

Corso di
Architettura degli Elaboratori – B
a.a. 2021/2022

Codifica dell'informazione: Numeri Binari

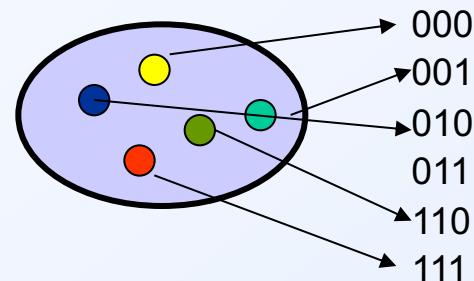
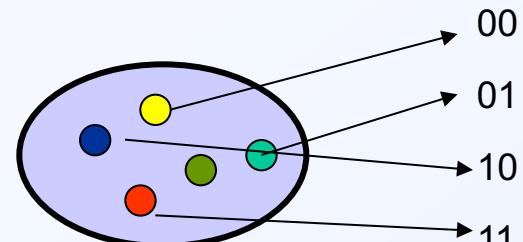
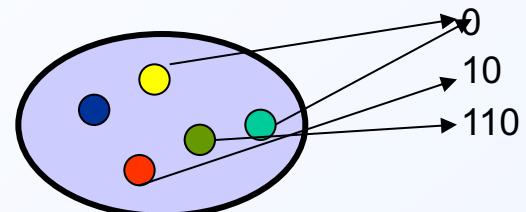
“Esistono 10 tipi di persone: quelle che utilizzano la codifica binaria e quelle che non la utilizzano”

Codifica dell'informazione

- **Codifica:** rappresentazione degli *elementi di un insieme* (anche infinito) mediante un *numero limitato di simboli* (cifre, numeri, segni grafici,...) seguendo una opportuna regola;

- **Proprietà** di una codifica:

- Non-ridondanza
- Lunghezza-costante
- Completezza e unicità



Codifica dell'informazione

- L'entità minima di informazione all'interno di un elaboratore prende il nome di **bit** (**binary digit** - cifra binaria). Mediante un bit possiamo distinguere due informazioni.
- Tutte le informazioni, anche le più complesse, sono rappresentate mediante sequenze di due soli simboli (**0** e **1**), ossia in forma **binaria** (o **digitale**).
- Si ha perciò bisogno di associare biunivocamente ad ogni informazione “elementare”: caratteri, numeri, immagini, ... una sequenza binaria che la rappresenti

Byte

- Con 1 bit si possono distinguere due diverse informazioni;
- Per distinguere più informazioni bisogna usare sequenze di bit.
- Le diverse configurazioni di n bit permettono di individuare 2^n informazioni diverse.
- **una sequenza di 8 bit viene chiamata “*byte*”**

Potenze di 2

2^0	1
2^1	2
2^2	4
2^3	8
2^4	16
2^5	32
2^6	64
2^7	128
2^8	256
2^9	512
2^{10}	1,024
2^{11}	2,048
2^{12}	4,096
2^{13}	8,192
2^{14}	16,384
2^{15}	32,768

2^{16}	65,536
2^{17}	131,072
2^{18}	262,144
2^{19}	524,288
2^{20}	1,048,576
2^{21}	2,097,152
2^{22}	4,194,304
2^{23}	8,388,608
2^{24}	16,777,216
2^{25}	33,554,432
2^{26}	67,108,864
2^{27}	134,217,728
2^{28}	268,435,456
2^{29}	536,870,912
2^{30}	1,073,741,824
2^{31}	2,147,483,648

Misure per capacità memorie

Byte	$8 = 2^3$ bit
Kilobit (Kbit o Kb)	$2^{10}(1024)$ bit
Megabit (Mbit o Mb)	$2^{20}(1048576)$ bit
Gigabit (Gbit o Gb)	$2^{30}(1073741824)$ bit
Terabit (Tbit o Tb)	$2^{40}(1099511627776)$ bit
Kilobyte (Kbyte o KB)	2^{10} byte
Megabyte (Mbyte o MB)	2^{20} byte
Gigabyte (Gbyte o GB)	2^{30} byte
Terabyte (Tbyte o TB)	2^{40} byte

Misure per capacità memorie

Sul libro di testo

Termine decimale	Abbreviazione	Valore	Termine binario	Abbreviazione	Valore	% maggiore
Kilobyte	KB	10^3	Kibibyte	KiB	2^{10}	2%
Megabyte	MB	10^6	Mebibyte	MiB	2^{20}	5%
Gigabyte	GB	10^9	Gibibyte	GiB	2^{30}	7%
Terabyte	TB	10^{12}	Tebibyte	TiB	2^{40}	10%
Petabyte	PB	10^{15}	Pebibyte	PiB	2^{50}	13%
Exabyte	EB	10^{18}	Exbibyte	EiB	2^{60}	15%
Zettabyte	ZB	10^{21}	Zebibyte	ZiB	2^{70}	18%
Yottabyte	YB	10^{24}	Yobibyte	YiB	2^{80}	21%

Misure per capacità canali

Byte	$8 = 2^3\text{bit}$
Kilobit/s (Kbit/s o Kb/s)	1000 bit/s
Megabit/s (Mbit/s o Mb/s)	1000000 bit/s
Gigabit/s (Gbit/s o Gb/s)	1000000000 bit/s

Misure per frequenze clock

MHz (Mega Hertz) 1000000 cicli/s

GHz (Giga Hertz) 1000000000 cicli/s

Codifica dei numeri

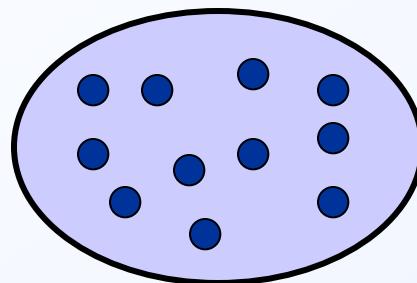
- Esigenza di rappresentare l'insieme infinito dei numeri mediante un numero limitato di segni grafici (cifre)
- Avendo un insieme finito di simboli, per denotare gli infiniti numeri li combino secondo una codifica



- **Sistemi di numerazione:**
 - **Cifre:** un insieme finito di simboli distinti
 - **Codifica:** insieme di regole che permette di associare ad una sequenza di cifre uno ed un solo numero
 - **Algoritmi:** per l'esecuzione delle operazioni fondamentali

Numero vs Numerale

- Numero è entità astratta (fuori da spazio e tempo): proprietà degli insiemi che hanno stessa quantità di elementi. **NON PUO' STARE IN MEMORIA!!!**
- Numerale: configurazione di simboli che denota (= identifica) un numero – non è astratta
- *Diversi* tipi di numerali rispondenti a codifiche diverse identificano lo stesso numero:
 - 11 in decimale, B in esadecimale, 13 in ottale, 1011 in binario denotano lo stesso numero



RIPASSINO DI MATEMATICA

Proprietà delle potenze:

- $x^0 = 1$
- $x^{-i} = 1 / x^i$
- $x^i * x^j = x^{i+j}$
- $(x^i)^j = x^{i*j}$
- $(x*y)^i = x^i * y^i$
- $x^i / x^j = x^{i-j}$
- $a * x^i + b * x^j = (a * x^{i-1} + b * x^{j-1}) * x$

Sistemi di numerazione

- ✗ Cifre ordinate in modo che ognuna sia maggiore della precedente di un'unità, a partire da zero:
& (zero), **%** (uno), **#** (due), ***** (tre)
- ✗ Il numero di cifre fornisce la base (quattro nell'esempio)
- ✗ Le cifre **hanno un valore posizionale**:
 - ✓ La sequenza di cifre **% * & % #** in notazione decimale sarebbe: $1 \times 4^4 + 3 \times 4^3 + 0 \times 4^2 + 1 \times 4^1 + 2 \times 4^0 = 454$
- ✗ se la somma di due cifre è maggiore del numero di cifre si crea un riporto

$$\begin{array}{r} \# + \\ * = \\ \hline \% \% \end{array} \qquad \begin{array}{r} 2 + \\ \hline 3 = \\ 5 \end{array}$$

Notazione posizionale

Forma generale di un numero decimale

	Centinaia	Decine	Unita'	decimi	centesimi				
	↓	↓	↓	↓	↓				
d_n	\dots	d_2	d_1	d_0	.	d_{-1}	d_{-2}	\dots	d_{-k}

$$\text{numero} = \sum_{i=-k}^n d_i \times 10^i$$

Forma generale di un numero in base b

c_n	\dots	c_2	c_1	c_0	.	c_{-1}	c_{-2}	\dots	c_{-k}
-------	---------	-------	-------	-------	---	----------	----------	---------	----------

$$\text{numero} = \sum_{i=-k}^n c_i \times b^i$$

Esempi

Base 2

cifre usate 0 e 1 (bit)

$$\begin{aligned}11010.1 &= 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} \\&= 16 + 8 + 2 + 0.5 = 26.5\end{aligned}$$

Base 8

cifre usate 0, 1, 2, 3, 4, 5, 6, 7

$$\begin{aligned}12120.5 &= 1 \times 8^4 + 2 \times 8^3 + 1 \times 8^2 + 2 \times 8^1 + 0 \times 8^0 + 5 \times 8^{-1} \\&= 4096 + 1024 + 64 + 16 + 0.625 = 5200.625\end{aligned}$$

Base 16

cifre usate 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

$$\begin{aligned}2E0A.3 &= 2 \times 16^3 + 14 \times 16^2 + 0 \times 16^1 + 10 \times 16^0 + 3 \times 16^{-1} \\&= 8192 + 3584 + 10 + 0.1875 = 11786.1875\end{aligned}$$

Il numero “mille” in

Binario:

1 1 1 1 1 0 1 0 0 0

$$1 \times 2^9 + 1 \times 2^8 + 1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$$

$$512 + 256 + 128 + 64 + 32 + 0 + 8 + 0 + 0 + 0 + 0$$

Ottale: 1 7 5 0

$$1 \times 8^3 + 7 \times 8^2 + 5 \times 8^1 + 0 \times 8^0$$

$$512 + 448 + 40 + 0$$

Decimale: 1 0 0 0

$$1 \times 10^3 + 0 \times 10^2 + 0 \times 10^1 + 0 \times 10^0$$

$$1000 + 0 + 0 + 0$$

Esadecimale: 3 E 8

$$3 \times 16^2 + 14 \times 16^1 + 8 \times 16^0$$

$$768 + 224 + 8$$

Conversione tra basi: dec → bin

	$\begin{array}{r} 171.25 \\ - 128 \\ \hline \end{array}$	= 2^7 Massima potenza di 2 ≤ 171.25
resto	$\begin{array}{r} 43.25 \\ - 32 \\ \hline \end{array}$	= 2^5 Massima potenza di 2 ≤ 43.25
resto	$\begin{array}{r} 11.25 \\ - 8 \\ \hline \end{array}$	= 2^3 Massima potenza di 2 ≤ 11.25
resto	$\begin{array}{r} 3.25 \\ - 2 \\ \hline \end{array}$	= 2^1 Massima potenza di 2 ≤ 3.25
resto	$\begin{array}{r} 1.25 \\ - 1 \\ \hline \end{array}$	= 2^0 Massima potenza di 2 ≤ 1.25
	$\begin{array}{r} 0.25 \\ - 0.25 \\ \hline 0.00 \end{array}$	= 2^{-2} Massima potenza di 2 ≤ 0.25
$171.25 \mid_{10} =$	$1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad . \quad 0 \quad 1 \mid_2$	
	7 6 5 4 3 2 1 0 -1 -2	

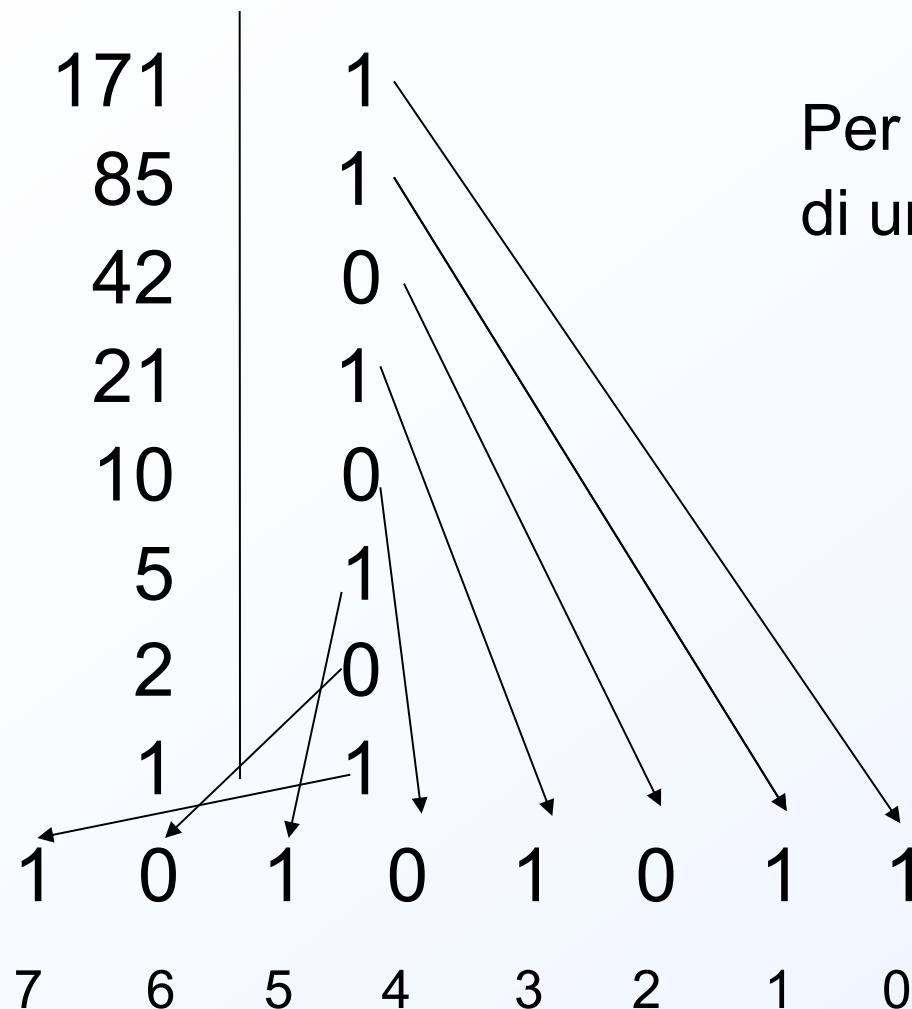
Conversione tra basi: dec → base r

Parte intera

1. Inizio;
2. Dividere il numero decimale per la base di arrivo r ;
3. Il resto della divisione è una cifra nella nuova base
a partire dalla cifra meno significativa ;
4. Il quoziente della divisione intera è il nuovo dividendo;
5. Se quoziente $\neq 0$ torna a 2);
6. Fine.

Conversione tra basi: dec → bin

Divisioni successive per 2



Per la parte intera
di un numero

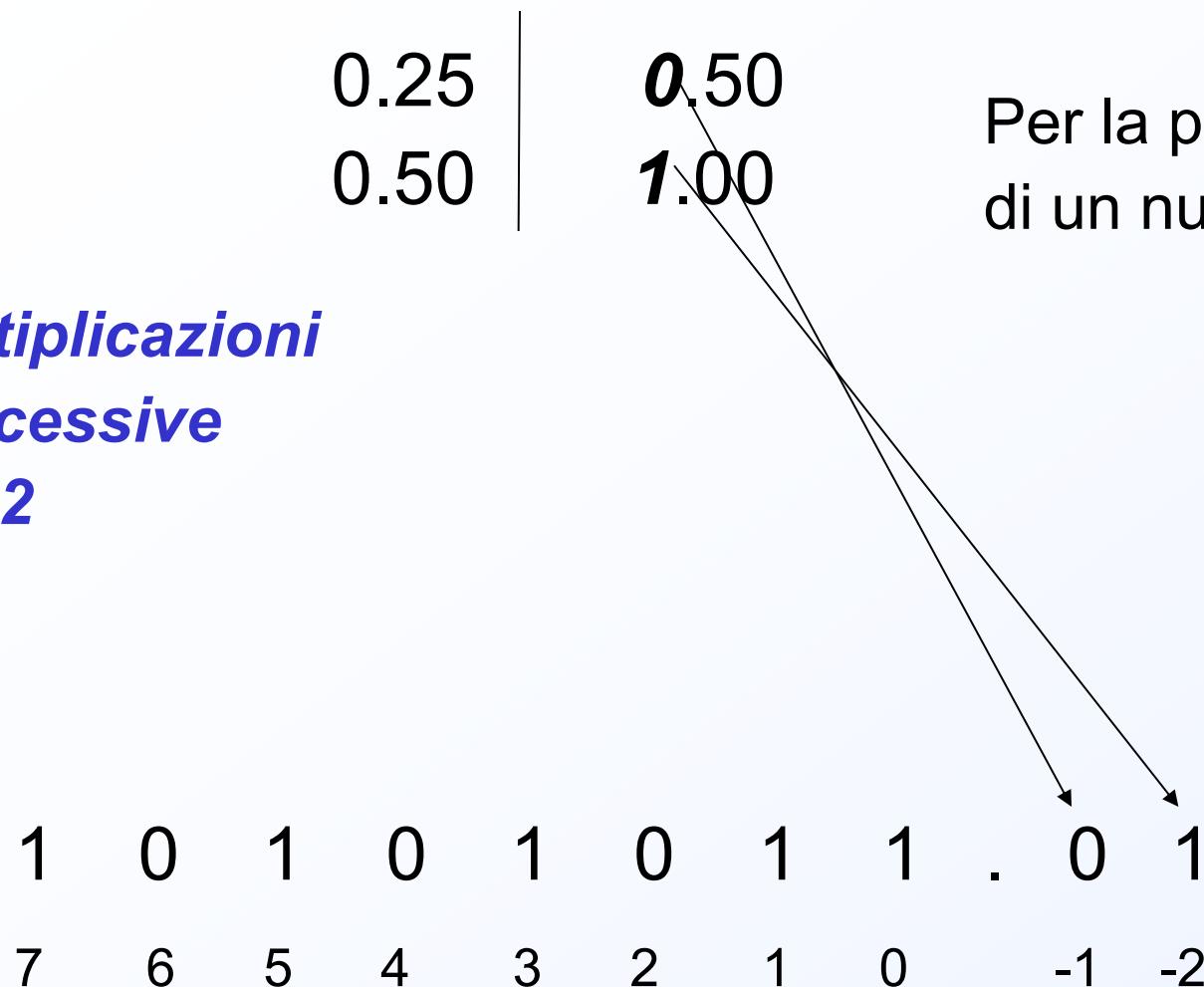
Conversione tra basi: dec -> base r

Parte frazionaria

1. Inizio;
2. Moltiplicare la parte frazionaria del numero decimale per la base di arrivo;
3. Separare parte intera e parte frazionaria;
4. La parte intera dà una cifra nella nuova base a partire dalla cifra più significativa ;
5. Se non ottengo 0 o non raggiungo la precisione richiesta torna a 2);
6. Fine.

Conversione tra basi: dec → bin

*Moltiplicazioni
successive
per 2*



Per la parte decimale
di un numero

Esempio 35,59375

35	1	Per la	0.59375	1.1875
17	1	parte intera	0.1875	0.375
8	0		0.375	0.75
4	0		0.75	1.5
2	0		0.5	1.0
1	1			

1 0 0 0 1 1 . 1 0 0 1 1

5 4 3 2 1 0 . -1 -2 -3 -4 -5

Un altro esempio 35,9

35	1	Per la	0.9	1.8
17	1	parte intera	0.8	1.6
8	0		0.6	1.2
4	0		0.2	0.4
2	0		0.4	0.8
1	1		0.8	1.6

1 0 0 0 1 1 . 1 1 1 0 0 1 ...

5 4 3 2 1 0 -1 -2 -3 -4 -5 -6 ...

Conversione tra basi: bin -> dec

$$I_r = d_{n-1}r^{n-1} + d_{n-2}r^{n-2} + \dots + d_2r^2 + d_1r^1 + d_0r^0$$

$$= ((\dots((d_{n-1}r + d_{n-2})r + d_{n-3})r + \dots + d_1)r + d_0$$

per gli interi: 11010

1	1
1	$1 + 2 \times 1 = 3$
0	$0 + 2 \times 3 = 6$
1	$1 + 2 \times 6 = 13$
0	$0 + 2 \times 13 = 26$

Conversione tra basi: bin -> dec

1. Inizio;
2. Partire dalla cifra più significativa;
3. Se raggiunta la cifra meno significativa andare a 7);
4. Moltiplicare per la base;
5. Sommare la cifra successiva;
6. Andare a 3);
7. Fine.

Conversione tra basi: bin \leftrightarrow ott

Binario \rightarrow Ottale

$$\begin{array}{r} 0 \quad 1 \quad 1 \\ \underbrace{\quad\quad}_{3} \end{array} \quad \begin{array}{r} 0 \quad 1 \quad 0 \\ \underbrace{\quad\quad}_{2} \end{array} \quad . \quad \begin{array}{r} 1 \quad 0 \quad 0(26.5) \\ \underbrace{\quad\quad}_{4} \end{array}$$

(26.5)

Ottale \rightarrow Binario

$$\begin{array}{r} 2 \\ \underbrace{0 \quad 1 \quad 0}_{1} \end{array} \quad \begin{array}{r} 5 \\ \underbrace{1 \quad 0 \quad 1}_{3} \end{array} \quad \begin{array}{r} 0 \quad 1 \quad 1 \\ \underbrace{\quad\quad}_{2} \end{array} \quad . \quad \begin{array}{r} 0 \quad 1 \quad 0(171.25) \\ \underbrace{\quad\quad}_{4} \end{array}$$

(171.25)

Conversione tra basi: bin \leftrightarrow esa

Binario \rightarrow Esadecimale

$$\begin{array}{ccccccccc} 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & . & 1 & 0 & 0 & 0 \end{array} \quad (26.5)$$
$$\underbrace{\hspace{2cm}}_{1} \quad \underbrace{\hspace{2cm}}_{A} \quad \quad \quad \underbrace{\hspace{2cm}}_{8} \quad (26.5)$$

Esadecimale \rightarrow Binario

$$\begin{array}{ccccccccc} A & & B & & & & 4 & & \\ \underbrace{\hspace{2cm}}_{1 0 1 0} & \underbrace{\hspace{2cm}}_{1 0 1 1} & . & & & & \underbrace{\hspace{2cm}}_{1 0 0} & & \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \end{array} \quad (171.25) \quad (171.25)$$

Conversione tra basi: ott \leftrightarrow esa

Conviene passare attraverso la *rappresentazione binaria*, es. 32.4 ->

- Conversione da ottale a binario -> 011 010 . 100
- Espansione in quaterne 0001 1010 . 1000
- Conversione da binario in esadecimale -> 1A.8
- Viceversa, conversione in binario -> 0001 1010 . 1000
- Raggruppamento in terne -> 011 010 . 100
- Conversione da binario in ottale -> 32.4

Addizione binaria

- A queste rappresentazioni si possono applicare le operazioni aritmetiche:

$$0+0=0$$

$$1+0=1$$

$$0+1=1$$

$1+1=0$ con riporto di 1 ovvero 10

- $1+1$ in decimale è uguale a 2 ma siamo nella notazione binaria che ha solo due cifre, 0 e 1

Addizione binaria

x_i	y_i	c_{i-1}	s_i	c_i
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

Addizione binaria

- x A queste rappresentazioni si possono applicare le operazioni aritmetiche:

riporti

1

1 0 +

1 0 =

1 0 0

Addizione binaria

- x A queste rappresentazioni si possono applicare le operazioni aritmetiche:

riporti

1

1 1 +

1 0 =

1 0 1

Addizione binaria

- x A queste rappresentazioni si possono applicare le operazioni aritmetiche:

riporti

1 1

1 1 +

1 1 =

1 1 0

Addizione binaria

- x A queste rappresentazioni si possono applicare le operazioni aritmetiche:

riporti

$$\begin{array}{r} 1 \ 1 \ 1 \\ 1 \ 1 \ 1 \quad + \\ 1 \ 1 \quad = \end{array}$$

1 0 1 0

Addizione binaria

ESEMPIO:

$$\begin{array}{r} 1 \ 0 \ 1 \ 1 \ + \\ 0 \ 1 \ 1 \ 1 \\ \hline 1 \ 0 \ 0 \ 1 \ 0 \end{array}$$

Moltiplicazione e Divisione per 2^m

- Moltiplicare un numero binario *positivo* per $X=2^m$ corrisponde a spostare verso *sinistra di m posizioni* le cifre corrispondenti alla sua rappresentazione aggiungendo zeri nelle m posizioni meno significative.

➤ Esempio

$$7 \times 8 = 56 \quad \rightarrow$$

$$00000111 \quad (7|_{10})$$

$$\begin{array}{r} 00000111000 \\ \swarrow \searrow \swarrow \searrow \swarrow \searrow \\ 00000111000 \quad (56|_{10}) \end{array}$$

8 = 2^3 : shift sinistra di 3 bit

- Dividere un numero binario *positivo* per $X=2^m$ (o moltiplicare per 2^{-m}) corrisponde a spostare verso *destra di m posizioni* le cifre corrispondenti alla sua rappresentazione aggiungendo zeri nelle m posizioni più significative.

➤ Esempio:

$$7 / 4 = 1.75 \quad \rightarrow$$

$$00000111 \quad (7|_{10})$$

4 = 2^2 : shift destra di 2 bit

$$00000001.11 \quad (1.75|_{10})$$

- Facciamo lo stesso in base 10!!!!!!

Memoria finita, numeri infiniti

- Non tutti i numeri si possono rappresentare in un computer.
- In più non è conveniente gestire numeri anche se finiti di dimensioni arbitrarie o diverse fra loro.
- Velocità deriva da hardware dedicato a calcoli: dimensione fissa degli operandi (lo vedremo).
- Ad esempio, in Java (ma anche in C abbiamo cose simili):
byte 8 bit, short 16 bit, int 32 bit, long 64 bit, float 32 bit, double 64 bit

Numeri a precisione finita

- Per i calcolatori la quantità di memoria per memorizzare un numero è fissata. Si possono rappresentare numeri con una quantità fissa di cifre: **numeri a precisione finita**
- **conseguenza: non chiusura rispetto alle operazioni elementari**

Esempio: numeri interi con tre cifre decimali:

- **Si possono rappresentare i numeri compresi tra 000 e 999; il numero successivo (1000) richiede una quarta cifra.**
- Non si può rappresentare il risultato della somma $600 + 700$ perchè il numero di cifre destinato alla rappresentazione è insufficiente: si ha un **overflow**.

Proprietà non valgono più

- Associativa:

$$(a + b) - c = a + (b - c)$$

$$(100 + 999) - 980 = 100 + (999 - 980)$$

- Distributiva

$$(a*c - b*c) = (a - b)*c$$

$$(100 * 10 - 50 * 10) = (100 - 50) * 10$$

Numeri a precisione finita

- Osservazione: il numero 999 può essere scritto come 10^3-1 ($1000-1$)
 - ***Con N cifre decimali si possono rappresentare i numeri interi da 0 a 10^N-1***
 - ***I calcolatori usano basi o radici diverse da 10, di solito 2, 8 e 16.***
 - ***I sistemi di numerazione basati su queste radici si chiamano rispettivamente: binari, ottali, esadecimali.***
- Esempi:
- con N cifre binarie si possono rappresentare i numeri interi da 0 a 2^N-1
 - con N cifre ottali si possono rappresentare i numeri interi da 0 a 8^N-1
 - con N cifre esadecimali si possono rappresentare i numeri interi da 0 a 16^N-1

Numeri a precisione finita

- ***Numeri naturali***
- Consideriamo la base due: con **3** cifre binarie si possono rappresentare i numeri compresi tra **0** (limite inferiore) e **2^3-1** (limite superiore).

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

Notazione posizionale

Il numero massimo rappresentabile con n cifre in base r risulta:

$$\begin{aligned}N_{max} &= (r - 1) r^{n-1} + (r - 1) r^{n-2} + \dots \\&\quad + (r - 1) r + (r - 1) r^0 \\&= \sum_{i=0}^{n-1} (r - 1) \cdot r^i \\&= r^n - 1\end{aligned}$$

Numeri a precisione finita

- ***Numeri relativi***
- Se si passa ai numeri relativi la scelta del limite inferiore della rappresentazione diventa meno ovvia
- Criterio ragionevole: “centrare” (nel migliore modo possibile) l’intervallo dei numeri rappresentabili intorno al valore zero, in modo da poter rappresentare tutti i numeri di valore assoluto minore o uguale ad un certo valore massimo

Sintassi vs semantica

- ✗ L'operazione tra due numeri deve essere realizzata applicando un algoritmo sulla rappresentazione dei numeri che dia come risultato la rappresentazione del numero risultante dall'operazione tra due numeri.
 - ✓ $n+m$: $f(n)+f(m)=f(n+m)$
- ✗ Cercheremo una codifica dei numeri relativi (con segno) che ci consenta di raggiungere questo obiettivo. Perché?
 - ✓ Perché vogliamo sfruttare l'operazione di somma binaria dei moduli (valori assoluti) che è semplice e naturale (può essere realizzata in maniera ottimale in hardware) e può essere l'unica operazione da usare sia per somme sia per sottrazioni

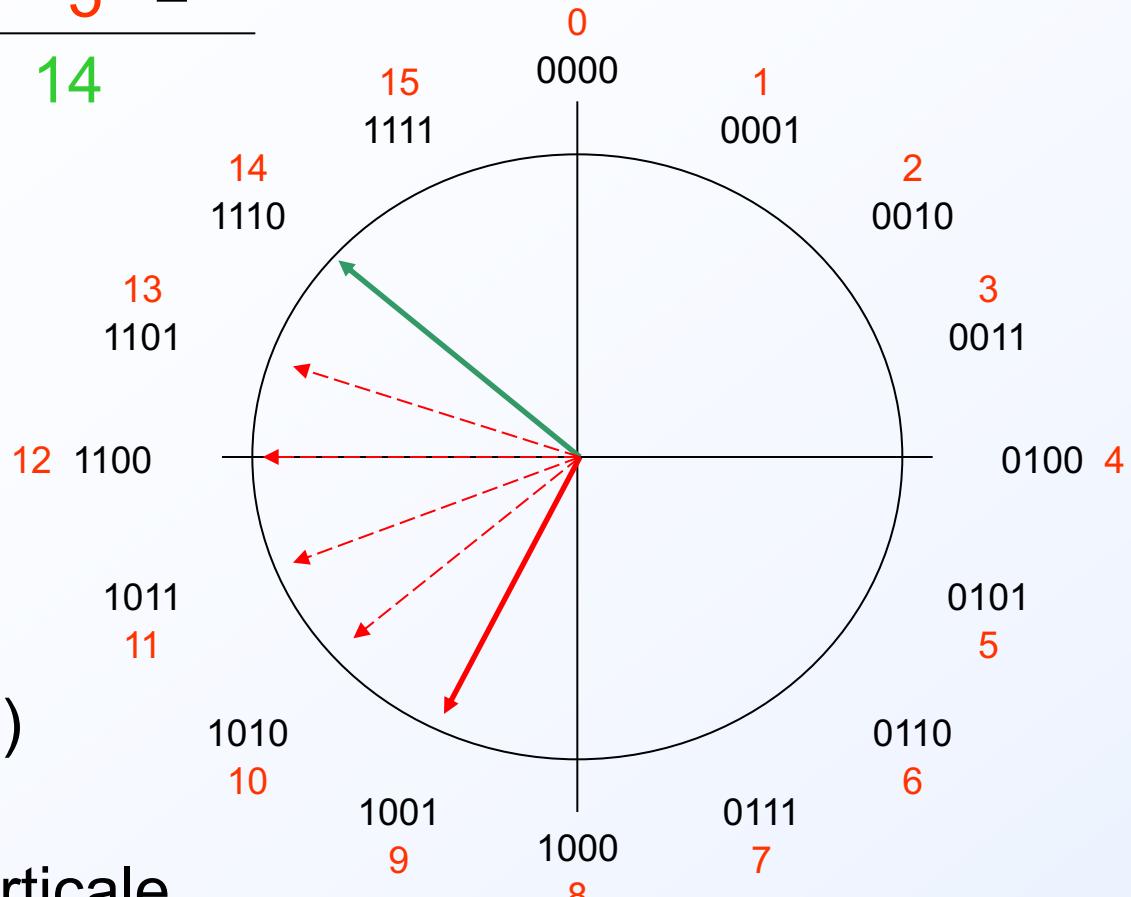
Numeri a precisione finita

- ***Numeri naturali***
- Consideriamo la base due: con tre cifre binarie si possono rappresentare i numeri compresi tra 0 (limite inferiore) e 2^3-1 (limite superiore).

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

Somma tra moduli (solo 4 cifre)!

$$\begin{array}{r} 1 \ 0 \ 0 \ 1 + \\ 0 \ 1 \ 0 \ 1 = \\ \hline 1 \ 1 \ 1 \ 0 \end{array} \quad \begin{array}{r} 9 + \\ 5 = \\ \hline 14 \end{array}$$



$$n+m: f(n)+f(m)=f(n+m)$$

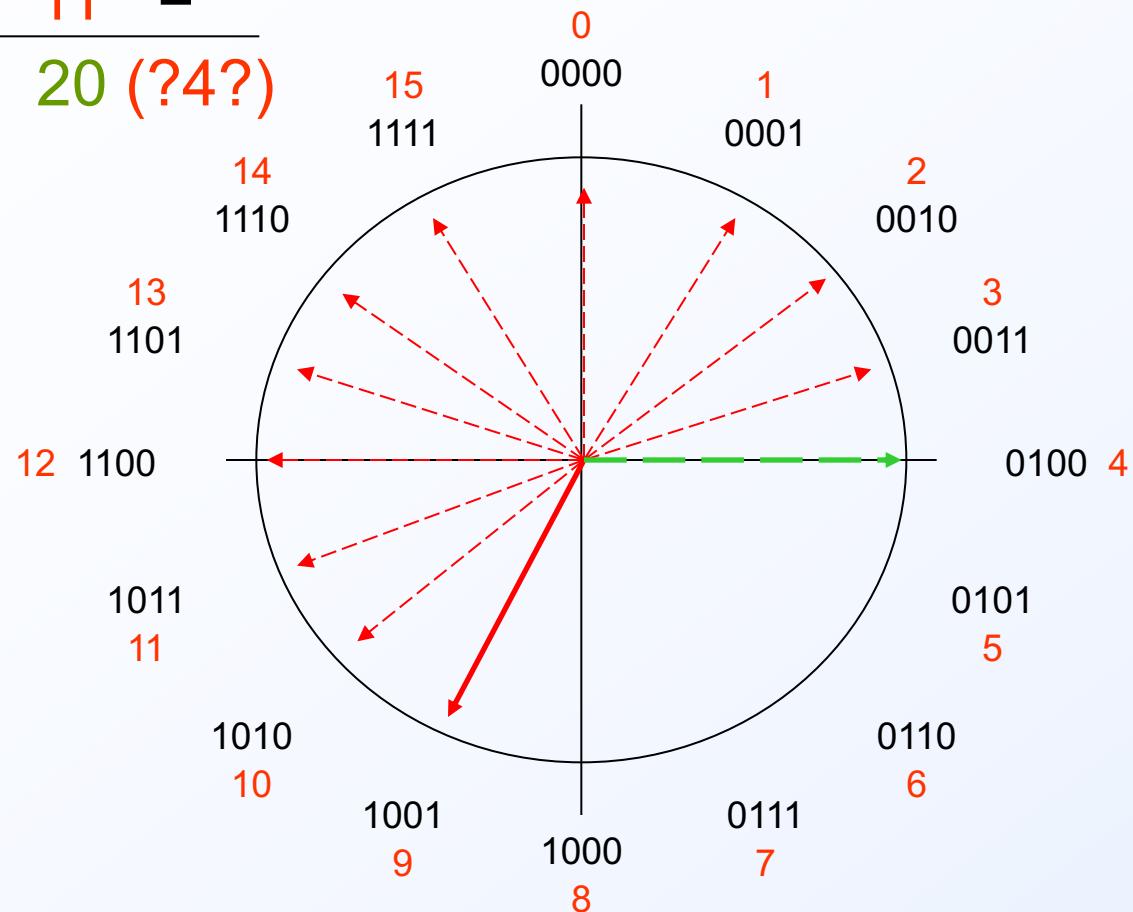
angolo orario sulla verticale

somma tra numeri come somma di angoli (in senso orario)

Somma tra moduli (solo 4 cifre)!

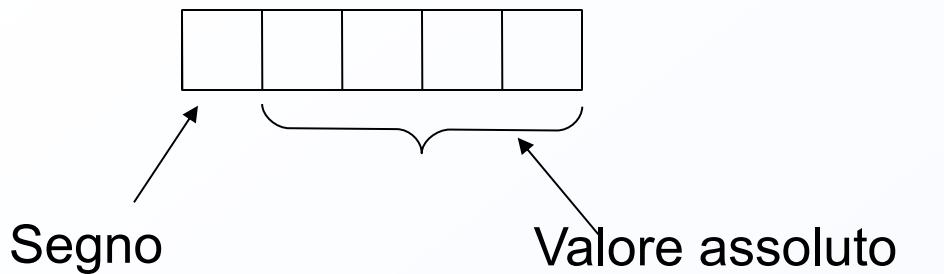
$$\begin{array}{r} 1\ 0\ 0\ 1 \\ + \\ 1\ 0\ 1\ 1 \\ \hline 1\ 0\ 1\ 0\ 0 \end{array} \quad \begin{array}{r} 9 \\ + \\ 11 \\ \hline 20\ (\text{?}4?) \end{array}$$

- overflow a causa del riporto
- il risultato ha superato il massimo valore rappresentabile su 4 bit
- si “ricomincia il giro” e si genera un riporto



Rappres. modulo e segno

1. *Modulo e segno*



Decimale	binario
11	= 0 1011
-11	= 1 1011

Rappresentazione su N cifre binarie:

- una cifra binaria (per convenzione quella più a sinistra) per codificare il segno
- le rimanenti $N-1$ cifre rappresentano la codifica in forma binaria fissa del valore assoluto del numero (che per definizione è un numero naturale)

Rappres. modulo e segno

- ***Intervallo dei numeri rappresentabile su N bit***

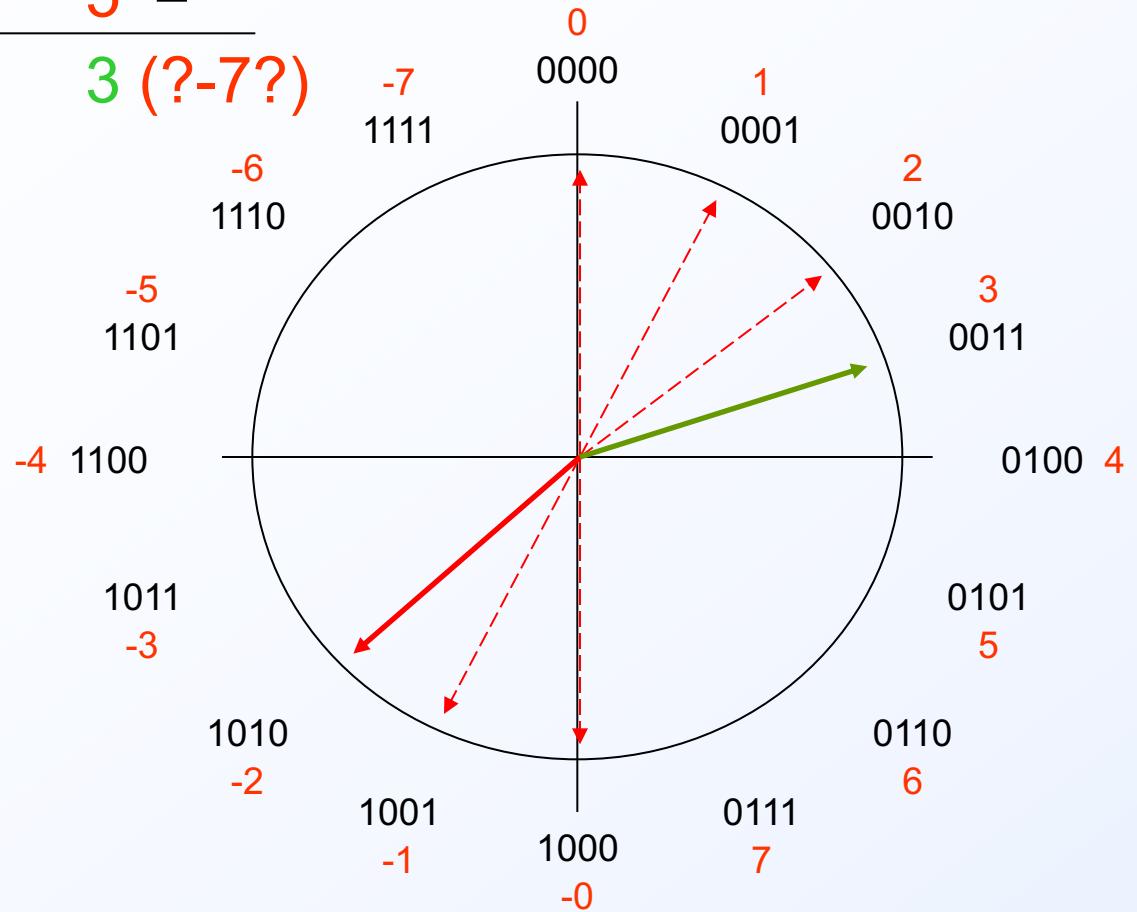
$$[- (2^{N-1} - 1), 2^{N-1} - 1]$$

- ***Svantaggi:***

- Lo zero può essere rappresentato in due modi diversi (00..00, ossia +0, e 10..00, ossia –0)
- L'operazione di somma tra due numeri non può essere realizzata applicando l'algoritmo di somma sulla rappresentazione dei moduli. Occorre tener conto della concordanza o discordanza dei segni: nel caso di numeri con segni discordi occorre identificare il numero di valor assoluto maggiore ed applicare l'algoritmo di sottrazione tra il modulo di questo ed il modulo dell'altro addendo; il segno del risultato sarà uguale al segno dell'addendo di valor assoluto maggiore
- Provate, con questa rappresentazione, a fare $X+(-X)$ con la somma sui moduli come fossero numeri naturali!!

Somma in modulo e segno

$$\begin{array}{r} 1 \ 0 \ 1 \ 0 + \\ 0 \ 1 \ 0 \ 1 = \\ \hline 1 \ 1 \ 1 \ 1 \end{array} \quad \begin{array}{r} -2 + \\ 5 = \\ \hline 3 \ (?-7?) \end{array}$$



Il complemento

- ✗ Per un attimo **dimentichiamo il problema del segno** e definiamo una nuova operazione su un numero X a precisione finita su K cifre binarie (si può generalizzare ad una base qualunque).
- ✗ Complemento a 1:
 - negare tutte le cifre di X (ossia sostituire 0 con 1 e viceversa 1 con 0)
- Complemento a 2:
 - **Metodo 1:** Calcolare il complemento a 1 di X e sommargli 1. Tralasciare l'eventuale cifra di riporto a sinistra di quella più significativa (overflow)
 - **Metodo 2:** Partire dalla cifra meno significativa di X e ricopiare tutte le cifre fino al primo 1 incluso. Invertire le cifre a sinistra di questo.

Esempi di complemento a 1

$$\begin{array}{r} X=12 \\ 3 \\ 15 \end{array} \quad \begin{array}{r} 1 \ 1 \ 0 \ 0 \\ 0 \ 0 \ 1 \ 1 \\ \hline 1 \ 1 \ 1 \ 1 \end{array} \quad \begin{array}{r} \downarrow \\ \downarrow \\ \downarrow \end{array} \quad \begin{array}{r} X=4 \\ 11 \\ 15 \end{array} \quad \begin{array}{r} 0 \ 1 \ 0 \ 0 \\ 1 \ 0 \ 1 \ 1 \\ \hline 1 \ 1 \ 1 \ 1 \end{array} \quad \begin{array}{r} \downarrow \\ \downarrow \\ \downarrow \end{array} \quad \begin{array}{r} X=15 \\ 0 \\ 15 \end{array} \quad \begin{array}{r} 1 \ 1 \ 1 \ 1 \\ 0 \ 0 \ 0 \ 0 \\ \hline 1 \ 1 \ 1 \ 1 \end{array} \quad \begin{array}{r} \downarrow \\ \downarrow \\ \downarrow \end{array}$$

- xDato X il suo complemento a 1 è il numero che, se sommato a X, restituisce sempre 2^K-1
- xDOppure, equivalentemente, dato X il suo complemento a 1 è il numero 2^K-1-X

Esempi di complemento a 2

$$\begin{array}{r} X=12 \quad 1 \ 1 \ 0 \ 0 \\ 4 \quad 0 \ 1 \ 0 \ 0 \\ 16 \quad 1 \ 0 \ 0 \ 0 \\ \hline 1 \ 0 \ 0 \ 0 \ 0 \end{array} \quad \text{(oppure 0)}$$
$$\begin{array}{r} X=4 \quad 0 \ 1 \ 0 \ 0 \\ 12 \quad 1 \ 1 \ 0 \ 0 \\ 16 \quad 1 \ 0 \ 0 \ 0 \ 0 \\ \hline 1 \ 0 \ 0 \ 0 \ 0 \end{array} \quad \text{(oppure 0)}$$
$$\begin{array}{r} X=15 \quad 1 \ 1 \ 1 \ 1 \\ 1 \quad 0 \ 0 \ 0 \ 1 \\ 16 \quad 1 \ 0 \ 0 \ 0 \ 0 \\ \hline 1 \ 0 \ 0 \ 0 \ 0 \end{array} \quad \text{(oppure 0)}$$

- Dato X il suo complemento a 2 è il numero che, se sommato a X , restituisce sempre 2^K (oppure zero dato che le cifre sono solo K)
- Oppure, equivalentemente, dato X il suo complemento a 2 è il numero $2^K - X$ (oppure, $0 - X = -X$ dato che le cifre sono solo K).

Numeri relativi in complemento a 1

Rappresentazione in complemento a 1 su N cifre binarie di un numero con segno X:

- la cifra binaria più a sinistra rappresenta il segno di X (0 = +, 1 = -)
- nel caso di segno positivo, X è rappresentato in forma binaria su N -1 cifre come nel caso dei numeri naturali. Il numero X deve essere inferiore a $2^{(N-1)}$
- nel caso di segno negativo, X si rappresenta come il complemento a 1 di $-X$ (ricordatevi di negare **tutte le cifre** della rappresentazione binaria del valore assoluto di X, segno incluso). Il numero $-X$ deve essere inferiore a $2^{(N-1)}$

Decimale binario

12 = 01100

-12 = 10011

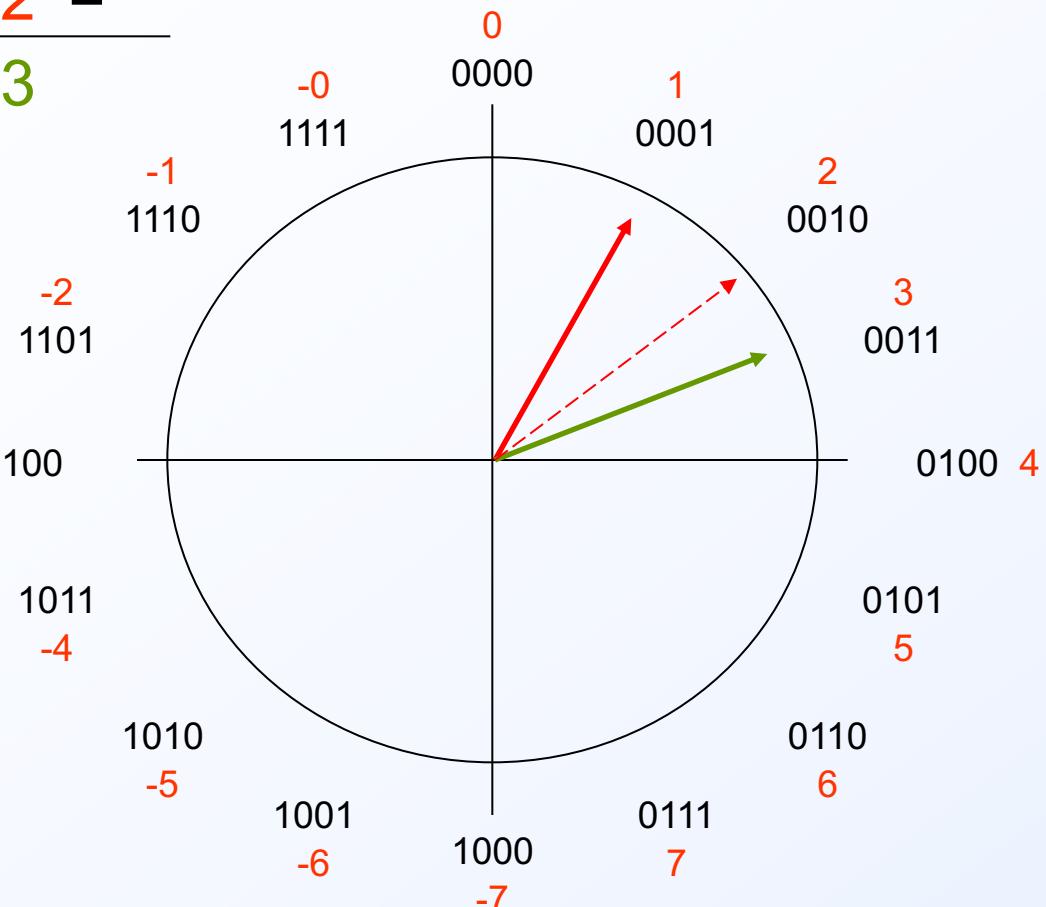
Numeri relativi in complemento a 1

- Caratteristiche molto simili alla rappresentazione modulo e segno, ossia:
 - **Semplicità della rappresentazione**
 - **Stesso intervallo dei numeri rappresentabili**
 $[-(2^{N-1}-1), 2^{N-1}-1]$
 - **Due rappresentazioni possibili per lo zero (in questo caso "000...0" per +0 e "111...1" per -0)**
- **Si riesce a sfruttare la somma binaria per eseguire la somma di numeri relativi in complemento a 1?**

Numeri relativi in complemento a 1

$$\begin{array}{r} 0\ 0\ 0\ 1 + \\ 0\ 0\ 1\ 0 = \\ \hline 0\ 0\ 1\ 1 \end{array} \quad \begin{array}{r} 1 + \\ 2 = \\ 3 \end{array}$$

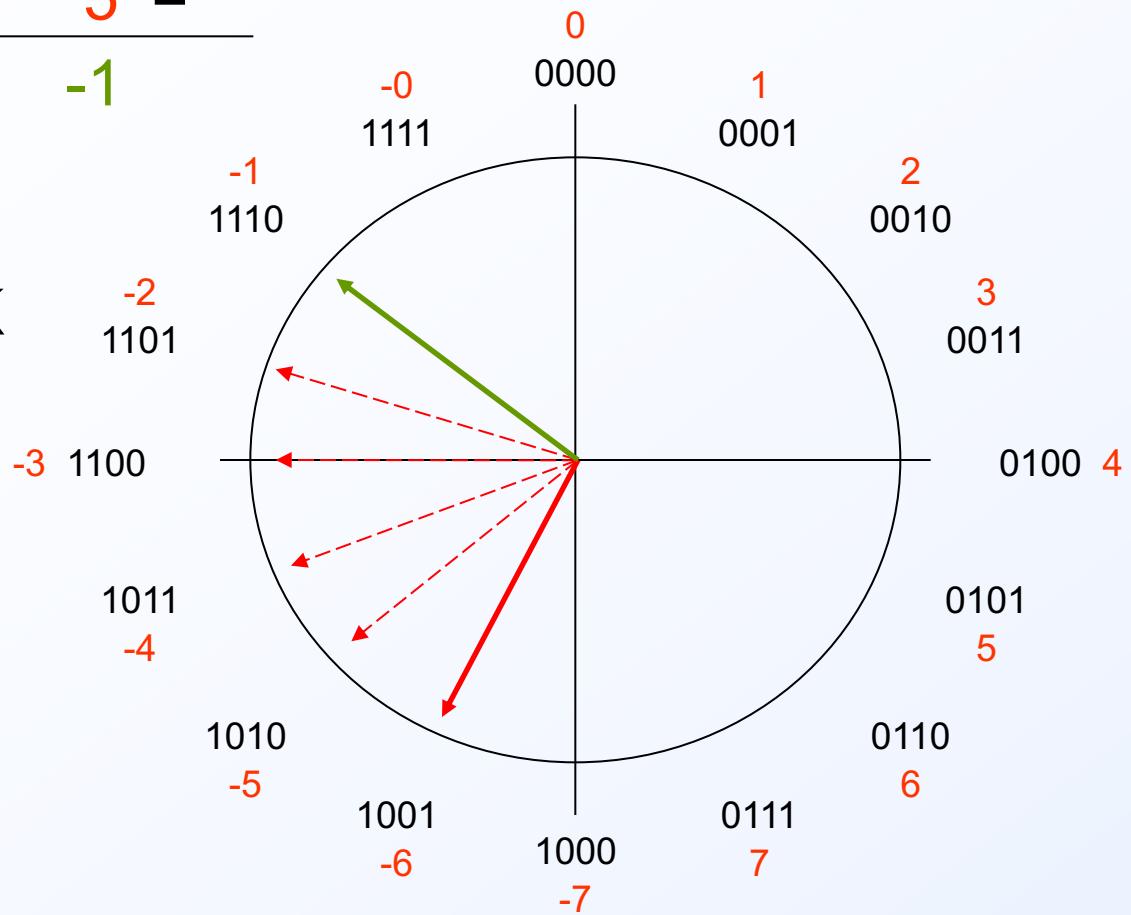
- Addendi positivi: OK sempre che non ci sia un overflow
- In tal caso il risultato non può essere rappresentato sul numero di cifre a disposizione



Numeri relativi in complemento a 1

$$\begin{array}{r} 1\ 0\ 0\ 1 + \\ 0\ 1\ 0\ 1 = \\ \hline 1\ 1\ 1\ 0 \end{array} \quad \begin{array}{r} -6 + \\ 5 = \\ \hline -1 \end{array}$$

Addendi discordi: OK
se il risultato ha
segno negativo

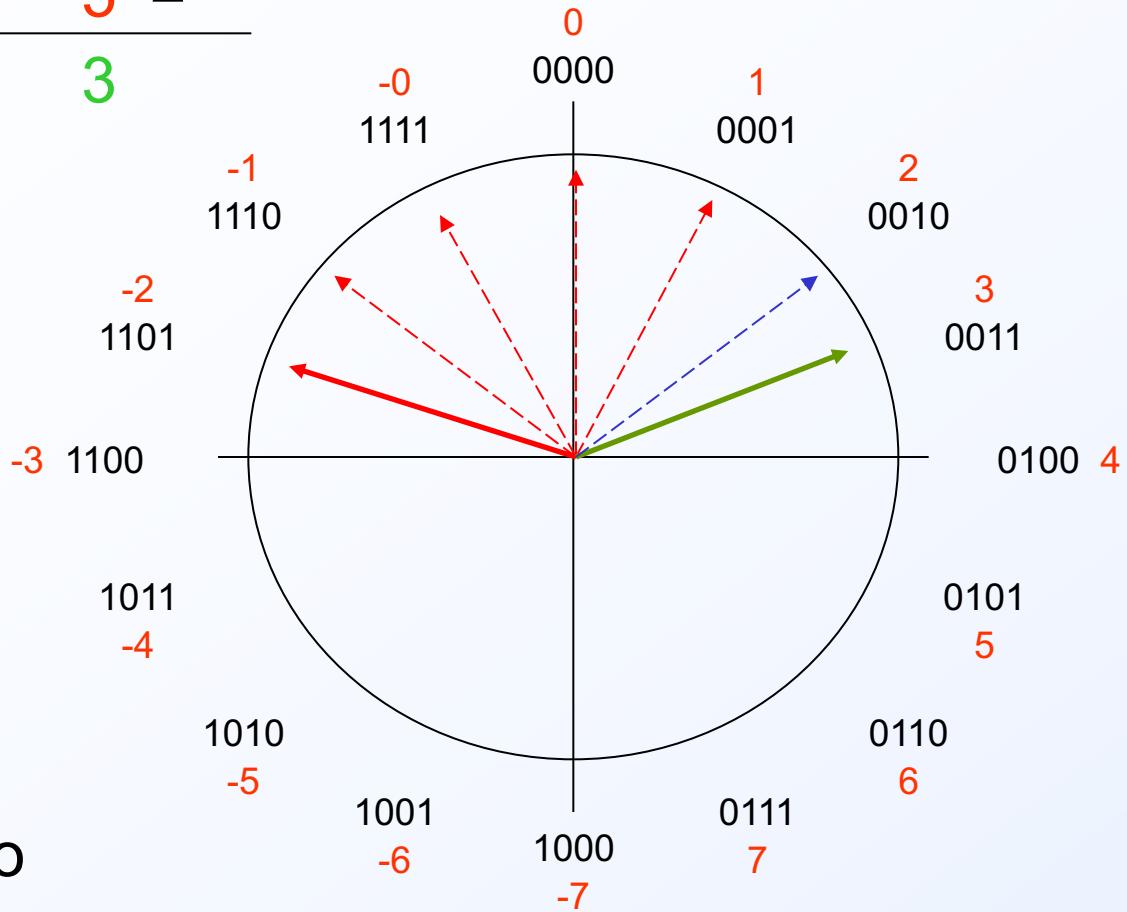


Numeri relativi in complemento a 1

$$\begin{array}{r} 1 \ 1 \ 0 \ 1 + \\ 0 \ 1 \ 0 \ 1 = \\ \hline 1 \ 0 \ 0 \ 1 0 \end{array} \quad \begin{array}{r} -2 + \\ 5 = \\ \hline 3 \end{array}$$

→ 1
0 0 1 1

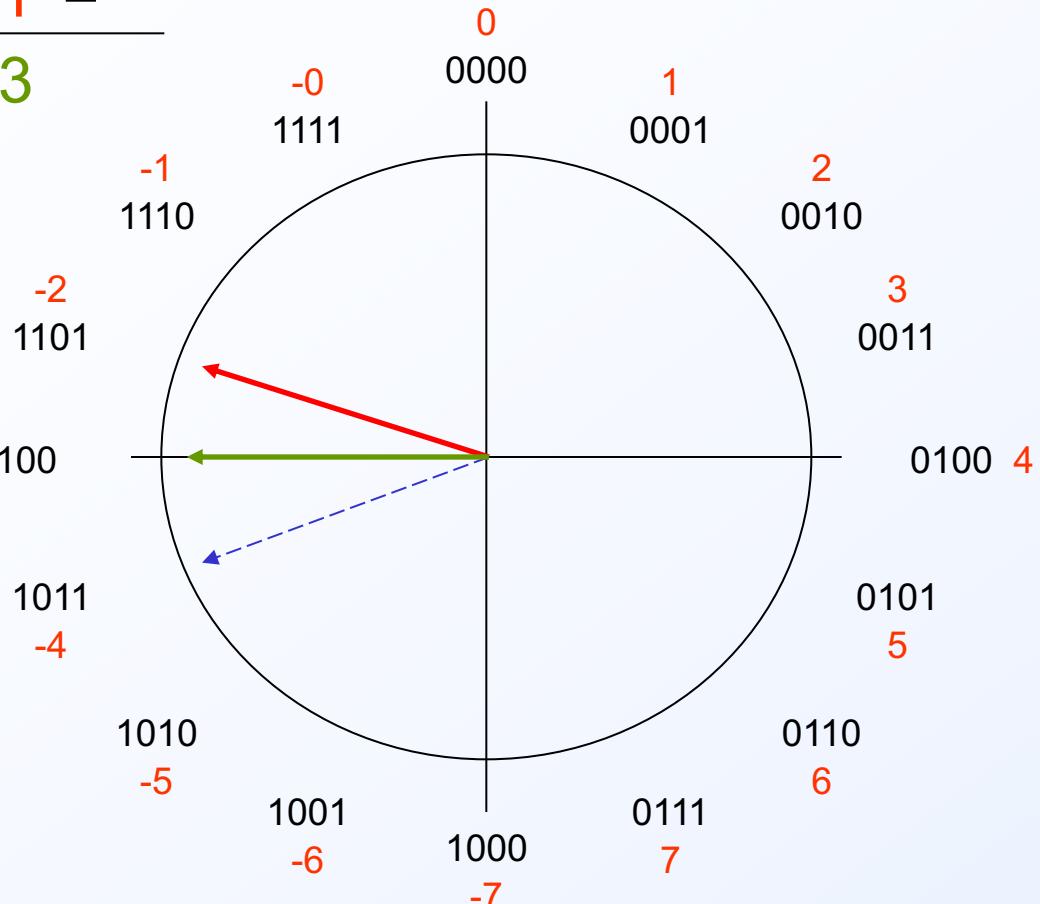
- Addendi discordi:
ma il risultato ha
segno positivo
- Risultato è quello
giusto diminuito di 1
- Si aggiunge il riporto
generato



Numeri relativi in complemento a 1

$$\begin{array}{r} 1 \ 1 \ 0 \ 1 + \\ 1 \ 1 \ 1 \ 0 = \\ \hline 1 \ 1 \ 0 \ 1 \ 1 \end{array} \quad \begin{array}{r} -2 + \\ -1 = \\ \hline -3 \end{array}$$

→ 1
1 1 0 0



- Addendi negativi: intanto non deve esserci un overflow
- In tal caso il risultato non può essere rappresentato sul numero di cifre a disposizione
- Risultato è quello giusto diminuito di 1
- Si aggiunge il riporto generato

Numeri relativi in complemento a 2

Rappresentazione in complemento a 2 su N cifre binarie di un numero con segno X:

- la cifra binaria più a sinistra rappresenta il segno ($0 = +$, $1 = -$)
- nel caso di segno positivo il numero è rappresentato in forma binaria su $N-1$ cifre come nel caso dei numeri naturali
- nel caso di segno negativo X si rappresenta come il complemento a 2 di $-X$

Decimale	binario
12	= 01100
-12	= 10100
-4	100
-3	101
-2	110
-1	111
0	000
1	001
2	010
3	011

Numeri relativi in complemento a 2

Alcune proprietà interessanti

- 0 si rappresenta con 000...00
- -1 si rappresenta con 111...11
- Il massimo numero positivo è 011..11
- Il minimo numero negativo è 100...00

Dato un numero negativo, scambiando 0 e 1 (operazione di complemento a 1) si ottiene il suo modulo diminuito di 1.

Es. su 4 bit: $-5_{10} = 1011_2$
 $0100_2 = 4_{10}$

Numeri relativi in complemento a 2

Caratteristiche:

- *Intervallo dei numeri rappresentabili su N bit:
[-2^{N-1}, 2^{N-1}-1]*
- *Unica rappresentazione per lo zero*
- *L'operazione di somma $\sum_{i=0}^{N-1} b_i 2^i$ effettuata operando sulla rappresentazione del numero indipendentemente dal segno produce, a meno di overflow, sempre il risultato corretto.*
- *Data una sequenza di N bit $b_{N-1}b_{N-2}\dots b_1b_0$ il numero rappresentato è dato da*

$$-b_{N-1} * 2^{N-1} + \sum_{i=0}^{N-2} b_i * 2^i$$

Ricordatevi che

- × Se ho a disposizione una sequenza di N bit in complemento a 2 allora, se il bit di segno è uguale a 0, NON si deve operare alcuna trasformazione per sapere quale intero positivo la sequenza rappresenta!!!

Numeri relativi in complemento a 2

Addendi Positivi ☺ OK

$$00111 + (+ 7)$$

$$\underline{00101} = (+ 5)$$

$$\underline{01100} \quad (+12)$$

Addendi Negativi ☺ OK

$$11001 + (- 7)$$

$$\underline{11011} = (- 5)$$

$$\cancel{\underline{110100}} \quad (- 12)$$

Addendi Segno opposto ☺ OK

$$00111 + (+ 7)$$

$$\underline{10110} = (- 10)$$

$$\underline{11101} \quad (- 3)$$

$$11001 + (- 7)$$

$$\underline{01010} = (+ 10)$$

$$\cancel{\underline{100011}} \quad (+ 3)$$

$$10100 + (- 12)$$

$$\underline{11011} = (- 5)$$

$$\underline{1 \ 01111} \quad (+ 15) ?!$$

overflow

Numeri relativi in complemento a 2

$$\begin{array}{r} \textcolor{red}{1} \ 1 \ 1 \ 1 \ 1 \ 0 \ 1 \ 0 \\ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0 + \\ 1 \ 1 \ 1 \ 1 \ 1 \ 0 \ 1 \ 0 \\ \hline \textcolor{blue}{1} \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \end{array}$$

riporti

scartato

$$\begin{array}{r} 10 \ + \\ - 6 \\ \hline 4 \end{array}$$

riporto
concorde
bit scartato

$$\begin{array}{r} \textcolor{red}{1} \ 0 \ 1 \ 1 \ 1 \ 0 \ 1 \ 0 \\ 1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0 + \\ 1 \ 0 \ 1 \ 1 \ 1 \ 0 \ 1 \ 1 \\ \hline \textcolor{blue}{1} \ 0 \ 1 \ 1 \ 0 \ 0 \ 1 \ 0 \ 1 \end{array}$$

riporti

scartato

$$\begin{array}{r} - 86 \ + \\ - 69 \\ \hline 101 \end{array}$$

riporto
discorde
overflow

Errore: il risultato è - 155

overflow

la somma di due numeri di segno uguale
non può dare risultato di segno diverso

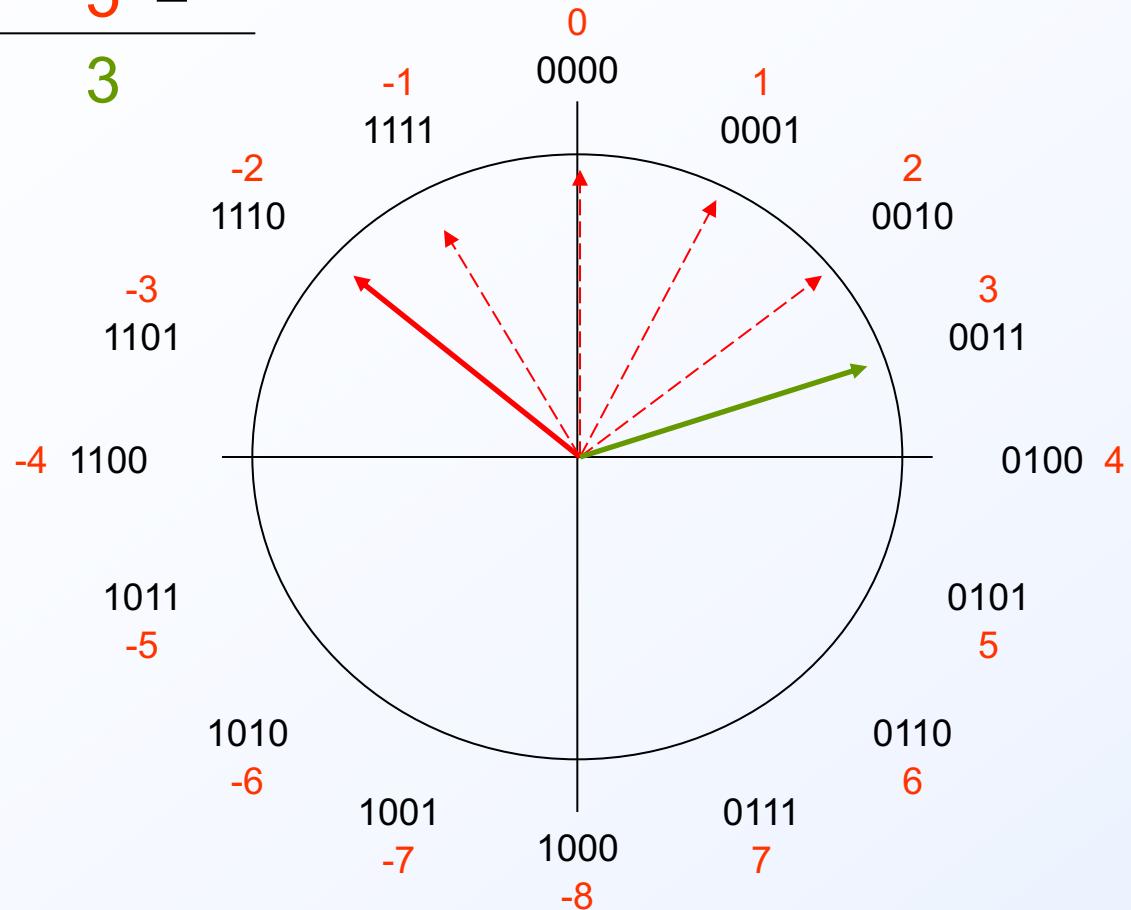
Overflow in complemento a 2

- ✗ Nella somma di due numeri relativi codificati nella rappresentazione del complemento a 2 su n cifre si ha overflow quando:
 - ✓ Criterio 1: ci sono addendi dello stesso segno **e** il segno del risultato è diverso dal segno degli addendi
- ✗ oppure
 - ✓ Criterio 2: il riporto dalla colonna n-2 alla colonna n-1 ed il riporto dalla colonna n-1 a quella oltre la cifra più significativa sono discordi (uno dei due è 0 e l'altro è 1)
- ✗ I due criteri sono equivalenti. Sfrutto quello che conviene

Numeri relativi in complemento a 2

$$\begin{array}{r} 1 \ 1 \ 1 \ 0 + \\ 0 \ 1 \ 0 \ 1 = \\ \hline 1 \ 0 \ 0 \ 1 \ 1 \end{array} \quad \begin{array}{r} -2 + \\ 5 = \\ \hline 3 \end{array}$$

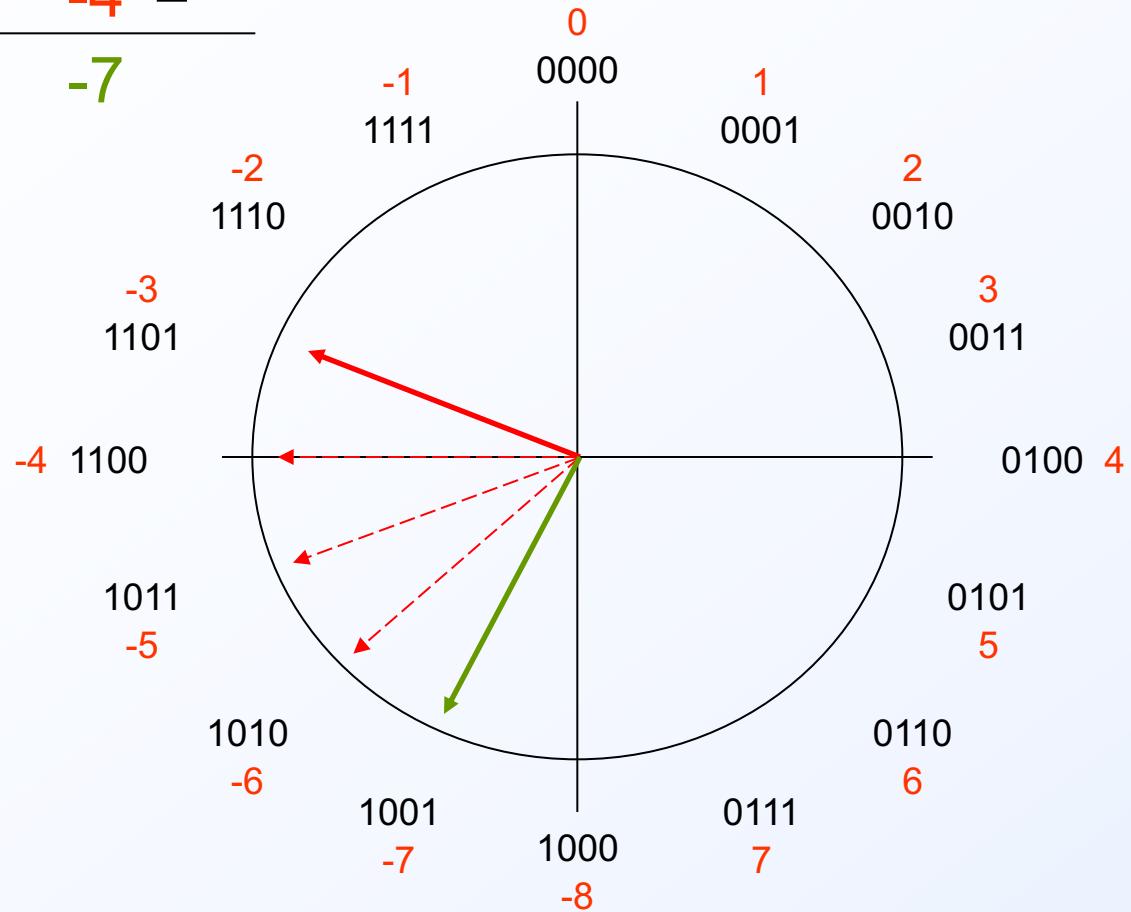
scartato



Numeri relativi in complemento a 2

$$\begin{array}{r} 1 \ 1 \ 0 \ 1 + \\ 1 \ 1 \ 0 \ 0 = \\ \hline 1 \ 1 \ 0 \ 0 \ 1 \end{array} \quad \begin{array}{r} -3 + \\ -4 = \\ -7 \end{array}$$

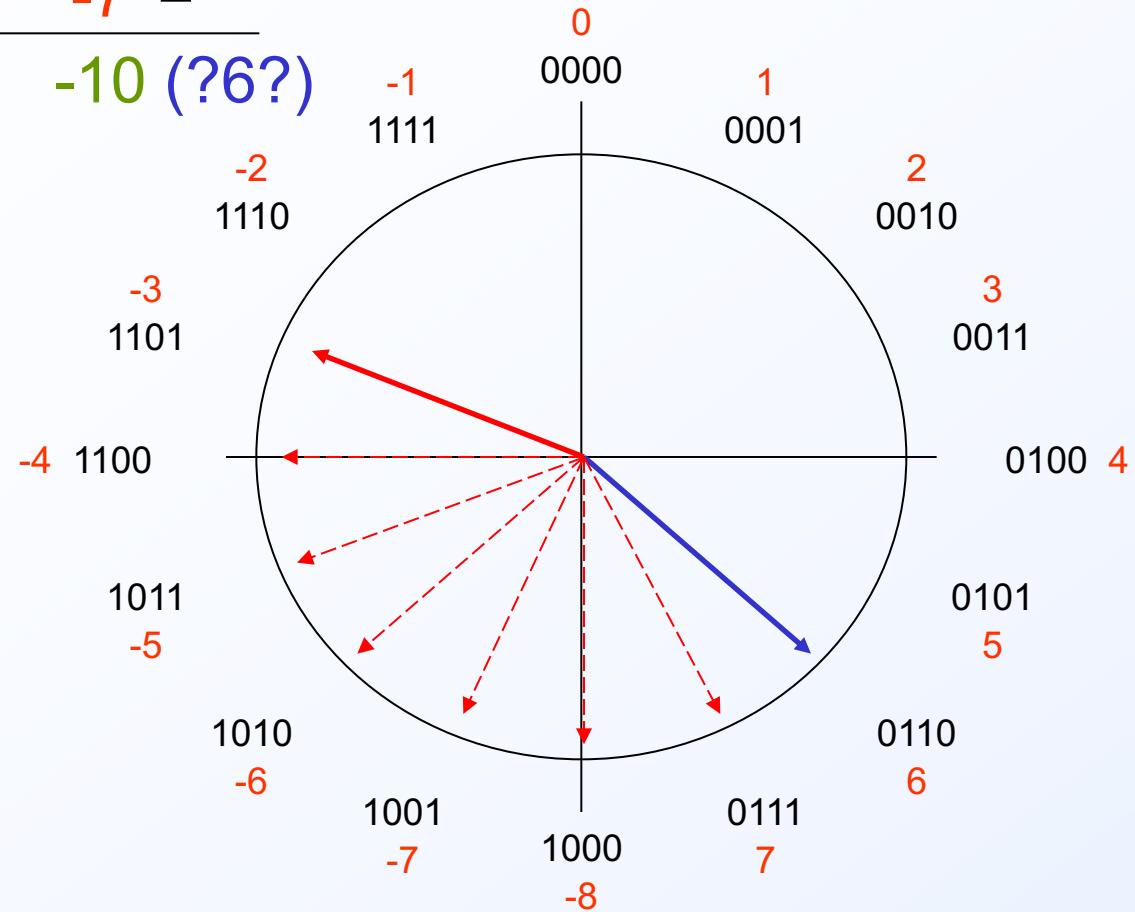
scartato



Numeri relativi in complemento a 2

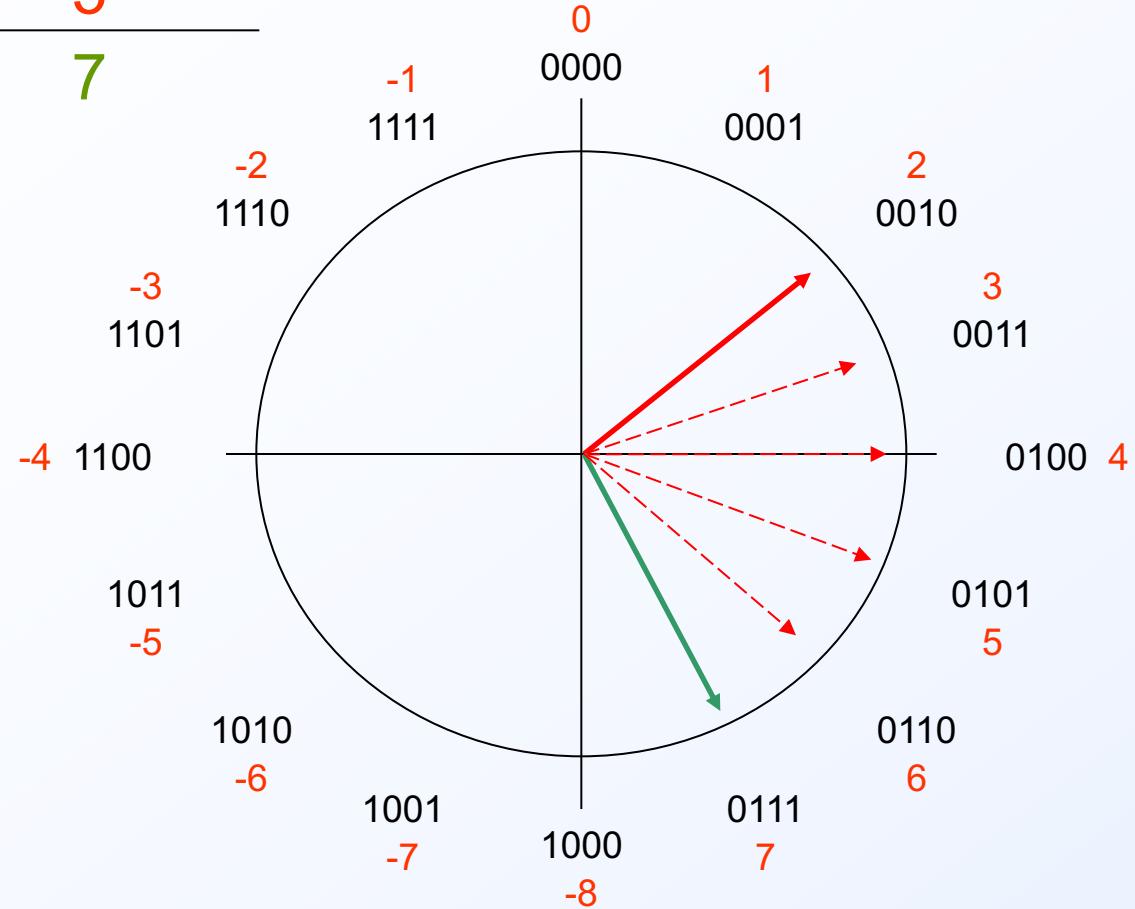
$$\begin{array}{r} 1 \ 1 \ 0 \ 1 + \\ 1 \ 0 \ 0 \ 1 = \\ \hline 1 \ 0 \ 1 \ 1 \ 0 \end{array} \quad \begin{array}{r} -3 + \\ -7 = \\ \hline -10 \ (\text{?6?}) \end{array}$$

scartato overflow



Numeri relativi in complemento a 2

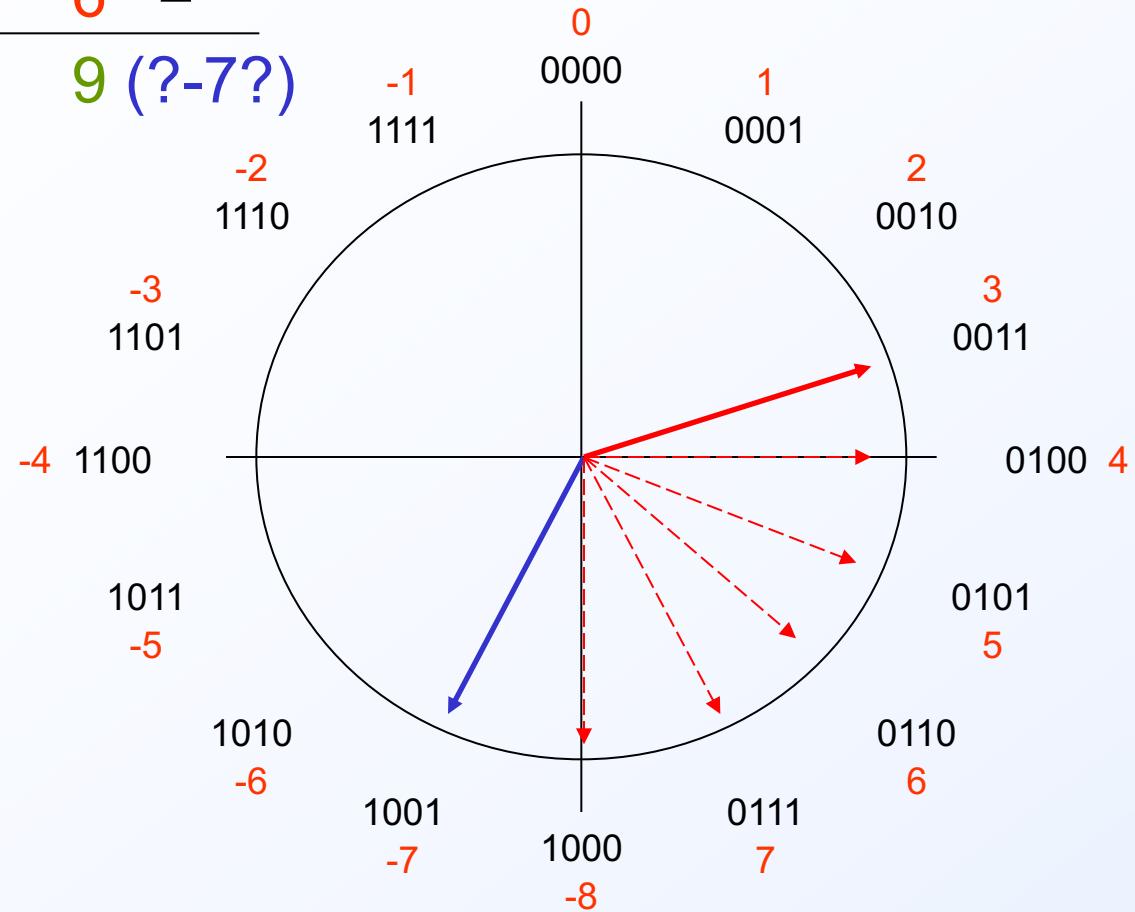
$$\begin{array}{r} 0 \ 0 \ 1 \ 0 + \\ 0 \ 1 \ 0 \ 1 = \\ \hline 0 \ 1 \ 1 \ 1 \end{array} \quad \begin{array}{r} 2 + \\ 5 = \\ \hline 7 \end{array}$$



Numeri relativi in complemento a 2

$$\begin{array}{r} 0 \ 0 \ 1 \ 1 + \\ 0 \ 1 \ 1 \ 0 = \\ \hline 1 \ 0 \ 0 \ 1 \end{array} \quad \begin{array}{r} 3 + \\ 6 = \\ \hline 9 \ (\text{?}-\text{?}) \end{array}$$

overflow



Estensione del segno

- ✗ Data la rappresentazione in complemento a 2 su N bit del numero X qual è la sua rappresentazione su M (M > N) bit?
- ✗ Basta copiare il bit di segno negli M – N bit più significativi
- ✗ Ad esempio:
 - X = 72 su N = 8 bit: **01001000**
 - X = 72 su M = 32 bit: **00000000 00000000 00000000 01001000**
 - X = -102 su N = 8 bit: **10011010**
 - X = -102 su M = 32 bit: **11111111 11111111 11111111 10011010**

Rappres. eccesso 2^{N-1}

4. Eccesso 2^{N-1} (per numeri rappresentati con N bit)

	-128	00000000	(-128 + 128 = 0)
	-127	00000001	(-127 + 128 = 1)
	.	.	
	.	.	
Ad esempio, usando otto cifre binarie, si ha la rappresentazione eccesso 128:	-2	01111110	(-2 + 128 = 126)
	.	.	
	.	.	
	25	10011001	(25 + 128 = 153)
	.	.	
	.	.	
	127	11111111	(127 + 128 = 255)

Rappres. eccesso 2^{N-1}

La rappresentazione su N cifre binarie di un numero è ottenuta sommando 2^{N-1} al numero stesso e codificando il numero ottenuto (che è positivo) in binario puro.

- ***Intervallo di rappresentazione dei numeri:***
 $[-2^{N-1}, 2^{N-1}-1]$
- ***Vantaggio: l'ordinamento viene mantenuto nella codifica***
- Questa codifica servirà per la rappresentazione dei numeri in virgola mobile

Rappres. eccesso X

4. Eccesso X

	-15	00000000	(-15 + 15 = 0)
	-14	00000001	(-14 + 15 = 1)
	.	.	.
Ad esempio, usando otto cifre binarie, si ha la rappresentazione eccesso 15:	0	00001111	(0 + 15 = 15)
	.	.	.
	25	00101000	(25 + 15 = 40)
	.	.	.
	240	11111111	(240 + 15 = 255)

Riassumendo

- ✗ Un informatico deve saper:
 - ✓ Che si possono denotare le quantità in basi (sistemi di numerazione) diversi da quella naturale (la base 10)
 - ✓ Scrivere un numero in base 10 (anche con la parte frazionaria) in qualunque altra base (sapendo spiegare perché)
 - ✓ Scrivere un numero in qualunque base (anche con la parte frazionaria) in base 10 (sapendo spiegare perché)
 - ✓ Conoscere le potenze di 2 (almeno fino a 2^{16})
 - ✓ Passare con immediatezza dalla base 2 a basi che sono potenze di 2 (le basi 8 e 16) e viceversa (sapendo spiegare perché)

Riassumendo

- ✗ Un informatico deve saper:
 - ✓ Effettuare la somma naturale di numeri in qualunque base e in particolare in base 2 (sapendo spiegare perché)
 - ✓ Effettuare il prodotto e la divisione per potenze della base (sapendo spiegare perché)
 - ✓ Conoscere le problematiche derivanti dall'aritmetica finita
 - ✓ Rappresentare un numero intero con segno usando N cifre in:
 - ◆ Modulo e segno
 - ◆ Complemento a 1
 - ◆ Complemento a 2
 - ◆ Eccesso 2^N
- sapendo spiegare perché la rappresentazione in complemento a 2 è soddisfacente rispetto alle altre

Riassumendo

- ✗ Un informatico deve saper:
 - ✓ Saper decodificare una sequenza di N cifre nel corrispondente numero intero con segno data la codifica usata
 - ✓ Spiegare il concetto di overflow e quali sono i criteri per segnalarlo nella somma di numeri relativi in complemento a 2

Qualche problema

- ✗ **Per tutti gli esercizi, lo svolgimento DEVE contenere tutti i calcoli ed i passaggi. Il solo risultato non sarà considerato sufficiente all'esame.**
 - ✓ **Esercizio 1:** Scrivere il numero 187 in base 2, 8, 16, 7, 5 e 9.
 - ✓ **Esercizio 2:** Considerate il numero 12. Quanto sarebbe in base 10 se fosse espresso in base 3, 4, 8, 16 e 2?
 - ✓ **Esercizio 3:** Considerate la sequenza 1001.1 in base 2. Convertitela in base 8 e 16
 - ✓ **Esercizio 4:** Considerate la sequenza 1001.1 in base 16. Convertitela in base 2 e 8
 - ✓ **Esercizio 5:** Considerate la sequenza 1001.1 in base 2 e moltiplicatela per 4 e 16. Dividetela per 2 e per 32. Inoltre, consideratela in base 5 e moltiplicatela per 125 e dividetela per 5. Verificate la correttezza di quanto avete fatto convertendo ogni risultato dalla base di lavoro alla base 10.

Qualche problema

- ✗ **Per tutti gli esercizi, lo svolgimento DEVE contenere tutti i calcoli ed i passaggi. Il solo risultato non sarà considerato sufficiente all'esame.**
 - ✓ **Esercizio 1:** Si consideri la sequenza 10001001. Dire quali numeri in base 10 sono rappresentati nelle diverse codifiche per i numeri relativi.
 - ✓ **Esercizio 2:** Considerate i numeri 12 e -21.
 - ◆ a) codificarli su 8 bit in complemento a 2:
 - ◆ b) sommarli in complemento a 2
 - ◆ c) decodificare il risultato
 - ✓ **Esercizio 3:** Codificare su N=8 bit:
 - ◆ -123 in complemento a 2
 - ◆ -14 in complemento a 1
 - ◆ -6 in modulo e segno
 - ◆ -110 in eccesso 2^{N-1}