Codifica 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 interi: somma, sottrazioni.

I numeri decimali

Il linguaggio

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

Il linguaggio è costituito da:
- Semantica (significato delle parole)
- Sintassi (organizzazione delle parole)

Cosa rappresentano le parole elaborate da un processore?

Con quali regole vengono concatenate?

Rappresentazione dell’informazione

Le parole di un linguaggio naturale hanno, in modo convenzionale, una semantica.

Le parole di un linguaggio possono essere rappresentate mediante sequenze di simboli. Questi simboli possono essere associati ai suoni e sono le lettere dell’**alfabeto** nel linguaggio naturale.

L’alfabeto italiano è costituito da 21 lettere (simboli): A, B, C ……Z.

Le stringhe di lettere dell’alfabeto rappresentano le parole, e.g. «t a v o l o» e sono associate alla semantica delle parole (a degli oggetti precisi).

• Diversi alfabeti possono essere usati per rappresentare gli stessi oggetti (alfabeto cinese, giapponese, indiano, latino…).

• I simboli degli alfabeti possono assumere diverse forme: segni su carta (lettere scritte), suoni (lettere pronunciate), livelli di tensione, fori su carta (schede perforate), segnali di fumo.

• La scelta della rappresentazione è in genere vincolata al tipo di utilizzo, di mezzo di comunicazione utilizzato, e al tipo di operazioni che devono essere fatte sul contenuto informativo.

Alfabeto binario

L'alfabeto italiano è composto da 21 simboli: A, B, C ….. Z.

L'alfabeto dei calcolatori è costituito da 2 simboli, che rappresentiamo come: 0, 1 (bit – binary digit).

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

Gruppi di 8 bit vengono chiamati **Byte**.

I gruppi di bit più frequentemente utilizzati da un elaboratore prende il nome di **Parola (word):** parole a 32 / 64 bit.

10110101 – un Byte

1000 1011 0101 0011 1100 0100 0001 – una parola a 32 bit

Osservazioni sulla codifica binaria

• 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.

Le parole degli elaboratori sono sequenze di bit. **Non hanno un significato univoco come nel linguaggio naturale.**

Una stringa binaria non ha significato semantico di per sé.

• 00000011 00101000 11010000 00100000 rappresenta 4 caratteri alfanumerici: («End_of_Text»; '('; 'D'; ' ').

• 00000011 00101000 11010000 00100000 rappresenta un'istruzione di addizione in MIPS su 32 bit (add $k0, $t0, $t9).

• 00000011001010001101000000100000 rappresenta un numero intero: 53,006,368

….

Codifica dei caratteri alfanumerici

Come facciamo a trasformare un linguaggio in un altro? Con una traduzione!

Come facciamo a trasformare il linguaggio naturale in un linguaggio binario?

Consideriamo un testo scritto e un insieme di caratteri (un alfabeto) costituito da:
- Caratteri alfabetici minuscoli / maiuscoli
- Caratteri speciali: è, è, ò, à, ù…
- Caratteri di controllo (a capo, spaziatura di una linea, cancella carattere….)
- Segni di interpunzione: |, ?, ^….

- Come rappresentiamo tutti i simboli di questo alfabeto con solo 2 simboli?

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 (American Standard Code for Information Interchange)

- •8 bit
- •0-31 codici di controllo.
- •32-127 standard ASCII
- •128-255 extended ASCII

Prima versione è del 1963

- Metto in corrispondenza gruppi di 8 caratteri binariy (Byte) con ogni simbolo dell’alfabeto del linguaggio naturale.

Esempio: “Casa” sarà scritta come: 01000011 01100001 01110011 01100001 – 67 97 115 97
Un elaboratore può quindi leggere questi 32 bit e comporre la parola «casa» sullo schermo.

Codifica e decodifica

Input: tastiera
« casa»

32 bit

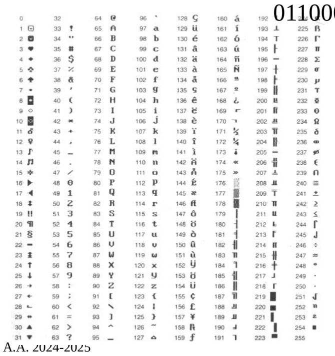
01000011
01100001
01110011
01100001

Processore

32 bit

01000011
01100001
01110011
01100001

Output: schermo
« casa»



Lo schermo è costituito da pixel (e.g. 1980 x 1024)

Ciascun pixel può mostrare un colore

Il Byte è trasformato in un pattern di accensione dei pixel che mostra il carattere a schermo.

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

Sommario

Rappresentazione binaria dell'Informazione

Sistema di numerazione binario

Conversione in e da un numero binario

Operazioni elementari su numeri binari interi: somma, sottrazioni

I numeri decimali

Codifica binaria

Per poter rappresentare un numero maggiore di informazioni (rispetto a vero/falso) è necessario utilizzare sequenze di bit.

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

Codifica implicita (associamo a una configurazione di bit, un oggetto di un insieme, e.g. una lettera dell'alfabeto)

Codifica esplicita (codifica numerica)

Esempio di codifica binaria implicita

Dato un insieme di elementi, identifico i diversi elementi con un numero d'ordine.

0	000	A = tigre
1	001	B = cane
2	010	C = topo
3	011	D = elefante
4	100	E = gatto
5	101	F = cervo
6	110	G = gazzella
7	111	H = rinoceronte

Quanti bit servono per rappresentare un'informazione?

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?

Sistemi di numerazione

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

Sistema di numerazione esplicito
Sistema di numerazione posizionale (decimale): {**cifra, peso}.**
Il peso è costituito dalla base elevata alla posizione della cifra.

1 ha un valore diverso in queste due scritture:

100	1 = un centinaio
1000	1 = un migliaio

Dipende dalla **posizione** dell'1.

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.

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 (**k ≥ 0**): $N = \sum_k c_k b_k \quad = \sum_k c_k 10^k$

Esempi:

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

k ≥ 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:

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

-oo < k < +oo

Codifica posizionale di un numero binario

Ciasun elemento, N, può essere rappresentato come combinazione lineare degli elementi della base:

$$N = \sum_k c_k b_k = \sum_k c_k \mathbf{2}^k$$

Esempio:

- $1011{,}01_2 = 1x2^3 + 0x2^2 + 1x2^1 + 1x2^0 + 0x2^{-1} + 1x2^{-2}$ $\qquad b_k = B^k = 2^k$

-oo < k < +oo

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

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

Esempio di codifica binaria su basi diverse

In decimale i caratteri utilizzati per rappresentare i numeri sono: {0, 1, 2, 3, 4, ….9}.
In binario si utilizzano sempre i simboli: {0,1} in esadecimale: {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E}

Decimale	Binario	Esadecimale
d_1d_0	$b_3b_2b_1b_0$	h_0
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Conversione binaria - esadecimale

$0111\ 1011\ 0011\ 1010_2$ = 0x7B3A

$FF01_{16} = 1111\ 1111\ 0000\ 0001_2$

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

Proprietà di potenze e logaritmi

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

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

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

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

$$\log_2(2^M) = M$$

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

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

$$5 = \log_2 (4 \text{ x } 8) = \log_2 4 + \log_2 8 = 2 + 3$$

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.	
- $oo < n < +oo$	
Numeri decimali	
- Risoluzione dipendente dalla codifica	real (ma non sono reali!)
- Numeri con la virgola	float

Tassonomia ed unità di misura

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})

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.

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 64 bit (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

Base 10 Base 2

Sommario

Rappresentazione binaria dell'Informazione

Sistema di numerazione binario

Conversione in e da un numero binario

Operazioni elementari su numeri binari interi: somma, sottrazione

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 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.

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

Base binaria:

$\mathbf{101\ 1101\ 0101}_{\mathbf{due}} = 1x2^{10} + 0x2^9 + 1x2^8 + 1x2^7 + 1x2^6 + 0x2^5 +$
$1x2^4 + 0x2^3 + 1x2^2 + 0x2^1 + 1x2^0 =$
$1024 + 256 + 128 + 64 + 16 + 4 + 1 = \mathbf{1493}_{\mathbf{dieci}} =$
$1x10^3 + 4x10^2 + 9x10^1 + 3x10^0$

“Spelling” di un numero

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

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

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

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

“estrazione” delle cifre decimali

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

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

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

$$N = \sum_k c_k b_k$$

1493/10 = 10^{-1} x $(3x10^0 + 9x10^1 + 4x10^2 + 1x10^3)$ =
$(3x10^{-1} + 9x10^0 + 4x10^1 + 1x10^2)$

“estrazione” delle cifre decimali

Vogliamo estrarre le cifre di 1493_{dieci}. Porto le cifre alla destra della virgola: $149 = 9x10^0 + 4x10^1 + 1x10^2$ decine + 3 unità

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

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

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

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.

Termina quando non è rimasto nulla del numero.

Conversione base 10 -> base 2

"estrazione" delle cifre binarie

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

Resto

1493 / 2 = 746 R = 1 ← Bit meno significativo (LSB – Least Significant Bit)

746 / 2 = 373 R = 0 373*2 + 0 = 746

373 / 2 = 186 R = 1 186*2 + 1 = 373 => (186*2+1)*2 = 373

186 / 2 = 93 R = 0

93 / 2 = 46 R = 1

46 / 2 = 23 R = 0

23 / 2 = 11 R = 1

11 / 2 = 5 R = 1

5 / 2 = 2 R = 1

2 / 2 = 1 R = 0

1 / 2 = 0 R = 1 ← Bit più significativo (MSB – Most Significant Bit)

$$N = \sum_k c_k b_k$$

Conversione base 10 -> base n: algoritmo

Un numero *x* in base 10 si trasforma in base *n, intera,* utilizzando 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

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** --> $1001\ 1111_{due}$

- 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

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

1110 ← Riporto 101

01011 + 1473 +

00110 = 719 =

-------- --------

10001 2192

11 + 6 = 17 in decimale

Il bit di somma vale sempre 0/1 Il bit di somma vale da 0 a 9

Il riporto vale sempre 0/1

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: Doppia codifica per lo 0.

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

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

2 – 2 = 0 => 2 + (-2) = 0

010 – 010 = 0 ? 010 + 101 = 111 = 000

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	

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*esimo*, fuori dalla capacità di rappresentazione.

Per numeri codificati su *4 bit* + 1 bit di segno:

7 + (-7) = 00111 + 11001 = 10000 = 2^4

5 + (-5) = 00101 + 11011 = 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^n.

Per numeri codificati su 4 bit + 1 bit di segno:

2^4 = 16 – 7 = 10000 + 11001 = (1)1001 = 9_{10}

2^4 = 16 – 5 = 10000 + 11011 = (1)1011 = 11_{10}

Sottrazione

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

E' equivalente ad effettuare la differenza: 11 – 13

$11_{10} = 001100_2$
$13_{10} = 001101_2$
$-13_{10} = 110011_2$

00110
01011 +
10011 =

11110 ➔ -2_{10}

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

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

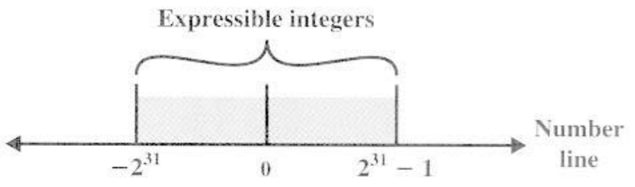
Nota bene. Queste rappresentazioni del numero -5_{10} sono sbagliate:
1000 0000 0000 1011 sarebbe sbagliato = $-16{,}395_{10}$
0000 0000 0000 1011 sarebbe sbagliato = $+11_{10}$

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

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 $0{,}3672_{\text{dieci}}$. 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**

$$N = \sum_k c_k b_k$$

K è negativo per le cifre dopo la virgola (la parte frazionaria del numero)

Conversione base 10 -> base 2

“estrazione” delle cifre binarie dopo la virgola

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

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.

Conversione base 10 -> base 2

"estrazione" delle cifre binarie dopo la virgola

Vogliamo rappresentare $0{,}626_{\mathbf{dieci}}$ in binario: $\mathbf{0{,}1010000001\ldots}_{\mathbf{due}}$

0,626 * 2 = 1,252 = 1 + 0,252 => 1
0,252 * 2 = 0,504 = 0 + 0,504 => 0
0,504 * 2 = 1,008 = 1 + 0,008 => 1
0,008 * 2 = 0,016 = 0 + 0,016 => 0
0,016 * 2 = 0,032 = 0 + 0,032 => 0
0,032 * 2 = 0,064 = 0 + 0,064 => 0
0,064 * 2 = 0,128 = 0 + 0,128 => 0
0,128 * 2 = 0,256 = 0 + 0,256 => 0
0,256 * 2 = 0,512 = 0 + 0,512 => 0
0,512 * 2 = 1,024 = 1 + 0,024 => 1
0,024 * 2 …….

$$N = \sum_k c_k b_k$$

$1 \cdot 2^{-1} + 1 \cdot 2^{-3} + 1 \cdot 2^{-10} = 1/2 + 1/8 + 1/1024 = 0{,}5 + 0{,}125 + 0{,}0009765625 = 0{,}6259765625$

$$\lim_{k\to\infty}\left(\sum_k c_k\, 2^{-k}\right) = 0{,}626$$

Errore di approssimazione $< 2^{-10}$

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