Rappresentazione dei numeri razionali

Valore di N = $\sum_{i=-m}^{n-1} c_i b^i$

n ≡ numero cifre parte intera

m ≡ numero cifre parte decimale

Es.

$13{,}375_{10} = 5*10^{-3} + 7*10^{-2} + 3*10^{-1} + 3*10^{0} + 1*10^{1}$

Rappresentazione in virgola fissa

- Precisione costante della parte frazionaria
- E_a costante

16 bit parte intera

16 bit parte frazionaria

13

codifica interi

375

codifica frazionaria

13,375

codifica parte frazionaria:
moltiplicazione per 2 sino a quando vale 0 e i bit corrispondono ai riporti nell'ordine prodotto:

Es.:

	x 2	riporto	
0,375	0,75	0	2^{-1}
0,75	1,5	1	2^{-2}
0,5	1,0	1	2^{-3}
0			

$0{,}375_{10}$=011 0000000…

- L'algoritmo può non convergere

0,1	|0,2	| 0	
0,2	|0,4	| 0	$0{,}1_{10}$=0.0001100011... (∞ bit)
0,4	|0,8	| 0	
0,8	|1,6	| 1	
0,6	|1,2	| 1	
0,2	|0,4	si ripete in modo periodico	

e se non converge si introduce un'approssimazione (imprecisione)

- Errore assoluto costante $2^{-16} = 0{,}000015$

- Inoltre la codifica dimostra che non esiste relazione tra il numero di cifre della parte frazionaria nel numero decimale e in quello corrispondente binario

- Rigidità della pre-divisione dei bit

8.750.000.000 non rappresentabile nei 16 bit della parte intera
16 bit frazionaria inutilizzati

0,0000000000875 16 bit parte intera inutilizzati
16 bit parte frazionaria a 0 (approssimazione)

Real numbers in finite representation
"a large grey area"

Before 1985
Non esiste un accordo su un format per real numbers (fighting)

1985 Agreement
Standard IEEE 754-1985 for binary FP arithmetic
"What Every Computer Scientist Should Know About Floating-Point Arithmetic", David Goldberg, ACM Computing Surveys, Vol 23, No 1, March 1991, pp. 5-48

- non evita tutti i problemi
- stabilisce vincoli sull'entità dei "rounding error" per le operazioni aritmetiche
- un'implementazione hdw è IEEE compliant se produce risultati uguali a quelli degli algoritmi IEEE

↓

Program1 (CPU1) = Program1 (CPU2)

Tutto risolto?
New York Times, nov. 1994 "Intel's Pentium problem persists"
(300 milioni di dollari per il ritiro)

2008 IEEE-754-2008 standard per decimal FP arithmetic

Altri problemi più avanti

Virgola mobile
(standard ANSI/IEEE 754-1985 binary FP arithmetics)

Obiettivi della rappresentazione:
- autoadattabilità tra parte intera e frazionaria
- errore relativo costante

Rappresentazione normalizzata $N = s\,M * 2^e$

$s=\pm$ $1 \leq M < 2$ $b=2$ $e=\pm$numero intero

Codifiche dedicate per 0, NaN ($\sqrt{-2}$, 0/0, ∞+-/∞) ∞

Es. $11_{10} = 1{,}375\ 2^3$ $0.25_{10} = 1\ 2^{-2}$

$\Downarrow$

Modelli di precisione

	hidden	n_s	n_M	n_e	e+ $(2^{(ne-1)}-1)$	precisione
singola precis. FP32	0	1	23	8 -126 – 127	e+127	24
doppia precis. FP64	0	1	52t	11 -1022 – 1023	e+1023	53
Intel 80 bit	1	1	63	15		64

s 0=positivo 1=negativo

M codifica della sola parte frazionaria

e codifica del valore positivo es. FP32 $\Rightarrow$ e+127

(1,) sottinteso in FP

Es.: 11_{10}= 1,375 2^3
s=0
M= 01100000000000000000000
e(3) + 127 = 130 $\Rightarrow$ 10000010

$\Downarrow$

0 10000010 01100000000000000000000
s e M

Esempi di intervalli

Tipi C	Bit	Intervallo possibile (float.h)
float	FP32	$-3,4*10^{38}$ ...$-1,1*10^{-38}$ (`FLT_MIN`) $1,1*10^{-38}$... $3.4*10^{38}$ (`FLT_MAX`) `FLT_MIN =` `(1+0,5+0,25+...)`2^{127}`=` 2^{128} `FLT_MIN = (1+0)`2^{-126}
double	FP64	$-1.7*10^{308}$...$-2.2*10^{-308}$ (`DBL_MIN`) $2.2*10^{-308}$ a $1.7*10^{308}$ (`DBL_MAX`)

Osservazione 1 FP vs v.fissa

8.750.000.000
v. fissa: non rappresentabile
FP32: $0,875*10^{10}$ = $1,018634065..*2^{33}$
= 0 10100000 00000100110001010011001

0,0000000000875
v.fissa: rappresentato come 0
FP32: $0,875*10^{-10}$ = $1,5032385..* 2^{-34}$
= 0 01011101 10000000110101000011110

Osservazione 2 Come si esegue una somma?

$S*2^E + T * 2^F$ con E>F

$\Downarrow$

- denormalization of T: shifting right T (+hidden bit) di (E-F) bits
- sum, normalize and rounding

Es. Mantissa da 7 bit

3 +	$1.5*2^1$		1000000
0.75=	$1.5 * 2^{-1}$	1]1000000 shift 2bit sx	0110000

		$1.875*2$ $\Leftarrow$	1110000

Osservazione 3 NON corrispondenza precisione decimale e FP

numero

1) 2,1 = $1,05 * 2^1 \Rightarrow$ 00001100110011…….

2) 1,5= $1,5 * 2^0 \Rightarrow$ 1(0..0) = 1,5

3) 1,8750= $1,875*2^0 \Rightarrow$ 111 (0…0) = 1,875

4)16,5625=$1,035*2^4 \Rightarrow$ 00001001

Corrispondenza precisa

Osservazione 4. Corrispondenza biunivoca tra decimale e FP

Si afferma che esiste in FP32/64 con decimali composti da 6/15 cifre decimali.

Come si deriva

Detta

- p la precisione binaria del sistema
- q una precisione decimale

Esiste la corrispondenza se

configurazioni binarie $\qquad$ configurazioni decimali

$$2^{(p-1)} \geq 10^q$$

Da cui q = $\lfloor (p-1) \log_{10} 2 \rfloor$ che per p-1=23/52 ottiene 6/15

Cosa significa

Numeri decimali composti da 6/15 cifre complessive sono in corrispondenza biunivoca .con configurazioni FP32/64

NON che i decimali con 6/15 cifre della parte frazionaria .....

Per capire meglio

ε (machine epsilon – FLT_EPSILON): distanza tra il numero 1 e quello immediatamente successivo in FP

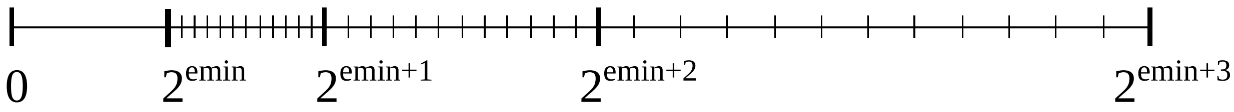
1.0000000... 0001 - 1.0000000...0000

	FP32	FP64
$\varepsilon = 2^{-(p-1)}$	$\varepsilon = 2^{-23}$	2^{-52}

Distanza tra un valore x e il successivo in FP (**u**nits in **l**ast **p**lace – ulp(x)) ?

$\text{ulp}(x) = \varepsilon * 2^e = 2^{-(p-1)+e}$ deriva da $(1+2^{-(p-1)})* 2^e - 1* 2^e$

- cresce al crescere di e
- è costante nell’area di valori che hanno lo stesso valore di e



Ciò significa che l’errore assoluto commesso da un’approssimazione è costante in ognuno degli intervalli precedenti e cresce al crescere del valore.

Per avere un’idea concreta

Distanza minima in prossimità di 0:

FP32 $2^{-23 -126} = 10^{-45}$

Distanza nell’intervallo [1,2[

FP32 $2^{-23 +0} = 10^{-7}$

FP64 $2^{-52 +0} =(10^{-16})$

Distanza 1 dove $2^{-(p-1) + e} =1$ ossia $-(p-1) + e =0$

FP32 e=23 ossia in $[2^{23}, 2^{24}[$ ossia [8.388608,16777216[

FP64 e=52 ossia in $[2^{52}, 2^{53}[$ ossia $[4.5*10^{15}, 9.07*10^{15}[$

Distanza massima

FP32 $2^{-23 +127} = 10^{31}$

FP64 $2^{-52 +1023} = 10^{292}$

Conclusione: il numero di cifre decimali varia in base all’intervallo considerato

$\Downarrow$

int 32 $\approx 4 *10^{9}$

FP 32 $\approx 6.8 *10^{38}$

Osservazione 5 La flessibilità nella gestione dei bit
Nell'esempio supponiamo di usare N=7 bit per la mantissa

origine	norm.	p.intera	p.fraz	mantissa	stored
2,1	$1{,}05 * 2^1$	10	00011001100..∞	0000110	2.093
1,5	$1{,}5 * 2^0$	1	100000000000	1000000	1.5
1,8750	$1{,}875*2^0$	1	111000000000	1110000	1.8750
16,5625	$1{,}035*2^4$	10000	100100000000	0000100	16.5

- si considerano gli "e" bit della parte intera da destra
- hidden bit
- si aggiungono N-e bit della parte frazionaria
- p.frazionaria descresce al crescere della p.intera
- l'arrotondamento è costante all'interno dell'intervallo [1,00….* 2^x e 1,00 * $2^{(x+1)}$[perché hanno lo stesso numero di bit della parte frazionaria.

In altri termini (consideriamo FP 32 per semplicità)

Intervallo [1..2[ossia numeri 1,xx $*2^0$
23 bit mantissa per la parte frazionaria ossia 8.000.000 combinazioni ossia 6 cifre decimali
Intervallo [2..4[ossia numeri 1,xx $*2^1$
1 bit mantissa per parte intera
22 bit mantissa per la parte frazionaria ossia 4.000.000 combinazioni ossia 2.000.000 a unità ossia ancora 6 cifre decimali

Intervallo [≈8milioni – 16777215] ossia 1,xx $*2^{23}$
23 bit mantissa per parte intera
0 bit mantissa per la parte frazionaria rappresentazione interi

Osservazione 6. A proposito dell'arrotondamento e dell'errore
Se la sequenza dei bit necessari è maggiore dei bit disponibili l'algoritmo approssima attraverso la funzione "round"

Errore assoluto Ea(x): spostamento che x subisce
Errore relativo Er(x): Ea(x) rapportato al valore di x

Dato x>0

x_{-} x x_{+}

down, toward 0 roundup

round to nearest

Consideriamo round to nearest

1. Ea(x) $0 <= |x - \text{round}(x)| < \text{ulp}(x)/2 = 2^{-p+e}$

2. Er(x) $= |(x - \text{round}(x)) / x|$

Dalla 1) e dato che $x > 2^e$ si deriva che:

$Er(x) < 2^{-p+e} / 2^e$ ossia $< 2^{-p} = \varepsilon/2$ (machine epsilon)

Er(x) è costante e va bene quindi per micro e macro numeri

Non va bene per domini applicativi nei quali il valore dipende da convenzioni

Es. Bounding Box Regione Lombardia del sistema UTM32/WGS84

- X∈ [459.973, 683.970]
  pi(19bit) pf(33) $=2^{19} * 2^{-52} = 0,1 * 10^{-6}$ mm
- Y∈ [4.949.981, 5.169.976]
  pi(22bit) pf(30) $e=2^{22} * 2^{-52} = 0,9 * 10^{-6}$ mm

Adesso è più chiaro perché….

Esempio 1. round superiore all'unità

```
#include <stdio.h>
int is=0, ix=7,i; float s=0.0, x=7.0;
void main()
{	for(i=1;i<=10000000; i++)  {s=s+x; is=is+ix;}
	printf("is= %d e s= %f",is,s);
}
```

Risultato is= 70.000.000 e s= 77.603.248 (inizio differenza intorno a 16.777.216)

Esempio 2. Denormalizzazione perde numeri piccoli

```
#include <stdio.h>
float a=0.0, b;
void main()
{       for(;;)  {b=a; a=a+1.0; printf("n= %f e n+1= %f",b,a);
                   if(b!=(a-1)) exit();
                 }
}
```

Problema a 16777215

Oppure che: numeri distanti o molto vicini tra loro

$$x^2-y^2 \neq (x-y)*(x+y)$$

Esempio 3. Uguaglianza non sempre funziona

```
#include <stdio.h>
#include <math.h>
int main ()
{float a=.1;float i; printf("\na inizio=%f10\n", a); ⇒ 0.1000001
  for (i=0.1; i!=10.0;i=i+0.1)   {a=a+0.1;  printf("\n%f10", i);}
```

ciclo ∞

passa da 9.90000210 a 10.00000210

Nota: Matlab introduce concetto di tolerance per l’uguaglianza

Esempio 4 Cancellation problem in a-b

$$f'(x) = \frac{df(x)}{dx} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon) - f(x)}{\varepsilon}$$

```
#include <stdio.h>   … <math.h>
int main()// derivata di sqrt(x)
{int c;double x, epsilon;   printf("x?");scanf("%lf",&x);epsilon=1.;
 while (epsilon > 1.e-17)
   {printf("\n x= %f, epsilon= %e,derivata=  %f",
              x, epsilon, (sqrt(x + epsilon)-sqrt(x))/epsilon);
     epsilon=epsilon/10.;
    }
}
```

Risultato   x?1.

```
 x= 1.000000, epsilon= 1.000000e+000,derivata=  0.414214
 x= 1.000000, epsilon= 1.000000e -001,derivata=  0.488088
 x= 1.000000, epsilon= 1.000000e -002,derivata=  0.498756
 x= 1.000000, epsilon= 1.000000e -003,derivata=  0.499875
 x= 1.000000, epsilon= 1.000000e -004,derivata=  0.499988
 x= 1.000000, epsilon= 1.000000e -005,derivata=  0.499999
 x= 1.000000, epsilon= 1.000000e -006,derivata=  0.500000
 x= 1.000000, epsilon= 1.000000e -007,derivata=  0.500000
 x= 1.000000, epsilon= 1.000000e -008,derivata=  0.500000
 x= 1.000000, epsilon= 1.000000e -009,derivata=  0.500000
 x= 1.000000, epsilon= 1.000000e -010,derivata=  0.500000
 x= 1.000000, epsilon= 1.000000e -011,derivata=  0.500000
 x= 1.000000, epsilon= 1.000000e -012,derivata=  0.500044
 x= 1.000000, epsilon= 1.000000e -013,derivata=  0.499600
 x= 1.000000, epsilon= 1.000000e -014,derivata=  0.488498
 x= 1.000000, epsilon= 1.000000e -015,derivata=  0.444089
 x= 1.000000, epsilon= 1.000000e -016,derivata=  0.000000
```

Esempio 5


Visita la Gallery Add-On
Italia Cambia
Tutti i siti Microsoft
Supporto tecnico Microsoft
bing
Home page Supporto tecnico
Centri di supporto
Ricerca avanzata
Acquista prodotti
Operazioni aritmetiche dei valori in virgola mobile potrebbe produrre risultati non accurati in Excel
Per visualizzare l'articolo tradotto automaticamente accanto all'originale in inglese, fare clic qui.
Attenzione: Questo è un articolo tradotto in automatico.
Visualizza i prodotti a cui si riferisce l'articolo.
In questa pagina
Espandi tutto | Chiudi tutto
Sommario
Altre risorse
Altri siti di supporto
Community
Richiedi assistenza
Traduzione articoli
Inglese (Stati Uniti)
Centri di supporto tecnico correlati
Excel 2010
Excel
Excel 2007
Excel 2002
Excel 2000