



Università degli Studi dell'Insubria  
Dipartimento di Scienze Teoriche e Applicate


---

## Architettura degli elaboratori

La rappresentazione dell'informazione nei calcolatori

Marco Tarini  
Dipartimento di Scienze Teoriche e Applicate  
marco.tarini@uninsubria.it

---



## Introduzione

---

- A livello di circuiti logici i calcolatori memorizzano (e rappresentano) valori basati su bit, gestiscono quindi zeri e uni.
- Poiché a noi interessa memorizzare ed elaborare numeri, parole, immagini, suoni, ecc. Occorre progettare un sistema che consenta di rappresentare le informazioni interessanti in termini di bit.
- Vedremo quindi come si rappresentano le informazioni fondamentali: numeri (naturali, interi, reali) e caratteri.
- Daremo dei cenni sulla rappresentazione di informazioni più sofisticate e complesse.

---

Architettura degli elaboratori - 2 - Aritmetica binaria



## La rappresentazione dei Numeri

- Cominciamo con i numeri interi non negativi (numeri naturali, incluso lo zero)
- Gli interi non negativi sono rappresentabili con diverse notazioni:
  - ▶ additiva romana: I, II, III, IV, V, .... IX, X, XI...
  - ▶ posizionale indo-araba: 1, 2, .. 10, 11, ... 100, ...




## Notazioni posizionali

- Ogni simbolo contribuisce con un valore che dipende dalla sua posizione e dalla base di rappresentazione  $B$ .
- Ad esempio, dato

$$d_k d_{k-1} \dots d_1 d_0$$

il valore denotato è dato dalla formula:

$$d_0 B^0 + d_1 B^1 + d_2 B^2 + d_3 B^3 + \dots + d_k B^k$$




## Notazione decimale

---

- La notazione decimale è la notazione posizionale avente base 10.
- Pertanto il valore di  $d_k d_{k-1} \dots d_1 d_0$  è dato dalla formula:
 
$$d_0 10^0 + d_1 10^1 + d_2 10^2 + d_3 10^3 + \dots$$
- Ad es.  $429 = 9 10^0 + 2 10^1 + 4 10^2$

		1	4	2	9
...	$10^4$	$10^3$	$10^2$	$10^1$	$10^0$

Architettura degli elaboratori
- 6 -
Aritmetica binaria



## Altre basi


---

- Ogni numero è esprimibile in modo univoco in una qualunque base

Stringa	Base	Calcolo valore	Valore in base 10
13	4	$4 * 1 + 3$	7
13	8	$8 * 1 + 3$	11
13	10	$10 * 1 + 3$	13
13	16	$16 * 1 + 3$	19

- Basi di particolare interesse:
  - ▶ base B=2 → due sole cifre: 0 e 1
  - ▶ base B=8 → otto cifre: 0, 1, 2, 3, 4, 5, 6, 7
  - ▶ base B=10 → dieci cifre: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
  - ▶ base B=16 → sedici cifre: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Architettura degli elaboratori
- 7 -
Aritmetica binaria




## Conversioni alla base 10

- La conversione da una base B a base 10 si può fare applicando direttamente la formula
 
$$\text{valore} = d_0 B^0 + d_1 B^1 + d_2 B^2 + d_3 B^3 + \dots$$
 e scrivendo il valore in base 10, come siamo abituati
- Ad es.  $351_6 = 1 + 5 * 6 + 3 * 6^2 = 1 + 30 + 108 = 139$

Rappresentazione in base 10 della cifra

Rappresentazione in base 10 della base

Architettura degli elaboratori
- 8 -
Aritmetica binaria



## Conversioni **dalla** base 10

- Vogliamo scrivere in un numero N in base B:
 
$$N = d_0 B^0 + d_1 B^1 + d_2 B^2 + d_3 B^3 + \dots + d_k B^k$$
- Dobbiamo quindi determinare il valore di  $d_0, d_1, d_2, \dots, d_k$
- Osservando che la formula precedente si può riscrivere in questo modo:
 
$$N = d_0 + B ( d_1 + B ( d_2 + B ( d_3 + \dots ) ) )$$
- Quindi  $d_0$  è dato dal resto della divisione del numero dato per B.
- Il risultato della divisione è  $d_1 + B ( d_2 + B ( d_3 + \dots ) ) = M$
- Quindi  $d_1$  è il resto di  $M/B$
- Ecc ...

Architettura degli elaboratori
- 9 -
Aritmetica binaria



## Algoritmo delle divisioni successive

- Dato un numero N in base dieci, per convertirlo in una stringa di caratteri che ne rappresenti la codifica in base B si opera per divisioni successive, calcolando i caratteri della stringa dal meno significativo al più significativo.

```
int convert (int s[], int n) {
    int i=0, r;
    s[i++] = n%B;
    n = n/B;
    while(n>0) {
        s[i++] = n%B;
        n = n/B;
    }
    return i;
}
```




## Esempi

- Convertiamo  $11_{10}$  in base 2.

N	resti
11	1
5	1
2	0
1	1
0	0

$1011 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 8 + 2 + 1 = 11$



## Esempi


---

- Convertiamo  $49_{10}$  in base 2.

N	resti
49	1
24	0
12	0
6	0
3	1
1	1
0	

$110001 = 1 + 16 + 32 = 49$

Architettura degli elaboratori
- 12 -
Aritmetica binaria



## Esempi

---

- Convertiamo  $57_{10}$  in base 4.

N	resti
57	1
14	2
3	3
0	

$321 = 3 \cdot 4^2 + 2 \cdot 4^1 + 1 \cdot 4^0 =$   
 $3 \cdot 16 + 2 \cdot 4 + 1 = 48 + 8 + 1 = 57$

Architettura degli elaboratori
- 13 -
Aritmetica binaria



## La rappresentazione binaria

- Poiché nei calcolatori è facile rappresentare l'informazione elementare in modo binario (0 e 1) siamo interessati soprattutto alla base 2.
- Nei calcolatori i numeri interi sono rappresentati in base due, cioè come sequenze di zeri e uni.



## Conversione tra una base e una sua potenza (es fra base 6 e base 36)

- Consideriamo due rappresentazioni in base B e P dello stesso valore  

$$d_0 B^0 + d_1 B^1 + d_2 B^2 + d_3 B^3 + \dots + d_k B^k$$


$$e_0 P^0 + e_1 P^1 + e_2 P^2 + e_3 P^3 + \dots + e_j P^j$$
- Consideriamo cosa succede quando P è una potenza di B, cioè  $P = B^m$ .  

$$e_0 P^0 + e_1 P^1 + e_2 P^2 + e_3 P^3 + \dots + e_j P^j$$

$$=$$

$$e_0 B^0 + e_1 B^m + e_2 B^{2m} + e_3 B^{3m} + \dots + e_j B^{jm}$$
- Se dividiamo per P  $e_0 P^0 + e_1 P^1 + e_2 P^2 + e_3 P^3 + \dots + e_j P^j$  otteniamo come resto  $e_0$
- Se dividiamo per P  $d_0 B^0 + d_1 B^1 + d_2 B^2 + d_3 B^3 + \dots + d_k B^k$  otteniamo come resto  $d_0 B^0 + d_1 B^1 + \dots + d_{m-1} B^{m-1}$
- Poiché le due rappresentazioni denotano lo stesso valore,  

$$e_0 = d_0 B^0 + d_1 B^1 + \dots + d_{m-1} B^{m-1}$$




### Conversione tra una base e una sua potenza (es fra base 6 e base 36)

---

- Essendo  $e_0 = d_0 B^0 + d_1 B^1 + \dots + d_{m-1} B^{m-1}$
- $e_0$  (la prima cifra in base P) è ricavabile osservando le prime m cifre ( $d_0, d_1, \dots, d_{m-1}$ ) in base B.
  - Basta applicare la formula  $e_0 = d_0 B^0 + d_1 B^1 + \dots + d_{m-1} B^{m-1}$
- Per determinare le prime m cifre in base B possiamo limitarci a convertire  $e_0$  in base B.
- Le cifre successive, ( $e_1$ , ecc.) si determinano applicando le divisioni successive:
  - otteniamo che l'i-esima cifra in base P corrisponde all'i-esimo gruppo di m cifre in base B

Architettura degli elaboratori
- 16 -
Aritmetica binaria



### Conversione tra una base e una sua potenza Un esempio familiare: fra base 10 e base 1000

---


- Immaginiamo che le 1000 «cifre» in base 1000 siano 000, 001, 002, 003, ..., 999
- $1000 = 10^3$ ,  
quindi ogni cifra in base 1000 corrisponderà a 3 cifre in base 10.
  - $\text{000}_{1000} = 0\ 0\ 0_{10}$
  - $\text{001}_{1000} = 0\ 0\ 1_{10}$
  - $\text{002}_{1000} = 0\ 0\ 2_{10}$
  - $\text{123}_{1000} = 1\ 2\ 3_{10}$
- es: popolazione mondiale (in base 10):  $7347650700_{10}$

comunemente scritta:  
 in eng: 7,347,650,700  
 in ita: 7.347.650.700

	cifra dei miliardi ( $1000^3$ )			cifra dei milioni ( $1000^2$ )			cifra delle migliaia			cifra delle unità		
base 1000 (4 cifre)	<u>007</u>			<u>347</u>			<u>650</u>			<u>700</u>		
base 10			7	3	4	7	6	5	0	7	0	0

Architettura degli elaboratori
- 17 -
Aritmetica binaria






### Conversione tra una base e una sua potenza

- Esempio: convertiamo  $N = 5731_8$  in base 2
- $8 = 2^3$ , quindi ogni cifra di  $N$  in base 8 corrisponderà a 3 cifre di  $N$  in base 2.
  - $5_8 = 101_2$
  - $7_8 = 111_2$
  - $3_8 = 011_2$
  - $1_8 = 001_2$
- Quindi  $5731_8 = 101\ 111\ 011\ 001_2$

5			7			3			1		
1	0	1	1	1	1	0	1	1	0	0	1

Architettura degli elaboratori - 18 - Aritmetica binaria



### Conversione tra basi che sono una potenza dell'altra

- Esempio: convertiamo  $N = 100\ 000\ 010\ 011_2$  in base 8
- $8 = 2^3$ , quindi ogni gruppo da tre cifre di  $N$  in base 2 corrisponderà a una cifra di  $N$  in base 8.
  - $100_2 = 4_8$
  - $000_2 = 0_8$
  - $010_2 = 2_8$
  - $011_2 = 3_8$
- Quindi  $100\ 000\ 010\ 011_2 = 4023_8$

1	0	0	0	0	0	0	1	0	0	1	1
4			0			2			3		

Architettura degli elaboratori - 19 - Aritmetica binaria




## Rappresentazione ottale ed esadecimale

- La rappresentazione binaria è quella usata nei calcolatori, ma per noi umani è abbastanza scomoda da usare, perché numeri anche non grandissimi vengono rappresentati da stringhe di simboli piuttosto lunghe.
  - ▶ Ad es.  $1432_{10} = 10110011000_2$
- Pertanto si usano molto spesso le rappresentazioni in base 8 (ottale) e 16 (esadecimale)
  - ▶ Danno rappresentazioni più compatte
    - $1432_{10} = 2630_8$
  - ▶ Sono convertibili da e in binario immediatamente, grazie alla proprietà vista.



## La rappresentazione esadecimale

- Comporta un piccolo problema: noi disponiamo di 10 cifre, mentre ce ne servono sedici.
- Soluzione: per i valori da 10 a 15 si usano le lettere da A a F.
- Pertanto F5A è un numero esadecimale.
- Per evitare confusione (ADA è un nome, ma può essere anche un numero esadecimale; oppure: 139 potrebbe esprimere un numero in base 10 oppure 16) si usano delle **convenzioni (sintattiche)**
- Es: i numeri esadecimali sono precedute dal prefisso 0x
  - ▶ 0xF5A, 0xDA, sono numeri esadecimali.
  - ▶ (e anche 0xf5a e 0xda lo sono)
- Oppure (in alcuni linguaggi, come C e C++): i numeri che cominciano con la cifra 0 sono considerati in base 8
  - ▶ 065 significa «cinquantatrè» ( $6 \times 8 + 5$ )
  - ▶ 65 significa «sessantacinque» (base 10)



## Conversione da binario a esadecimale


- 16 è  $2^4$ , quindi per ottenere la rappresentazione esadecimale di un numero binario basta raggrupparne i bit a quattro a quattro.
- Esempi:

0	1	0	1	0	1	1	1	1	1	0	0	1	0	0	1
5				7				C				9			

3				F				B				6			
0	0	1	1	1	1	1	1	1	0	1	1	1	0	1	0

Architettura degli elaboratori
- 22 -
Aritmetica binaria



## Operazioni aritmetiche


- Tutte le notazioni posizionali utilizzano le stesse regole, indipendentemente dalla base di rappresentazione adottata.
- Le familiari regole della base 10 restano valide.

124 +
235 =
-----
359

358 +
754 =
-----
1112

11 ← riporti

Architettura degli elaboratori
- 23 -
Aritmetica binaria



### Alcune somme in base 2


---

		11 ←	
101 +	101 +		riporti
10 =	111 =		
-----	-----		
111	1100		

- In generale:
  - $0+0=0$
  - $0+1=1+0=1$
  - $1+1=0$  col riporto di 1

---

Architettura degli elaboratori - 24 - Aritmetica binaria



### Somme in base 16


---

- Valgono sempre le stesse regole

	1 ←	
1F +		riporto
35 =		
-----		
54		

---

Architettura degli elaboratori - 25 - Aritmetica binaria




### E per la sottrazione ...

- Valgono le stesse regole.
- Esempio in base 2, 16 e 10:

-1 -1		-1	-1
1011 0101 -	prestiti	B5-	181-
0110 1100 =		6C=	108=
-----		-----	-----
0100 1001		49	73

Diagram illustrating subtraction with borrowing (prestiti) in binary, hexadecimal (B5, 6C), and decimal (181, 108). Arrows show the borrowing process from higher bits to lower bits.

Architettura degli elaboratori - 26 - Aritmetica binaria



### Rappresentazione dei numeri interi senza segno nei calcolatori

- Con N bit si possono rappresentare tutti i numeri interi senza segno (positivi o nulli) compresi tra 0 e  $2^N - 1$ .
- N è solitamente 32 o 64 (anche se su processori più datati può essere 8 o 16).
  - N = 8: numeri da 0 a 255
  - N = 16: numeri da 0 a 65,535
  - N = 32: numeri da 0 a 4,294,967,295

Architettura degli elaboratori - 27 - Aritmetica binaria



## L'overflow

- Supponiamo di rappresentare gli interi su 16 bit.  
(se usassimo più bit il problema non cambierebbe).
- Supponiamo che A, B e C siano variabili intere contenenti rispettivamente i valori 30945, 54667 e 0.
- Eseguiamo l'operazione  $C = A+B$ : ci aspettiamo che C contenga il valore  $30945 + 54667 = 85612$ .
- Invece C contiene 20076. Perché?
- Perché 85612 è un numero troppo grosso per essere rappresentato (come intero) su 16 bit.
- In questi casi si suol dire che c'è stato un overflow, ovvero un traboccamento delle capacità dei registri (o delle celle di memoria).
- NB: per questo motivo nel linguaggio C le istruzioni  $T = A+B-C$ ; e  $T = B-C+A$ ; possono avere effetti diversi.



## Osservazione

- A causa dell'overflow, la manipolazione ideale dei numeri e quella reale dei calcolatori differiscono sensibilmente.
- Occorre fare attenzione.
- A livello assembler l'overflow è segnalato da un apposito flag.




## Prodotto. Esercizio concettuale

1. Ricordati come si fa la moltiplicazione fra numeri a più cifre (guarda sul tuo sussidiario delle elementary :-D )
  - ▶ si tratta di un *algoritmo*
  - ▶ ...che opera sulla *rappresentazione in base 10* dei due termini
  - ▶ ...e produce la *rappresentazione in base 10* del loro prodotto
  - ▶ ...tipicamente *implementato* con carta e penna
  - ▶ ...(e usa le «*tabelline*»)
2. Adatta questo algoritmo alla base 2
  - ▶ otterrai un algoritmo... più semplice
  - ▶ (pronto ad essere implementato in un circuito!)



## Tabelline (in 1ma elementare)

x	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9
2	0	2	4	6	8	10	12	14	16	18
3	0	3	6	9	12	15	18	21	24	27
4	0	4	8	12	16	20	24	28	32	36
5	0	5	10	15	20	25	30	35	40	45
6	0	6	12	18	24	30	36	42	48	54
7	0	7	14	21	28	35	42	49	56	63
8	0	8	16	24	32	40	48	56	64	72
9	0	9	18	27	36	45	54	63	72	81



In base 2, c'est plus facile


---

<b>x</b>	<b>0</b>	<b>1</b>
<b>0</b>	<b>0</b>	<b>0</b>
<b>1</b>	<b>0</b>	<b>1</b>

Architettura degli elaboratori

- 32 -

Aritmetica binaria



Prodotto: algoritmo delle scuole elementari  
(ma in base 2)

---

100011 × ← **A**

1011 = ← **B**

1 1 1 1 1 1

1 ×		100011	+	
1 ×		100011	+	
0 ×		000000	+	
1 ×		100011	=	
11000001				


← **A × B**

Architettura degli elaboratori

- 33 -

Blocchi funzionali combinatori





## Riassunto (so far)

---

- Per rappresentare numeri naturali con 0 e 1:  
facile, scriviamoli in base 2
  
- Abbiamo rivistito:
  - ▶ come convertire dalla familiare base 10 una base generica, e vicev.
  - ▶ case particolare: come convertire fra base 2 e base 8 o 16
  
- Tutti i comuni algoritmi per effettuare operazioni fra numeri scritti in base 10 sono generalizzabili a base qualunque
  - ▶ e quindi in base 2 (anzi diventano più semplici)
  - ▶ es.: somme, sottrazioni, prodotti...
  - ▶ caso particolare: prodotti / divisioni per potenze di 2 come «shift»
  
- Come rappresentare numeri interi generici? (cioè con segno)
  - ▶ «modulo e segno» non è molto conveniente...

Architettura degli elaboratori
- 34 -
Aritmetica binaria

Ci sono 10 tipi di studenti:

Quelli che conoscono l'aritmetica binaria  
e quelli che non la conoscono

Architettura degli elaboratori
- 35 -
Aritmetica binaria

Numeri con le mani...




Quanti numeri posso esprimere usando le due mani?



Z

numeri interi




## Da N a Z

---

- Abbiamo visto come rappresentare i numeri *naturali*,  $\mathbf{N} = \{0, 1, 2, \dots\}$ 
  - ▶ binario!
- Era estendiamo a numeri *interi*  $\mathbf{Z} = \{\dots -2, -1, 0, +1, +2, \dots\}$ 
  - ▶ (cioè con segno)
- In un elaboratore, questo pone alcuni problemi
- *Di rappresentazione:*
  - ▶ Come rappresentare il "segno meno"
- *Di esecuzione delle operazioni :*
  - ▶ Come eseguire le operazioni in modo efficiente  
(L'esecuzione delle operazioni aritmetiche è un'operazione frequentissima: non può essere inefficiente!)
- Prima soluzione: «modulo e segno»
  - ▶ spoiler: vedremo che non è molto conveniente...

Architettura degli elaboratori
- 39 -
Aritmetica binaria




## Rappresentazione in modulo e segno

---

- Il modo più semplice e intuitivo per rappresentare i numeri negativi consiste nel trattarli come si fa normalmente fuori dai calcolatori: usando un simbolo per il segno e altri simboli per il valore assoluto.
- Usa un bit (MSB Most Significant Bit – quello più a sx) per rappresentare esplicitamente il segno
  - ▶ 0 = + = numero positivo
  - ▶ 1 = - = numero negativo
- Usa gli altri bit disponibili per rappresentare il valore assoluto come numero binario puro

Architettura degli elaboratori
- 40 -
Aritmetica binaria




## Rappresentazione in modulo e segno

- Esempio (su 8 bit):
  - MSB (bit 7) = segno,
  - bit 6...bit 0 = valore assoluto

-2 → 10000010  
+5 → 0000101

- Nota: il segno è completamente disgiunto dal valore assoluto; la posizione del bit di segno è concettualmente irrilevante (si mette a sinistra per analogia con quanto facciamo di solito).

Architettura degli elaboratori - 41 - Aritmetica binaria



## Rappresentazione in modulo e segno: caratteristiche

- La rappresentazione dei numeri interi mediante il bit di segno ha dei grossi difetti pratici:
  - Il valore 0 ha due distinte rappresentazioni
    - 10000000 → - 0
    - 00000000 → + 0

(due sintassi diverse per una stessa semantica ☹ )

Es diventa più complicato realizzare l'operatore di uguaglianza («sono uguali questi due numeri?»):

- Maggior complessità realizzativa
- maggior costo
- non molto utilizzata.

Architettura degli elaboratori - 42 - Aritmetica binaria



## Rappresentazione in complemento a 2


- Il bit più significativo di una stringa di  $n$  bit ha peso  $-2^{n-1}$  anziché  $+2^{n-1}$  (tutti gli altri bit mantengono il peso come in binario puro)
- Il valore di un intero  $v$  espresso in questa notazione è dato dalla formula:  

$$v = d_0 * B^0 + B^1 * d_1 + B^2 * d_2 + \dots + B^{n-2} * d_{n-2} - B^{n-1} * d_{n-1}$$
  - ▶ dove  $d_k$  ( $k=0..n-1$ ) sono le cifre binarie della rappresentazione del numero.
- La formula differisce da quella usata per i numeri naturali solo per il peso negativo del MSB.
- Esempio (su 8 bit):
  - ▶ il bit 7 ha peso -128 anziché +128
  - ▶ 11010101 denota il valore:  $-128 + 64 + 16 + 4 + 1 = -43$
  - ▶ 01000111 denota il valore:  $64 + 4 + 2 + 1 = +71$



## Rappresentazione in complemento a 2: caratteristiche generali

- La rappresentazione non è completamente posizionale, ma in parte sì.
- Guardando il bit più significativo (MSB) si capisce se il numero è positivo (o nullo) o negativo.
- Lo zero ha un'unica rappresentazione (costituita da tutti bit nulli).



### Rappresentazione in complemento a 2 di un numero positivo dato

- Rappresentiamo su k bit il numero N:
- Se  $N \geq 0$  basta assegnare al MSB il valore 0 e codificare N sui rimanenti k-1 bit.  
NB: questo implica che sono rappresentabili i numeri fino a  $2^{k-1}-1$
- Esempio su 8 bit (k=8):  
 $N = +96$ ;  $Cp_2(+96) = 01100000$   
 Prova:  $01100000_2 = (64+32)_{10} = 96_{10}$

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


base 2 naturale

128	64	32	16	8	4	2	1
-----	----	----	----	---	---	---	---

complemento a 2

-128	64	32	16	8	4	2	1
------	----	----	----	---	---	---	---

Architettura degli elaboratori - 45 - Aritmetica binaria



### Rappresentazione in complemento a 2 di un numero negativo dato

- Un numero negativo rappresentato in complemento a due comincia sempre per 1
  - Di posizione con valore negativo ce n'è una sola: se in corrispondenza c'è uno zero, il numero non può essere negativo.

X

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

naturale

128	64	32	16	8	4	2	1
-----	----	----	----	---	---	---	---


$N = 128 + X$

compl. a 2

-128	64	32	16	8	4	2	1
------	----	----	----	---	---	---	---

$N = -128 + X$


Architettura degli elaboratori - 46 - Aritmetica binaria



### Rappresentazione in complemento a 2 di un numero negativo dato

- Dato  $N \geq 0$ , vogliamo rappresentare  $-N$  in complemento a 2 mediante una stringa  $S$  di 8 bit.
- La stringa  $S$  interpretata come numero in compl. a 2 vale  $-128+X = -N$ 
  - Quindi,  $X=128-N$
- La stringa  $S$  interpretata come numero naturale vale  $128+X$ , cioè  $128+128-N = 256-N$
- In conclusione, la stringa  $S$  che denota  $-N$  in complemento a 2 è la stessa che denota  $256-N$  in binario.
- Esempio:
  - $-N = -56$
  - $256 - 56 = 200 = 128+64+8 \rightarrow 11001000$
  - $Cp_2(-56) = 11001000$
  - Prova: l'interpretazione in compl. a 2 di 11001000 è  $-128+64+8 = -56$

Architettura degli elaboratori
- 47 -
Aritmetica binaria



### Rappresentazione in complemento a 2 di un numero negativo dato

$-N = -74$   
 $-128+X = -74$   
 $X = 128-74 = 54 = 0110110$   
 $256-74 = 182$

	X							
	┌───────────────────┐							
	1	0	1	1	0	1	1	0
naturale	128	64	32	16	8	4	2	1
	N=128+54=182							
compl. a 2	-128	64	32	16	8	4	2	1
	N=-128+54=-74							

Architettura degli elaboratori
- 48 -
Aritmetica binaria