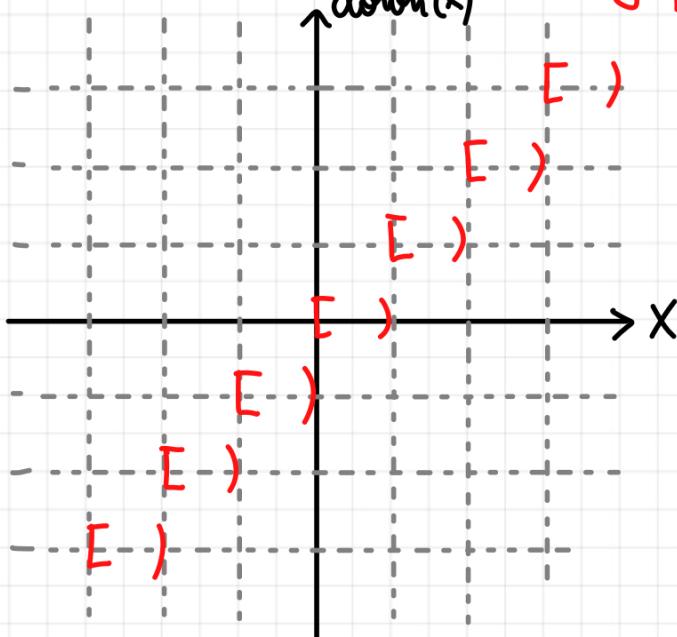


# Arhitectura calculatoarelor

## (curs 9 - SG)

B. Rotunjire spre  $-\infty$  (rotunjire în jos / downwards)

$\rightarrow x^*$  este cel mai mare întreg pentru care  $x^* \leq x$



if  $x \geq 0$  downwards  $\equiv$  inwards

if  $x$  - repre. în  $C_2$ , downwards  $\equiv$  trunchiere biti fractionali

Dacă  $X$  este în cod S.-M., rotunjirea în jos este echivalentă cu:

$$\begin{cases} \text{trunchiere la partea întreagă, dacă } X \geq 0 & x^* = x_{m-1}x_{m-2}\dots x_0 \\ X^* = \begin{cases} x_{n-1}x_{n-2}\dots x_1x_0 - 1, & \text{if } x_{-1}x_{-2}\dots x_{-m} \neq 0, \text{ dacă } X < 0 \\ x_{n-1}x_{n-2}\dots x_1x_0, & \text{if } x_{-1}x_{-2}\dots x_{-m} = 0 \end{cases} \end{cases}$$

$$-1,025 \longrightarrow -2$$

$$-3,5 \longrightarrow -4 \quad -4 \leq -3,5$$

$$[-1,0) \longrightarrow -1$$

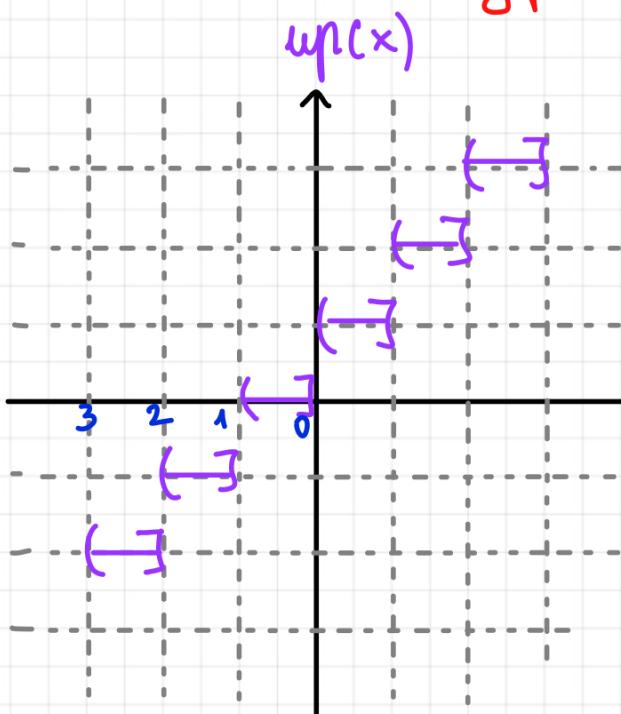
$$+3,5 = 0^S 011.1_{C_2} \longrightarrow -3,5 = 1\ 100 \circled{1}_{C_2} = -2 + 100 = -8 + 4 = -4$$

$$+5,625 = 0\ 101.101_{C_2}$$

$$-5,625 = 1\ 010.001_{C_2} = -2^3 + 010 = -8 + 2 = -6 \quad \checkmark$$

C. Rotunjire spre  $+\infty$  (rotunjirea în sus/upwards)

$x^*$  este cel mai mic întreg pentru care  $x^* \geq x$



$$0,99 \rightarrow 1$$

$$-5,025 \rightarrow -5$$

$$0,011 \rightarrow 1$$

Dacă  $x$  este negativ, rotunjirea în sus este echivalentă cu rotunjirea spre 0.

$$\begin{cases} x_{m-1} x_{m-2} \dots x_0, & \text{if } x \leq 0 \text{ and } x_{-1} x_{-2} \dots = 0 \\ x^* & \\ x_{m-1} x_{m-2} \dots x_0 + 1, & \text{if } x > 0 \text{ and } x_{-1} x_{-2} \dots \neq 0 \end{cases}$$

Caracteristicile rotunjirii în sus și în jos:

▷ eroile sunt în aceeași direcție  $\Rightarrow$  eroarea totală este acumulată mai repede

ex:  $0,1 + 1,7 + 2 + 3,1 + 4,5$

upwards:  $1 + 2 + 2 + 4 + 5$

downwards:  $0 + 1 + 2 + 3 + 4$

▷ formează o limită superioară/inferioră pentru rezultat  $\Rightarrow$  aritmetică intervalelor

### b.) Rotunjire la cel mai apropiat număr par

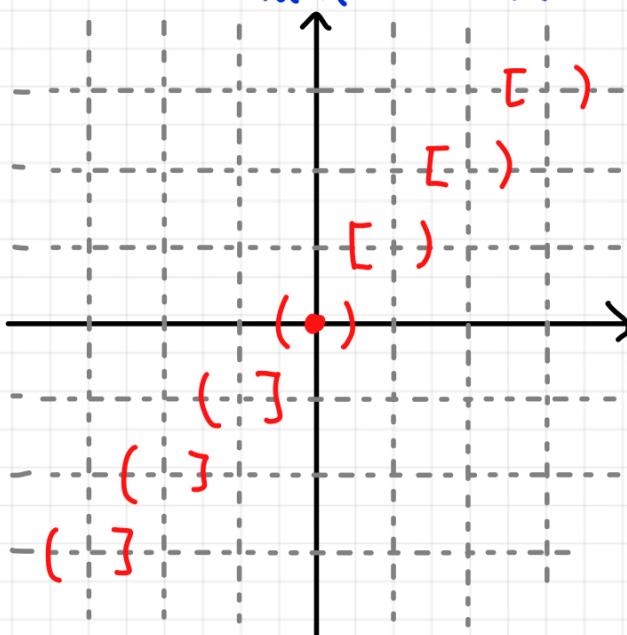
Modul de rotunjire IEEE 754 la cel mai apropiat număr par este derivat din modul de rotunjire la cel mai apropiat număr.

### b') Rotunjire la cel mai apropiat număr

$$\text{if } x \geq 0 \quad x^* = \begin{cases} x_{m-1}x_{m-2}\dots x_0, & \text{if } .x_{-1}x_{-2}\dots x_{-m} < \frac{1}{2} \\ x_{m-1}x_{m-2}\dots x_0 + 1, & \text{if } .x_{-1}x_{-2}\dots x_{-m} \geq \frac{1}{2} \end{cases}$$

În mod similar se poate defini modul de rotunjire la cel mai apropiat număr pentru valori negative.

$rtm(x)$  round to nearest



$$+2,2 = 2$$

$$+2,99 = 3$$

$$+4,5 = 5$$

## Analiza acumulării erorii

→ se consideră  $x > 0$ , având doar 2 biți de parte fraționată  
 $\Rightarrow x_{-3} = x_{-4} = \dots = x_{-m} = 0$

Intrări		Iesiri	
$x_{-1}$	$x_{-2}$	$x^* = \text{rtm}(x)$	$\xi = x^* - x$
0	0	$x_{m-1}x_{m-2} \dots x_1x_0$	0
0	1	$x_{m-1}x_{m-2} \dots x_1x_0$	$-\frac{1}{4}$
1	0	$x_{m-1}x_{m-2} \dots x_1x_0 + 1$	$\frac{1}{2}$
1	1	$x_{m-1}x_{m-2} \dots x_1x_0 + 1$	$\frac{1}{4}$

$$.00_{(2)} = .0_{(10)} < \frac{1}{2}$$

$$.01_{(2)} = .25_{(10)}$$

$$.10_{(2)} = .5_{(10)} \geq \frac{1}{2} \quad \left(1 - \frac{1}{2}\right)$$

$$.11_{(2)} = .75_{(10)}$$

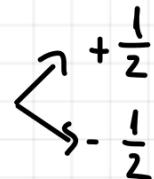
$$1 - \frac{3}{4} = \frac{1}{4}$$

Dacă cele 4 cazuri de mai sus sunt echiprobaabile de-a lungul unei secvențe de calcule, eroarea medie se obține astfel:

$$\xi_{\text{mean}} = \frac{0 - \frac{1}{4} + \frac{1}{2} + \frac{1}{4}}{4} = \frac{1}{8}$$

Dacă linia 3 este mai probabil să apară în comparație cu celelalte,  $\xi_{\text{mean}}$  poate deveni mai mare decât  $\frac{1}{8}$

Soluția problemei acumulării erorilor: se împarte cazul  $.x_{-1}x_{-2} = .10$  în două sub-cazuri, cu probabilitate egală (sau cât mai aproape de egal), astfel încât unul dintre sub-cazuri să fie rotunjit în sus și celălalt în jos.



O abordare posibilă pentru împărțirea cazului de rotunjire a unei părți fractionare de  $\frac{1}{2}$  cu probabilități egale ar fi să fie inspectat bitul cel mai puțin semnificativ al părții întregi,  $x_0$ , diferențind astfel între numere pare și impare. Pentru numerele pozitive, dacă  $x_0 = 0$  (numere pare), rotunjirea ar putea fi făcută în jos, în timp ce pentru  $x_0 = 1$  (numere impare), rotunjirea ar fi făcută în sus.

⇒ de aici apără rotunjirea la cel mai apropiat nr. par

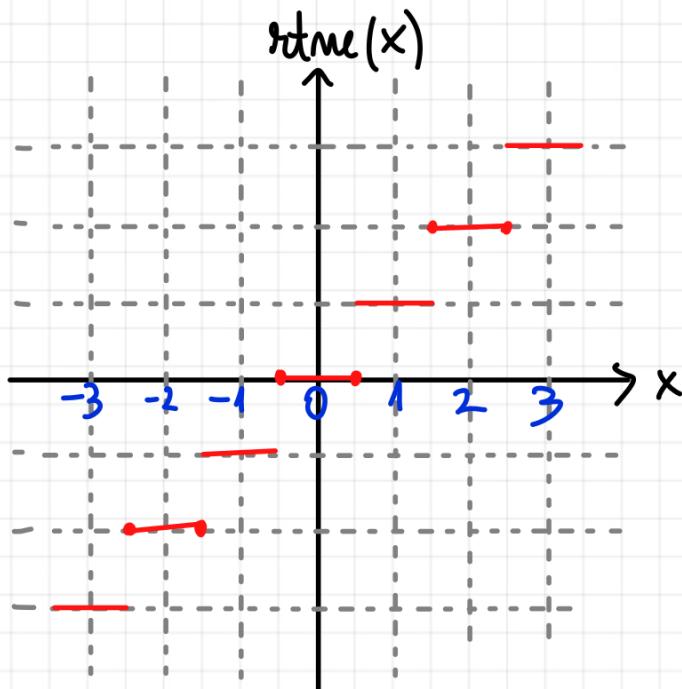
### b.) Rotunjirea la cel mai apropiat număr întreg

Fără a pierde din generalitate, se consideră  $x$  pozitiv.  $x^*$  rotunjit la cel mai apropiat număr întreg este definit ca:

$$x \geq 0 \quad x^* = \begin{cases} x_{n-1}x_{n-2}\dots x_1x_0, & \text{dacă } x_{-1}x_{-2}\dots x_{-m} < \frac{1}{2}, \text{ SAU} \\ & \text{dacă } x_{-1}x_{-2}\dots x_{-m} = \frac{1}{2} \text{ SI } x_0 = 0 \\ x_{n-1}x_{n-2}\dots x_1x_0 + 1, & \text{dacă } x_{-1}x_{-2}\dots x_{-m} > \frac{1}{2}, \text{ SAU} \\ & \text{dacă } x_{-1}x_{-2}\dots x_{-m} = \frac{1}{2} \text{ SI } x_0 = 1 \end{cases}$$

par

impar



$$4,51 \rightarrow 5$$

$$4,49 \rightarrow 4$$

$$3,49 \rightarrow 3$$

$$[-0,5; 0,5] \rightarrow 0$$

$$(0,5; 1,5] \rightarrow 1$$

$$[1,5; 2,5] \rightarrow 2$$

### 3.3. Reguli de normalizarea și rotunjirea unui rezultat FP.

Se consideră significanții  $X_M$  și  $Y_M$ , pe  $m$  biți:

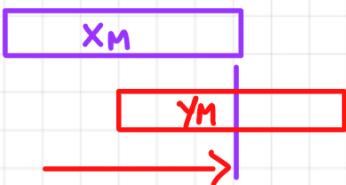
$$X_M = 1 \cdot x_{m-2} \cdot x_{m-3} \dots x_i \dots x_1 x_0$$

$$Y_M = 1 \cdot y_{m-2} y_{m-3} \dots y_i \dots y_1 y_0$$

$m+d$  capacitate în urma extinderii  
poate ajunge  $2^m$

nu ar fi ok.

Se consideră  $X_E \geq Y_E$  a.î. pentru adunarea significanților,  $Y_M$  trebuie aliniat prim deplasare la dreapta cu  $d = |X_E - Y_E|$  poziții.



Alinierea lui  $Y_M$  se poate face:

cu precizie infinită: sunt păstrați toți biții lui  $Y_M$ , inclusiv cei cu ponderi mai mici ca  $2^{-m+1}$

cu precizie finită: sunt păstrați doar 3 biți dintre toți cei care au ponderi mai mici ca  $2^{-m+1}$



$\Rightarrow$  3 biți stichy

①  $g$ : bit de gardă - cu ponderarea  $2^{-M}$

▷ protejează împotriva pierderii de precizie

▷ ultimul bit shiftat la dreapta în interiorul capacitatei de stocare

②  $r$ : bit de rotunjire : -cu ponderarea  $2^{-m-1}$

▷ utilizat pt. rotunjirea rezultatului

▷ penultimul bit shiftat la dreapta

③  $s$ : bitul sticky : - cu ponderarea  $2^{-m-2}$

▷ se obține ca rezultat al unui SAV logic între toți biții mai puțini semnificațiovi care au fost deplasati la dreapta din  $Y_M$ , exceptând  $g$  și  $r$

După aliniere  $Y_M \rightarrow Y_{Mal}$

Se consideră rezultatul FP al unei adunări  $\Rightarrow Z_M = x_M + Y_{Mal}$ , cu condiția  $x_E \geq Y_E$

Datorită transportului care poate fi generat de adunarea semnificativelor:

$$Z_M = Z_M \cdot Z_{M-1} \cdot Z_{M-2} \cdot Z_{M-3} \dots Z_1 \cdot Z_0; g \text{ și } s$$

↪ 2 biți înainte de .  $\Leftrightarrow Z_M - Cout$

Operatia de normalizare pentru  $Z_M$  nu produce  $Z_{M_m}$ :

$$Z_{M_m} = 1 \cdot Z_{M-2_m} Z_{M-3_m} \dots Z_{1_m} Z_{0_m} | R \quad S$$

→ Pei 2 biti sunt utilizati pentru efectuarea operatiei de rotunjire, succesiive normalizării

### Cazuri de normalizare

$$Z_M = Z_m Z_{m-1} \cdot Z_{m-2} Z_{m-3} \dots Z_1 Z_0 : g \quad r \quad s$$

$$Z_{M_m} =$$

$$1 \cdot Z_{M-2_m} Z_{M-3_m} \dots Z_{1_m} Z_{0_m} | R \quad S$$

1) Caz 1  $Z_m=1$

=> deplasare la dreapta nr.

1 bit,  $Z_E++$

/ virgula se mută la stânga

$$1 \cdot Z_{M-1} Z_{M-2} \dots Z_2 Z_1 | Z_0 \quad g \text{ ORR } ORS \\ ; Z_E++$$

2) Caz 2  $Z_m=0$

$$Z_{m-1}=1$$

$$1 \cdot Z_{M-2} Z_{M-3} \dots Z_1 Z_0 | g \quad r \text{ OR } s$$

$Z_M$  deja normalizat

3) Caz 3  $Z_m=0$

$$Z_{m-1}=0$$

$$Z_{M-2}=1$$

=> deplasăm nr. la stânga

cu 1 bit,  $Z_E--$

$$1 \cdot Z_{M-3} Z_{M-2} \dots Z_0 g | r \quad s \\ Z_E--;$$

/ deplasăm virgula la dreapta

$$4) \text{ Gaz } 4 \quad z_m = 0$$

$$z_{m-1} = 0$$

$$z_{m-2} = 0$$

$$z_{m-3} = 1$$

$$1. \ z_{m-4} \ z_{m-5} \dots \ g \ 0 \ 0 \ 0$$

deplasăm nr la stânga  
cu 2 pozitii,  $z_E = z_{E-2}$

! Dacă  $z_M$  necesită o shiftare la stânga cu 2 sau mai multe pozitii atunci după ce atasez bitul  $g$  lui  $z_M$ ,  $z_M$  este completat cu biti de 0.  
În plus, se alege  $R = S = 0$

→ Rotunjirea lui  $z_M$  utilizează bitii  $R$  și  $S$  pentru implementarea tuturor celor 4 moduri de rotunjire.

### Reguli de rotunjire

#### Mod de rotunjire

	$z_{M_m} \geq 0$	$z_{M_m} \leq 0$
spre 0	— <small>(nu contează R și S)</small>	<small>discard R, S</small>
spre -∞	<small>discard R, S</small>	<small>if(R OR S) then <math>z_{M_m} - 1</math></small>
spre +∞	<small>if(R OR S) then <math>z_{M_m} + 1</math></small>	—
la cel mai apropiat nr. par	<small>if(R and (S OR <math>z_m</math>)) then <math>z_{M_m} + 1</math></small>	<small>if(R and (S OR <math>z_m</math>)) then <math>z_{M_m} - 1</math></small>

spre +∞,  $z_{M_m} \geq 0$  :    1. 0001 → 2

1. xy și  $x,y \geq 1 \Rightarrow$  incrementăm

### 3.4. Adunarea / scăderea cu rotunjire a numărelor FP

În ceea ce privește modurile de rotunjire  $\rightarrow$  Văduri  
 → eroarea de rotunjire nu este corelată cu diferența exponentilor celor 2 operațiuni.

Ex:

Case	Example operation	Operation with infinite precision and followed by rounding	Operation with rounding	Error $\epsilon$
a	$1.000 - 1.011 \cdot 2^{-6}$	$\begin{array}{r} 1.000 \\ - 0.000001011 \\ \hline 0.111100101 \\ 1.1100 \\ + 1 \quad rs \\ \hline 10.000 \rightarrow [1.000] \end{array} \quad \left( 1 - \frac{11}{512} \right)$ $c_{\text{out}}$	$\begin{array}{r} 1.000 \\ - 0.000001 \\ \hline 0.111100 \\ 1.1100 \\ + 1 \quad lgrs \\ \hline 10.000 \rightarrow [1.000] \end{array}$	$1 - \frac{501}{512} = \frac{11}{512}$
b	$1.000 - 1.011 \cdot 2^{-5}$	$\begin{array}{r} 1.000 \\ - 0.00001011 \\ \hline 0.111100101 \\ 1.1100 \\ + 1 \quad rs \\ \hline [1.111] \end{array} \quad \left( 1 + \frac{7}{8} \cdot 2^{-1} \right)$	$\begin{array}{r} 1.000 \\ - 0.000011 \\ \hline 0.111100 \\ 1.1100 \\ + 1 \quad grs \\ \hline [1.111] \end{array}$	$\frac{15}{16} - \frac{245}{256} = -\frac{5}{256} (= -\frac{10}{512})$
c	$1.000 - 1.011 \cdot 2^{-4}$	$\begin{array}{r} 1.000 \\ - 0.0001011 \\ \hline 0.1110001 \\ 1.1100 \\ + 1 \quad rs \\ \hline 1.111 \rightarrow [1.111] \end{array} \quad \left( 1 + \frac{7}{8} \cdot 2^{-1} \right)$	$\begin{array}{r} 1.000 \\ - 0.000101 \\ \hline 0.111000 \\ 1.1100 \\ + 1 \quad lgrs \\ \hline 1.111 \rightarrow [1.111] \end{array}$	$\frac{15}{16} - \frac{117}{128} = \frac{3}{128} (= \frac{12}{512})$
d	$1.000 - 1.011 \cdot 2^{-3}$	$\begin{array}{r} 1.000 \\ - 0.001011 \\ \hline 0.110101 \\ 1.1010 \\ + 1 \quad rs \\ \hline [1.101] \end{array} \quad \left( 1 + \frac{5}{8} \cdot 2^{-1} \right)$	$\begin{array}{r} 1.000 \\ - 0.001011 \\ \hline 0.110100 \\ 1.1010 \\ + 1 \quad grs \\ \hline [1.101] \end{array}$	$\frac{13}{16} - \frac{53}{64} = -\frac{1}{64} (= -\frac{8}{512})$
e	$1.000 - 1.011 \cdot 2^{-2}$	$\begin{array}{r} 1.000 \\ - 0.01011 \\ \hline 0.10100 \\ 1.0100 \\ + r \\ \hline [1.010] \end{array} \quad \left( 1 + \frac{1}{4} \cdot 2^{-1} \right)$	$\begin{array}{r} 1.000 \\ - 0.010110 \\ \hline 0.101000 \\ 1.0100 \\ + grs \\ \hline [1.010] \end{array}$	$\frac{5}{8} - \frac{21}{32} = -\frac{1}{32} (= -\frac{16}{512})$
f	$1.000 - 1.011 \cdot 2^{-1}$	$\begin{array}{r} 1.000 \\ - 0.1011 \\ \hline 0.0000 \\ 0.0000 \\ + \\ \hline [1.010] \end{array} \quad \left( 1 + \frac{1}{4} \cdot 2^{-1} \right)$	$\begin{array}{r} 1.000 \\ - 0.101100 \\ \hline 0.010000 \\ 1.0000 \\ + grs \\ \hline [1.010] \end{array}$	$\frac{5}{16} - \frac{5}{16} = 0$
g	$1.100 + 1.011 \cdot 2^{-1}$	$\begin{array}{r} 1.100 \\ + 0.1011 \\ \hline 10.0000 \\ c_{\text{out}} 1.0000 \\ + 1 \quad rs \\ \hline 1.001 \rightarrow [1.001] \end{array} \quad \left( 1 + \frac{1}{8} \cdot 2^{-1} \right)$	$\begin{array}{r} 1.100 \\ + 0.1011 \\ \hline 10.0000 \\ 1.0000 \\ + 1 \quad rs \\ \hline 1.001 \rightarrow [1.001] \end{array}$	$\frac{9}{4} - \frac{35}{16} = \frac{1}{16}$

Se va utiliza un format FP simplificat și redus:

- inspirat de IEEE 754, cu câmpuri mai înguste

1 bit pentru semn  
3 biți pentru câmpul de exponent ( $e = 3$ )  
3 biți pentru partea fractionară a significandului

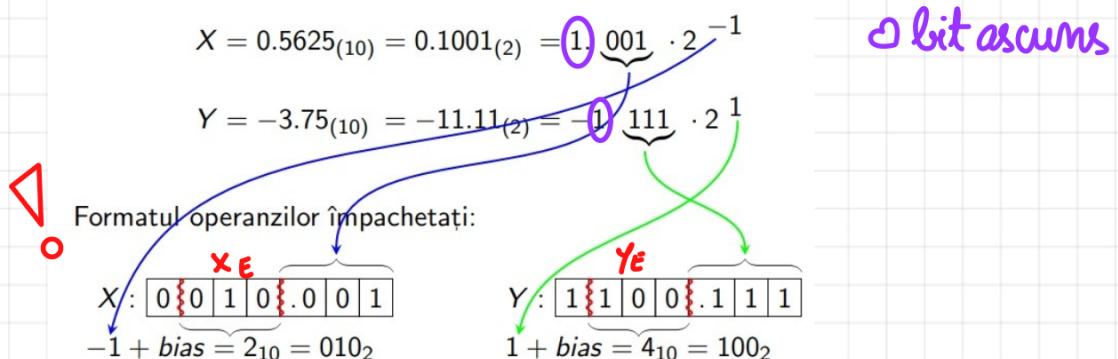
→ aceeași relație pentru bias ca în IEEE 754

$$\text{bias} = 2^{e-1} - 1 = 3$$

→ aceleași exceptii

Um ex căre va fi explicat pe pasi:

Se consideră următorii operanzi:



Algoritmul de adunare cu rotunjire a numerelor de virgulă mobilă

### ① Despacetare operanzi

► atacare bit ascuns

► verificare exceptii: unul din operanzi este  $0, \pm\infty, \text{NaN}$

$X: \boxed{0 \ 0 \ 1 \ 0 \ 1 \ . \ 0 \ 0 \ 1} \quad Y: \boxed{1 \ 1 \ 0 \ 0 \ 1 \ . \ 1 \ 1 \ 1}$

$X_E \quad X_M$

$Y_E \quad Y_M$

## 2. Calcularea diferenței exponentilor $d = X_E - Y_E$

- ▶ dacă  $d < 0$  rezultă că  $|X| < |Y| \xrightarrow{X_E < Y_E}$   $\text{swap}(x, y)$ 
  - ▶ se interschimbă operanzii
  - ▶ se alege exponentul rezultatului,  $Z_E = Y_E$
- ▶ dacă  $d \geq 0$ 
  - ▶ se alege exponentul rezultatului,  $Z_E = X_E$

Interschimbarea operanzilor permite reducerea ariei unității FP (păstrează facilitatea de deplasare la dreapta pentru aliniere doar pentru  $Y$ ).

$$d = X_E - Y_E = 2 - 4 = -2 < 0$$

$$\Rightarrow \text{swap}(x, y) \text{ și } Z_E = Y_E = 4$$

$X: 1\cancel{1}00\cancel{1}111$        $Y: 0\cancel{0}10\cancel{1}.001$   
 (vechiul y)

## 3. Dacă $\text{sign}(x) \neq \text{sign}(y) \Rightarrow$ se complementează de doi $Y_M$

- ▶ semne diferite implică operația de scădere
- ▶ complementarea de doi permite utilizarea unui sumator binar pentru scăderea significanților
- ▶ se complementează de doi doar significandul  $Y_M$  (reduce aria unității FP evitând includerea aceleiași facilități și pentru  $X_M$ )

$$\text{ex: } 2 \cdot 2^3 - 2 \cdot 2^1 = \left[ 2 - \frac{2}{2^2} \right] \cdot 2^3$$

$$X - Y = X + (-Y)$$

$$\begin{aligned} \text{sign}(x) &= 1 \\ \text{sign}(y) &= 0 \end{aligned} \quad \Rightarrow \text{complementăm } Y_M \text{ c}_2$$

INITIAL

$$Y: 0\cancel{0}10\cancel{1}.001 \rightarrow Y_M$$

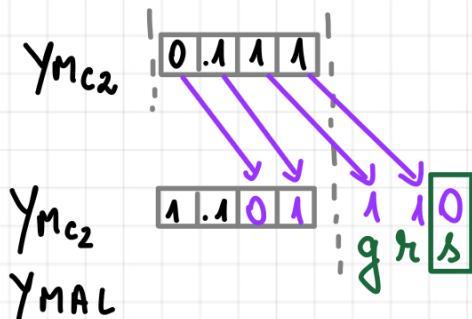
bUPĂ  $c_2$

$$0\cancel{0}10\cancel{0}.111 \rightarrow Y_{Mc_2} = 0.111$$

! Este un pas optional (doar dacă diferență semnele)

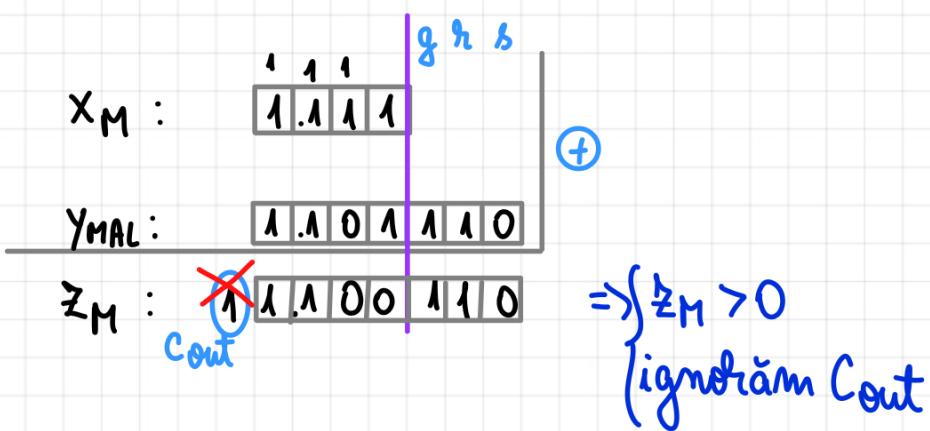
4. Se aliniază  $Y_M$  prin deplasare la dreapta cu  $|d|$  poziții,  $Y_M$  prin aliniere este referit prin  $Y_{MAL}$ .

- dacă în Pasul 3  $Y_M$  a fost complementat de doi, la deplasarea la dreapta, se introduc biți de 1 în pozițiile mai semnificative ale lui  $Y_M$ , în loc de 0
- sunt păstrați biții sticky:  $g$ ,  $r$  și  $s$



5. Se adună cele 2 significanze  $Z_M = X_M + Y_{MAL}$

- dacă  $sign(X) = sign(Y)$  potențialul bit de transport de ieșire **Cout** păstrat generat este păstrat (este parte din rezultat)
- dacă  $sign(X) \neq sign(Y)$  și **nu** este generat bitul de transport de ieșire  $\Rightarrow Z_M$  este negativ și trebuie complementat de doi
- dacă  $sign(X) \neq sign(Y)$  și este generat bitul de transport de ieșire  $\Rightarrow Z_M$  este pozitiv și bitul transport este neglijat



6. Pre-normalizare

! Având în vedere tabelul cu metodele de rotunjire

Din cauză că adunăm unitate la significandul lui  $Z_M$ , această adunare poate genera Cout  $\Rightarrow 10\ldots$

$\Rightarrow$  avem nevoie de post normalizare (un singur bit de parte întreagă)

- rezultată pe baza cazurilor de normalizare din secțiunea 3.3
    - este determinat  $Z_{M_n}$
    - $Z_E$  poate fi modificat
  - verificarea exceptiilor:
    - dacă  $Z_E = Z_{E_{MAX}} (2^e - 2 = 6)$  și  $Z_M$  necesită o deplasare la dreapta cu 1 bit  $\Rightarrow$  Overflow
    - dacă  $Z_E = Z_{E_{min}} (1)$  și  $Z_M$  necesită o deplasare la stânga cu cel puțin 1 bit  $\Rightarrow$  Underflow
- produce nr. de normalizări, dar tolerate

$Z_M$       

dim 5.

e posibil să am și un carry out  
dar în acest caz m-a făcut mereu

Regula 2 tabel:

$Z_M$  este deja normalizat ( $Z_{m_0} = 0$ ;  $Z_{m-1} = 1$ )

$$\Rightarrow Z_{M_m} = \begin{array}{|c|c|c|c|c|} \hline & 1 & . & 1 & 0 & 0 \\ \hline \end{array} \quad \Rightarrow Z_E = 4$$

$\underbrace{\phantom{000}}_{Z_{m_0}}$

⑦ Se determină valourile R și S (pe baza acelorași cazuri de normalizare din secțiunea 3.3)

$Z_M : \begin{array}{|c|c|c|c|c|} \hline & 1 & . & 1 & 0 & 0 \\ \hline & g & r & s & & \\ \hline \end{array}$

Regula 2.  $\Rightarrow R = g = 1$

$S = r \text{ OR } s = 1$

⑧ Rotunjirea lui  $Z_M$ , obținându-l pe  $Z_M^*$

- pe baza regulilor de rotunjire din secțiunea 3.3
- dacă rotunjirea generează transport de ieșire  $\rightarrow$  cout
- se va post-normaliza rezultatul
  - se deplacează la dreapta  $Z_M^*$  cu 1 bit
  - este incrementat  $Z_E$
  - verificarea exceptiei de Overflow

Post-normalizare: 1.0 parte fractionară  $Z_M$

$\Rightarrow$  deplasare cu un bit la dreapta,  $Z_E++$

! Trebuie verificat dacă  $Z_E$  este la valoarea maximă

Se consideră modul de rotunjire la cel mai apropiat număr par. Potrivit regulilor din 3.3, condiția  $R \text{ and } (S \text{ or } z_{0_n})$  este adevărată  $\Rightarrow$  este incrementat  $Z_{M_n}$ .

if  $(R \text{ and } (S \text{ or } \Sigma_M))$  then  $\Sigma_{M_m} + t$

(1 and (1 or 0)) then  $\Sigma_{M_{m+1}}$  [true]

$$\Sigma_{M_m} : \boxed{1.100}$$

$$\sum_{m=1}^{\infty} \frac{1}{m^2} = 1.645$$

$\Rightarrow$  nu avem nicio de post normalizare

$$\Sigma_F = 4$$

### 9. Determinarea semnului rezultatului:

- dacă  $sign(X) = sign(Y) \Rightarrow sign(Z) = sign(X)$
  - dacă  $sign(X) \neq sign(Y)$ ,  $sign(Z)$  este obținut din tabelul următor

Intrări		Ieșiri		
Interschimbare (în Pasul 2)	Complementare de doi (în Pasul 5)	$sign(X)$	$sign(Y)$	$sign(Z)$
DA		+	-	-
DA		-	+	+
NU	DA	+	-	-
NU	DA	-	+	+
NU	NU	+	-	+
NU	NU	-	+	-

sem mole dim  
pasul 1

Pentru exemplul nostru , avem prima linie  $\Rightarrow \text{sign}(z) = -$

swap ② ✓  $\Rightarrow \text{sign}(z) = \text{sign}(y) = -$

## 10. Împachetarea rezultatului

→ Utilizează cămpurile lui  $Z$ , determinate anterior

significandul lui  $\geq$

1.101

## exponențial lui 2

semnul lui 2

Z impachetat

$$z_E = 4 = \underline{100}_{(2)}$$

$S \neq E$  parte fractionară

Verificare:

$$x = 0.5625$$

$$y = -3.75$$

$$z = x + y = -3.1875 \text{ (rezultat cu precizie infinită)}$$

Unpack z

1	1	0	0	1	.	1	0	1
---	---	---	---	---	---	---	---	---

$$z = (-1)^{\text{sign}} \cdot 2^{\text{E-bias}} \cdot z_M =$$

$$= (-1)^1 \cdot 2^{4-3} \cdot 1.101_{(2)} =$$

$$= -2 \cdot 1.101_{(2)} =$$

$$= 11.01_{(2)} =$$

$$= 3,25$$

Din intervalul  $[-3.25, -3.00]$ , doar marginile au reprezentări valide în formatul considerat. Considerând poziția rezultatului exact relativ la mijlocul intervalului,  $-3.125$ , rezultatul  $Z$  este rotunjit la  $-3.25$ .

