3.5 F.p. addition/substraction with rounding.

- Rounding error is not correlated with exponents' difference.

$1.75 \times 2^{32} + 1.25 \times 2^{25}$ → 4

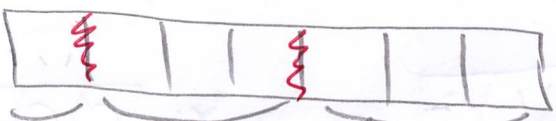
$1.75 \times 2^{32} - 1.25 \times 2^{31}$ → 1

- Use a simplified floating point format
  - inspired by IEEE 754 standard
  - fewer bits are used:
    - ⇒ scaled down format
    - ⇒ exceptions
    - ⇒ same exponent for f.p. 0's value.

1 bit for sign
3 bits for exponent field
4 bits for the significand ⇒ 1 hidden bit ⇒ 3 fr. part bits

IEEE 754 – single precision: 1 – sign, 8 – exp, 24 – signif.

Packed format



sign | exponent | fractional part of significand

Consider: bias $= 2^{e-1} - 1 = 2^{3-1} - 1 = 3$

$X = 0.5625_{(10)} = 0.1001_{(2)} \times 2^{0} = 1.001 \times 2^{-1}$ (exponent: $0 + 3 = 3$; $-1 + 3 = 2$)

$Y = -3.75_{(10)} = 11.11_{(2)} \times 2^{0} = 1.111 \times 2^{1}$ (exponent: $0 + 3 = 3$; $1 + 3 = 4$)

Packed operands

X: 0 | 0 1 0 | 0 0 1

Y: 1 | 1 0 0 | 1 1 1

F.p. addition with rounding – Algorithm

Step 1 Unpack operands
- adding the hidden bit (supplementary bit)
- check for exceptions:
  - one or both the operands are $0, \pm\infty$, NaN, ...)

X: 0 | 0 1 0 | 1 0 0 1 (hidden bit)

$X_e = 2$ X_m

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

$Y_e = 4$ Y_m

Step 2 Compute exponents' difference, $d = X_e - Y_e$

- if $d < 0 \Rightarrow |X| < |Y|$
  - → SWAP $X \leftrightarrow Y$
  - → chose Z_e as Y_e
- if $d \geq 0$
  - → chose Z_e as X_e

Save area → reduce complexity by only allowing Y_n to be Right Shifted for alignment.

$d = X_e - Y_e = 2 - 4 = -2$!

⇒ SWAP $X \leftrightarrow Y$, $Z_e = Y(Y_e)$

alligment / $Y_e \to X_e$

$X \pm Y = (X_n \pm Y_n \cdot 2) \cdot 2^{X_e}$ if $X_e \geq Y_e$

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

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

Step 3 if $\text{sign}(X) \neq \text{sign}(Y)$ → Complement of 2 for Y_n

Save area → reduce complexity by only allowing Y_n to be Complemented of 2

$\text{sign}(X) = 1\ (-)$
$\text{sign}(Y) = 0\ (+)$
⇒ Complement of 2 for Y_n

$X < 0 \to X = -|X|$
$Y \geq 0 \quad Y = |Y|$
$X + Y = -|X| + |Y|$

$-A \equiv$ Complement of 2 to value of A

subtraction

for BCD addition:
$-10_{(10)} \equiv +6_{(10)}$
$10_{(10)} =$ | 1 | 0 | 1 | 0 |
Complementation of 2: ↓ ↓ ↓ ↓
$6_{(10)} =$ 0 1 1 0

$|X| + (-|Y|)$ Complement of 2
$= -(-|X| + |Y|)$

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

$Y_{n_{C2}} = 0.1\ 1\ 1$

Step 4. Allign y_n.: RShift of y_n by 1 dl bits.

- if y_n was Complemented of 2 in Step 3:
  - introduce bits of 1's in y_n's mob while RShifting instead of 0's
  - preserve the sticky bit: g, r, s

$y_{n_{C2}} = \boxed{0.111}$

$d = -2$

was y_n C2 in Step 3? YES

$\Rightarrow$ RShift $y_{n_{C2}}$ by $1-21 = 256$ bits
- introduce values of 1's while RShifting.

$y_{n_{C2\,al}} = \underbrace{1.1\boxed{01}}_{}\,\underbrace{\boxed{11}}_{g\ r}\,\underbrace{0}_{s}$

Step 5 Add the 2 significands: $z_n = x_n + y_{n_{al}}$

- if sign(x) == sign(y)
  - if cout is generated, preserve it
- if sign(x) ≠ sign(y)
  - if no cout is generated $\Rightarrow$ result's significand is negative
    $\rightarrow$ Complement of 2 the z_n (result's significand)
  - if cout is generated $\Rightarrow$ result's significand is positive
    $\rightarrow$ discard cout bit

$$\begin{array}{lll} x_n = & 1.111 & \overset{g}{1}\ \overset{r}{1}\ \overset{s}{1} \\ y_{n_{cal}} = & 1.101 & 1\ 1\ 0 \\ \hline z_n = \cancel{1} & 1.100 & 1\ 1\ 0 \end{array} \quad +$$

cout

z_n is positive and sign(x) ≠ sign(y) $\Rightarrow$ discard cout

Step 6 Renormalization.
- according to the rules from 3.4.
  - determine z_{m_n}, can update z_E
  - exception checking:
    - if $z_E == z_{E_{max}}$ ($2^3-2=6$) and z_m requires a 1-bit RShift $\Rightarrow$ Overflow
    - if $z_E == z_{E_{min}}$ (1) and z_m requires 1 or more bit LShift $\Rightarrow$ Underflow

ignored Cout from Step 5

$z_m = 01.100 \mid 110$ (g, r, s)

$\Rightarrow$ Rule 2) from 3.4

$z_{m_n} = 1.100$ (z_{3n} z_{2n} z_{1n} z_{0n}) $\quad z_E = 4$

Step 7 Calculate values of R, S
- get them from the one rule of 3.4 as the one applied in Step 6

Rule 2) from 3.4.

$\Rightarrow R = g = 1$

$S = (r \text{ or } s) = 1 \underset{\text{logic or}}{+} 0 = 1$

Step 8 Rounding of z_m: determine z_m^* from z_{m_n}
- takes into account the IEEE 754 rounding mode set for the f.p. addition.
- according to the Rounding rules from 3.4.
- if rounding generates a Cout $\Rightarrow$ postnormalization.

Postnormalization: - 1-bit RShift of z_m^*
- $z_E + +$ $\Rightarrow$ possible OVERFLOW

Consider the rounding mode to be set to
round to nearest even

condition is: R and (S or rem)

rem = 0
R = 1
S = 1

$R \cdot (S + rem) = 1(1+0) = 1$ = TRUE

$\Rightarrow m_n ++$

$m_m = 1.100 +$

cout 1

$m_n^* \quad 01.101$

no carry always generated
$\Rightarrow e_b$ is not modified.

$m_n^* = 1.101 \qquad e_b = 4$

Step 9 Choose the result's sign.

- if $sign(x) == sign(y) \Rightarrow sign(z) = sign(x)$
- otherwise.

SWAP (Step 2)	Complementation of 2 (Step 5)	sign(X)	sign(Y)	sign(Z)
YES		+	−	−
YES		−	+	+
NO	YES	+	−	−
NO	YES	−	+	+
NO	NO	+	−	+
NO	NO	−	+	−

$Y_e \geq X_e$ (Y)
$|Y| \geq |X|$

$X_e \geq Y_e$

SWAP? YES $\Rightarrow sign(z) = sign(y)$
↳ before the SWAP

$\Rightarrow sign(z) = 1 (-)$

Step 10 Pack the result

1 | 1 | 0 | 0 . 1 | 0 | 1

$\therefore sign = 1$
exponent = 4
significand = 1.101

Verification:

$X = 0.5625$ $\quad Y = -3.75$

$Z = X + Y = -3.1875$ (with infinite precision)

Z | 1 | 1 | 0 | 0 | 1 | 0 | 1 |

unpacking: $sign = 1$

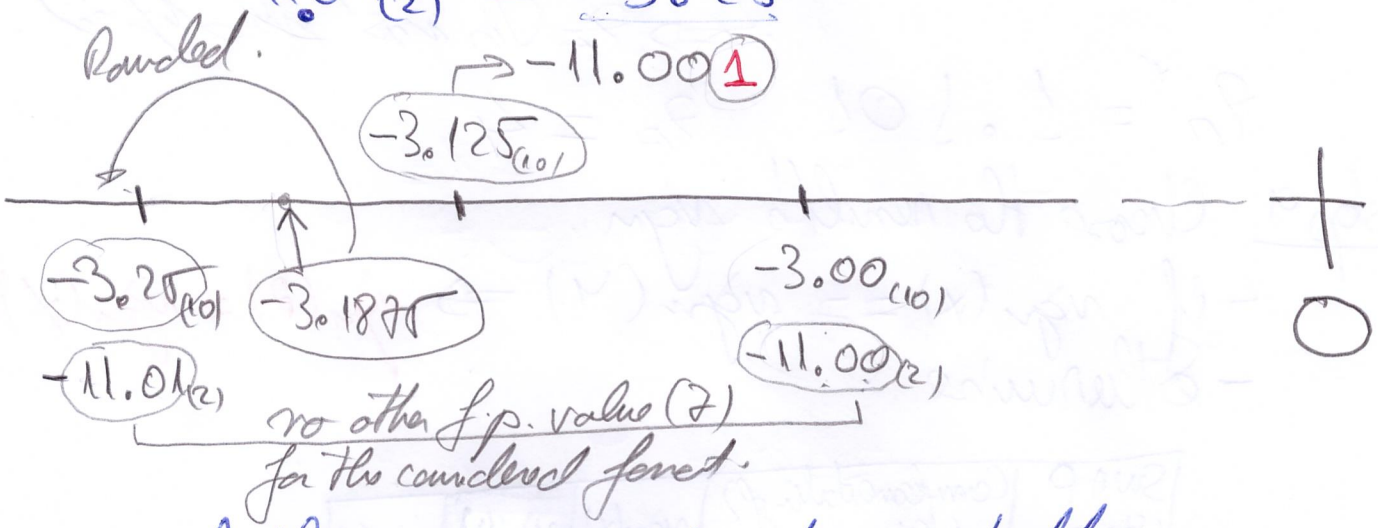
$exp = 100 = 4$

$Z_m = 1.101$ (hidden bit)

$Z = (-1)^{sign} * 2^{exp - bias} * Z_m =$

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

$= -11.01_{(2)} = -3.25$



Design of the personalization shifters

- cases Step6 and Step7
  → according to the rules for 3.4
- purely combinational design.

Result of Step5 (possible carry)

$Z_n = \widehat{z_4}\ z_3\ z_2\ z_1\ z_0 \mid g\ r\ s$

Output of Step6 & Step7

$Z'_m = 1.z_3 z_2 z_1 z_0 \mid R\ S$

Normalization cases.

$z_{m} = 1.\ z_{3n}\ z_{2n}\ z_{1n}\ z_{0n} \mid R\ \ S$

A) z_n is already normalized (ll_{z_0}) — $1.\ z_2\ z_1\ z_0 \mid q\ \ (z_2 \& z_1)$

B) z_n needs a 1-bit LShift (l_1) — $1.\ z_1\ z_0\ q \mid r\ \ 1$

C) z_n needs a 2-bit LShift (l_2) — $1.\ z_0\ q\ 0 \mid 0\ \ 0$

D) z_n needs a 3-bit LShift (l_3) — $1.\ q\ 0\ 0 \mid 0\ \ 0$

E) z_n needs a 1-bit RShift (r_1) — $1.\ z_3\ z_2\ z_1 \mid z_0\ (q \& r \& 1)$

Associate each of the 5 cases to a boolean variable

$$z_{2n} = z_2 \cdot ll_{z_0} + z_1 \cdot l_1 + z_0 \cdot l_2 + q \cdot l_3 + z_3 \cdot r_1$$
$$+ z_2 \cdot r_1$$
$$+ z_1 \cdot r_1$$
$$+ z_0 \cdot r_1$$
$$+ (q \& r \& 1) \cdot r_1$$

$$z_{1n} = z_1 \cdot ll_{z_0} + z_0 \cdot l_1 + q \cdot l_2$$

$$z_{0n} = z_0 \cdot ll_{z_0} + q \cdot l_1$$

$$R = q \cdot ll_{z_0} + r \cdot l_1$$

$$S = (r \& 1) \cdot ll_{z_0} + 1 \cdot l_1$$