

RAPPRESENTAZIONE DELL'INFORMAZIONE

Rappresentazione dell'Informazione (1/2)

L'informazione è qualcosa di astratto.
Per poterla manipolare bisogna rappresentarla.

In un calcolatore i vari tipi di informazione (testi, figure, numeri, musica,...) si rappresentano per mezzo di sequenze di bit (cifre binarie).

Bit è l'abbreviazione di Binary digIT, numero binario.

Il bit è l'unità di misura elementare dell'informazione, ma anche la base del sistema numerico utilizzato dai computer. Può assumere soltanto due valori: 0 o 1.

Byte è l'unità di misura dell'informazione che corrisponde ad 8 bit.

Rappresentazione dell'Informazione (2/2)

Quanta informazione può essere contenuta in una sequenza di n bit?

L'informazione corrisponde a tutte le possibili disposizioni con ripetizione di due oggetti (0 ed 1) in n caselle (gli n bit), ossia 2^n

Esempio: $n=2$.

00

01

10

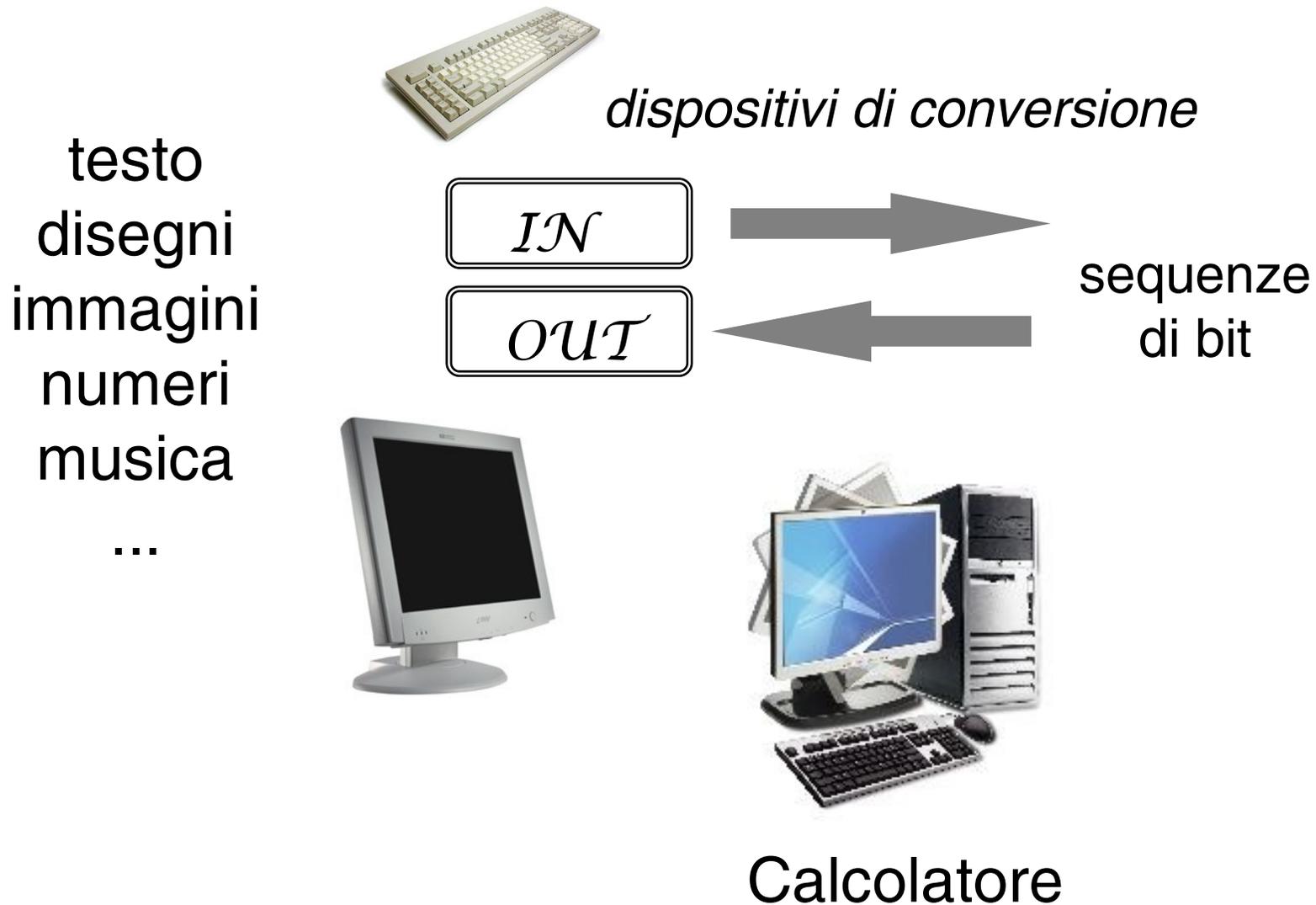
11

ATTENZIONE: Una stessa sequenza di bit può rappresentare informazione differente.

Per esempio 01000001 rappresenta

- l'intero 65
- il carattere 'A'
- il colore di un puntino sullo schermo

Calcolatore e Informazione



Rappresentazione del testo (1/2)

**Codifica ASCII (American Standard Code for Information Interchange)
Standard su 7 bit (il primo bit del byte sempre 0)**

Sequenze	Caratteri
00110000	0
00110001	1
00110010	2
...	...
00111001	9
...	...
01000001	A
01000010	B
01000011	C
...	...
01100001	a
01100010	b
...	...

01000011
01100001
01110010
01101111
00100000
01100001
01101110
01101001
01100011
01101111
00101100

Caro amico,



Rappresentazione del testo (2/2)

Codifica ASCII – Tabella di corrispondenza

3 cifre più significative

000	001	010	011	100	101	110	111	
NUL	DLE	SP	0	@	P	`	p	0000
SOH	XON	!	1	A	Q	a	q	0001
STX	DC2	"	2	B	R	b	r	0010
ETX	XOFF	#	3	C	S	c	s	0011
EQT	DC4	\$	4	D	T	d	t	0100
ENQ	NAK	%	5	E	U	e	u	0101
ACK	SYN	&	6	F	V	f	v	0110
BEL	ETB	'	7	G	W	g	w	0111
BS	CAN	(8	H	X	h	x	1000
HT	EM)	9	I	Y	i	y	1001
LF	SUB	*	:	J	Z	j	z	1010
VF	ESC	+	;	K	[k	{	1011
FF	FS	,	<	L	\	l		1100
CR	GS	-	=	M]	m	}	1101
SO	RS	.	>	N	^	n	~	1110
SI	US	/	?	O	_	o	DEL	1111

4 cifre meno
significative

RAPPRESENTAZIONE DEI NUMERI NATURALI

Rappresentazione dei numeri naturali (1/11)

Base dieci

◆ Cifre: 0, 1, 2, 3, 4, 6, 7, 8, 9

◆ Rappresentazione posizionale

$(123)_{dieci}$ *significa* $1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0$

Base due

◆ Cifre: 0, 1

◆ Rappresentazione posizionale

$(11001)_{due}$ *significa* $1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 (= 25)_{dieci}$

Rappresentazione dei numeri naturali (2/11)

Data una base $\beta \geq \text{due}$

Ogni numero naturale N minore di β ($N < \beta$) è associato ad un simbolo elementare detto **cifra**

BASE	CIFRE
due	0 1
cinque	0 1 2 3 4
otto	0 1 2 3 4 5 6 7
sedici	0 1 2 3 4 5 6 7 8 9 A B C D E F

Rappresentazione dei numeri naturali (3/11)

I numeri naturali maggiori o uguali a β possono essere rappresentati da una sequenza di cifre secondo la *rappresentazione posizionale*

Se un numero naturale $N \geq \beta$ è rappresentato in base β dalla sequenza di cifre:

$$a_{p-1}a_{p-2} \dots a_1a_0$$

allora N può essere espresso come segue:

$$N = \sum_{i=0}^{p-1} a_i \beta^i = a_{p-1} \beta^{p-1} + a_{p-2} \beta^{p-2} + \dots + a_2 \beta^2 + a_1 \beta + a_0$$

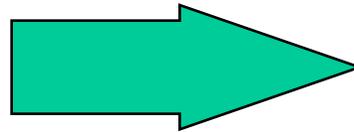
dove a_0 è detta «cifra meno significativa» e a_{p-1} «cifra più significativa»

Chiamiamo questa formula «formula della sommatoria»

Rappresentazione dei numeri naturali (4/11)

Data una sequenza di cifre in base β , a quale numero naturale corrisponde?

Sequenze
di simboli
(in base β)



Sequenze
di simboli
(in base 10)

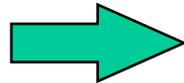
$a_{p-1}a_{p-2} \dots a_1a_0$

$N?$

656_8

$A03_{16}$

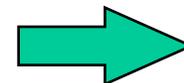
1101_2



$$6 \cdot 8^2 + 5 \cdot 8^1 + 6 \cdot 8^0$$

$$10 \cdot 16^2 + 0 \cdot 16^1 + 3 \cdot 16^0$$

$$1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$$



430

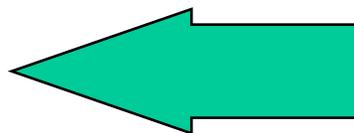
2563

13

Rappresentazione dei numeri naturali (5/11)

Data la base β ed un numero naturale N , trovare la sequenza di cifre che rappresenta N in base β

Sequenze
di simboli
(in base β)



Sequenze
di simboli
(in base 10)

$a_{p-1}a_{p-2} \dots a_1a_0$

N

Rappresentazione dei numeri naturali (6/11)

Esempio: da base 10 a base 2

$N = 23$

inizio

	QUOZIENTE	RESTO	Rappresentazione
	23	-	
div 2	11	1	$11*2+1$
div 2	5	1	$((5*2)+1)*2+1$
div 2	2	1	$((((2*2)+1)*2)+1)*2+1$
div 2	1	0	$(((((1*2)+0)*2)+1)*2)+1)*2+1$
div 2	0	1	$(((((0*2+1)*2+0)*2+1)*2+1)*2)+1$

fine

$(10111)_{due}$
 $a_4 a_3 a_2 a_1 a_0$

che in base dieci vale $1*2^4 + \dots + 1 = (23)_{dieci}$, cvd

Rappresentazione dei numeri naturali (7/11)

Sia **mod** il resto e **div** il quoziente della divisione intera

Procedimento mod/div

Se $N = 0$ porre $a_0 = 0$; \Rightarrow fine

Altrimenti: porre $q_0 = N$ e poi eseguire la seguente procedura iterativa:

$$q_1 = q_0 \text{ div } \beta \quad a_0 = q_0 \text{ mod } \beta$$

$$q_2 = q_1 \text{ div } \beta \quad a_1 = q_1 \text{ mod } \beta$$

...

$$q_p = q_{p-1} \text{ div } \beta \quad a_{p-1} = q_{p-1} \text{ mod } \beta$$

fino a quando q_p diventa uguale a 0

Il procedimento si arresta quando $q_p = 0$ (più precisamente subito dopo aver calcolato a_{p-1}). Inoltre p è proprio il numero di cifre necessario per rappresentare N in base β

Esempio:

$$23 \text{ div } 2 = 11$$

$$23 \text{ mod } 2 = 1$$

NB: Il risultato della **mod** è sempre una cifra valida in base β , perché restituisce sempre un numero fra 0 e $\beta-1$ (estremi inclusi).

Rappresentazione dei numeri naturali (8/11)

Inizio $q_0 = N$

	QUOZIENTE	RESTO
	$q_0 = N$	-
div β	$q_1 = q_0 / \beta$	a_0
div β	$q_2 = q_1 / \beta$	a_1
div β	$q_3 = q_2 / \beta$	a_2
div β	$q_4 = q_3 / \beta$	a_3
div β	$q_5 = q_4 / \beta$	a_4
div β	$q_6 = q_5 / \beta$	a_5
	$q_7 = \mathbf{0}$	a_6

fine

$$N = a_6 \cdot \beta^6 + a_5 \cdot \beta^5 + a_4 \cdot \beta^4 + a_3 \cdot \beta^3 + a_2 \cdot \beta^2 + a_1 \cdot \beta^1 + a_0 \cdot \beta^0$$

Rappresentazione dei numeri naturali (9/11)

Potenza della base: $2^p \leftrightarrow \underbrace{(100\dots00)}_p_{due}$

Intervallo di rappresentabilità con p bit



p	Intervallo $[0, 2^p-1]$
8	$[0, 255]$
16	$[0, 65535]$
32	$[0, 4294967295]$

NB: per la generica base β ,
l'intervallo di rappresentabilità
con p cifre è $[0, \beta^p-1]$

Rappresentazione dei numeri naturali (10/11)

Calcolatore lavora con un numero finito di bit

- ◆ Supponiamo che $p = 16$ bit
- ◆ $A = 0111011011010101$ (30421)
 $B = 1010100001110111$ (43127)
- ◆ Poiché $A + B$ (73552) è maggiore di $2^p - 1$ (65535), quindi non è rappresentabile su p bit, si dice che la somma ha dato luogo ad **overflow**
- ◆ In generale, ci vogliono $p+1$ bit per la somma di due numeri di p bit

$$A = 1111111111111111 \quad (65535)$$

$$B = 1111111111111111 \quad (65535)$$

$$11111111111111110 \quad (131070)$$

Rappresentazione dei numeri naturali (11/11)

Somma fra due numeri naturali espressi in binario.

Per sommare 10011 e 101101, basta metterli in colonna allineandoli a destra (come si fa con i numeri in base 10) e poi tenere presenti le seguenti regole:

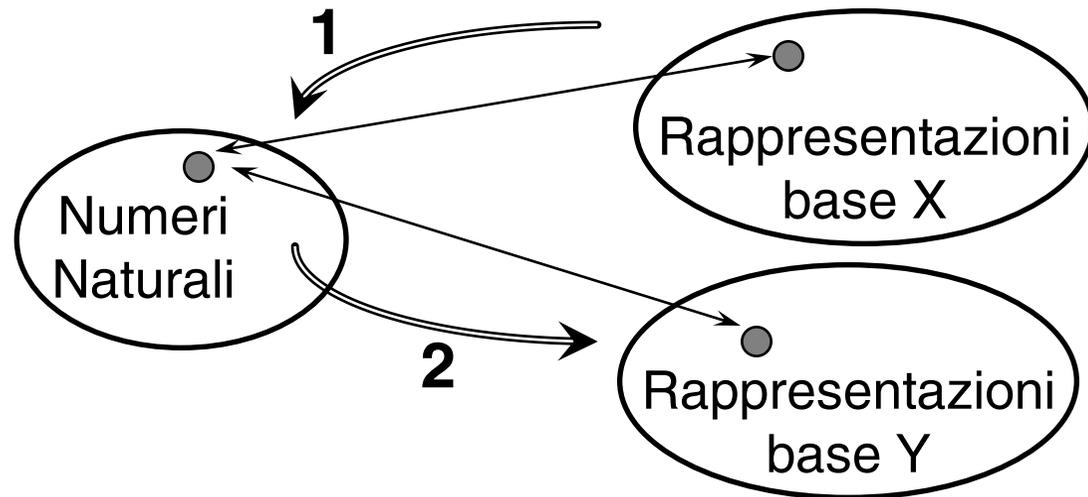
		genera un riporto	genera un riporto	eventuale riporto generatosi precedentemente
		1	11	
0 +	0 +	1 +	1 +	1 +
0 =	1 =	0 =	1 =	1 =
-----	-----	-----	-----	-----
0	1	1	10	11

Esempio: calcolare la somma di 101100 e 111010

11 ← riporti generati

$$\begin{array}{r}
 101100 + \\
 111010 = \\
 \hline
 1101110
 \end{array}$$

Numeri naturali - Cambio di base (1/2)



Da base X a base Y

Trovare la rappresentazione in base 9 di $(1023)_{cinque}$

1) Trasformo in base 10 ed ottengo 138

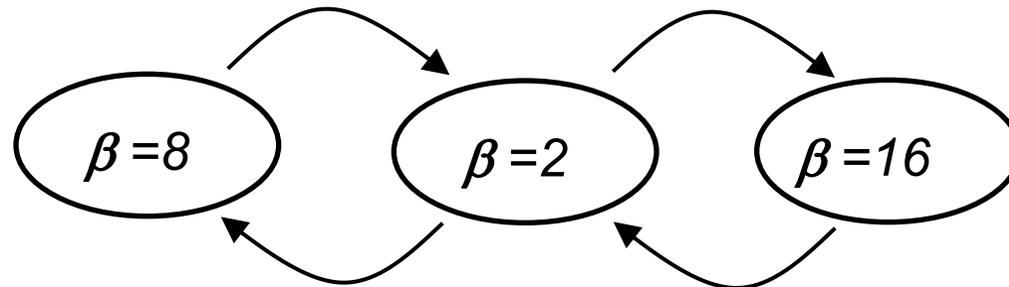
2) Applico il procedimento `mod/div` ed ottengo $(163)_{nove}$

Numeri naturali - Cambio di base (2/2)

Casi particolari (potenze di 2)

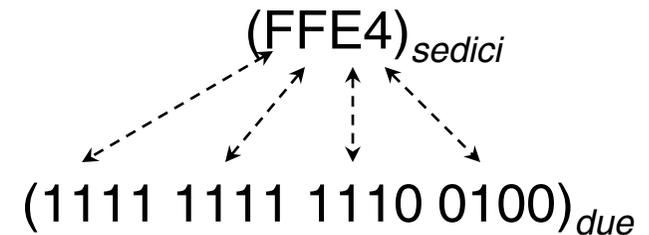
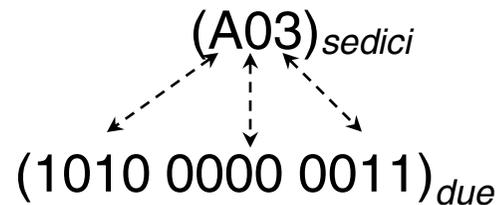
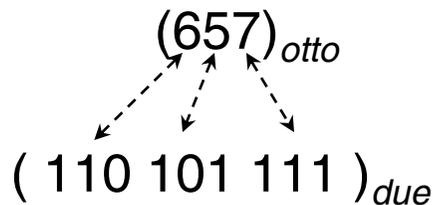
$\beta = 8$

0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111



$\beta = 16$

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111



RAPPRESENTAZIONE DEI NUMERI INTERI

Prima rappresentazione: *rappresentazione in modulo e segno*

Trasformazione $a \Rightarrow A$

Sia a (es. $a=+3$, $a=-3$, ...) il numero intero che si vuole rappresentare *in modulo e segno* su p bit. La rappresentazione A (es. 0011, 1011, ...) di a è data da:

$$A = a_{p-1}, \dots, a_0 = (\text{segno}_a, ABS_a)$$

dove ABS_a è la rappresentazione (come numero naturale) del valore assoluto di a su $p-1$ bit (in particolare ABS_a è rappresentato mediante i bit a_{p-2}, \dots, a_0), e segno_a è un unico bit che vale 0 se $a \geq 0$ e 1 se $a \leq 0$.

NB: ad $a=0$ corrispondono due rappresentazioni: *+zero* (0,00..0) e *-zero* (1,00..0)

Ad esempio, per $p = 4$ si ha:

$a=+3 \Rightarrow \text{segno}_a = 0$, $ABS_a = 011$ (naturale 3 rappresentato su 3 cifre) $\Rightarrow A=0011$

$a=-3 \Rightarrow \text{segno}_a = 1$, $ABS_a = 011$ (naturale 3 rappresentato su 3 cifre) $\Rightarrow A=1011$

Prima rappresentazione: *rappresentazione in modulo e segno*

Trasformazione $\mathbf{A} \Rightarrow \mathbf{a}$

Data una rappresentazione \mathbf{A} di un intero **in modulo e segno** su p bit, come si risale all'intero \mathbf{a} da esso rappresentato?

La rappresentazione \mathbf{A} va vista come composta di due parti: $\mathbf{A} = (\underbrace{a_{p-1}}, \underbrace{a_{p-2} \dots a_0})$
dopodiché:

$$\mathbf{a} = (a_{p-1} == 0) ? +ABS_a : -ABS_a$$

dove ABS_a è il naturale corrispondente ad $a_{p-2} \dots a_0$

Ad esempio, per $p = 4$ si ha:

$\mathbf{A} = 0011 \Rightarrow$ viene visto come $(0,011) \Rightarrow \mathbf{a} = +011$ (*+tre*)

$\mathbf{A} = 1011 \Rightarrow$ viene visto come $(1,011) \Rightarrow \mathbf{a} = -011$ (*-tre*)

NB: 0000 rappresenta *+zero*, mentre 1000 rappresenta *-zero*.

Numeri Interi (3/14)

A	$\{0,1\}^4$	a
0	0000	+0
1	0001	+1
2	0010	+2
3	0011	+3
4	0100	+4
5	0101	+5
6	0110	+6
7	0111	+7
8	1000	-0
9	1001	-1
10	1010	-2
11	1011	-3
12	1100	-4
13	1101	-5
14	1110	-6
15	1111	-7

**Rappresentazione di interi
in modulo e segno su
calcolatore con $p=4$ bit**

numero negativo \Leftrightarrow bit più significativo
della rappresentazione uguale a 1
(zero rappresentato due volte)

Intervallo di rappresentabilità in modulo e segno su p bit
(doppia rappresentazione dello zero)

$$[-(2^{p-1}-1), +2^{p-1}-1]$$

- $p = 4$ $[-7, +7]$
- $p = 8$ $[-127, +127]$
- $p = 16$ $[-32767, +32767]$

NB: prima di applicare l'algoritmo per $\mathbf{a} \Rightarrow \mathbf{A}$ occorre verificare che a sia rappresentabile su p bit, altrimenti l'algoritmo conduce a risultati sbagliati.

Ad esempio:

tentando di rappresentare $\mathbf{a} = -9$ su $p = 4$ bit, si ottiene:

$\mathbf{A} = (\text{segno}_a, \text{ABS}_a)$, dove $\text{segno}_a = 1$ e $\text{ABS}_a = 9$

ma il naturale 9 (1001) non è rappresentabile su 3 bit!!

Seconda rappresentazione: *rappresentazione in complemento a 2* Trasformazione $a \Rightarrow A$

Sia a (es. $a=+3$, $a=-3$, ...) il numero intero che si vuole rappresentare *in complemento a 2* su p bit. La rappresentazione A (es. 0011, 1101, ...) di a è data da:

$$A = a_{p-1} \dots a_0 = (a \geq 0) ? ABS_a : (due^p - ABS_a)$$

dove sia ABS_a che $(due^p - ABS_a)$ sono rappresentati in base due come numeri naturali su p bit.

Ad esempio, per $p = 4$ si ha:

$a = 0 \Rightarrow ABS_a = 0$, e il naturale 0 ha rappr. 0000 su 4 bit $\Rightarrow A = 0000$

$a = 1 \Rightarrow ABS_a = 1$, e il naturale 1 ha rappr. 0001 su 4 bit $\Rightarrow A = 0001$

$a = 7 \Rightarrow ABS_a = 7$, e il naturale 7 ha rappr. 0111 su 4 bit $\Rightarrow A = 0111$

$a = -1 \Rightarrow ABS_a = 1$, $16-1=15$, dove il naturale 15 ha rappr. 1111 su 4 bit $\Rightarrow A = 1111$

$a = -2 \Rightarrow ABS_a = 2$, $16-2=14$, dove il naturale 14 ha rappr. 1110 su 4 bit $\Rightarrow A = 1110$

$a = -8 \Rightarrow ABS_a = 8$, $16-8=8$, dove il naturale 8 ha rappr. 1000 su 4 bit $\Rightarrow A = 1000$

NB: La rappresentazione in complemento a 2 è anche detta in *complemento alla base*. Infatti lo stesso procedimento può essere generalizzato per rappresentare interi in basi diverse da due.

Seconda rappresentazione: *rappresentazione in complemento a 2* **Trasformazione $A \Rightarrow a$**

Data una rappresentazione $A = a_{p-1} \dots a_0$ di un intero **in complemento a due** su p bit, come si risale all'intero a da esso rappresentato?

$$a = (a_{p-1} == 0) ? +A : -(due^p - A)$$

dove sia A che $(due^p - A)$ vengono visti come naturali su p bit.

Ad esempio, per $p = 4$ si ha:

$$A = 0000 \Rightarrow a = 0 \text{ (zero)}$$

$$A = 0001 \Rightarrow a = +1 \text{ (+uno)}$$

$$A = 0111 \Rightarrow a = +7 \text{ (+sette)}$$

$$A = 1000 \Rightarrow 16-8 = 8 \Rightarrow a = -8 \text{ (-otto)}$$

$$A = 1001 \Rightarrow 16-9 = 7 \Rightarrow a = -7 \text{ (-sette)}$$

$$A = 1111 \Rightarrow 16-15=1 \Rightarrow a = -1 \text{ (-uno)}$$

Numeri Interi (7/14)

A	$\{0,1\}^4$	a
0	0000	0
1	0001	+1
2	0010	+2
3	0011	+3
4	0100	+4
5	0101	+5
6	0110	+6
7	0111	+7
8	1000	-8
9	1001	-7
10	1010	-6
11	1011	-5
12	1100	-4
13	1101	-3
14	1110	-2
15	1111	-1

Rappresentazione di interi in complemento a due su calcolatore con $p=4$ bit

Anche in questo caso:

*numero negativo \Leftrightarrow bit più significativo
della rappresentazione uguale a 1*

Inoltre, a differenza della
rappresentazione in modulo e segno,
non viene sprecata nessuna
rappresentazione (lo zero è
rappresentato una volta sola)

Intervallo di rappresentabilità in complemento a 2 su p bit

$$\left[-2^{p-1}, +2^{p-1}-1\right]$$

- $p = 4$ $[-8, +7]$
- $p = 8$ $[-128, +127]$
- $p = 16$ $[-32768, +32767]$

NB: prima di applicare l'algoritmo per $a \Rightarrow A$ occorre verificare che a sia rappresentabile su p bit, altrimenti l'algoritmo conduce a risultati sbagliati.

Ad esempio, tentando di rappresentare $a=-9$ su $p=4$ bit, si ottiene:

$$A = (2^4 - 9) = 16 - 9 = 7 \Rightarrow 0111,$$

ma è sbagliato! Infatti -9 non è rappresentabile su 4 bit, ne servono almeno 5!

Che il risultato fosse sbagliato si poteva dedurre dal fatto che la rappresentazione del negativo -9 iniziava per 0 (0111 è infatti la rappresentazione di $a=+7$ su 4 bit!).

(su 5 bit, $a=-9$ ha rappresentazione $(2^5 - 9) = 23 \Rightarrow A=10111$)

Numeri Interi (9/14)

Terza Rappresentazione: *rappresentazione con bias* Trasformazione $a \Rightarrow A$

Sia a (es. $a=+3$, $a=-3$, ...) il numero intero che si vuole rappresentare *in rappresentazione con bias* su p bit. La rappresentazione A (es. 1010, 0100, ...) di a è data da:

$$A = a_{p-1} \dots a_0 = a + (2^{p-1} - 1)$$

dove $a+(2^{p-1} - 1)$ è supposto essere non negativo e dunque viene rappresentato come un naturale su p bit. La quantità $(2^{p-1} - 1)$ è detta *bias* e $A = a + bias$

Ad esempio, per $p = 4$ si ha:

$a = 0 \Rightarrow 0+(2^{4-1}-1) = 7,$	e il naturale 7 ha rapp. 0111 su 4 bit $\Rightarrow A = 0111$
$a = 1 \Rightarrow 1+(2^{4-1}-1) = 8,$	e il naturale 8 ha rapp. 1000 su 4 bit $\Rightarrow A = 1000$
$a = 8 \Rightarrow 8+(2^{4-1}-1) = 15,$	e il naturale 15 ha rapp. 1111 su 4 bit $\Rightarrow A = 1111$
$a = -1 \Rightarrow -1+(2^{4-1}-1) = 6,$	e il naturale 6 ha rapp. 0110 su 4 bit $\Rightarrow A = 0110$
$a = -2 \Rightarrow -2+(2^{4-1}-1) = 5,$	e il naturale 5 ha rapp. 0101 su 4 bit $\Rightarrow A = 0101$
$a = -7 \Rightarrow -7+(2^{4-1}-1) = 0,$	e il naturale 0 ha rapp. 0000 su 4 bit $\Rightarrow A = 0000$

NB1: questa rappresentazione è anche detta *rappresentazione con polarizzazione*

NB2: come si vedrà nelle prossime slides, questa rappresentazione viene utilizzata nella rappresentazione dei numeri reali in virgola mobile

Terza Rappresentazione: *rappresentazione con bias* Trasformazione $A \Rightarrow a$

Sia A (es. 1010, 0100, ...) la rappresentazione di un numero intero nella *rappresentazione con bias* su p bit. Il numero intero a (es. $a=+3$, $a=-3$, ...) associato ad A è dato da:

$$a = A - (2^{p-1} - 1)$$

dove A viene visto come numero naturale su p bit. Dunque $a = A - \text{bias}$

Ad esempio, per $p = 4$ si ha:

$$\begin{aligned} A = 0111 &\Rightarrow 7 - (2^{4-1} - 1) = 0 \Rightarrow a = 0 \\ A = 1000 &\Rightarrow 8 - (2^{4-1} - 1) = 1 \Rightarrow a = 1 \\ A = 1111 &\Rightarrow 15 - (2^{4-1} - 1) = 8 \Rightarrow a = 8 \\ A = 0110 &\Rightarrow 6 - (2^{4-1} - 1) = -1 \Rightarrow a = -1 \\ A = 0101 &\Rightarrow 5 - (2^{4-1} - 1) = -2 \Rightarrow a = -2 \\ A = 0000 &\Rightarrow 0 - (2^{4-1} - 1) = -7 \Rightarrow a = -7 \end{aligned}$$

NB: Lo zero viene rappresentato una sola volta (come accade in compl. a 2).

Intervallo di rappresentabilità nella *rappresentazione con bias*

$$[-2^{(p-1)}+1, +2^{(p-1)}], \text{ ossia } [-bias, bias+1]$$

- $p = 4$ $[-7,+8]$ ($bias=7$)
- $p = 5$ $[-15,+16]$ ($bias=15$)
- $p = 7$ $[-63,+64]$ ($bias=63$)
- $p = 10$ $[-511,+512]$ ($bias=511$)
- $p = 14$ $[-8191,+8192]$ ($bias=8191$)

NB: prima di applicare l'algoritmo per $a \Rightarrow A$ occorre verificare che a sia rappresentabile su p bit, altrimenti l'algoritmo conduce a risultati sbagliati.

Ad esempio, tentando di rappresentare $a=-9$ su $p=4$ bit, si ottiene:
 $A=-9+(2^{4-1}-1)=-9+7 = -2$, che non è un numero naturale!

Numeri Interi (12/14)

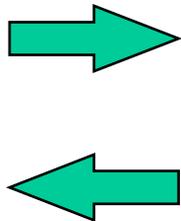
A	$\{0,1\}^5$	a^{ms}	a^{c2}	a^{bias}
0	00000	+0	0	-15
1	00001	+1	+1	-14
2	00010	+2	+2	-13
3	00011	+3	+3	-12
4	00100	+4	+4	-11
5	00101	+5	+5	-10
6	00110	+6	+6	-9
7	00111	+7	+7	-8
8	01000	+8	+8	-7
9	01001	+9	+9	-6
10	01010	+10	+10	-5
11	01011	+11	+11	-4
12	01100	+12	+12	-3
13	01101	+13	+13	-2
14	01110	+14	+14	-1
15	01111	+15	+15	0
16	10000	-0	-16	+1
17	10001	-1	-15	+2
18	10010	-2	-14	+3
19	10011	-3	-13	+4
20	10100	-4	-12	+5
21	10101	-5	-11	+6
22	10110	-6	-10	+7
23	10111	-7	-9	+8
24	11000	-8	-8	+9
25	11001	-9	-7	+10
26	11010	-10	-6	+11
27	11011	-11	-5	+12
28	11100	-12	-4	+13
29	11101	-13	-3	+14
30	11110	-14	-2	+15
31	11111	-15	-1	+16

**Rappresentazioni degli
interi su $p=5$ bit
messe a confronto**

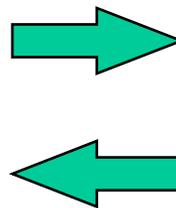
Osservazione:
*In complemento a due, $a=-1$
è sempre rappresentazione
dalla sequenza di p bit a 1
(11....11), qualunque sia il
valore di p .*

Numeri Interi (13/14)

$$\begin{array}{r} -2 \\ +1 \\ \hline -1 \end{array}$$



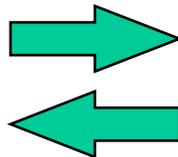
*compl.
a due*



1110
0001
1111

*Sommatore per
naturali*

*Operazioni su
numeri*



*Operazioni sulle
rappresentazioni*

QUESTO è il motivo per cui i calcolatori rappresentano gli interi in complemento a due: non occorre una nuova circuiteria per sommare e sottrarre numeri interi, viene utilizzata la stessa dei numeri naturali!

Numeri Interi (14/14)

Sommando due numeri interi si verifica un **overflow** quando i due numeri hanno lo stesso segno ed il risultato ha segno diverso

$$\begin{array}{r} 7+5=12 \\ \begin{array}{r} 0111 + (+7) \\ 0101 = (+5) \\ \hline 1100 = (-4) \text{ overflow!} \end{array} \end{array}$$

$$\begin{array}{r} -6-8=-14 \\ \begin{array}{r} 1010 + (-6) \\ 1000 = (-8) \\ \hline 10010 = (+2) \text{ overflow!} \end{array} \end{array}$$

$$\begin{array}{r} 7-1=6 \\ \begin{array}{r} 0111 + (+7) \\ 1111 = (-1) \\ \hline 10110 = (+6) \text{ corretto!} \end{array} \end{array}$$

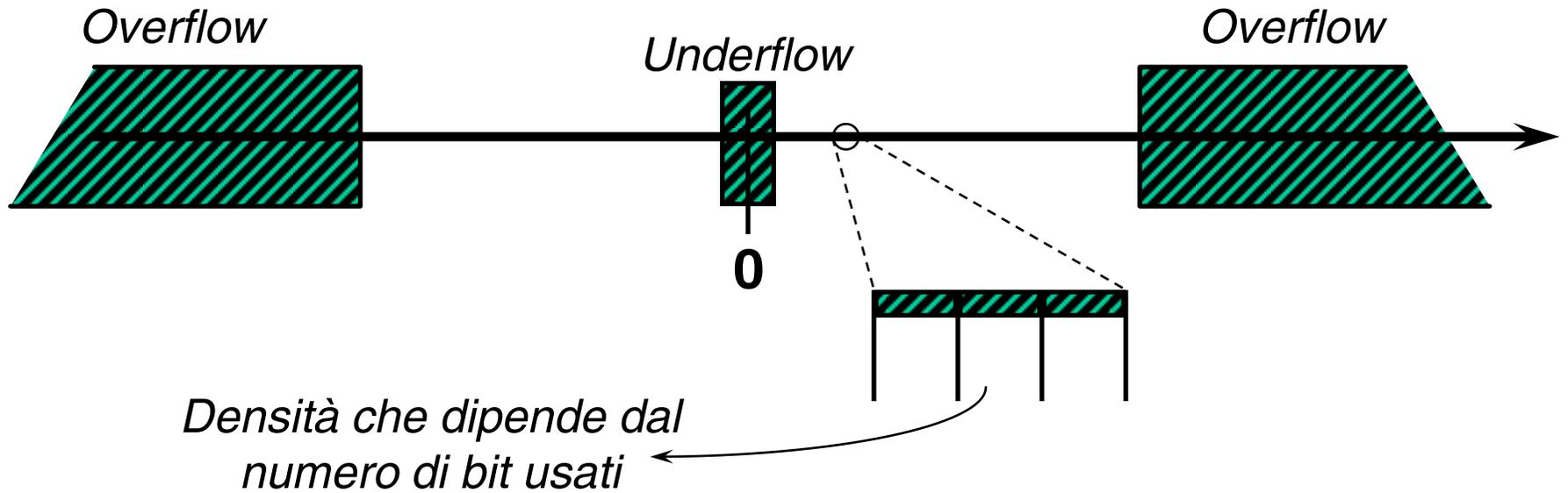
NB: L'eventuale (p+1)-esimo bit viene sempre scartato

RAPPRESENTAZIONE DEI NUMERI REALI

Numeri Reali

**Sottoinsieme
discreto** dei Numeri
Razionali

Sequenze
di bit



Numeri Reali – Virgola fissa (1/5)

- ❑ Si usa un numero fisso di bit per la parte intera ed un numero fisso di bit per la parte frazionaria.
- ❑ Sia r un numero reale da rappresentare. Di seguito, indichiamo con $I(r)$ la parte intera e con $F(r)$ la parte frazionaria di r
- ❑ Siano p i bit per rappresentare r : f per la parte frazionaria e $(p-f)$ i bit la parte intera:

$$R = a_{p-f-1} \dots a_0 a_{-1} \dots a_{-f} \quad r \cong \sum_{i=-f}^{p-f-1} a_i \beta^i = a_{p-f-1} \beta^{p-f-1} + \dots + a_0 \beta^0 + \underbrace{a_{-1} \beta^{-1} + \dots + a_{-f} \beta^{-f}}_{\substack{\text{parte} \\ \text{frazionaria}}}$$

Esempio: +1110.01 in base *due* vale +14.25 in base *dieci*

- ❑ NB: La virgola non si rappresenta
- ❑ La parte intera $I(r)$ si rappresenta con le tecniche note
- ❑ Per la parte frazionaria si usa il procedimento seguente:

Si usa la così detta procedura *parte frazionaria-parte intera*:

$$f_0 = F(r)$$

Se $f_0 \neq 0$ eseguire la seguente procedura iterativa:

$$f_{-1} = F(f_0 * 2) \quad a_{-1} = I(f_0 * 2)$$

$$f_{-2} = F(f_{-1} * 2) \quad a_{-2} = I(f_{-1} * 2)$$

...

fino a che f_j è uguale a zero oppure si è raggiunta la precisione desiderata

Esempio:

$$p = 16 \text{ e } f = 5$$

$$r = +331,6875$$

Numeri Reali – Virgola fissa(3/5)

QUOZIENTE	RESTO
331	-
165	1
82	1
41	0
20	1
10	0
5	0
2	1
1	0
0	1



- Dalla rappresentazione dei numeri naturali: $331 \Leftrightarrow 101001011$;

Numeri Reali – Virgola fissa(4/5)

F	I
$f_{.1}=F(0.6875*2=1.375)=0.375$	$a_{.1}=I(1.375)=1$
$f_{.2}=F(0.375*2=0.75)=0.75$	$a_{.2}=I(0.75)=0$
$f_{.2}=F(0.75*2=1.5)=0.5$	$a_{.3}=I(1.5)=1$
$f_{.3}=F(0.5*2=1.0)=0$	$a_{.4}=I(1.0)=1$



- $r = +10100101110110 \Rightarrow R = 0010100101110110$
- Parte frazionaria ->
 $1 * 2^{-1} + 0 * 2^{-2} + 1 * 2^{-3} + 1 * 2^{-4} \Rightarrow 0,5 + 0,125 + 0,0625 \Rightarrow 0,6875$
- Dall'algorithmo precedente: $0,6875 \Leftrightarrow 0,1011$
- $r = (+101001011,1011)$ base due

ERRORE DI TRONCAMENTO

Rappresentare $r = 2.3$ con $f = 6$.

F	I
$f_{.1}=F(0.3*2=0.6)=0.6$	$a_{.1}=I(0.6)=0$
$f_{.2}=F(0.6*2=1.2)=0.2$	$a_{.2}=I(1.2)=1$
$f_{.3}=F(0.2*2=0.4)=0.4$	$a_{.3}=I(0.4)=0$
$f_{.4}=F(0.4*2=0.8)=0.8$	$a_{.4}=I(0.8)=0$
$f_{.5}=F(0.8*2=1.6)=0.6$	$a_{.5}=I(1.6)=1$
$f_{.6}=F(0.6*2=1.2)=0.2$	$a_{.6}=I(1.2)=1$

Per esprimere r sono necessarie un numero infinito di cifre!

(10.0 1001 1001 1001... = 10.01001 dove 1001 è la parte periodica)

Ne abbiamo a disposizione solo 6. Quindi si opera un troncamento alla 6^a cifra dopo la virgola.

Quindi, in realtà si rappresenta non r ma il numero r'

$$r' = 10.010011$$

$$r' = 2 + 2^{-2} + 2^{-5} + 2^{-6} = 2 + 0.25 + 0.03125 + 0.015625 = 2.296875$$

L'errore di troncamento è $(2.3-2.296875)=0,00312499$ ($< 2^{-6} = 0.015625$)

CALCOLO DI FISICA ASTRONOMICA

$m_e = 9 \times 10^{-28}$ gr; $m_{\text{sole}} = 2 \times 10^{33}$ gr \Rightarrow Intervallo di variazione $\approx 10^{60}$.

Sarebbero quindi necessarie almeno 62 cifre, di cui 28 per la parte frazionaria.

Si hanno tante cifre perché l'intervallo da rappresentare è grande \Rightarrow si separa l'intervallo di rappresentabilità dalla precisione, cioè dal numero di cifre.

Notazione Scientifica

$$r = \pm m \cdot \beta^e \quad m = \text{mantissa}; e = \text{esponente}$$

L'intervallo di rappresentabilità è fissato dal numero di cifre dell'esponente.

L'accuratezza è determinata dal numero di cifre della mantissa.

Diverse rappresentazioni

3.14 0.314 10^{+1} 31.4 10^{-1}

Se ne sceglie una in particolare, che verrà chiamata **forma normalizzata**, in modo da avere una rappresentazione unica

Rappresentazione di un numero binario in virgola mobile:

+1110.01 numero reale binario in virgola fissa



esponenti espressi in binario

$+1.11001 \cdot due^{+11}$ $+111001 \cdot due^{-10}$

(alcune possibili rappresentazioni in virgola mobile)

Numeri reali *normalizzati*:

- mantissa con parte intera costituita da un solo bit di valore 1
- rappresentazione è una tripla costituita da tre numeri naturali

$$r \leftrightarrow R = \langle s, E, F \rangle$$

Standard IEEE 754-1985

La rappresentazione R è composta da tre naturali (s , E ed F), dove:

s = codifica del segno (1 bit)

F = codifica della parte frazionaria della mantissa su G bit

E = codifica dell'esponente su K bit

$$r = (s == 0)? [(1+f) \cdot 2^e] : [-(1+f) \cdot 2^e]$$

$f = F/2^G$ è la parte frazionaria della mantissa ($m=1+f= 1+F/2^G$)

$e = +E - (2^{K-1} - 1)$ è l'esponente rappresentato dal numero naturale E (*rappresentazione con polarizzazione*)

Conseguenza dell' "uno implicito" \Rightarrow *lo zero non è rappresentabile!*

Half precision: 16 bit, $K = 5$ e $G = 10$. Esempi di trasformazione $\mathbf{R} \Rightarrow \mathbf{r}$

Esempio 1

$\mathbf{R} = \{1,10011,1110100101\}$ rappresenta il numero reale negativo con:

$$f = F/2^G = F/2^{\text{dieci}} = 0.1110100101$$

$$e = E - (2^{K-1} - 1) = +10011 - (+01111) = +100 \quad (\text{bias}=\text{quindici})$$

$$\mathbf{r} = -1.1110100101 \cdot \text{due}^{+\text{quattro}} = -11110.100101$$

$$\mathbf{r} = -30.578125 \text{ in base } \text{dieci}$$

Esempio 2

$\mathbf{R} = \{0,01111,0000000001\}$ rappresenta il numero reale positivo:

$$f = F/2^M = 1/2^G = \text{due}^{-\text{dieci}} = 0.0009765625$$

$$e = E - (2^{K-1} - 1) = 01111 - 01111 = 15 - 15 = 0$$

$$\mathbf{r} = +1.0000000001 \cdot \text{due}^{+\text{zero}} = 2^0 + 2^{-10} = 1 + 0.0009765625 = 1.0009765625$$

$$\mathbf{r} = + 1.0009765625 \text{ in base } \text{dieci}$$

Half precision, esempi di trasformazione $r \Rightarrow R$

Esempio A – Rappresentare $r=2$ in half precision

La rappresentazione di $r=2$ è $R = \{0,10000,0000000000\}$. Infatti, decodificando:

$$f = F/2^G = 0$$

$$e = E - (2^{K-1} - 1) = 10000 - 01111 = 1$$

$$r = +1.0000000000 \cdot due^{+1} = 2 \text{ in base dieci}$$

Esempio B – Rappresentare $r=0.5$ in half precision

La rappresentazione di $r = 0.5$ è $R = \{0,01110,0000000000\}$. Infatti, decodificando:

$$f = F/2^G = 0$$

$$e = E - (2^{K-1} - 1) = 01110 - 01111 = -1$$

$$r = +1.0000000000 \cdot due^{-1} = 0.5 \text{ in base dieci}$$

Half precision, esempi di trasformazione $r \Rightarrow R$

Esempio C – Rappresentare $r=2.3$ in half precision

Sapendo che r ha rappresentazione in virgola fissa $10.0\underline{1001}$,
 iniziamo a portarlo in forma normalizzata: $1.00\underline{1001} \cdot due^{+1}$

Ora ci servono 10 cifre per la parte frazionaria della mantissa, le
 restanti verranno scartate, provocando il consueto *errore di*
troncamento:

$$1.00 \underbrace{1001 1001}_{10 \text{ cifre}} \cancel{1001 1001} \dots \cdot due^{+1}$$

F (prime 10 cifre a destra della virgola. Le altre vengono ignorate)

A questo punto la rappresentazione è immediata (per trovare E si
 faccia riferimento all'esempio A: $\mathbf{R}=\{0,10000,0010011001\}$)

Esempio (numeri con modulo massimo, i cosiddetti $\pm \infty$):

$\mathbf{R} = \{0, 11111, 1111111111\}$ (massimo numero rappresentabile, $+\infty$)

$\mathbf{R} = \{1, 11111, 1111111111\}$ (minimo numero rappresentabile, $-\infty$)

$$f = F/2^G = F/2^{\text{dieci}} = 0.1111111111$$

$$e = +E - (2^{K-1} - 1) = +11111 - (+01111) = +10000 \text{ (+sedici)}$$

$$|\mathbf{r}| = 1.1111111111 \cdot \text{due}^{+10000} < 2^{(2^{(K-1)+1})} = \mathbf{2^{\text{bias}+2}} = 2^{17} = 131072$$

$$|\mathbf{r}| = 131008 \cong 1.3 \cdot 10^{+5} \text{ in base } \textit{dieci}$$

Riassumendo, nella rappresentazione *half precision*:

il « $+\infty$ » corrisponde a $\cong 2^{+17}$

il « $-\infty$ » corrisponde a $\cong -2^{+17}$

NB: L'intervallo di rappresentabilità $[-\infty, +\infty]$ è approssimabile dal seguente: $[\cong -2^{\text{bias}+2}, \cong 2^{\text{bias}+2}]$

Esempio (numeri con modulo minimo, i cosiddetti ± 0):

$\mathbf{R} = \{0, 00000,00000000000\}$ (minimo numero positivo, +0)

$\mathbf{R} = \{1,00000,00000000000\}$ (massimo numero negativo, -0)

$$f = F/2^G = F/2^{\text{dieci}} = 0.00000000000$$

$$e = + E -(2^{K-1} - 1) = +00000 - (+01111) = -01111 = \text{-quindici}$$

$$|\mathbf{r}| = 1.00000000000 \cdot \text{due}^{-01111} = \mathbf{2^{-bias}}$$

$$|\mathbf{r}| = 2^{-15} \cong 0.31 \cdot 10^{-4}$$

Riassumendo, nella rappresentazione *half precision*:

il «+0» corrisponde a $2^{-15} \cong 0.31 \cdot 10^{-4}$

il «-0» corrisponde a $-2^{-15} \cong -0.31 \cdot 10^{-4}$

Numeri Reali – Virgola mobile(9/10)

Standard IEEE 754-1985 prevede:

Rappresentazione in **single precision** su **32 bit** con **K = 8** e **G = 23**

Intervallo di rappresentabilità:

$$\begin{array}{l} \text{massimo modulo} \quad \approx 2^{(bias+2)} = 2^{+129} \approx 6.8 \cdot 10^{+38} \\ \text{minimo modulo} \quad = 2^{-bias} = 2^{-127} \approx 0.58 \cdot 10^{-38} \end{array}$$

Rappresentazione in **double precision** su **64 bit** con **K = 11** e **G = 52**

Intervallo di rappresentabilità:

$$\begin{array}{l} \text{massimo modulo} \quad \approx 2^{(bias+2)} = 2^{+1025} \approx 3.6 \cdot 10^{+308} \\ \text{minimo modulo} \quad = 2^{-bias} = 2^{-1023} \approx 1.1 \cdot 10^{-308} \end{array}$$

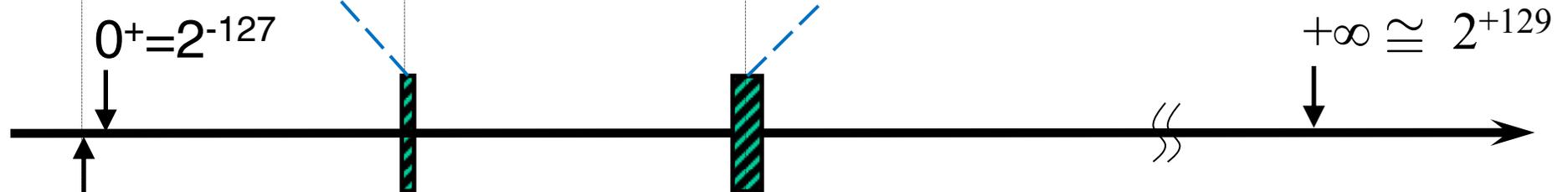
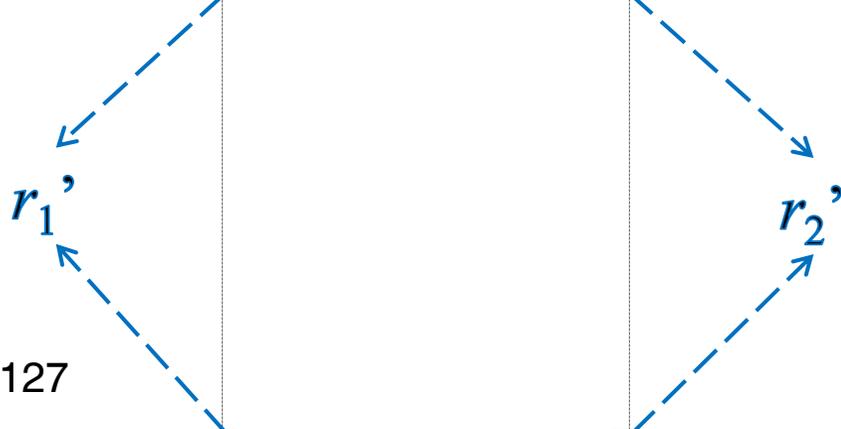
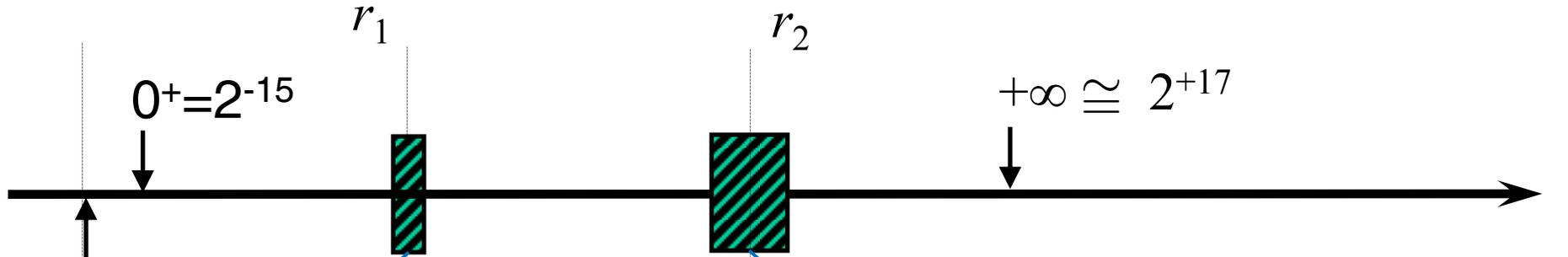
IEEE 754-2008 aggiunge:

rappresentazione in **quadruple precision** su **128 bit** con **K = 15** e **G = 112**

$$\begin{array}{l} \text{massimo modulo} \quad \approx 2^{(bias+2)} = 2^{+16385} \approx 1.2 \cdot 10^{+4932} \\ \text{minimo modulo} \quad = 2^{-bias} = 2^{-16383} \approx 1.6 \cdot 10^{-4932} \end{array}$$

Numeri Reali - Virgola mobile (10/10)

half precision



single precision

INTRODUZIONE ALL'INFORMATICA

Informatica (definizione informale): è la scienza della rappresentazione e dell'elaborazione dell'informazione

Informatica (definizione formale dell'Association for Computing Machinery - ACM): è lo studio sistematico degli algoritmi che descrivono e trasformano l'informazione, la loro teoria e analisi, il loro progetto, e la loro efficienza, realizzazione e applicazione.

Algoritmo: sequenza precisa e finita di operazioni, comprensibili e perciò eseguibili da uno strumento informatico, che portano alla realizzazione di un compito.

Esempi di algoritmi:

- Istruzioni di montaggio di un elettrodomestico
- Somma in colonna di due numeri
- Bancomat

Compito dell'esperto informatico: data la descrizione di un problema, produrre algoritmi (cioè capire la sequenza di passi che portano alla soluzione del problema) e codificarli in programmi.

- La descrizione di un problema non fornisce in generale un modo per risolverlo.
- La descrizione del problema deve essere chiara e completa.

Calcolatori Elettronici come esecutori di algoritmi: gli algoritmi vengono descritti tramite programmi, cioè sequenze di istruzioni scritte in un opportuno linguaggio comprensibile al calcolatore.

Algoritmo (1)

Algoritmo: sequenza precisa (non ambigua) e finita di operazioni, che portano alla realizzazione di un compito.

Le operazioni utilizzate appartengono ad una delle seguenti categorie:

1. Operazioni sequenziali

Realizzano una singola azione. Quando l'azione è terminata passano all'operazione successiva.

2. Operazioni condizionali

Controllano una condizione. In base al valore della condizione, selezionano l'operazione successiva da eseguire.

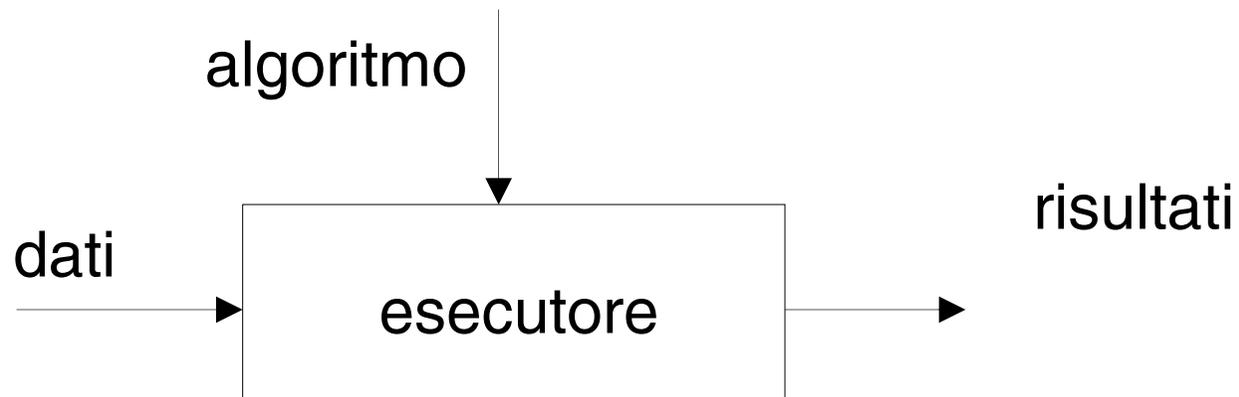
3. Operazioni iterative

Ripetono l'esecuzione di un blocco di operazioni, finchè non è verificata una determinata condizione.

Algoritmo (2)

L'esecuzione delle azioni nell'ordine specificato dall'algoritmo consente di risolvere il problema.

Risolvere il problema significa produrre risultati a partire da dati in ingresso



L'algoritmo deve essere applicabile ad un qualsiasi insieme di dati in ingresso appartenenti al dominio di definizione dell'algoritmo (se l'algoritmo si applica ai numeri interi deve essere corretto sia per gli interi positivi che per gli interi negativi)

Calcolo equazione $ax+b = 0$

- leggi i valori di a e di b
- calcola $-b$
- dividi quello che hai ottenuto per a e chiama x il risultato
- stampa x

Calcolo del massimo fra due numeri

- leggi i valori di a e di b
- **se** $a > b$ stampa a **altrimenti** stampa b

Algoritmo (4)

Calcolo del massimo di un insieme

- scegli un elemento come massimo provvisorio *max*
- per ogni elemento *i* dell'insieme:
 - se** $i > max$ eleggi *i* come nuovo massimo provvisorio *max*
- il risultato è *max*

Stabilire se una parola *P* precede alfabeticamente una parola *Q*.

Ipotesi: *P* e *Q* di uguale lunghezza ≥ 1

leggi *P* e *Q*; **inizializza** trovato a 0

- **ripeti finché** (trovato vale 0 e lettere non finite)
 - se** prima lettera di *P* < prima lettera di *Q*
 - allora** scrivi vero; trovato = 1;
 - altrimenti se** prima lettera di *P* > prima lettera di *Q*
 - allora** scrivi falso; trovato = 1;
 - altrimenti** togli da *P* e da *Q* la prima lettera
- **se** trovato vale 0 **allora** scrivi falso

Algoritmo (5)

Eseguibilità: ogni azione deve essere eseguibile dall'esecutore in un tempo finito

Non-ambiguità: ogni azione deve essere univocamente interpretabile dall'esecutore

Finitezza: il numero totale di azioni da eseguire, per ogni insieme di dati in ingresso, deve essere finito

Algoritmi equivalenti

- ◆ hanno lo stesso dominio di ingresso
- ◆ hanno lo stesso dominio di uscita
- ◆ in corrispondenza degli stessi valori del dominio di ingresso producono gli stessi valori del dominio di uscita

- ◆ Due algoritmi equivalenti
 - Forniscono lo stesso risultato, ma possono avere diversa efficienza e possono essere profondamente diversi

Algoritmo (6)

Esempio: calcolo del Massimo Comun Divisore (MCD) fra due interi M e N

Algoritmo 1

- Calcola l'insieme A dei divisori di M
- Calcola l'insieme B dei divisori di N
- Calcola l'insieme C dei divisori comuni
- il massimo comun divisore è il massimo divisore contenuto in C

Algoritmo 2 (di Euclide)

Se due numeri, m e n , sono divisibili per un terzo numero, x , allora anche la loro differenza è divisibile per x .

Per dimostrarla, si può utilizzare la proprietà distributiva.

Supponiamo $m > n$.

$$m = kx$$

$$n = hx$$

$$m - n = kx - hx = x(k - h)$$

Dunque si può dire che: **$\text{MCD}(m, n) = \text{MCD}((m - n), n)$**

Algoritmo

- **ripeti finché** ($M \neq N$):
 - **se** $M > N$, sostituisci a M il valore $M - N$
 - **altrimenti** sostituisci a N il valore $N - M$
- il massimo comun divisore corrisponde a M (o a N)

Proprietà essenziali degli algoritmi:

Correttezza:

- un algoritmo è corretto se esso perviene alla soluzione del compito cui è preposto, senza difettare in alcun passo fondamentale.

Efficienza:

- un algoritmo è efficiente se perviene alla soluzione del compito cui è preposto nel modo più veloce possibile, compatibilmente con la sua correttezza.

Programmazione

La formulazione testuale di un algoritmo in un linguaggio comprensibile ad un calcolatore è detta **PROGRAMMA**.

Ricapitolando, per risolvere un problema:

- Individuazione di un procedimento risolutivo
- Scomposizione del procedimento in un insieme ordinato di azioni – **ALGORITMO**
- Rappresentazione dei dati e dell'algoritmo attraverso un formalismo comprensibile al calcolatore: **LINGUAGGIO DI PROGRAMMAZIONE**

Linguaggi di Programmazione (1)

Perché non usare direttamente il linguaggio naturale?

Il LINGUAGGIO NATURALE è un insieme di parole e di regole per combinare tali parole che sono usate e comprese da una comunità di persone

- non evita le ambiguità
- non si presta a descrivere processi computazionali automatizzabili

Occorre una nozione di linguaggio più precisa.

Un LINGUAGGIO di programmazione è una notazione formale che può essere usata per descrivere algoritmi.

Si può stabilire quali sono gli elementi linguistici primitivi, quali sono le frasi lecite e se una frase appartiene al linguaggio.

Linguaggi di Programmazione (2)

Un linguaggio è caratterizzato da:

SINTASSI - insieme di regole formali per la scrittura di programmi, che fissano le modalità per costruire frasi corrette nel linguaggio

SEMANTICA - insieme dei significati da attribuire alle frasi (sintatticamente corrette) costruite nel linguaggio

- a parole (poco precisa e ambigua)
- mediante azioni (semantica operativa)
- mediante funzioni matematiche (semantica denotazionale)
- mediante formule logiche (semantica assiomatica)

Definizione di linguaggio

Alfabeto V (lessico)

- insieme dei simboli con cui si costruiscono le frasi

Universo linguistico V^*

- insieme di tutte le frasi (sequenze finite) di elementi di V

Linguaggio L su alfabeto V

- un sottoinsieme di V^*

Come definire il sottoinsieme di V^* che definisce il linguaggio?

Specificando in modo preciso la sintassi delle frasi del linguaggio TRAMITE una **grammatica formale**

Grammatiche

Grammatica $G = \langle V, VN, P, S \rangle$

V	insieme finito di simboli terminali
VN	insieme finito di simboli non terminali
P	insieme finito di regole di produzione
S	simbolo non terminale detto simbolo iniziale

Data una grammatica G , si dice Linguaggio LG generato da G l'insieme delle frasi di V

- Derivabili dal simbolo iniziale S
- Applicando le regole di produzione P

Le frasi di un linguaggio di programmazione vengono dette programmi di tale linguaggio.

Grammatica BNF (1)

GRAMMATICA BNF (Backus-Naur Form) è una grammatica le cui regole di produzione sono della forma

$$X ::= A$$

X simbolo non terminale

A sequenza di simboli (terminali e non terminali)

Una grammatica BNF definisce quindi un linguaggio sull'alfabeto terminale V mediante un meccanismo di derivazione (riscrittura)

A deriva da X se esiste una sequenza di derivazioni da X ad A

$$X ::= A_1$$
$$A_2$$
$$\dots$$
$$A_n$$

unica regola che indica l'**alternativa** fra A_1, \dots, A_n

Grammatica BNF (2)

$G = \langle V, VN, P, S \rangle$

$V = \{ \text{lupo, canarino, bosco, cielo, mangia, vola, canta, ., il, lo} \}$

$VN = \{ \text{frase, soggetto, verbo, articolo, nome} \}$

$S = \text{frase}$

Produzioni P

frase ::= **soggetto verbo .**

soggetto ::= **articolo nome**

articolo ::= il
lo

nome ::= lupo
canarino
bosco
cielo

verbo ::= mangia
vola
canta

Esempio: derivazione della frase
"il lupo mangia."

frase -> **soggetto verbo.**

-> **articolo nome verbo.**

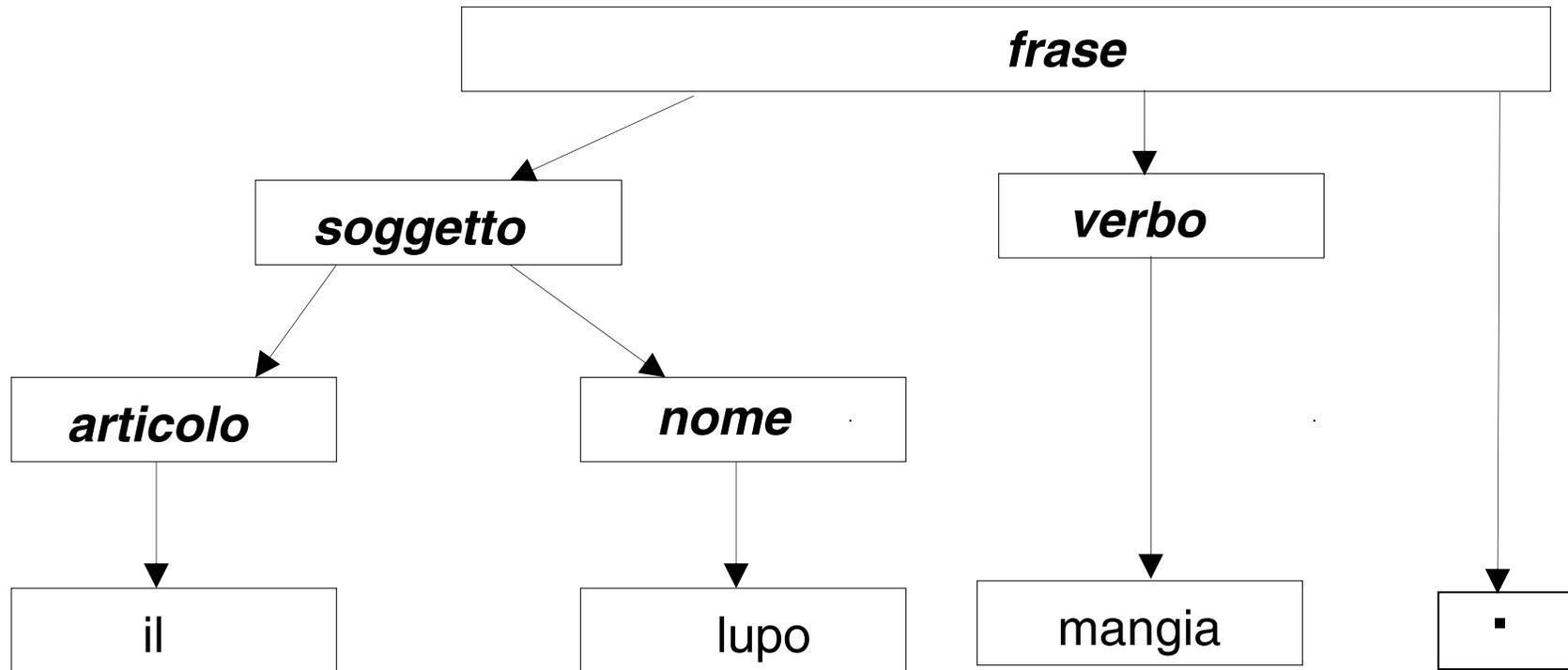
-> il **nome verbo.**

-> il lupo **verbo.**

-> il lupo mangia.

derivazione left-most

Albero sintattico



Albero sintattico: albero che esprime il processo di derivazione di una frase usando una data grammatica

Esistono programmi per generare analizzatori sintattici per linguaggi descritti con BNF

Sintassi e semantica

Scrivere un programma sintatticamente corretto non implica che il programma faccia quello per cui è stato scritto

La frase “il lupo vola” è sintatticamente corretta ma non è semanticamente corretta (non è significativa)

// Somma di due numeri interi inseriti da tastiera

```
int main()
{
    int a, b;
    cout << “immettere due numeri” << endl;
    cin >> a; cin >> b;
    int c = a + a; ←
    cout << “Somma: “ << c << endl;
    return 0;
}
```

Approccio compilato

Sviluppo di un programma (approccio compilato):

- editing: scrivere il testo e memorizzarlo su supporti di memoria permanenti
- compilazione
- linking
- esecuzione

Compilatore: traduce il programma sorgente in programma oggetto

- ◆ ANALISI programma sorgente
 - analisi lessicale
 - analisi sintattica
- ◆ TRADUZIONE
 - generazione del codice
 - ottimizzazione del codice

Esempi: C, C++, Fortran, Pascal, ecc...

Sviluppo di un programma (approccio interpretato):

- editing: scrivere il testo e memorizzarlo su supporti di memoria permanenti
- interpretazione
 - ◆ ANALISI programma sorgente
 - analisi lessicale
 - analisi sintattica
 - ◆ ESECUZIONE

ESEMPIO: Matlab, Python, Java, Javascript, R, Visual Basic, ecc...