Fondamenti di Informatica

Allievi Automatici

A.A. 2015-16

Codifica dell'Informazione

Cenni all'aritmetica del Calcolatore

Prime nozioni di algebra di Boole

L’informazione

- **Messaggio** che **apporta conoscenza:**
  - C’è una situazione iniziale, di **ignoranza**
  - C’è un evento fisico di qualche tipo
  - C’è una situazione finale, in cui la conoscenza è superiore a quella iniziale
- Es: la lettura di questa slide
  - All’inizio non sapevate che cosa fosse l’informazione
  - Alcuni (molti) fotoni hanno colpito la vostra retina in un certo modo
  - Ora sapete che cosa è l’informazione!

Altri esempi di informazione

- Esempi di informazione:
  - Un suono può apportare informazione:
    - Può portare una notizia
    - Può apportare conoscenza sull'ambiente circostante (il suono del clacson di una macchina che sta arrivando)
    - L'informazione può anche *semplicemente* essere il fatto che si è mossa dell'aria
- Lo stesso messaggio può portare quantità di informazione diverse a seconda dello stato di chi lo riceve (ognuno lo interpreta, lo elabora)

Elaborazione

- Che cosa si può fare, con questa informazione?
  - La notizia: trascriverla, ritrasmetterla, eventualmente dopo averla modificata, tradotta, sintetizzata
  - Il segnale: registrarlo, usarlo per reagire (fare un salto all'indietro), **misurarne** l'intensità
  - Il suono: modificarlo, registrarlo, utilizzarlo per creare messaggi visivi, per controllare dei proiettori luminosi, …

Elaborazione **automatica**

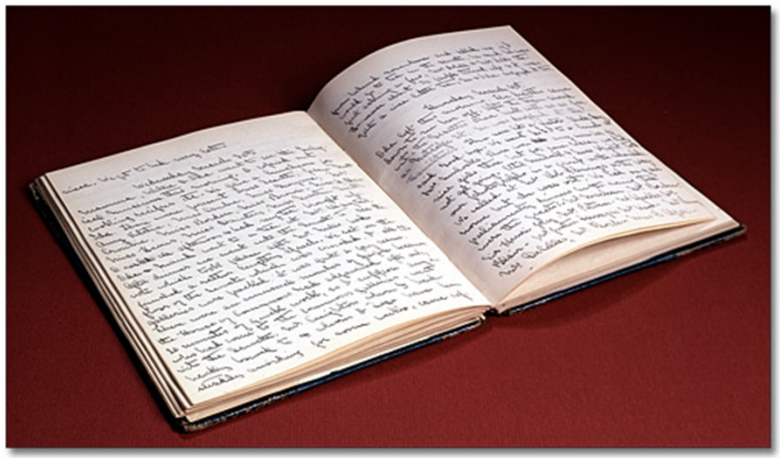
- I calcolatori elettronici sono in grado di compiere molto bene alcune di queste **elaborazioni di informazione…**
- Purché:
  - L'informazione sia rappresentata con una codifica opportuna (digitale, cioè codificata **numericamente**)
  - Le elaborazioni da compiere siano descritte in modo **algoritmico**

Informazione numerica (digitale)

- Che cosa intendiamo per informazione?
- **Informazione**: in effetti, di non facile definizione
- Per i nostri fini (capire cosa può fare un calcolatore) possiamo considerare **informazione** ciò che viene trasmesso da un **messaggio** rappresentabile con un **numero** (intero)

Esempi

- Un diario contiene di certo dell'informazione. Si tratta di informazione che può essere manipolata da un calcolatore?



Informazione

- “Quello che c’è scritto” è informazione che può essere rappresentata con un **numero**?
- Sì:
  - possiamo assegnare un numero di due cifre numero ad ogni lettera dell’alfabeto (‘a’=10,’b’=11 etc), compresi gli spazi
  - chiamiamo la coppia di cifre corrispondente ad una certa lettera **codice** della lettera (ad esempio: 10 per ‘a’)
  - possiamo **trascrivere** il diario utilizzando i nostri codici, uno dietro l’altro
  - Otteniamo un numero intero, molto grande, che **codifica** il diario
  - L’informazione contenuta nella codifica è *sufficiente* a ricostruire “quello che c’è scritto” nel diario

Cosa ce ne facciamo, dell'informazione?

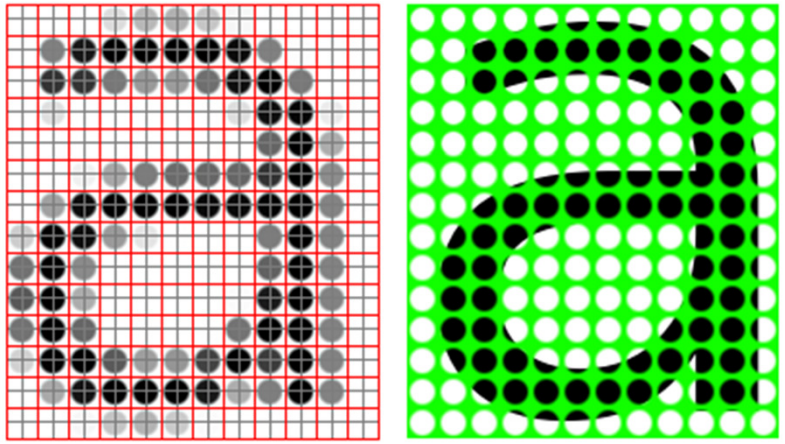
- Che cosa può fare un calcolatore con il numero che rappresenta il diario?
  - può **memorizzarlo**
  - può eventualmente **trasmetterlo**
  - può **misurare** la frequenza d'uso di alcune parole
  - può **controllare** che le parole siano scritte in italiano corretto
  - può cercare di capire la **psicologia** dello scrivente?

Tutta l'informazione?

- Qualcuno potrebbe obiettare che nel diario c'è molto di più di quello che c'è scritto…
- Per esempio: la ***grafia*** (il modo con cui chi ha scritto ha scritto quello che ha scritto) potrebbe contenere dell'informazione
- Possiamo tradurre l'informazione sulla grafia in un numero?

Codificare un'immagine

a

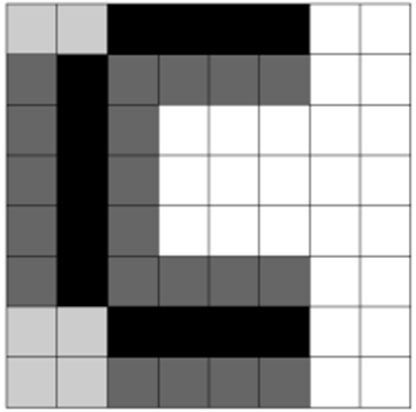


Ad ogni punto facciamo corrispondere un numero che ne codifica il livello di grigio. Mettendo di seguito tutte le cifre di tali numeri in un ordine dato, l'immagine è codificata da un solo grande numero.

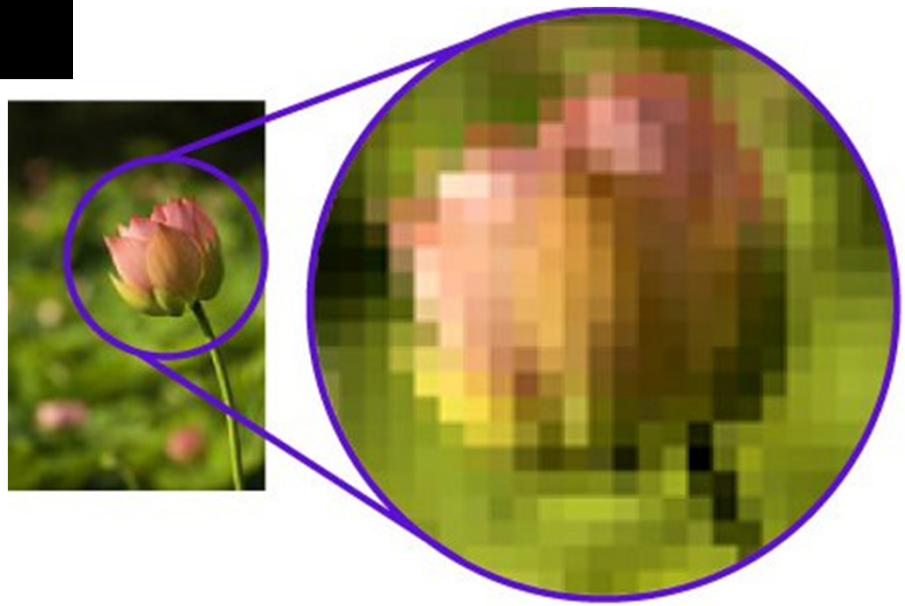
0	0	8	41	77	119	178	221	234	248
0	8	28	41	77	128	192	234	248	255
0	8	28	41	119	178	221	248	255	248
0	8	41	77	128	192	248	255	248	234
8	28	77	119	178	234	255	248	221	192
28	77	119	178	221	248	255	234	221	192
51	95	128	192	234	255	248	234	178	150
77	119	150	221	248	248	234	192	150	128

100	100	0	0	0	0	255	255
50	0	50	50	50	50	255	255
50	0	50	255	255	255	255	255
50	0	50	255	255	255	255	255
50	0	50	255	255	255	255	255
50	0	50	50	50	50	255	255
100	100	0	0	0	0	255	255
100	100	50	50	50	50	255	255

(a)



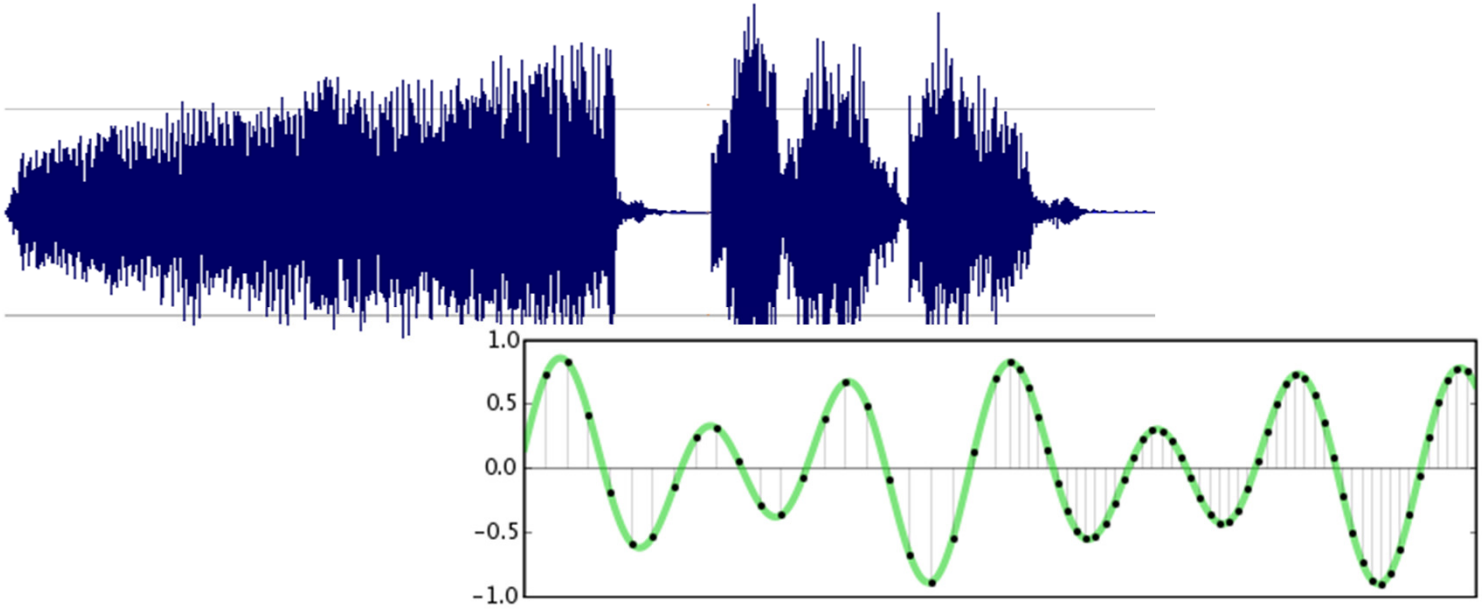
(b)



E il suono?

- E il suono? Anche tutta l'informazione contenuta in un brano musicale la posso trasformare in un numero?
- Ma che cos'è, in questo caso, l'"informazione"?
  - Ad esempio: la conoscenza che serve per "riprodurre" acusticamente il brano (cioè per riprodurne il suono)

I numeri del suono



- La percezione del suono è fisicamente causata dalla variazione, nel tempo, della pressione dell'aria in prossimità del timpano.
- La pressione si può misurare (*cioè convertire in un numero*) a intervalli di tempo piccoli (campionamento), i valori sono numeri..
- …anche un brano musicale si può rappresentare con un numero intero

Insomma…

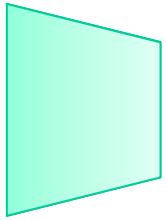
- …molta dell'informazione con cui abbiamo a che fare e che può interessarci elaborare…
- ...può essere rappresentata con un numero intero
- Cosa **non** può esserlo?
  - Per Pitagora, niente (tutto è numero)
  - Per il pancomputazionalismo, nemmeno (tutta la realtà fisica sarebbe rappresentabile digitalmente)

In questo corso…

- Non tratteremo il problema di acquisire l'informazione che vogliamo elaborare
  - Immagineremo di averla già disponibile, in forma numerica
- Non tratteremo il problema di comunicare i risultati dell'elaborazione in modo efficacemente fruibile
  - Ci accontenteremo di «visualizzarli da qualche parte»
- Impareremo, invece, come far compiere al calcolatore le elaborazioni che desideriamo sulla nostra informazione **codificata**, per trasformarla:
  - Impareremo, cioè, a **programmare il calcolatore**

Calcolo e ambiente

input

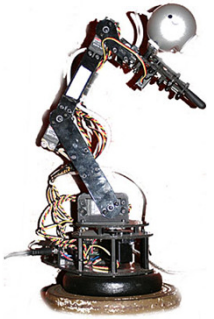


172312

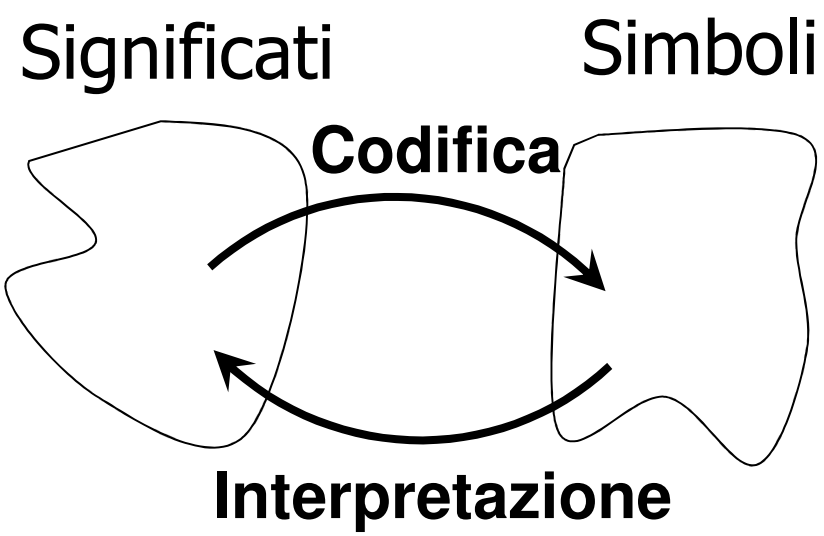
Calcolatore

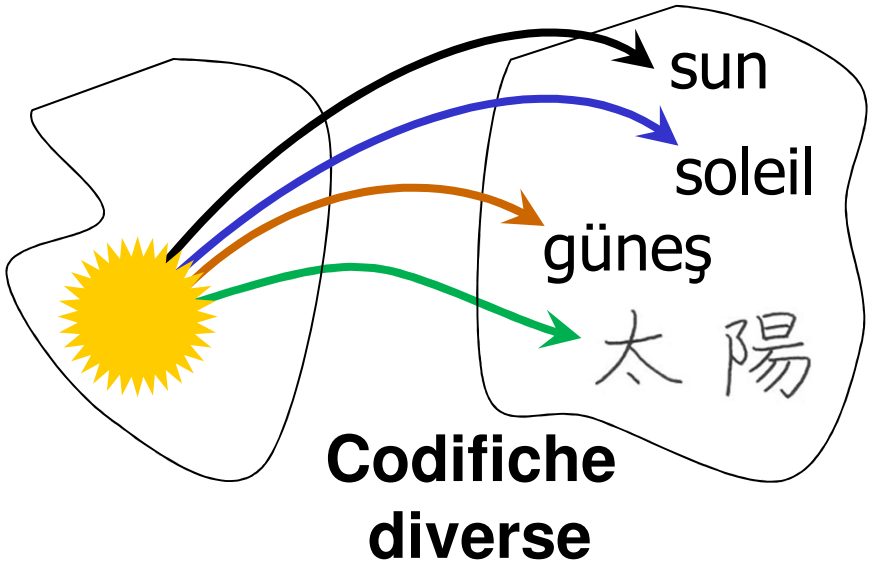
514232

output

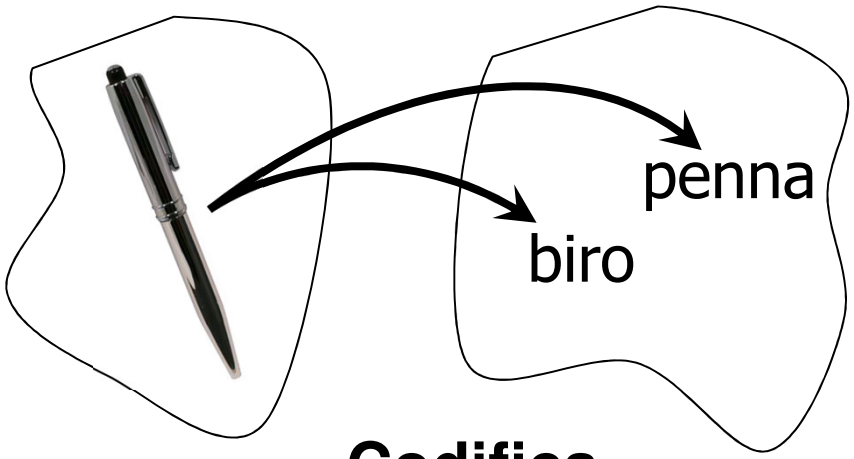


Rappresentare l'informazine:
associare **significati** e **simboli**



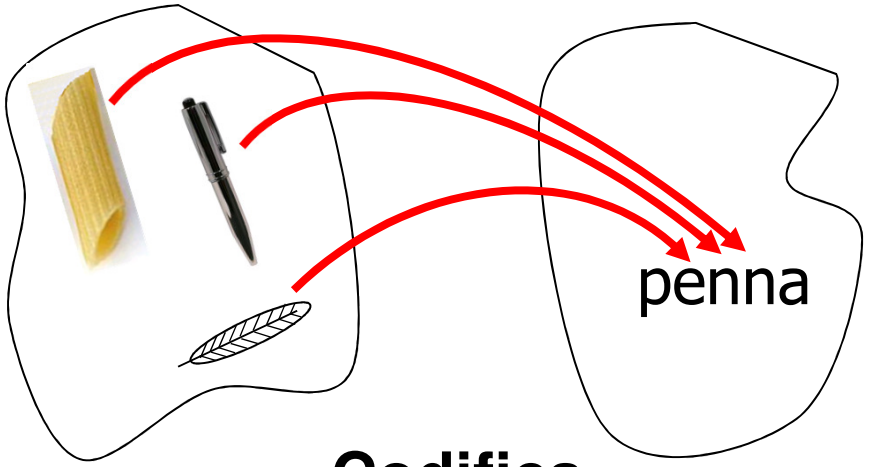


Rappresentare l'informazine:
associare **significati** e **simboli**



Codifica ridondante

(l'interpretazione è univoca)



Codifica ambigua

(l'interpretazione **non** è univoca)

Codifica dell'informazione

- Rappresentare (codificare) le informazioni
  - con un insieme limitato di simboli (detto *alfabeto* $\mathcal{A}$)
  - in modo non ambiguo (algoritmi di traduzione tra codifiche)
- Esempio: numeri interi
  - Codifica decimale (**dec**, in base dieci)
  - $\mathcal{A}$ = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }, $|\mathcal{A}|$ = dieci
    - "sette" : 7_{dec}
    - "ventitre" : 23_{dec}
    - "centotrentotto" : 138_{dec}
  - Notazione *posizionale*
    - dalla cifra più significativa a quella meno significativa
    - ogni cifra corrisponde a una diversa potenza di dieci

Numeri naturali

- ***Notazione posizionale***: permette di rappresentare un qualsiasi numero naturale (intero non negativo) nel modo seguente:

  la sequenza di **cifre c_i**:

  $$c_n \; c_{n-1} \; \ldots \; c_1$$

  rappresenta in **base B** ≥ 2 il valore:

  $$c_n \times B^{n-1} + c_{n-1} \times B^{n-2} + \ldots + c_1 \times B^0$$

  dove: $c_i \in \{0, 1, 2, \ldots, B-1\}$ per ogni $1 \leq i \leq n$

- La notazione decimale tradizionale è di tipo posizionale (ovviamente con B = dieci)
- Esistono notazioni non posizionali
  - Ad esempio i numeri romani: II IV VI XV XX VV

Numeri naturali in varie basi

- Base generica: B
  - $\mathcal{A}$ = { ... }, con $|\mathcal{A}|$ = B, sequenze di n simboli (cifre)
  - $c_n\, c_{n-1} \ldots c_2\, c_1 = c_n \times B^{n-1} + \ldots + c_2 \times B^1 + c_1 \times B^0$
  - Con n cifre rappresentiamo B^n numeri: da 0 a B^n-1
- "ventinove" in varie basi
  - B = otto $\mathcal{A}$ = {0,1,2,3,4,5,6,7} $29_{dec} = 35_8$
  - B = cinque $\mathcal{A}$ = {0,1,2,3,4} $29_{dec} = 104_5$
  - B = tre $\mathcal{A}$ = {0,1,2} $29_{dec} = 1002_3$
  - B = sedici $\mathcal{A}$ = {0,1,...,8,9,A,B,C,D,E,F} $29_{dec} = 1D_{16}$
- Codifiche notevoli
  - Esadecimale (sedici), ottale (otto), binaria (due)

Codifica **binaria**

- Usata dal calcolatore per **tutte** le informazioni
  - B = due, $\mathcal{A}$ **= { 0, 1 }**
  - **BIT** (crasi di "**BI**nary dig**IT**"):
    - unità **elementare** di informazione
  - Dispositivi che assumono **due** stati
    - Ad esempio due valori di tensione V_A e V_B
- *Numeri binari naturali:*

  la sequenza di **bit b_i** (cifre binarie):

  $$b_n\ b_{n-1}\ \ldots\ b_1 \qquad \text{con } b_i \in \{0, 1\}$$

  rappresenta in base 2 il valore:

  $$b_n \times 2^{n-1} + b_{n-1} \times 2^{n-2} + \ldots + b_1 \times 2^0$$

Numeri binari naturali (bin)

- Con n bit codifichiamo 2^n numeri: da 0 a 2^n-1
- Con 1 Byte (cioè una sequenza di 8 bit):
  - $00000000_{bin} = 0_{dec}$
  - $00001000_{bin} = 1 \times 2^3 = 8_{dec}$
  - $00101011_{bin} = 1 \times 2^5 + 1 \times 2^3 + 1 \times 2^1 + 1 \times 2^0 = 43_{dec}$
  - $11111111_{bin} = \Sigma_{n = 1,2,3,4,5,6,7,8}\ 1 \times 2^{n-1} = 255_{dec}$
- Conversione bin $\rightarrow$ dec e dec $\rightarrow$ bin
  - bin$\rightarrow$dec: $11101_{bin} = \Sigma_i\, b_i 2^i = 2^4+2^3+2^2+2^0 = 29_{dec}$
  - dec$\rightarrow$bin: ***metodo dei resti***

Conversione dec → bin

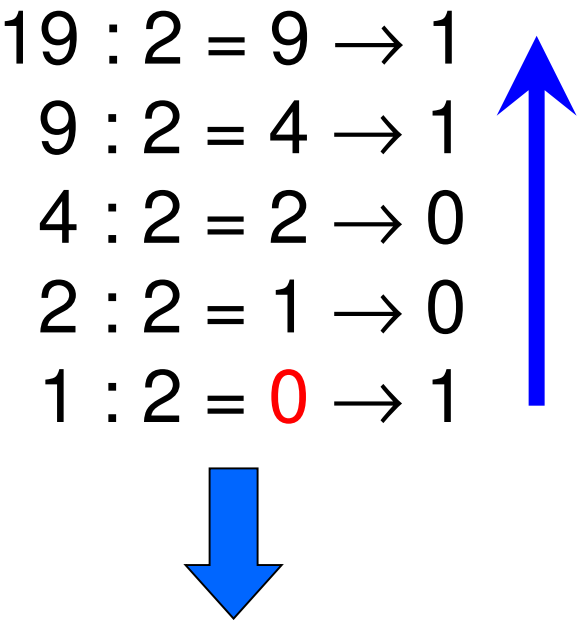
Metodo dei resti: si calcolano i resti delle divisioni per due

In pratica basta:

1. Decidere se il numero è pari (resto 0) oppure dispari (resto 1), e annotare il resto
2. Dimezzare il numero (trascurando il resto)
3. Ripartire dal punto 1. fino a quando si ottiene 0 come risultato della divisione

Ecco un esempio, per quanto modesto, di **algoritmo**

si ottiene 0: fine

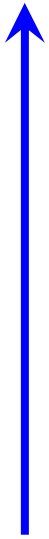


Metodo dei resti

29 : 2 = 14 (1)
14 : 2 = 7 (0)
7 : 2 = 3 (1)
3 : 2 = 1 (1)
1 : 2 = **0** (1)

$\mathbf{29}_{\mathbf{dec}} = 11101_{bin}$

76 : 2 = 38 (0)
38 : 2 = 19 (0)
19 : 2 = 9 (1)
9 : 2 = 4 (1)
4 : 2 = 2 (0)
2 : 2 = 1 (0)
1 : 2 = **0** (1)



$\mathbf{76}_{\mathbf{dec}} = 1001100_{bin}$

Del resto 76 = 19x4 = **10011**00
Per raddoppiare, in base due, si aggiunge uno zero in coda, così come si fa in base dieci per decuplicare

N.B. Il metodo funziona con tutte le basi!

$29_{10}=45_{6}=32_{9}=27_{11}=21_{14}=10_{29}$

Ma i numeri non sono solo interi…

- Rappresentazione dei numeri **interi** anche **negativi**
  - Ci sono due diverse codifiche interessanti
    - In **modulo e segno** (in pratica non usata)
    - in **complemento a 2** (quella usata universalmente)
- Rappresentazione dei numeri **frazionari**
  - Stabilendone la "precisione" (in "virgola fissa")
    - cioè stabilendo il numero di cifre dopo la virgola
- Rappresentazione dei numeri "**reali**"
  - In realtà, di una loro approssimazione (in "virgola mobile")
    - si rappresenta un "campionamento del continuo"

Codifica di testi, immagini, suoni, ...

- Caratteri: sequenze di bit
  - Codice ASCII: utilizza 7(8) bit: 128(256) caratteri
  - 1 Byte (l'8° bit può essere usato per la *parità*)
- Testi: sequenze di caratteri (cioè di bit)
- Immagini: sequenze di bit
  - bitmap: sequenze di pixel (n bit, 2^n colori)
  - jpeg, gif, pcx, tiff, …
- Suoni (musica): sequenze di bit
  - wav, mid, mp3, ra, …
- Filmati: immagini + suoni
  - sequenze di …?

… "rivoluzione" digitale

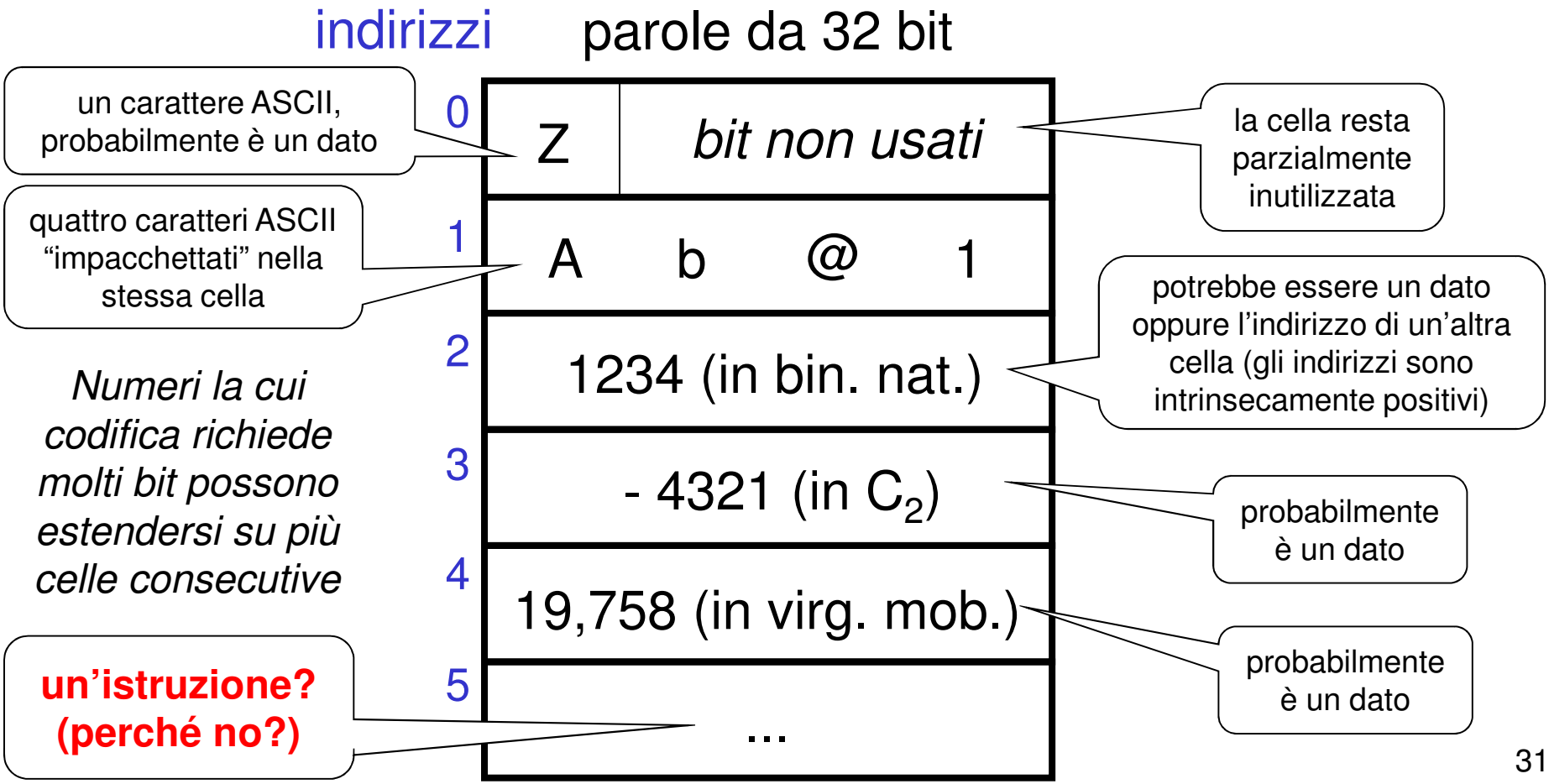
Operazioni sui numeri binari

- Dove sono memorizzati i numeri, in un calcolatore?
  - Ad esempio in "registri" di memoria, cioè schiere di circuiti (hardware) che contengono sequenze di bit
    - Essendo hardware, contengono necessariamente un numero predefinito di bit (ovviamente non di più, ma neanche di meno)
  - Se il numero di bit è "cablato", un registro può contenere numeri fino a un massimo prefissato
  - Non possono esserci bit "vuoti" (la tensione è V_A o V_B)

Dentro al calcolatore... Informazione e memoria

- Una *parola di memoria* è in grado di contenere una *sequenza* di $n \geq 1$ bit
- Di solito si ha: $n = 8, 16, 32$ o 64 bit
- Una parola di memoria può dunque contenere gli *elementi d'informazione* seguenti:
  - Un carattere (o anche più di uno)
  - Un numero intero in binario naturale o in C_2
  - Un numero frazionario in virgola mobile
  - Alcuni bit della parola possono essere non usati
- Lo stesso può dirsi dei registri della CPU

Ad esempio...


indirizzi
parole da 32 bit
un carattere ASCII, probabilmente è un dato
quattro caratteri ASCII "impacchettati" nella stessa cella
Numeri la cui codifica richiede molti bit possono estendersi su più celle consecutive
un'istruzione? (perché no?)
0
1
2
3
4
5
Z
bit non usati
A b @ 1
1234 (in bin. nat.)
- 4321 (in C2)
19,758 (in virg. mob.)
...
la cella resta parzialmente inutilizzata
potrebbe essere un dato oppure l'indirizzo di un'altra cella (gli indirizzi sono intrinsecamente positivi)
probabilmente è un dato
probabilmente è un dato

Algebra di Boole
ed
Elementi di Logica

Cenni all'algebra di Boole

- L'*algebra di Boole* (inventata da G. Boole, britannico, seconda metà '800), o *algebra della logica*, si basa su *operazioni logiche*
- Le operazioni logiche sono applicabili a *operandi logici*, cioè a operandi in grado di assumere solo i valori ***vero*** *e* ***falso***
- Si può rappresentare *vero* con il bit **1** e *falso* con il bit **0** (convenzione di *logica positiva*)

Operazioni logiche fondamentali

- Operatori logici binari (con 2 operandi logici)
  - Operatore **OR**, o *somma logica*
  - Operatore **AND**, o *prodotto logico*
- Operatore logico unario (con 1 operando)
  - Operatore **NOT**, o *negazione*, o *inversione*

- Poiché gli operandi logici ammettono due soli valori, si può definire compiutamente ogni operatore logico tramite una ***tabella*** di associazione operandi-risultato

Operatori logici di base e loro tabelle di verità

A	B	A or B
0	0	0
0	1	1
1	0	1
1	1	1

(somma logica)

A	B	A and B
0	0	0
0	1	0
1	0	0
1	1	1

(prodotto logico)

A	not A
0	1
1	0

(negazione)

Le tabelle elencano tutte le possibili combinazioni in ingresso e il risultato associato a ciascuna combinazione

Espressioni logiche (o Booleane)

- Come le espressioni algebriche, costruite con:
  - Variabili logiche (letterali): p. es. A, B, C = 0 oppure 1
  - Operatori logici: and, or, not
- Esempi:

  A or (B and C)

  (A and (not B)) or (B and C)
- **Precedenza**: l'operatore "not" precede l'operatore "and", che a sua volta precede l'operatore "or"

  A and not B or B and C = (A and (not B)) or (B and C)
- Per ricordarlo, si pensi OR come "+" (più), AND come "×" (per) e NOT come "–" (cambia segno)

Tabella di verità di un'espressione logica

A	B	NOT ((A OR B) AND (NOT A))
0	0	1
0	1	0
1	0	1
1	1	1

Specificano i valori di verità
per tutti i possibili valori
delle variabili

Tabella di verità di un'espressione logica

A and B or not C

A B C	**X** = A and B	**Y** = not C	**X** or **Y**
0 0 0	0 and 0 = 0	not 0 = 1	0 or 1 **= 1**
0 0 1	0 and 0 = 0	not 1 = 0	0 or 0 **= 0**
0 1 0	0 and 1 = 0	not 0 = 1	0 or 1 **= 1**
0 1 1	0 and 1 = 0	not 1 = 0	0 or 0 **= 0**
1 0 0	1 and 0 = 0	not 0 = 1	0 or 1 **= 1**
1 0 1	1 and 0 = 0	not 1 = 0	0 or 0 **= 0**
1 1 0	1 and 1 = 1	not 0 = 1	1 or 1 **= 1**
1 1 1	1 and 1 = 1	not 1 = 0	1 or 0 **= 1**

A che cosa servono le espressioni logiche?

- A *modellare* alcune (non tutte) forme di *ragionamento*
  - A = è vero che 1 è maggiore di 2 ? (sì o no, qui è no) = 0
  - B = è vero che 2 più 2 fa 4 ? (sì o no, qui è sì) = 1
  - A and B = è vero che 1 sia maggiore di 2 e che 2 più 2 faccia 4 ?
    Si ha che A and B = 0 and 1 = 0, dunque no
  - A or B = è vero che 1 sia maggiore di 2 o che 2 più 2 faccia 4 ?
    Si ha che A or B = 0 or 1 = 1, dunque sì
- OR, AND e NOT vengono anche chiamati *connettivi logici,* perché funzionano come le congiunzioni coordinanti “o” ed “e”, e come la negazione “non”, del linguaggio naturale
- Si modellano ragionamenti (o *deduzioni*) basati solo sull’uso di “o”, “e” e “non” (non è molto, ma è utile)

Che cosa **non** si può modellare tramite espressioni logiche?

- Le espressioni logiche (booleane) *non modellano*:
  - Domande *esistenziali*: "***c'è almeno*** un numero reale x tale che il suo quadrato valga –1 ?" (si sa bene che *non c'è*)
    $\exists x \mid x^2 = -1$ è falso
  - Domande *universali*: "***ogni*** numero naturale è la somma di quattro quadrati di numeri naturali ?" (si è dimostrato *di sì*)
    $\forall x \mid x = a^2+b^2+c^2+d^2$ è vero ("teorema dei 4 quadrati")

    Più esattamente andrebbe scritto: $\forall x \; \exists a,b,c,d \mid x = a^2+b^2+c^2+d^2$
- $\exists$ e $\forall$ sono detti "operatori di quantificazione", o quantificatori
- La logica che usa solo or, and, not è il ***calcolo proposizionale***
- Aggiungendo gli operatori di quantificazione si ha il ***calcolo dei predicati*** (che è una logica molto più complessa)