\ 0101 & 1010 & 1001 \\ 0111 & 1001 & 0001 \end{matrix}$$

$$= (101010010001)_2$$

• NUMERO = iè suo valore
NUMERALE = la sua rappresentazione

somma

moltiplicazione

su piattaforma!
rappr. complemento

4. Rappresentazione numeri interi con segno

num interi con segno $\xrightarrow{\text{BIT DI SEGNO}} \text{COMPLEMENTAZIONE ALLA BASE}$ (2)

(1) 1° bit per segno: 0: pos, 1: neg

es. 4 bit: 0001 0: pos | 0001: modulo = 1 $\rightarrow +1$

0000 0: pos | 0000: modulo = 0 $\rightarrow +0$ {?

1000 1: neg | 0000: modulo = 0 $\rightarrow -0$ {?

per la somma occorre verificare il segno dei 2 addendi
si rischia overflow/underflow

(2) COMPLEMENTO A 2 (# fisso di bit, es. 4)

positivi: si rappresentano come uelle modalita'
con modulo e segno. 4 bit totali, 1 per il segno,
3 per il valore assoluto.

(0000 - 0111 \rightarrow il max è 7)

negativi: 1° bit vale 1, è il segno negativo

gli altri si scrivono in base

a alla COMPLEMENTAZIONE: $(\min = 1000 = -8)$

1 - si invertono tutti i bit

2 - si somma 1

1 - si lasciano invariati tutti i bit
finché al 1° 1 inclusivo

2 - si invertono i restanti

COMPLEMENTO A 3 BIT

lo zero ha 11 rappresentazione (000)

ma ad es non c'è possibile rappresentare

+4 (-4 complementato 100 $\rightarrow 100 = -4$)

utilità: $\ominus 5$ preudo +5, convertito in binario = 0101
 complemento = 1011 $\rightarrow -5$
~~1100~~ ~~complemento~~
~~dal binario al negativo complemento =~~
~~= 0100 che è 4. Il numero originale~~
~~è quindi = -4~~

cioè

hai:

numero dec \rightarrow preudi il suo val. ass. \rightarrow converti in bin complement.

negat.

numero in
complem. a 2
negativo \rightarrow complemento \rightarrow converti
in decimale

e' uguale al valore
assoluto del numero
originale, ma il numero
depnominato avrà segno -

numero in
complem. a 2
negativo \rightarrow converti moltiplicando
la 1^a cifra (asx) per
-8 invece che 8

$$\rightarrow \text{es. } (1100)_2 = 1(-8) + 1(4) + 0(2) + 0(1) = -8 + 4 = (-4)_{10}$$

• operazioni su complemento a 2:

somma: si sommano i 2 numeri come fossero positivi, nella somma si include il bit di segno come se facesse parte del numero

sottrazione: si complementa il 2° numero e si somma al 1°

es somma $0001 +$ es. sottraz.

$$\begin{array}{r} \underline{1111} \\ - \underline{1000} \\ \hline \end{array}$$

↓ ↓

$0001 - 0101 = 1011$

$c(0101) = 1011$

$\Rightarrow 0001 + 1011 =$

elimina usciato

• max e min numeri rappresentabili con n bit

$$TOT = 2^n \text{ valori}$$

2^{n-1} usati per valori ≥ 0

$\rightarrow 2^{n-1} - 1$ è max

$\rightarrow -2^{n-1}$ è minimo

si va quindi da -2^{n-1} a $2^{n-1} - 1$

perché c'è 0

• numeri in PYTHON

x rapp. 1 intero usa 1 # arbitrario di
locazione di memoria, ognuna da 32 bits,
ognuna x rappresentare 1 intero
scritto, in binario e senza segno, su 30 bit
(segno contenuto a parte sui bit specifici)

5. rappresentazioni dei numeri reali

$$0, c_1 c_2 c_3 \dots = \underbrace{c_1 \times b^{-1} + c_2 \times b^{-2} + c_3 \times b^{-3} + \dots}_{\text{dopo lo } 0,}$$

$$(0,645)_8 = 6 \times 8^{-1} + 4 \times 8^{-2} + 5 \times 8^{-3} = \frac{6}{8} + \frac{4}{8 \times 8} + \frac{5}{8 \times 8 \times 8}$$

$$= 0, \underline{\underline{622265625}}$$

BINARIO → DECIMALE di $(110,101)_2$

$$\begin{array}{r} 1 & 1 & 0 \\ \times 2^2 & \times 2^1 & \times 2^0 \\ \hline = 4 & 2 & 0 \\ = 6 & & \end{array} \quad \begin{array}{r} 1 & 0 & 1 \\ \times 2^{-1} & \times 2^{-2} & \times 2^{-3} \\ \hline 0.5 & 0 & 0.125 \\ 6.25 & & \end{array}$$

BASE 6

~~DECIMALE~~ → **DECIMALE** di $(32,3)_6$

$$\begin{array}{r} 3 & 2 & 3 \\ \times 6^1 & \times 6^0 & \times 6^{-1} \\ \hline = 18 & 2 & \frac{3}{6} \\ = 20, & 5 & \end{array}$$

BINARIO

~~DECIMALE~~ → **ESADECANALE** di $(110,101)_2$

$$\begin{array}{r} 0 \ 1 \ 1 \ 0 \quad , \quad 1 \ 0 \ 1 \ 0 \\ \downarrow \qquad \qquad \qquad \downarrow \\ \text{gruppo} \quad , \quad \text{gruppo} \\ 6 \qquad \qquad \qquad A \\ \text{a sinistra} \quad \text{a destra} \\ \text{ultimo} \quad \text{primo} \\ \text{bit} \end{array}$$

→ poiché le due basi, 2 e 16

sono 1 potenza dell'altro

$$2^4 = 16 \quad 4 = \# \text{ valori in 1 blocco}$$

DECIMALE \rightarrow BASE B (Quale è detto dei termini)

es decimale \rightarrow base 2 (binario)

$$(0,6875)_{10} \rightarrow (?)_2$$

$$0,6875 \cdot 2 = 1,375$$

$$0,375 \cdot 2 = 0,75$$

$$0,75 \cdot 2 = 1,5$$

$$0,5 \cdot 2 = 1$$

non si va in
senso opposto
QUINDI:

0,

↓

0, 1 0 1 1 1

- rapporto generale x la finitività del numero:
se la base di avendo contiene tutti i numeri primi di quella di fattura allora il numero frazionario con cifre finite si converte sempre in uno con cifre finite.

- si dà trasformare in base b. cerchi o, c, c₂, c₃...

$$\text{tc } u = c_1 \times b^{-1} + c_2 \times b^{-2} + c_3 \times b^{-3} \Leftrightarrow$$

$$u \times b = c_1 \times b^0 + c_2 \times b^{-1} + c_3 \times b^{-2} + \dots = c_1 + 0, c_2, c_3$$

cioè: la parte intera di $u \times b$ è la 1^a cifra frazione
di u in base b. Torna questa da $u \times b$ si può
moltiplicare ancora per b per ottenere la 2^a
cifra etc.

- alla virgola fissa

virgo la mobile

prefissato di bit
totali,
suolo visi in
bit per parte intera
e bit per parte
frazionaria

prefissato di bit
totali ma la
posizione della
virgola all'intero
di essi può
essere

+ preciso
- gamma

numero \rightarrow mantissa

(Valore senza virgola)

+ gamma
- preciso esponente (posizione delle
virgole)

$$351 \# 0 \rightarrow 0,351, 1023 \# 2 \rightarrow 10,23, 01023 \# 0000$$

$$21 \# 5 \rightarrow 21000,0$$

nel sistema con un solo mobile la posizione dello virpolo indica di quanto va moltiplicato il numero, considerato come valore frazionario (0,n)

$$u = 1023 \# 2 \rightarrow 0, 1023 \times 10^2 = 10,23$$

cioè $u \# p \rightarrow 0, u \times 10^p$ puo' essere un altro bene

formula: $u = m \times b^e$

numero = mantissa \times base^{esponente}

es quanto vale il numero decimale che ha
mantissa 0,5235 ed esponente -2

$$u = \text{mantissa} \times b^{-2}$$

vuo' dire

base

$$u = 0,5235 \times 10^{-2} = \frac{0,5235}{100} = 0,005235$$

calcoli in un solo mobile: somma

1) si modifica il num con esp < in modo che
abbia lo stesso dell'altro

2) si sommano le mantisse

3) se il risultato è ≥ 1 , si divide x 10 e

si aumenta l'esp. di 1

in generale x aumentare l'esp di x, basta appun-
gere x zero davanti alla mantissa (dopo virpolo)

in base generica: muovere lo virpolo:

$$\underline{0, \text{mantissa} \times b^{\text{esponente}}}$$

in base
 b

quasi
base

in base
 b

FORMATO IEEE 754: rapp. i numeri in binario con 64 bit
(di segno / di esponente / 52 di mantissa)

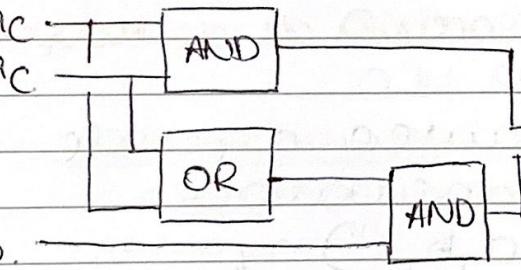
6. logica proposizionale / logica e calcolo proposizionale

$00000 = 0$	{	somma cifra singola:		
$00001 = 1$		rep.	$1^{\text{a}}C$	$2^{\text{a}}C$
$00010 = 2$			0	0
$00011 = 3$			0	1
$00100 = 4$			0	1
$00101 = 5$			0	1
$00110 = 6$			0	1
$00111 = 7$			0	1
$01000 = 8$			0	1
$01001 = 9$			0	1
$01010 = 10$			0	1
$01011 = 11$			1	0
$01100 = 12$			1	0
$01101 = 13$			1	0
$01110 = 14$			1	0
$01111 = 15$			1	1
base 2			1	1
base 10			1	1

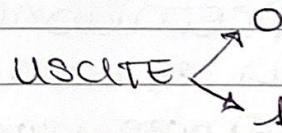
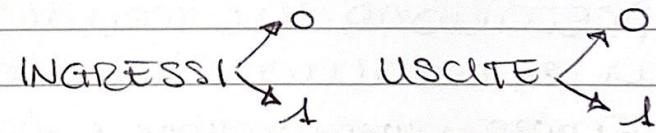
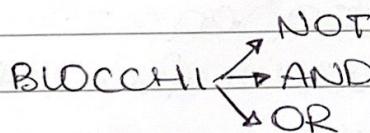
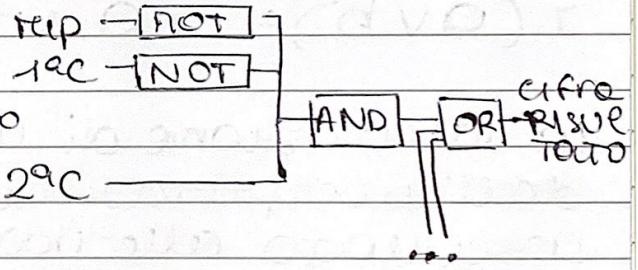
RIPORTO = 1 se $1^{\text{a}}C = 2^{\text{a}}C = 1$

2) vecchio ripporto = 1
e $1^{\text{a}}C \vee 2^{\text{a}}C = 1$

CIFRA DEL RIPORTO



CIFRA DEL RISULTATO



NOT : $\text{RIPORTO} \rightarrow \text{NOT} \rightarrow \text{USCITA}$ cioè $\begin{array}{c} \text{UP} \\ 0 \\ 1 \end{array}$ $\begin{array}{c} \text{USC} \\ 1 \\ 0 \end{array}$
: anche !, \neg

AND: $\begin{array}{c} 1^{\text{a}}C \\ 2^{\text{a}}C \end{array} \rightarrow \text{AND} \rightarrow \text{USCITA}$ cioè $\begin{array}{c} 1^{\text{a}}C \\ 0 \\ 0 \\ 0 \end{array} \begin{array}{c} 2^{\text{a}}C \\ 1 \\ 0 \\ 1 \end{array} \begin{array}{c} \text{USCITA} \\ 0 \\ 1 \\ 0 \end{array}$
: anche \wedge , \wedge

OR: $\begin{array}{c} 1^{\text{a}}C \\ 2^{\text{a}}C \end{array} \rightarrow \text{OR} \rightarrow \text{USCITA}$ cioè $\begin{array}{c} 1^{\text{a}}C \\ 0 \\ 0 \\ 1 \end{array} \begin{array}{c} 2^{\text{a}}C \\ 1 \\ 0 \\ 1 \end{array} \begin{array}{c} \text{USCITA} \\ 1 \\ 1 \\ 1 \end{array}$
: anche \vee , \vee

accini logici (ingressi, uscita, \neg , \vee , \wedge) !, 11, 82

↳ valutazione (dato circuito si calcola uscita)
↳ sintesi (dato elaborazione si disegna il circ.)

in generale 0 → false 1 → true

regole:

$$a \wedge a = a; a \vee a = a$$

$$a \wedge b = b \wedge a; a \vee b = b \vee a$$

$$a \wedge (b \wedge c) = (a \wedge b) \wedge c; a \vee (b \vee c) = (a \vee b) \vee c$$

$$a \wedge 0 = 0$$

$$a \wedge 1 = a$$

$$a \vee 0 = a$$

$$a \vee 1 = 1$$

$$a \wedge \neg a = 0$$

$$a \vee \neg a = 1$$

$$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$$

$$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$$

$$\neg(a \wedge b) = \neg a \vee \neg b \quad 1^{\circ} \text{ Teorema di de Morgan}$$

$$\neg(a \vee b) = \neg a \wedge \neg b \quad 2^{\circ} \text{ Teorema di de Morgan}$$

La valutazione di una formula dipende dall'interpretazione (cioè dal valore assegnato alle variabili a, b, \dots), un'interpretazione che rende vera la formula si chiama MODELLO della formula.

us modello: ① data s'interpretazione e s'formula, valutare la formula (= trovare l'uscita di s circuito dati i suoi ingressi)
② dato s'insieme di s'interpretazioni trovare la formula che vale → solo per quell'

us logica: ① valutare formula dati formula e interpretaz.
② sostituire formula al circuito ancora { 3. trovare se è s'modello di s'formula
non fatti } 4. effettuare deduzioni

- ① si fa in base all'interpretazione data
- ② si fa in base alla 1^a riga delle tabelle, che si "traduce" con a, b, ..., 7, 1, V.
- ③ se modello può non esistere

la formula con le variabili ha 2^a interpretazioni

teorema: ogni formula può essere trasformata in una formula equivalente nella forma

$$(F \vee \neg b) \wedge (b \vee D) \Rightarrow \text{appunto } F \vee D$$

(regola di risoluzione: una fine si ottiene o \Rightarrow la formula è insoddisfacibile)

ad sottoformule che sono V di variabili e variabili rechte: FORMA NORMALE CONJUNTIVA (CNF). La soddisficiabilità desiderata si verifica con programma

4. deduzioni

implicazione: \models

es. $a \vee b, \neg a \models b$

$$a \models a \vee b$$

$$a \vee b \models b$$

$$a \vee b, a \vee \neg b \models a$$

$$a \vee b, a \vee \neg b, \neg a \vee b \models a \wedge b$$

$$a \vee b, \neg b \vee c \models a \vee c$$

contrario. $a \vee b \not\models a$

$$a \vee b, b \vee c \not\models a \vee c$$

$$a \not\models a \wedge b$$

implicazione materiale: \rightarrow

indica esplicitamente formule in forma "se allora"

abbrevia: " " \rightarrow "indica " " \neg "

cioè $a \rightarrow b$ abbrevia $\neg a \vee b$

è implicazione materiale, a contrario della

conseguenza logica, risulta vera o falsa a

secondo dei valori assegnati ad a e b

cioè a seconda del contesto può risultare V o F

la semantica della logica non include
informazioni sul contesto

un'implicazione che vale solo nello specifico
contesto va considerata come $A \rightarrow B$

equivalenza: 2 formule sono equivalenti se hanno gli stessi modelli

equiv. di de Morgan: $\neg(a \vee b)$ equiv. $\neg a \wedge \neg b$
 $\neg(a \wedge b)$ equiv. $\neg a \vee \neg b$

se A e B sono equivalenti, $A \Leftrightarrow B$

RETROAZIONE NON FATTA