Rappresentazione dell'informazione

Prof. Alberto Borghese
Dipartimento di Informatica
alberto.borghese@unimi.it

Università degli Studi di Milano

Riferimenti al testo: Paragrafi 2.4, 2.9, 3.1, 3.2, 3.5 (codifica IEEE754)

Sommario

Rappresentazione binaria dell'Informazione

Sistema di numerazione binario

Conversione in e da un numero binario

Operazioni elementari su numeri binari: somma, sottrazioni.

I numeri decimali

Proprietà di potenze e logaritmi

$2^K \times 2^M = 2^{(K+M)}$
$2^3 \times 2^2 = 2^{(3+2)} = 32$

$$2^{K^M} = 2^{K*M} = 2^{M^K}$$

$$2^{-K} = \frac{1}{2^K}$$

Il logaritmo è l'operazione inversa dell'elevamento a potenza:

- $N = 2^M \Leftrightarrow M = \log_2(N)$
- $32 = 2^5 \Leftrightarrow 5 = \log_2(32)$

Se ho M cifre, dove ciascuna cifra può assumere 2 valori, il numero totale di combinazioni sarà 2^M

$\log_2(2^M) = M$

$$\log_2 K = -\log_2\left(\frac{1}{K}\right)$$

$$\log_2 KM = \log_2 K + \log_2 M$$

$5 = \log_2 (4 \times 8) = \log_2 4 + \log_2 8 = 2 + 3$

Il linguaggio

Per farsi capire da un calcolatore, occorre parlare la sua stessa lingua.

Semantica (significato dei simboli -> codifica).
Sintassi.

Rappresentazione dell'informazione

Non solo conteggio, ma anche enumerazione di oggetti….

Noi possiamo rappresentare gli oggetti tramite parole composte a paretire da un alfabeto di simboli: A,B,…,Z,0,1,…,9,…

• Diversi alfabeti possono essere usati per rappresentare gli stessi oggetti.

• I simboli degli alfabeti possono assumere diverse forme: segni su carta, livelli di tensione, fori su carta, segnali di fumo.

• La scelta della rappresentazione è in genere vincolata al tipo di utilizzo ed al tipo di operazioni che devono essere fatte sulle informazioni stesse

Alfabeto binario

I computer memorizzano ed elaborano le informazioni sotto forma di bit (Binary Digit – Cifra Binaria)

• Un bit è **l'unità di informazione base** e può assumere due valori: – vero o falso – acceso o spento – …………

• Rappresentazione binaria.

• Il linguaggio di base mediante il quale ogni informazione deve essere codificata ha un alfabeto di 2 soli simboli (0 e 1)

Codifica binaria

Per poter rappresentare un numero maggiore di informazioni è necessario utilizzare sequenze di bit.

Il processo secondo cui si fa corrispondere ad un'informazione una configurazione di bit prende il nome di codifica dell'informazione

Quanti oggetti diversi possiamo rappresentare con parole binarie di k bit?

• Con una parola di 1 bit rappresentiamo 2 oggetti (1 bit ha due possibili valori).

• Supponiamo di avere parole di k bit. Quanti oggetti riescono a rappresentare?

2^k

Esempio di codifica binaria

• Quanti oggetti diversi possiamo rappresentare con parole binarie di 3 bit?

0	000	A
1	001	B
2	010	C
3	011	D
4	100	E
5	101	F
6	110	G
7	111	H

Codifica dei caratteri alfanumerici

Quanti bit devono avere le parole binarie usate per identificare i 26 caratteri diversi dell'alfabeto inglese (es: A,B,…,Z)?

$2^4 < 26 < 2^5$

Quanti bit devono avere le parole binarie usate per identificare 26+26 oggetti diversi (es: A,B,…,Z, a,b,. .... z)?

$2^5 < 52 < 2^6$

Quanti bit servono per 100 oggetti?

ceil [$\log_2 100$]

0		32		64	@	96	`	128	Ç	160	á	192	└	224	α
1	☺	33	!	65	A	97	a	129	ü	161	í	193	┴	225	ß
2	☻	34	"	66	B	98	b	130	é	162	ó	194	┬	226	Γ
3	♥	35	#	67	C	99	c	131	â	163	ú	195	├	227	π
4	♦	36	$	68	D	100	d	132	ä	164	ñ	196	─	228	Σ
5	♣	37	%	69	E	101	e	133	à	165	Ñ	197	┼	229	σ
6	♠	38	&	70	F	102	f	134	å	166	ª	198	╞	230	µ
7	•	39	'	71	G	103	g	135	ç	167	º	199	╟	231	τ
8	◘	40	(	72	H	104	h	136	ê	168	¿	200	╚	232	Φ
9	○	41	)	73	I	105	i	137	ë	169	⌐	201	╔	233	Θ
10	◙	42	*	74	J	106	j	138	è	170	¬	202	╩	234	Ω
11	♂	43	+	75	K	107	k	139	ï	171	½	203	╦	235	δ
12	♀	44	,	76	L	108	l	140	î	172	¼	204	╠	236	∞
13	♪	45	-	77	M	109	m	141	ì	173	¡	205	═	237	φ
14	♫	46	.	78	N	110	n	142	Ä	174	«	206	╬	238	ε
15	☼	47	/	79	O	111	o	143	Å	175	»	207	╧	239	∩
16	►	48	0	80	P	112	p	144	É	176	░	208	╨	240	≡
17	◄	49	1	81	Q	113	q	145	æ	177	▒	209	╤	241	±
18	↕	50	2	82	R	114	r	146	Æ	178	▓	210	╥	242	≥
19	‼	51	3	83	S	115	s	147	ô	179	│	211	╙	243	≤
20	¶	52	4	84	T	116	t	148	ö	180	┤	212	╘	244	⌠
21	§	53	5	85	U	117	u	149	ò	181	╡	213	╒	245	⌡
22	▬	54	6	86	V	118	v	150	û	182	╢	214	╓	246	÷
23	↨	55	7	87	W	119	w	151	ù	183	╖	215	╫	247	≈
24	↑	56	8	88	X	120	x	152	ÿ	184	╕	216	╪	248	°
25	↓	57	9	89	Y	121	y	153	Ö	185	╣	217	┘	249	∙
26	→	58	:	90	Z	122	z	154	Ü	186	║	218	┌	250	·
27	←	59	;	91	[	123	{	155	¢	187	╗	219	█	251	√
28	∟	60	<	92	\	124	¦	156	£	188	╝	220	▄	252	ⁿ
29	↔	61	=	93	]	125	}	157	¥	189	╜	221	▌	253	²
30	▲	62	>	94	^	126	~	158	₧	190	╛	222	▐	254	■
31	▼	63	?	95	_	127	⌂	159	ƒ	191	┐	223	▀	255	

Il codice ASCII la rappresentazione dell'informazione alfanumerica

- 8 bit
- 0-31 codici di controllo.
- 128-255 extended ASCII

L'UNICODE

http://www.unicode.org. Codifica su 8, 16, 32 bit alfabeti diversi.

Latin	Malayalam	Tagbanwa	General Punctuation
Greek	Sinhala	Khmer	Spacing Modifier Letters
Cyrillic	Thai	Mongolian	Currency Symbols
Armenian	Lao	Limbu	Combining Diacritical Marks
Hebrew	Tibetan	Tai Le	Combining Marks for Symbols
Arabic	Myanmar	Kangxi Radicals	Superscripts and Subscripts
Syriac	Georgian	Hiragana	Number Forms
Thaana	Hangul Jamo	Katakana	Mathematical Operators
Devanagari	Ethiopic	Bopomofo	Mathematical Alphanumeric Symbols
Bengali	Cherokee	Kanbun	Braille Patterns
Gurmukhi	Unified Canadian Aboriginal Syllabic	Shavian	Optical Character Recognition
Gujarati	Ogham	Osmanya	Byzantine Musical Symbols
Oriya	Runic	Cypriot Syllabary	Musical Symbols
Tamil	Tagalog	Tai Xuan Jing Symbols	Arrows
Telugu	Hanunoo	Yijing Hexagram Symbols	Box Drawing
Kannada	Buhid	Aegean Numbers	Geometric Shapes

Osservazioni sulla codifica binaria

Il linguaggio di un elaboratore elettronico è fatto di due segnali: **on** e **off**, rappresentati dai simboli **1** e **0 (alfabeto binario).**

• Sia le istruzioni che i dati sono rappresentati da ***parole (word)*** di numeri binari. Sequenze di bit trattate come un unicum nell'elaboratore.

• Un alfabeto binario non limita le funzionalità di un elaboratore a patto di avere parole di lunghezza sufficiente.

• Una stringa binaria non ha significato semantico di per sé.

• 00000011 00101000 11010000 00100000 rappresenta 4 caratteri alfanumerici: ♥ (╨ spazio.
• 00000011 00101000 11010000 00100000 rappresenta un'istruzione di addizione in MIPS su 32 bit (add $k0, $t0, $t9).
• 00000011001010001101000000100000 rappresenta un numero intero decimale 53,006,368.

Codifica negli elaboratori

Non esiste una semantica nella codifica.

Un elaboratore elabora stringhe di simboli binari.

Diverse elaborazioni producono risultati diversi:
- print su schermo (associa ai byte, delle matrici di pixel)
- La Unità di controllo associa alla parola i comandi e i dati delle istruzioni (configura la CPU).
- La ALU esegue la somma sulle parole che rappresentano i numeri.

Ci occupiamo oggi di codifica numerica.

Sommario

Rappresentazione binaria dell'Informazione

Sistema di numerazione binario

Conversione in e da un numero binario

Operazioni elementari su numeri binari: somma, sottrazioni.

I numeri decimali

Numerazione Simbolica

Sistema di numerazione mediante simboli (numerazione romana: I, V, X, L, C, M) il cui valore non dipende dalla posizione: e.g. XXXI = 31, XI = 11....

Sistema di numerazione posizionale (decimale): **cifra + peso.**
Il peso è la base elevata alla posizione della cifra.

1 ha un valore diverso nelle due scritture:
100
1000

Numerazione Posizionale

Alfabeto della numerazione:
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9} numerazione araba decimale.
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F} numerazione esadecimale.
{0, 1} numerazione binaria.

Sistemi di numerazione binario, ottale ed esadecimale.

Conversioni decimale -> binario e viceversa.

Numeri

Matematica — Informatica

Numeri naturali

- Risoluzione 1 unità — unsigned
- Contano quantità tangibili

- $0 \leq n < +oo$

Numeri relativi

- Risoluzione 1 unità — int, integer
- Contano quantità tangibili e il loro complemento.

- $0 \leq n < +oo$

Numeri decimali

- Risoluzione dipende — real (ma non sono reali!)
- Numeri con la virgola — float

Codifica posizionale di un numero intero

Fondata sul concetto di **base**: B = [b_0, b_1, b_2, b_3,.....].

Ciasun elemento, N, può essere rappresentato come combinazione lineare degli elementi della base: $N = \sum_k c_k b_k = \sum_k c_k 10^k$

Esempi:

- $764_{10} = 7x10^2 + 6x10^1 + 4x10^0$ — $b_k = B^k = 10^k$
- $12_{10} = 1x10^1 + 2x10^0$ — $b_k = B^k = 10^k$
- $100_2 = 1x2^2 + 0x2^1 + 0x2^0 = 4_{10}$ — $b_k = B^k = 2^k$

$k \geq 0$

Codifica posizionale di un numero decimale

Fondata sul concetto di **base**: B = [b_0, b_1, b_2, b_3,.....].

Ciasun elemento, N, può essere rappresentato come combinazione lineare degli elementi della base: $N = \sum_k c_k b_k = \sum_k c_k 10^k$

Esempi:

- $764,3_{10} = 7x10^2 + 6x10^1 + 4x10^0 + 3x10^{-1}$ $\quad b_k = B^k = 10^k$
- $12,21_{10} = 1x10^1 + 2x10^0 + 2x10^{-1} + 1x10^{-2}$ $\quad b_k = B^k = 10^k$
- $100,11_2 = 1x2^2 + 0x2^1 + 0x2^0 + 1x2^{-1} + 1x2^{-2} = 4,75_{10}$ $\quad b_k = B^k = 2^k$

-oo < k < +oo

Tassonomia ed unità di misura

Hertz - numero di ciclo al secondo nei moti periodici (clock).

- MIPS - Milioni di istruzioni per secondo.
- MFLOPS - Milioni di istruzioni in virgola mobile (FLOating point) al secondo.

Prefissi:

k - chilo (mille:10^3).
M - mega (un milione:10^6).
G - giga (un miliardo: 10^9).
T – tera (1000 miliardi: 10^{12})
P – peta (1,000,000 miliardi: 10^{15})

m - milli (un millesimo: 10^{-3})
µ - micro (un milionesimo: 10^{-6})
n - nano (un miliardesimo: 10^{-9})
p – pico (un millesimo di miliardo: 10^{-12})
f – femto (un milionesimo di miliardo: 10^{-15})

Terminologia

Bit = binary digit.

- 1 byte = 8 bit.
- 1kbyte = 2^{10}byte = 1,024 byte
- 1Mbyte = 2^{20}byte = 1,048,576 byte.
- 1Gbyte =2^{30}byte = 1,073,741,824 byte.
- 1Tbyte =2^{40}byte = 1,099,511,627,776 byte.

- Parola (word) numero di bit trattati come un unicum dall'elaboratore.
- Le parole oggi arrivano facilmente a 64bit (8 byte).

Approssimazione

Multipli del bit						
Prefissi SI			Prefissi binari			
Nome	Simbolo	Multipli	Nome	Simbolo	Multipli	
kilobit	kbit	10^3	**kibibit**	Kibit	2^{10}	1024 bit
megabit	Mbit	10^6	mebibit	Mibit	2^{20}	1 024 Kib
gigabit	Gbit	10^9	gibibit	Gibit	2^{30}	1 048 576 Kib = 1 gibibit
terabit	Tbit	10^{12}	tebibit	Tibit	2^{40}	1 024 Gbit
petabit	Pbit	10^{15}	pebibit	Pibit	2^{50}	1 024 Tbit
exabit	Ebit	10^{18}	exbibit	Eibit	2^{60}	1 024 Pbit
zettabit	Zbit	10^{21}	zebibit	Zibit	2^{70}	1 024 Ebit
yottabit	Ybit	10^{24}	yobibit	Yibit	2^{80}	1 024 Zbit

Sommario

Rappresentazione binaria dell'Informazione

Sistema di numerazione binario

Conversione in e da un numero binario

Operazioni elementari su numeri binari (somma, sottrazione e moltiplicazione intera).

I numeri decimali

Conversione da base 2 a base 10

Un numero $N=[c_0, c_1, c_2, c_3,...]$ in base 10, $B = [b_0, b_1, b_2, b_3...]$ si trasforma in base n, R = $[r_0, r_1, r_2, r_3...]$, facendo riferimento alla formula:

$$N = \sum_k c_k b_k = \sum_{k=0}^{N-1} d_k r^k$$

• ciascuna cifra k-esima viene moltiplicata per la base corrispondente: $r_k = n^k$.

• i valori così ottenuti sono sommati per ottenere il numero in notazione decimale.

101 1101 0101$_{\mathbf{due}}$ = $1x2^{10} + 0x2^9 + 1x2^8 + 1x2^7 + 1x2^6 + 0x2^5 + 1x2^4 + 0x2^3 + 1x2^2 + 0x2^1 + 1x2^0 =$
1024 + 256 + 128 + 64 + 16 + 4 +1 = 1493

“Spelling” di un numero

$$1493 = 3x1 \quad + 9x10 \; + 4x100 + 1x1000$$
$$= 3x10^0 + 9x10^1 + 4x10^2 + 1x10^3$$

Vogliamo rappresentare $1493_{\mathbf{dieci}}$

Unità **1493 = 10 x 149 + 3** ← Cifra meno significativa

Decine **(10x) 149 = 10 x 14 + 9**

Centinaia **(100x) 14 = 10 x 1 + 4**

Migliaia **(1000x) 1 = 10 x 0 + 1** ← Cifra più significativa

“estrazione” delle cifre decimali

$$1493 = 3x1 \quad + 9x10 \; + 4x100 + 1x1000$$
$$= 3x10^0 + 9x10^1 + 4x10^2 + 1x10^3$$

Vogliamo estrarre le cifre di $1493_{\mathbf{dieci}}$. Porto le cifre alla destra della virgola:

1493 / 10 = 149,3 → **149 decine** → **3 unità (resto)**

“estrazione” delle cifre decimali

$149 = 9x10^0 + 4x10^1 + 1x10^2$ decine

149 / 10 =14,9 → 14 centinaia → 9 decine (resto)

$14 = 4x10^0 + 1x10^1$ centinaia

14 / 10 = 1,4 → 1 migliaia → 4 centinaia (resto)

“estrazione” delle cifre decimali

1493 = 3x1 + 9x10 + 4x100 + 1x1000

$= 3x10^0 + 9x10^1 + 4x10^2 + 1x10^3$

Vogliamo estrarre le cifre di 1493_{dieci}. Porto le cifre alla destra della virgola:

1493 / 10 = 149,3 → esamino 149 → 3 unità
149 / 10 =14,9 → esamino 14 → 9 decine
14 / 10 = 1,4 → esamino 1 → 4 centinaia
1 / 10 = 0,1 → termina → 1 migliaia

Le cifre sono il resto della divisione ricorsiva del numero.

Meccanismo di "estrazione"

Vogliamo estrarre le cifre di $1493_{\mathbf{dieci}}$. Porto le cifre alla destra della virgola.

Utilizzo la divisione intera per la base 10, il resto rappresenta la cifra decimale meno significativa.

1493 / 10 = 149   con R = 3 → 3 unità

149 / 10 =14   con R = 9 → 9 decine

14 / 10 = 1   con R = 4 → 4 centinaia

1 / 10 = 0   con R = 1 → 1 migliaia

Termina perchè non è rimasto nulla del numero.

Conversione base 10 -> base 2
"estrazione" delle cifre binarie

Vogliamo rappresentare $1493_{\mathbf{dieci}}$ in binario: $\mathbf{10111010101}_{\mathbf{due}}$

1493 / 2 = 746 + 1 ← Bit meno significativo (LSB – Least Significant Bit)
746 / 2 = 373 + 0   **373*2 + 0 = 746**
373 / 2 = 186 + 1   **186*2 + 1 = 373 => (186*2+1)*2 = 373**
186 / 2 = 93 + 0
93 / 2 = 46 + 1
46 / 2 = 23 + 0
23 / 2 = 11 + 1
11 / 2 = 5 + 1
5 / 2 = 2 + 1
2 / 2 = 1 + 0
1 / 2 = 0 + 1 ← Bit più significativo (MSB – Most Significant Bit)

Perché funziona?

Prendiamo il numero N0 e dividiamo per 2.
Se N0 è pari il resto, R0, sarà 0, altrimenti sarà 1. Infatti:
int (N0 / 2) * 2 + N0 = E0.

Chiamiamo N1 = int(N0 / 2) e procediamo.

Prendiamo E1 e dividiamo per 2. Chiamiamo N2 = int(N1 / 2). Se N1 è pari il resto, R1, sarà 0, altrimenti sarà 1. N1 = int (N1 / 2) * 2 + R1

Chiamiamo N2 = int(E1 / 2) e procediamo.

Prendiamo N2 e dividiamo per 2. Se N2 è pari il resto, R2, sarà 0, altrimenti sarà 1. N2 = int (N2 / 2) * 2 + R2.

Si procede fino a quando Ni non è < 2 (=1).

N0 = int [N0 / 2] * 2 + R0.
N0 = int [N1] * 2 + R0.
N0 = int [int (N1 / 2) * 2 + R1] * 2 + R0.
N0 = int [int(N2) * 2 + R1] * 2 + R0.
N0 = int [int[int(N2 / 2) * 2 +R2] * 2 + R1] * 2 + R0.

Esempio:
$N0 = 5_{10} = ?_2$
R0 = resto(5/2) = 1

N1 = quoz(5/2) = 2

R1 = resto(2/2) = 0

N2 = quoz(2/2) = 1

R2 = resto(1/2) = 0

$N0 = R0 + R1 * 2 + R2 * 2^2$
= 1 0 1

int(N2 / 2) = 0

Conversione base 10 -> base n: algoritmo

Un numero *x* in base 10 si trasforma in base *n* usando il seguente procedimento:

• Dividere il numero *x* per *n*
• Il resto della divisione è la cifra di posto 0 in base *n*
• Il quoziente della divisione è a sua volta diviso per *n*
• Il resto ottenuto a questo passo è la cifra di posto 1 in base *n*

• Si prosegue con le divisioni dei quozienti ottenuti al passo precedente fino a che l'ultimo quoziente è 0.

• l'ultimo resto è la cifra più significativa in base *n*

Esercizi

Dati i numeri decimali 23456, 89765, 67489, 121331, 2453, 111010101

- si trasformino in base 3
- si trasformino in base 7
- si trasformino in base 2

- Dati i numeri 23456_7, 121331_5, 2453_8, 111010101_2 convertire ciascun numero in decimale e in binario

Codifica esadecimale

Il codice esadecimale viene utilizzato come **forma compatta per rappresentare numeri binari**:

- 16 simboli: 0,1,…,9,A,B,…,F

- Diverse notazioni equivalenti:

0x9F

$9F_{16}$

9Fhex

$0x9F = 9x16^1 + 15x16^0 = 159_{10}$

Cifre esadecimali	Valori decimali	Equivalenti binari
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Valori posizionali esadecimali	Valori decimali
16^{-3}	1/4096=0.000 244 140 625
16^{-2}	1/256=0.003 906 25
16^{-1}	1/16=0.062 5
16^0	1
16^1	16
16^2	256
16^3	4 096
16^4	65 536
16^5	1 048 576

Conversione esadecimale -> binario

Vogliamo rappresentare 9Fhex in binario. E' semplice.

• Ogni simbolo viene convertito in un numero binario di 4 cifre:
9**hex** --> 1001_{due}
F**hex** --> 1111_{due}
9F**hex** --> 10011111_{due}

• È sufficiente ricordarsi come si rappresentano in binario i numeri decimali da 0 a 15 (o derivarli)

Conversione binario -> esadecimale

Da binario ad esadecimale si procede in modo analogo:

•Ogni gruppo di 4 cifre viene tradotto nel simbolo corrispondente:

Esempio: convertire $01101011_{\mathbf{due}}$ in esadecimale:
$1011_{\mathbf{due}}$ --> $B_{\mathbf{hex}}$
$0110_{\mathbf{due}}$ --> $6_{\mathbf{hex}}$
$0110\ 1011_{\mathbf{due}}$ --> $6B_{\mathbf{hex}}$

00000011001010001101000000100000 - add $k0, $t0, $t9

0x0328d020

Codifica dei numeri relativi (interi) su N bit

Occorre coprire tutti gli N bit a disposizione. Codifica su 16 bit:

Numeri naturali: $11_{10} = 1011_2$ = 0000 0000 0000 1011

Inserisco 0 fino a coprire tutti i bit; gli zeri sono parte integrante del numero

Numeri relativi: $+5_{10} = 0101_2$ = 0000 0000 0000 0101 = $+5_{10}$

Numeri relativi: $-5_{10} = 1011_2$ = 1111 1111 1111 1011 = -5_{10}

Replico il primo bit, quello del segno

Nota bene:

1000 0000 0000 1011 sarebbe sbagliato = $-16{,}395_{10}$

0000 0000 0000 1011 sarebbe sbagliato = 11_{10}

Sommario

Sistema di numerazione binario

Rappresentazione binaria dell'Informazione

Conversione in e da un numero binario

Operazioni elementari su numeri binari: somma, sottrazione

I numeri decimali

Somma

111 ← Riporto

 1011 +
  110 =

10001

11 + 6 = 17 in decimale

Vorrei definire solo l'operazione di somma e non utilizzare la sottrazione

Numeri negativi: complemento a 1

I numeri negativi sono complementari ai numeri positivi: a + (-a) = 0

Codifica in complemento a 1: il numero negativo si ottiene cambiando 0 con 1 e viceversa.

Problema:

000	0
001	1
010	2
011	3
100	-3
101	-2
110	-1
111	0

Doppia codifica per lo 0.

Doppia negazione riporta il numero al numero positive (scambio 2 volte gli 0 con 1 e gli 1 con 0).

Numeri negativi: complemento a 2

I numeri negativi sono complementari ai numeri positivi: a + (-a) = 0

Codifica in complemento a 2: il numero negativo si ottiene cambiando 0 con 1 e viceversa, e sommando 1.

000	0	
001	1	negativo: 110 + 1 = 111 = -1
010	2	negativo: 101 + 1 = 110 = -2
011	3	negativo: 100 + 1 = 101 = -3
100	-4	
101	-3	
110	-2	
111	-1	

$(x_{31} \text{ x } -2^{31}) + (x_{30} \text{ x } 2^{30}) + (x_{29} \text{ x } 2^{29}) + \dots + (x_1 \text{ x } 2^1) + (x_0 \text{ x } 2^0)$

NB La prima cifra è il bit di segno.

Perché complemento a 2?

- La rappresentazione in complemento a due deve il suo nome alla proprietà in base alla quale la somma senza segno di un numero di *n* bit e del suo complemento è pari a 2^n (peso del bit n+1)

7 + (-7) =	0111 + 1001 = 10000	2^4
5 + (-5) =	0101 + 1011 = 10000	2^4

- e quindi il complemento (o negazione) di un numero *x* in complemento a due è pari a $2^n - x$, ovvero il suo complemento a 2.

$2^4 - 7 = 1001$

$2^4 - 5 = 1011$

Doppia negazione

I numeri negativi sono complementari ai numeri positivi: a + (-a) = 0

Segue che **-(-a) = +a**

Codifica in complemento a 2: il numero negativo si ottiene cambiando 0 con 1 e viceversa, e sommando 1.

000	0	
001	1	10 + 1 = 11
010	2	
110	-2	
111	-1	

Esempio:
$-(-2)_{10} = +2_{10}$

$-(010)_2$ => Complemento a 1 => 010 => Sommo 1 (complemento a 2) => 110_2
$-(110)_2 = 2_{10}$ Complemento a 1 => 001 => Sommo 1 (complemento a 2) => 010_2 c.v.d.

Sottrazione

Sommo i seguenti 2 numeri 11 + (-13):
$01011_2 = 11_{10}$
$10011_2 = -13_{10}$

E' equivalente ad effettuare la differenza: 11 – 13

00110
01011 +
10011 =

11110 ➔ -2_{10}

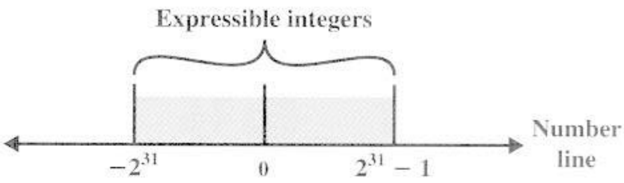
$+13_{10} = 01101_2$ => complemento a 1 => 10010 + 1 = $10011_2 = -13_{10}$

Capacità di rappresentazione Numeri Interi (relativi)

Interi con segno su N bit. Range: $-2^{N-1} \leq n \leq 2^{N-1} - 1$.

Esempio: Visual C++. Intero è su 4byte (word di 32 bit):

$$-2^{31} = -2.147.483.650 \leq n \leq 2.147.483.649 = 2^{31} - 1$$



(a) Twos complement integers

Risoluzione della codifica: 1 unità

Sommario

Sistema di numerazione binario

Rappresentazione binaria dell’Informazione

Conversione in e da un numero binario

Operazioni elementari su numeri binari: somma, sottrazione.

I numeri decimali

Codifica IEEE754 dei numeri reali anche in forma denormalizzata.

Conversione base 10 -> base n: algoritmo

Un numero *x.y* in base 10 si trasforma in base *n* usando il seguente procedimento.

Per la parte intera, *x,* si applica l'algoritmo visto in precedenza:

• Dividere il numero *x* per *n*
• Il resto della divisione è la cifra di posto 0 in base *n*
• Il quoziente della divisione è a sua volta diviso per *n*
• Il resto ottenuto a questo passo è la cifra di posto 1 in base *n*

• Si prosegue con le divisioni dei quozienti ottenuti al passo precedente fino a che l'ultimo quoziente è 0.

• l'ultimo resto è la cifra più significativa in base *n*

«Estrazione» delle cifre decimali contenute dopo la virgola

Vogliamo estrarre le cifre di 0,3672$_{\mathbf{dieci}}$. Porto le cifre alla **sinistra** della virgola:

0,3672 * 10 = 3,672 → esamino 0,672 → 3 decimi
0,672 * 10 =6,72 → esamino 0,72 → 6 centesimi
0,72 * 10 = 7,2 → esamino 0,2 → 7 millesimi
0,2 * 10 = 2,0 → termina → 2 decimillesimi

Conversione base 10 -> base 2

"estrazione" delle cifre binarie dopo la virgola

Vogliamo rappresentare $0{,}625_{\mathbf{dieci}}$ in binario: $\mathbf{0{,}101}_{\mathbf{due}}$

0,625 * 2 = 1,250 = 1 + 0,250 => 1
0,250 * 2 = 0,500 = 0 + 0,5 => 0
0,500 * 2 = 1,000 = 1 + 0,0 => 1
0,0000

$1*2^{-1} + 0*2^{-2} + 1*2^{-3} = 1/2 + 1/8 = 0{,}5 + 0{,}125 = 0{,}625$

Conversione base 10 -> base n: algoritmo per la parte frazionaria

Un numero *x,y* in base 10 si trasforma in base *n* usando il seguente procedimento.

Per la parte frazionaria, *y*:

• Moltiplicare il numero *y* per *n*
• La prima cifra del risultato coincide con la cifra di posto 1 dopo la virgola.
• Si elimina la parte intera ottenuta e si considera la nuova parte frazionaria.
• La parte frazionaria ottenuta viene moltiplicata per la base *n*.
• La prima cifra del risultato coincide con la cifra di posto 2 dopo la virgola.
• Si prosegue con le moltiplicazioni della parte frazionaria fino a quando non diventa 0 o non si esaurisce la capacità di rappresentazione.

Sommario

Sistema di numerazione binario

Rappresentazione binaria dell'Informazione

Conversione in e da un numero binario

Operazioni elementari su numeri binari: somma, sottrazione.

I numeri decimali