2.5 Floating point division

$$y_1 = M_1 \times^{E_1}$$
$$y_2 = M_2 x^{E_2}$$

$$z = (M_1 \div M_2) \times^{(E_1 - E_2)}$$

1. Normalise y_1 & y_2
2. Unsign if necessary
3. Divide mantissas.
4. Subtract exponents.
5. Resign (if necessary)
6. Normalise

Example

1.0100110 e 0010

÷ 0.0010110 e 0100

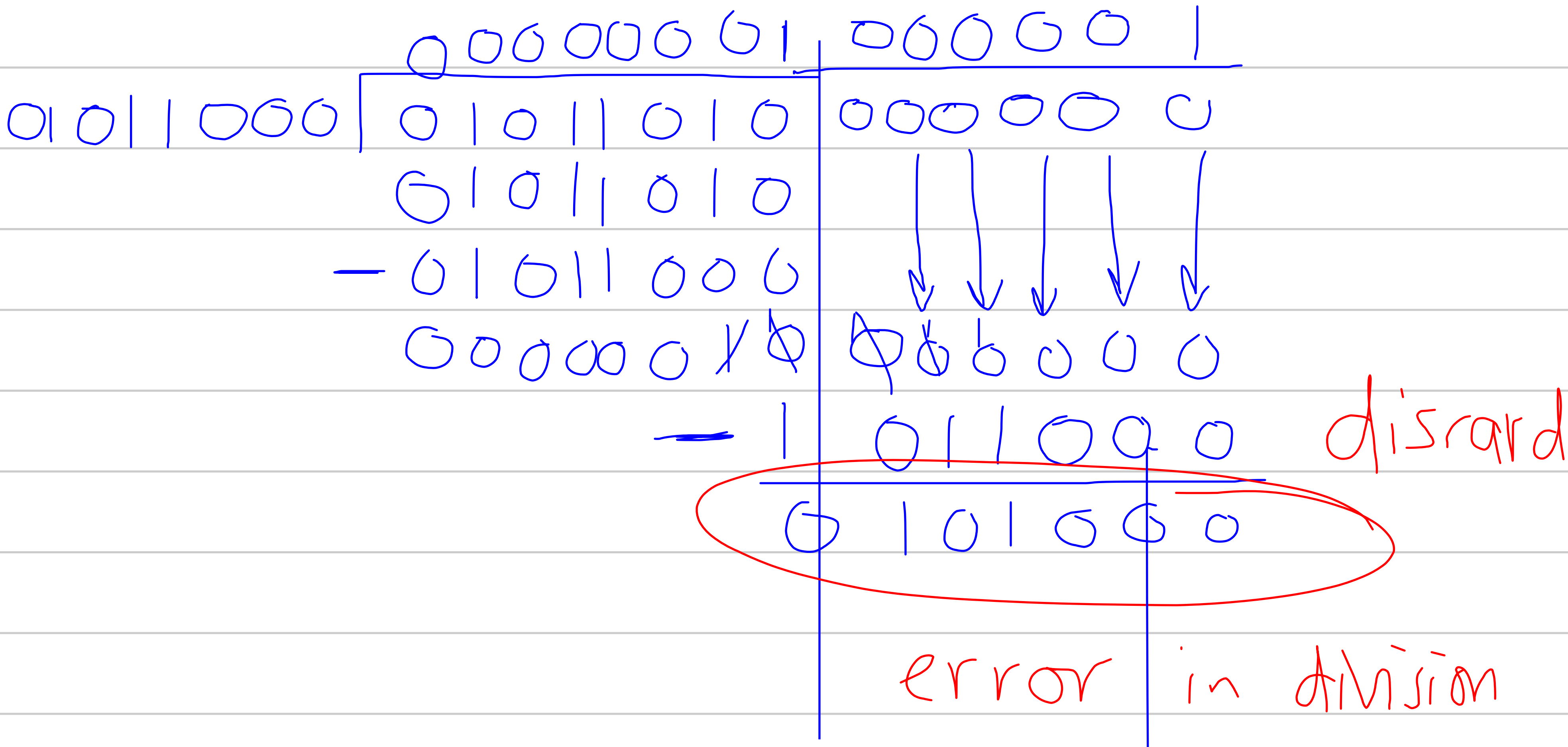
Unsign: 1.0100110 e 0010 => 0.1011010 e 0010 (Normalised)

Normalise #2: 0.1011000 e 0010

Example continued:

$M_1 = 0.1011010$

$M_2 = 0.1011000$



$M_1 \div M_2 = 001.0000001 \; e \; 0000$

Reformatting into Q0.7

$\therefore \; M_1 \div M_2 = 0.10000001 \; e \; 0001$

Subtract exponents:

$E_1 - E_2$

$$\begin{array}{r} 0010 \\ -\,0010 \\ \hline 0000 \end{array}$$

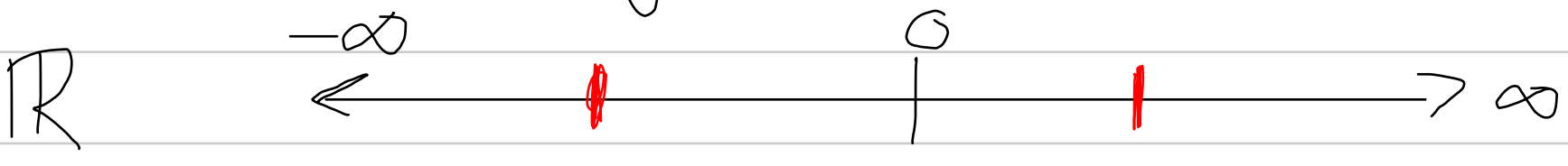
Example continued

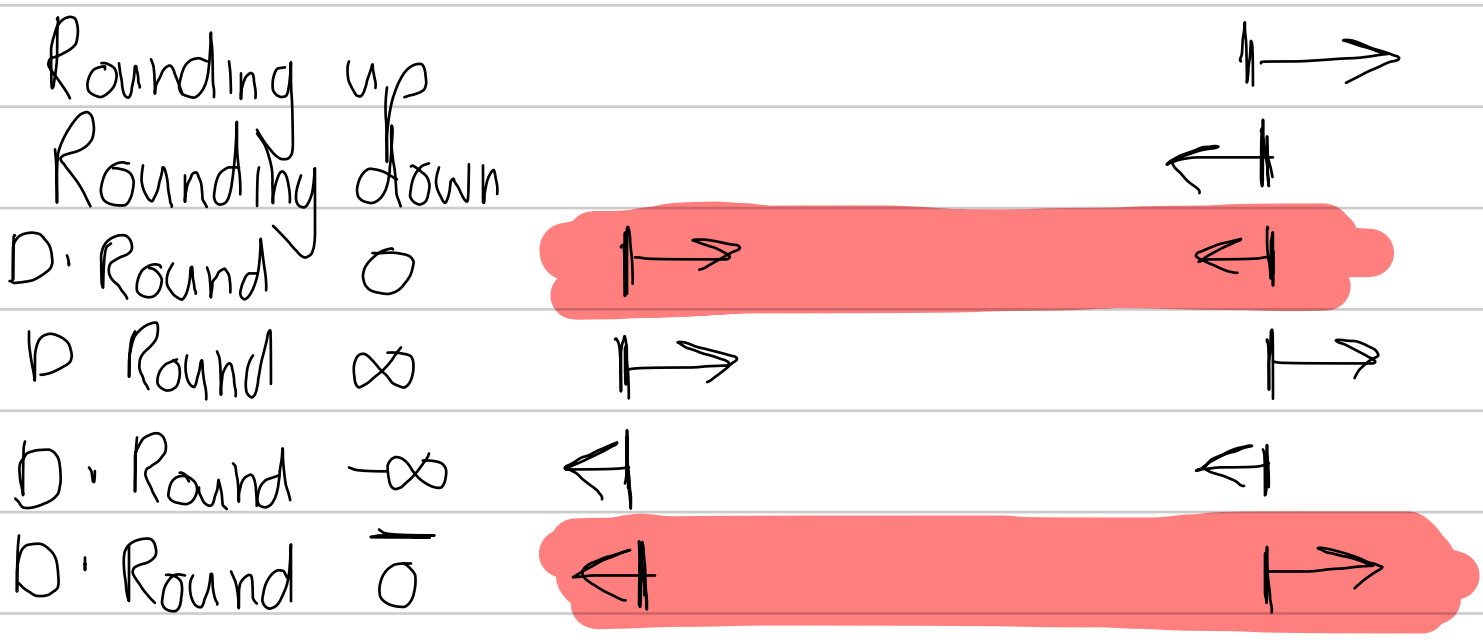
$0.1000001_e(0001 + 0000)$

Resign since y_1 was negative

$z = 1.0111111_e 0001$ ⟶

2.6 Rounding and Guard bits





Guard digits

In Q 0.m subtraction/addition adding a guard bit <u>after</u> the LSB improves precision.

Add guard digit before normalisation step

Round guard digit towards 0.

* $0.101e001 - 0.011e000$

Without guard digit	With guard digit
$0.101e001$	$0.101\ 0\ e001$
$-\ 0.001e001$	$0.001\ 1\ e001$
$0.100e001$	$0.011\ 1\ e001$

$0.101e001 = 1.25_{10}$

$0.011e000 = 0.375_{10}$

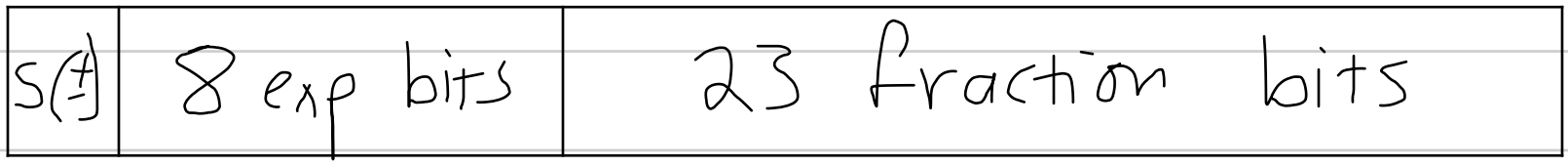
$\Rightarrow 1.25_{10} - 0.375 = 0.875$

$0.100e001 = 1_{10}$

$0.111\ 0\ e000 = 0.875$

2.7 IEEE Floating point format

First proposed in 1985. Now present in every mobile device, PC etc... Is the format used to store data on fixed/permanent storage.



1 Sign bit
8 Exponent bits Q7.0 signed
23 Fractional bits Q0.23 unsigned

Exponent has a bias of 127:

$E_{min} = 1 - 127 = -126$

$E_{max} = 254 - 127 = +127$

(offset of 127 subtracted from stored value)

Stored exponent of 0x00 and 0xFF interpreted differently.

Exponent	Mantissa 0	Mantissa Non-0	Equation
0x00	+0/-0	sub-normal	$(-1)^{sb} \times 2^{-126} \times 0.FB$
0x01 ⇒ 0xFE	Normalised value		$(-1)^{sb} \times 2^{e-127} \times 1.FB$
0xFF	$+\infty/-\infty$	NaN	Quiet signalling

$+1_{10} \rightarrow$ 3F80 0000 (IEEE FP)