1

Dalla precedente lezione

La macchina di Von Neumann:

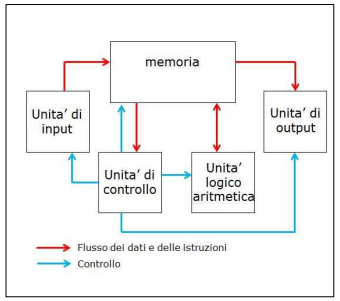
e' uno schema dell'architettura del calcolatore

Principali componenti:

- La memoria
- L'unita' di controllo
- L'unita' logico aritmetica
- Le unita' di Input Output

In particolare, nella memoria del calcolatore sono memorizzati

DATI e ISTRUZIONI



La macchina di Von Neumann

Problema:
come sono rappresentate in memoria tali informazioni?

2

Tipi di dati

Per tipo di dato si intende **una classificazione** che definisce

- i **possibili valori** che un certo dato puo' assumere
- Le **operazioni** che possono essere fatte su quel dato
- Il **significato** in memoria di quel dato

I **tipi di dati elementari** sono:

- Il tipo intero
- Il tipo alfanumerico
- Il tipo logico
- Il tipo reale

3

Tipo di dato intero

Il **tipo intero** indica
il sottinsieme dei numeri relativi (interi positivi e negativi)
che possono essere rappresentati nella memoria di un calcolatore

In generale, nella **rappresentazione posizionale**,
le cifre di un numero sono i
coefficienti di una combinazione lineare delle potenze della base

ESEMPIO: base b=10 $123 = 1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0$

Nella memoria del calcolatore, poiche' le locazioni di memoria
sono costituite da bit, i dati di tipo intero
sono rappresentati col sistema posizionale binario (base *b=2*)

4

rappresentazione binaria pura

Analogamente con il sistema binario ($b = 2$)

Il numero intero ***K=18*** (in base *b=10*)
puo' essere rappresentato in base *b=2* come

K=10010

INFATTI

$$10010 = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 10^2 + 1 \cdot 2^1 + 0 \cdot 10^0 = 16 + 2 = 18$$

(conversione da base b=2 a base b=10)

5

conversione da base 2 a base 10

si effettuano successive divisioni per la base b=2
fino ad ottenere quoziente 0,
e si prendono i resti in ordine inverso

K = 18

		Q	R
18	2	9	0
9	2	4	1
4	2	2	0
2	2	1	0
1	2	0	1

K = 10010

6

operazioni aritmetiche

le operazioni si svolgono secondo le normali regole aritmetiche

addizione: 18+7

(riporti)		(1)	(1)			
(18)	1	0	0	1	0	+
(7)	0	0	1	1	1	=
	1	1	0	0	1	(25)

moltiplicazione: 11 * 5

(11)			1	0	1	1	*
(5)				1	0	1	=
			1	0	1	1	
		0	0	0	0		
	1	0	1	1			
	1	1	0	1	1	1	(55)

7

rappresentazione in memoria

In una locazione di memoria
si rappresentano le cifre binare
del numero intero

rappresentazione di 18

locazione di memoria con **L=8** bit

0	**0**	**0**	**1**	**0**	**0**	**1**	**0**

coefficienti di 2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0

Rappresentazione binaria pura
(rappresentazione unsigned)

8

rappresentazione binaria pura

La limitazione dello spazio implica un
Massimo e un **Minimo intero rappresentabile**

0	0	0	0	0	0	0	0	MIN = 0	minimo intero rappresentabile
1	1	1	1	1	1	1	1	MAX = 255	massimo intero rappresentabile

Con **L = 8 bit** un numero intero **K** e' rappresentabile se

$$0 \leq K \leq 255$$

In generale, con **L bit**, un intero **K** e' rappresentabile se

$$0 \leq K \leq 2^L - 1$$

rappresentazione dei numeri relativi

1) RAPPRESENTAZIONE SEGNO E MODULO

il primo bit della locazione di memoria contiene il segno

Ad esempio 1 = - e 0 = +

rappresentazione di **+18**

0	0	0	1	0	0	1	0

segno

rappresentazione di **-18**

1	0	0	1	0	0	1	0

segno

Avendo **L=7** bit per la rappresentazione del modulo si ha **MAX=127**

un intero K e' rappresentabile se $-127 \leq K \leq 127$

In generale K e' rappresentabile se $-(2^{L-1} - 1) \leq K \leq 2^{L-1} - 1$

rappresentazione segno e modulo

La rappresentazione segno e modulo ha due difetti:

1) doppia rappresentazione dello 0

rappresentazione di **+0**

0	0	0	0	0	0	0	0

rappresentazione di **-0**

1	0	0	0	0	0	0	0

2) impossibilita' di effettuare la sottrazione come somma algebrica tra numeri di segno opposto ($X - Y \neq X + (-Y)$)

+18	**0**	0	0	1	0	0	1	0	
				+					
-5	**1**	0	0	0	0	1	0	1	
				=					
	1	0	0	1	0	1	1	1	**-23**

necessita' di due circuiti distinti per la somma e la differenza all'interno dell'ALU

Rappresentazione dei numeri relativi

2) RAPPRESENTAZIONE COMPLEMENTO A 2 (con L bit)

- La rappresentazione di un **numero positivo** coincide con la rappresentazione unsigned, utilizzando al piu' L-1 bit
- La rappresentazione di un **numero negativo** si effettua in due passi:
  a) si **cambiano i bit 0 in 1 e viceversa** della rappresentazione del valore assoluto (complemento a 1)
  b) si **aggiunge 1** al risultato

Esempio rappresentazione di **K = -18 con 8 bit**

rappresentazione di 18	0	0	0	1	0	0	1	0
	1	1	1	0	1	1	0	1
	1	1	1	0	1	1	1	0

complemento a 1

aggiunta di 1 al risultato

rappresentazione di -18

Il primo bit di un numero negativo e' sempre 1
(ma non deve essere visto come il segno)

osservazioni

La conversione da binario a decimale si effettua con i passi inversi:

- Se il primo **bit=0 (numero positivo)** si converte con la rappresentazione unsigned
- Se il primo **bit=1 (numero negativo)** allora:
  - Si sottrae 1 alla rappresentazione
  - Si effettua il complemento a 1 e si ottiene l'opposto del numero da convertire

Esempio:

numero negativo! | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |

sottrai 1 al risultato

| 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |

complemento a 1

| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |

rappresentazione dell'opposto.
Quindi k = -18

13

rappresentazione complemento a 2

La rappresentazione complemento a 2 permette di eseguire la differenza come somma algebrica di numeri con segno opposto

eseguiamo la somma algebrica tra **18** e **-18**

18 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |

\+

-18 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |

=

si ignora l'ultimo riporto → (1) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 !!

Inoltre

- | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | e' l'unica rappresentazione dello 0!
- Con L=8 bit, un numero K e' rappresentabile se $-128 \leq K \leq 127$
- In generale con L bit, un numero K e' rappresentabile se $-2^{L-1} \leq K \leq 2^{L-1} - 1$

14

rappresentazione complemento a 2

1) E' possibile determinare l'opposto di un numero negativo mediante:
- sottrai 1 alla rappresentazione
- effettua il complemento a 1

Esempio:

-18 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |

sottrai 1

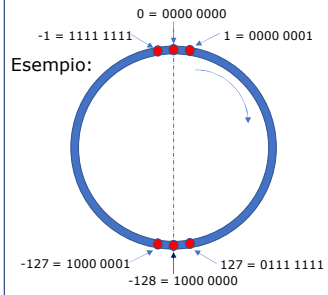
| 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |

complemento a 1

18 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |

2) E' possibile interpretare la rappresentazione in maniera "circolare", dove gli opposti sono simmetrici rispetto all'asse verticale

0 e -128 non hanno opposti!!



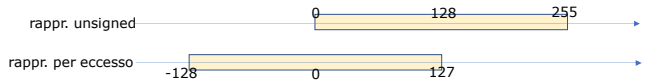
15

Rappresentazione dei numeri relativi

3) RAPPRESENTAZIONE PER ECCESSO (CON BIAS O CON SHIFT)

Fissato un eccesso S, a partire dalla rappresentazione unsigned,
si mappa l'intervallo [-S; S-1] sull'intervallo [0; 255]

Esempio: **S=128**



Ogni numero **K** nell'intervallo [-128, 127]
viene allora rappresentato unsigned con **128+K** nell'intervallo [0, 255]

Esempi

K = 18 ➡ 128+18 = 146

| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |

rappresentazione di 146 unsigned

K = -18 ➡ 128-18 = 110

| 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |

rappresentazione di 110 unsigned

16

osservazioni

- il **primo bit** di un numero negativo e' sempre 0 (ma non deve essere visto come il segno)
- La **conversione da binario a decimale** si effettua con i passi inversi
  - Si effettua la conversione unsigned
  - Si sottrae 128

Esempio:

1	0	0	1	0	0	1	0

Rappresentazione unsigned = 146

K = 146-128 = **18**

17

inoltre

- | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | e' l'unica rappresentazione dello 0!
- Con L=8 bit, un numero K e' rappresentabile se $-128 \leq K \leq 127$
- In generale con L bit, un numero K e' rappresentabile se $-2^{L-1} \leq K \leq 2^{L-1} - 1$
- E' possibile eseguire la differenza come somma algebrica di numeri di segno opposto

eseguiamo la somma algebrica tra **18** e **-18**

1	0	0	1	0	0	1	0	18 + 128
+								
0	1	1	0	1	1	1	0	-18 + 128
=								
(1) 0	0	0	0	0	0	0	0	0 + 256 (!!) bisogna sottrarre 128

corrisponde a 2 volte 128

1	0	0	0	0	0	0	0	0 + 128

rappresentazione di 0 eccesso 128

18

rappresentazione per eccesso: osservazioni

1) La rappresentazione per eccesso e la rappresentazione complemento a 2 coincidono a meno del bit del segno

Esempio:

-18 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |

rappresentazione complemento a 2

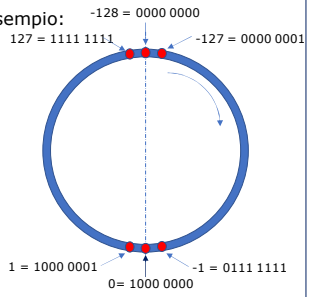
-18 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |

rappresentazione eccesso 128

2) Anche la rappresentazione per eccesso ha una interpretazione circolare

0 e -128 non hanno opposti!!

Esempio:



19

ricapitolando

L = 8	Unsigned	Segno e modulo	Complemento a 2	Per eccesso
Intervallo di rappresentabilita'	[0 ; 255]	[-127 ; 127]	[-128 ; 127]	[-128 ; 127]
Rappr. di 0	0000 0000	0000 0000 1000 0000	0000 0000	1000 0000
Rappr. di MIN	0000 0000	1111 1111	1000 0000	0000 0000
Rappr. di MAX	1111 1111	0111 1111	0111 1111	0000 0000
Possibile somma algebrica	NO	NO	SI	SI

Nelle moderne CPU viene utilizzata la rappresentazione complemento a 2 con L = 32

20

in generale

L generico	Unsigned	Segno e modulo	Complemento a 2	Per eccesso
Intervallo di rappresentabilita'	$MIN = 0$ $MAX = 2^{L} - 1$	$MIN = -(2^{L-1} - 1)$ $MAX = 2^{L-1} - 1$	$MIN = -2^{L-1}$ $MAX = 2^{L-1} - 1$	$MIN = -2^{L-1}$ $MAX = 2^{L-1} - 1$

Nelle moderne CPU viene utilizzata la rappresentazione
complemento a 2 con L = 32

MIN = - 2147483648 **MAX = 2147483647**

21

operazioni aritmetiche

Sul tipo di dati intero sono definite le tradizionali operazioni aritmetiche:

- Addizione
- Sottrazione
- Moltiplicazione
- Divisione

OSSERVAZIONE

Poiche' nella rappresentazione del tipo intero non e' previsto il punto decimale, la divisione tra numeri interi produce sempre un numero intero per troncamento

Esempio: A=5, B=2 ➔ A/B = 2

22

Overflow

La situazione per cui il **risultato** di una operazione tra due numeri rappresentabili **non e' rappresentabile** e' detta

OVERFLOW

Esempio: *b=2, L=32* (situazione reale!)

MAX = $2^{L-1} - 1 \sim 10^9$

Se *K=13* si ha

K! = 6227020800 > MAX

Quindi

13! e' una operazione che provoca overflow

23

esempi

Se *MAX=127,* quanto fa *60+70*?

Entrambi gli addendi sono rappresentabili, ma **il risultato non e' rappresentabile**!

Se *MAX=127,* quanto fa *60+70-40*?

Tutti gli addendi sono rappresentabili ma
(60+70) – 40 **non e' rappresentabile**
mentre
60 + (70-40) **e' rappresentabile**

Alcune proprieta' dell'aritmetica tradizionale non valgono!

24

overflow

La situazione per cui il **risultato** di una operazione tra due numeri rappresentabili **non e' rappresentabile** e' detta

OVERFLOW

Esempio: *b=2, L=32* (situazione reale!)

$MAX = 2^{L-1} - 1 \sim 10^9$

Se *K=13* si ha

K! = 6227020800 > MAX

Quindi

13! e' una operazione che provoca overflow

tipo di dato alfanumerico

Il **tipo di dati alfanumerico** indica l'insieme finito di caratteri solitamente usato nel linguaggio naturale scritto:

- Lettere dell'alfabeto (maiuscole e minuscole)
- Le cifre decimali (0, 1, 2, …, 9)
- I caratteri speciali e di punteggiatura (!, @, #, $, [, {, +, -, …)

I caratteri sono combinati tra loro in **stringhe**

codici alfanumerici

Il tipo di dato alfanumerico e' rappresentato mediante opportuni codici

Codice: funzione di trasformazione di un carattere in un insieme di bit

Esistono vari tipi di codici

- EBCDIC (codice a 8 bit, utilizzato in calcolatori IBM negli anni 50 del XX sec.)
- Fieldata (codice a 7 bit, utilizzato nei calcolatori Univac negli anni 60 del XX sec.)
- ASCII (codice a 7 bit + 1 di controllo, nato come standard nel 1968)

carattere	Codice EBCDIC	Codice Fieldata	Codice ASCII
0	1111 0000	011 0000	0011 0000
1	1111 0001	011 0001	0011 0001
2	1111 0010	011 0010	0011 0002
A	1100 0001	000 0110	0100 0001
B	1100 0010	000 0111	0100 0010
C	1100 0011	000 1000	0100 0011
#	1111 1011	000 0011	0010 0011
@	1111 1100	000 0000	0100 0000
=	1111 1110	010 0100	0011 1101

codice ASCII

American Standard Code for Information Exchange

Dec	Hex	Oct	Bin	Char	Dec	Hex	Oct	Bin	Char	Dec	Hex	Oct	Bin	Char	Dec	Hex	Oct	Bin	Char
0	0x00	000	0000000	NUL	32	0x20	040	0100000	space	64	0x40	100	1000000	@	96	0x60	140	1100000	`
1	0x01	001	0000001	SOH	33	0x21	041	0100001	!	65	0x41	101	1000001	A	97	0x61	141	1100001	a
2	0x02	002	0000010	STX	34	0x22	042	0100010	"	66	0x42	102	1000010	B	98	0x62	142	1100010	b
3	0x03	003	0000011	ETX	35	0x23	043	0100011	#	67	0x43	103	1000011	C	99	0x63	143	1100011	c
4	0x04	004	0000100	EOT	36	0x24	044	0100100	$	68	0x44	104	1000100	D	100	0x64	144	1100100	d
5	0x05	005	0000101	ENQ	37	0x25	045	0100101	%	69	0x45	105	1000101	E	101	0x65	145	1100101	e
6	0x06	006	0000110	ACK	38	0x26	046	0100110	&	70	0x46	106	1000110	F	102	0x66	146	1100110	f
7	0x07	007	0000111	BEL	39	0x27	047	0100111	'	71	0x47	107	1000111	G	103	0x67	147	1100111	g
8	0x08	010	0001000	BS	40	0x28	050	0101000	(	72	0x48	110	1001000	H	104	0x68	150	1101000	h
9	0x09	011	0001001	TAB	41	0x29	051	0101001	)	73	0x49	111	1001001	I	105	0x69	151	1101001	i
10	0x0A	012	0001010	LF	42	0x2A	052	0101010	*	74	0x4A	112	1001010	J	106	0x6A	152	1101010	j
11	0x0B	013	0001011	VT	43	0x2B	053	0101011	+	75	0x4B	113	1001011	K	107	0x6B	153	1101011	k
12	0x0C	014	0001100	FF	44	0x2C	054	0101100	,	76	0x4C	114	1001100	L	108	0x6C	154	1101100	l
13	0x0D	015	0001101	CR	45	0x2D	055	0101101	-	77	0x4D	115	1001101	M	109	0x6D	155	1101101	m
14	0x0E	016	0001110	SO	46	0x2E	056	0101110	.	78	0x4E	116	1001110	N	110	0x6E	156	1101110	n
15	0x0F	017	0001111	SI	47	0x2F	057	0101111	/	79	0x4F	117	1001111	O	111	0x6F	157	1101111	o
16	0x10	020	0010000	DLE	48	0x30	060	0110000	0	80	0x50	120	1010000	P	112	0x70	160	1110000	p
17	0x11	021	0010001	DC1	49	0x31	061	0110001	1	81	0x51	121	1010001	Q	113	0x71	161	1110001	q
18	0x12	022	0010010	DC2	50	0x32	062	0110010	2	82	0x52	122	1010010	R	114	0x72	162	1110010	r
19	0x13	023	0010011	DC3	51	0x33	063	0110011	3	83	0x53	123	1010011	S	115	0x73	163	1110011	s
20	0x14	024	0010100	DC4	52	0x34	064	0110100	4	84	0x54	124	1010100	T	116	0x74	164	1110100	t
21	0x15	025	0010101	NAK	53	0x35	065	0110101	5	85	0x55	125	1010101	U	117	0x75	165	1110101	u
22	0x16	026	0010110	SYN	54	0x36	066	0110110	6	86	0x56	126	1010110	V	118	0x76	166	1110110	v
23	0x17	027	0010111	ETB	55	0x37	067	0110111	7	87	0x57	127	1010111	W	119	0x77	167	1110111	w
24	0x18	030	0011000	CAN	56	0x38	070	0111000	8	88	0x58	130	1011000	X	120	0x78	170	1111000	x
25	0x19	031	0011001	EM	57	0x39	071	0111001	9	89	0x59	131	1011001	Y	121	0x79	171	1111001	y
26	0x1A	032	0011010	SUB	58	0x3A	072	0111010	:	90	0x5A	132	1011010	Z	122	0x7A	172	1111010	z
27	0x1B	033	0011011	ESC	59	0x3B	073	0111011	;	91	0x5B	133	1011011	[	123	0x7B	173	1111011	{
28	0x1C	034	0011100	FS	60	0x3C	074	0111100	<	92	0x5C	134	1011100	\	124	0x7C	174	1111100	|
29	0x1D	035	0011101	GS	61	0x3D	075	0111101	=	93	0x5D	135	1011101	]	125	0x7D	175	1111101	}
30	0x1E	036	0011110	RS	62	0x3E	076	0111110	>	94	0x5E	136	1011110	^	126	0x7E	176	1111110	~
31	0x1F	037	0011111	US	63	0x3F	077	0111111	?	95	0x5F	137	1011111	_	127	0x7F	177	1111111	DEL

Il codice ASCII rappesenta oggi lo standard per la rappresentazione del tipo alfanumerico

stringhe di caratteri

Ogni carattere del codice ASCII e' rappresentabile in una **locazione di memoria da 8 bit**

Se si ha necessita' di memorizzare piu' caratteri, si concatenano piu' locazioni di memoria

Esempio:
La parola NAPOLI puo' essere rappresentata utilizzando 6 locazioni di memoria consecutive

1	0	1	0	1	1	1	0	N
1	0	1	0	0	0	0	1	A
1	0	1	1	0	0	0	0	P
1	0	1	0	1	1	1	1	O
1	0	1	0	1	1	0	0	L
1	0	1	0	1	0	0	1	I

operazioni sui caratteri

Sul tipo di dato Alfanumerico sono definite le operazioni di

- **Confronto** (secondo l'ordinamento alfabetico, conservato dal codice ASCII)
- **Concatenazione** (es: pesce//cane = pescecane)

OSSERVAZIONE

Ovviamente sul sottinsieme del tipo alfanumerico
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

non sono definite le operazioni aritmetiche

tipo di dato logico

Il **tipo di dato Logico** indica l'insieme costituito da due soli elementi:

{ vero ; falso }

Si utilizza quindi nella verifica della **veridicita' di una proposizione**

Esempio:

P: Napoli e' in Campania — **P** e' una proposizione **vera**

Q: 3 e' maggiore di 5 — **Q** e' una proposizione **falsa**

rappresentazione

Per rappresentare i due valori in memoria sarebbe sufficiente un solo bit

MA

Poiche' la minima quantita' di memoria indirizzabile e' la locazione di memoria

Il tipo di dato logico e' rappresentato utilizzando un'intera locazione

vero

0	0	0	0	0	0	0	1

falso

0	0	0	0	0	0	0	0

operazione di negazione (NOT)

L'operazione di **negazione** e' un'operazione logica che agisce su un dato logico, cambiandone il valore

Esempio:
Siano due numeri A=3 e B=5

P: A e' minore di B
(vero)

not P: A **non** e' minore di B
(falso)

P	not P
V	F
F	V

33

Operazione di congiunzione (AND)

L'operazione di **congiunzione** e' un'operazione logica che agisce su due dati logici, ed e' vera solo se entrambi i dati sono veri

Esempi:
Siano A=3 X=4 B=5

P: X maggiore di A(vero)
Q: X minore di B (vero)

P and Q:
X maggiore di A **e** X minore di B
(cioe' X compreso tra A e B) = vero

P	Q	P and Q
V	V	V
V	F	F
F	V	F
F	F	F

34

operazione di disgiunzione (OR)

L'operazione di **disgiunzione** e' un'operazione logica che agisce su due dati logici, ed e' vera se almeno uno dei dati e' vero

Esempi:
Siano A=3 X=6 B=5

P: X minore di A (falso)
Q: X maggiore di B (vero)

P OR Q:
X minore di A **oppure** X maggiore di B
(cioe' X esterno a [A , B]) = vero

P	Q	P or Q
V	V	V
V	F	V
F	V	V
F	F	F

35

osservazioni

In un'espressione logica le operazioni vengono eseguite nel seguente ordine:

1. NOT
2. AND
3. OR

Esempio: P=falso, Q=falso, T=falso
NOT P OR Q AND T = vero

E' possibile alterare l'ordine mediante le parentesi

Esempio: P=falso, Q=falso, T=falso
(NOT P OR Q) AND T = falso

Valgono le formule di De Morgan

NOT (P AND Q) = NOT P OR NOT Q
NOT (P OR Q) = NOT P AND NOT Q

36

operazioni aritmetiche e logiche

esiste una naturale corrispondeza
tra i valori **VERO / FALSO** con i valori **1 / 0**

attraverso questa corrispondenza e' possible interpretare le
operazioni aritmetiche tra numeri binari come **operazioni logiche**

La somma tra due bit **X** e **Y** produce due bit **S** (somma) e **R** (riporto)
secondo la seguente tabella

X	Y	R	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

⟺

X	Y	R	S
Falso	Falso	Falso	Falso
Falso	Vero	Falso	Vero
Vero	Falso	Falso	Vero
Vero	Vero	Vero	Falso

tavola della verita' della somma tra due bit

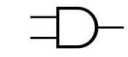
37

(semi)addizionatore

X	Y	R	S
Falso	Falso	Falso	Falso
Falso	Vero	Falso	Vero
Vero	Falso	Falso	Vero
Vero	Vero	Vero	Falso

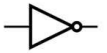
Un **(semi)addizionatore** e' un
circuito elettronico che realizza
la **somma tra due bit**

Puo' essere realizzato attraverso
componenti elementari che
implementano le operazioni logiche
(porte logiche)



PORTA **AND**

PORTA **OR**



PORTA **NOT**

Le porte logiche sono circuiti che hanno
1 o 2 valori in input e producono 1 valore in output

38

(semi)addizionatore

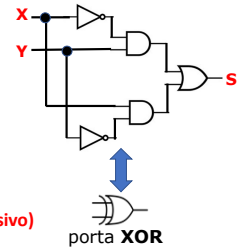
X	Y	R	S
Falso	Falso	Falso	Falso
Falso	Vero	Falso	Vero
Vero	Falso	Falso	Vero
Vero	Vero	Vero	Falso

R e' sempre Falso, tranne quando
X e Y sono entrambe Vero

S e' Vero quando X e Y, ma non entrambi,
sono Vero

R = X AND Y

S = (NOT X AND Y) OR (X AND NOT Y)



L'operazione per il calcolo di **S** e' detta **XOR (OR esclusivo)**

porta **XOR**

39

addizionatore

Un **addizionatore** e' un circuito elettronico che realizza
la **somma tra tre bit (due operandi e il riporto)**

Produce un risultato **S** e un nuovo riporto **R'**

R	X	Y	R'	S
Falso	Falso	Falso	Falso	Falso
Falso	Falso	Vero	Falso	Vero
Falso	Vero	Falso	Falso	Vero
Falso	Vero	Vero	Vero	Falso
Vero	Falso	Falso	Falso	Vero
Vero	Falso	Vero	Vero	Falso
Vero	Vero	Falso	Vero	Falso
Vero	Vero	Vero	Vero	Vero

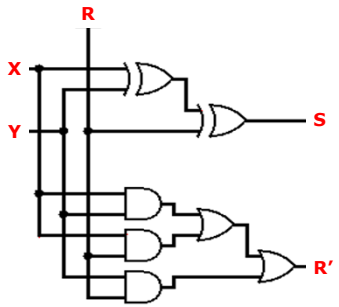
tavola della verita' della somma di tre bit

40

addizionatore

R' = (**X AND Y**) **OR** (**X AND R**) **OR** (**Y AND R**)

S = **X XOR Y XOR R**



41

riassumendo

Rappresentazione in memoria di alcuni tipi di dati fondamentali

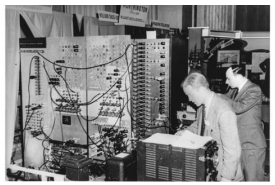
- Il tipo intero (basato sul sistema binario)
- Il tipo alfanumerico (basato sul codice ASCII)
- Il tipo logico

42

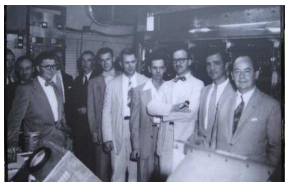
un po' di storia (3)

John Von Neumann (1903-1957)

- Matematico e fisico ungherese di origine ebraiche, si trasferisce negli Stati Uniti nel 1933
- Nel 1946, con l'articolo "draft report on the EDVAC", introduce le idee di Turing nello sviluppo del primo calcolatore elettronico general-purpose con programma memorizzabile: l'EDVAC. La "macchina di Von Neumann" e' ancora il modello di riferimento delle moderne architetture.
- L'EDVAC utilizzava valvole termoioniche, aveva una memoria di 1024 parole da 44 bit e eseguiva una somma in 0.8 msec. l'I/O avveniva con schede perforate
- Von Neumann si occupo' anche delle applicazioni della matematica alla meccanica quantistica, alla cibernetica, all'economia, alla biologia, alla teoria dei giochi e all'intelligenza artificiale.



L'EDVAC (courtesy of Computer History Museum)



Von Neumann, primo a destra
(courtesy of Computer History Museum)

43