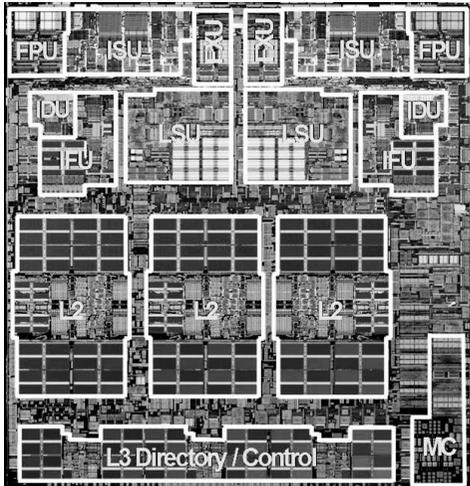




Università degli Studi di Cassino



Corso di Calcolatori Elettronici I

*Rappresentazione dei dati
numerici
Aritmetica dei registri*

Anno Accademico 2007/2008

Francesco Tortorella

BIG IDEA: Bits can represent anything!!

- Caratteri
 - 26 lettere \Rightarrow 5 bits ($2^5 = 32$)
 - Minuscole/maiuscole + punteggiatura \Rightarrow 7 bits (in 8) (“ASCII”)
 - Codice standard per rappresentare tutti I linguaggi del mondo \Rightarrow 8,16,32 bits (“Unicode”)
www.unicode.com
- Valori logici
 - 0 \Rightarrow False, 1 \Rightarrow True
- Colori
 - 3 valori di intensità per i tre colori fondamentali RGB (3 x 8 bit = 24 bit)
- Locazioni / indirizzi comandi
- **Ricorda:** N bits \Rightarrow al più 2^N oggetti

Come rappresentiamo i numeri ?

- Base di numerazione: dieci
 - **Cifre: 0 1 2 3 4 5 6 7 8 9**
- Rappresentazione posizionale
 - **possibile per la presenza dello zero**

Esempio:

3201 =

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

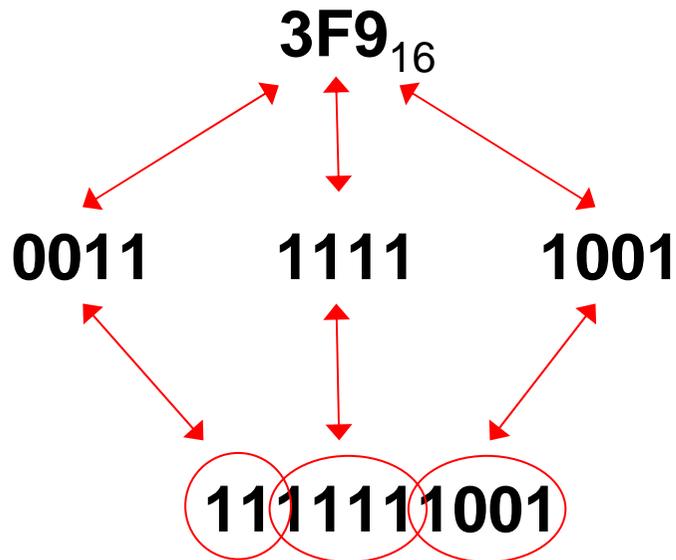
In generale ...

- Rappresentazione in base B \rightarrow B-1 cifre
 - 0 1 2 ... B-1
- Rappresentazione dei numeri:
 - $d_{31}d_{30} \dots d_2d_1d_0$ è un numero a 32 cifre
 - valore =
$$d_{31} \times B^{31} + d_{30} \times B^{30} + \dots + d_2 \times B^2 + d_1 \times B^1 + d_0 \times B^0$$

Altre basi

- B=2 :
 - cifre: 0 1
 - 1011010 →
 $1x2^6 + 0x2^5 + 1x2^4 + 1x2^3 + 0x2^2 + 1x2 + 0x1 =$
 $64 + 16 + 8 + 2 = 90$ 7 cifre binarie → 2 cifre decimali
- B=16 :
 - cifre: 0 1 2 3 4 5 6 7 8 9 A B C D E F
 - 524 →
 $5x16^2 + 2x16 + 4x1 = 1316$
3 cifre esadecimali → 4 cifre decimali

Siccome $16=2^4$, il passaggio tra le rappresentazioni in base 2 e in base 16 è molto semplice:



	base	
10	16	2
00	0	0000
01	1	0001
02	2	0010
03	3	0011
04	4	0100
05	5	0101
06	6	0110
07	7	0111
08	8	1000
09	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Quale base usare ?

- Decimale
 - naturale per gli esseri umani.
- Esadecimale
 - utile (agli esseri umani) per esaminare lunghe stringhe di bit
- Binaria
 - rappresentazione ottimale per il calcolatore

... perché non usare una codifica binaria della rappresentazione in base 10 ?

Conversione base 10 → base 2 (interi)

Come ottenere la rappresentazione in base 2 di un numero intero T rappresentato in base 10 ?

Supponiamo:

$$T = c_{n-1}x2^{n-1} + c_{n-2}x2^{n-2} + \dots + c_2x2^2 + c_1x2^1 + c_0x2^0$$

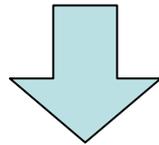
$$c_i \in \{0,1\}$$

Non conosciamo:

- le cifre c_i
- il numero di cifre n

Conversione base 10 \rightarrow base 2 (interi)

$$\begin{aligned} T &= c_{n-1}x2^{n-1} + c_{n-2}x2^{n-2} + \dots + c_2x2^2 + c_1x2^1 + c_0x2^0 = \\ &= (c_{n-1}x2^{n-2} + c_{n-2}x2^{n-3} + \dots + c_2x2^1 + c_1) x2 + c_0 = \\ &= Q_0x2 + c_0 \end{aligned}$$



$$Q_0 = T \text{ div } 2 \quad c_0 = T \text{ mod } 2$$

$$Q_0 = (c_{n-1}x2^{n-3} + c_{n-2}x2^{n-4} + \dots + c_2)x2 + c_1 = Q_1x2 + c_1$$

$$Q_1 = Q_0 \text{ div } 2 \quad c_1 = Q_0 \text{ mod } 2$$

Conversione base 10 → base 2 (interi)

```
void convint(int T,int c[],int &n)
{
    int Q;
    n=0;Q=T;
    do {
        c[n]=Q%2;
        Q=Q/2;
        n++;
    } while (Q!=0);
}
```

La conversione genera le cifre a partire da quella meno significativa

Esempio:

$$75_{10} \rightarrow ?_2$$

Aritmetica in base 2

Le operazioni aritmetiche si svolgono in maniera analoga a quanto si fa in base 10.

+	0	1
0	0	1
1	1	10

*	0	1
0	0	0
1	0	1

“tavola pitagorica” in base 2

Aritmetica dei registri

- I registri di memoria sono supporti di lunghezza finita
- Ciò impone delle restrizioni all'insieme di numeri rappresentabili e, di conseguenza, dei vincoli all'aritmetica
- Registro a N bit $\rightarrow 2^N$ valori diversi rappresentabili
 - Es.: 8 bit \rightarrow 256 valori possibile rappresentare l'intervallo [0,255]

Aritmetica dei registri

Non ci sono problemi nel caso in cui l'operazione produce un risultato rappresentabile nel registro

0	1	1	0	1	1	0	1
---	---	---	---	---	---	---	---

 + 1 0 9 +

1	0	0	0	1	0	0	1
---	---	---	---	---	---	---	---

 = 1 3 7 =

1	1	1	1	0	1	1	0
---	---	---	---	---	---	---	---

2	4	6
---	---	---

Aritmetica dei registri

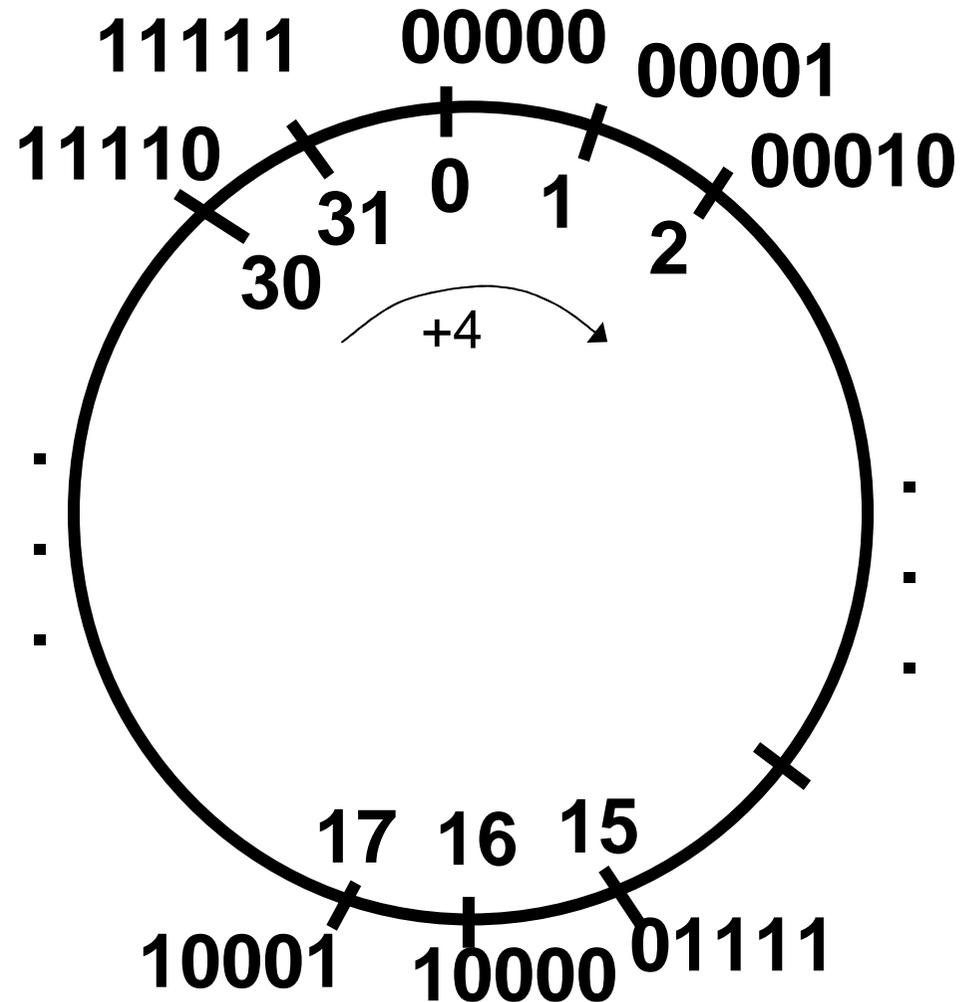
Se l'operazione fornisce un risultato R non rappresentabile, si produce un riporto uscente dal registro, mentre all'interno rimane una parte della rappresentazione del risultato ($R \bmod 2^N$)

1	1	1	0	1	1	0	1	+	2	3	7	+
1	0	0	0	1	0	0	1	=	1	3	7	=
1	0	1	1	1	0	1	1	0				

3	7	4	
1	1	8	

Aritmetica dei registri

- L'aritmetica dei registri a N bit è caratterizzata da una congruenza mod 2^N
- Quindi, per N=5:
 - $30+4=2$!



Aritmetica dei registri

- Il riporto uscente dal registro, generato da un'addizione tra numeri interi, si definisce *carry*
- Il prestito uscente dal registro, generato da una sottrazione tra numeri interi, si definisce *borrow*

$\begin{array}{r} 4 \\ + 2 \\ \hline 6 \end{array}$	$\begin{array}{r} 00100 \\ + 00010 \\ \hline 0 00110 \end{array}$	$\begin{array}{r} 10 \\ + 26 \\ \hline 4 \end{array}$	$\begin{array}{r} 01010 \\ \underline{11010} \\ 1 00100 \end{array}$	carry
---	---	---	--	--------------

$\begin{array}{r} 4 \\ - 2 \\ \hline 2 \end{array}$	$\begin{array}{r} 00100 \\ - 00010 \\ \hline 0 00010 \end{array}$	$\begin{array}{r} 10 \\ - 26 \\ \hline 16 \end{array}$	$\begin{array}{r} 01010 \\ \underline{11010} \\ 1 10000 \end{array}$	borrow
---	---	--	--	---------------

Rappresentazione dei dati

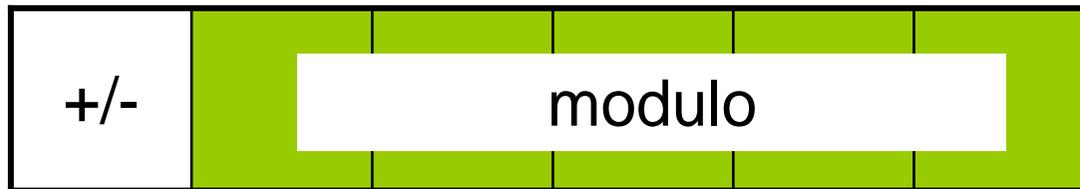
Rappresentazione in segno e modulo

Rappresentazione in complementi alla base

Rappresentazione per eccessi

Rappresentazione dei numeri negativi

- Soluzione più immediata: segno + modulo



Possibile
convenzione:

0 → + 1 → -

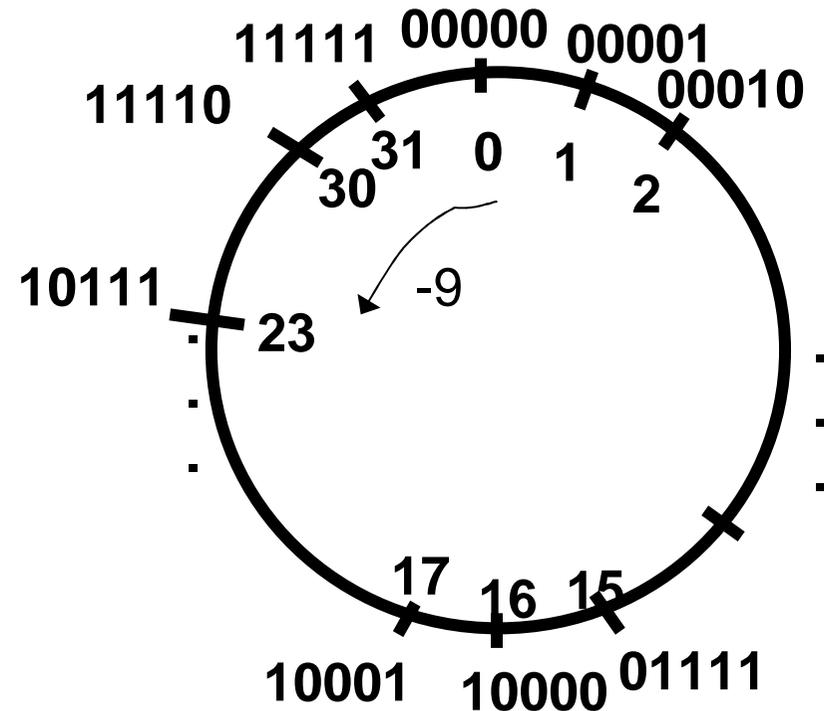
- Problemi
 - dove mettere il segno ?
 - doppia rappresentazione per lo zero (+0, -0)
 - operazioni alquanto complicate

Rappresentazione dei numeri negativi

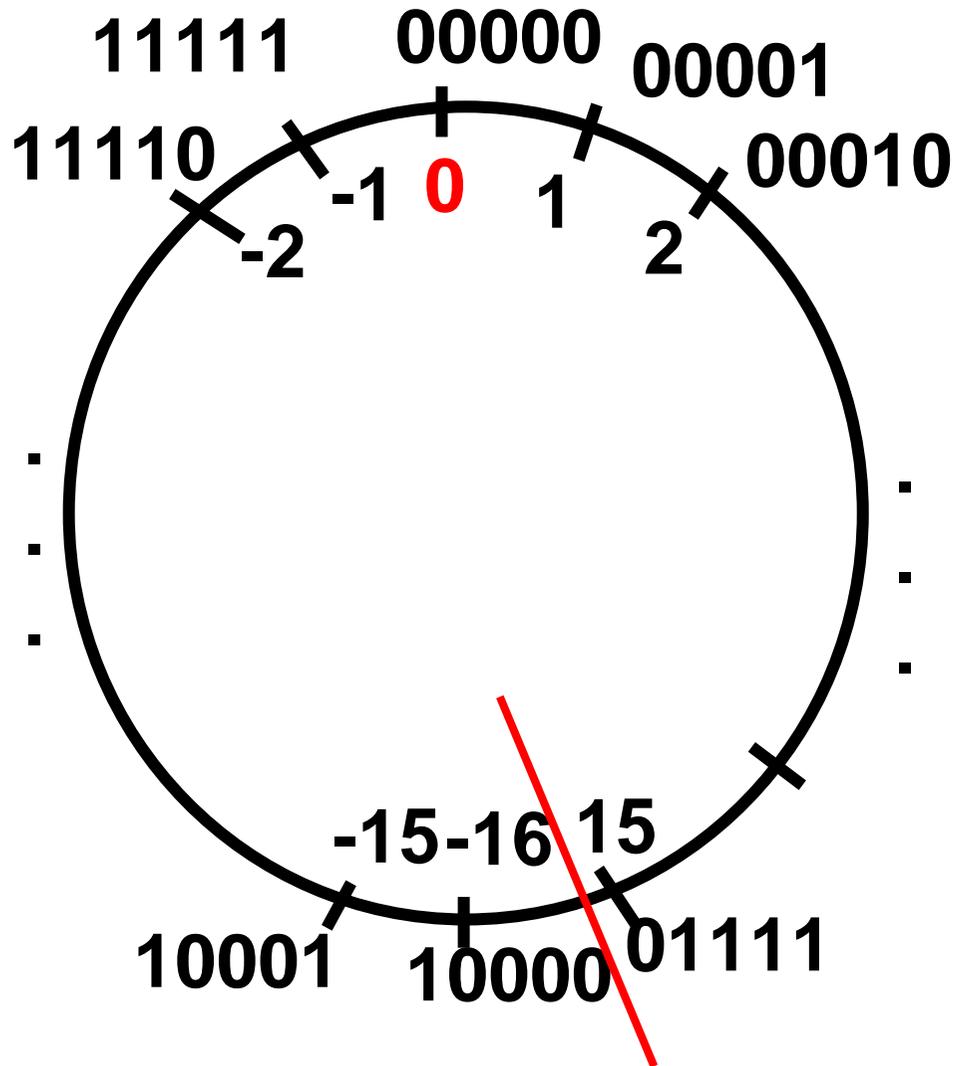
- Soluzione alternativa

- Che cosa succede in un registro a N bit quando si sottrae un numero da 0 ?

$$\begin{array}{r} \boxed{0} \boxed{0} \boxed{0} \boxed{0} \boxed{0} \quad - \\ \boxed{0} \boxed{1} \boxed{0} \boxed{0} \boxed{1} \quad = \\ 1 \boxed{1} \boxed{0} \boxed{1} \boxed{1} \boxed{1} \end{array}$$



Complementi alla base



Caratteristiche:

- 2^{N-1} non-negativi
- 2^{N-1} negativi
- uno zero
- quanti positivi ?
- confronto ?
- rappr. dello zero

Complementi alla base

- L'intervallo di numeri rappresentati è $[-2^{N-1} \quad +2^{N-1}-1]$
- La rappresentazione di un numero x nell'intervallo è data da $R(x)=(x+2^N) \bmod 2^N$
- Il bit più significativo è indicativo del segno (“bit di segno”)

00000	0
00001	+1
00010	+2
00011	+3
.	.
.	.
01110	+14
01111	+15
<hr/>	
10000	-16
10001	-15
10010	-14
.	.
.	.
11101	-3
11110	-2
11111	-1

Calcolo rapido del complemento alla base

- Per ottenere rapidamente la rappresentazione in complemento alla base di un numero negativo su N bit
 - si estrae la rappresentazione del valore assoluto del numero su N bit
 - si complementano le cifre ad una ad una
 - si aggiunge 1
- Es.: complemento alla base su 8 bit di -33
 $33_{10} = 00100001$ $11011110 + 1 = 11011111$

Operazioni in complemento alla base

- Le addizioni si realizzano direttamente sulle rappresentazioni in quanto $R(x+y)=R(x)+R(y)$
- Anche le sottrazioni si valutano tramite addizioni, ponendo $x-y$ come $x+(-y)$; di conseguenza $R(x-y)=R(x)+R(-y)$
- Nel caso in cui l'operazione produce un numero al di fuori dell'intervallo di rappresentazione si ha un *overflow*

Operazioni in complemento alla base

$$\begin{array}{r}
 +4 \quad 0100 \\
 \underline{+2 \quad +0010} \\
 +6 \quad 0|0110
 \end{array}$$

$$\begin{array}{r}
 +4 \quad 0100 \\
 \underline{-2 \quad +1110} \\
 +2 \quad 1|0010
 \end{array}$$

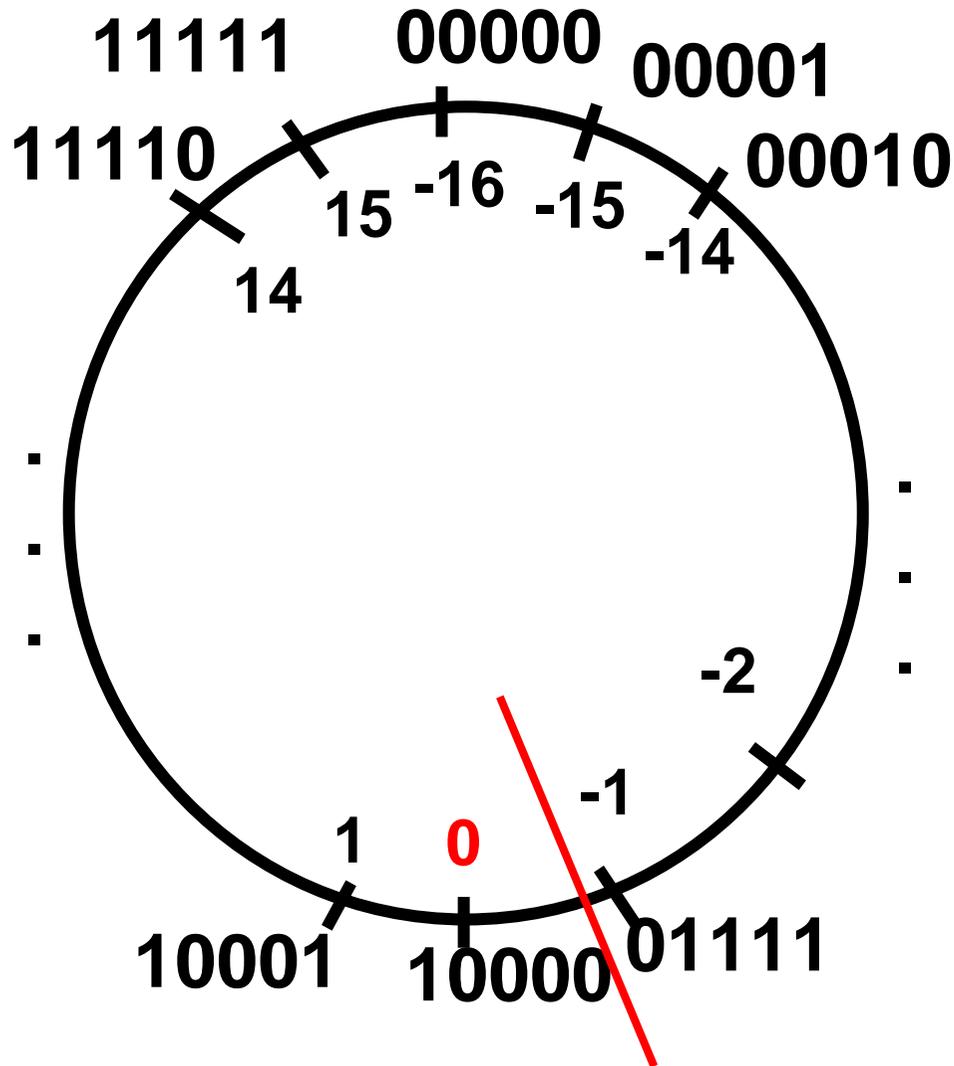
$$\begin{array}{r}
 -4 \quad 1100 \\
 \underline{-2 \quad +1110} \\
 -6 \quad 1|1010
 \end{array}$$

$$\begin{array}{r}
 +5 \quad 0101 \\
 \underline{+4 \quad +0100} \\
 -7 \quad 0|1001
 \end{array}$$

$$\begin{array}{r}
 -6 \quad 1010 \\
 \underline{-3 \quad +1101} \\
 +7 \quad 1|0111
 \end{array}$$

overflow

Rappresentazione per eccessi (polarizzata)



Caratteristiche:

- 2^{N-1} non-negativi
- 2^{N-1} negativi
- uno zero
- quanti positivi ?
- confronto ?
- rappr. dello zero

Eccessi

- L'intervallo di numeri rappresentati è $[-2^{N-1} \quad +2^{N-1}-1]$
- La rappresentazione di un numero x nell'intervallo è data da $R(x)=x+2^{N-1}$
- Il bit più significativo è indicativo del segno (“bit di segno”)

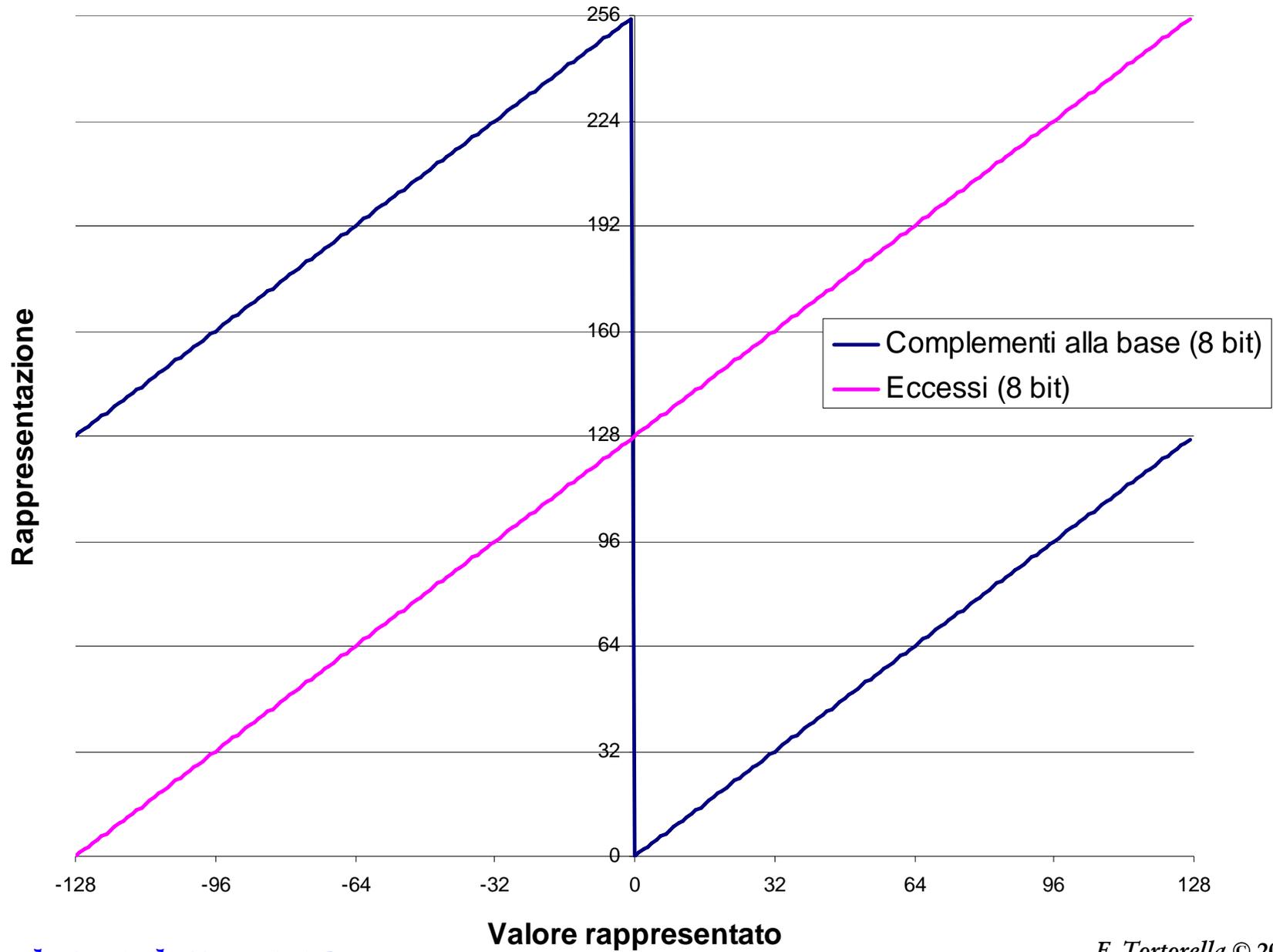
00000	-16
00001	-15
00010	-14
00011	-13
.	.
.	.
01110	-2
01111	-1
<hr/>	
10000	0
10001	+1
10010	+2
.	.
.	.
11101	+13
11110	+14
11111	+15

Operazioni in eccessi

- Le addizioni si realizzano direttamente sulle rappresentazioni in quanto $R(x+y)=R(x)+R(y)$
- Anche le sottrazioni si valutano tramite addizioni, ponendo $x-y$ come $x+(-y)$; di conseguenza $R(x-y)=R(x)+R(-y)$
- **Achtung!** Siccome $R(x)+R(y)=x+y+2^{N-1}+2^{N-1}$, il risultato necessita di una correzione
- Nel caso in cui l'operazione produce un numero al di fuori dell'intervallo di rappresentazione si ha un **overflow**

Confronto tra complementi alla base ed eccessi

- Entrambe permettono di realizzare una sottrazione tramite addizione (macchine aritmetiche più semplici)
- Le operazioni in eccessi richiedono un aggiustamento finale
- La rappresentazione in complementi rende più difficile il confronto



Numeri signed e unsigned

- Un registro di N bit può rappresentare:
 - Numeri assoluti nel range $[0, 2^N-1]$ → numeri **unsigned**
 - Numeri relativi nel range $[-2^{N-1}, 2^{N-1}-1]$ → numeri **signed**
- } **C**
- Dalla stringa di bit nel registro non si può risalire al tipo di numero memorizzato. Quali sono le conseguenze ?
 - Operazioni aritmetiche indipendenti dalla rappresentazione
→ nessuna conseguenza
 - Confronto dipendente dalla rappresentazione
→ due tipi di confronto
 - $X = 10001$ $Y = 01110$

$X > Y ?$

- unsigned: SI $(17 > 14)$
- signed: NO $(-15 < +14)$