Arhitectura Calculatoarelor

Oprițoiu Flavius
flavius.opritoiu@cs.upt.ro

20 Noiembrie 2024
27 Noiembrie 2024
4 Decembrie 2024

Cap. 3 Analiza funcțională și sinteza unităților aritmetice de virgulă mobilă

În general, se consideră operanzi IEEE 754 de virgulă mobilă (floating point - FP), cu excepția cazului în care se specifică altfel. Numerele IEEE 754 pot fi:

- împachetate (sau normalizate) : pentru stocare/transmisie de date
- despachetate : utilizate în timpul calculelor

Fie $X = X_M 2^{X_E}$ și $Y = Y_M 2^{Y_E}$. Cele patru operații aritmetice fundamentale sunt definite mai jos:

- $X + Y = (X_M + Y_M 2^{Y_E - X_E}) 2^{X_E}$, dacă $X_E \geq Y_E$
- $X - Y = (X_M - Y_M 2^{Y_E - X_E}) 2^{X_E}$, dacă $X_E \geq Y_E$
- $XY = X_M Y_M 2^{X_E + Y_E}$
- $\dfrac{X}{Y} = \dfrac{X_M}{Y_M} 2^{X_E - Y_E}$

3.1 - Operații și arhitecturi FP (contin.)

Pe baza definițiilor operațiilor fundamentale de aritmetică FP, o unitate de aritmetică FP are 2 subunități:

- calculul exponentului: efectuarea adunării sau scăderii exponenților
- calculul significandului: efectuarea

$+$, $-$, $*$ sau $/$	significanzilor
$*2^{Y_E - X_E}$	pentru alinierea significanzilor

Cele două subunități operează doar cu operanzi de virgulă fixă:

- subunitatea exponentului folosește numere întregi de virgulă fixă (reprezentate în cod exces)
- subunitatea semnificandului operează cu numere fracționare de virgulă fixă

3.2 - Rotunjire

Rotunjire: conversia unei reprezentări de precizie ridicată la o reprezentare de precizie scăzută (pentru stocare/transmisie). Moduri de rotunjire IEEE 754:

- spre 0
- spre $+\infty$
- spre $-\infty$
- la cel mai apropiat număr par

În IEEE 754, rotunjirea face referire la biții fracționari cu ponderi mai mici decât ponderea celui mai puțin semnificativ bit al significandului. Pentru concizie, doar în acest paragraf, rotunjirea vizează conversia unui număr cu parte întreagă și fracționară într-un număr întreg.

Se consideră X, cu:

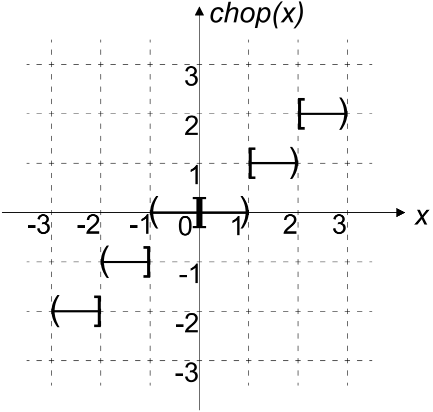
$$X = x_{n-1}x_{n-2}\dots x_1x_0.x_{-1}x_{-2}\dots x_{-m}$$

Fie $X^\star$ valoarea rotunjită a lui X, cu $X^\star$ fiind un întreg:

$$X^\star = x_{n-1}^\star x_{n-2}^\star \dots x_1^\star x_0^\star$$

A Rotunjire spre 0 (rotunjire spre interior)

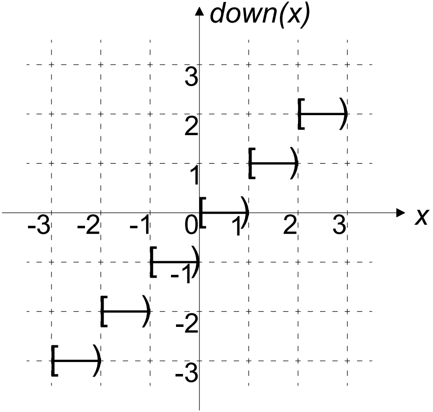
$X^\star$ este cel mai mare întreg pentru care $|X^\star| \leq |X|$



Dacă X este în cod S.-M., rotunjirea spre interior este echivalentă cu trunchierea la partea întreagă.

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

$X^\star$ este cel mai mare întreg pentru care $X^\star \leq X$



Dacă X este pozitiv, rotunjirea în jos este echivalenta cu rotunjirea spre 0.

B Rotunjire spre $-\infty$ (continuare)
Dacă X este în cod C2, rotunjirea în jos este echivalentă cu trunchierea la partea întreagp.

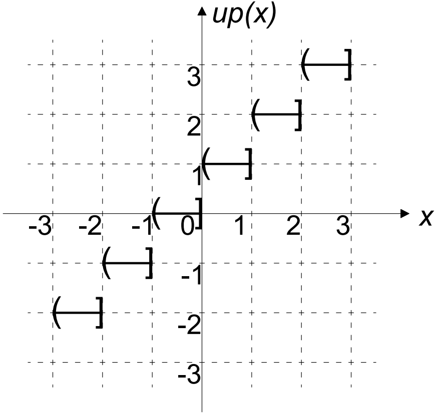
Dacă X este în cod S.-M., rotunjirea în jos este echivalentă cu:

$\nearrow$ trunchiere la partea întreagă, dacă $X \geq 0$

$$\searrow X^{\star} = \begin{cases} x_{n-1}x_{n-2}\ldots x_1x_0 - 1, & \text{if } .x_{-1}x_{-2}\ldots x_{-m} \neq 0 \\ x_{n-1}x_{n-2}\ldots x_1x_0, & \text{if } .x_{-1}x_{-2}\ldots x_{-m} = 0 \end{cases}, \text{dacă } X < 0$$

(C) Rotunjire spre $+\infty$ (rotunjire în sus)

$X^\star$ este cel mai mic întreg pentru care $X^\star \geq X$



Dacă X este negativ, rotunjirea în sus este echivalenta cu rotunjirea spre 0.

Caracteristicile rotunjirii în sus și în jos:

- erorile sunt în aceeași direcție $\Rightarrow$ erorea totală este acumulată mai repede
- furnizează o limită superioară/inferioară pentru rezultat $\Rightarrow$ aritmetica intervalelor

D' Rotunjire la cel mai apropiat număr par

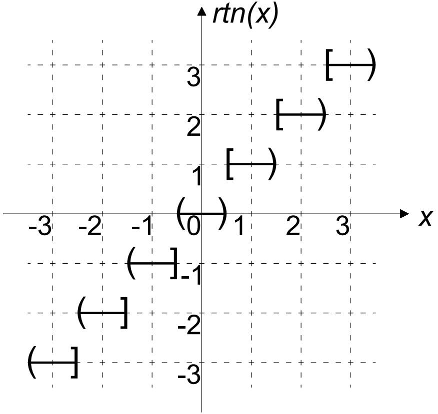
Modul de rotunjire IEEE 754 la cel mai apropiat număr par este derivat din modul de *rotunjire la cel mai apropiat număr*

Fără a pierde din generalitate, se consideră X pozitiv. $X^{\star}$ rotunjit la cel mai apropiat număr este definit ca:

$$X^{\star} = \begin{cases} x_{n-1}x_{n-2}\ldots x_1x_0, & \text{dacă } .x_{-1}x_{-2}\ldots x_{-m} < \frac{1}{2} \\ x_{n-1}x_{n-2}\ldots x_1x_0 + 1, & \text{dacă } .x_{-1}x_{-2}\ldots x_{-m} \geq \frac{1}{2} \end{cases}$$

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

(D') Rotunjire la cel mai apropiat număr (continuare)



D' Rotunjire la cel mai apropiat număr (continuare)

Analiza acumulării erorii: se consideră X pozitiv, având doar 2 biți de parte fracționară.

Intrări		Ieșiri	
x_{-1}	x_{-2}	$X^\star = rtn(X)$	$\epsilon = X^\star - X$
0	0	$x_{n-1}x_{n-2}\dots x_1x_0$	0
0	1	$x_{n-1}x_{n-2}\dots x_1x_0$	$-\frac{1}{4}$
1	0	$x_{n-1}x_{n-2}\dots x_1x_0 + 1$	$\frac{1}{2}$
1	1	$x_{n-1}x_{n-2}\dots x_1x_0 + 1$	$\frac{1}{4}$

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

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

Dacă linia 3 a tabelului de mai sus este mai probabil să apară în comparație cu celelalte, ϵ_{mean} poate deveni mai mare decât $\frac{1}{8}$.

D' Rotunjire la cel mai apropiat număr (continuare)
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.

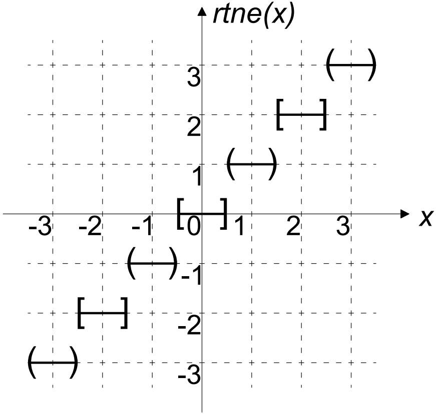
O abordare posibilă pentru împărțirea cazului de rotunjire a unei părți fracționare de $\frac{1}{2}$ cu probabilități egale ar fi să fie inspectat bitul cel mai puțin semnificativ al părții întregi, x_0, diferențiind astfel între numere pare și impare. Pentru numerele pozitive, dacă $x_0 = 0$ (numere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

D Rotunjire la cel mai apropiat număr par

Fără a pierde din generalitate, se consideră X pozitiv. $X^\star$ rotunjit la cel mai apropiat număr par este definit ca:

$$X^\star = \begin{cases} x_{n-1}x_{n-2}\dots x_1x_0, & \text{dacă } .x_{-1}x_{-2}\dots x_{-m} < \frac{1}{2},\ \text{SAU} \\ & \text{dacă } .x_{-1}x_{-2}\dots x_{-m} = \frac{1}{2} \text{ ȘI } x_0 = 0 \\ & \\ x_{n-1}x_{n-2}\dots x_1x_0 + 1, & \text{dacă } .x_{-1}x_{-2}\dots x_{-m} > \frac{1}{2},\ \text{SAU} \\ & \text{dacă } .x_{-1}x_{-2}\dots x_{-m} = \frac{1}{2} \text{ ȘI } x_0 = 1 \end{cases}$$

(D) Rotunjire la cel mai apropiat număr par (continuare)



3.3 Reguli de normalizarea și rotunjirea unui rezultat FP

Se consideră significanzii X_M și Y_M, pe m biți:

$$X_M = 1.x_{m-2}x_{m-3}\ldots x_i \ldots x_1x_0$$
$$Y_M = 1.y_{m-2}y_{m-3}\ldots y_i \ldots y_1y_0$$

Se consideră $X_E \geq Y_E$ așa încât pentru adunarea significanzilor, Y_M trebuie aliniat prin deplasare la dreapta cu $d = |X_E - Y_E|$ poziții.

Alinierea lui Y_M se poate face:

- cu precizie infinită: sunt păstați 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}

Cei 3 biți păstrați la alinierea lui Y_M cu precizie finită se numesc *biți sticky*:

- g: bit de gardă, cu ponderea 2^{-m}; protejează împotriva pierderii de precizie
- r: bit de rotunjire, cu ponderea 2^{-m-1}; utilizat pentru rotunjirea rezultatului
- s: bitul sticky, cu ponderea 2^{-m-2}; se obține ca rezultat al unui *SAU* logic între toți ceilalți biți mai puțin semnificativi care au fost deplasați la dreapta din Y_M, exceptând biții g și r

După aliniere, Y_M devine Y_{Mal}.

Se consideră rezultatul FP al unei adunări $\Rightarrow Z_M = X_M + Y_{Mal}$, cu condiția ca $X_E \geq Y_E$.

Deoarece adunarea significanzilor poate genera un transport de ieșire, rezultă că forma lui Z_M este:

$$Z_M = z_m \ z_{m-1} \ . \ z_{m-2} \ z_{m-3} \ldots z_1 \ z_0 \mid g \ r \ s$$

Operația de normalizare pentru Z_M va produce Z_{M_n}, cu

$$Z_{M_n} = 1 \ . \ z_{m-2_n} \ z_{m-3_n} \ldots z_{1_n} \ z_{0_n} \mid R \ S$$

Cei doi biți, R și S, sunt necesari pentru efectuarea operației de rotunjire, succesive normalizării.

3.3 Reguli de normalizarea și rotunjirea unui rezultat FP (contin.)

Cazuri de normalizare:

$Z_{M_n} =$		1.	z_{m-2_n}	z_{m-3_n}	$\ldots$	z_{1_n}	z_{0_n}	R	S
Caz 1)	$z_m = 1$	1.	z_{m-1}	z_{m-2}	$\ldots$	z_2	z_1	z_0	$(g\ \underline{OR}\ r\ \underline{OR}\ s)$
$\Rightarrow$ depl. dreapta 1 bit, $Z_E{+}{+}$									
Caz 2)	$z_m = 0$, $z_{m-1} = 1$	1.	z_{m-2}	z_{m-3}	$\ldots$	z_1	z_0	g	$(r\ \underline{OR}\ s)$
$\Rightarrow$ Z_M este deja normalizat									
Caz 3)	$z_m = 0$, $z_{m-1} = 0$, $z_{m-2} = 1$	1.	z_{m-3}	z_{m-4}	$\ldots$	z_0	g	r	s
$\Rightarrow$ depl. stânga 1 bit, $Z_E{-}{-}$									
Caz 4)	$z_m = 0$, $z_{m-1} = 0$, $z_{m-2} = 0$, $z_{m-3} = 1$	1.	z_{m-4}	z_{m-5}	$\ldots$	g	0	0	0
$\Rightarrow$ depl. stânga 1 bit, $Z_E{-}{=}2$									

Dacă normalizarea lui Z_M necesită o operație de deplasare la stânga cu 2 sau mai mulți biți:

- se atașază bitul g lui Z_M, după noua poziție a bitului z_0
- se completeaza pozitiile libere mai puțin semnificative din Z_M cu 0
- se alege $R = S = 0$

Rotunjirea lui Z_{M_n} utilizează biții R și S determinați anterior:

- sunt folosiți R și S pentru implementarea tuturor celor 4 moduri de rotunjire

Reguli de rotunjire

Mod de rotunjire	$Z_{M_n} > 0$	$Z_{M_n} < 0$
spre 0	——— (sunt neglijați R și S)	———
spre $-\infty$	———	$if(R\ \underline{or}\ S)\ then\ Z_{M_n} - 1$
spre $+\infty$	$if(R\ \underline{or}\ S)\ then\ Z_{M_n} + 1$	———
la cel mai apropiat nr. par	$if(R\ \underline{and}(S\ \underline{or}\ z_{0_n}))\ then\ Z_{M_n} + 1$	$if(R\ \underline{and}(S\ \underline{or}\ z_{0_n}))\ then\ Z_{M_n} - 1$

Eroarea de rotunjire nu este corelată cu diferența exponenților celor 2 operanzi (vezi Fig. 5.12 din [Vlad12]).

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 fracționară a significandului
- aceeași relație de calcul al bias-ului ca în IEEE 754
  $bias = 2^{e-1} - 1 = 3$
- aceleași excepții ca în IEEE 754

Se consideră următorii operanzi:

$$X = 0.5625_{(10)} = 0.1001_{(2)} = 1.\underbrace{001} \cdot 2^{-1}$$

$$Y = -3.75_{(10)} = -11.11_{(2)} = -1.\underbrace{111} \cdot 2^{1}$$

Formatul operanzilor împachetați:

$$X: \boxed{0}\underbrace{\boxed{0}\boxed{1}\boxed{0}}_{-1 + bias = 2_{10} = 010_2}\overbrace{\boxed{.\,0}\boxed{0}\boxed{1}}$$

$$Y: \boxed{1}\underbrace{\boxed{1}\boxed{0}\boxed{0}}_{1 + bias = 4_{10} = 100_2}\overbrace{\boxed{.\,1}\boxed{1}\boxed{1}}$$

Algoritmul de adunare cu rotunjire a numerelor de virgulă mobilă:

Pasul 1: Despachetare operanzi

- atașare bit ascuns
- verificare excepții: unul din operanzi este 0, $\pm\infty$, *NaN*

X : | 0 | 0 | 1 | 0 | 1 | .0 | 0 | 1 |

Y : | 1 | 1 | 0 | 0 | 1 | .1 | 1 | 1 |

Pasul (2): Calcularea diferenței exponenților, $d = X_E - Y_E$

- dacă $d < 0$ rezultă că $|X| < |Y|$
  - 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).

Calculează $d = X_E - Y_E = -2$. Pentru că d este negativ $\Rightarrow$ INTERSCHIMBĂ operanzii și alege $Z_E = Y_E = 4$.

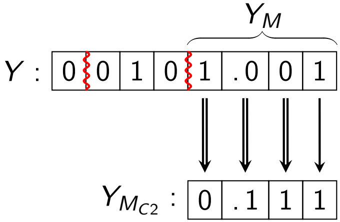
X : | 1 | 1 | 0 | 0 | 1 | .1 | 1 | 1 |

Y : | 0 | 0 | 1 | 0 | 1 | .0 | 0 | 1 |

3.4 Adunarea/scăderea cu rotunjire a numerelor FP (contin.)

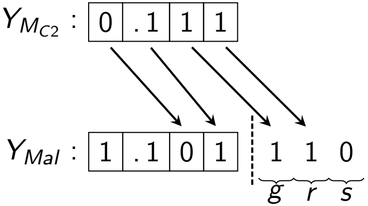
Pasul 3: Dacă $sign(X) \neq sign(Y) \Rightarrow$ se complementează de doi Y_M

- semne diferite implică operația de scădere
  - complementarea de doi permite utilizarea unui sumator binar pentru scăderea significanzilor
- se complementează de doi doar significandul Y_M (reduce aria unității FP evitând includerea aceleași facilități si pentru X_M)



Pasul 4: Se aliniază Y_M prin deplasare la dreapta cu $|d|$ poziții, Y_M după 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



Pasul 5: Se adună cei 2 significanzi: $Z_M = X_M + Y_{Mal}$

- dacă $sign(X) = sign(Y)$ potențialul bit de transport de ieșire 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

$$
\begin{array}{rccccc|ccc|c}
 & & & & & & g & r & s & \\
X_M: & & 1 & .1 & 1 & 1 & & & & \\
Y_{Mal}: & & 1 & .1 & 0 & 1 & 1 & 1 & 0 & + \\ \hline
Z_M: & \underbrace{\cancel{1}}_{c_{out}} & 1 & .1 & 0 & 0 & 1 & 1 & 0 &
\end{array}
$$

Pasul 6: Pre-normalizare

- relizată pe baza cazurilor de normalizare din secțiunea 3.3
  - este determinat Z_{M_n}
  - Z_E poate fi modificat
- verificarea excepțiilor:
  - 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 stănga cu cel puțin 1 bit $\Rightarrow$ *Underflow*

Z_M este deja normalizat (cazul de normalizare 2 din secțiunea 3.3)

Z_M : | 1 | .1 | 0 | 0 ¦ 1 | 1 | 0 | (din Pasul 5)

Z_M este deja normalizat

Z_{M_n} : | 1 | .1 | $\underbrace{0 \mid 0}_{z_{0_n}}$ |

$Z_E = 4$ (din Pasul 2)

Pasul 7: Se determină valorile biților R și S (pe baza acelorași cazuri de normalizare din secțiunea 3.3)

$$Z_M : \boxed{1}\boxed{.1}\boxed{0}\boxed{0}\,\underbrace{\boxed{1}}_{g}\underbrace{\boxed{1}}_{r}\underbrace{\boxed{0}}_{s} \text{ (din Pasul 5)}$$

Z_M este deja normalizat

$R = g = 1$

$S = (r\ \underline{or}\ s) = 1$

Pasul 8: Rotunjirea lui Z_{M_n} obținându-l astfel pe $Z_M^\star$:

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

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

$$\begin{array}{rc|c|c|c|c|l}
\cline{3-6}
Z_{M_n}: & & 1 & .1 & 0 & 0 & \\
\cline{3-6}
 & & & & & 1 & + \\
\hline
Z_M^\star: & \underbrace{0}_{c_{out}} & 1 & .1 & 0 & 1 & \\
\cline{3-6}
\end{array}$$

$Z_E = 4$ (din Pasul 6)

Pasul 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	-	+	$-$

Important: Coloanele $sign(X)$ și $sign(Y)$ fac referire la semnele inițiale ale operanzilor (din Pasul 1).

Pentru exemplul considerat, din cauza interschimbării din Pasul 2, potrivit primei linii din tabelul de mai sus $\Rightarrow sign(Z) = -$

3.4 Adunarea/scăderea cu rotunjire a numerelor FP (contin.)

Pasul 10: Împachetarea rezultatului

Utilizează câmurile rezultatului Z, determinate anterior:

significandul lui Z: $\boxed{1} \underbrace{\boxed{.1}\boxed{0}\boxed{1}}$

exponentul lui Z: $4_{10} = \underbrace{100_2}$

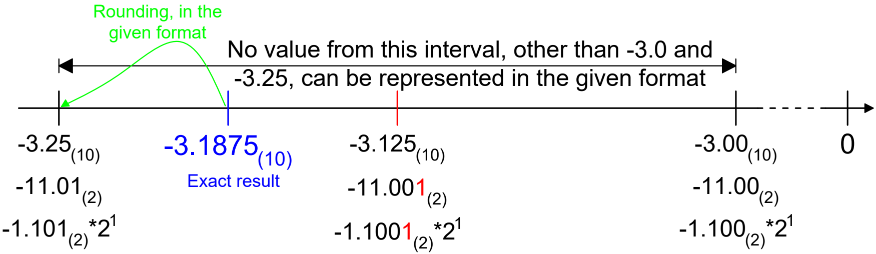
semnul lui Z: $-$

Z împachetat: $\boxed{1}\overbrace{\boxed{1}\boxed{0}\boxed{0}}\overbrace{\boxed{.1}\boxed{0}\boxed{1}}$

Verificare: Determinarea rezultatului exact:
$X + Y = 0.5625 - 3.75 = -3.1875.$

Valoarea lui Z:
$Z = (-1)^{Z_{sign}} * 2^{Z_E - bias} * Z_M = -1 * 2^{4-3} * 1.101_2 = -3.25$

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.



Proiectarea unui dispozitiv pentru pre-normalizare:

- acoperă pașii 6 și 7
- proiectat după regulile din secțiunea 3.3
- design combinațional

Rezultatul pasului 5, pentru formatul considerat, este:

$$Z_M = z_{4_n} z_{3_n}\ .\ z_{2_n}\ z_{1_n} z_{0_n} \mid g\ r\ s$$

Rezultatul pasului 6 combinat cu pasul 7 este:

$$Z_{M_n} = 1\ .\ z_{2_n}\ z_{1_n} z_{0_n} \mid R\ S$$

Pornind de la regulile de normalizare din secțiunea 3.3, pentru formatul dat, Z_M poate necesita:

- deplasare la dreapta cu 1 bit (r_1), sau
- pastrarea fără modificare (rezultat gata normalizat) (l/r_0), sau
- deplasare la stânga cu 1 bit (l_1), sau
- deplasare la stânga cu 2 biți (l_3), sau
- deplasare la stânga cu 3 biți (l_3)

Cele 5 cazuri de normalizare sunt identificate de cele 5 condiții/variabile între paranteze: r_1, l/r_0, l_1, l_2, l_3.

Una și doar una dintre cele 5 condiții va fi activă pentru o pereche dată de operanzi FP de însumat după algoritmul de adunare cu rotunjire.

Cazurile de normalizare împreună cu condițiile asociate:

Z_{M_n}	1.	z_{2_n}	z_{1_n}	z_{0_n}	R	S
Z_M este normalizat (l/r_0)	1.	z_2	z_1	z_0	g	$(r \ \underline{OR} \ s)$
Z_M depl. stânga 1 bit (l_1)	1.	z_1	z_0	g	r	s
Z_M depl. stânga 2 bit (l_2)	1.	z_0	g	0	0	0
Z_M depl. stânga 3 bit (l_3)	1.	g	0	0	0	0
Z_M depl. dreapta 1 bit (r_1)	1.	z_{m-1}	z_{m-2}	z_1	z_0	$(g \ \underline{OR} \ r \ \underline{OR} \ s)$

În consecință, se pot scrie ecuații booleene pentru cei 5 biți generați la ieșirea pasului 6 combinat cu pasul 7:

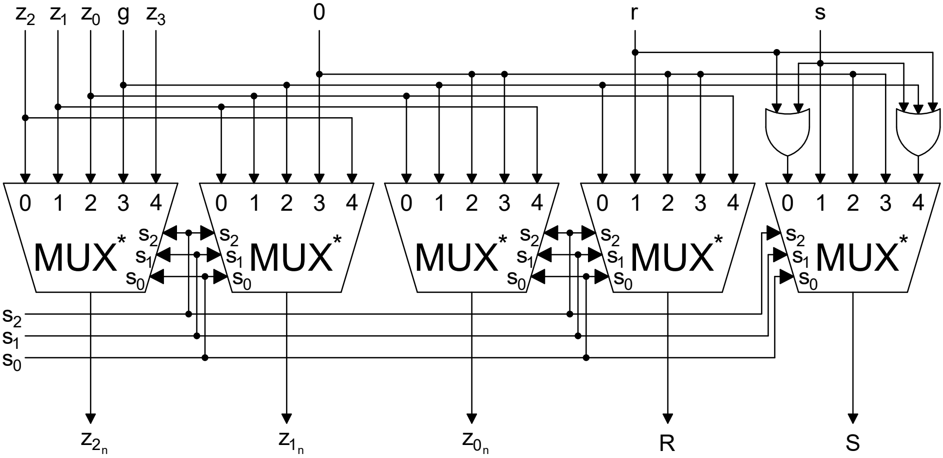
$$\begin{array}{lclllllll}
z_{2_n} & = & z_2 & \cdot l/r_0 + z_1 & \cdot l_1 + z_0 & \cdot l_2 + g & \cdot l_3 & + z_3 & \cdot r_1 \\
z_{1_n} & = & z_1 & \cdot l/r_0 + z_0 & \cdot l_1 + g & \cdot l_2 & & + z_2 & \cdot r_1 \\
z_{0_n} & = & z_0 & \cdot l/r_0 + g & \cdot l_1 & & & + z_1 & \cdot r_1 \\
R & = & g & \cdot l/r_0 + r & \cdot l_1 & & & + z_0 & \cdot r_1 \\
S & = & (r \ \underline{OR} \ s) & \cdot l/r_0 + s & \cdot l_1 & & & + (g \ \underline{OR} \ r \ \underline{OR} \ s) & \cdot r_1
\end{array}$$

Deoarece, pentru o pereche de operanzi FP adunați prin algoritmul de adunare cu rotunjire, una și doar una din cele 5 condiții poate fi activă $\Rightarrow$ acestea pot fi codificate pe mai puțini biți.

Se consideră variabilele s_2, s_1 și s_0 pentru codificarea celor 5 condiții. Condificarea este descrisă în tabelul următor:

Intrări					Ieșiri		
r_1	l_3	l_2	l_1	l/r_0	s_2	s_1	s_0
0	0	0	0	1	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	1	0	0	0	0	1	1
1	0	0	0	0	1	0	0

Arhitectura dispozitivului pentru pre-normalizare este descrisă mai jos:



Simbolurile *MUX*⋆ sunt multiplexoare incomplete, cu 3 linii de selecție dar având doar 5 intrări de date.

[Vlad12] M. Vlăduțiu, *Computer Arithmetic: Algorithms and Hardware Implementations*. Springer, 2012.