LA CODIFICA DELL'INFORMAZIONE

Introduzione ai sistemi informatici

D. Sciuto, G. Buonanno, L. Mari, McGraw-Hill

Cap.2

Codifica dati e istruzioni

- Algoritmi = **istruzioni** che operano su **dati**.
- Per scrivere un programma è necessario rappresentare **istruzioni** e **dati** in un formato tale che l'esecutore automatico sia capace di
  - **memorizzare** istruzioni e dati;
  - **manipolare** istruzioni e dati.

Codifica dati e istruzioni

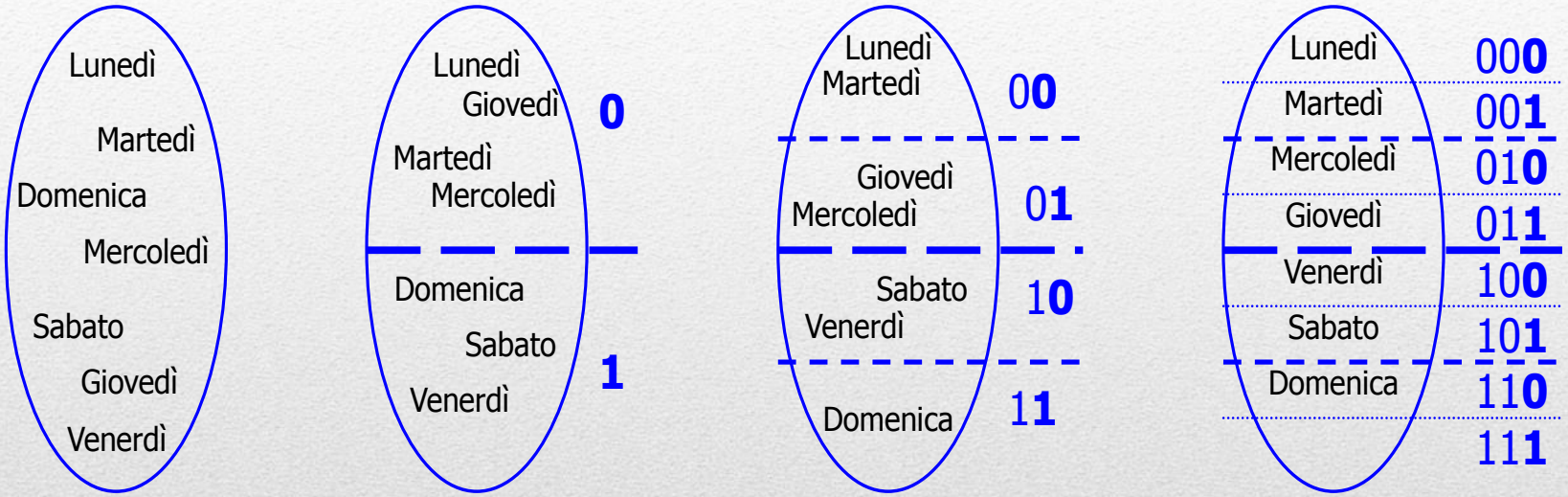
- **Alfabeto dei simboli**
  - cifre "**0**", "**1**", …, "**9**", separatore decimale ("**,**"), separatore delle migliaia ("**.**") e segni positivo ("**+**") o negativo ("**–**").
- **Regole di composizione** (sintassi), che definiscono le successioni "ben formate"
  - "**1.234,5**" è la rappresentazione di un numero;
  - "**1,23,45**" non lo è.
- **Codice** (semantica)
  - "**1.234,5**" = $1\times10^3 + 2\times10^2 + 3\times10^1 + 4\times10^0 + 5\times10^{-1}$
  - "**1,23,45**" = ??

- Lo stesso alfabeto può essere utilizzato con codici diversi:
  - "**123,456**" = $1\times10^2 + 2\times10^1 + 3\times10^0 + 4\times10^{-1} + 5\times10^{-2} + 6\times10^{-3}$, [IT]
  - "**123,456**" = $1\times10^5 + 2\times10^4 + 3\times10^3 + 4\times10^2 + 5\times10^1 + 6\times10^0$, [UK]

Codifica Binaria

- **Alfabeto binario**: usiamo dispositivi con solo due stati
- Problema: assegnare un **codice univoco** a tutti gli oggetti compresi in un insieme predefinito (e.g. studenti)
- Quanti **oggetti** posso codificare con **k** bit:
  - 1 bit $\Rightarrow$ 2 stati (0, 1) $\Rightarrow$ 2 oggetti (e.g. Vero/Falso)
  - 2 bit $\Rightarrow$ 4 stati (00, 01, 10, 11) $\Rightarrow$ 4 oggetti
  - 3 bit $\Rightarrow$ 8 stati (000, 001, …, 111) $\Rightarrow$ 8 oggetti
  - …
  - **k bit $\Rightarrow$ 2^k stati $\Rightarrow$ 2^k oggetti**
- Quanti **bit** mi servono per codificare **N** oggetti:
  - $N \leq 2^k \Rightarrow k \geq \log_2 N \Rightarrow$ **$k = \lceil \log_2 N \rceil$** (intero superiore)
- Attenzione: c'è l'ipotesi implicita che i codici abbiano tutti la **stessa lunghezza**

I giorni della settimana in binario



1 bit
2 "gruppi"

2 bit
4 "gruppi"

3 bit
8 "gruppi"

Codifica binaria dei caratteri

- Quanti sono gli oggetti compresi nell'insieme?
  - 26 lettere maiuscole + 26 minuscole $\Rightarrow$ 52
  - 10 cifre
  - Circa 30 segni d'interpunzione
  - Circa 30 caratteri di controllo (EOF, CR, LF, …)

  circa 120 oggetti complessivi $\Rightarrow$ $\mathbf{k = \lceil \log_2 120 \rceil = 7}$
- Codice ASCII: utilizza 7 bit e quindi può rappresentare al massimo 2^7=128 caratteri
  - Con 8 bit (= byte) rappresento 256 caratteri (ASCII esteso)
  - Esistono codici più estesi (e.g. UNICODE) per rappresentare anche i caratteri delle lingue orientali

bit, Byte, KiloByte, MegaByte, ...

bit = solo due stati, "0" oppure "1".

Byte = 8 bit, quindi $2^8 = 256$ stati

KiloByte [**KB**] = 2^{10} Byte = 1024 Byte ~ 10^3 Byte

MegaByte [**MB**] = 2^{20} Byte = 1 048 576 Byte ~ 10^6 Byte

GigaByte [**GB**] = 2^{30} Byte ~ 10^9 Byte

TeraByte [**TB**] = 2^{40} Byte ~ 10^{12} Byte

PetaByte [**PB**] = 2^{50} Byte ~ 10^{15} Byte

ExaByte [**EB**] = 2^{60} Byte ~ 10^{18} Byte

ASCII su 7 bit

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
010	sp	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
011	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
101	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
110	`	a	b	c	d	e	f	g	h	I	j	k	l	m	n	o
111	p	q	r	s	t	u	v	w	x	Y	z	{	|	}	~	can c

La codifica delle istruzioni

- Si segue lo schema presentato per i caratteri alfanumerici:
  - **quali e quante** sono le istruzioni da codificare?
  - qual è la **lunghezza** delle successioni di bit da utilizzare ?
  - qual è la **corrispondenza** tra istruzioni e successioni di bit ?

Istruzioni aritmetico-logiche	
Codice	**Istruzione**
01100000	ADD
01100100	SUB
01111110	AND
… … …	… … …

Istruzioni per il trasferimento dati	
Codice	**Istruzione**
00010101	LOAD
00110110	STORE
… … …	… … …
… … …	… … …

Istruzioni di controllo	
Codice	**Istruzione**
10011001	IF_EQ
10110110	GOSUB
10101100	RETURN
… … …	… … …

Oltre al codice operativo

- … è necessario far riferimento ai **dati** necessari per completare l'esecuzione dell'istruzione,
  - e.g. addizione: è necessario che sia specificato (anche **implicitamente**) dove leggere i due operandi da sommare e dove scrivere il risultato;
- il **numero** dei dati da specificare è variabile, in funzione delle istruzioni.

Numeri naturali

- Sistema di numerazione posizionale in base **b**
  - $c_k c_{k-1} \ldots c_0$ rappresenta $c_k \times b^k + c_{k-1} \times b^{k-1} + \ldots + c_0 \times b^0$
  - $b=10 \Rightarrow 1101_{dieci}$ indica $1 \times 10^3 + 1 \times 10^2 + 0 \times 10 + 1 \times 10^0$
- Conversione **binario ⇒ decimale**
  - basta scrivere il numero secondo la notazione posizionale utilizzando già il sistema decimale
  - $b=2 \Rightarrow 1101_{due}$ indica $1 \times 2^3 + 1 \times 2^2 + 0 \times 2 + 1 \times 2^0 = 13_{dieci}$
- Conversione **decimale ⇒ binario**
  - Si potrebbe utilizzare lo stesso metodo indicato sopra, ma è molto complesso
  - $b=10 \Rightarrow 345_{dieci}$ indica $11 \times 1010^2 + 100 \times 1010^1 + 101 \times 1010^0 =$
    $11 \times 1100100 + 100 \times 1010 + 101 \times 1 =$
    101011001

Conversione decimale binaria

Si calcolano i resti delle divisioni per due

18 : 2 =	9	resto 0
9 : 2 =	4	resto 1
4 : 2 =	2	resto 0
2 : 2 =	1	resto 0
1 : 2 =	0	resto 1

10010

137 : 2 =	68	resto 1
68 : 2 =	34	resto 0
34 : 2 =	17	resto 0
17 : 2 =	8	resto 1
8 : 2 =	4	resto 0
4 : 2 =	2	resto 0
2 : 2 =	1	resto 0
1 : 2 =	0	resto 1

10001001

Numeri interi

- Alfabeto binario
  - anche il segno è rappresentato da 0 o 1
  - è indispensabile indicare il numero **k** di bit utilizzati
- **Codifica con Modulo e segno**
  - **1** bit di segno (0 positivo, 1 negativo)
  - **k – 1** bit di modulo
    - Esempio: $+6_{dieci} = 0110_{ms}$ $-6_{dieci} = 1110_{ms}$
    - si rappresentano i valori da $-2^{k-1}+1$ a $2^{k-1}-1$
      - con 4 bit i valori vanno da –7 a +7
      - con 8 bit i valori vanno da –127 a +127
    - Attenzione: ci sono due rappresentazioni dello 0
      - con 4 bit sono $+0_{dieci} = 0000_{ms}$ $-0_{dieci} = 1000_{ms}$

Diverse codifiche/interpretazioni

Codice	Nat	MS
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7

Codice	Nat	MS
1000	8	-0
1001	9	-1
1010	10	-2
1011	11	-3
1100	12	-4
1101	13	-5
1110	14	-6
1111	15	-7

Ottali ed esadecimali

- Utili per rappresentare sinteticamente i valori binari
- **Ottali (base b = 8)**
  - Alfabeto ottale: cifre comprese tra 0 e 7
    - $354_{otto} = 3\times8^2 + 5\times8^1 + 4\times8^0 = 192+40+4 = 236_{dieci}$
    - $1461_{otto} = 1\times8^3 + 4\times8^2 + 6\times8^1 + 1\times8^0 = 512+256+48+1 = 817_{dieci}$
  - Ogni cifra ottale corrisponde a tre cifre binarie:
    - 11101100_{due} = [11] [101] [100] = 354_{otto}
    - 1100110001_{due} = [1] [100] [110] [001] = 1461_{otto}
- **Esadecimali (base b = 16)**
  - Alfabeto esadecimale: cifre 0 – 9 + lettere A – F
    - $EC_{sedici} = 14\times16^1 + 12\times16^0 = 224 + 12 = 236_{dieci}$
    - $331_{sedic} = 3\times16^2 + 3\times16^1 + 1\times16^0 = 768+48+1 = 817_{dieci}$
  - Ogni cifra esadecimale corrisponde a quattro cifre binarie:
    - 11101100_{due} = [1110] [1100] = EC_{sedici}
    - 1100110001_{due} = [11] [0011] [0001] = 331_{sedici}

Numeri razionali

- Rappresentazione in **virgola fissa**
  - $0.1011_{due} = 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3} + 1\times2^{-4}$
    $= 0.5 + 0.125 + 0.0625 = 0.6875_{dieci}$
  - $11.101_{due} = 1\times2^{1} + 1\times2^{0} + 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3}$
    $= 2 + 1 + 0.5 + 0.125 = 3.625_{dieci}$
- Rappresentazione in **virgola mobile** (float)
  - usata spesso anche in decimale per rappresentare numeri molto grandi o molto piccoli: 0.1357×10^{64}
    - **mantissa** – parte frazionaria compresa tra 0 e 1 [0.1357]
    - **esponente** – numero **intero**
  - utilizza 1 bit per il segno (s), h bit per l'esponente (e) e k bit per la mantissa (m): $\mathbf{R = s \times m \times 2^{e}}$

Numeri razionali

- Rappresentazione in **virgola mobile** (notazione scientifica)
  - il numero -123.450.000.000$_{dieci}$ viene rappresentato come -0.12345 x $10^{12}{}_{dieci}$
  - il numero 0.0000012345$_{dieci}$ viene rappresentato come 0.12345 x $10^{-5}{}_{dieci}$
  - vale anche per i numeri binari: il numero 101010000$_2$ diventa 0.10101 x 10 $^{01001}{}_{due}$
  - Quanti bit utilizzare per la rappresentazione? Si può modificare la precisione dei risultati, ma standard dell'IEEE:
    - a precisione singola (32 bit: 8 esponente, 23 mantissa)
    - a precisione doppia (64 bit: 11 esponente, 52 mantissa)

Approfondimento: il complemento a 2

Numeri interi in complemento a 2

- Alfabeto binario
  - anche il segno è rappresentato da 0 o 1
  - è indispensabile indicare il numero **k** di bit utilizzati
- **Complemento a 2**
  - **X** corrisponde al binario naturale di $2^k + X$

    $+6_{dieci} \Rightarrow 2^4+6 = 22 \Rightarrow [1]0110 \Rightarrow 0110_{C2}$

    $-6_{dieci} \Rightarrow 2^4-6 = 10 \Rightarrow [0]1010 \Rightarrow 1010_{C2}$
  - si rappresentano i valori da -2^{k-1} a $2^{k-1}-1$
    - con 4 bit i valori vanno da –8 a +7
    - con 8 bit i valori vanno da –128 a +127
  - Attenzione: c'è una sola rappresentazione dello 0
    - con 4 bit è $+0_{dieci} = 0000_{C2}$ mentre $1000_{C2} = -8_{dieci}$

Complemento a 2 – Alcune osservazioni

- i numeri positivi iniziano con **0**, quelli negativi con **1**
- data la rappresentazione di un numero su **k** bit, la rappresentazione dello stesso numero su **k+1** bit si ottiene aggiungendo (a sn) un bit uguale al primo (**estensione del "segno"**)
  - Rappresentazione di –6 su 4 bit = 1010
  - Rappresentazione di –6 su 5 bit = 11010
  - Rappresentazione di –6 su 8 bit = 11111010
- la sottrazione si effettua come somma algebrica
  - 4 – 6 = +4 + (–6) = 0100 + 1010 = 1110 = –2
  - 9 – 6 = +9 + (–6) = 01001 + 11010 = [1]00011 = +3

ho fissato a priori il n. di bit da usare

Diverse codifiche/interpretazioni

Codice	Nat	MS	C2
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7

Codice	Nat	MS	C2
1000	8	-0	-8
1001	9	-1	-7
1010	10	-2	-6
1011	11	-3	-5
1100	12	-4	-4
1101	13	-5	-3
1110	14	-6	-2
1111	15	-7	-1