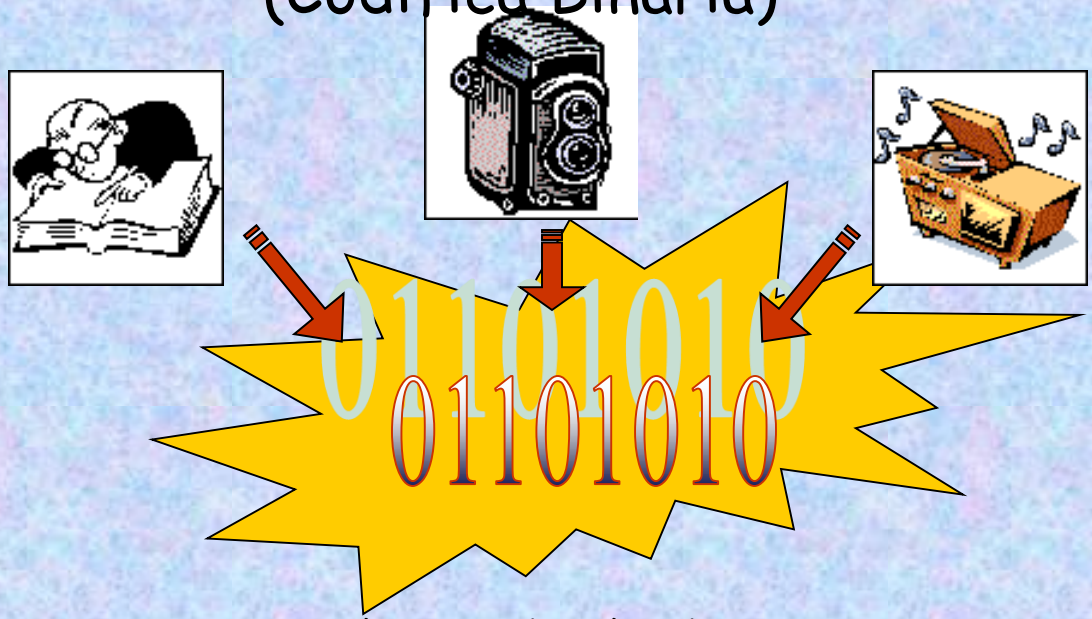
Rappresentazione dell'Informazione

CODIFICA DELL'INFORMAZIONE

(Codifica Binaria)



Bit	0/1 (si/no)
Byte	00010010 (8 bit)
Kilobyte	2^{10} = 1024 byte
Megabyte	2^{20} ~ 1.000.000 byte
Gigabyte	2^{30} ~ 1.000.000.000 byte

Rappresentazione dei Naturali

$N = 0,1,2,\ldots$

La Notazione Posizionale (in base p)

$N_p \equiv a_n\ a_{n-1}\ a_{n-2}\ \ldots\ a_1\ a_0$

$N_p = a_n \times p^n + a_{n-1} \times p^{n-1} + \ldots + a_1 \times p + a_0$

Esempio (base 10)

$543 = 5 \times 10^2 + 4 \times 10^1 + 3 \times 10^0 = 500 + 40 + 3$

Rappresentazione dei Naturali

$N = 0,1,2,\ldots$

La Notazione Additiva (numeri romani)

$N_p \equiv a_n\, a_{n-1}\, a_{n-2} \ldots a_1\, a_0$

$N_p = a_n + a_{n-1} + \ldots + a_1 + a_0$

I = 1, V = 5, X = 10, L = 50,
C = 100, D = 500, M = 1000

Esempio

DCCCII = D + C + C + C + I + I
= 500 + 100 + 100 + 100 + 1 + 1 = 802

Le Notazioni Usate in Informatica

Binaria (base 2) $a_i = 0,1$

Ottale (base 8) $a_i = 0,1,2,3,4,5,6,7$

Esadecimale (base 16) $a_i = 0,1,2,\ldots,9,A,B,C,D,E,F$

Decimale	Binaria	Ottale	Esadec.
10	1010	12	A
124	1111100	174	7C

Codifica e decodifica

(Da binario a decimale e viceversa)

$N_2 = 101010$

$N_{10} = 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

$= 32 + 8 + 2 = 42$

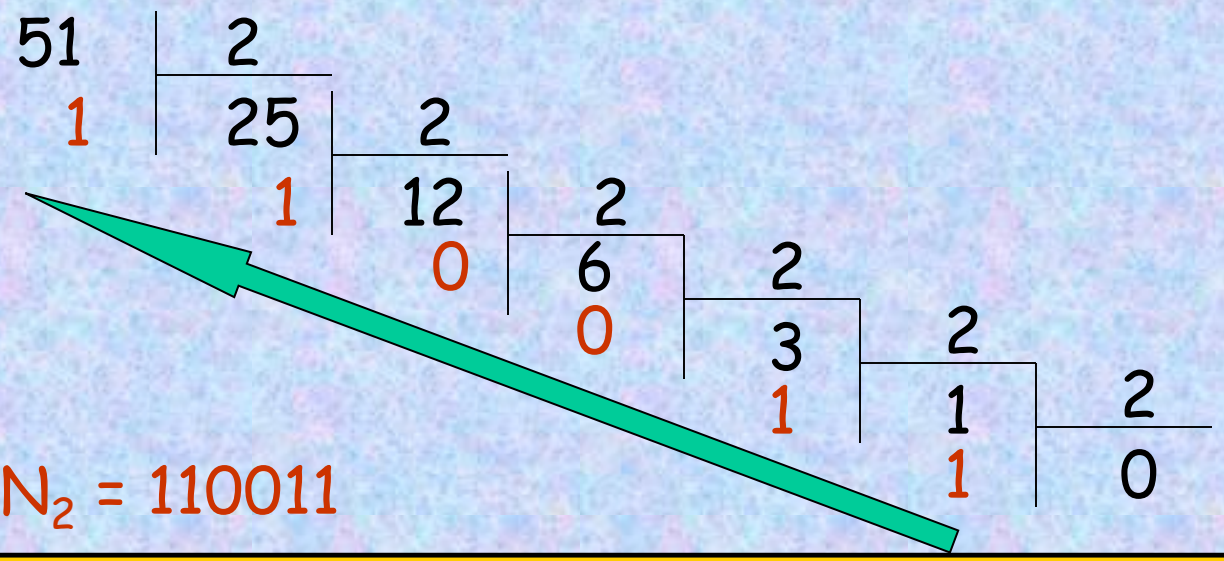
$N_2 = 11011$

$N_{10} = \quad 2^0 + 2^1 + 2^3 + 2^4 \quad = 1 + 2 + 8 + 16 = 27$

Codifica e decodifica

Da decimale a binario – **conversione per divisione**

$N_{10} = 51$ $N_2 = ???$



$51 = 2x25+1 = 2x(2x12+1)+1 = 2x(2x(2x6+0)+1)+1 =$

$= 2x(2x(2x(2x3+0)+0)+1)+1 =$

$= 2x(2x(2x(2x(2x1+1)+0)+0)+1)+1 = 2^5+2^4+2^1+2^0$

L'algoritmo di conversione per moltiplicazione mi permette di effettuare la conversione in base, anche per numeri non interi. Per un numero decimale, ad esempio 0,7 funziona in questa maniera:

N = 0,7 **B = 2**

0,7 * 2 =	1	+ 0,4
0,4 * 2 =	0	+ 0,8
0,8 * 2 =	1	+ 0,6
0,6 * 2 =	1	+ 0,2
0,2 * 2 =	0	+ 0,4
0,4 * 2 =	0	+ 0,8

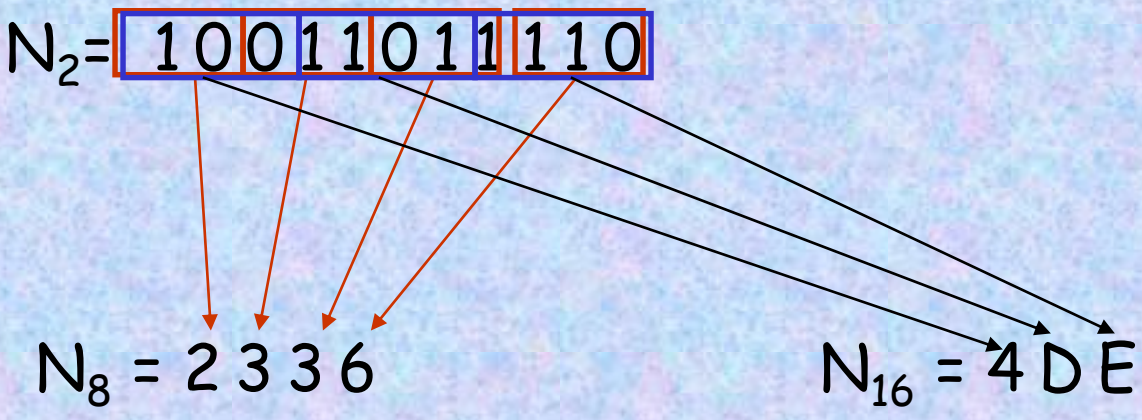
E così via …

più si va avanti e più alta e l'approssimazione.

Il procedimento di conversione per moltiplicazione è il solo che ci consente di effettuare la rappresentazione di un numero razionale non negativo. ***Inoltre per quanto riguarda la parte intera del numero posso adottare sia l'algoritmo di conversione per divisione sia quello per moltiplicazione.***

Da Binario a Ottale

Da Binario a esadecimale



Per trasformare un numero binario in ottale, operiamo la trasformazione considerando gruppi di 3 bit

Per trasformare un numero binario in esadecimale, operiamo la trasformazione considerando gruppi di 4 bit

Rappresentazione degli Interi

N = 0,+1,-1,+2,-2,+3,-3,…

Come possiamo rappresentare il *segno* di un numero?

Aggiungiamo un ulteriore bit che
poniamo a 1 se il numero è negativo!

Esempio

N_{10} = +14 N_2 = 01110

N_{10} = -14 N_2 = 11110

Con k bit si possono codificare tutti gli interi

$$-2^{k-1}+1 \leq N \leq 2^{k-1}-1$$

Esistono due codifiche dello 0

0000…0 1000…0

Rappresentazione degli Interi

(Rappresentazione in complemento a 2)

Supponiamo di avere a disposizione k bit

La rappresentazione di –N si ottiene facendo la conversione in binario del numero 2^k-N

Con k bit si possono codificare tutti gli interi

$$-2^{k-1} \leq N \leq 2^{k-1}-1$$

$0000\ldots0_2 \equiv 0_{10}$

$1000\ldots0_2 \equiv -2^k{}_{10}$

Esempio (con 5 bit)

$N_{10} = +14 \qquad N_2 = 01110$

$N_{10} = -14 \qquad 2^k-14 = 18 \qquad N_2 = 10010$

Rappresentazione di interi in complemento a due

Supponiamo di avere a disposizione una cella di otto bit compreso il bit di segno. Il bit di segno può assumere due soli simboli: **0** quando il numero è **positivo** e **1** quando il numero è **negativo**.

Regola del complemento a due?

A partire da destra verso sinistra il primo uno che incontriamo non si complementa invece tutti gli altri bit si complementano.

Premettendo che su un byte il più piccolo numero rappresentabile è –128, e il più grande è + 127.

Rappresentiamo prima +15:

(0)	0	0	0	1	1	1	1

Rappresentiamo adesso -15 con la tecnica del complemento a 2:

(1)	1	1	1	0	0	0	1

Rappresentazione di interi in complemento a uno

Supponiamo di avere a disposizione una cella di otto bit compreso il bit di segno. Il bit di segno può assumere due soli simboli: **0** quando il numero è **positivo** e **1** quando il numero è **negativo**.

Regola del complemento a uno?

Si complementano tutti i bit, lo 0 diventa 1 e1 diventa 0.

In questo caso con un byte il più piccolo numero rappresentabile è –127, e il più grande è + 127 otteniamo quindi una rappresentazione in meno

Rappresentiamo prima +15:

(0)	0	0	0	1	1	1	1

Rappresentiamo adesso -15 con la tecnica del complemento a 2:

(1)	1	1	1	0	0	0	0

VANTAGGI/SVANTAGGI

COMPLEMENTO A 1

- Facilità nella implementazione della complementazione
- Effettuare degli aggiustamenti nei risultati delle operazioni matematiche

COMPLEMENTO A 2

- Complessità nell'implementazione della complementazione
- Non bisogna fare nessuna correzione

Regola generale della somma di tre bit:

a_j	B_j	c_j	S_j	C_{j+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Esercizio n.1: Rappresentare in complemento a 1 e a 2 la somma di A=+100 e B=-47

Complemento a 1

	1	1	0	0	0	0	0	0	0
A=+100		(0)	1	1	0	0	1	0	0
B=-47		(1)	1	0	1	0	0	0	0
+53		(0)	0	1	1	0	1	0	0

Regola:
Se A+B>0 allora effettuare la correzione aggiungendo 1 al risultato

Complemento a 2

	1	1	0	0	0	0	0	0	0
A=+100		(0)	1	1	0	0	1	0	0
B=-47		(1)	1	0	1	0	0	0	1
+53		(0)	0	1	1	0	1	0	1

Esercizio n.2: Rappresentare in complemento a 1 e a 2 la somma di A=-100 e B=+47

	0	0	1	1	1	1	1	1	0
A=-100		(1)	0	0	1	1	0	1	1
B=+47		(0)	0	1	0	1	1	1	1
-53		(1)	1	0	0	1	0	1	0
+53		(0)	0	1	1	0	1	0	1

	0	0	1	1	1	1	0	0	0
A=-100		(1)	0	0	1	1	1	0	0
B=+47		(0)	0	1	0	1	1	1	1
-53		(1)	1	0	0	1	0	1	1
+53		(0)	0	1	1	0	1	0	1

OVERFLOW E UNDERFLOW

Esercizio n.3: Rappresentare in complemento a 1 e a 2 la somma di A=+100 e B=+28

Complemento a 1

	0	**1**	**1**	**1**	**1**	**0**	**0**	**0**
A=+100	(0)	1	1	0	0	1	0	0
B=+28	(0)	0	0	1	1	1	0	0
+128 ?	(1)	0	0	0	0	0	0	0
OVERFLOW								

Complemento a 2

	0	**1**	**1**	**1**	**1**	**1**	**0**	**0**	**0**
A=+100	(0)	1	1	0	0	1	0	0	
B=+28	(0)	0	0	1	1	1	0	0	
+128 ?	(1)	0	0	0	0	0	0	0	
OVERFLOW									

Esercizio n.4: Rappresentare in complemento a 1 e a 2 la somma di A=-100 e B=-28

	1	**0**	**0**	**0**	**0**	**1**	**1**	**0**
A=-100	(1)	0	0	1	1	0	1	1
B=-28	(1)	1	1	0	0	0	1	1
-128 ?	(0)	1	1	1	1	1	1	0
UNDERFLOW								

	1	**1**	**1**	**1**	**1**	**1**	**0**	**0**	**0**
A=-100	(1)	0	0	1	1	1	0	0	
B=-28	(1)	1	1	0	0	1	0	0	
-128	(1)	0	0	0	0	0	0	0	
é il più piccolo rappresentabile in C2, è l'unico che non ha il suo complemento									

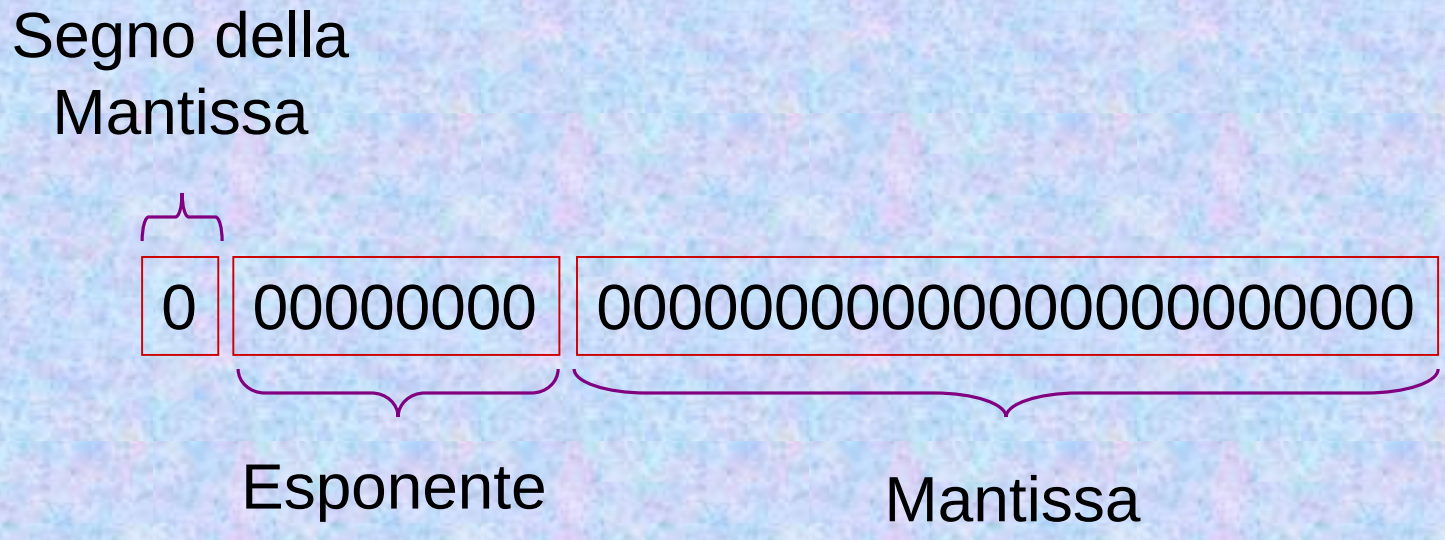
Rappresentare numeri razionali

- Il numero di bit utilizzati per rappresentare i numeri in un computer è fissato
  - Con 8 bit si rappresentano i numeri da 0 a 255 (oppure da –128 a 127)
- Rappresentare i numeri razionali con la virgola in posizione fissa è un inutile spreco di bit (34.56 = 0000034.5600000000)
- La precisione varia a seconda del valore del numero

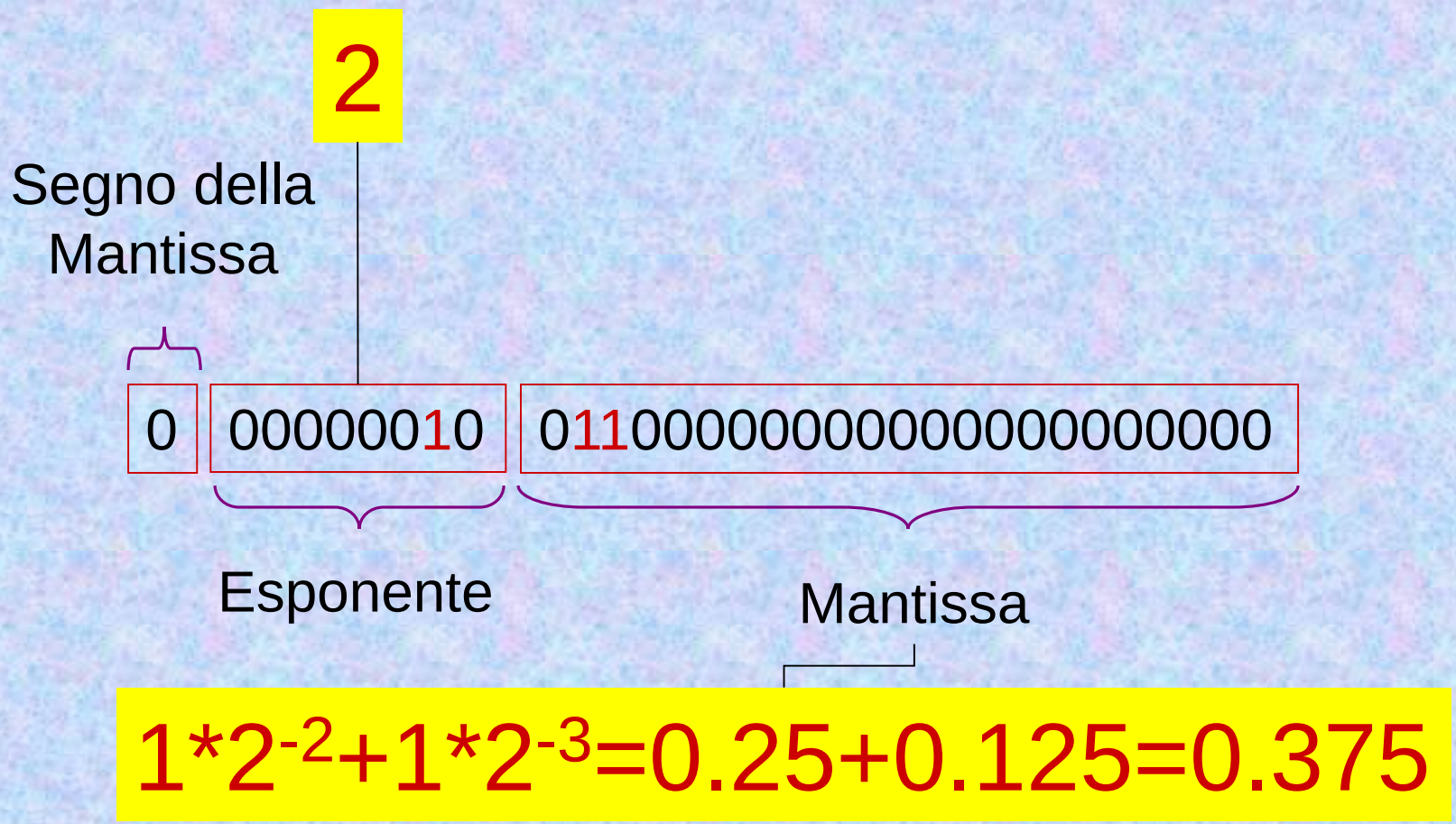
Rappresentazione in Virgola Mobile

- I numeri si devono esprimere utilizzando il maggior numero di cifre significative → virgola mobile
- I numeri si rappresentano in notazione esponenziale con mantissa ed esponente
- Si assegnano pochi bit all'esponente e il resto (m) alla mantissa
- La mantissa è compresa tra 0 e $1-2^{-m}$

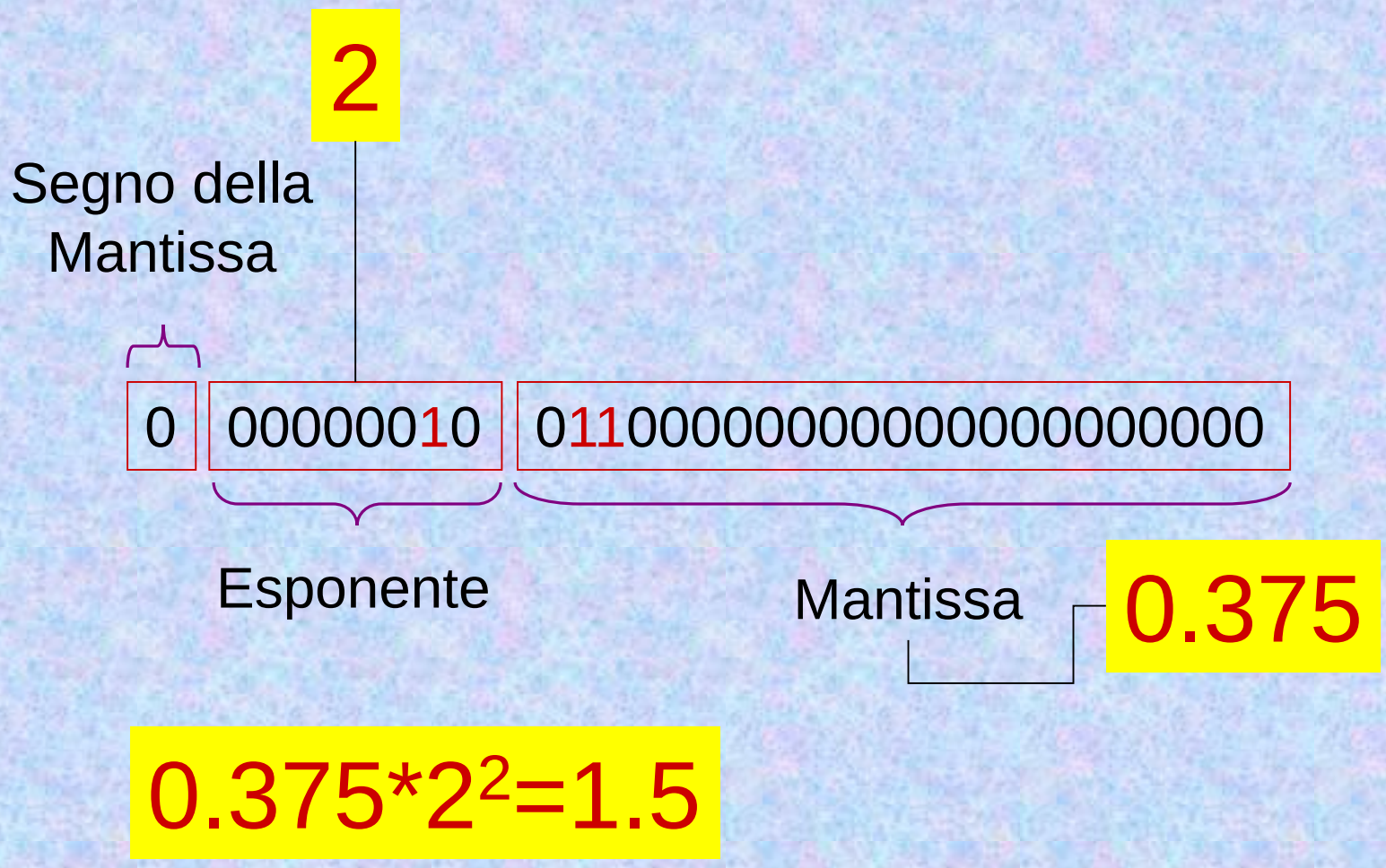
Rappresentazione in virgola mobile



virgola mobile: un esempio



virgola mobile: un esempio



$0.375*2^2=1.5$

Operazioni Logiche

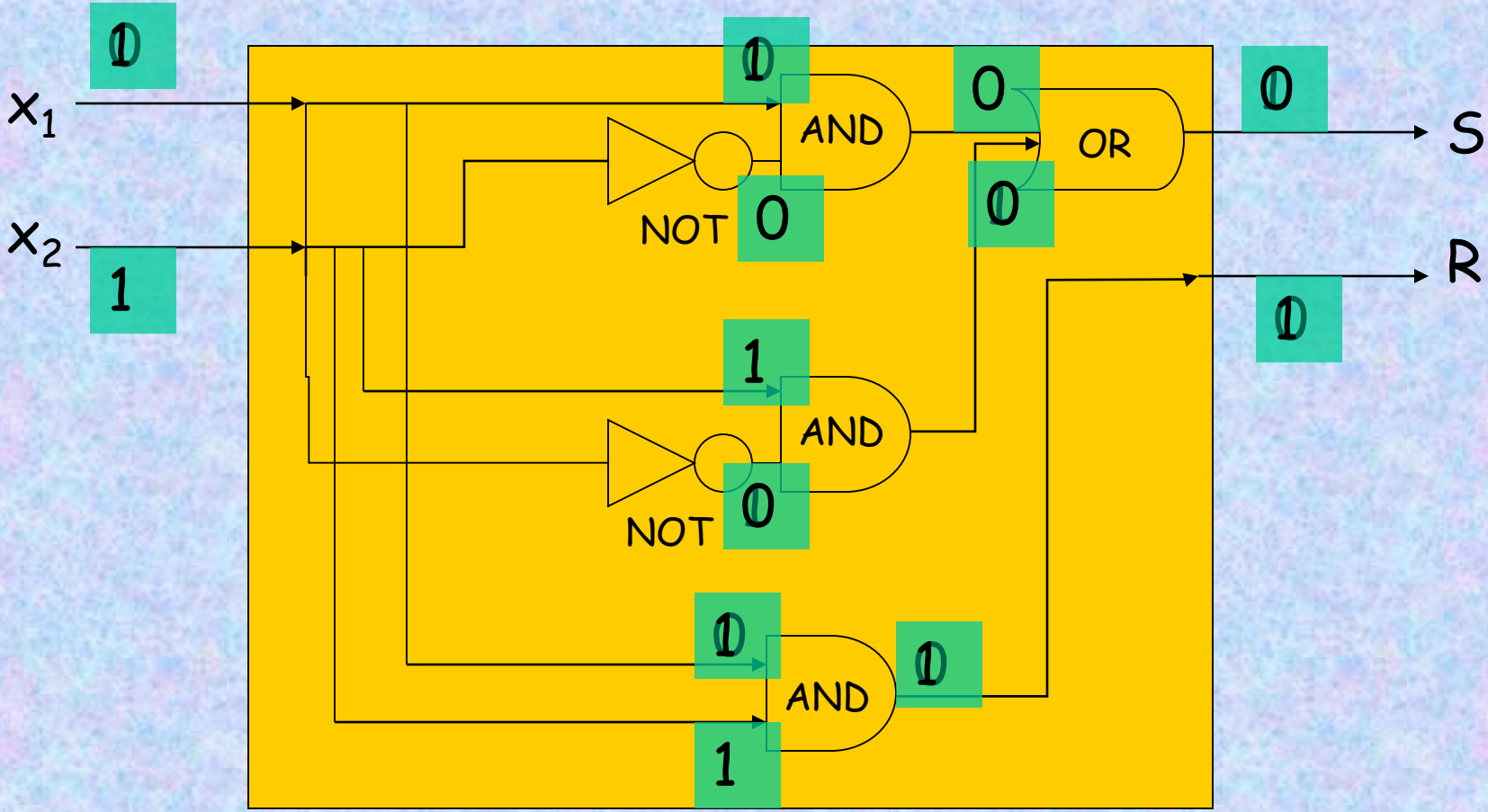
OR	0	1
0	0	1
1	1	1

AND	0	1
0	0	0
1	0	1

NOT	
0	1
1	0

Circuiti Logici

(Semiaddizionatore)



$S = (x_1 \text{ AND NOT}(x_2)) \text{ OR } (x_2 \text{ AND NOT}(x_1))$

$R = x_1 \text{ AND } x_2$

Codifica dei Caratteri

I caratteri di un testo sono codificati come sequenze di bit

I codici storici sono

ASCII 7 bit
(American Standard Code for Inform. Interchange)

EBCDIC 8 bit

BCD 4 bit

ISO (ISO-Latin 1)
Oggi: UNICODE

Caratteri

- Un carattere è un simbolo utilizzato per rappresentare un fonema o un numero
  - N.B. : il carattere 8 è concettualmente diverso dal valore 8 (che si può rappresentare, ad esempio, in numeri romani come VIII)
- Nei computer alcuni elementi pseudo-grafici sono rappresentati da caratteri
  - CR: Carriage Return (torna all'inizio della riga)
  - LF: Line Feed (avanza di una riga)
  - BEL: Bell (campanello)

Caratteri

- I caratteri alfabetici e quelli speciali (CR, LF, BEL, etc.) si rappresentano associando ad ognuno di essi un numero binario di 8 bit (codice ASCII) o 32 bit (Unicode)

- Alcuni esempi:

A ↔ 65	**a ↔ 97**
C ↔ 67	**c ↔ 99**
Z ↔ 90	**z ↔ 122**

Tipi di dati

- La stessa stringa di bit può rappresentare
  - Numeri interi positivi
  - Numeri interi con segno
  - Numeri razionali (reali)
  - Valori logici
  - Caratteri
- Per poterla interpretare è necessario conoscerne il tipo