Laboratorio di Informatica

per chimica industriale e
chimica applicata e ambientale

LEZIONE 2

Rappresentazione delle informazioni:
numeri e caratteri

Codice

- La relazione che associa ad ogni successione ben formata di simboli di un *alfabeto* il dato corrispondente è detta **codice**.
- Un codice mette quindi in relazione le successioni di simboli con il significato loro attribuito.

Rappresentazione delle informazioni

- I calcolatori moderni sono detti **calcolatori digitali**
- *digit* in inglese significa “cifra”
- tutte le informazioni vengono rappresentate in forma numerica:
  - numeri
  - caratteri
  - immagini statiche e in movimento
  - suoni

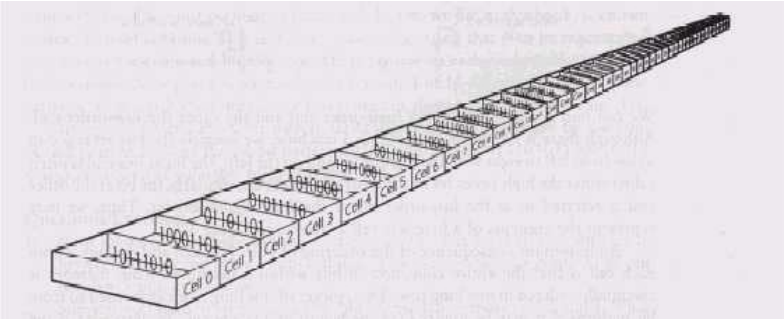
Rappresentazione delle informazioni

- Noi rappresentiamo i numeri utilizzando un alfabeto di dieci simboli (0, 1, …, 9): rappresentazione posizionale decimale (in **base** dieci).
- Per i calcolatori si usa un alfabeto binario:
- Alfabeto binario: costituito da due simboli
  - convenzionalmente “0” e “1”
- binit o **bit** (*binary digit*, cifra binaria): elemento che assume un valore binario
- il **bit** è anche l’unità elementare di informazione

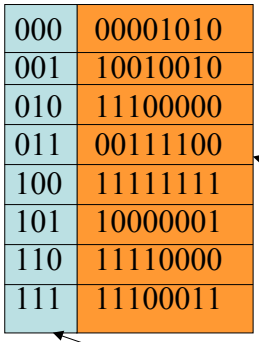
Rappresentazione delle informazioni

- Nella memoria del computer è possibile memorizzare esclusivamente sequenze di bit e il processore può processare soltanto dati espressi come sequenze di bit.

Le memorie centrali



Le memorie “usano” i bit



000	00001010
001	10010010
010	11100000
011	00111100
100	11111111
101	10000001
110	11110000
111	11100011

Logicamente sono organizzate in **celle indirizzabili**

Le celle contengono tutte lo stesso numero di bit, detto **lunghezza**

Esempio:
celle di lunghezza 8

Gli indirizzi hanno una lunghezza fissata, che determina ***l’espansione massima della memoria***

Esempio: con indirizzi di 3 bit
si hanno al massimo 8 celle

Dimensioni tipiche

- Celle di memoria
  - 8 bit, 16 bit
- Indirizzi
  - da 16 bit a 64 bit
- Ma quante sono le celle indirizzabili con n bit? Impariamo ora
  - a fare qualche conto
  - a conoscere le unità di misura
  - come si rappresentano i dati numerici

Rappresentazione delle informazioni

- La rappresentazione dei caratteri e dei numeri
- Vedremo in altre lezioni la rappresentazione di altri tipi di informazione

La codifica binaria

- Avendo a disposizione un solo bit si possono rappresentare
  - **due** elementi diversi:
  - si assegna al primo la ***codifica* 0**
  - al secondo la ***codifica* 1**

La codifica binaria

- Con 2 bit si possono rappresentare **$4 = 2^2$** elementi (oggetti, individui, ...) diversi, assegnando a ciascuno una ***codifica*** diversa:
  - **paperino codifica 00**
  - **qui codifica 01**
  - **quo codifica 10**
  - **qua codifica 11**
- con 3 bit si possono rappresentare **$8 = 2^3$** elementi diversi
- …..
- **con n bit si possono rappresentare 2^n elementi diversi**

Unità di misura: il byte

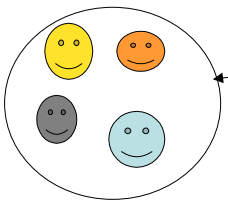
- ***Una sequenza di 8 bit è detta byte, ed è una unità di misura della occupazione di memoria***
- La scelta del byte ha ragioni storiche
  - per codificare le lettere dell'alfabeto, cifre da 0 a 9, lo spazio, la virgola, …, per un numero totale compreso tra 90 e 120, sono necessari almeno 7 bit (2^7=128)
  - se ne utilizzano 8: i 7 bit indispensabili + 1 bit detto di parità
- 8 è una lunghezza tipica delle celle di memoria

Rappresentazione dei caratteri

- Il codice utilizzato da sempre per la codifica binaria dei caratteri è il codice ASCII, che associa ad ogni carattere una sequenza di 7 bit (ASCII standard).
  - 128 caratteri (cifre, lettere minuscole e maiuscole, punteggiatura, caratteri speciali)
  - La codifica rispetta l'ordine alfanumerico
- Un byte contiene 8 bit - ASCII esteso, non standard
  - 256 caratteri (+ lettere accentate, ecc.)
- Un codice a 8 bit non è però sufficiente se si vuole rappresentare un insieme di caratteri di cardinalità maggiore di 256, come per esempio quello dei caratteri cinesi.
- Per questo motivo è stato introdotto un codice a 32 bit per la rappresentazione dei caratteri, chiamato *Unicode*, che permette di rappresentare fino a 2^{32} elementi distinti.

Rappresentazione dei numeri

I numeri sono entità matematiche astratte; vanno distinti dalla loro rappresentazione



Il numero cardinale **quattro**: cardinalità degli insiemi contenenti quattro elementi

rappresentazione 1: **IV**

rappresentazione 2: **4**

ecc.

Codifica - numeri naturali

- **Sistema di numerazione arabico**:
  - introdotto in Europa nel Medio Evo
  - **in base dieci**: utilizza le dieci cifre 0, 1, …, 9
  - è una **notazione posizionale**: il valore di ogni cifra dipende dalla sua posizione nella successione di simboli che rappresenta il numero.
- Es. $1234 = 1x10^3 + 2x10^2 + 3x10^1 + 4x10^0$

Codifica - numeri naturali

- In generale, per un numero composto di *n* cifre si ha che:

$$c_{n-1}\, c_{n-2} \ldots c_1\, c_0 =$$
$$= c_{n-1}x10^{n-1} + c_{n-2}x10^{n-2} + \ldots + c_{n-2}\ldots c_1 x10^1 + c_0 x10^0$$

- Si chiamano **cifre piú significative** quelle associate ai pesi maggiori.
- La cifra c_{n-1} è la piú significativa e c_0 è la cifra meno significativa.

Codifica - numeri naturali

- **Sistema di numerazione additivo**:
  - il significato dei simboli che compongono un numero è indipendente dalla posizione in cui compaiono
  - Es. sistema con un unico simbolo, per l'unità
  - Es. sistema di numerazione romano, con i simboli I, V, X, L, C, D, M.

Codifica - numeri naturali

Sistemi di numerazione in base B

• Se la base della numerazione è **B**, si hanno a disposizione **B** cifre, comprese tra **0** e **B–1**.

$c_n\, c_{n-1} \ldots c_1\, c_0 =$

$= \mathbf{c_n \cdot B^n + c_{n-2} \cdot B^{n-1} + \ldots + c_1 \cdot B^1 + c_0 \cdot B^0}$

$= \Sigma_{i:0..n} \ \mathbf{c_i \cdot B^i}$

Codifica binaria - numeri naturali

- Esempio:

$101100_{due} = (1x2^5 + 0x2^4 + 1x2^3 + 1x2^2 + 0x2^1 + 0x2^0)_{dieci}$

$= (32 + 8 + 4)_{dieci} = 44_{dieci}$.

- Tramite m cifre in base B è possibile rappresentare B^m numeri naturali, da 0 a B^{m-1}.

Aritmetica finita

- A livello HDW i calcoli numerici vengono eseguiti dalla
  - ALU Unità Logico Aritmetica
  - che si trova nella CPU e lavora con i registri della CPU
- Siccome i registri hanno lunghezza prefissata L, numeri la cui rappresentazione binaria richiede più di L cifre non sono rappresentabili.

Aritmetica finita

- Si usa dunque un'aritmetica finita, cioè con un numero massimo di cifre binarie disponibili;
- Nell'aritmetica finita dei calcolatori
  - i numeri relativi sono rappresentati in complemento, come vedremo
  - i numeri "reali" sono rappresentati in virgola mobile, come vedremo
- Siccome il numero di cifre massimo è limitato, la precisione raggiungibile nella rappresentazione dei numeri reali è limitata; abbiamo le seguenti precisioni, che spiegheremo poi:
  - semplice 32 bit
  - doppia 64 bit
  - estesa 128 bit

Codifica binaria - numeri naturali

- Con una successione di n bit si rappresentano i 2^n numeri naturali, da 0 a 2^n-1
- Per i numeri naturali si usano di solito 32 bit
  - $2^{32}-1 = 4.294.967.295 \cong 4x10^9$
- Raddoppiando la lunghezza, il massimo numero rappresentabile aumenta esponenzialmente. Se si utilizzano 64 bit
  - $2^{64}-1 \cong 1{,}6x10^{19}$

I primi 16 numeri binari

0	= **0**	1000	= **8**
1	= **1**	1001	= 9
10	= **2**	1010	= **10**
11	= **3**	1011	= **11**
100	= **4**	1100	= **12**
101	= **5**	1101	= **13**
110	= **6**	1110	= **14**
111	= **7**	1111	= **15**

Sistemi di num. usati in informatica

- Base **2**: quella in cui lavora il calcolatore
  - cifre 0,1
- Base **10**: quella dell'utente umano
  - cifre 0,1,2,3,4,5,6,7,8,9
- Base **8**: per abbreviare i numeri binari
  - cifre 0,1,2,3,4,5,6,7
- Base **16**: per abbreviare i numeri binari
  - cifre 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Fare i conti: proprietà notevoli

- (pn1) **1** seguito da **n 0** rappresenta B^n; ad es.
  - base 2: 100000 = 2^5
  - base 10: 100000 = 10^5
  - base 8: 100000 = 8^5
  - base 16: 100000 = 16^5
- (pn2) **n *cifre massime*** rappresentano B^n-1; ad es:
  - base 2: 11111 = $2^5 - 1$
  - base 10: 99999 = $10^5 - 1$
  - base 8: 77777 = $8^5 - 1$
  - base 16: FFFFF = $16^5 - 1$

Dalla rappresentazione al numero

- Si applica la definizione
  - $c_n\, c_{n-1} \ldots c_1\, c_0 = c_n \cdot B^n + c_{n-2} \cdot B^{n-1} + \ldots + c_1 \cdot B^1 + c_0 \cdot B^0$
- Esempi
  - base 2: $1011 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 =$
  - base 8: $2705 = 2 \cdot 8^3 + 7 \cdot 8^2 + 0 \cdot 8^1 + 5 =$
  - base 16: $3F01 = 3 \cdot 16^3 + 15 \cdot 16^2 + 0 \cdot 16^1 + 1 =$

Dal numero alla rappresentazione

- Si usano divisione intera **div** e resto **mod**, e si applica la:
- ***proprietà notevole delle rappresentazioni in base*** **B**:
  - **n mod B** è rappresentato dalla cifra $\mathbf{c_0}$ meno significativa della rappresentazione di **n** in base **B**
  - **n div B** è rappresentato dalle cifre precedenti
  - Ad es., nella base 10:
    - **1537 mod 10 = 7** è rappresentato da **7**
    - **1537 div 10 = 153** è rappresentato da **153**
- *La rappresentazione emerge attraverso divisioni intere successive, raccogliendo i resti, che corrispondono alle cifre, partendo da quella meno significativa*

Dal numero alla rappresentazione

Ad esempio, se la base è 2,

si ripete la divisione intera per 2

e si raccolgono i quozienti interi ed i resti come cifre

numero	**div** base	quoz.	resto	
6	div 2 =	3	0	Cifra posto 0
3	div 2 =	1	1	Cifra posto 1
1	div 2 =	0	1	Cifra posto 2

6 = 110 ← Le cifre ottenute corrispondono alla rappresentazione binaria

Dal numero alla rappresentazione

numero	divisore	quoziente	resto	
71	2	35	1	cifra bin. meno significativa
35	2	17	1	
17	2	8	1	
8	2	4	0	
4	2	2	0	
2	2	1	0	
1	2	0	1	cifra bin. piú significativa

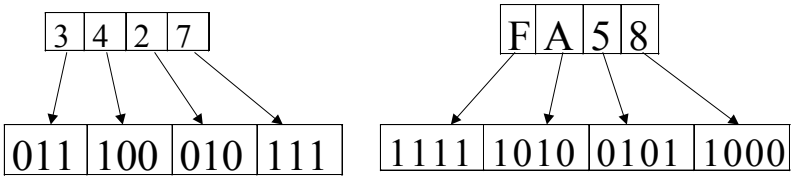
$71_{dieci} = 1000111_{due}$

Conversioni basi 8,16 - base 2

- La conversione dalla base 2 alla base 8 può essere fatta per parti, considerando volta per volta una tripla di cifre binarie.
- Es. 001.010.110.111 = 1267_{otto}
- La conversione dalla base 2 alla base 16 può essere fatta per parti, considerando volta per volta una quadrupla di cifre binarie.
- Es. 0010.1011.0111 = $2B7_{sedici}$

Conversioni basi 8,16 - base 2

- Le basi 8 e 16 si usano perché si hanno le conversioni illustrate in figura:



Si ha cioè un passaggio molto semplice

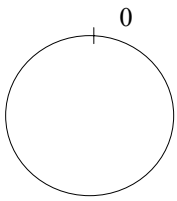
lunghezze base 10 - base 2

- La situazione non è così semplice per la base 10, perché 10 non è potenza di 2
- La prima potenza di 10 vicina ad una potenza di 2 è:
  - $10^3 \sim 2^{10} = 1024$
- Per questo 1 K = 1024
  - $1\ K = 2^{10}$
  - $1\ K \sim 10^3$

Aritmetica finita

- La rappresentazione grafica dei numeri sulla retta permette di comprenderne alcune proprietà

- Per comprendere le proprietà dell'aritmetica finita occorre passare alla rappresentazione su circonferenza



Codifica binaria - interi relativi

- **Codifica con modulo e segno**: consiste nell'indicare il segno seguito dal valore assoluto, come succede normalmente nella codifica decimale.
- Il primo bit indica il segno
  - 0 per positivo
  - 1 per negativo
- Gli altri n–1 bit rappresentano il valore ass.

Codifica binaria - interi relativi

- **Codifica con modulo e segno**
- Esempi

  0011 = 3

  0000 = 0

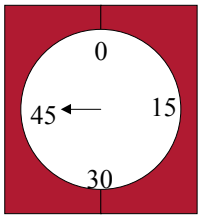
  1000 = – 0

  1011 = – 3

- Ha il difetto di duplicare la rappresentazione del numero 0, cosa che può complicare l'esecuzione ed il controllo delle operazioni aritmetiche.

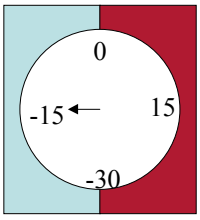
Interi relativi in complemento

- ***In complemento a 60***, si ha l'aritmetica dell'orologio senza la lancetta delle ore (segna sempre l'ora zero)



Interi assoluti

da 0 a 59



negativi: quanto manca all'ora?

da 0 a 29 non negativi
da -30 a -1 negativi

Codifica binaria - interi relativi

- **Rappresentazione in complemento a due**: dati *n* bit, un numero negativo $-x$ si rappresenta con il valore binario corrispondente a $2^n - x$.
- Ad esempio, con successioni di 4 bit
  - 0000 0 1101 – 3
  - 0001 1 1110 – 2
  - 0010 2 1111 – 1

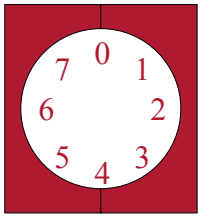
Codifica binaria - interi relativi

Complemento a 2 con 3 bit

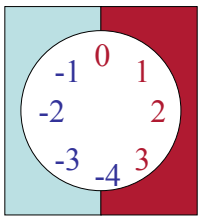
- Se usiamo una cella di 3 bit abbiamo
  - 000 $\rightarrow_{+1}$ 001 $\rightarrow_{+1}$ 010 $\rightarrow_{+1}$ 011 $\rightarrow_{+1}$ 100 $\rightarrow_{+1}$ 101 $\rightarrow_{+1}$ 110 $\rightarrow_{+1}$ 111
  - 111 $\rightarrow_{+1}$ 1000;
  - ma, siccome abbiamo solo 3 bit: 111 $\rightarrow_{+1}$ 000
  - cioè si torna a 0, come nell'orologio dopo il minuto 59
  - Dunque una cella di 3 bit è
  - come un orologio con ore di $8 = 2^3$ minuti

Complemento a 2 con 3 bit

- Cioè, in complemento a 2 con 3 bit significa
  - in complemento a 8 = 2^3



Valori assoluti



Valori in complemento;
per i blu: quanto manca a 8?

Complemento a 2 con 3 bit: rappresentazione binaria e bit del segno

n assoluto — **n** in compl. a 2

000	**0**	**0**
001	**1**	**1**
010	**2**	**2**
011	**3**	**3**

100	**4**	**-4**
101	**5**	**-3**
110	**6**	**-2**
111	**7**	**-1**

1° bit
0: +

1° bit detto
bit del segno

1° bit
1: -

Codifica binaria - interi relativi

Rappresentazione in complemento a due

- Con n bit possono essere rappresentati gli interi compresi tra -2^{n-1} e $+(2^{n-1} - 1)$.
- Operativamente, si complementano i bit della rappresentazione binaria del modulo x del numero e si somma 1 al risultato
- Ad esempio, con successioni di 4 bit (2^4=16)

  $5_{dieci} = 0101$

  $-5_{dieci} = 1010 + 1 = 1011$

Codifica binaria - interi relativi

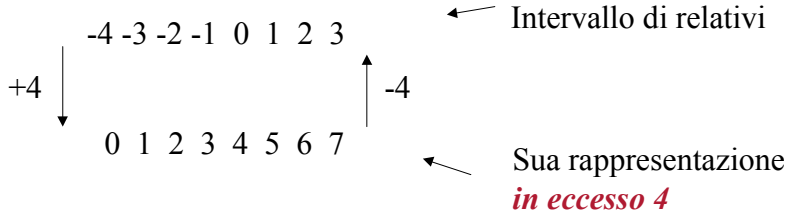
Complemento a 2 e a 1

- ***In complemento a 2 con n bit***: complemento a $\mathbf{2^n}$
  - $0 .. (2^{\mathbf{n-1}} - 1)$ rappresentano $0 .. (2^{\mathbf{n-1}} - 1)$
    - Hanno bit del segno = 0
  - $2^{\mathbf{n-1}} .. (2^{\mathbf{n}} - 1)$ rappresentano $-2^{\mathbf{n-1}} .. -1$
    - Hanno bit del segno = 1

- *In complemento a 1 con n bit*: come sopra, ma ora il complemento è a $2^n - 1$

Interi relativi in eccesso (o biased)

- Per rappresentare un intervallo -N..+(N-1) mediante positivi si può traslarlo di +N
  - -N..+(N-1) traslato di +N diventa 0..2N-1
  - La traslazione **N** è detta ***eccesso o bias***
  - Esempio:



Codifica binaria - interi relativi
in eccesso (o biased)

- **Con n bit:**
  - eccesso = 2^{n-1};
  - talora eccesso = $2^{n-1} - 1$

- Esempio con **8 bit**
  - **eccesso = 128**
    - -128..+127 $_{-128}\leftrightarrow_{+128}$ 0..255
  - talora eccesso = 127
    - -127..+127 $_{-127}\leftrightarrow_{+127}$ 0..254
      - 255 è trattato come 0

Gli errori di overflow

- L'intervallo dei numeri interi (assoluto o relativi) rappresentabili con un fissato numero di bit è limitato
- Eseguendo un'operazione su numeri rappresentabili, può accadere che il risultato esca dall'intervallo rappresentabile; in tal caso si dice che
  - ***si ha un errore di overflow***
- ESEMPIO con 8 bit
  - Caso di valori assoluti: massimo rappresentabile = 255
    - Overflow 150 + 150 = 300 perché > 255
  - Caso complemento a 2: intervallo rappr.: -128..127
    - Overflow -100 - 100 = -200 perché < -128

Numeri frazionari

Rappresentazione con la virgola:

$$c_n..c_0,c_{-1}..c_{-k} = c_nB^n + .. + c_0B^0 + c_{-1}B^{-1} + .. + c_{-k}B^{-k}$$

Dalla rappresentazione al numero

- Per passare dalla rappresentazione in base B al numero si applica la def.:
  - $c_n..c_0,c_{-1}..c_{-k} = \mathbf{c_n B^n + .. + c_0 B^0 + c_{-1} B^{-1} + .. + c_{-k} B^{-k}}$
- Esempio col numero binario 101,01
  - tralasciando il contributo nullo delle cifre 0:
    101,01 = $\mathbf{1 \cdot 2^2 + 1 \cdot 2^0 + 1 \cdot 2^{-2} = 4 + 1 + 1/4 = 5{,}25}$

Dal numero alla rappresentazione

- Si trattano separatamente parte intera e frazionaria; per la intera, si procede come già visto;
  - per la frazionaria, si applica la proprietà:
    - sia 0, $c_{-1}c_{-2}..c_{-k} = \mathbf{n}$; allora
    - $c_{-1}, c_{-2}..c_{-k} = \mathbf{B \cdot n}$
    - Esempio in decimale: $\mathbf{0{,}231 \cdot 10 = 2{,}31}$
      moltiplicando per la base **10** emerge la cifra **2**
- Cioè, per far emergere c_{-1} basta moltiplicare per **B**; moltiplicando ancora per **B** emergerà c_{-2} e così via

Dal numero alla rappresentazione

Si riporta la parte frazionaria della riga precedente

Si riporta la parte intera come cifra

Contiene le cifre binarie, nell'ordine

parte fraz.	× base	prod.	parte intera
0,25	**× 2 =**	**0,5**	**0**
0,5	**× 2 =**	**1,0**	**1**
0	**× 2 =**		

Ci si arresta quando la parte frazionaria si azzera o quando abbiamo un numero di cifre abbastanza elevato

0,25 = 0,01

Sui numeri periodici

- Un numero può essere non periodico in una base e perodico in un'altra
  - esempio da base 10 a 2: **0,4** = 0,0110011001100….

0,4	**2**	**0,8**	0
0,8	**2**	**1,6**	1
0,6	**2**	**1,2**	1
0,2	**2**	**0,4**	0
0,4	**2**	**0,8**	0

Si ripete la prima riga!

Codifica - numeri razionali

- **Notazione scientifica**: un numero viene rappresentato come

  $\pm m \times 10^{p}$ Es. $123.000.000 = 1{,}23 \times 10^{8}$

- Se, piú in generale, la base è B,

  $\pm m \times B^{p}$

- Il coefficiente m è detto **mantissa** (la convenzione è di inserire implicitamente la virgola decimale subito dopo la prima cifra).
- P, detto **caratteristica**, è l'esponente a cui elevare la base B.

Es. $1{,}23 \times 10^{8}$ base 10, segno +, mantissa 123, caratteristica 8

Codifica binaria - numeri razionali

- La rappresentazione binaria dei numeri razionali che usa la notazione scientifica è detta **rappresentazione in virgola mobile** (*floating point*).
- $101010000_{due} = 1{,}0101 \times 10^{1000}{}_{due}$
- Cambiando il numero di cifre dedicato alla rappresentazione di mantissa ed esponente cambia la precisione dei risultati che si ottengono.
- L'interesse a uniformare la precisione di calcolo ha condotto alla definizione di uno standard internazionale proposto dall'*Institute of Electrical and Electronic Engineers* (IEEE).

Virgola mobile (floating point)

- Virgola mobile: **m E e** con
  - mantissa **m** per numeri diversi da 0: **1 ¡ m < B**
  - **e** esponente
  - significato **m E e** = $m \cdot B^e$
  - Esempio decimale: 344,013 in virgola mobile si scrive:
    - 3,44013 E 3 = $3{,}44013 \cdot 10^3$
  - Esempio binario: 10,1001 in virgola mobile si scrive:
    - 1,01001 E 10 = $(1+2^{-2}+2^{-5}) \cdot 2^2$

Rappresentazione floating Point: precisione singola, 32 bit

1 bit segno
0 +
1 -

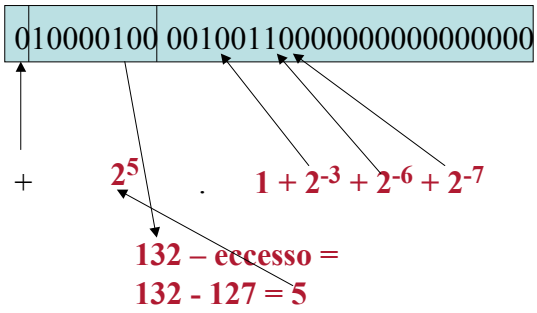
8 bit per l'esponente in eccesso 127

l'esponente rappresentato è
E = (e-127)

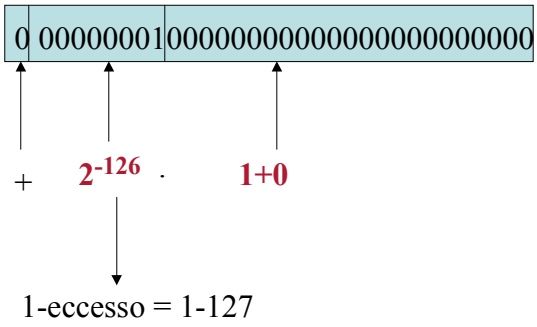
23 bit per la parte frazionaria F della mantissa

La mantissa è 1,F

esempio



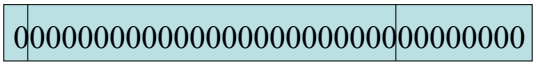
Il minimo positivo rappresentabile con precisione singola



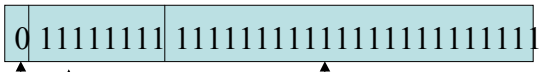
Lo zero?

Perché per il minimo positivo normalizzato si usa l'esponente
00000001 (che rappresenta –126) e non
00000000 (che rappresenta –127)?

Risposta: i numeri normalizzati hanno mantissa
M = 1,F
e quindi sempre >=1; per rappresentare lo 0
si usa un truccaccio: l' esponente 0000000 serve
per lo zero, che è:



Il massimo positivo rappresentabile con precisione singola



$+ \quad 2^{127} \quad \cdot \quad (1 + \Sigma_{i:1..23} \; 2^{-i})$

$= \Sigma_{i:0..23} \; 2^{127-i}$

Codifica - numeri reali

- La rappresentazione esatta di un numero irrazionale richiederebbe un numero infinito di cifre
  - Es. $1/3 = 0{,}3333333\ldots$ $\pi = 3{,}14159\ldots..$
- Un problema analogo sorge per numeri con valore assoluto molto grande o molto piccolo, in cui il numero di cifre richiesto non sarebbe infinito, ma molto elevato
- In questi casi possiamo considerare solo le cifre piú significative.

Codifica - errori di arrotondamento

- Qualunque sia la codifica scelta, la rappresentazione dei numeri nel calcolatore è soggetta ad **approssimazioni**.
- Il limitato numero di cifre disponibili nella mantissa porta ad errori di arrotondamento quando si debbano rappresentare numeri con una mantissa più lunga.
- Tali approssimazioni si propagano nel corso della esecuzione delle operazioni causando **errori numerici** anche importanti.
- Il **calcolo numerico** è la disciplina che studia le proprietà dell'esecuzione delle operazioni tramite calcolatore e valuta l'entità degli errori introdotti durante l'esecuzione.