

Standard concordato per fare in modo che dispositivi di produttori diversi possano comunicare fra loro.

Prevede 2 rappresentazioni possibili dei numeri reali:

- 32 BIT
- 64 BIT

cio' cambia i valori
(valore max e min)
dei numeri rappresentabili

→ a loro volta si distinguono 2 tipi di rappresentazioni

- FORMA NORMALIZZATA
- FORMA NON NORMALIZZATA
(per rappresentare i numeri più piccoli)

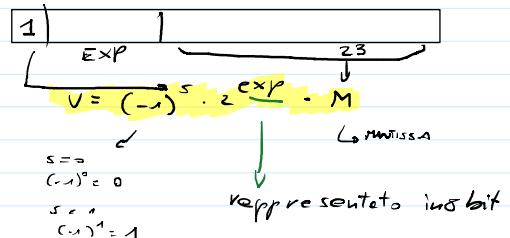
E a loro volta si distinguono le rappresentazioni
in forma normalizzata e in forma non normalizzata

FORMA NORMALIZZATA



VALORE (MODULO)

Rappresento n. negativo in virgola mobile



per inserire il valore $v=1 \rightarrow$

ma posso rappresentare 1 anche con $v = (-1)^S \cdot 2^{EXP} \cdot M$

ovvero posso ottenerlo con $1 \cdot 1 = 2 \cdot \frac{1}{2} = 4 \cdot \frac{1}{4} \dots$

che io intendo evitare (normalizzazione)

Per normalizzare pongo un vincolo, es. $M \geq 1$ dunque $v=1 = 1 \cdot 1$ ok ✓

$$\begin{aligned} &= 2 \cdot \frac{1}{2} \text{ NO!} \\ &= 4 \cdot \frac{1}{4} \text{ NO!} \end{aligned}$$

$$V = 3 = 1 \cdot 3 = \frac{1}{2} \cdot 6 = \frac{1}{2} \cdot 12$$

unica scrittura

normale

$$\begin{array}{|c|c|} \hline 2^0 & 2^1 & 2^2 & 2^3 \\ \hline 1 & 1 & & \\ \hline \end{array}$$

$M \geq 1$

$$\begin{array}{|c|c|c|c|c|} \hline 2^0 & 2^1 & 2^2 & 2^3 \\ \hline 1 & 1 & 1 & 1 \\ \hline \end{array}$$

$1 \leq M < 2 \rightarrow$ può essere
maggiore di 1
ma con potenze
 2^n con $n < 0$

REGOLA NEW

Quindi posso dare per scontato

REGOLA DELLE
STANDARD IEEE-754

SEMPRE A

Quindi posso dare per scontato
che la prima cifra sia 1, posso
non scriverla

maggiore di s
ma con potenze
z in cui n < 0
quindi < 2

BIT	1	8	z^{-1}	z^0	z^1	\dots	z^3	$\dots z^{-23}$
Dunque			1	0	1			

$$M = 1 + M'$$

codifica in
forma normalizzata
 $1 \leq M < 2$

Rappresento modulo e segno:
in M rappresento tutti zen,
mentre implicitamente mi
rimane 1.

Alcuni valori sono normalizzabili,
altri, come zero, non lo sono.

Per rappresentare numeri in forma non normalizzata si codifica l'esponente (exp)

uso una rappresentazione in eccesso a 128 $m = -1 = 127$
(perché in 8BIT)

per rappresentare
 $E = +2 \rightarrow E + 127 = 129$ e lo rappresento nei 8BIT
(exp)

1 0 0 0 0 0 0 0

Dunque ho $S = 0$, l'exp ottenuto è $M = 0 + 1$

$$\text{il valore } v = +z^0 \cdot 1 = 1$$

d'exp può assumere i valori: $-126 \leq EXP \leq 127$

Il valore -127 in eccesso 127 è tutti zen
ma non è ammissibile, in quanto
non è normalizzabile

in realtà posso codificare
da -127 a 128 ma:

- -127 si usa x la forma
non normalizzata
- 128 si usa x i casi
di "errore" (vedi dopo)

FORMA NON NORMALIZZATA

Rappresentare n. in forma non normalizzata significa M rappresentabile
con $M < 1$

Dunque $M = 0 + M'$

$z^{-1} z^0 z^1 z^2 \dots z^{-23}$

anche lo
zero

possò
rappresentare
n. molto piccoli

possò rappresentare
 z^{-127} e prendere i bit successivi

equivalente

zero



posso rappresentare

z^{-127} e prendere i bit successivi

$$z^{-127} \cdot z^{-1} = z^{-128}$$

$$\dots z^{-123} \dots z^{-129}$$

ovvero tutti i numeri minori di z^{-127}

equivale
a una
virgola
fissa

Posso dunque rappresentarlo con tutti zero.

- Se exp ha 8 bit con tutti 1, nell'eccesso 127 non posso rappresentare -127

Tale standard pone 3 simboli con cui rappresentare certi valori; (valori di "errore")

+∞ -∞ NaN

La divisione per 0 permette, secondo lo standard floating point, di fornire un risultato comunque significativo

- ottengo +∞, rappresentato da 127

- per -∞ cambio i 1 bit di segno

- NaN (Not a Number, e nemmeno +∞ e -∞) si può generare come per 0%. Esso viene codificato con 1100 in exp e con

base 10
un solo
bit su
l'uno

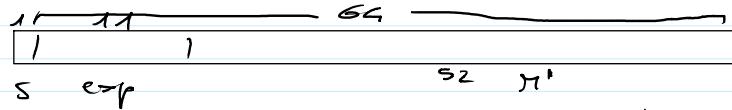
→ valore in M'FO; il bit di segno non viene preso in considerazione in questo caso.

Osservazione: NaN è ottenibile con moltissime combinazioni (basta un solo (o più) 1 nelle strings di M')

N.B. Posso anche rappresentare +0 (tutti zero) e -0 (con 1 nel segno rispetto alla rappresentazione di +0)

Rappresentazione a 64 bit: i combi solo la quantità di bit riservati

IEEE 754



ho eccesso 1023 posso scrivere $-1022 \leq \text{exp} \leq 1023$

n. picc. 0

potenze grandi

NOTE.

exp. 0 zero (64 zero) e n. piccoli $\rightarrow -1023$ (tutti zero) NaN NOTE.

$$+1024 \begin{cases} +\infty & S=0, M'=0 \\ -\infty & S=1, M'=0 \\ \text{NaN} & S=1, \text{almeno un } 1 \text{ in } M' \end{cases}$$

corrisponde a:
32 bit \rightarrow float
64 bit \rightarrow double

Dal punto di vista pratico è comodo avere sia la forma normalizzata che non, oltre al fatto di poter avere un "feedback" su eventuali valori non numerici (di "errore").

Inoltre, se penso a 64 bit, penso a 2^{64} numeri possibili, poiché non tutte le codifiche sono dei valori numerici. L'idea di attribuire un'efficienza (buon funzionamento) al codice fa riferimento al concetto di **ridondanza**.

85

$$328\pi \cdot \frac{1}{2} \times 100000 = 110 \text{ cm}^3$$

$$+ \quad \downarrow \quad M' = \frac{1}{2} + \frac{1}{4} = \frac{3}{4} \quad \text{or} \quad M = M' - 1 = \frac{3}{4} - 1 = \frac{1}{4}$$

F. NO RNA LIBERATA?

5) ALTRIMENTI SANCTESE VITI ZEMI (costante -127)

$$\downarrow \\ \text{EXP} i 32 + 64 = 36 \rightarrow 36 - 127 = -31$$

$$+ z^{-31} \cdot \underline{\frac{7}{16}} = 7 \cdot z^{-33}$$

$$H = \begin{pmatrix} H' \\ 0 \end{pmatrix} + \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$$

$$\text{EXP} = 178 + 99 + 1 = 183 - 177 = 66$$

$$-1 \cdot 2^{66} \cdot 1 = -2^{66}$$

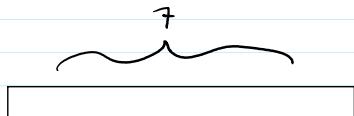
$$F. \begin{matrix} \downarrow \\ \text{NON NON RUE.} \end{matrix}$$

C 0 0 6 | 1 1 1 1 1 1 1 | C

$$643216842 \rightarrow M = 0 + 1 = 1$$

$$\text{Exp} = 127 - 127 = 0 \quad + 2^0 \cdot 1 = 1$$

$$1 \mid 10000000 | 00 \dots \quad 9 \quad \equiv -2$$



CODECE LUNGHEZZA FISSA | = equivale alla quantità
(es. 7 bit) | di risorse usate nel
nostro codice

N° COMBINAZIONI POSSIBILI è FISSATO : 2^7 → il codice
(ogni bit assume 0 o 1) ← 128 → può considerare fino a
 2^7 numeri diversi

ma non sempre è necessario

Usa tutte le codece per scrivere dei:
caratteri, facendo riferimento
alla tabella ASCII

?	a	A	.
!	b	B	?
?	c	C	;
,	.	.	.
,	,	,	{
,	,	,	}
3	z	Z	

il numero di
combinazioni
usate deve
essere \leq N° comb. possibili → × essere
(difficile usarli tutti) EFFICIENTE

Ridondanza

$$R = \frac{\text{N}^{\text{°}} \text{ comb. possibili}}{\text{N}^{\text{°}} \text{ comb. usate}} \geq 1$$

→

$$\text{es. ASCII con 30 caratteri } R = \frac{128}{30}$$

SOLITAMENTE SI USANO
CODECI CON $R < 1$:

$$1 \leq R < 2$$

dello codice minima

✓ 2^6 non bastano per rappresentare 30 elementi,
allora usare 2^7 ma mi accorgo che
già ho sprecato dell. spazio
($128 > 30$)

Se $R \geq 2$ il codice è
ridondante e può essere
rappresentato con un bit
in meno, senza
perdita di informazioni

Il problema della ridondanza appare nella rappresentazione di:

∞, -∞, NaN
↓
1 combinazione
↓
1 comb.

in particolare
questo "errore"
ha tante rappresentazioni
binarie diverse

es. tutti i valori 1
oppure tutti meno
l'ultima cifra,
sono 1

$$\text{Laha } (2^{23} - 1) \cdot 2_{\text{bit}}$$

$$R = \frac{2^4}{3} \rightarrow \text{comb. possibili}$$

3 → comb. usate

tale ridondanza però mi permette, in questo codice, di determinare se si tratta di un numero oppure no guardando solo l'exp

CONFRONTO

Sicuramente evitabile, ma l'obiettivo del codice non è rappresentare i non numeri, ma i numeri, poi si è deciso di rappresentare anche i non numeri, perciò come risorse.

→ posso
da una
grande ridondanza

→ si accetta perché è una parte marginale, il codice non viene principalmente usato per questa rappresentazione

N.B.
Posso certo costruire un codice che faciliti la rappresentazione
Di non numeri, ma così avrei una ridondanza e difficoltà nella rappresentazione dei numeri (obiettivi diversi, difficoltà diverse)

Rappresentazione e compressione di informazioni

venerdì 9 ottobre 2020 16:29

$$\text{Ridondanza } R = \frac{\text{N}^{\circ} \text{ comb. possibili}}{\text{N}^{\circ} \text{ comb. usate}} \geq 1$$

(perché le usate sono al massimo uguali alle possibili, non possono essere maggiori (non possono usare cose non esistenti))

poss. ogive
sul minimo n° di BIT per poter rappresentare tutti gli elementi necessari

2)

Voglio rappresentare 65 caratteri, devo usare \geq 6 BIT ovvero: 2^7 comb. possibili

Cio' utilizzando un codice a lunghezza fissa

$$R = \frac{128}{65}$$

- Posso migliorare se uso un codice con lunghezza variabile

NUMERAZIONE ROMANA

I	= 1
V	= 5
X	= 10
L	= 50
C	= 100
D	= 500
M	= 1000

con 7 simboli: posso rappresentare le date

Si hanno delle regole per rappresentare i numeri:

- all'interno vi sono anche delle operazioni di somma o sottrazione (l'ordine conta)

la funzione non è crescente con l'aumentare dei valori (nei decrescenti)

$$\begin{array}{ll} 3 \text{ simboli } III = 3 \\ 2 \text{ simboli } VI = 6 \\ 1 \text{ simbolo } M = 1000 \end{array}$$

$$4 = 1111 = IV$$

CIFRE MIN, CIFRE MAX = DIFF.
CIFRE MAX, CIFRE MIN = SOMMA

entrambe rappresentazioni valide



noto che uso due diverse quantità di simboli

$$IV = 4 = IIII$$

MANAGGIOMA (2 CIFRE < 4)

1 |
2 II
3 III
4 IV
5 V
6 VI
7 VII
8 VIII
9 IX

→ necessita davvero di 4 cifre?

perché non IIIX?

10 X

perché per regola nelle operazioni di sottrazione hanno solo 1 simb.

→ ma posso cambiare le regole

$$\begin{aligned} 13 &= XIX \\ 109 &= CIX \\ &\quad 100 + 10 - 1 \\ 150 &= CL \\ 143 &= CXLIX \\ &\quad 100 + 50 - 10 + 10 - 1 \\ &\quad \underbrace{\text{decine}}_{\text{decine}} \quad \underbrace{\text{unita}}_{\text{unità}} \end{aligned}$$

→ osserviamo che CXLIX è una rappresentazione ridondante potrei scrivere CIL

• cambiando le regole

Se volevo potrei costruire un nuovo codice, in informatica avrei inoltre bisogno di un carattere in più che dica "fine numero" oltre ai caratteri usati per rappresentare i numeri

\downarrow
1 x □

CMII TOT	1	= 1
	v	= 5
	x	= 10
	L	= 50
	C	= 100
	D	= 500
	M	= 1000
	*	= STOP

ES 3 BIT

VARI CASI DEL 3 BIT			
RAPPRESENTAZIONE BINARIA		SIMBOLO	
b ₂	b ₁	b ₀	
0	0	0	• STOP
1	0	0	1
2	0	1	v
3	0	1	x
4	1	0	L
5	1	0	c
6	1	1	d
7	1	1	m

Può pensare a una nuova numerazione, "genovese", che usa una menica minima di rappresentare i numeri

$$8 = 11 \times (10-2)$$

posso sostituire i simboli con la rappresentazione binaria

servono 12 bit (3 cifre + terminatore, = 4 cifre)

01011 | 01011 | 01111 | 01010

ogni cifra (simbolo) è rappresentata da 3 bit
4 simboli = 4 * 3 bit

Se trovo un valore c = successivo faccio la somma, se > la differenza

• Come rappresento lo zero?

uso 3 bit (termino subito)

$\boxed{0|0|0}$ = stringa vuota = 0
lunghezza variabile \rightarrow ha implicito lo zero

Non ho bisogno di un carattere zero esplicito

Può confrontare con la rappresentazione binaria in lunghezza fissa.

Voglio scrivere fino a 1000 (11)
o 2¹⁰ (1023)

numerazione "genovese"
(lung. variabile)

quanti bit? 3
• punto alle caratteri + terminatore

Poiché uso un codice a lunghezza fissa, qualcosa
Sia il valore uso 10 bit sempre
(i valori troppo grandi (> 1023) non posso scriverli)

0 \Rightarrow 3 bit
 $\frac{\text{RISTORANTE}}{10 \text{ BIT}}$ quantità 10 bit
1 \Rightarrow 6 bit ||> inferiore di

(i valori ~~hanno~~ sono grandi ($> 10^2$)
non γ vissi scrivere li)

$O \Rightarrow 3315$

RISK AWARE ~~BY~~ BY

$\Sigma \rightarrow \Sigma$ bit

$$G \Rightarrow \mathfrak{g}_{\text{bit}}$$

$s \Rightarrow s_{\text{bit}}$

prova A + e
d inferiore di
BIT

andare in
danni così
di numeri

altri una
gratifica-
zione
superiore
di poteri

Pojo a servire anche numeri grandi

es. 2000 => 3 BIT

Apre altre possibilità di comprimere le informazioni

es- BIP : si usa un codice e lunghezza variabile
minimizza la quantità di bit senza ovvero
le "spartume" dei vari caratteri più complessi

Si fa una stima dell' **occorrenza** (frequenza) dei veri caratteri, in modo tale da generare la rappresentazione

es. i numeri '3' della
num. genovese, se
molto frequenti
vengono **OTTIMIZZATI**,
riscrivendo le rappresentazioni
del 3 con meno bit.

es. Considero una stringa di caratteri (uso il codice ASCII - 7 bit) \times
carattere

"LA MIA CARA MATERA CUCINA MOLTO BENE" → un po' come
terminatore

Sono in totale (compresi gli spazi) 36 caratteri o 7 BPI = 901, BPI USATI
e di 252?

Posso comprimere x uscire meno di 36 caratteri?
e quindi un codice che uscire meno di 257?

D'altra mano da non tutti i correttori sono usati, solo alcuni risultano e soprattutto posso costruire una tabella ridotta (posso togliere anche $x, y, z \dots$)

$$\checkmark \star A \subset \mathbb{F} \quad T = 1$$

$$B = 1 \quad U = 1$$

$$V_C = 3 \text{ V} \Rightarrow V = 6 \text{ V}$$

۱۰۷

zetteri che

$$= \text{canal}^{\text{G}}$$

V T A C - 1 - 1

B = 1 U = 1
V C = 3 spazio = 6 V
V E = 2 terminazione = 1
L = 2 I = 2 ✓

V * M = 5
N = 2
O = 2
R = 1

Sono i caratteri che appaiono nelle stringhe

= 16 caratteri rappresentabili su 4 BIT = 2⁴ = 16 segno

3G - 4

Posso poi contare quante volte le lettere appaiono

A campo più volte

posso usare un codice a lunghezza

variabile con meno o più di 4 bit (i quali a lunghezza fissa = 4 bit)
usando meno bit per i caratteri più frequenti

3G - 4

Posso usare una via di mezzo fra i due tipi di codice → **CODICE A ESPANSIONE**

Per rappresentare i 3 caratteri mi servono anche solo 2 bit (→ 3 carabin (max 4))

0 → A

0 1 → M

1 → spazio

1 1 → RISPARMIA 4 BIT → comincio configurazione:

1	1	1
---	---	---

↓ ↓ ↓

FISSI L'ESPRESSIBILITÀ

↓

1	1	1	1
---	---	---	---

prendo altri caratteri

0 → C

1 → E

0 → I

1 1 → espansione a 7 bit

Riraggruppo ancora 8 caratteri (mi servono altri 3 bit)

esponendo in base al carattere che voglio rappresentare, se i primi 2 bit sono 1

allora so de i caratteri non sono fra i più frequenti e così via

1	1	1	1	1	1
---	---	---	---	---	---

Nella migliore delle ipotesi

le cose vanno bene con i

caratteri più frequenti

• A, M, spazio = 18 vengono

codificati ciascuno con 2 bit

• 7 caratteri = 6 bit

• 11 " 7 bit → rappresentati col max livello di espansione

uso un totale di $18 \cdot 2 + 7 \cdot 4 + 11 \cdot 7$
 $= 141 \text{ bit}$

Rispetto ai 252 del codice ASCII 36.7
(che riporta i 111 bit) = 13 byte

e 3 bit rispetto al codice BSC.

Tipi di lunghezza di codice

venerdì 9 ottobre 2020 17:59

• CODICE A LUNGHEZZA FISSA

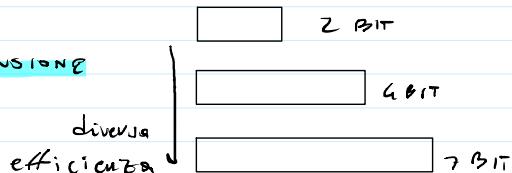


impiega lo stesso numero di bit per tutti i valori che voglio rappresentare

PRO: molto semplice

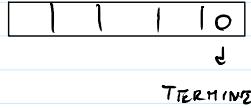
CONTRO: - spreco di bit per i numeri piccoli
- limite max

• CODICE A ESPANSIONE



Lunghezze predefinite ma ne ho più di un'una, in base al valore che voglio rappresentare (no limiti)

• CODICE A LUNGHEZZA VARIABILE



numero di bit variabile sino a un carattere di 'terminazione'

PRO: non ho limiti di lunghezza

CONTRO: più difficile

N.B. Quasi sempre le informazioni in un sistema di calcolo vengono rappresentate con un codice a espansione