Számábrázolás, adattípusok

Kormányos Andor

Komplex Rendszerek Fizikája Tanszék

2021. szeptember 8.

Motiváció

- valós számok ábrázolása a számítógépen mindig közelítést igényel
- ez fontos lehet pl ha nagyon kicsi vagy nagyon nagy számokkal végzünk műveleteket: olyan hibák léphetnek fel, amelyek az matematikai számolásokban ismeretlenek

Adattípusok a számítógépben

Legkisebb információhordozó egység: bit ("binary digit"): 0 vagy 1 érték (pl. a computer memóriájában egy cella tartalmaz töltést v nem, a hard disk egy cellájában a mágnesezettség, stb.)

A memória alapegysége a bájt (byte)

- 1 bájt = 8 bit 0 vagy 1 értékkel: $2^8 = 256$ különböző érték
- ez kb. **betűk** és **karakterek** tárolására elegendő
- minden bájtnak közvetlen **memóriacíme** van

Hogy tárolhatók 256-nál nagyobb egész számok?

- több bájtot kell egybefogni
- 2 bájt = 16 bit = 65536 különböző érték
- 4 bájt = 32 bit = kb. 4 milliárd különböző érték
- ha szükséges, 1 bit használható előjelnek
- ilyenkor az ábrázolható tartomány kb. $\pm 2^{b-1}$ lesz

Számábrázolás

Tízes számrendszer:

- $107 = 10^2 \cdot 1 + 0 \cdot 10^1 + 7 \cdot 10^0$
- minden 10-hatvány együtthatója tíz értéket vehet fel

Számábrázolás

Tízes számrendszer:

- $107 = 10^2 \cdot 1 + 0 \cdot 10^1 + 7 \cdot 10^0$
- minden 10-hatvány együtthatója tíz értéket vehet fel

Számítógép: bináris számrendszer (sokkal hatékonyabb hardveres szintű műveletek)

- számábrázolás alapja: kettes számrendszer
- a 2 hatványait használjuk

$107 = 64 + 32 + 8 + 2 + 1$

0	1	1	0	1	0	1	1
2^7	$2^6 (= 64)$	$2^5 (= 32)$	2^4	$2^3 (= 8)$	2^2	$2^1 (= 2)$	$2^0 (= 1)$

- pozitív egész számok ábrázolására
- 1 bájt: $0 - 255$
- 2 bájt: $0 - 65535$, stb.
- egy bitet el lehet használni előjelnek, ekkor az ábrázolható legnagyobb szám megfeleződik

Fixpontos ábrázolás

Hasonlóan működhetne a törtszámok ábrázolása

$1.625 = 1 + 1/2 + 0 + 1/8$

0	0	0	1	1	0	1	0
2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}

- az ábrázolható tartomány nagyon kicsi
- van egy legkisebb lépésköz (felbontás)

Lebegőpontos ábrázolás

10-es számrendszerben gyakran használjuk a normálalakot

- véges sok tizedest adunk meg, valamint egy kitevőt:

$$+2{,}3454612 \times 10^3 \qquad (+0.23454612\text{E}+4)$$

- előjel
- mantissza
- kitevő

Lebegőpontos számok kettes számrendszerben

$$(-1)^s \cdot \underbrace{(1 + \sum_{i=1}^{t} x_i 2^{-i})}_{f} \cdot 2^{e-B}$$

- s: előjel (1 bit)
- f: mantissza
- e: kitevő v. karakterisztika
- B: konstans

Lebegőpontos számok kettes számrendszerben

$$(-1)^s \cdot \underbrace{(1 + \sum_{i=1}^{t} x_i 2^{-i})}_{f} \cdot 2^{e-B}$$

- s: előjel (1 bit)
- f: mantissza
- e: kitevő v. karakterisztika
- B: konstans

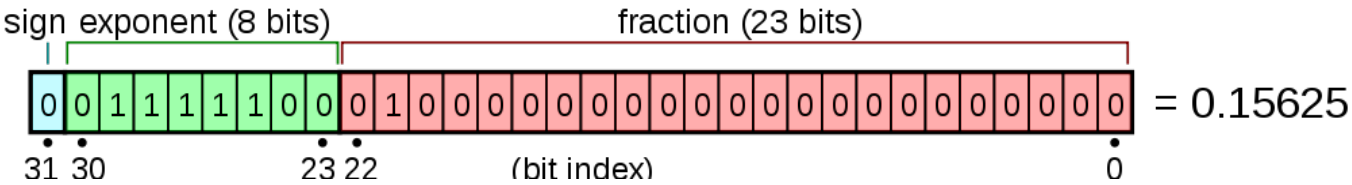
Feladat

Írjuk fel a 0.1-t 2-es számrendszerbeli lebegőpontos számként a fenti alakban! Mit tapasztalunk?

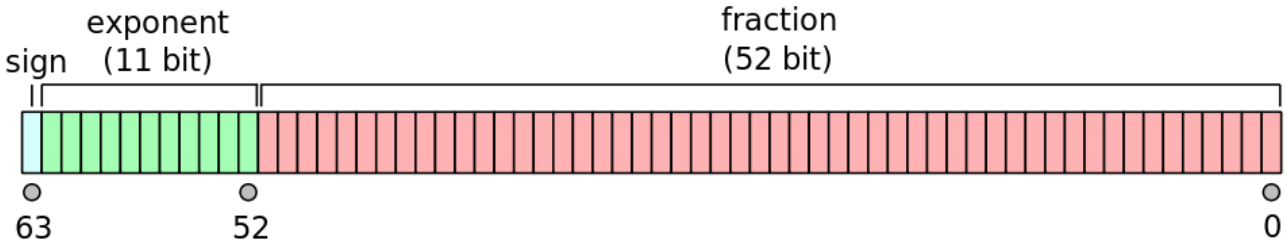
IEEE 754 szabvány

32, illetve 64 bitet kell kiosztani s, f és e tárolására

- IEEE 754 szabvány
- float: 32 bit, 0 körül kb. 7 tizedes pontosság



- double: 64 bit, 0 körül kb. 16 tizedes pontosság



- speciális értékek: $\pm\infty$, NaN, előjeles nulla

Lebegőpontos ábrázolás felbontása

A problémát illusztrálhatjuk a következő egyszerű példával.

Az egyszerűség kedvéért tegyük fel, hogy 10-es számrendszerben dolgozunk. Egy lebegőpontos számot akarunk ábrázolni, három helyjegy használható a mantisszára és egy jegy az exponensre:

$$mantissa \times 10^{exponens}$$

Lebegőpontos ábrázolás felbontása

A problémát illusztrálhatjuk a következő egyszerű példával.

Az egyszerűség kedvéért tegyük fel, hogy 10-es számrendszerben dolgozunk. Egy lebegőpontos számot akarunk ábrázolni, három helyjegy használható a mantisszára és egy jegy az exponensre:

$$mantissa \times 10^{exponens}$$

Ha az exponens 0, akkor a minden egész szám a $0 - 999$ intervallumban pontosan ábrázolható.
Ha az exponens 1, akkor az előbbi számhalmaz minden tagját lényegében megszorozzuk 10-el, tehát a $0 - 9990$ intervallumba eső számokat kapunk. Most azonban csak a 10 többszörösei reprezentálhatóak pontosan, mert továbbra is csak 3 helyjegy használható a mantisszára.

Lebegőpontos ábrázolás felbontása

A problémát illusztrálhatjuk a következő egyszerű példával.

Az egyszerűség kedvéért tegyük fel, hogy 10-es számrendszerben dolgozunk. Egy lebegőpontos számot akarunk ábrázolni, három helyjegy használható a mantisszára és egy jegy az exponensre:

$$mantissa \times 10^{exponens}$$

Ha az exponens 0, akkor a minden egész szám a $0 - 999$ intervallumban pontosan ábrázolható.
Ha az exponens 1, akkor az előbbi számhalmaz minden tagját lényegében megszorozzuk 10-el, tehát a $0 - 9990$ intervallumba eső számokat kapunk. Most azonban csak a 10 többszörösei reprezentálhatóak pontosan, mert továbbra is csak 3 helyjegy használható a mantisszára.

Hasonló probléma lép fel akkor is, ha 2-es számrendszer dolgozunk, a nulla körül a legpontosabb a számábrázolas.

Lebegőpontos felbontás hatása a műveletek eredményére

Például:

- ha a két szám között sok nagyságrend eltérés van, sokat romolhat a pontosság
- egyes esetekben azonos nagyságrendű számok kivonásánál is romolhat a pontosság

Lebegőpontos felbontás hatása a műveletek eredményére

Például:

- ha a két szám között sok nagyságrend eltérés van, sokat romolhat a pontosság
- egyes esetekben azonos nagyságrendű számok kivonásánál is romolhat a pontosság

További olvasnivaló (sok részlet, példák):
David Goldberg:
What every computer scientist should know about floating-point arithmetic