Rappresentazione dell'Informazione
Codifica Binaria

Corso di

Architettura dei Sistemi a Microprocessore

Luigi Coppolino

Dipartimento di Ingegneria

Università degli Studi di Napoli "Parthenope"

Rappresentazione dell'Informazione

- Tutta **l'informazione** in un calcolatore è rappresentata tramite numeri
- Questi numeri sono espressi come **sequenze di 0 e 1**
- I calcolatori **elaborano l'informazione** effettuando **operazioni su numeri**

- I sistemi di comunicazione **scambiano informazioni spostando sequenze di numeri**

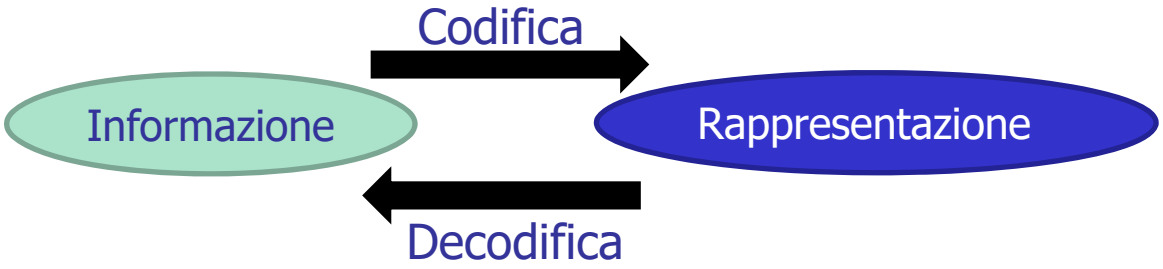
Problema

- Abbiamo informazioni (numeri, testi, immagini, suoni. . .) che vogliamo rappresentare (e poter elaborare) in un calcolatore.

- Vincolo: per motivi tecnologici un calcolatore lavora solo con i valori 0 e 1

Codifica e Decodifica

- Il processo che permette di ottenere la rappresentazione delle informazioni



- Il processo inverso è invece la decodifica
- La codifica è una convenzione!
- E´ il modo in cui associamo un'informazione ad una sua rappresentazione, ad esempio quella binaria.

Rappresentazione dell'Informazione

- I computer sono sistemi digitali perché lavorano con digit (0 e 1)
- **Bit:** valore 0 (tensione bassa), valore 1 (tensione alta)
- **Byte**: sequenza di 8 bit. Esempio: 01110010
- L'informazione viene creata ed acceduta in unità di informazione dette **word**. Una word è creata come un multiplo di byte.
- Lunghezze di una word: 8 (byte), 16, 32, 64, 128 bit.
- Un word a k bit può contenere 2^k valori (da 0 a 2^{k-1}).

Numeri Naturali

- Numerazione araba: dieci cifre (0..9), **notazione posizionale**: il valore di una cifra dipende dalla sua posizione.
- La cifra più a sinistra ha un valore maggiore

- Ad esempio: 312, le tre cifre hanno valore diverso a seconda della posizione che occupano.

- La numerazione romana è **non posizionale** ma additiva. Ad esempio: MCCCX.

Numeri Naturali

- **Sistemi di numerazione posizionale** in base **p**:

$$N_p = a_n \times p^n + a_{n-1} \times p^{n-1} + \ldots + a_1 \times p^1 + a_0 = \sum_{i=0}^{n} a_i \times p^i$$

- Se **p=10** si avrà:

$$745 = 7 \times 10^2 + 4 \times 10^1 + 5 \times 10^0$$

- Se **p=4** si avrà:

$$213 = 2\text{x}4^2 + 1\text{x}4^1 + 3\text{x}4^2$$

Numeri Binari, Ottali, Esadecimali

- Sistema **Binario**
  - p=2
  - Alfabeto: {0,1}

$$10101101 = (1x2^7+0x2^6+1x2^5+0x2^4+1x2^3+1x2^2+0x2^1+1x2^0)$$
$$= (128 + 32 + 8 + 4 + 1) = 173_{10}$$

$$01101010 = (0x2^7+1x2^6+1x2^5+0x2^4+1x2^3+0x2^2+1x2^1+0x2^0)$$
$$= (64 + 32 + 8 + 2) = 106_{10}$$

$$00101111 = (0x2^7+0x2^6+1x2^5+0x2^4+1x2^3+1x2^2+1x2^1+1x2^0)$$
$$= (32 + 8 + 4 + 2 + 1) = 47_{10}$$

Numeri Binari, Ottali, Esadecimali

- Sistema **Ottale**
  - p=8
  - Alfabeto: {0,1,2,3,4,5,6,7}

$$254_8 = 2x8^2 + 5x8^1 + 4x8^0 = 128 + 40 + 4 = 172_{10}$$

- Sistema **Esadecimale**
  - p=16
  - Alfabeto={0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}

$$B7F_{16} = 11x16^2 + 7x16^1 + 15x16^0 = 2943_{10}$$

Numeri Binari, Ottali, Esadecimali

decimale	binario	ottale	esadecimale
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10

Conversione da base 2 a base 10

- Si divide il numero per due e si prende il resto
- L'elenco dei resti si legge dall'alto verso il basso

173 : 2 = 86 → 1
86 : 2 = 43 → 0
43 : 2 = 21 → 1
21 : 2 = 10 → 1
10 : 2 = 5 → 0
5 : 2 = 2 → 1
2 : 2 = 1 → 0
1 : 2 = 0 → 1 → 10101101

Conversione da base 2 a base 8 e 16

- La conversione da base 2 a base 8:
  - Si divide il numero in **triple** e ogni tripla si traduce nella cifra ottale (01.101.100 → 154)
- La conversione da base 2 a base 16:
  - Si divide il numero in **quadruple** e ogni quadrupla si traduce nella cifra esadecimale (0011.1011 → 3B).

Esercizio

- Convertire in base 10 e in base 16 i seguenti numeri:
  - 1101010
  - 1110001
  - 1010101110
- Convertire in base 2 i seguenti numeri in base 10:
  - 139
  - 255
  - 180

Somma tra Numeri Naturali in base 2

- Somma tra numeri naturali binari:

$$\begin{aligned} 0+0&=0 \quad \text{riporto} = 0 \\ 0+1&=1 \quad \text{riporto} = 0 \\ 1+0&=1 \quad \text{riporto} = 0 \\ 1+1&=0 \quad \text{riporto} = 1 \end{aligned}$$

- Es. 1101+111

riporto	1	1	1	1	
		1	1	0	1 +
		0	1	1	1 =
somma	1	0	1	0	0

Rappresentazione dei Numeri Naturali

- Con *n* bit si possono rappresentare 2^n diversi numeri naturali. Quali saranno?

- Dalla codifica utilizzata si deduce che i numeri rappresentabili appartengono all'intervallo:

$$[0\,,2^n-1]$$

MSB e LSB

- Dato un numero binario, si definisce **Most Significant Bit (MSB)** il bit che ha il **valore più grande**.
- Il bit più significativo è indicato alle volte come il *"bit più a sinistra"* nelle architetture "**Big-Endian**"
- In tali architetture vige la convenzione di scrivere le cifre più significative a sinistra.

MSB e LSB

- Dato un numero binario, si definisce **Least Significant Bit (LSB)** il bit che ha il **valore più piccolo**.
- Il bit più significativo è indicato alle volte come il *"bit più a destra"* nelle architetture "**Big-Endian**"

Rappresentazione di Interi: il Segno

- Per gli **interi con segno**, il bit più significativo (**MSB**) rappresenta il segno:

  0 positivo

  1 negativo

- Esistono tre modi di rappresentare un numero con segno:
  - **Modulo e segno**
  - **Complementi a 1**
  - **Complementi a 2**

Rappresentazione Modulo e Segno

- Rappresentazione in **modulo** e **segno**
  - Il MSB indica il segno: 1 = negativo, 0 = positivo
  - I restanti bit indicano il modulo
  - Esempio: 1101 = −5, 0111 = +7
  - notazione intuitiva ma...
  - scomoda per operazioni aritmetiche, lo zero ha due rappresentazioni!
- In questa codifica, con n bit si rappresentano i valori nell'intervallo: $[-2^{n-1}+1, 2^{n-1}-1]$

Rappresentazione di Interi in Complemento a 2

- Serve una rappresentazione che faciliti lo svolgimento delle operazioni:
  - Rappresentazione in complemento a 2
    - Dati n bit, un numero positivo N è rappresentato in modo standard (come abbiamo visto per i naturali)
    - $-N$, invece si rappresenta come $2^n - N$
  - Metodo operativo per rappresentare $-N$:
    - Rappresentare il modulo N in modo standard
    - Complementare tutti i bit ($1 \rightarrow 0$, $0 \rightarrow 1$)
    - Sommare 1

Rappresentazione di Interi in Complemento a 2

- In questa codifica, con n bit si rappresentano i valori nell'intervallo: $[-2^{n-1}, 2^{n-1} - 1]$
- Ad es. Con 3 bit rappresentiamo I numeri (-4,3)

decimale	binario in C2	decimale	binario in C2
-4	100	0	000
-3	101	$+1$	001
-2	110	$+2$	010
-1	111	$+3$	011

- Il primo bit indica ancora il segno
- Lo zero ha una sola codifica

Esercizio

- Convertire in complemento a 2 i seguenti numeri:
  - 12, -12, -8, -101, -54

Rappresentazione di Interi in Complemento a 2

- Come passare da complemento a 2 a base 10
- Algoritmo inverso:
  - Sottrarre 1
  - Complementare a 1
  - Convertire da binario a decimale e aggiungere il segno
- Metodo facilitato di verifica:
  - Convertire in decimale con l'algoritmo standard assegnando al bit più significativo valore negativo
  - Esempio: $10100 = -1 \times 2^4 + 1 \times 2^2 = -12$

Somma tra Numeri Interi in base 2

- Somma tra numeri naturali interi con segno:
  - Rappresentare i numeri in complemento a 2
  - Effettuare la somma in modo standard
  - Non considerare l'eventuale riporto sul bit di segno
- Esempio: 60-54=60+(-54)

riporto	1	1	1	1				
		0	1	1	1	1	0	0 +
		1	0	0	1	0	1	0 =
somma	(1)	0	0	0	0	1	1	0

Overflow

- Sommiamo due numeri in complemento a 2 rappresentati con *n* bit, quindi appartenenti a $[-2^{n-1}, 2^{n-1} - 1]$
- Può succedere che il risultato cada al di fuori dell'intervallo
- In altre parole: n bit non bastano per rappresentare il risultato! $\rightarrow$ **OVERFLOW**
- Come riconoscerlo?
  - può succedere solo quando si sommano due operandi dello stesso segno: **se il segno del risultato è diverso da quello degli operandi** è avvenuto un overflow!
  - gli ultimi due riporti sono diversi tra loro (01 o 10)

Overflow

- Esempio

riporto	1	0			
		1	0	0	+
		1	0	1	=
somma	(1)	0	0	1	

Rappresentazione Numeri Frazionari

- Numeri Frazionari: compresi tra 0 e 1

$$(0.a_{-1}a_{-2}\ldots a_{-n})_b = a_{-1} \times b^{-1} + a_{-2} \times b^{-2} + \ldots + a_{-n} \times b^{-n}$$

Esempio:

$$(0.587)_{10} = 5 \times 10^{-1} + 8 \times 10^{-2} + 7 \times 10^{-3}$$

- Conversione da base 2 a base 10

$$(0.1011)_2 = 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4} = (0.6875)_{10}$$

Rappresentazione Numeri Razionali

- Rappresentazione in **virgola mobile**
- Un numero razionale N è espresso come: $N = m \times 10^e$
  - *m* è la **mantissa**
  - *e* è **l'esponente**

Rappresentazione dei Caratteri

- Tre possibili rappresentazioni:
  - **ASCII standard**: un carattere è rappresentato con 7 bit (ASCII = American Standard Code for Information Interchange)
  - **ASCII estesa**: un carattere è rappresentato con 8 bit
  - **UNICODE** : un carattere è rappresentato con 16 bit (MS Windows ne usa una simile)

Rappresentazione dei Caratteri

American Standard Code for Information Interchange (ASCII) utilizza codici a 7-bit

Esempi:

carattere	rappresentazione
A	1 0 0 0 0 0 1
7	0 1 1 0 1 1 1
+	0 1 0 1 0 1 1

Byte	Cod.	Char
00000000	0	Null
00000001	1	Start of heading
00000010	2	Start of text
00000011	3	End of text
00000100	4	End of transmit
00000101	5	Enquiry
00000110	6	Acknowledge
00000111	7	Audible bell
00001000	8	Backspace
00001001	9	Horizontal tab
00001010	10	Line feed
00001011	11	Vertical tab
00001100	12	Form Feed
00001101	13	Carriage return
00001110	14	Shift out
00001111	15	Shift in
00010000	16	Data link escape
00010001	17	Device control 1
00010010	18	Device control 2
00010011	19	Device control 3
00010100	20	Device control 4
00010101	21	Neg. acknowledge
00010110	22	Synchronous idle
00010111	23	End trans. block
00011000	24	Cancel
00011001	25	End of medium
00011010	26	Substitution
00011011	27	Escape
00011100	28	File separator
00011101	29	Group separator
00011110	30	Record Separator
00011111	31	Unit separator

Byte	Cod.	Char
00100000	32	Spc
00100001	33	!
00100010	34	"
00100011	35	#
00100100	36	$
00100101	37	%
00100110	38	&
00100111	39	'
00101000	40	(
00101001	41	)
00101010	42	*
00101011	43	+
00101100	44	,
00101101	45	-
00101110	46	.
00101111	47	/
00110000	48	0
00110001	49	1
00110010	50	2
00110011	51	3
00110100	52	4
00110101	53	5
00110110	54	6
00110111	55	7
00111000	56	8
00111001	57	9
00111010	58	:
00111011	59	;
00111100	60	<
00111101	61	=
00111110	62	>
00111111	63	?

Byte	Cod.	Char
01000000	64	@
01000001	65	A
01000010	66	B
01000011	67	C
01000100	68	D
01000101	69	E
01000110	70	F
01000111	71	G
01001000	72	H
01001001	73	I
01001010	74	J
01001011	75	K
01001100	76	L
01001101	77	M
01001110	78	N
01001111	79	O
01010000	80	P
01010001	81	Q
01010010	82	R
01010011	83	S
01010100	84	T
01010101	85	U
01010110	86	V
01010111	87	W
01011000	88	X
01011001	89	Y
01011010	90	Z
01011011	91	[
01011100	92	\
01011101	93	]
01011110	94	^
01011111	95	_

Byte	Cod.	Char
01100000	96	`
01100001	97	a
01100010	98	b
01100011	99	c
01100100	100	d
01100101	101	e
01100110	102	f
01100111	103	g
01101000	104	h
01101001	105	i
01101010	106	j
01101011	107	k
01101100	108	l
01101101	109	m
01101110	110	n
01101111	111	o
01110000	112	p
01110001	113	q
01110010	114	r
01110011	115	s
01110100	116	t
01110101	117	u
01110110	118	v
01110111	119	w
01111000	120	x
01111001	121	y
01111010	122	z
01111011	123	{
01111100	124	|
01111101	125	}
01111110	126	~
01111111	127	Del

Concetto di Codifica

- **Domanda**: Cosa rappresenta la stringa 01000011?

- **Risposta**:
  - Se interpretata come un numero naturale è 67
  - Se interpretata come carattere è C. Dipende!

- **La codifica è una convenzione!**