

# Predstavljanje realnih brojeva

\* Mogu se predstaviti na dva načina

- Sa tvz. fiksnim zarezom ili tačkom (eng. fixed point)

➢ Fiksira se broj pozicija koje se koriste za prestavljanje celog deo broja (n) i broj pozicija za predstavljanje razlomljenog dela broja (m).

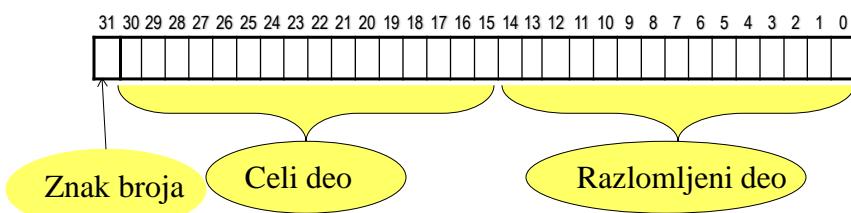
- Sa pokretnim zarezom ili tačkom (eng. floating point)

➢ eksponencijalni zapis broja

## Predstavljanje sa fiksnim zarezom

\* Fiksira se broj pozicija koje se koriste za prestavljanje celog deo broja (n) i broj pozicija za predstavljanje razlomljenog dela broja (m).

\* Za n=16 i m=15 sadržaj 32-bitnog registra bio bi:



Veoma mali opseg brojeva se može predstaviti

## Predstavljanje u pokretnom zarezu

- \* U praksi, neke vličine imaju mnogo malu ili mnogo veću vresnost od onih koje se fiksnim zarezom mogu predstaviti.
- \* Primer:  $1.0 \cdot 10^{25}$  ili  $2.57 \cdot 10^{-33}$
- \* Broj predstavljen u brojnom sistemu sa osnovom  $b$  se može zapisati u eksponencijalnom obliku na sledeći način:

$$m \cdot b^E$$

gde je:

$m$  – mantisa,  
 $E$  - eksponent

## Normalizovani oblik mantise

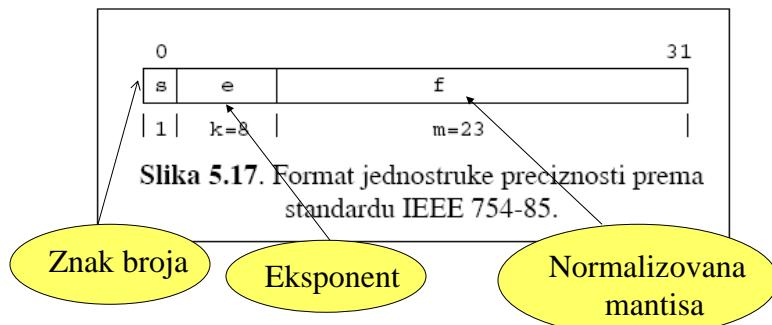
- \* U upotrebi su dva normalizovana oblika broja

- Stariji oblik  $a = M \cdot B^E$  u kome je  $M$  normalizovana mantisa,  $1/B \leq M < 1$ ,  $B$  osnova eksponenta,  $B \in \{2, 8, 10, 16\}$  i  $E$  eksponent.
- Noviji oblik, definisan standardom IEEE 754-85,
  - $a = Z \cdot B^E$  u kome je  $Z$  mantisa  $1 \leq Z < 2$ , a osnova eksponenta  $B=2$ .
  - Mantisa je oblika  $Z = z.f = 1.b_1...b_n$ 
    - **0.1** postaje  **$1.0 \times 2^{-1}$**
    - **0.01** postaje  **$1.0 \times 2^{-2}$**
    - **0.11** postaje  **$1.1 \times 2^{-1}$**
  - $1.1$  je već normalizovano i jednako je  **$1.1 \times 2^0$**
  - $10.01$  postaje  **$1.001 \times 2^1$**

## Predstavljanje brojeva u pokretnom zarezu

- \* Za kodiranje znaka brojeva sa pokretnim zarezom koristi se prosto označavanje znaka (znak i apsolutna vrednost broja).
- \* Eksponent E predstavlja se u polarizovanom, tj. sa pomerajem, u obliku e:  $e=E+P$ , gde je P veličine  $P=2^{k-1}$  (stariji oblik) ili  $P=2^{k-1}-1$ , a k je broj binarnih cifara eksponenta
- \* Reprezentacija broja sadrži tri komponente
  - Znak broja,
  - Eksponent (koji se pamti sa pomerajem),
  - Normalizovana mantisu oblika 1.bbb...

## Predstavljanje brojeva u pokretnom zarezu: IEEE 754-85 standard



## Primer

\* Predstaviti broj 27.125 u 32-bitnom registru ukoliko se za predstavljanje broja koristi pokretni zarez.

$$(27.125)_{10} = (11011.001)_2 = (1.1011001 \cdot 10^{100})_2$$

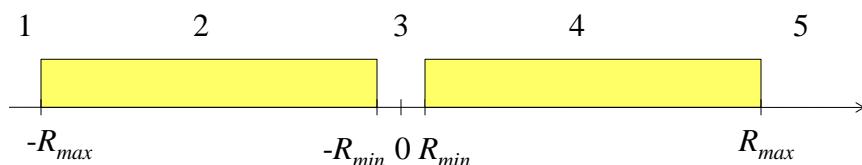
Eksponent se pamti sa pomerajem:

$$p = (2^7 - 1)_{10} = (0111111)_2$$

eksponent: 0111111 + 100 = 10000011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	1	0	0	0	0	0	1	1	1	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

## Opseg brojeva koji se mogu predstaviti u pokretnom zarezu



$$R_{min} = 0.12 \cdot 10^{-37} \text{ - minimalna vrednost}$$

$$R_{max} = 0.34 \cdot 10^{39} \text{ - maksimalna vrednost}$$

Obeležene oblasti:

1 - Oblast negativnog prekoračenja ( $-\infty$ )

5 - Oblast pozitivnog prekoračenja ( $+\infty$ )

3 - Oblast mašinske nule (0=)

2 - opseg negativnih brojeva

4 - opseg pozitivnih brojeva

## Specijalne vrednosti

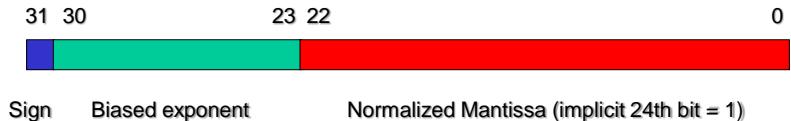
- \* U IEEE formatu se vrednosti eksponenta 0 i 255 koriste da označe specijalne vrednosti podatka
  - Zbog toga je najmanja vrednost eksponenta 00000001
    - To je ustvari  $1 \cdot 127 = -126$
  - Najveća vrednost eksponenta je 11111110
    - To je ustvari  $254 - 127 = +127$
  - ako je  $e$  (eksponent) =255 (sve jedinice), a  $f = 0$ , vrednost se tretira kao  $+\infty$  ili  $-\infty$ , u zavisnosti od znaka,  $s$ ;
  - $e=0$  (sve nule),  $f=0$ , predstavlja pozitivnu ili negativnu nulu, u zavisnosti od znaka,  $s$ ;
  - $e=0$ ,  $f \neq 0$ , predstavlja denormalizovani oblik broja.
    - To su brojevi koji ne mogu biti normalizovani, tj. Brojevi manji od  $2^{-126}$
    - U ovom slučaju bit ispred decimalne tačke u mantisi broja je 0, a prava vrednost eksponenta je -126
  - $e=255$ ,  $f \neq 0$ , ova vrednost se tretira kao "Nije broj" (Not a number – NaN) i koristi se da signalizira različite izuzetke (exceptions)
    - Obično se javlja kao rezultat ilegalnog deljenja:
      - $0/0$  ili  $\infty/\infty$

- \* Standard IEEE 754-85 predviđa četiri formata za predstavljanje brojeva sa pokretnim zarezom

- sa jednostrukom preciznošću,
- sa dvostrukom preciznošću,
- sa jednostrukom proširenom preciznošću, i
- dvostrukom proširenom preciznošću

## Predstavljanje FP brojeva – single precision

- IEEE 754 single precision



$$(-1)^s \times F \times 2^{E-127}$$

Exponent	Mantissa	Object Represented
0	0	0
0	non-zero	denormalized
1-254	anything	FP number
255	0	pm infinity
255	non-zero	NaN

11

## Predstavljanje FP brojeva – double precision

- IEEE 754 double precision



$$(-1)^s \times F \times 2^{E-1023}$$

Exponent	Mantissa	Object Represented
0	0	0
0	non-zero	denormalized
1-2046	anything	FP number
2047	0	pm infinity
2047	non-zero	NaN

## Aritmetičke operacije sa pokretnim zarezom (Floating point – FP)

- \* Pri obavljanju aritmetičkih operacija mantisa  $Z=z,f$  se uzima u punom obliku, uključujući i celi deo tj. cifru  $z$ .
  - Tek pri slanju rezultata van aritmetičke jedinice, iz normalizovane mantise može se odstraniti celi deo, cifra  $z=1$  ili 0.
- \* U toku izvršenja FP operacija mogu nastupiti sledeći događaji:
  - prekoračenje eksponenta
    - pozitivni eksponent prevaziđa maksimalnu vrednost koja se može predstaviti
      - to znači da je broj i suviše veliki i može se smatrati da je  $+\infty$  ili  $-\infty$  u zavisnosti od znaka mantise
  - potkoračenje eksponenta
    - negativni eksponent je manji od minimalne moguće vrednosti (npr. -200 je manje od -127)
      - to znači da je broj i suviše mali i može se smatrati da je 0
  - potkoračenje mantise
    - u procesu poravnavanja mantise cifre mantise mogu ispasti van opsega sa desne strane
      - npr ako treba sabrati  $2 \times 10^3 + 0.2 \times 10^0$ , a koriste se samo 3 cifre za mantisu, manji broj će ispasti van opsega  $(2+0.0002) \times 10^3$
  - prekoračenje mantise
    - sabiranje dva broja istog zanaka može dovesti do generisanja prenosa, van cifre najveće težine.
      - ovaj problem se može rešiti ponovnom normalizacijom i inkrementiranjem eksponenta.

## Sabiranje-oduzimanje

- \* Sabiranje odnosno oduzimanje obavlja se svođenjem oba sabirka na isti eksponent.
- \* Pri tome veći sabirak mora ostati normalizovan, a eksponent manjeg sabirka izjednačava sa eksponentom većeg sabirka.
  - Naravno da to zahteva pomeranje mantise manjeg broja udesno za broj binarnih pozicija određen razlikom eksponenata većeg i manjeg sabirka.
    - ovaj postupak može dovesti do potkoračenja mantise

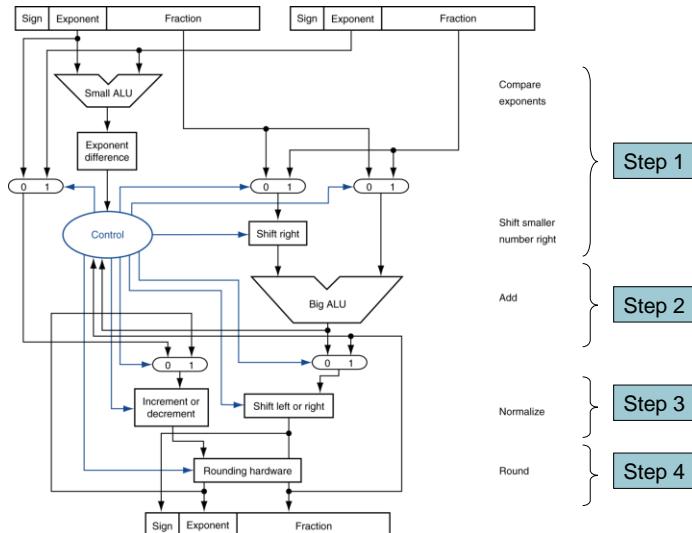
## Sabiranje/oduzimanje

- \* Obavlja se u 5 koraka
- \*  $C = A + B$ ,  $A = a \cdot 2^p$ ,  $B = b \cdot 2^q$
- \* korak1: Poredjenje eksponenata  $p$  i  $q$  da bi se pronašao veći,  $r = \max(p, q)$  i razlika  $t = |p - q|$ .
- \* korak2: Pomeriti za  $t$  mesta u desno mantisu manjeg broja da bi se izjednačili eksponenti pre sabiranja
- \* korak3: Sabiranje mantisa i dobijanje medjurezultata
- \* korak4: Ako je potrebno, normalizovati vrednost rezultata, svesti vrednost razlomka na 23 cifre iza binarne tačke,
- \* korak5: Proveriti rezultat na prekoračenje-potkoračenje:
  - \* prekoračenje: ako je  $e_c > 254$ , uzeti  $e_c = 255$ ,  $f_c = 0$ ;
  - \* potkoračenje: ako je  $e_c = 0$  i  $Z = 0, f_c$ , pri čemu je  $f_c < 2^{-23}$ , uzeti  $Z_c = 0, 0$

## Primer

- \*  $9.25 \times 10^3 + 8.22 \times 10^2 = ?$
- \* Denormalizovati broj sa manjim eksponentom:
  - $9.25 \times 10^3 + 0.822 \times 10^3$
- \* Sabrati brojeve:
  - $9.25 \times 10^3 + 0.822 \times 10^3 = 10.072 \times 10^3$
- \* Normalizovati rezultat:
  - $10.072 \times 10^3 = 1.0072 \times 10^4$
- \* Primer – binarno sabiranje
- \*  $1001 + 10$  or  $1.001 \times 2^3 + 1.0 \times 2^1$
- \* Denormalizovati broj sa manjim eksponentom:
  - $1.001 \times 2^3 + 0.01 \times 2^3$
- \* Sabrati brojeve:
  - $1.001 \times 2^3 + 0.01 \times 2^3 = 1.011 \times 2^3$
- \* Rezultat je već normalizovan

## FP sabirač

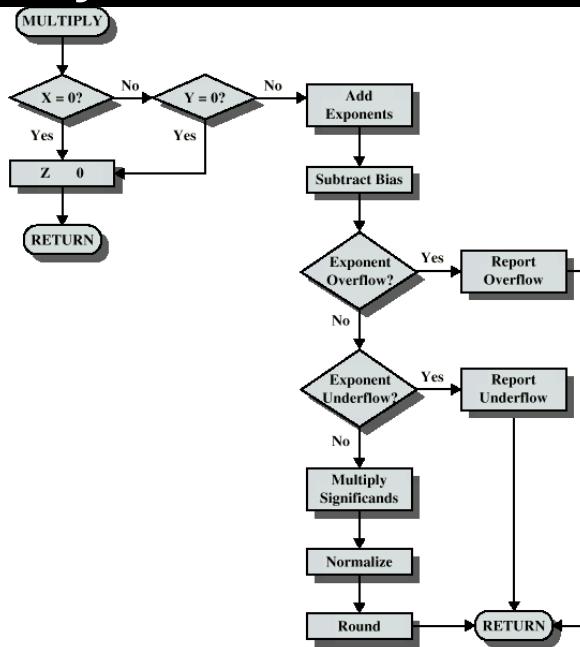


18

## FP Množenje

1. Sabrati eksponente i oduzeti pomeraj P (polarizaciju):  
 $e_c = e_a + e_b - P.$ 
  - Oduzimanje pomeraja (polarizacije) P je potrebno da bi se eksponent rezultata dobio u istom obliku (naime i  $e_a$  i  $e_b$  su polarizovani, pa je potrebno oduzeti jedan pomeraj)
2. Pomnožiti mantise i odrediti znak rezultata  
 $S_c = S_a \oplus S_b.$
3. Ako je potrebno, normalizovati vrednost rezultata, svesti vrednost razlomka na 23 cifre iza binarne tačke,
4. Proveriti rezultat na prekoračenje-potkoračenje,

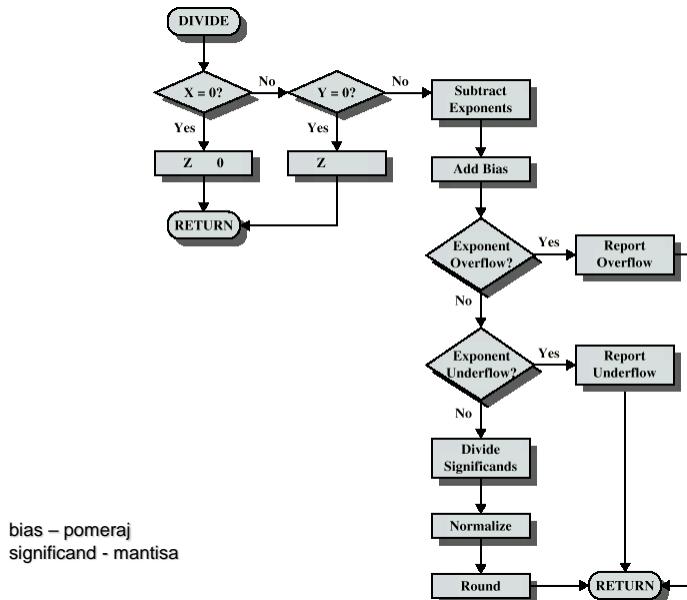
## FP množenje



## FP deljenje

1. Oduzeti eksponente i dodati P:  $e_c = e_a - e_b + P$ .
  - Dodavanje polarizacije P neophodno je da bi se eksponent rezultata zadržao u polarizovanom obliku
2. Podeliti mantise i odrediti znak rezultata  
 $S_c = S_a \oplus S_b$ .
3. Ako je potrebno, normalizovati vrednost rezultata, svesti vrednost razlomka na 23 cifre iza binarne tačke,
4. Proveriti rezultat na prekoračenje-potkoračenje

# FP deljenje



## Problemi sa FP operacijama zbog ograničene veličine registara

### \* Pre izvršenja FP operacije eksponent i mantisa svakog operanda se pune u registre ALUa.

- U slučaju mantise dužina registra je skoro uvek veća od dužine mantise +1.
- Registr ALUa sadrži dodatne bitove koji se zovu bitovi zaštite (guard bits) koji se koriste da se desna strana mantise dopuni nulama. Razlog je ilustrovan na sledećoj slici.

$$\begin{array}{r}
 x = 1.000\ldots00 \times 2^1 \\
 -y = 0.111\ldots11 \times 2^1 \\
 \hline
 z = 0.000\ldots01 \times 2^1 \\
 = 1.000\ldots00 \times 2^{-22}
 \end{array}$$

(a) Binary example, without guard bits

$$\begin{array}{r}
 x = 1.000\ldots00\ 0000 \times 2^1 \\
 -y = 0.111\ldots11\ 1000 \times 2^1 \\
 \hline
 z = 0.000\ldots00\ 1000 \times 2^1 \\
 = 1.000\ldots00\ 0000 \times 2^{-23}
 \end{array}$$

(b) Binary example, with guard bits

**Figure 10.25** The Use of Guard Bits

Neka su brojevi vi koje treba oduzeti  $X=1.000\ldots00 \times 2^1$  i  $Y=1.111\ldots11 \times 2^0$ .

• Prvo što mora da se uradi je da se eksponent manjeg broja izjednači sa eksponentom većeg.

• To zahteva da se mantisa manjeg broja, u ovom primeru, pomeri u desno za jednu bit poziciju. Ovo je prikazano na sl.a.

• U ovom procesu pomeranja Y je izgubio jednu cifru u mantisi;

• Nakon oduzimanja i normalizovanja mantise rezultata dobija se rezultat  $2^{-22}$ .

Na sl. b je prikazano kako izgleda ova operacija kada se koriste bitovi zaštite. Sada bit najmanje težine operanda Y nije izgubljen zbog poravnavanja eksponenata i rezultat je  $2^{-23}$ . Dobijeni rezultati se razlikuju dva puta

## Rešenje problema

- \* Kada se radi sa podacima u pokretnom zarezu, dobijeni rezultet može imati više od 24 (single precision) ili 53 (double precision) binarne cifre.
  - s obzirom na ograničeni broj bitova koji se koriste za predstavljanje brojeva, neophodno je izvršiti zaokruživanje dobijenog rezultata na 24 tj. 53 cifre.
  - da bi greška zbog zaokruživanja bila što manja u toku izračunavanja se radi sa većim brojem cifara
  - analize su pokazale su da je za svođenje mantise rezultata na određenu dužinu (24 odnosno 53 bita) dovoljno izračunati mantisu rezultata sa dve odnosno tri dodatne cifre u odnosu na predviđenu dužinu mantisa.
- \* Te cifre su cifra zaštite, cifra zaokruženja i lepljivi bit (engl. guard digit, round digit i sticky bit, respektivno).
  - Cifra zaštite
  - cifra zaokruženja
  - Lepljivi bit (Sticky bit): treća dodatana cifra na desnom kraju mantise – koristi se da razreši situacije kao što su 0.50...00 vs. 0.50...01

## Tri dodatna bita

- \* Format mantise sa tri dodatna bita:

mantissa format plus extra bits:

1.XXXXXXXXXXXXXXXXXXXXXXX 0 0 0

  ^                          ^                          ^                          ^  
  |                          |                          |                          |  
  |                          |                          |                          |  
  | - hidden bit  
  | - 23 bit mantissa from a representation  
  | - guard bit (g)  
  | - round bit (r)  
  | - sticky bit (s)

## Primer

izgled mantise: 1100000000000000000000100

pretpostavimo da mantisa mora biti pomerena za 8 pozicija desno  
pre izvršenja operacije sabiranja

	g r s
Before first shift:	1.1100000000000000000000100 0 0 0
After 1 shift:	0.111000000000000000000000010 0 0 0
After 2 shifts:	0.0111000000000000000000000001 0 0 0
After 3 shifts:	0.0011100000000000000000000000 1 0 0
After 4 shifts:	0.0001110000000000000000000000 0 1 0
After 5 shifts:	0.0000111000000000000000000000 0 0 1
After 6 shifts:	0.0000011100000000000000000000 0 0 1
After 7 shifts:	0.0000001110000000000000000000 0 0 1
After 8 shifts:	0.0000000111000000000000000000 0 0 1

## Zaokruživanje

\* Druga stvar koja utiče na preciznost rezultata je politika zaokruživanja.

- U toku izvršenja ALU operacija mantisa se uglavnom pamti sa većim brojem bitova.
- Pri povratku na standardni FP format, ekstra bitovi se moraju eliminisati tako da se dobije rezultat sa što manjom greškom usled zaokruživanja.

\* IEEE standard definiše tri načina zaokruživanja:

- zaokruživanje prema nuli:
  - prosti odsecanje.
- zaokruživanje prema  $-\infty$ :
  - vrednost na koju se zaokružuje je blizu prave vrednosti, ali ne veća od nje
- zaokruživanje prema  $+\infty$ :
  - vrednost na koju se zaokružuje je blizu prave vrednosti, ali ne manja od nje
- zaokruživanje na najbližu vrednost:
  - rezultat se zaokružuje na najbližu (parnu) vrednost

## IEEE zaokruživanje - primeri

### \* Metod 1: zaokruživane prema 0 (odsecanje)

- Npr: ako je decimalna vrednost .778

- na raspolaganju su 3 decimalne cifre za predstavljanje broja .778
- na raspolaganju su 2 decimalne cifre za predstavljanje broja .77
- na raspolaganju je 1 decimalna cifra za predstavljanje broja, .7

### \* Primer: ako se koriste g r s bitovi, samo se odsecaju

- 1.1100000000000000000000000000100 0 0 0
- 1.1100000000000000000000000000100 (mantissa used)
- 1.1100000000000000000000000000110 1 1 0
- 1.1100000000000000000000000000110 (mantissa used)
- 1.0000000000000000000000000000111 0 1 1
- 1.0000000000000000000000000000111 (mantissa used)

## IEEE zaokruživanje - primeri

### \* Metod 2: zaokruživanje prema $+\infty$ :

- uvek se zaokružuje prema većoj vrednosti
  - 1.23
  - ako je 1 decimalna cifra na raspolaganju: 1.3
  - -2.86 se zaokružuje na -2.8
- examples in the floating point format with guard, round and sticky bits: g r s
  - 1.1100000000000000000000000000100 0 0 0
  - 1.1100000000000000000000000000100 (mantissa used, exact representation)
  - 1.1100000000000000000000000000100 1 0 0
  - 1.1100000000000000000000000000101 (rounded "up")
  - -1.1100000000000000000000000000100 1 0 0
  - -1.1100000000000000000000000000100 (rounded "up")
  - 1.1100000000000000000000000000001 0 1 0
  - 1.1100000000000000000000000000010 (rounded "up")
  - 1.1100000000000000000000000000001 0 0 1
  - 1.1100000000000000000000000000010 (rounded "up")

## IEEE zaokruživanje - primeri

### \* Metod 3: zaokruživanje prema $-\infty$

- uvek se vrši zaokruživanje na manju vrednost
  - 1.23 se zaokružuje na 1.2
  - -2.86 se zaokružuje na -2.9
- 
- examples in the floating point format with guard, round and sticky bits: g r s
    - 1.1100000000000000000000000000000100 0 0 0
    - 1.1100000000000000000000000000000100 (mantissa used, exact represent.)
  
    - 1.1100000000000000000000000000000100 1 0 0
    - 1.1100000000000000000000000000000100 (rounded "down")
  
    - -1.1100000000000000000000000000000100 1 0 0
    - -1.1100000000000000000000000000000101 (rounded "down")

## IEEE zaokruživanje - primeri

### \* Method 4: zaokruživanje na najbližu (parnu) vrednost

- examples in the floating point format with guard, round and sticky bits:
- g r s
  - 1.1100000000000000000000000000000100 0 0 0
  - 1.1100000000000000000000000000000100 (mantissa used, exact representation)
  
  - 1.1100000000000000000000000000000100 1 1 0
  - 1.1100000000000000000000000000000100
  
  - 1.1100000000000000000000000000000100 0 1 0
  - 1.1100000000000000000000000000000100
  
  - 1.1100000000000000000000000000000100 1 1 1
  - 1.1100000000000000000000000000000100
  
  - 1.1100000000000000000000000000000100 0 0 1
  - 1.1100000000000000000000000000000100
  
  - 1.1100000000000000000000000000000100 1 0 0 (the "halfway" case)
  - 1.1100000000000000000000000000000100 (lsb is a zero)
  
  - 1.1100000000000000000000000000000100 1 0 0 (the "halfway" case)
  - 1.1100000000000000000000000000000100 (lsb is a zero)