

La codifica dell'informazione

```

000101010100011010101000011010100001100010111
111000001111110101010010010010100111010101010
0010101010000110000111110000011111101010100
01110101010001101010101000010110101000110001011
1111000010101011110101000010010100111010101010
0010101010000110000111111000010111101010100
00010010101001101010100001011010100001100001011
1111000001111110101010010010010100111010101010
0010101010000110000111111000011111101010100
00101010000011000011111111111111111101010100
01111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111
0010101010000110000111111000010111101010100
0001001010100110101010000111010100001100001011
1111000001111110101010010010100111010101010
0010101010000110000111111000011111101010100
0111111111111111111111111111111111111111111
00010010101001101010100001011010100001100001011
1111000001111110101010010010010100111010101010
0010101010000110000111111000011111101010100
01110101000110000101101010100001011010101000111
111100000111111010101000010010100111010101010
0010101010000110000111111000011111101010100
00010010101001101010100001011010100001100001011
1111000001111110101010010010010100111010101010
0010101010000110000111111000011111101010100
01110101000110000101101010100001011010101000111
111100000111111010101000010010100111010101010
00101010100001100001111110000110111101010100

```

(continua)

Codifica dei numeri

- Il codice ASCII consente di codificare le cifre decimali da "0" a "9" fornendo in questo modo un metodo per la rappresentazione dei numeri
- Il numero 324 potrebbe essere rappresentato dalla sequenza di byte: 00110011 00110010 00110100

3	2	4
---	---	---
- Questa rappresentazione non è efficiente e, soprattutto, non è adatta per eseguire le operazioni aritmetiche sui numeri

Codifica dei numeri

- Sono stati pertanto studiati codici alternativi per
 - rappresentare i numeri in modo efficiente
 - eseguire le usuali operazioni

Codifica dei numeri: il sistema decimale

- Sistema **posizionale** in cui ogni cifra di un numero assume un valore che dipende dalla sua posizione

$$365 = 3 \times 100 + 6 \times 10 + 5 \times 1$$

$$365 = 3 \times 10^2 + 6 \times 10^1 + 5 \times 10^0$$

Si deve fare la somma dei prodotti di ciascuna cifra moltiplicata per la base elevata all'esponente che rappresenta la posizione della cifra stessa (partendo da 0).



Codifica dei numeri

- La notazione posizionale può essere usata con qualunque **base** creando così sistemi di numerazione diversi
- Per ogni sistema di numerazione si usa un numero di cifre uguale alla base
- Esempi**
 - Nella base 10 si usano le cifre 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
 - In base 5 si usano le cifre 0, 1, 2, 3, 4
 - In base 3 si usano le cifre 0, 1, 2

Codifica dei numeri

In informatica si usano prevalentemente le numerazioni binaria (base 2), ottale (base 8) ed esadecimale (base 16)



Sistema binario

- Utilizza una notazione posizionale basata su 2 cifre (0 e 1) e sulle potenze di 2

Esempio: $10011 =$

$$1x2^4 + 0x2^3 + 0x2^2 + 1x2^1 + 1x2^0 = 19$$

- Esempio: $20011 = ????$

Sistema ottale

- Utilizza una notazione posizionale basata su 8 cifre (0,1, ..., 7) e sulle potenze di 8

Esempio: $10011 =$

$$1x8^4 + 0x8^3 + 0x8^2 + 1x8^1 + 1x8^0 = 4105$$

Esempio: $3057 =$

$$3x8^3 + 0x8^2 + 5x8^1 + 7x8^0 = 1583$$

Sistema esadecimale

- Utilizza una notazione posizionale basata su 16 cifre (0,1,2,...,9,A,B,C,D,E,F) e sulle potenze di 16

Esempio: $10011 =$

$$1x16^4 + 0x16^3 + 0x16^2 + 1x16^1 + 1x16^0 = 65553$$

Esempio: $AAC3 =$

$$10x16^3 + 10x16^2 + 12x16^1 + 3x16^0 = 43715$$

Sistemi di numerazione

- Per evitare ambiguità si può scrivere esplicitamente la base di un numero
- Avremo

$$10011_2 < 10011_8 < 10011_{10} < 10011_{16}$$

Conversione da base 10 a base 2

- Per convertire un numero M in base 2 si devono trovare i resti delle divisioni per due di M e dei quozienti successivi

- Esempio: 210_{10}

210	2	resto 0
105	2	resto 1
52	2	resto 0
26	2	resto 0
13	2	resto 1
6	2	resto 0
3	2	resto 1
1	2	resto 1
0		

Conversione da base 10 a base 2

- Leggendo la sequenza dal basso verso l'alto si ottiene il numero

$$11010010_2$$

- Per una corretta verifica basta riconvertire il risultato alla base 10

Per le altre basi il procedimento è lo stesso, cambiando il divisore

Conversione da base 10 a base 8

- Per convertire un numero in base 8 si devono trovare i resti delle divisioni successive del numero per la base 8

- Esempio: 741_{10}

741	8	resto 5	↑	= 1345 ₈
92	8	resto 4		
11	8	resto 3		
1	8	resto 1		
0				

Conversione da base 10 a base 16

- Per convertire un numero in base 16 si devono trovare i resti delle divisioni successive del numero per la base 16

- Esempio: 41660_{10}

41660	16	resto 12 = C	↑	= A2BC ₁₆
2603	16	resto 11 = B		
162	16	resto = 2		
10	16	resto 10 = A		
0				

Conversione da base 2 a base 8 e base 16

- È semplice!
- Da base 2 a base 8: basta prendere i bit a **gruppi di 3** e trovare il numero ottale corrispondente (che sarà compreso tra 0 e 7)
- Da base 2 a base 16: basta prendere i bit a **gruppi di 4** e trovare il numero esadecimale corrispondente, (che sarà compreso tra 0 e F)

Rappresentazione dei numeri

- I numeri vengono distinti in tre categorie
 - Interi positivi
 - Interi con segno (positivi e negativi)
 - Reali (positivi e negativi con virgola)
- Ogni categoria viene rappresentata in modo diverso

NB: indipendentemente dalla rappresentazione scelta, essa sarà di tipo finito e consentirà di rappresentare solo un sottoinsieme finito di numeri!

Codici a lunghezza fissa

- Se si usa un numero **prestabilito** di cifre si ha un codice a **lunghezza fissa**
- In questo modo si pone anche un **limite** al numero massimo rappresentabile
- Esempio: qual è il numero più grande rappresentabile con 4 cifre?

in base 10	9999	
in base 2	1111	(= 15 ₁₀)
in base 16	FFFF	(= 65535 ₁₀)
in base 8	7777	(= 4095 ₁₀)

Codici a lunghezza fissa

- Numeri maggiori di quello **massimo** rappresentabile causano problemi di **overflow**
(ovvero per essere rappresentati richiedono più cifre di quelle a disposizione)
- Esempio: 4 cifre

in base 10	9999	+ 1 =	10000 ₁₀
in base 2	1111	+ 1 =	10000 ₂ (= 16 ₁₀)
In base 16	FFFF	+ 1 =	10000 ₁₆ (= 65536 ₁₀)
in base 8	7777	+ 1 =	10000 ₈ (= 4096 ₁₀)

Numeri interi positivi

- In generale, con **N** cifre a disposizione e base **b** il più grande numero (intero positivo) rappresentabile si può esprimere come

$$b^N - 1$$

- Esempio: $N=4$
 - in base 10 $9999 = 10^4 - 1$
 - in base 2 $1111 = 2^4 - 1$
 - in base 16 $FFFF = 16^4 - 1$
 - in base 8 $7777 = 8^4 - 1$

Operazioni

- Vediamo solo il caso della addizione nella codifica binaria
 - ✓ si mettono in colonna i numeri da sommare
 - ✓ si calcola il riporto ogni volta che la somma parziale supera il valore 1

0 + 0 = 0	con riporto 0
0 + 1 = 1	con riporto 0
1 + 0 = 1	con riporto 0
1 + 1 = 0	con riporto 1

Codici a lunghezza variabile

- Possiamo rappresentare i numeri usando un **numero variabile di cifre** (che dipende dal valore che si vuole rappresentare) introducendo un **simbolo speciale** che indica dove termina la rappresentazione di un numero e inizia quella del numero successivo

- Esempio

1001#11#1 codice a lungh. variabile, # separatore
 100100110001 codice a lungh. fissa, 4 bit per pixel

Codici a espansione

- Permettono di definire dei codici a **lunghezza variabile** senza far uso del carattere di separazione
 - ✓ Si stabiliscono dei formati di rappresentazione che fanno uso di un numero diverso di cifre
 - ✓ Si definiscono delle regole per distinguere tra i vari formati

Codici a formati multipli predefiniti

- ✓ 4 formati di rappresentazione a 4, 7, 10, 16 bit
- ✓ identificati da 00, 01, 10, 11

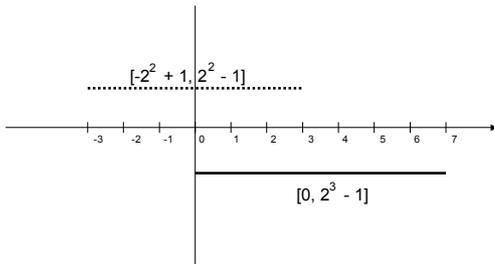
0	0000	36	1000000000
1	0001	37	1000000001
2	0010	38	1000000010
3	0011	39	1000000011
4	010000
5	010001	291	1011111111
6	010010	292	1100000000000000
7	010011	293	1100000000000001
8	0100100	294	1100000000000010
...
...	...	16674	1101111111111111
35	0111111	16675	1111111111111111

Codici a espansione con cifra di espansione

- ✓ 4 formati di rappresentazione a 4, 7, 10, 16 bit
- ✓ identificati da 0, 10, 110, 111

0	0000	40	1100000000
1	0001	41	1100000001
2	0010	42	1100000010
3	0011
...	...	167	1101111111
7	0111	168	1110000000000000
8	100000	169	1110000000000001
9	100001	170	1110000000000010
10	100010
11	100011
...	...	8358	1110111111111111
39	1011111	8359	1111111111111111

Numeri interi con segno



N=3, codice a lungh. fissa

Numeri interi con segno

- Si può pensare di usare un bit per il **segno**
 - ✓ "0" identifica "+"
 - ✓ "1" identifica "-"
- Gli altri bit vengono usati per codificare il **valore assoluto (modulo)** del numero

Numeri interi con segno

- Problemi
 - ✓ Il numero 0 ha due rappresentazioni (-0 = 100...0 e +0 = 000...0)
 - ✓ Per l'operazione di somma si deve tener conto dei segni degli addendi
- Sono state pertanto introdotte altre tecniche di codifica
 - ✓ Complemento a 1
 - ✓ Complemento a 2

Complemento a 1

- La cifra più a sinistra indica il segno
- Per i **numeri positivi** (cifra più a sinistra "0") si codifica il valore assoluto del numero
- Per i **numeri negativi** (cifra più a sinistra "1") si devono **negare** tutte le cifre (dove c'è "1" mettere "0", dove c'è "0" mettere "1")
- *Esempi*
 - ✓ N=8, +24 = 00011000, -24 = 11100111
 - ✓ Per trasformare 10011111 si devono complementare i suoi bit e trovare il valore corrispondente a 1100000, ovvero 96. Il numero cercato è -96

NB: ci vuole un codice a **lunghezza fissa**, altrimenti il complemento a 1 non funziona!

Complemento a 1

- L'operazione di somma funziona nel caso di **addendi positivi e addendi con segno opposto con risultato negativo**
- Negli altri casi (somma tra numeri negativi oppure somma tra addendi discordanti e con risultato positivo) il risultato è decrementato di 1
- Per risolvere questo problema è stata introdotta una nuova codifica: **complemento a 2**
- Verrà spiegata nel corso di Architetture degli Elaboratori insieme ad altre codifiche 😊

La rappresentazione dell'informazione

indipendentemente dall'informazione di partenza si ottiene sempre una sequenza di bit

```

000100101010011101010100001010101000011100010111
11110000111111101010001001010100111010101010
001011010100001110001011111000011111101010100
0111011010001101010100001001010100110001011
1111000010101011110101000001001010001110101010
001011010100011100010111110000011011110101010
00010010101001110101010000101010100001110001011
11110000111111010101001001010100111010101010
1111000011111101010100001001010100111010101010
1111000010101011110101000010010101001110101010
0010110101000111000101111100000110111101010100
000100101010011101010100001011000001100001111
1111000011111101010100001001010100111010101010
0010110101000111000101111100000110111101010100
011101010100111010101000010101010011100010111
11110000101010111101010000010010101001110101010
0010110101000111000101111100000110111101010100
    
```