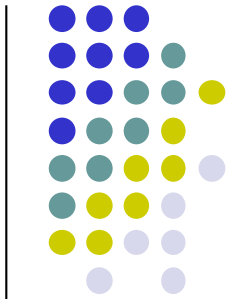
Codifica numeri naturali

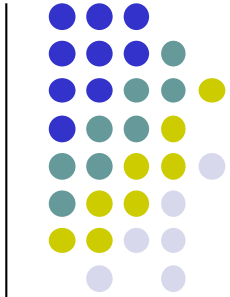


- Poiché l'unità elementare di informazione in un elaboratore è il bit, che corrisponde alle due cifre *0* e *1,* in modo naturale viene utilizzato il sistema di numerazione posizionale in base *2*

- Fissato il numero di bit *k* da utilizzare nella rappresentazione
  1. si converte il numero di partenza nella base *2*
  2. si antepongono bit uguali a *0* al numero determinato fino ad ottenere complessivamente esattamente *k* bit

  NB: l'aggiunta dei bit pari a *0* nel passo 2. è necessaria perché nella memorizzazione del numero bisogna specificare per ogni bit (anche per i più significativi che convenzionalmente non indichiamo quando pari a *0*) lo stato del relativo dispositivo bistabile; in caso contrario si potrebbero avere errori di rappresentazione.

Esempio

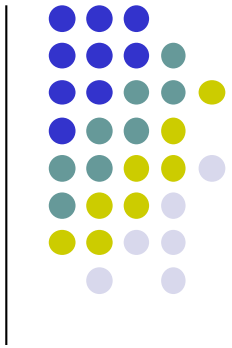


- *43 (con k=8 e B'=2)*

43:2 = 21 con resto 1
21:2 = 10 con resto 1
10:2 = 5 con resto 0
5:2 = 2 con resto 1
2:2 = 1 con resto 0
*1:2 = **0** con resto 1*

Quindi $43=101011_2$, la cui rappresentazione a *8* bit è

00101011

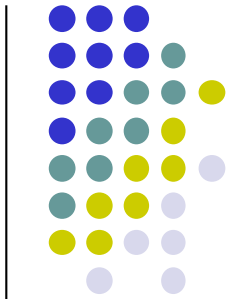


- Chiaramente, fissato il numero di bit *k*, è possibile rapprentare al più 2^k numeri interi, che vanno da *0 (0…0)* a 2^k*-1 (1…1)*

- Quindi, l'intervallo dei numeri rappresentabile con *k* bit è $[0,2^k-1]$

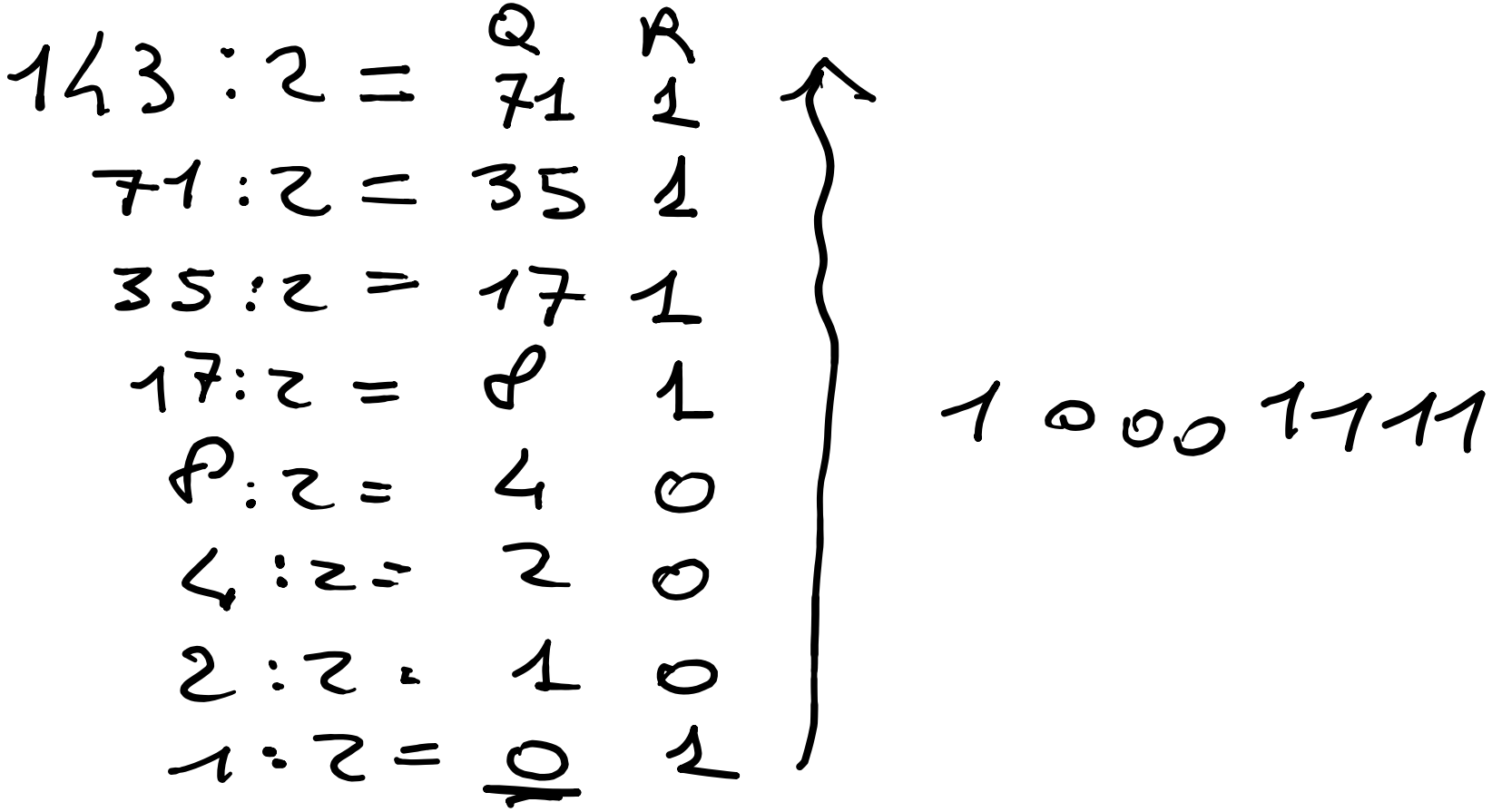
- Overflow (trabocco): errore che si verifica quando si tenta di rappresentare un numero al di fuori dell'intervallo, ad esempio quando il risultato di un'operazione aritmetica è troppo grande

Esercizi di conversione tra basi diverse (1/2)

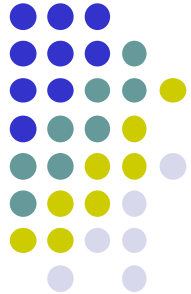


- Convertire in binario i seguenti numeri in base 10
  - 143, 312, 91, 123
- Convertire in base 10 i seguenti numeri binari
  - 1011, 1000111, 1010001, 11111000
- Convertire in base 10 i seguenti numeri in base esadecimale
  - 1AB0, ABCD, F0E1, 1234
- Convertire in base 10 i seguenti numeri ottali
  - 76022, 1010, 6663, 1234
- Convertire i seguenti numeri decimali in notazione esadecimale e ottale
  - 3500, 531
- Convertire i seguenti numeri dalla base 16 alle basi 8, 4 e 2
  - D15, 64, ABCD, FFE

1) Convertire in binario i seguenti numeri in base 10
143, 312, 91, 123
2) Convertire in base 10 i seguenti numeri binari
1011, 1000111, 1010001, 11111000



1) Convertire in binario i seguenti numeri in base 10
143, 312, 91, 123
2) Convertire in base 10 i seguenti numeri binari
1011, 1000111, 1010001, 11111000

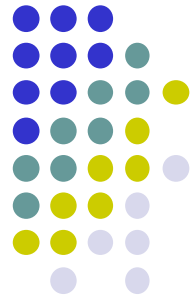


$1000111 =$

$1 \cdot 2^0 + 1 \cdot 2^1 + 1 \cdot 2^2 + 1 \cdot 2^6 =$

$= 1 + 2 + 4 + 64 = 71$

- Convertire in base 10 i seguenti numeri in base esadecimale
  - 1AB0, ABCD, F0E1, 1234
- Convertire in base 10 i seguenti numeri ottali
  - 76022, 1010, 6663, 1234

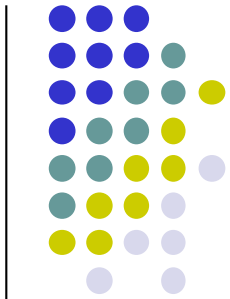


$$1AB0 = \underset{(11)}{B} \cdot 16 + \underset{(10)}{A} \cdot 16^2 + 1 \cdot 16^3 =$$

$$= \text{calcolatrice :)}$$

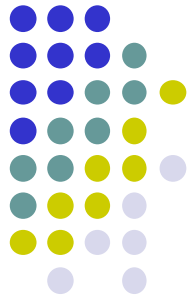
$$1010 = 1 \cdot 8^1 + 1 \cdot 8^3 = 8 + 512 = 520$$

Esercizi di conversione tra basi diverse (2/2)



- Convertire i seguenti numeri dalla base 8 alle basi 16, 4 e 2
  - 67201, 10777, 73601, 64
- Convertire i seguenti numeri
  - 201102 da base 3 a base 16
  - 3201 da base 4 a base 7
  - 303 da base 4 a base 6
  - 754 da base 9 a base 16
- Convertire i seguenti numeri dalla base 9 alla base 3
  - 82704, 64, 108887, 12345
- Convertire i seguenti numeri dalla base 3 alla base 9
  - 211200212, 21022, 20001, 202101

Aritmetica Binaria



SOMMA

0+0=0

0+1=1

1+0=1

1+1=0 e riporto di1

ESEMPIO

```
 1 1      1 1 1          riporti
  11 001 011 +    I addendo
  01 101 110 =    II addendo
 100 111 001
```

SOTTRAZIONE

0-0=0

0-1=1 e prestito di 1

1-0=1

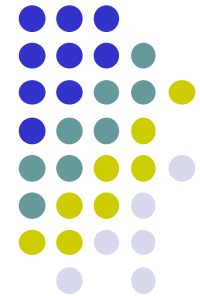
1-1=0

ESEMPIO

```
 1 1      1 1 1          prestiti
 100 111 001 -    minuendo
  11 001 011 =    sottraendo
  01 101 110
```

come nell'aritmetica decimale, i riporti/prestiti
rappresentano quantità addizionali da sommare/sottrarre

Aritmetica Binaria: moltiplicazione



La moltiplicazione richiede il calcolo dei prodotti parziali e la loro somma, così come nella classica aritmetica in base 10

0x0=0

0x1=0

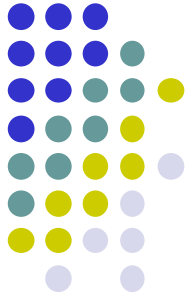
1x0=0

1x1=1

ESEMPIO

```
  10110x
    101=
  -----
  10110
 00000
10110
-------
1101110
```

Aritmetica Binaria: divisione



- Chiaramente la divisione per 0 non è definita
- Come nell'aritmetica classica in base 10, la divisione prevede sottrazioni tra parti del dividendo e del divisore

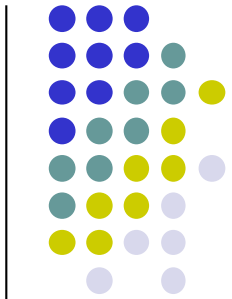
0:1=0

1:1=1

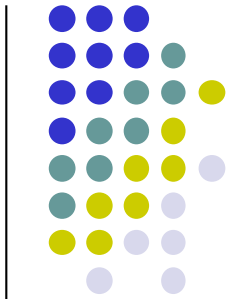
ESEMPIO

```
1101001101:10001
 10001        110001
 -----
  10010
  10001
  -----
    11101
    10001
    -----
     1100
```

Codifica numeri interi (positivi e negativi)



- Esistono diverse codifiche
- Tra le più note:
  - rappresentazione in modulo e segno
    - più semplice e diretta
  - rappresentazione in complemento a *2*
    - ha il pregio di poter effettuare somme algebriche, ossia la sottrazione *a-b* equivale ad effettuare la somma *a+(-b)*
    - *a* livello di circuteria elettronica ciò consente di poter effettuare somme e sottrazioni in modo unificato tramite un unico dispositivo sommatore che opera sulle codifiche di *a* e *-b*
  - *… (tra un po' di slides ci ritorniamo)…*



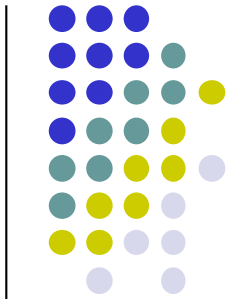
- Rappresentazione in modulo e segno a *k* bit:
  - *1* bit per il segno, solitamente il più significativo posto ad *0* per indicare il segno + e ad *1* per indicare il segno –
  - *k-1* bit per il modulo o valore assoluto, secondo la codifica dei numeri naturali

- L'intervallo dei numeri rappresentabili quindi è

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

- Si noti che esistono due codifiche possibili per il numero *0*, ossia *00…0* (corrispondente a *+0*) e *10…0* (corrispondente a *-0*)

Esempio



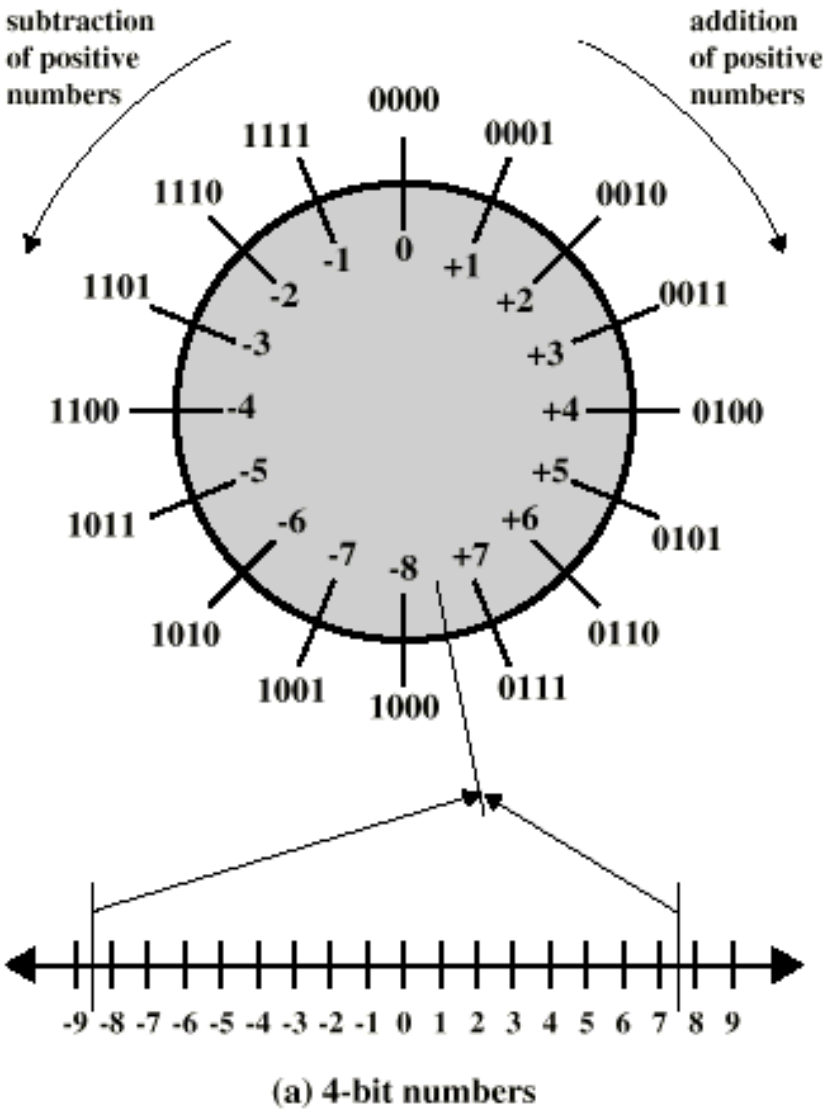
Rappresentazione in modulo e segno a *k=8* bit dei numeri *26* e *-26*

26:2 = 13 con resto 0
13:2 = 6 con resto 1
6:2 = 3 con resto 0
3:2 = 1 con resto 1
*1:2 = **0** con resto 1*

Quindi $26=11010_2$, per cui

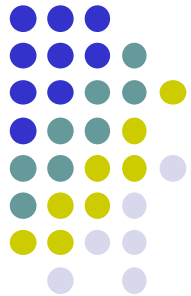
1. rappresentazione di *26*: *00011010*
2. rappresentazione di *-26*: *10011010*

- Rappresentazione in complemento a 2 a *k* bit:



(a) 4-bit numbers

Valore binario $b_3b_2b_1b_0$	Notazione modulo e segno	Notazione complemento a 2
0000	+ 0	0
0001	+ 1	+ 1
0010	+ 2	+ 2
0011	+ 3	+ 3
0100	+ 4	+ 4
0101	+ 5	+ 5
0110	+ 6	+ 6
0111	+ 7	+ 7
1000	– 0	– 8
1001	– 1	– 7
1010	– 2	– 6
1011	– 3	– 5
1100	– 4	– 4
1101	– 5	– 3
1110	– 6	– 2
1111	– 7	– 1



- Poiché esiste un'unica codifica del numero *0* (*-0* non viene rappresentato), l'intervallo dei numeri rappresentabili è

  $[-2^{k-1}, 2^{k-1}-1]$

- I numeri non negativi coincidono con la rappresentazione in modulo e segno

- Infatti la rappresentazione di un numero non negativo si ottiene semplicemente convertendolo in binario

- La rappresentazione di un numero negativo $-N$ si ottiene facendo la conversione in binario a k bit del numero 2^k-N

- Una semplice regola di conversione:
  - si converte in binario a k bit il numero N
  - si complementano tutti i bit
  - si somma *1*

- Oppure: si complementano tutti i bit da sinistra verso destra, fino all'ultimo bit pari ad 1 escluso

Complemento BIT A BIT