RAPPRESENTAZIONE DEI DATI IN MEMORIA

Fondamenti di Programmazione 2021/2022

Francesco Tortorella

(si ringrazia la prof. Sabrina Senatore per l'impiego di alcune sue slides)

Il problema della memorizzazione

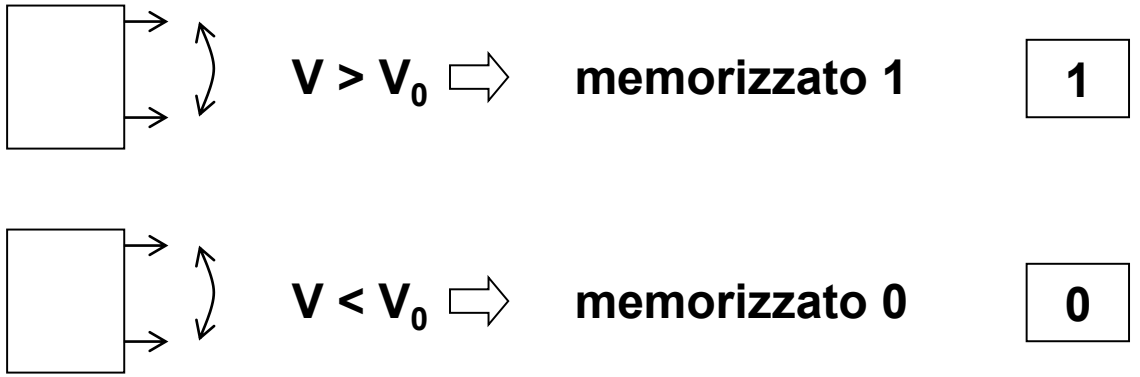
- Nei nostri algoritmi/programmi abbiamo la necessità di memorizzare diversi tipi di informazioni all'interno dell'esecutore.
- Quali informazioni?
  - **Dati**
  - **Istruzioni**

Misurare la quantità di informazione

- Per affrontare correttamente il problema della memorizzazione, dobbiamo avere la possibilità di misurare la quantità di informazione che va memorizzata.
- Qual è la quantità minima di informazione che possiamo considerare?
- "Possiamo assumere che *informazione* è tutto ciò che può consentire di ridurre il nostro grado di incertezza in merito ad una particolare situazione"

Misurare la quantità di informazione

- La più piccola unità di informazione memorizzabile (e quindi utilizzabile) è il **bit**, che può assumere uno tra due valori possibili (es. 0 o 1).
- Il dispositivo utilizzato per memorizzare un bit è un **elemento bistabile**, cioè un dispositivo elettronico che può assumere uno tra due stati stabili (es. due livelli differenti di tensione), ognuno dei quali viene fatto corrispondere a 0 o a 1 (**cella di memoria**).



Cella di memoria: operazioni

- Sono possibili due operazioni su una cella di memoria
- **Operazione di scrittura**: la cella di memoria viene caricata con un determinato valore (0 o 1) che permane memorizzato finché:
  - la cella viene alimentata elettricamente
  - non si esegue un'altra operazione di scrittura che modifica il valore precedentemente memorizzato
- **Operazione di lettura**: si accede alla cella di memoria per consultarne il valore e copiarlo su un'altra cella di memoria.

Quanti bit?

- Con un solo bit è possibile gestire un'informazione binaria, cioè un'informazione che può specificare uno tra due valori possibili (es. un punto di un'immagine bianco o nero).
- Quanti stati possibili può assumere un insieme di bit ?

00
01
10
11

000
001
010
011
100
101
110
111

0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111

2 bit → 4 stati
3 bit → 8 stati
4 bit → 16 stati
...

Il registro di memoria

- **Un insieme di N celle elementari può assumere uno tra 2^N stati possibili**.
- Un tale insieme è organizzato in un **registro di memoria**.
- Il registro costituisce un supporto per la memorizzazione di un'informazione che può assumere uno tra 2^N valori possibili. In particolare un insieme di 8 bit forma un **byte**.
- Sul registro sono possibili operazioni di lettura e scrittura che interessano contemporaneamente tutte le celle di memoria contenute nel registro.

Il registro di memoria

- Possibile usare il registro come supporto per la memorizzazione delle nostre informazioni?
- Quali sono le condizioni?
- Un registro consente di memorizzare una tipologia di informazione il cui generico valore appartiene ad un insieme **finito** e **discreto**
- Le informazioni con cui abbiamo a che fare rispettano queste condizioni?

Il registro di memoria

- Per il momento consideriamo informazioni "adeguate"
  - Un insieme di 7 colori
  - Un insieme di 10 numeri naturali
  - Un insieme di 12 suoni
- Quanti bit sono necessari per memorizzare
  - Un colore?
  - Un numero naturale?
  - Un suono?

Il registro di memoria

- Un secondo problema è che abbiamo un'unica tipologia di supporto, ma diverse tipologie di informazioni.
- È quindi necessario adottare una **codifica** del tipo di dato considerato: occorre, cioè, trovare una rappresentazione del dato gestibile con le possibilità offerte dal sistema di elaborazione.

Che cosa possiamo fare con un registro da 8 bit?

- registro da 8 bit ➡ $2^8 = 256$ stati possibil

Numeri naturali [0,255]

0 ↔ 00000000
1 ↔ 00000001
....
255 ↔ 11111111

Numeri interi [-128,127]

-128 ↔ 00000000
-127 ↔ 00000001
0 ↔ 10000000
+127 ↔ 11111111

Numeri reali [0,1[

0.0000 ↔ 00000000
0.0039 ↔ 00000001
0.0078 ↔ 00000010
....
0.9961 ↔ 11111111

Caratteri

A ↔ 01000001
a ↔ 01100001

0 ↔ 00110000
1 ↔ 00110001

La codifica

- Che cos'è una codifica?
- Una stessa informazione può essere rappresentata in diversi modi
  - “il numero 5”: V, 5.00, 101, IIIII, cinque, five, …
- Tale informazione deve essere *comprensibile* affinché chi riceve tale rappresentazione sia capace di recuperare l'informazione che il mittente intendeva inviare.
- E’ importante quindi conoscere e condividere il **sistema di codifica** (o codifica o codice).

La codifica

- Tipicamente un sistema di codifica usa un insieme di simboli (**alfabeto**), combinazioni di questi simboli (configurazioni, **stati**).
- Ogni informazione può essere rappresentata da una combinazione di simboli appartenenti all'alfabeto stabilito che costituisce la codifica associata.
- La codifica è il processo che mette in **corrispondenza biunivoca** un valore particolare dell'informazione ed una particolare combinazione di simboli.

Un esempio di codifica numerica

- Alfabeto dei simboli
  - cifre "**0**", "**1**", …, "**9**", separatore decimale ("**,**"), separatore delle migliaia ("**.**") e segni positivo ("**+**") o negativo ("**–**").
- Regole di composizione (sintassi), che definiscono le combinazioni "corrette"
  - "**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]

Un altro esempio di codifica numerica

- Alfabeto dei simboli
  - cifre “**I**”, “**V**”, “**X**” , “**L**”, “**C**”, “**D**”, “**M**”
- Regole di composizione (sintassi), che definiscono le combinazioni “corrette”
  - “**MCMLXVIII**”, “**MMXX**”, sono rappresentazioni di numeri;
  - “**LDLIM**” non lo è.
- Codice (semantica)
  - “**MCMLXVIII**” = 1000+(1000-100)+ 50+ 10+ 5+ 1 +1 + 1
  - “**LDLIM**” = ??

Rappresentazione dati: numeri

- La modalità di rappresentazione dei numeri (interi) che noi usiamo di consueto corrisponde ad un sistema **posizionale** e **pesato**.
- L'alfabeto è composto da 10 cifre (0, 1, 2, …, 9) ed ogni sequenza corrisponde ad un numero che è la somma pesata delle cifre che la costituiscono.
- $3707 = 3 \times 10^3 + 7 \times 10^2 + 0 \times 10^1 + 7 \times 10^0$
- **Fondamentale la presenza dello 0**

Rappresentazione dati: numeri

- La nostra modalità consueta assume 10 come **base di numerazione b**:
  - Cifre: **0 1 2 3 4 5 6 7 8 9**
  - Pesi: potenze di 10
- Unica base 10? Che succede se b=8?
  - Cifre: **0 1 2 3 4 5 6 7**
  - Pesi: potenze di 8
  - Esempio: $757_8 = 7 \times 8^2 + 5 \times 8^1 + 7 \times 8^0$ = quattrocentonovantacinque = 495_{10}

Rappresentazione dati: numeri

- In generale, per una base b:
  - Cifre: **0 1 2 3 … b-1**
  - Pesi: potenze di b (b^0, b^1, b^2, b^3, …)
  - La sequenza di cifre $\mathbf{c_k c_{k-1} \ldots c_0}$ rappresenta il numero
    $c_k \times b^k + c_{k-1} \times b^{k-1} + \ldots + c_1 \times b^1 + c_0 \times b^0$
- Per cui, una sequenza di cifre può indicare numeri diversi, a seconda della base impiegata:
  - $357_{10} = 3 \times 10^2 + 5 \times 10^1 + 7 \times 10^0$
  - $357_8 = 3 \times 8^2 + 5 \times 8^1 + 7 \times 8^0$

Base di numerazione 2

- Che succede per b = 2?
  - Cifre: **0 1**
  - Pesi: potenze di 2 (2^0, 2^1, 2^2, 2^3, …)
- **Cifre direttamente rappresentabili all'interno di un registro di memoria**

Conversione di base

- Convertire in base 10 un numero rappresentato in base 2 è semplice:
  - $10011_2 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 16 + 2 + 1 = 19_{10}$
- Come convertire in base b un numero rappresentato in base 10?
- Dato un numero T, vogliamo ottenere la sequenza di cifre in base b $\mathbf{c_k c_{k-1} \ldots c_0}$ tale che
  $c_k \times b^k + c_{k-1} \times b^{k-1} + \ldots + c_1 \times b + c_0 = T$

Conversione di base

- Se dividiamo T per b: $T = Q_0 \times b + r$ (con $r < b$)
- … ma $T = c_k \times b^k + c_{k-1} \times b^{k-1} + \ldots + c_1 \times b + c_0 =$
  $= (c_k \times b^{k-1} + c_{k-1} \times b^{k-2} + \ldots + c_1) \times b + c_0$
- E quindi: $Q_0 = c_k \times b^{k-1} + c_{k-1} \times b^{k-2} + \ldots + c_1$ e $r = c_0$
- Se dividiamo T per b otteniamo un quoziente Q_0 ed un resto che costituisce la prima cifra della sequenza (c_0)
- Possiamo applicare la stessa operazione a Q_0, ottenendo un quoziente Q_1 ed un resto che coincide con c_1
- Ripetiamo il procedimento finché si ottiene un quoziente nullo.

Esempio

$573_{10} : 2_{10}$	$\Rightarrow$	*quoziente*	286_{10}	*resto* 1
$286_{10} : 2_{10}$	$\Rightarrow$	*quoziente*	143_{10}	*resto* 0
$143_{10} : 2_{10}$	$\Rightarrow$	*quoziente*	71_{10}	*resto* 1
$71_{10} : 2_{10}$	$\Rightarrow$	*quoziente*	35_{10}	*resto* 1
$35_{10} : 2_{10}$	$\Rightarrow$	*quoziente*	17_{10}	*resto* 1
$17_{10} : 2_{10}$	$\Rightarrow$	*quoziente*	8_{10}	*resto* 1
$8_{10} : 2_{10}$	$\Rightarrow$	*quoziente*	4_{10}	*resto* 0
$4_{10} : 2_{10}$	$\Rightarrow$	*quoziente*	2_{10}	*resto* 0
$2_{10} : 2_{10}$	$\Rightarrow$	*quoziente*	1_{10}	*resto* 0
$1_{10} : 2_{10}$	$\Rightarrow$	*quoziente*	0_{10}	*resto* 1

$$1\ 000\ 111\ 101_2 = 573_{10}$$

Esercizio

- Convertire 27 in base 3
- Convertire 31 in base 2

Base 8 e base 16

- Le considerazione fatte per le basi decimale e binaria si possono estendere ad altre basi.
- Due basi comunemente usate sono ottale e esadecimale
  - Ottale: 0, 1, 2, 3, 4, 5, 6, 7
  - Esadecimale: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
- Utili per rappresentare sinteticamente i valori binari

Base 8

- Siccome $8 = 2^3$, esiste un modo rapido per la conversione di base da 2 a 8
  1. Si considera il numero in base 2 e, partendo da destra, si divide in gruppi di 3 cifre binarie. Se dopo l'operazione avanzano una o due cifre si aggiungono tanti zeri quanti bastano a coprire un gruppo di tre
  2. Ogni gruppo va poi convertito nella corrispondente cifra ottale.
- Esempi:
  - 11101101_2 = [11] [101] [101] = [011] [101] [101] = 355_8
  - 1100110011_2 = [1] [100] [110] [011] = [001] [100] [110] [011] = 1463_8

Base 8

- In maniera analoga si realizza la conversione da base 8 a base 2
  - Dato un numero rappresentato in base 8, si sostituisce ogni cifra ottale con la terna di cifre binarie che rappresenta quella cifra in base 2, eliminando alla fine eventuali 0 a sinistra
- Esempi:
  - 756_8 = [111] [101] [110] = 111101110_2
  - 134_8 = [001] [011] [100] = 001011100_2 = 1011100_2

Base 16

- Siccome $16 = 2^4$, esiste un modo rapido anche per la conversione di base da 2 a 16
  1. Si considera il numero in base 2 e, partendo da destra, si divide in gruppi di 4 cifre binarie. Se dopo l'operazione avanzano una, due o tre cifre si aggiungono tanti zeri quanti bastano a coprire un gruppo di quattro
  2. Ogni gruppo va poi convertito nella corrispondente cifra esadecimale.
- Esempi:
  - $11101101_2 = [1110]\ [1101] = ED_{16}$
  - $1100110011_2 = [11][0011][0011] = [0011][0011][0011] = 333_{16}$

Base 16

- In maniera analoga si realizza la conversione da base 16 a base 2
  - Dato un numero rappresentato in base 16, si sostituisce ogni cifra esadecimale con la quaterna di cifre binarie che rappresenta quella cifra in base 2, eliminando alla fine eventuali 0 a sinistra
- Esempi:
  - $7A4_{16}$ = [0111] [1010] [0100] = 011110100100_2 = 11110100100_2
  - $13C_{16}$ = [0001] [0011] [1100] = 000100111100_2 = 100111100_2

Rappresentazione dati: caratteri

- Si consideri l'insieme dei caratteri più comuni:
  - 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 $\Rightarrow$ $k = \lceil \log_2 120 \rceil = 7$
- Codice ASCII: utilizza 7 bit e quindi può rappresentare al massimo 2^7=128 caratteri
  - Con 8 bit (= byte) si raddoppia rappresentando 256 caratteri (ASCII esteso)

Codifica ASCII a 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	{	|	}	~	canc

- Come si legge?
- Il carattere “A” in codifica binaria corrisponde a 1000001