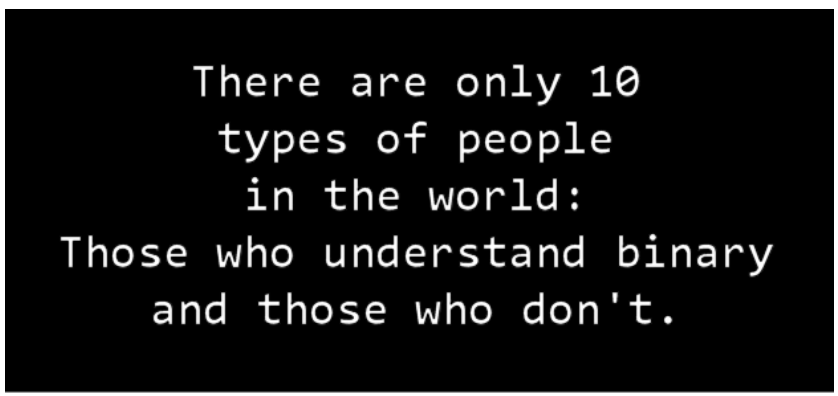
Tecniche della Programmazione, lez. 12

Sistemi di numerazione e aritmetica binaria

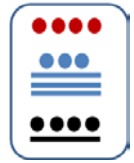
Rappresentazione dei numeri

- interi: complemento a due
- caratteri
- reali: Floating Point

E parliamo anche di `espressione condizionale` e *obfuscation*



Sistemi di numerazione: NUMERI e NUMERALI



Un numero è un concetto … il NUMERALE rappresenta quel concetto.
E ci sono vari modi per rappresentare i numeri.

Un numerale è scritto usando CIFRE

sistema di numerazione decimale

CIFRE decimali

0	1	2	3	4
5	6	7	8	9

millenovecentosessantanove → 1969

millenovecentosessantanove → MDCCCCLXIX

M	D	C	L
X	V	I	

CIFRE romane

sistema di numerazione Romano (*)

(*) MCMLXIX

CIFRE maya

Numero	Forma verticale	Forma orizzontale	Numero	Forma verticale	Forma orizzontale
0			10		
1			11		
2			12		
3			13		
4			14		
5			15		
6			16		
7			17		
8			18		
9			19		

sistema di numerazione Maya

millenovecentosessantanove

→

☺ Vedi Esercizi

Sistemi di numerazione posizionali

Quello che usiamo normalmente (sistema decimale) è un sistema posizionale in base 10.

$$1969 = 1\times10^3 + 9\times10^2 + 6\times10^1 + 9\times10^0$$

"posizionale" = ogni cifra ha un peso, che è una potenza della base; la potenza dipende dalla posizione della cifra nel numerale

numero espresso in base ***b***, con un numerale di n cifre (in cui ogni cifra è una tra le ***b*** possibili): $(c_{n-1}c_{n-2}\ldots c_2c_1c_0)_b$

$(1\ 9\ 6\ 9)_{10}$

corrisponde al valore (nel sistema decimale)

$$\sum_{k=n-1}^{0} c_k \cdot b^k$$

$$(c_{n-1}c_{n-2}\ldots c_2c_1c_0)_b = c_{n-1}\cdot b^{n-1} + c_{n-2}\cdot b^{n-2} + \ldots + c_1\cdot b^1 + c_0\cdot b^0$$

Quali sono i numerali qui?
☺ Vedi Esercizi

A partire dal NUMERALE in base 2: Calcolo del NUMERALE in base 10

Si usa il **POLINOMIO** che abbiamo visto prima

n cifre in base b ➔ numerale in base 10

$(c_{n-1}c_{n-2}\ldots c_2c_1c_0)_b = \sum_{k=n-1}^{0} c_k \cdot b^k$ (NB operazioni in base 10)

1011 = ☺
10111 = ☺
0001011 = ☺

Vedi Esercizi

A partire dal NUMERALE in base 10: Calcolo del NUMERALE in base b

dato un numero decimale, ci piacerebbe trovare le cifre del numerale in base b, facendo i conti in base di partenza (decimale)
Cominciamo facile: con divisioni successive posso isolare le cifre del numeral decimale, una per una, dalla meno significatva alla piu` significativa

$$(c_{n-1}c_{n-2}\ldots c_2c_1c_0)_{10}$$

$$numero = \frac{numero}{10} \cdot 10 + RESTO \qquad RESTO = c_0$$

$$\frac{numero}{10} = \frac{numero}{10} \Big/ 10 \cdot 10 + RESTO \qquad RESTO = c_1$$

e cosi` via

Ex. $1969 = (1969/10)\cdot 10 + 9$
$196 = (196/10)\cdot 10 + 6$
$19 = (19/10)\cdot 10 + 9$
$1 = (1/10)\cdot 10 + 1$
0 ... stop

Ehm, qui "numero" si intende "numerale", dato che lavoriamo sempre con le cifre.

Ora definiamo il metodo delle divisioni successive per ottenere la il numero binario corrispondente ad uno decimale, ma il principio e` generale – Proma di proseguire vedi Approfondimenti – e poi rivedili dopo ...

Vedi Approfondimenti

Metodo dele DIVISIONI SUCCESSIVE (Da NUMERALE in base 10 a NUMERALE in base 2

Applicando lo stesso metodo visto due slide fa, si possono calcolare le cifre in del numerale in base 2, come RESTI delle divisioni successive.

Lo facciamo con un esempio $(1969)_{10} \rightarrow (c_{n-1}c_{n-2}\ldots c_2c_1c_0)_2$

(1969/2)	=	984	resto 1	$c_0 = 1$
(984/2)	=	492	resto 0	$c_1 = 0$
(492/2)	=	246	resto 0	$c_2 = 0$
(246/2)	=	123	resto 0	$c_3 = 0$
(123/2)	=	61	resto 1	$c_4 = 1$
(61/2)	=	30	resto 1	$c_5 = 1$
(30/2)	=	*continua tu* ☺		

quando ci si ferma? qual e` il numerale binario finale?

$(1969)_{10} = (c_{n-1}c_{n-2} \ldots 110001)_2$

Metodo delle DIVISIONI SUCCESSIVE (Da NUMERALE in base 10 a NUMERALE in base 2

Applicando lo stesso metodo visto due slide fa, si possono calcolare le cifre in del numerale in base 2, come RESTI delle divisioni successive.

Lo facciamo con un esempio $(1969)_{10} \rightarrow (c_{n-1}c_{n-2}\ldots c_2c_1c_0)_2$

(1969/2)	=	984	resto 1	$c_0 = 1$
(984/2)	=	492	resto 0	$c_1 = 0$
(492/2)	=	246	resto 0	$c_2 = 0$
(246/2)	=	123	resto 0	$c_3 = 0$
(123/2)	=	61	resto 1	$c_4 = 1$
(61/2)	=	30	resto 1	$c_5 = 1$
(30/2)	=	15	resto 0	$c_6 = 0$
(15/2)	=	7	resto 1	$c_7 = 1$
(7/2)	=	3	resto 1	$c_8 = 1$
(3/2)	=	1	resto 1	$c_9 = 1$
(1/2)	=	0	resto 1	$c_{10} = 1$

... stop

Se, inopinatamente, non lo hai fatto prima ...
Vedi per bene gli Approfondimenti

$(1969)_{10} = (11110110001)_2$

I NUMERALI BINARI possono esprimere anche NUMERI con parte FRAZIONARIA ...

Quello che usiamo di solito (sistema decimale) è un sistema posizionale in base 10.

$$756.25 = 7x10^2 + 5x10 + 6 + 2x10^{-1} + 5x10^{-2}$$

"posizionale" = ogni cifra ha un peso, che è una potenza della base; la potenza dipende dalla posizione della cifra nel numerale

Analogamente si possono scrivere numerali binari che esprimono numeri con parte frazionaria

$$756.25 = 1011110100.01 =$$
$$= 1x2^9 + 1x2^7 + 1x2^6 + 1x2^5 + 1x2^4 + 1x2^2 + 0x2^{-1} + 1x2^{-2}$$

$$141.43359375 = 10001101.01101111 = 1x2^7 + 1x2^3 + 1x2^2 + 1x2^0 + 1x2^{-2} + 1x2^{-3} + 0x2^{-4} + 1x2^{-5} + 1x2^{-6} + 1x2^{-7} + 1x2^{-8}$$

anche numeri con parte frazionaria ...

numero espresso in base *b*, con n cifre per la parte intera e m cifre per la parte frazionaria (cifre della base b):

$$(c_{n-1}c_{n-2}\ldots c_2c_1c_0.c_{-1}c_{-2}\ldots c_{-(m-1)}c_{-m})_b$$

corrisponde al valore (nel sistema decimale)

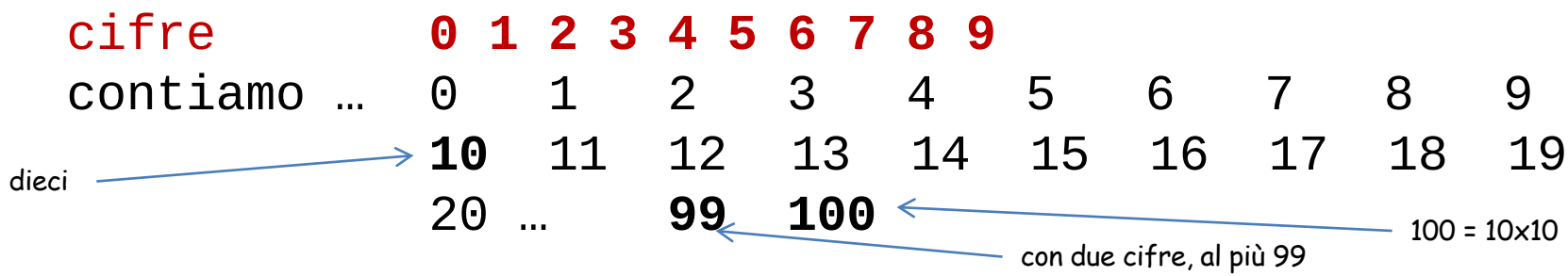
$$(c_{n-1}c_{n-2}\ldots c_2c_1c_0.c_{-1}\ldots\ldots c_{-m})_b = \sum_{k=n-1}^{-m} c_k \cdot b^k$$

$$c_{n-1} \cdot b^{n-1} + c_{n-2} \cdot b^{n-2} + \ldots + c_1 \cdot b^1 + c_0 \cdot b^0 +$$

$$+ c_{-1} \cdot b^{-1} + c_{-2} \cdot b^{-2} + \ldots + c_{-m} \cdot b^{-m}$$

basi

base 10



Con 5 cifre [0, 99999]

Con 7 cifre [0, 9999999]

NB con n cifre si rappresentano b^n numeri naturali in base b da 0 a $(b^n - 1)$ … $[0, (b^n - 1)]$

INTERVALLO DI RAPPRESENTABILITà

basi

NB con n cifre si rappresentano b^n numeri naturali in base b da 0 a $(b^n - 1)$ … [0, $(b^n - 1)$]

base 10

cifre	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
contiamo …	0	1	2	3	4	5	6	7	8	9
dieci →	**10**	11	12	13	14	15	16	17	18	19
	20 …		**99**	**100** ← 100 = 10x10						

con due cifre, al più 99

base 8

cifre	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
contiamo …	0	1	2	3	… ☺			
otto →	**?**						quindici → **?**	
sedici →	20 …					? ← venti		
	…	**77**	100	101	102 …			
		$(63)_{10}$	$(64 = 8x8)_{10}$					

$(21)_8 = 2*8^1 + 1*8^0 = (17)_{10}$ diciassette

basi

NB con n cifre si rappresentano b^n numeri naturali in base b da 0 a $(b^n - 1)$ … [0, $(b^n - 1)$]

base 10

cifre	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
contiamo …	0	1	2	3	4	5	6	7	8	9
	10	11	12	13	14	15	16	17	18	19
	20 …		**99**	**100**						

dieci

100 = 10x10

con due cifre, al più 99

base 8

cifre	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
contiamo …	0	1	2	3	4	5	6	7
	10	11	12	13	14	15	16	17
	20	21	22	23	24	…		
	…	**77**	100	101	102	…		
		$(63)_{10}$	$(64 = 8x8)_{10}$					

otto

sedici

quindici

venti

$(21)_8 = 2*8^1 + 1*8^0 = (17)_{10}$ diciassette

$(47)_8$ = ☺ trentanove

$(101)_8$ = ☺

basi

NB con n cifre si rappresentano b^n numeri naturali in base b da 0 a $(b^n - 1)$ … $[0, (b^n - 1)]$

base 10

cifre	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
contiamo …	0	1	2	3	4	5	6	7	8	9
dieci →	**10**	11	12	13	14	15	16	17	18	19
	20 …		**99**	**100**						

100 = 10x10

con due cifre, al più 99

base 8

cifre	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
contiamo …	0	1	2	3	4	5	6	7
otto →	**10**	11	12	13	14	15	16	17
sedici →	20	21	22	23	24	…		
	…	**77**	100	101	102 …			
		$(63)_{10}$	$(64 = 8x8)_{10}$					

quindici

venti

$(21)_8 = 2*8^1 + 1*8^0 = (17)_{10}$ diciassette

$(47)_8 = 4*8^1 + 7*8^0 = (39)_{10}$ trentanove

$(101)_8 = 1*8^2 + 0*8^1 + 1*8^0 = (65)_{10}$ sessantacinque

basi

NB con n cifre si rappresentano b^n numeri naturali in base b da 0 a $(b^n - 1)$ … $[0, (b^n - 1)]$

base 8 (Sistema ottale – o anche *octal*, *oct*)

cifre	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
contiamo …	0	1	2	3	4	5	6	7
	10	11	12	13	14	15	16	17
	20	21	22	23	24	…		
	…	**77**	100	101				

$(31)_8 = (?)_{10}$

Vedi Approfondimenti

basi

NB con n cifre si rappresentano b^n numeri naturali in base b da 0 a $(b^n - 1)$ … [0, $(b^n - 1)$]

base 8

cifre	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
contiamo …	0	1	2	3	4	5	6	7
	10	11	12	13	14	15	16	17
	20	21	22	23	24	…		
	…	**77**	100	101				

Con 5 cifre [0, 77777] cioè da 0 a trentaduemilasettecentosessantaset

Con 7 cifre [0, 7777777] cioè da 0 a $(2.097.151)_{10}$

☺ check a casa …

basi

NB con n cifre si rappresentano b^n numeri naturali in base b da 0 a $(b^n - 1)$ … $[0, (b^n - 1)]$

base 10

cifre	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
contiamo …	0	1	2	3	4	5	6	7	8	9
	10	11	12	13	14	15	16	17	18	19
	20 …		**99**	**100**						

dieci

100 = 10x10

con due cifre, al più 99

base 8

cifre	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
contiamo …	0	1	2	3	4	5	6	7
	10	11	12	13	14	15	16	17
	20	21	22	23	24	…		
	…	**77**	100	101	102	…		

quindici

otto

venti

sedici

$(63)_{10}$ $(64 = 8x8)_{10}$

base 2

cifre	**0**	**1**
contiamo …	0	1
	10	11
$(4=2x2)_{10}$	**100**	101 110 111
	1000	1001 1010 1011 $(12)_{10}$ 1100 1101 1110 $(15)_{10}$ 1111
	…	

con due cifre, si rappresentano 4 numeri in totale (num max: 3)

con tre cifre, max numero rappresentato: 7

base 2

base 2

```
        cifre          0 1
        contiamo …     0   1
                       10  11
(4=2x2)10              100 101 110 111
                       1000 1001 1010 1011 1100 1101 1110 1111
                       …
```

(12)10 (15)10

con due cifre, si rappresentano 4 numeri in totale (max 3)

con tre cifre, max numero rappresentato: 7

			INTERVALLO DI RAPPRESENTABILITA`
Con 2 cifre	[0, 11]	da 0 a tre	[0, (2^2-1)]
Con 5 cifre	[0, 11111]	da 0 a (2^5-1)	[0, 31] decimale
Con 7 cifre	[0, 1111111]	da 0 a (2^7-1)	[0, 127] decimale

NB con n cifre si rappresentano b^n numeri naturali in base b da 0 a (b^n -1) … [0, (b^n -1)]

INTERVALLO DI RAPPRESENTABILITA`

base 2

numerale binario con al piu` n cifre:
INTERVALLO DI RAPPRESENTABILITA`
e` la sequenza di numeri naturali $[0, (2^n - 1)]$

```
cifre          0 1
contiamo …     0   1
               10  11
               100 101 110 111
               1000 1001 1010 1011 1100 1101 1110 1111
```

con 32 bit si rappresentano i numeri naturali in $[0, (2^{32} - 1)]$

con 5 bit si rappresentano i numeri naturali in $[0, 31]$

Quindi se lo spazio per i numerali è limitato, altrettanto limitato è l'insieme dei numeri rappresentati dai numerali

Quanti bit servono per rappresentare

64 256 796 1025 2047

?

Prova, almeno i primi 3 e poi **Vedi Approfondimenti**

operazioni in binario (+)

Gli algoritmi sono i medesimi nei vari sistemi posizionali … ma
1 + 1 = 0 con resto 1

5 bit

SOMMA

```
        1
  0 0 1 0 1   (5)10
  0 1 0 0 1   (9)10
  ---------
0 1 1 1 0
```
$(14)_{10}$

NB

```
    1 1
  1 0 1 0 1   (21)10
  0 1 1 1 0
  ---------
1 0 0 0 1 1
overflow
```

1 + 1 = 0 con riporto di 1
sulla cifra vicina a sinistra

operazioni in binario (-)

Gli algoritmi sono i medesimi nei vari sistemi posizionali … ma

1 + 1 = 0 con resto 1

0 – 1 = 1 con prestito

5 bit

SOMMA

```
      1
  0 0 1 0 1   (5)10
  0 1 0 0 1   (9)10
  ---------
  0 1 1 1 0   (14)10
```

NB

```
    1 1
  1 0 1 0 1   (21)10
  0 1 1 1 0
  ---------
1 0 0 0 1 1
overflow
```

5 bit

SOTTRAZIONE

```
     10
  0 1 0 0 1
  0 0 1 0 1
  ---------
  0 0 1 0 0   (4)10
```

0 - 1 = 1 con "prestito" di 1 da sinistra

(il "prestito" funziona cosi`: invece di fare 0-1, facciano 10-1, e 10-1 e` 1 (due meno uno ...)

operazioni in binario

Gli algoritmi sono i medesimi nei vari sistemi posizionali …

5 bit
moltiplicazione

```
            0 1 0 0 1
            0 0 1 0 1
            ---------
            0 1 0 0 1
          0 0 0 0 0
        0 1 0 0 1
      0 0 0 0 0
    0 0 0 0 0
    -----------------
    0 0 0 1 0 1 1 0 1
```

operazioni in binario

Gli algoritmi sono i medesimi nei vari sistemi posizionali …

5 bit
moltiplicazione

```
                0 1 0 0 1
                0 0 1 0 1
                ---------
                0 1 0 0 1
              0 0 0 0 0
            0 1 0 0 1
          0 0 0 0 0
        0 0 0 0 0
        -----------------
        0 0 0 1 0 1 1 0 1
```

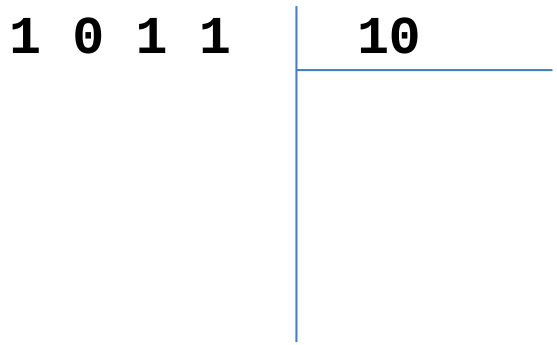
= … ☺ **calcolo del numero decimale a partire dalla rappresentazione binaria**

Prova e poi **Vedi Esercizi**

operazioni in binario

Gli algoritmi sono i medesimi nei vari sistemi posizionali …

5 bit
divisione



operazioni in binario

5 bit
divisione

```
1 0 1 1 | 10
        |
        |
```

10 diviso 10 fa 1 con resto -

```
1 0 1 1 | 10
  -     | 1
        |
```

operazioni in binario

Gli algoritmi sono i medesimi nei vari sistemi posizionali …

5 bit
divisione

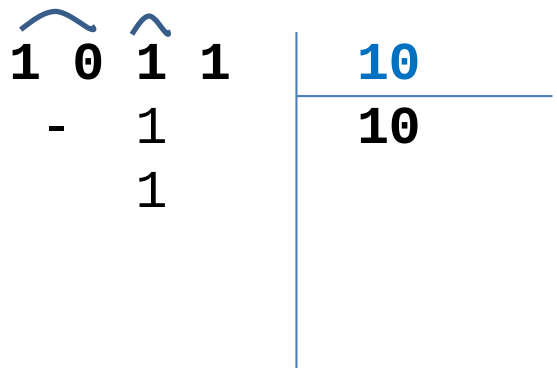
```
1 0 1 1  |  10
         |
         |
```

```
1 0 1 1  |  10
-        |  1
         |
         |
```

```
1 0 1 1  |  10
- 1      |  1
         |
         |
```

operazioni in binario

5 bit
divisione



1 diviso **10** fa 0 col resto di 1

operazioni in binario

5 bit
divisione

```
1 0 1 1 | 10
  -   1 | 10
      1 |
```

```
1 0 1 1 | 10
  -   1 | 10
    1 1 |
```

11 diviso **10** fa 1 col resto di 1
(oh, e` tre diviso due ... ok?)

```
1 0 1 1 | 10
  -   1 | 101
    1 1 |
      1 |
```

operazioni in binario

ma sono esatti questi calcoli? Il quoziente che ci iene e` giusto? Provare a verificare e poi **Vedi Approfondimenti**

5 bit
divisione

```
1 0 1 1 | 10
  -   1 |---
    1 1 | 101
      1 |
```

```
1 0 1 1 | 10
  -   1 |---
    1 1 | 101.
    1 0 |
```

poi si abbassa "niente" cioe` zero
e si prosegue con **10**
che diviso per **10** fa 1, ma qui, avendo abbassato "niente", cioe` zero, siamo nella zona frazionaria del quoziente, quindi **.1**

```
1 0 1 1 | 10
  -   1 |-----
    1 1 | 101.1
    1 0 |
      - |
```

Conversione $(n)_{10}$ --> $(\ldots)_2$

Il procedimento è quello per divisioni successive (divisioni per la base 2). Ogni divisione fornisce un resto, che è la cifra nella rappresentazione di arrivo (binaria)

$(796)_{10} = (?)_2$

796/2 =	398	Resto 0
/2	199	0
/2	99	1
/2	49	1
/2	24	1
/2	12	0
/2	6	0
/2	3	0
/2	1	1
/2	0	1

ordine delle cifre

$(796)_{10} = (1100011100)_2$

peso 512 peso 256 16 8 4

Conversione $(n)_{10}$ --> $(.)_2$ per parte frazionaria

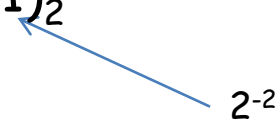
Il procedimento è per moltiplicazioni successive (per la base 2).
Se il risultato è >=1 la cifra corrispondente nel numero di arrivo è 1; altrimenti è zero. Si prosegue con la parte frazionaria, fino a ottenere 0, o a stancarsi

$(0.25)_{10} = (?)_2$

.25x2 = .5	parte intera 0
.5x2 1	" " 1
0	fine

$(0.25)_{10} = (0.01)_2$

ordine delle cifre



$(0.23)_{10} = (?)_2$

Conversione $(n)_{10}$ --> $(.)_2$ per parte frazionaria

Il procedimento è per moltiplicazioni successive (per la base 2).
Se il risultato è >=1 la cifra corrispondente nel numero di arrivo è 1; altrimenti è zero. Si prosegue con la parte frazionaria, fino a ottenere 0, o a verificare l'esistenza di un periodo, **o a stancarsi**(*)

$(0.23)_{10} = (?)_2$

.23x2 =	.46	0
.46x2	.92	0
.92x2	1.84	1
.84x2	1.68	1

☺

0.92X2 è 1.84:
la cifra 1 viene usata per continuare a costruire il numerale, mentre la parte frazionaria (0,84) viene usata per continuare il processo di moltiplicazioni successive

ordine delle cifre

Conversione $(n)_{10}$ --> $(.)_2$ per parte frazionaria

Il procedimento è per moltiplicazioni successive (per la base 2).
Se il risultato è >=1 la cifra corrispondente nel numero di arrivo è 1; altrimenti è zero. Si prosegue con la parte frazionaria, fino a ottenere 0, o a verificare l'esistenza di un period, **o a stancarsi**(*)

$(0.23)_{10} = (?)_2$

.23x2 =	.46	0
.46x2	.92	0
.92x2	1.84	1
.84x2	1.68	1
.68x2	1.36	1
.36x2	.72	0
.72x2	1.44	1
.44x2	.88	0
.88x2	1.76	1
.76x2	1.52	1
.52x2	1.04	1
uff	fine(*) ...	

0.92X2 è 1.84: la cifra 1 viene usata per continuare a costruire il numerale, mentre la parte frazionaria (0,84) viene usata per continuare il processo di moltiplicazioni successive

ordine delle cifre

$(0.23)_{10} = (0.00111010111)_2$ rappresentazione esatta in decimale ma approssimata in binario

(*) nel senso che si è raggiunta la precisione voluta e non si vogliono calcolare altre cifre

Conversione $(n)_{10}$ --> $(.)_2$ per parte frazionaria

Il procedimento è per moltiplicazioni successive (per la base 2).
Se il risultato è >=1 la cifra corrispondente nel numero di arrivo è 1; altrimenti è zero. Si prosegue con la parte frazionaria, fino a ottenere 0, **o a stancarsi**

$(0.23)_{10} = (?)_2$

.23x2 =	.46	0
.46x2	.92	0
[illegible]	[illegible]	[illegible]
.52x2	1.04	1
uff	fine	

delle cifre

QUINDI

$(796.25)_{10} = (1100011100.01)_2$

$(796.23)_{10} = (1100011100.00111010111)_2$ circa

$(0.25)_{10} = (0.00111010111)_2$ rappresentazione esatta in decimale ma approssimata in binario

Conversione $(n)_2$ <--> $(.)_8$

Fantastico! Le cifre binarie corrispondono a quelle ottali, just prese a gruppi di 3.
E otto è due alla terza!

Es.

$(796)_{10} = (1\ 1\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 0)_2$

gruppi di tre da destra verso sinistra; se mancano cifre si aggiungono zeri a sinistra

$= (001\ 100\ 011\ 100)_2 = (1\ 4\ 3\ 4)_8$

base 16

NB con n cifre si rappresentano b^n numeri naturali in base b da 0 a $(b^n - 1)$ … $[0, (b^n - 1)]$

base 16

cifre		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
contiamo …	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
sedici	20	…		**F1**	**F2**	**…**	**FF**									

con due cifre, al più $15 \times 16 + 15 = 16^2 - 1$

rispetto a 10, 8, 2 … si hanno numerali di solito più corti, dato che ci sono più cifre a disposizione

$(15)_{10} = (1111)_2 = F_{16}$

base 16

NB con n cifre si rappresentano b^n numeri naturali in base b da 0 a $(b^n - 1)$ … $[0, (b^n - 1)]$

base 16

cifre		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
contiamo …	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
sedici	20	…		**F1**	**F2**	…	**FF**									

con due cifre, al più $15 \times 16 + 15 = 16^2 - 1$

rispetto a 10, 8, 2 … si hanno numerali di solito più corti, dato che ci sono più cifre a disposizione

$(15)_{10} = (\ 1111\ )_2 = F_{16}$

Conversione $(\ n\ )_2$ <--> $(\ .\ )_{16}$

Fantastico! … gruppi di 4. E sedici è due alla quarta!

Es.

$(796)_{10} = (\ 1\ 1\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ )_2$

$= (\ 0011\ 0001\ 1100\ )_2 = (\ 3\ 1\ C\ )_{16}$

Esercizi – numeri binari

Quali numeri naturali sono rappresentati dai seguenti numerali binari?

100010 e qual è il più grande numero rappresentabile con questo numero di cifre?

1101 e qual è il più grande numero rappresentabile con questo numero di cifre?

111101 e qual è il più grande numero rappresentabile con questo numero di cifre?

10101001 e qual è il più grande numero rappresentabile con questo numero di cifre?

10010110000000 e qual è il più grande numero rappresentabile con questo numero di cifre?

1010000000101000 e qual è il più grande numero rappresentabile con questo numero di cifre?

101001101 e qual è il più grande numero rappresentabile con questo numero di cifre?

Per questi numeri, poi, scrivere la rappresentazione ottale e quella esadecimale.

Per i seguenti numeri decimali, scrivere la rappresentazione in binario puro.

0.703125 125.1875 132.6875 912.75 1216.1875 5509.8125

Per i seguenti numeri naturali, rappresentati in forma decimale, stabilire il minimo numero di cifre necessarie per la rappresentazione in binario puro, e poi scrivere la rappresentazione medesima.

151 212 918 96000 31718 55 181 987987 128128128

Siete in
Tecniche della Programmazione, lez. 12

Rappresentazione dei numeri

- numeri interi: complemento a 2

Remember

con N cifre binarie (N bit) si possono scrivere i numeri naturali da 0 a 2^N-1

intervallo di rappresentabilita` per i numeri naturali espressi in forma binaria, con N cifre (N bit):

$$[0, 2^N-1]$$

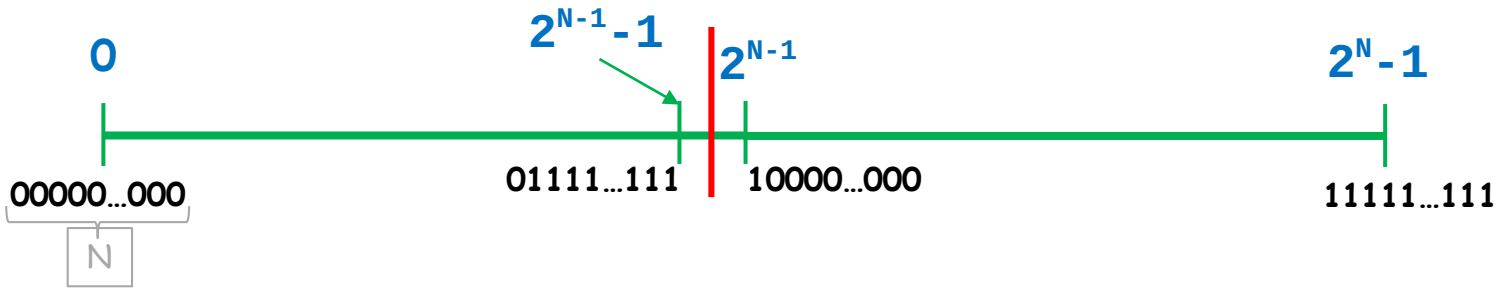
Esempio
sia N==4
Quali sono i numerali binari corrispondenti?
E a quali numeri naturali, zero compreso, corrispondono questi numerali?

fare una tabellina e poi proseguire ;)

Remember

intervallo di rappresentabilita` per i numeri naturali espressi in forma binaria, con N cifre (N bit):

$$[0, 2^N-1]$$



ESEMPIO : Numerali con 4 bit

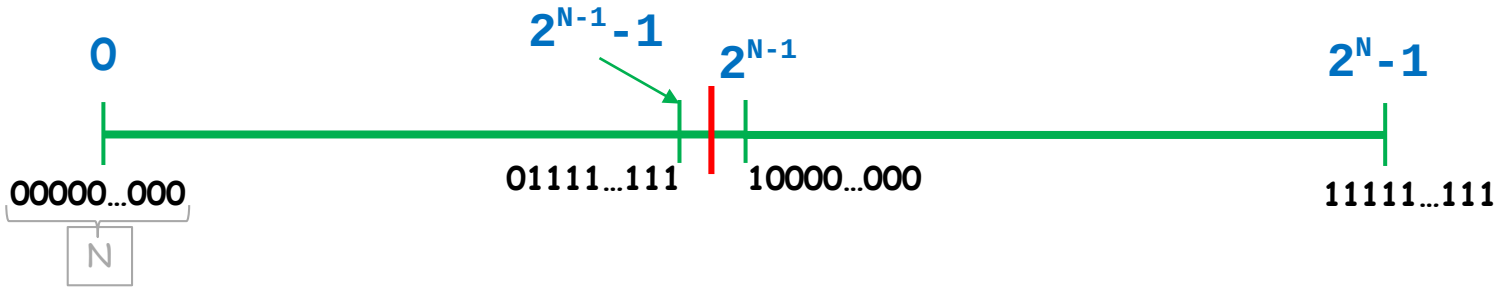
0 0 0 0	0
0 0 0 1	1
0 0 1 0	2
0 0 1 1	3
0 1 0 0	4
0 1 0 1	5
0 1 1 0	6
0 1 1 1	7
1 0 0 0	8
1 0 0 1	9
1 0 1 0	10
1 0 1 1	11
1 1 0 0	12
1 1 0 1	13
1 1 1 0	14
1 1 1 1	15

con N = 4, abbiamo 2^4 numerali, cioè 16 da **0000** a **1111** che possono rappresentare i numeri naturali da 0 a 2^4-1 cioè l'intervallo [0, 15]

☺ **replicare questa slide, intera, per N=5** $[0, 2^5-1] = [0, 32-1] = [0, 31]$

Remember

con N cifre binarie (N bit) si possono scrivere i numeri naturali da 0 a 2^N-1



Se N = 32, allora abbiamo 2^{32} numerali,

da 00000000000000000000000000000000 a 11111111111111111111111111111111

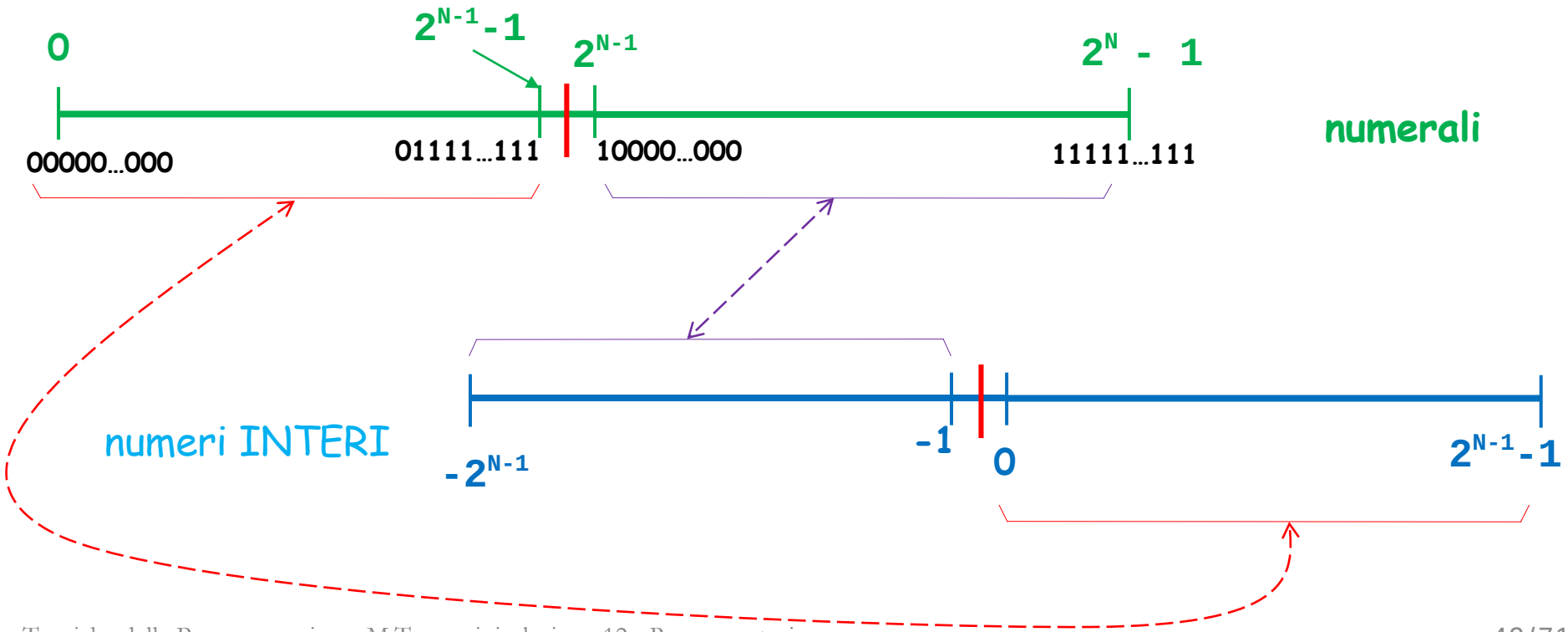
che possono rappresentare i numeri naturali
da 0 a $2^{32}-1$
cioè l'intervallo $[0, 2^{32}-1]$

Rappresentazione dei numeri interi in *Complemento a 2*

Solo che non si può vivere solo con i naturali …
Vorremmo rappresentare, **con i numerali disponibili**, quanti più numeri INTERI possibile, un pò positivi e un pò negativi (diciamo metà e metà)

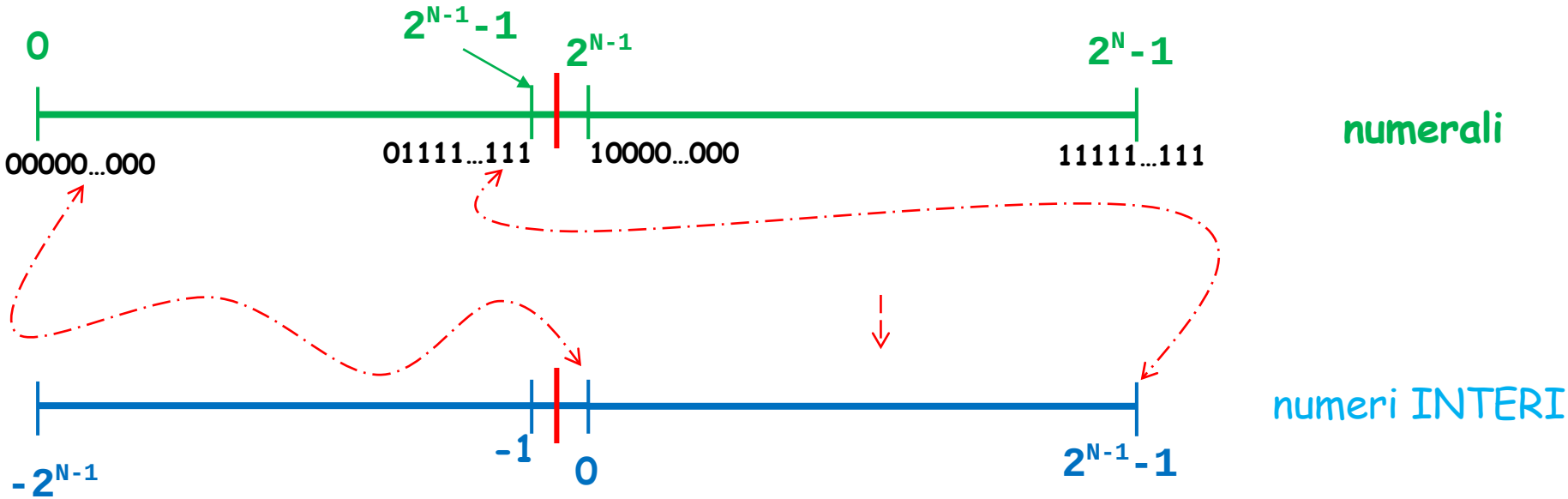
Con N cifre binarie (N bit) i **numerali disponibili** sono 2^N :

- **ne usiamo metà per i numeri negativi (sono 2^{N-1} numerali)**
- **usiamo gli altri 2^{N-1} numerali per i numeri positivi e per lo zero**



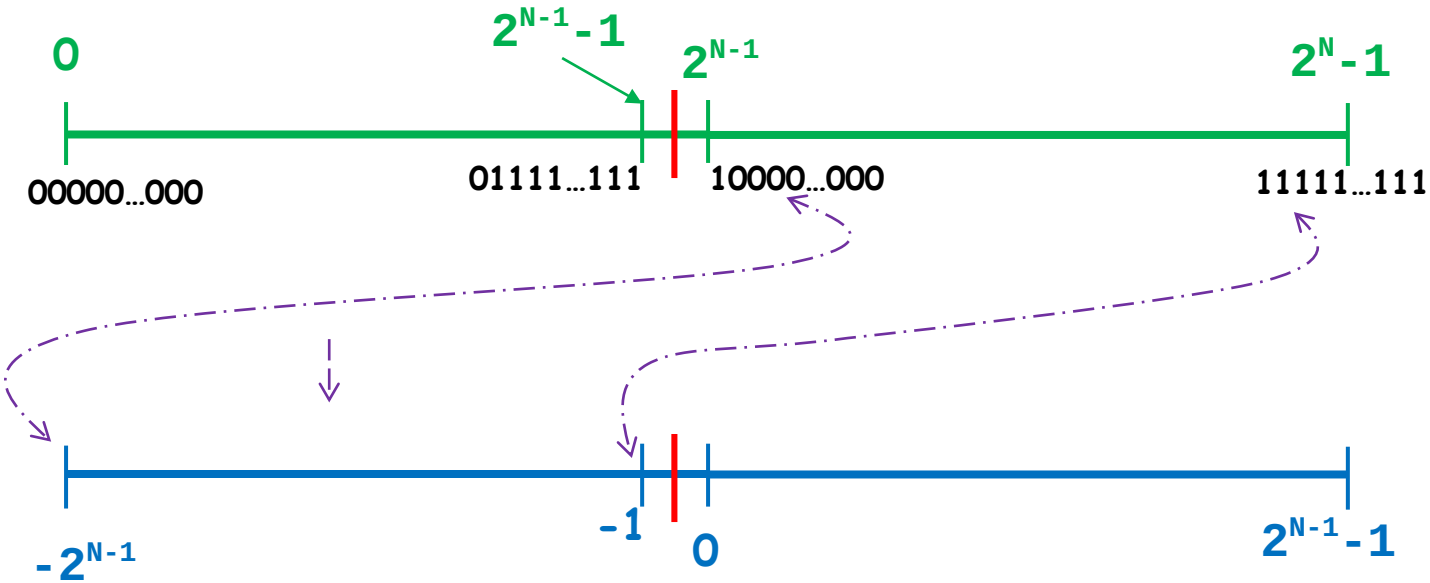
Rappresentazione in *Complemento a 2 (Tipo int)*

dettaglio su alcuni numeri / numerali estremi ...



Rappresentazione in *Complemento a 2 (Tipo int)*

dettaglio su alcuni numeri / numerali estremi ...

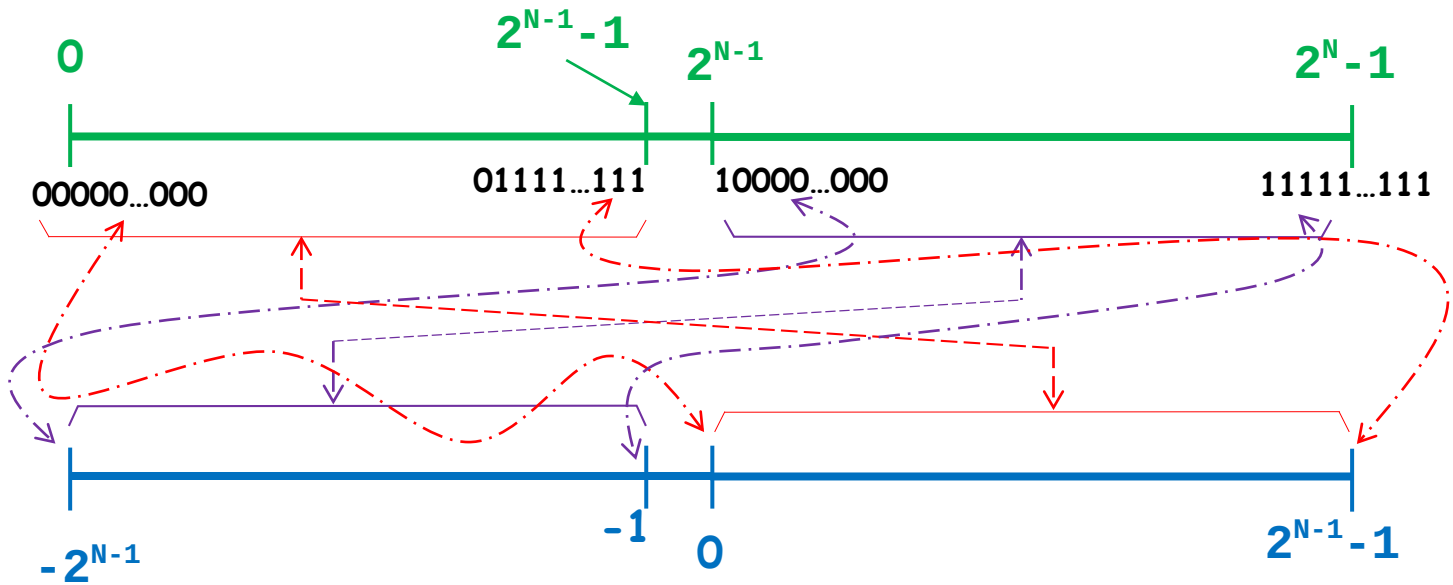


Rappresentazione in *Complemento a 2 (Tipo int)*

Solo che non si può vivere solo con i naturali …
Vorremmo rappresentare, con i numerali disponibili, quanti più numeri INTERI possibile, un pò positivi e un pò negativi (diciamo metà e metà)

Con N cifre binarie (N bit) i **numerali disponibili** sono 2^N :

- **ne usiamo metà per i numeri negativi (sono 2^{N-1} numerali)**
- **usiamo gli altri 2^{N-1} numerali per i numeri positivi e per lo zero**



Con N = 32, i 2^{32} numerali rappresentano i numeri interi nell'intervallo $[-2^{31}, 2^{31}-1]$ (intervallo di rappresentabilità)

Rappresentazione in *Complemento a 2 (Tipo int)*

Solo che non si può vivere solo con i naturali …
Vorremmo rappresentare, con i numerali disponibili, quanti più numeri INTERI possibile, un pò positivi e un pò negativi (diciamo metà e metà)

Con N cifre binarie (N bit) i **numerali disponibili** sono 2^N :

- **ne usiamo metà per i numeri negativi (sono 2^{N-1} numerali)**
- **usiamo gli altri 2^{N-1} numerali per i numeri positivi e per lo zero**

Con N = 32, i 2^{32} numerali rappresentano i numeri interi nell'intervallo $[-2^{31}, 2^{31}-1]$ (intervallo di rappresentabilità)

NB
Se N e` il numero di bit usati per i numerali, l'intervallo di rappresentabilita` dei numeri interi in complemento a 2 si esprime come

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

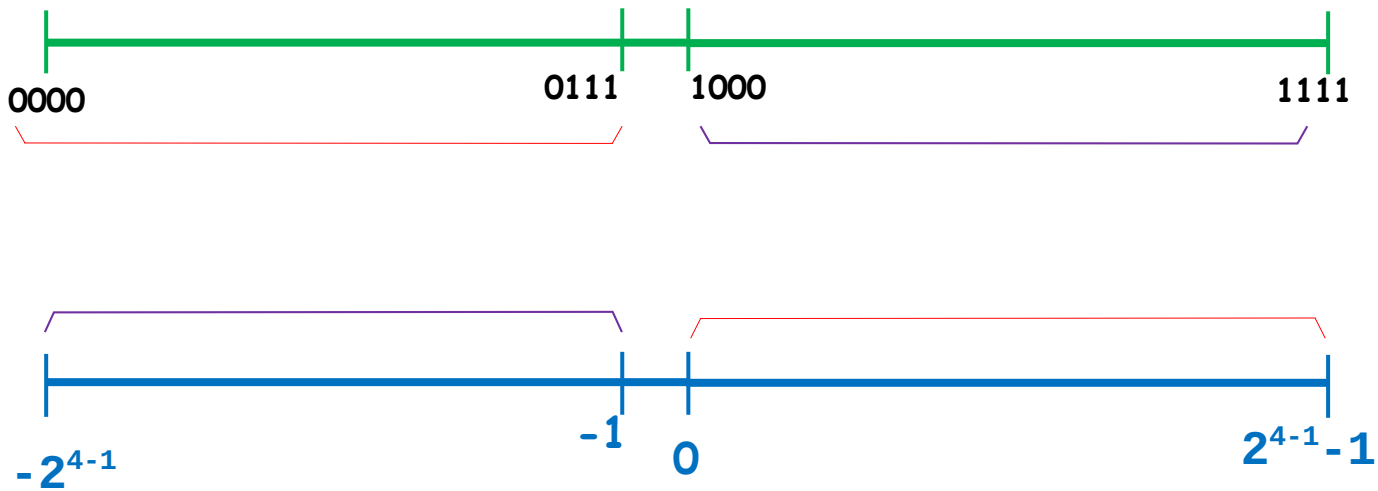
Tipo int (rappresentazione in Complemento a 2)

Solo che non si può vivere solo con i naturali …
Vorremmo rappresentare, con i numerali disponibili, quanti più numeri INTERI possibile, un pò positivi e un pò negativi (diciamo metà e metà)

Con N cifre binarie (N bit) i **numerali disponibili** sono 2^N :

- **ne usiamo metà per i numeri negativi (sono 2^{N-1} numerali)**
- **usiamo gli altri 2^{N-1} numerali per i numeri positivi e per lo zero**

Con N = 4, i 24 numerali rappresentano i numeri interi nell'intervallo $[-2^3 , 2^3-1] = [-8, 7]$



ESEMPIO: Numerali con 4 bit

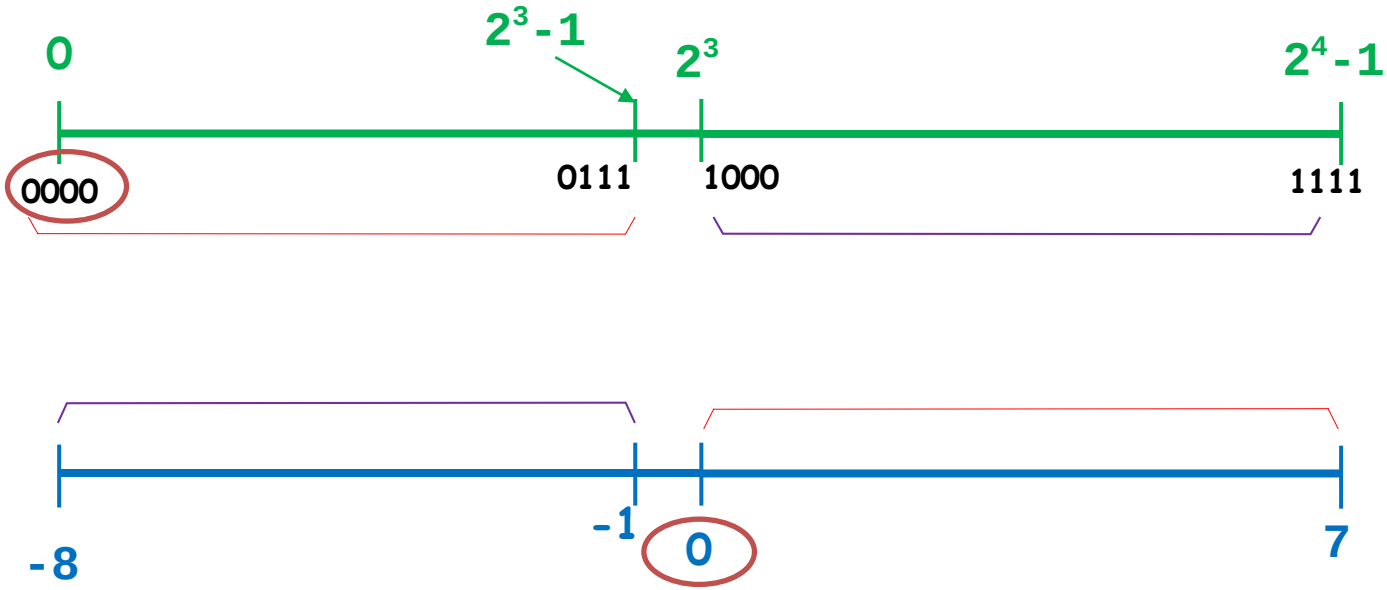
	naturali	interi
0 0 0 0	0	0
0 0 0 1	1	1
0 0 1 0	2	2
0 0 1 1	3	3
0 1 0 0	4	4
0 1 0 1	5	5
0 1 1 0	6	6
0 1 1 1	7	7
1 0 0 0	8	-8
1 0 0 1	9	-7
1 0 1 0	10	-6
1 0 1 1	11	-5
1 1 0 0	12	-4
1 1 0 1	13	-3
1 1 1 0	14	-2
1 1 1 1	15	-1

Tipo int (rappresentazione in Complemento a 2)

Solo che non si può vivere solo con i naturali …
Vorremmo rappresentare, con i numerali disponibili, quanti più numeri INTERI possibile, un pò positivi e un pò negativi (diciamo metà e metà)

Con N cifre binarie (N bit) i **numerali disponibili** sono 2^N :

- **ne usiamo metà per i numeri negativi (sono 2^{N-1} numerali)**
- **usiamo gli altri 2^{N-1} numerali per i numeri positivi e per lo zero**



ESEMPIO: Numerali con 4 bit

	naturali	interi
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	8	-8
1001	9	-7
1010	10	-6
1011	11	-5
1100	12	-4
1101	13	-3
1110	14	-2
1111	15	-1

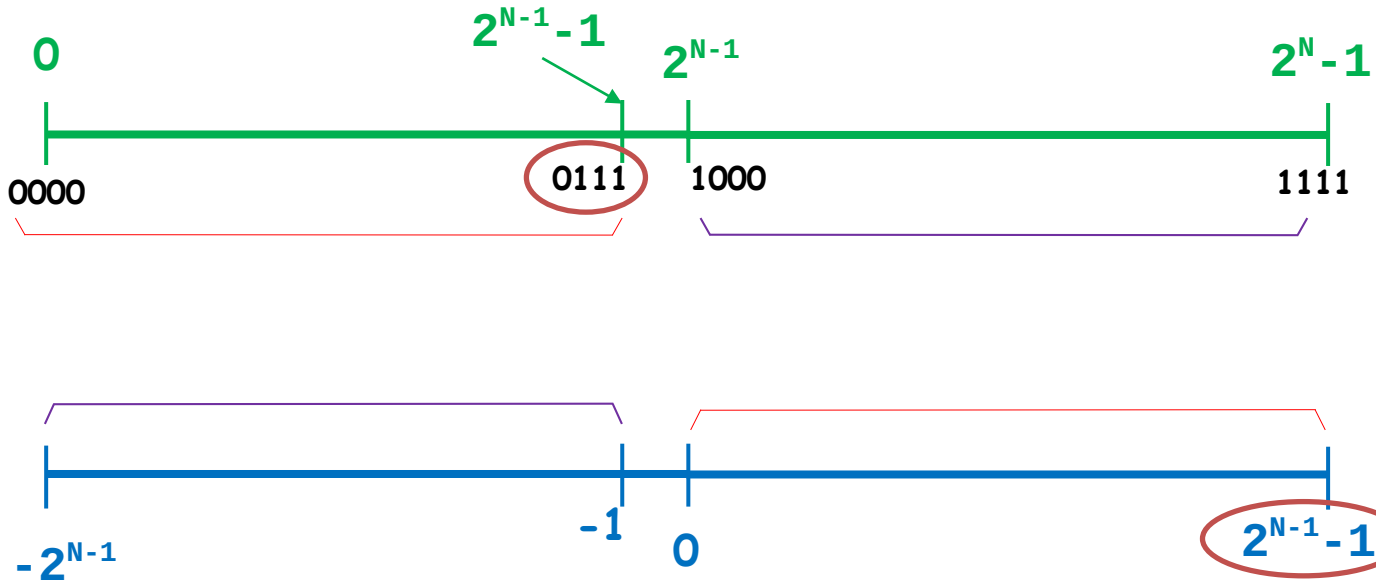
Con N = 4, i 2^4 numerali rappresentano i numeri interi nell'intervallo $[-2^3 , 2^3-1]$

Tipo int (rappresentazione in Complemento a 2)

Solo che non si può vivere solo con i naturali …
Vorremmo rappresentare, con i numerali disponibili, quanti più numeri INTERI possibile, un pò positivi e un pò negativi (diciamo metà e metà)

Con N cifre binarie (N bit) i **numerali disponibili** sono 2^N :

- **ne usiamo metà per i numeri negativi (sono 2^{N-1} numerali)**
- **usiamo gli altri 2^{N-1} numerali per i numeri positivi e per lo zero**



ESEMPIO: Numerali con 4 bit

	naturali	interi
0 0 0 0	0	0
0 0 0 1	1	1
0 0 1 0	2	2
0 0 1 1	3	3
0 1 0 0	4	4
0 1 0 1	5	5
0 1 1 0	6	6
0 1 1 1	7	7
1 0 0 0	8	-8
1 0 0 1	9	-7
1 0 1 0	10	-6
1 0 1 1	11	-5
1 1 0 0	12	-4
1 1 0 1	13	-3
1 1 1 0	14	-2
1 1 1 1	15	-1

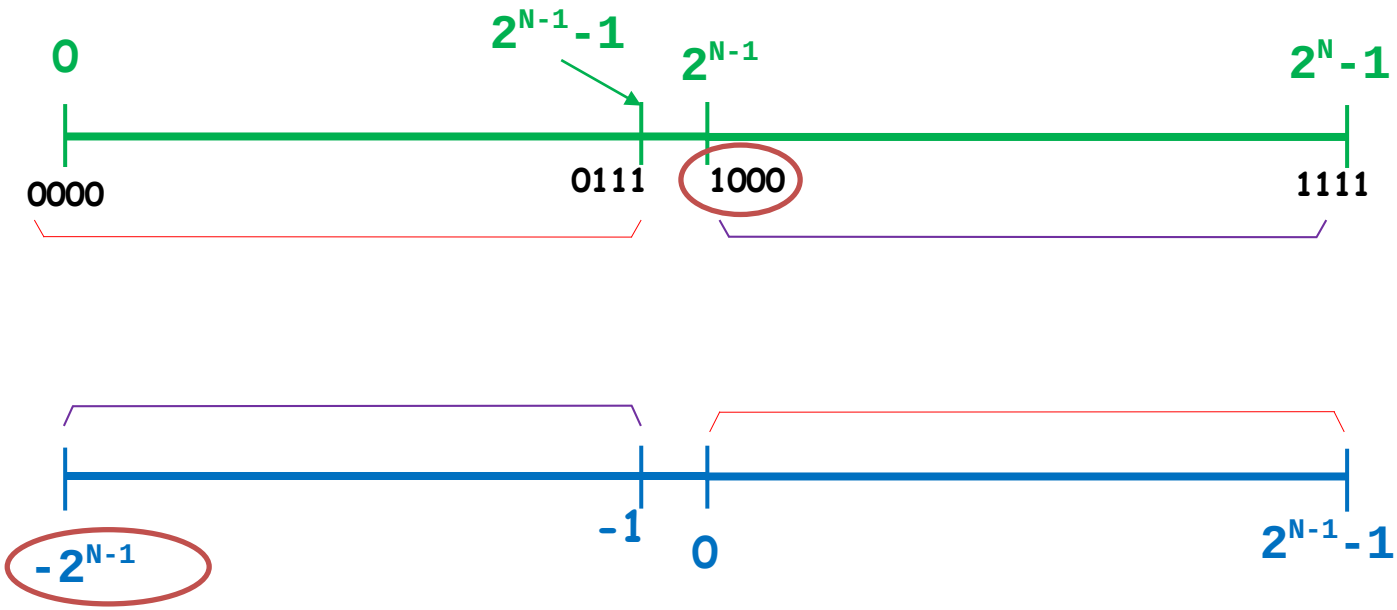
Con N = 4, i 2^4 numerali rappresentano i numeri interi nell'intervallo $[-2^3-, 2^3-1]$

Tipo int (rappresentazione in Complemento a 2)

Solo che non si può vivere solo con i naturali …
Vorremmo rappresentare, con i numerali disponibili, quanti più numeri INTERI possibile, un pò positivi e un pò negativi (diciamo metà e metà)

Con N cifre binarie (N bit) i **numerali disponibili** sono 2^N :

- **ne usiamo metà per i numeri negativi (sono 2^{N-1} numerali)**
- **usiamo gli altri 2^{N-1} numerali per i numeri positivi e per lo zero**



ESEMPIO: Numerali con 4 bit

	naturali	interi
0 0 0 0	0	0
0 0 0 1	1	1
0 0 1 0	2	2
0 0 1 1	3	3
0 1 0 0	4	4
0 1 0 1	5	5
0 1 1 0	6	6
0 1 1 1	7	7
1 0 0 0	8	-8
1 0 0 1	9	-7
1 0 1 0	10	-6
1 0 1 1	11	-5
1 1 0 0	12	-4
1 1 0 1	13	-3
1 1 1 0	14	-2
1 1 1 1	15	-1

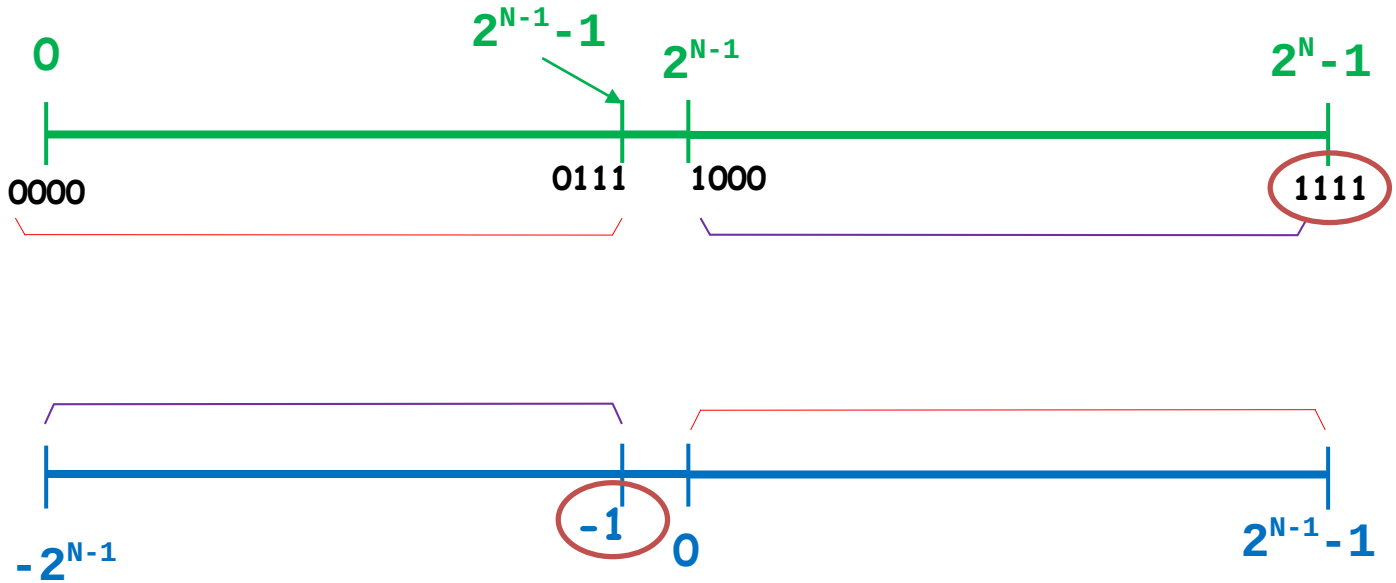
Con N = 4, i 2^4 numerali rappresentano i numeri interi nell'intervallo $[-2^3-, 2^3-1]$

Tipo int (rappresentazione in Complemento a 2)

Solo che non si può vivere solo con i naturali …
Vorremmo rappresentare, con i numerali disponibili, quanti più numeri INTERI possibile, un pò positivi e un pò negativi (diciamo metà e metà)

Con N cifre binarie (N bit) i **numerali disponibili** sono 2^N :

- **ne usiamo metà per i numeri negativi (sono 2^{N-1} numerali)**
- **usiamo gli altri 2^{N-1} numerali per i numeri positivi e per lo zero**



ESEMPIO: Numerali con 4 bit

	naturali	interi
0 0 0 0	0	0
0 0 0 1	1	1
0 0 1 0	2	2
0 0 1 1	3	3
0 1 0 0	4	4
0 1 0 1	5	5
0 1 1 0	6	6
0 1 1 1	7	7
1 0 0 0	8	-8
1 0 0 1	9	-7
1 0 1 0	10	-6
1 0 1 1	11	-5
1 1 0 0	12	-4
1 1 0 1	13	-3
1 1 1 0	14	-2
1 1 1 1	15	-1

Con N = 4, i 2^4 numerali rappresentano i numeri interi nell'intervallo $[-2^3-, 2^3-1]$

Tipo int: circolarità della rappresentazione

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	8	-8
1001	9	-7
1010	10	-6
1011	11	-5
1100	12	-4
1101	13	-3
1110	14	-2
1111	15	-1

Usando N==4 bit

1111+1 = 0000

Rappresentazione in Complemento a 2 (calcolo)

Dato un numero **n**, lo rappresentiamo in complemento a 2 tramite il numerale binario **rappr(n)**

Attention please – ci sono tre momenti in questo ragionamento …

1) Con quanti bit, al minimo, possiamo scrivere **rappr(n)** ?
   Sia $\overline{N}$ questo numero...

2) Stabilito $\overline{N}$, decidiamo il numero **N** con cui vogliamo scrivere **rappr(n)**
   Sara` $N \geq \overline{N}$ e di solito scegliamo $N = \overline{N}$ ...

3) e poi calcoliamo il numerale **rappr(n)**, scritto su N bit

Rappresentazione in Complemento a 2 (fasi 1 e 2)

Dato un numero **n**, lo rappresentiamo in complemento a 2 tramite il numerale binario **rappr(n)**

Attention please – ci sono tre momenti in questo ragionamento …

1) "Qual è il minimo numero -- $\overline{N}$ -- di bit, che basta per rappresentare n ?"

La determinazione del numero $\overline{N}$ "minimo necessario" proviene da un'analisi dell'intervallo di rappresentabilità

per diversi K si considera l'intervallo di rappresentabilita`

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

facendosi queste domande
… il numero n è nell'intervallo di rappresentabilità per numerali di K bit?
… K è il minimo numero di bit necessari per rappresentare il numero n?

2) Facciamo che il numero di bit con cui scriviamo **rappr(n)** sia $N = \overline{N}$ … per brevita`

Rappresentazione in Complemento a 2 (fase 3: calcolo)

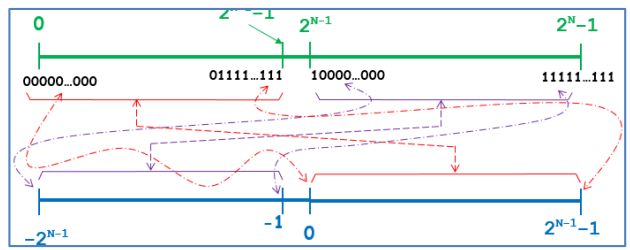
Dato un numero **n**, lo rappresentiamo in complemento a 2 tramite il numerale binario **rappr(n)**

Attention please – ci sono tre momenti in questo ragionamento …

3) Con N il numero di bit usati per scrivere i numerali

- se n è positivo && n <= $2^{N-1}-1$ **n ∈ [0, $2^{N-1}-1$]**
  rappr(n) = il normale numerale binario per n, in N bit
  (il primo bit e` 0, ovviamente)

- se n è negativo && n >= -2^{N-1} **n ∈ [-2^{N-1}, 0)**
  rappr(n) = il numerale binario, in N bit, che rappresenta il numero positivo $2^N - |n|$
  (com'è il primo bit?)



esempio: N=12

rappr(796) = "796 in binario su 12 bit" = ☺
rappr(-796) = "rappr(2^{12} – |-796|) in 12 bit" = ☺

Rappresentazione in Complemento a 2 (fase 3: calcolo)

3) Con N il numero di bit usati per scrivere i numerali

- se n è positivo && n <= 2^{N-1} -1 — $n \in [0, 2^{N-1} - 1]$
  rappr(n) = il normale numerale binario per n, in N bit
  (il primo bit e` 0, ovviamente)

- se n è negativo && n >= -2^{N-1} — $n \in [-2^{N-1}, 0)$
  rappr(n) = il numerale binario, in N bit, che rappresenta il numero positivo $2^N - |n|$
  (com'è il primo bit? E` 1, ovviamente)

esempio: N=12

rappr(796) = "796 in binario su 12 bit" =
rappr(-796) = "rappr($2^N - |n|$) in N bit" =

le due rappresentazioni sono tra le le seguenti? Quale è quale?

110011100100 001100011100
001100011100 110011100100
110011100100 001100011100

Rappresentazione in Complemento a 2 (fase 3: calcolo)

3) Con N il numero di bit usati per scrivere i numerali

- se n è positivo && n <= 2^{N-1} -1 — $n \in [0, 2^{N-1} -1]$

  rappr(n) = il normale numerale binario per n, in N bit
  (il primo bit e` 0, ovviamente)

- se n è negativo && n >= -2^{N-1} — $n \in [-2^{N-1}, 0)$

  rappr(n) = il numerale binario, in N bit, che rappresenta il numero positivo $2^N - |n|$
  (com'è il primo bit? E` 1, ovviamente)

esempio: N=12

rappr(796) = "796 in binario su 12 bit" = **001100011100**
(numero naturale 796)

rappr(-796) = "rappr($2^N - |n|$) in N bit"
= 4096-796 in binario su 12 bit = **110011100100**
(numero naturale 3300)

Rappresentazione in Complemento a 2 (esempi)

intervallo dei numeri rappresentabili in compl. a 2 con N bit (2^N numerali)

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

con 4 bit, ci sono 2alla4 numerali disponibili:
intervallo di rappresentabilità $[-2^3 , 2^3 - 1] = [-8, 7]$

intervallo dei numeri rappresentabili in compl. a 2 con 8 bit (2^8 numerali)

$[-2^7 , 2^7 - 1] = [-128, 127]$

adesso per esempio assumiamo n = 796

… con 10 bit (2^{10} numerali) ☺ intervallo di rappresentabiltà?
e … n è nell'intervallo?

Rappresentazione in Complemento a 2 (esempi)

intervallo dei numeri rappresentabili in compl. a 2 con N bit (2^N numerali)

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

con 4 bit, ci sono 2alla4 numerali disponibili:
intervallo di rappresentabilità $[-2^3 , 2^3 - 1] = [-8, 7]$

intervallo dei numeri rappresentabili in compl. a 2 con 8 bit (2^N numerali)

$$[-2^7 , 2^7 - 1] = [-128, 127]$$

adesso per esempio assumiamo n = 796

… con 10 bit (2^{10} numerali) $[-2^9 , 2^9 - 1]$ = ☺

Rappresentazione in Complemento a 2 (esempi)

...
intervallo dei numeri rappresentabili in compl. a 2 con 8 bit (2^N numerali)
$[-2^7 , 2^7 - 1] = [-128, 127]$

adesso per esempio assumiamo n = 796

con 10 bit (2^{10} numerali) $[-2^9 , 2^9 - 1] = [-512, 511]$

no, 796 non è nell'intervallo
10 bit non bastano

Rappresentazione in Complemento a 2 (esempi)

...
intervallo dei numeri rappresentabili in compl. a 2 con 8 bit (2^N numerali)
$[-2^7 , 2^7 - 1] = [-128, 127]$

adesso per esempio assumiamo n = 796

... con 10 bit (2^{10} numerali) $[-2^9 , 2^9 - 1] = [-512, 511]$

... con 11 bit (2^{11} numerali) ☺

Rappresentazione in Complemento a 2 (esempi)

...
intervallo dei numeri rappresentabili in compl. a 2 con 8 bit (2^N numerali)
$[-2^7 , 2^7 - 1] = [-128, 127]$

adesso per esempio assumiamo n = 796

... con 10 bit (2^{10} numerali) $[-2^9 , 2^9 - 1] = [-512, 511]$

... con 11 bit (2^{11} numerali) $[-2^{10} , 2^{10} - 1] = [-1024, 1023]$

796 è nell'intervallo (11 bit bastano)

remember: $N >= \overline{N}$

se riusciamo a rappresentare il numero con $\overline{N}$ bit, ci riusciamo anche con numeri di bit mggiori ...

... con 12 bit (2^{12} numerali) ☺
... con 13 bit (2^{13} numerali) ☺

Rappresentazione in Complemento a 2 (esempi)

intervallo dei numeri rappresentabili in compl. a 2 con 8 bit (2^N numerali)

$[-2^7 , 2^7 - 1] = [-128, 127]$

… con 10 bit (2^{10} numerali) $[-2^9 , 2^9 - 1] = [-512, 511]$

$[-1024, 1023]$

… con 11 bit (2^{11} numerali) $[-2^{10} , 2^{10} - 1]$ = ☺

796 è nell'intervallo

se riusciamo a rappresentare il numero con $\bar{N}$ bit, ci riusciamo anche con numeri di bit maggiori …

… con 12 bit (2^{12} numerali) $[-2^{11} , 2^{11} - 1] = [-2048, 2047]$

796 è nell'intervallo

… con 13 bit (2^{13} numerali) [-4096, 4095]

796 è nell'intervallo

Rappresentazione in Complemento a 2 (esempio)

(N=numero di bit usati per scrivere i numerali)

Dato un numero n,

- se n è positivo && $n <= 2^{N-1} - 1$
  rappr(n) = normale numerale binario in N bit (il primo bit è necessariamente 0)
- se n è negativo && $n >= -2^{N-1}$
  rappr(n) = $2^N - |n|$

esempio:

-796 quanti bit servono al minimo? 10? ☺

Rappresentazione in Complemento a 2 (esempio)

(N=numero di bit usati per scrivere i numerali)

Dato un numero n,

- se n è positivo && $n <= 2^{N-1} - 1$
  rappr(n) = normale numerale binario in N bit (il primo bit è necessariamente 0)
- se n è negativo && $n >= -2^{N-1}$
  rappr(n) = $2^N - |n|$

esempio:

-796 quanti bit servono al minimo? 10? ☺

no $-796 \notin [-2^9, 2^9 - 1] == [-512, 511]$

ne servono 11: $-796 \in [-2^{10}, 2^{10} - 1]$

quindi **rappr(-796)** = $2^{11} - 796 = 1252$

$(1252)_{10} = (\ ?\ )_2$

la rappresentazione di -796, con n=11 bit
è il numerale binario corrispondente al numero naturale 1252
(scritto su almeno 11 bit) … suvvia, scrivi quel numerale …

Rappresentazione in Complemento a 2 (esempio)

(N=numero di bit … sufficiente a che n sia rappresentabile)

Dato un numero n,

- se n è positivo && n <= $2^{N-1}-1$ **rappr(n)** = normale numerale binario in N bit (il primo bit è necessariamente 0)
- se n è negativo && n >= 2^{N-1} **rappr(n)** = $2^N - |n|$

esempio: -796

quanti bit servono al minimo? 10? no $-796 \notin [-2^9, 2^9-1]$

ne servono 11: $-796 \in [-2^{10}, 2^{10}-1]$

rappr(-796) = $2^{11} - 796 = 1252$

$(1252)_{10} = (\ 10011100100\ )_2$

1252/2 =	626	313	156	78	39	19	9	4	2	1	0
	0	0	1	0	0	1	1	1	0	0	1

☺ provare con N=14 invece che 11… che numerale viene?

Rappresentazione in Complemento a 2 (esempio)

(N=numero di bit … sufficiente a che n sia rappresentabile)

Dato un numero n,

- se n è positivo && n <= $2^{N-1}-1$ **rappr(n)** = normale numerale binario in N bit (il primo bit è necessariamente 0)
- se n è negativo && n >= 2^{N-1} **rappr(n)** = $2^N - |n|$

esempio: -796

quanti bit servono al minimo? 10? no $-796 \notin [-2^9, 2^9-1]$

ne servono 11: $-796 \in [-2^{10}, 2^{10}-1]$

rappr(-796) = $2^{11} - 796 = 1252$

$(1252)_{10} = (\ 10011100100\ )_2$

1252/2 =	626	313	156	78	39	19	9	4	2	1	0
	0	0	1	0	0	1	1	1	0	0	1

su 32 bit

796 00000000000000000000001100011100

-796 11111111111111111111110011100100

☺ c'era sempre da provare con N==14 …

Rappresentazione in Complemento a 2 – metodo alternativo

(N=numero di bit … sufficiente a che n sia rappresentabile)

Dato un numero n,

- se (n è positivo) && (n <= 2^{N-1} -1) **rappr(n)** = normale numerale binario in N bit (il primo necessariamente 0)
- se (n è negativo) && (|n| <= 2^{N-1}) la **rappr(n)** si ottiene
  - rappresentando in binario puro il numero positivo corrispondente (|n|)
  - invertendo i bit di tale rappresentazione
  - e sommando 1

esempio: -796 NB per la rappresentazione binaria pura bastano 10 bit, ma per la rappr. in compl. a 2 ne servono 11

- **$(796)_{10}$ =** **1 1 0 0 0 1 1 1 0 0**
- **in 11 bit** **0 1 1 0 0 0 1 1 1 0 0**
- **inversione** **1 0 0 1 1 1 0 0 0 1 1**
- **+ 1** **1 0 0 1 1 1 0 0 1 0 0**
- **rappr(-796) =** 10011100100

provare con N=14 …

su 32 bit

796 00000000000000000000001100011100

-796 11111111111111111111110011100100

Rappresentazione in Complemento a 2
DECODIFICA da rappr(n) a n …

(N=numero di bit)

Data la rappr(n), qual è n?
Si procede seguendo il metodo di rappresentazione al contrario …

- se n è positivo (cioè rappr(n) inizia con 0, si decodifica normalmente, come in binario puro
- se n è negativo (rappr(n) inizia con 1) n si ottiene
  - sottraendo 1
  - invertendo i bit
  - decodificando il numero binario normalmente

esempio: rappr(n) = 10110001 (è negativo)

```
- -79              1 0 1 1 0 0 0 1 -
- sottr 1                        1
                   1 0 1 1 0 0 0 0
- inversione       0 1 0 0 1 1 1 1
                   e questo ↑ è 79
                   quindi il numerale di partenza rappresenta -79
```

Esercizi – complemento a due

Per i seguenti numeri interi, stabilire il minimo numero di cifre necessarie per la rappresentazione in complemento a due, e poi scrivere la rappresentazione medesima.

19 -19 65 715 -716 32000 -231231

Riscrivere poi tutti i numeri, in complemento a due su 32 bit.

(Quanti '1' contiene -231231?

scrivilo, prima di guardare la prossima slide

Esercizi – complemento a due

Per i seguenti numeri interi, stabilire il minimo numero di cifre necessarie per la rappresentazione in complemento a due, e poi scrivere la rappresentazione medesima.

19 -19 65 715 -716 32000 -231231

Riscrivere poi tutti i numeri, in complemento a due su 32 bit.

(Quanti '1' contiene -231231?

meno di 15? Sbagliato.
Si intende quanti 1 contiene la rappresentazione in complemento a due di 231231)

Tecniche della Programmazione, lez. 12

- Approfondimenti

A partire dal NUMERALE in base 10: Calcolo del NUMERALE in base b

dato un numero decimale, ci piacerebbe trovare le cifre del numerale in base b, facendo i conti in base di partenza (decimale)
Cominciamo facile: con divisioni successive posso isolare le cifre del numeral decimale, una per una, dalla meno significatva alla piu` significativa

$$(c_{n-1}c_{n-2}\ldots c_2c_1c_0)_{10}$$

$$numero = \frac{numero}{10} \cdot 10 + RESTO \qquad RESTO = c_0$$

$$\frac{numero}{10} = \frac{\frac{numero}{10}}{10} \cdot 10 + RESTO \qquad RESTO = c_1$$

e cosi` via

Ex. $1969 = (1969/10)\cdot 10 + 9$
$196 = (196/10)\cdot 10 + 6$
$19 = (19/10)\cdot 10 + 9$
$1 = (1/10)\cdot 10 + 1$
0 ... stop

Ehm, qui "numero" si intende "numerale", dato che lavoriamo sempre con le cifre.

Ora definiamo il metodo delle divisioni successive per ottenere la il numero binario corrispondente ad uno decimale, ma il principio e` generale – Proma di proseguire vedi Approfondimenti – e poi rivedili dopo ...

Vedi Approfondimenti

Calcolo delle cifre in base b per un numero decimal

per ritrovare la prima cifra (c_0) del numero rappresentato da

$$(c_{n-1}c_{n-2}\ldots c_2c_1c_0)_b$$

$$numero = \frac{numero}{b} \cdot b + RESTO \qquad RESTO = c_0$$

le operazioni sono fatte nel sistema "di partenza"

$$\sum_{k=n-1}^{0} c_k \cdot b^k \;/\; b \cdot b + c_0 = \sum_{k=n-1}^{0} c_k \cdot b^k$$

$$(c_{n-1} \cdot b^{n-1} + c_{n-2} \cdot b^{n-2} + \ldots + c_1 \cdot b^1 + c_0 \cdot b^0) \;/\; b \cdot b + c_0 =$$

$$= c_{n-1} \cdot b^{n-1} + c_{n-2} \cdot b^{n-2} + \ldots + c_1 \cdot b^1 + c_0 \cdot b^0$$

Calcolo delle cifre in base b per un numero decimal

$$(c_{n-1} \cdot b^{n-1} + c_{n-2} \cdot b^{n-2} + \ldots + c_1 \cdot b^1 + c_0 \cdot b^0) \;/\; b \cdot b + c_0 =$$

$$= c_{n-1} \cdot b^{n-1} + c_{n-2} \cdot b^{n-2} + \ldots + c_1 \cdot b^1 + c_0 \cdot b^0$$

NB se usassi QUESTO NUMERO

per ripetere l'operazione ...

dividendolo per b e poi moltiplicando per b ... otterrei c_1

e continuando si ottengono anche le altre cifre:

Il metodo si chiama ...

Metodo delle divisioni successive

per trovare le cifre (in base b) che rappresentano il numero dato (usando la rappresentazione e le operazioni decimali del numero)

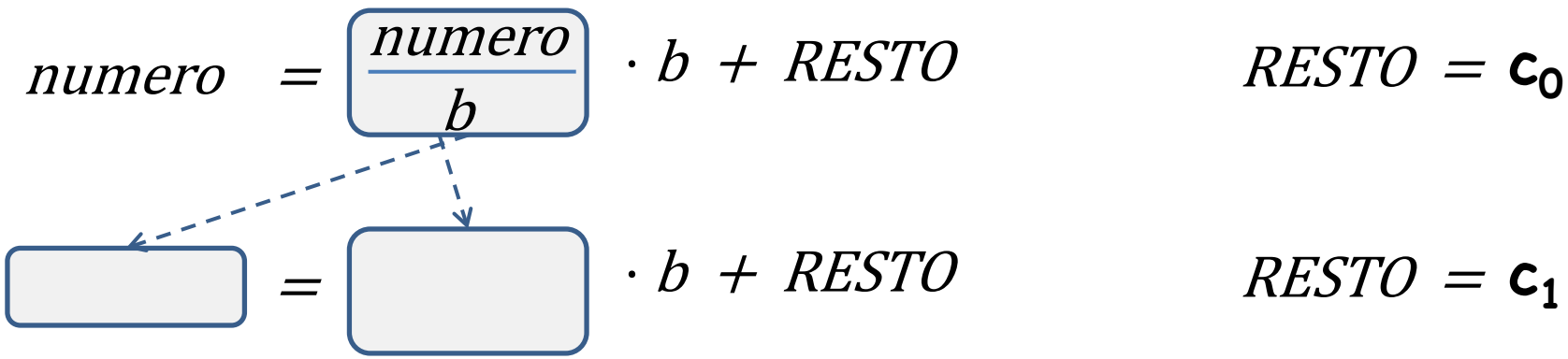
$$\textbf{numero} = (c_{n-1}c_{n-2}\ldots c_2c_1c_0)_b$$

$$numero = \frac{numero}{b} \cdot b + RESTO \qquad RESTO = c_0$$

Metodo delle divisioni successive

per trovare le cifre (in base b) che rappresentano il numero dato (usando la rappresentazione e le operazioni decimali del numero)

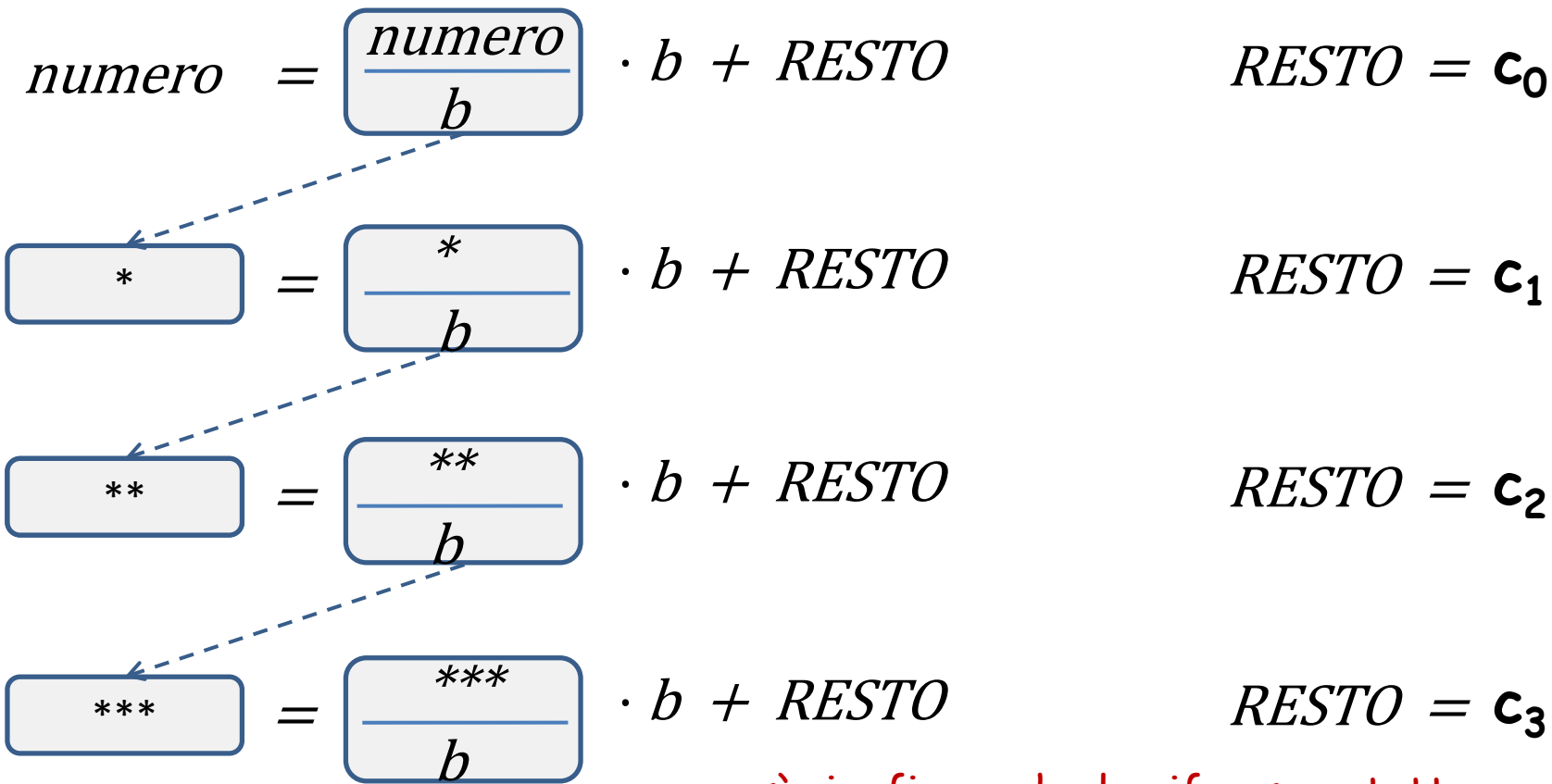
numero = $(c_{n-1}c_{n-2}\ldots c_2c_1c_0)_b$



Metodo delle divisioni successive

per trovare le cifre (in base b) che rappresentano il numero dato (usando la rappresentazione e le operazioni decimali del numero)

numero = $(c_{n-1}c_{n-2}\ldots c_2c_1c_0)_b$

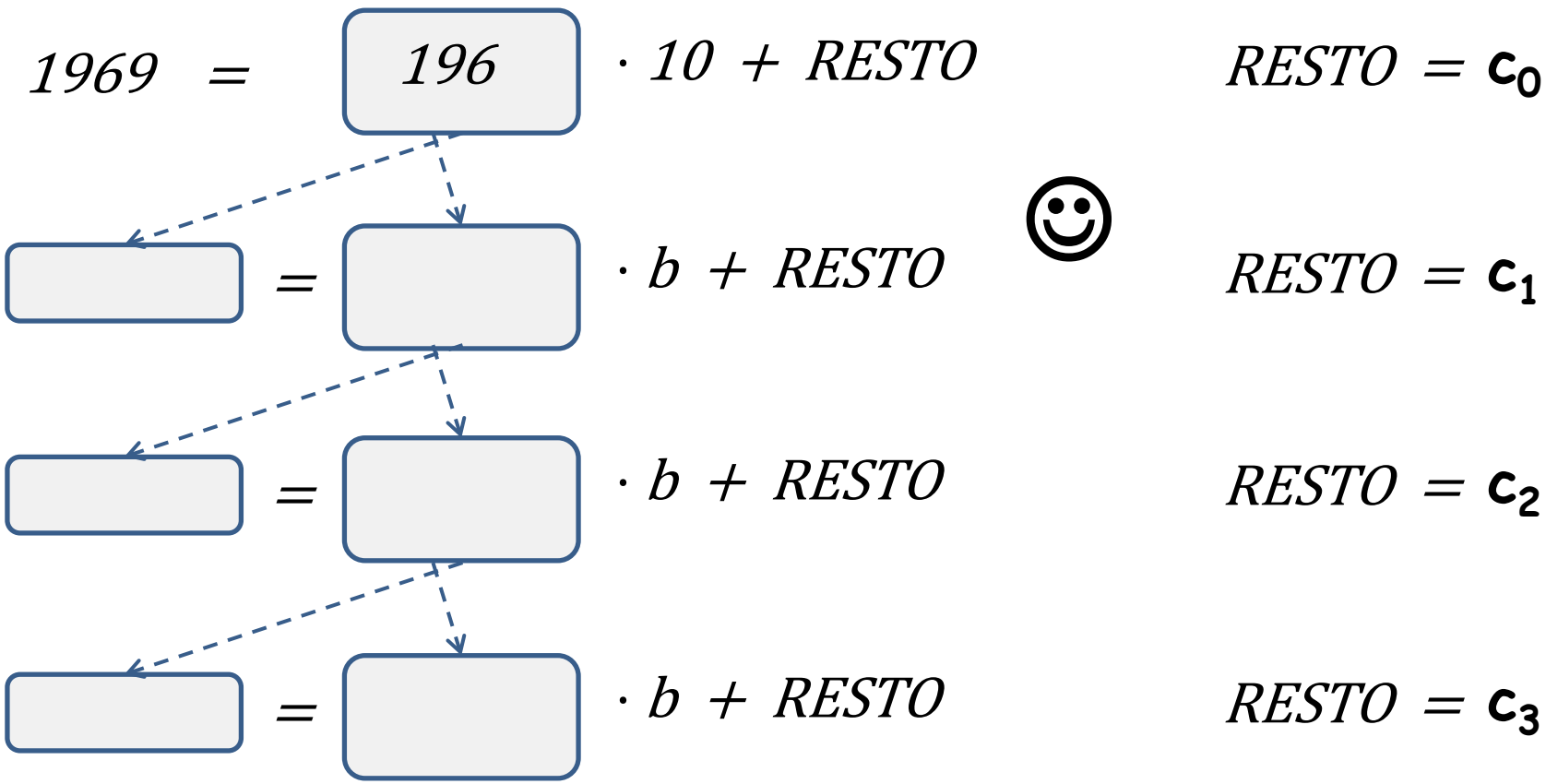


e così via, fino a che le cifre sono tutte

Metodo delle divisioni successive

per trovare le cifre (in base b) che rappresentano il numero dato (usando la rappresentazione e le operazioni decimali del numero)

numero = $(\ldots c_5 c_4 c_3 c_2 c_1 c_0)_b$

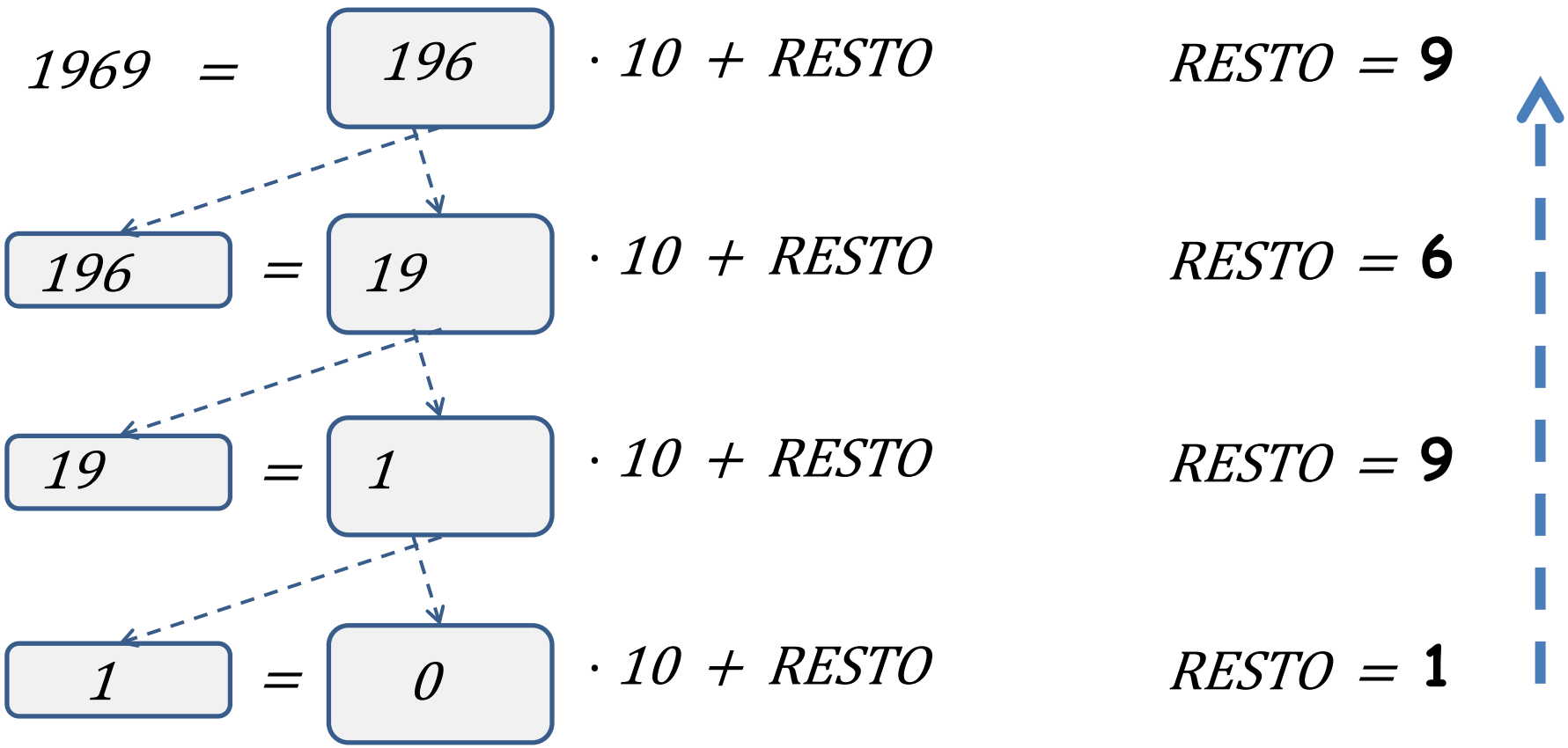


Metodo delle divisioni successive

per trovare le cifre (in base b) che rappresentano il numero dato (usando la rappresentazione e le operazioni decimali del numero)

quando la divisione (intera) viene 0 … il resto è l'ultima cifra

numero $= (c_{n-1}c_{n-2}\ldots c_2c_1c_0)_b$



basi

NB con n cifre si rappresentano b^n numeri naturali in base b da 0 a $(b^n - 1)$ … $[0, (b^n - 1)]$

base 8 (Sistema ottale – o anche *octal*, *oct*)

cifre	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
contiamo …	0	1	2	3	4	5	6	7
	10	11	12	13	14	15	16	17
	20	21	22	23	24	…		
	…	**77**	100	101				

$(31)_8 = (?)_{10}$

Vedi Approfondimenti

feste confuse (dagli informatici)

base 8 (Sistema ottale – o anche *octal*, *oct*)

cifre	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
contiamo …	0	1	2	3	4	5	6	7
	10	11	12	13	14	15	16	17
	20	21	22	23	24	…		
	…	**77**	100	101				

$(31)_8 = (3 \times 8 + 1 = 25)_{10}$

quindi (Dec = sistema decimale …)

Dec 25 = Oct 31

feste confuse (dagli informatici)

base 8 (Sistema ottale – o anche *octal*, *oct*)

cifre	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
contiamo …	0	1	2	3	4	5	6	7
	10	11	12	13	14	15	16	17
	20	21	22	23	24	…		
	…	**77**	100	101				

$(31)_8 = (3 \times 8 + 1 = 25)_{10}$

quindi (Dec = sistema decimale …)

Dec 25 = Oct 31

base 2

numerale binario con al piu` n cifre:
INTERVALLO DI RAPPRESENTABILITA`
e` la sequenza di numeri naturali $[0, (2^n - 1)]$

base 2

cifre	**0 1**
contiamo …	0 1
	10 11
	100 101 110 111
	1000 1001 1010 1011 1100 1101 1110 1111

con 32 bit si rappresentano i numeri naturali in $[0, (2^{32} - 1)]$

con 5 bit si rappresentano i numeri naturali in [0, 31]

Quindi se lo spazio per i numerali è limitato, altrettanto limitato è l'insieme dei numeri rappresentati dai numerali

Quanti bit servono per rappresentare 64 256 ?

basi

NB con n cifre si rappresentano b^n numeri naturali in base b da 0 a $(b^n - 1)$ … $[0, (b^n - 1)]$

base 2

cifre	**0**	**1**					
contiamo …	0	1					
	10	11					
	100	101	110	111			
	1000	1001	1010	1011	1100	1101	1110	1111

con 32 bit si rappresentano i numeri naturali in $[0, (2^{32} - 1)]$

con 5 bit si rappresentano i numeri naturali in $[0, 31]$

Quindi se lo spazio per i numerali è limitato, altrettanto limitato è l'insieme dei numeri rappresentati dai numerali

Quanti bit servono per rappresentare 64 256 ?

6 8

?

basi

NB con n cifre si rappresentano b^n numeri naturali in base b da 0 a $(b^n - 1)$ … $[0, (b^n - 1)]$

base 2

cifre	**0 1**
contiamo …	0 1
	10 11
	100 101 110 111
	1000 1001 1010 1011 1100 1101 1110 1111

con 32 bit si rappresentano i numeri naturali in $[0, (2^{32} - 1)]$

con 5 bit si rappresentano i numeri naturali in $[0, 31]$

Quindi se lo spazio per i numerali è limitato, altrettanto limitato è l'insieme dei numeri rappresentati dai numerali

Quanti bit servono per rappresentare 64 256 ?

6 8

$[0, (2^6 - 1)]$! !

?

basi

NB con n cifre si rappresentano b^n numeri naturali in base b da 0 a $(b^n - 1)$ … $[0, (b^n - 1)]$

base 2

cifre	**0 1**
contiamo …	0 1
	10 11
	100 101 110 111
	1000 1001 1010 1011 1100 1101 1110 1111

con 32 bit si rappresentano i numeri naturali in $[0, (2^{32} - 1)]$

con 5 bit si rappresentano i numeri naturali in $[0, 31]$

Quindi se lo spazio per i numerali è limitato, altrettanto limitato è l'insieme dei numeri rappresentati dai numerali

Quanti bit servono per rappresentare 64 256 ?

6 8

intervallo di rappresentabilità

$[0, (2^6 - 1)]$! !

$[0, 63]$ non contiene 64

6 non bastano per 64;
8 non bastano per 256

!

basi

NB con n cifre si rappresentano b^n numeri naturali in base b da 0 a $(b^n - 1)$ … $[0, (b^n - 1)]$

base 2

cifre	**0 1**
contiamo …	0 1
	10 11
	100 101 110 111
	1000 1001 1010 1011 1100 1101 1110 1111

con 32 bit si rappresentano i numeri naturali in $[0, (2^{32} - 1)]$

con 5 bit si rappresentano i numeri naturali in $[0, 31]$

Quindi se lo spazio per i numerali è limitato, altrettanto limitato è l'insieme dei numeri rappresentati dai numerali

Quanti bit servono per rappresentare 64 256 ?

6 8

$[0, (2^8 - 1)]$! !

[0, 255] non contiene 256

intervallo di rappresentabilità

!

basi

NB con n cifre si rappresentano b^n numeri naturali in base b da 0 a $(b^n - 1)$ … $[0, (b^n - 1)]$

base 2

cifre	**0 1**
contiamo …	0 1
	10 11
	100 101 110 111
	1000 1001 1010 1011 1100 1101 1110 1111

con 32 bit si rappresentano i numeri naturali in $[0, (2^{32} - 1)]$

con 5 bit si rappresentano i numeri naturali in $[0, 31]$

Quindi se lo spazio per i numerali è limitato, altrettanto limitato è l'insieme dei numeri rappresentati dai numerali

Quanti bit servono per rappresentare 64 256 796 ? 1025 ?

7 **9**

$[0, (2^7 - 1)]$ $[0, (2^9 - 1)]$

ok, con questo numero di cifre ce la facciamo

basi

NB con n cifre si rappresentano b^n numeri naturali in base b da 0 a $(b^n - 1)$ … $[0, (b^n - 1)]$

base 2

cifre	**0 1**
contiamo …	0 1
	10 11
	100 101 110 111
	1000 1001 1010 1011 1100 1101 1110 1111

con 32 bit si rappresentano i numeri naturali in $[0, (2^{32} - 1)]$

con 5 bit si rappresentano i numeri naturali in $[0, 31]$

Quindi se lo spazio per i numerali è limitato, altrettanto limitato è l'insieme dei numeri rappresentati dai numerali

Quanti bit servono per rappresentare 796 1025 2047?

10 11

$[0, (2^{10} - 1)]$ $[0, (2^{11} - 1)]$

basi

NB con n cifre si rappresentano b^n numeri naturali in base b da 0 a $(b^n - 1)$ … $[0, (b^n - 1)]$

base 2

cifre	**0** **1**
contiamo …	0 1
	10 11
	100 101 110 111
	1000 1001 1010 1011 1100 1101 1110 1111

con 32 bit si rappresentano i numeri naturali in $[0, (2^{32} - 1)]$

con 5 bit si rappresentano i numeri naturali in [0, 31]

Quindi se lo spazio per i numerali è limitato, altrettanto limitato è l'insieme dei numeri rappresentati dai numerali

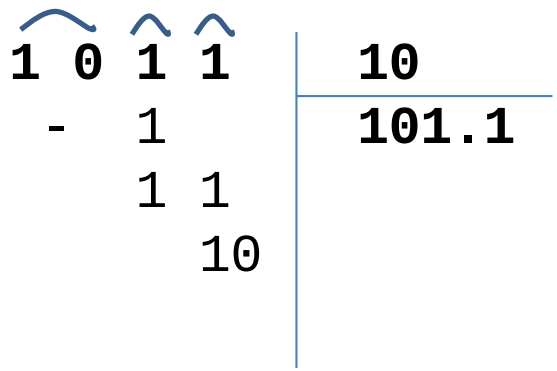
Quanti bit servono per rappresentare

1025	2047
11	11
$[0, (2^{11} - 1)]$	$[0, (2^{11} - 1)]$

operazioni in binario

Gli algoritmi sono i medesimi nei vari sistemi posizionali …

5 bit
divisione



1011 / 10 è 101.1

è vero? ☺

operazioni in binario

Gli algoritmi sono i medesimi nei vari sistemi posizionali …

5 bit
divisione

```
1 0 1 1 | 10
  -   1 | 101.1
    1 1 |
      10|
```

1011 = $1x2^3 + 0x2^2 + 1x2 + 1 = 11$

101.1 = $1x2^2 + 0x2^1 + 1 + 1x2^{-1} = 5.5$

11/2=5.5 effettivamente

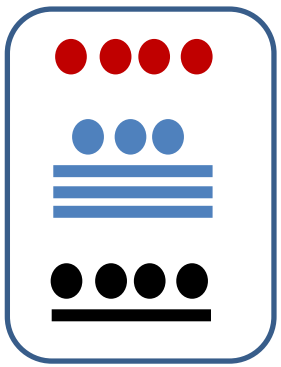
e 101.1 moltiplicato 10 quanto fa? ☺

Tecniche della Programmazione, lez. 12

- Esercizi

ACME manuals: Maya number system

millenovecentosessantanove →



= 4 * (20*20) = 1600

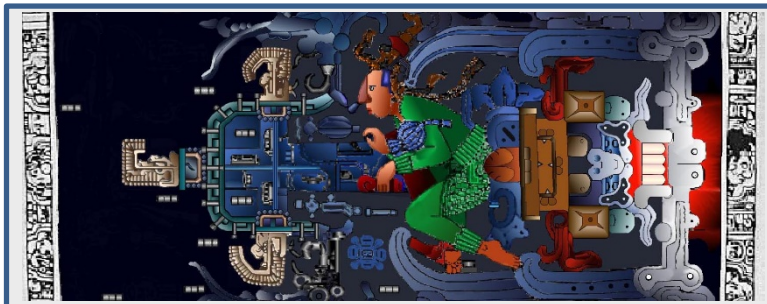
= 18 * (20) = 360

= 9 * (1) = 9

Uso base …
peso della cifra più in basso: 1;
peso della seconda cifra, salendo, 20;
peso della terza cifra 20^2 cioè 400
… 20^3…20^4…20^5 …

Numero	Forma verticale	Forma orizzontale	Numero	Forma verticale	Forma orizzontale
0			10		
1			11		
2			12		
3			13		
4			14		
5			15		
6			16		
7			17		
8			18		
9			19		

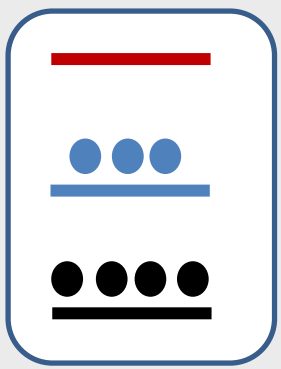
sistema di numerazione Maya;
Numeri in **base 20**, scritti verticalmente – cifra più pesante più in alto



Uso mistico

18*20 invece di 20*20
(fa 360, che è special)…

millenovecentosessantanove →



= 5 * (18*20) = 1800

= 8 * 20 = 160

= 9 * 1 = 9

Sistemi di numerazione posizionali

Quello che usiamo normalmente (sistema decimale) è un sistema posizionale in base 10.

$$1969 = 1\times10^3 + 9\times10^2 + 6\times10^1 + 9\times10^0$$

"posizionale" = ogni cifra ha un peso, che è una potenza della base; la potenza dipende dalla posizione della cifra nel numerale

numero espresso in base ***b***, con un numerale di n cifre (in cui ogni cifra è una tra le ***b*** possibili): $(c_{n-1}c_{n-2}\ldots c_2c_1c_0)_b$

corrisponde al valore (nel sistema decimale)

$$\sum_{k=n-1}^{0} c_k \cdot b^k$$

numerale

$$(c_{n-1}c_{n-2}\ldots c_2c_1c_0)_b = c_{n-1}\cdot b^{n-1} + c_{n-2}\cdot b^{n-2} + \ldots + c_1\cdot b^1 + c_0\cdot b^0$$

A partire dal NUMERALE in base 2: Calcolo del NUMERALE in base 10

Si usa il POLINOMIO che abbiamo visto prima

n cifre in base b ➔ numerale in base 10

$(c_{n-1}c_{n-2}\ldots c_2c_1c_0)_b = \sum_{k=n-1}^{0} c_k \cdot b^k$ (NB operazioni in base 10)

$1011 = 1x2^0 + 1x2^1 + 0x2^2 + 1x2^3$

10111 = ☺
0001011 = ☺

A partire dal NUMERALE in base 2: Calcolo del NUMERALE in base 10

Si usa il POLINOMIO che abbiamo visto prima

n cifre in base b ➔ numerale in base 10

$(c_{n-1}c_{n-2}\ldots c_2c_1c_0)_b = \sum_{k=n-1}^{0} c_k \cdot b^k$ (NB operazioni in base 10)

$1011 = 1x2^0 + 1x2^1 + 0x2^2 + 1x2^3$

$10111 = 1x2^0 + 1x2^1 + 1x2^2 + 0x2^3 + 1x2^4$

0001011 = ☺

A partire dal NUMERALE in base 2: Calcolo del NUMERALE in base 10

Si usa il POLINOMIO che abbiamo visto prima

n cifre in base b ➔ numerale in base 10

$(c_{n-1}c_{n-2}\ldots c_2c_1c_0)_b = \sum_{k=n-1}^{0} c_k \cdot b^k$ (NB operazioni in base 10)

$1011 = 1x2^0 + 1x2^1 + 0x2^2 + 1x2^3$

$10111 = 1x2^0 + 1x2^1 + 1x2^2 + 0x2^3 + 1x2^4$

$0001011 = 1x2^0 + 1x2^1 + 0x2^2 + 1x2^3 + 0x2^4 + 0x2^5 + 0x2^6$

operazioni in binario

Gli algoritmi sono i medesimi nei vari sistemi posizionali …

5 bit
moltiplicazione

```
                0 1 0 0 1
                0 0 1 0 1
                ---------
                0 1 0 0 1
              0 0 0 0 0
            0 1 0 0 1
          0 0 0 0 0
        0 0 0 0 0
        -----------------
        0 0 0 1 0 1 1 0 1
```

= … ☺ **calcolo del numero decimale a partire dalla rappresentazione binaria**

Prova e poi **Vedi dopo**

operazioni in binario

5 bit
moltiplicazione

```
                0 1 0 0 1
                0 0 1 0 1
                ---------
                0 1 0 0 1
              0 0 0 0 0
            0 1 0 0 1
          0 0 0 0 0
        0 0 0 0 0
        -----------------
        0 0 0 1 0 1 1 0 1
```

calcolo del nunmero decimale a partire dalla rappresentazione binaria

$= 1x2^5 + 0x2^4 + 1x2^3 + 1x2^2 + 0x2^1 + 1 = 45$

operazioni in binario

5 bit
moltiplicazione

```
              0 1 0 0 1
              0 0 1 0 1
              ---------
              0 1 0 0 1
            0 0 0 0 0
          0 1 0 0 1
        0 0 0 0 0
      0 0 0 0 0
      ---------------
      0 0 0 1 0 1 1 0 1
```

calcolo del nunmero decimale a partire dalla rappresentazione binaria

$= 1x2^5 + 0x2^4 + 1x2^3 + 1x2^2 + 0x2^1 + 1 = 45$

operazioni in binario

5 bit
moltiplicazione

```
                0 1 0 0 1
                0 0 1 0 1
                ---------
                0 1 0 0 1
              0 0 0 0 0
            0 1 0 0 1
          0 0 0 0 0
        0 0 0 0 0
        ------------------
        0 0 0 1 0 1 1 0 1
```

calcolo del nunmero decimale a partire dalla rappresentazione binaria

$= 1x2^5 + 0x2^4 + 1x2^3 + 1x2^2 + 0x2^1 + 1 = 45$

operazioni in binario

5 bit
moltiplicazione

```
                0 1 0 0 1
                0 0 1 0 1
                ---------
                0 1 0 0 1
              0 0 0 0 0
            0 1 0 0 1
          0 0 0 0 0
        0 0 0 0 0
        ---------------
        0 0 0 1 0 1 1 0 1
```

calcolo del nunmero decimale a partire dalla rappresentazione binaria

$= 1x2^5 + 0x2^4 + 1x2^3 + 1x2^2 + 0x2^1 + 1 = 45$

operazioni in binario

5 bit
moltiplicazione

```
              0 1 0 0 1
              0 0 1 0 1
              ---------
              0 1 0 0 1
            0 0 0 0 0
          0 1 0 0 1
        0 0 0 0 0
      0 0 0 0 0
      ---------------
      0 0 0 1 0 1 1 0 1
```

calcolo del nunmero decimale a partire dalla rappresentazione binaria

$= 1x2^5 + 0x2^4 + 1x2^3 + 1x2^2 + 0x2^1 + 1 = 45$

operazioni in binario

5 bit
moltiplicazione

```
              0 1 0 0 1
              0 0 1 0 1
              ---------
              0 1 0 0 1
            0 0 0 0 0
          0 1 0 0 1
        0 0 0 0 0
      0 0 0 0 0
      -----------------
      0 0 0 1 0 1 1 0 1
```

calcolo del nunmero decimale a partire dalla rappresentazione binaria

$= 1x2^5 + 0x2^4 + 1x2^3 + 1x2^2 + 0x2^1 + 1 = 45$

operazioni in binario

5 bit
moltiplicazione

```
              0 1 0 0 1
              0 0 1 0 1
              ---------
              0 1 0 0 1
            0 0 0 0 0
          0 1 0 0 1
        0 0 0 0 0
      0 0 0 0 0
      -----------------
      0 0 0 1 0 1 1 0 1
```

calcolo del nunmero decimale a partire dalla rappresentazione binaria

$= 0x2^8 + 0x2^7 + 0x2^6 + 1x2^5 + 0x2^4 + 1x2^3 + 1x2^2 + 0x2^1 + 1 = 45$