Finite Precision Systems

Finite precision systems

In scientific notation numbers are represented as

$$x = \pm s \times \beta^E$$

where, $s \rightarrow$ significand; $\beta \rightarrow$ base; $E \rightarrow$ exponent;

Finite precision systems

In scientific notation numbers are represented as

$$x = \pm s \times \beta^E$$

where, $s \rightarrow$ significand; $\beta \rightarrow$ base; $E \rightarrow$ exponent;

The representation is not unique as, for example in base 10,

$$55 = 5.5 \times 10 = 0.55 \times 10^2 = \cdots$$

Finite precision systems

In scientific notation numbers are represented as

$$x = \pm s \times \beta^E$$

where, $s \rightarrow$ significand; $\beta \rightarrow$ base; $E \rightarrow$ exponent;

The representation is not unique as, for example in base 10,

$$55 = 5.5 \times 10 = 0.55 \times 10^2 = \cdots$$

Setting the restriction $1 \leq s < \beta$, makes the representation unique for all nonzero numbers.

Finite precision systems

In scientific notation numbers are represented as

$$x = \pm s \times \beta^E$$

where, $s \rightarrow$ significand; $\beta \rightarrow$ base; $E \rightarrow$ exponent;

The representation is not unique as, for example in base 10,

$$55 = 5.5 \times 10 = 0.55 \times 10^2 = \cdots$$

Setting the restriction $1 \leq s < \beta$, makes the representation unique for all nonzero numbers. This is called the *normalized representation of a number.*

Finite precision systems

In scientific notation numbers are represented as

$$x = \pm s \times \beta^E$$

where, $s \rightarrow$ significand; $\beta \rightarrow$ base; $E \rightarrow$ exponent;

The representation is not unique as, for example in base 10,

$$55 = 5.5 \times 10 = 0.55 \times 10^2 = \cdots$$

Setting the restriction $1 \leq s < \beta$, makes the representation unique for all nonzero numbers. This is called the *normalized representation of a number.*

Therefore, in a normalized representation,

$$x = \pm \underbrace{(b_0.b_1 b_2 \cdots)_\beta}_{=s} \times \beta^E,$$

where $1 \leq b_0 \leq \beta - 1, 0 \leq b_j \leq \beta - 1$, for $j \geq 1$.

Finite precision system

Consider a finite precision system $(\beta, p, E_{\text{min}}, E_{\text{max}})$ indicating that numbers stored are 0 and all other normalized numbers in base β, with s stored upto p places and E_{min} and E_{max} being the smallest and largest values of the exponent.

Finite precision system

Consider a finite precision system $(\beta, p, E_{\min}, E_{\max})$ indicating that numbers stored are 0 and all other normalized numbers in base β, with s stored upto p places and $E_{\min}$ and $E_{\max}$ being the smallest and largest values of the exponent. Here

$$p \rightarrow \textit{precision of the system}$$

The nonzero numbers in the system are

$$x = \pm(b_0.b_1 \cdots b_{p-1})_\beta \times \beta^E,\ 1 \leq b_0 \leq \beta - 1, E_{\min} \leq E \leq E_{\max}.$$

They will be called *(normalized) floating point numbers and they satisfy* $N_{\min} \leq |x| \leq N_{\max}$, where

Finite precision system

Consider a finite precision system $(\beta, p, E_{\min}, E_{\max})$ indicating that numbers stored are 0 and all other normalized numbers in base β, with s stored upto p places and $E_{\min}$ and $E_{\max}$ being the smallest and largest values of the exponent. Here

$$p \rightarrow \textit{precision of the system}$$

The nonzero numbers in the system are

$$x = \pm(b_0.b_1 \cdots b_{p-1})_\beta \times \beta^E,\ 1 \leq b_0 \leq \beta - 1, E_{\min} \leq E \leq E_{\max}.$$

They will be called *(normalized) floating point numbers and they satisfy* $N_{\min} \leq |x| \leq N_{\max}$, where

$N_{\min} = \beta^{E_{\min}} \rightarrow$ smallest positive normalized floating point number

$N_{\max} = (\beta - \beta^{-p+1})\beta^{E_{\max}} \rightarrow$ largest normalized floating point number.

Finite precision system

Consider a finite precision system $(\beta, p, E_{\min}, E_{\max})$ indicating that numbers stored are 0 and all other normalized numbers in base β, with s stored upto p places and $E_{\min}$ and $E_{\max}$ being the smallest and largest values of the exponent. Here

$$p \rightarrow \textit{precision of the system}$$

The nonzero numbers in the system are

$$x = \pm(b_0.b_1 \cdots b_{p-1})_\beta \times \beta^E,\ 1 \leq b_0 \leq \beta - 1, E_{\min} \leq E \leq E_{\max}.$$

They will be called *(normalized) floating point numbers and they satisfy* $N_{\min} \leq |x| \leq N_{\max}$, where

$N_{\min} = \beta^{E_{\min}} \rightarrow$ smallest positive normalized floating point number

$N_{\max} = (\beta - \beta^{-p+1})\beta^{E_{\max}} \rightarrow$ largest normalized floating point number.

Any other number that is not a floating point number has to be rounded by a floating point number by some rule of rounding.

Finite precision systems

Suppose $x = \pm(b_0.b_1\cdots)_\beta \times \beta^E$, $1 \leq b_0 \leq \beta - 1$.

Finite precision systems

Suppose $x = \pm(b_0.b_1\cdots)_\beta \times \beta^E,\ 1 \le b_0 \le \beta - 1.$

The options for rounding x by a floating point number in $(\beta, p, E_{\min}, E_{\max})$ are

$x_- \rightarrow$ the largest floating point number less than or equal to x.

$x_+ \rightarrow$ the smallest floating point number greater than or equal to x.

Finite precision systems

Suppose $x = \pm(b_0.b_1 \cdots)_\beta \times \beta^E$, $1 \leq b_0 \leq \beta - 1$.

The options for rounding x by a floating point number in $(\beta, p, E_{\min}, E_{\max})$ are

$x_- \rightarrow$ the largest floating point number less than or equal to x.

$x_+ \rightarrow$ the smallest floating point number greater than or equal to x.

Therefore for $N_{\min} \leq |x| \leq N_{\max}$,

$$x_- = \begin{cases} (b_0.b_1 \cdots b_{p-1})_\beta \times \beta^E & \text{if } x > 0, \\ -[(b_0.b_1 \cdots b_{p-1})_\beta + \beta^{1-p}] \times \beta^E & \text{if } x < 0, \end{cases}$$

$$x_+ = \begin{cases} [(b_0.b_1 \cdots b_{p-1})_\beta + \beta^{1-p}] \times \beta^E & \text{if } x > 0, \\ -(b_0.b_1 \cdots b_{p-1})_\beta \times \beta^E & \text{if } x < 0 \end{cases}$$

Finite precision systems

Suppose $x = \pm(b_0.b_1 \cdots)_\beta \times \beta^E$, $1 \leq b_0 \leq \beta - 1$.

The options for rounding x by a floating point number in $(\beta, p, E_{\min}, E_{\max})$ are

$x_- \rightarrow$ the largest floating point number less than or equal to x.

$x_+ \rightarrow$ the smallest floating point number greater than or equal to x.

Therefore for $N_{\min} \leq |x| \leq N_{\max}$,

$$x_- = \begin{cases} (b_0.b_1 \cdots b_{p-1})_\beta \times \beta^E & \text{if } x > 0, \\ -[(b_0.b_1 \cdots b_{p-1})_\beta + \beta^{1-p}] \times \beta^E & \text{if } x < 0, \end{cases}$$

$$x_+ = \begin{cases} [(b_0.b_1 \cdots b_{p-1})_\beta + \beta^{1-p}] \times \beta^E & \text{if } x > 0, \\ -(b_0.b_1 \cdots b_{p-1})_\beta \times \beta^E & \text{if } x < 0 \end{cases}$$

Example: For $x = 99.95$ which is not a floating point number in the system $(10, 3, -1, 2)$ after normalization,

$$x_- = 9.99 \times 10 = 99.9 \text{ and } x_+ = 10^2 = 100$$

whereas for $y = -x$,

$$y_- = -10^2 \text{ and } y_+ = -9.99 \times 10 = -99.9.$$

Finite precision systems

For $|x| > N_{\max}$,

$$x_- = \begin{cases} N_{\max} & \text{if } x > 0, \\ -\infty & \text{if } x < 0, \end{cases} \qquad x_+ = \begin{cases} \infty & \text{if } x > 0, \\ -N_{\max} & \text{if } x < 0, \end{cases}$$

and for $|x| < N_{\min}$,

$$x_- = \begin{cases} 0 & \text{if } x > 0, \\ -N_{\min} & \text{if } x < 0, \end{cases} \qquad x_+ = \begin{cases} N_{\min} & \text{if } x > 0, \\ 0 & \text{if } x < 0, \end{cases}$$

Finite precision systems

For $|x| > N_{\max}$,

$$x_- = \begin{cases} N_{\max} & \text{if } x > 0, \\ -\infty & \text{if } x < 0, \end{cases} \qquad x_+ = \begin{cases} \infty & \text{if } x > 0, \\ -N_{\max} & \text{if } x < 0, \end{cases}$$

and for $|x| < N_{\min}$,

$$x_- = \begin{cases} 0 & \text{if } x > 0, \\ -N_{\min} & \text{if } x < 0, \end{cases} \qquad x_+ = \begin{cases} N_{\min} & \text{if } x > 0, \\ 0 & \text{if } x < 0, \end{cases}$$

Rounding modes:

$fl(x) \rightarrow$ rounded value of x.
Depending on the rule of rounding, $fl(x) = x_+$ or $fl(x) = x_-$.

Rounding in finite precision

Rounding Mode	**Rule**
Round down/Round toward $-\infty$	$fl(x) = x_-$
Round up/Round towards ∞	$fl(x) = x_+$
Round towards zero	$fl(x) = \begin{cases} x_- & \text{if } x \geq 0; \\ x_+ & \text{if } x \leq 0 \end{cases}$
Round to nearest	$fl(x)$ is either x_- or x_+ whichever is nearer to x except that $fl(x) = \infty$ if $x > N_{\max}$ and $fl(x) = -\infty$ if $x < -N_{\max}$. If x is the mid point of (x_-, x_+), then the one among x_- and x_+ with the lesser value of b_{p-1} is chosen.

Rounding in finite precision

Example: In the finite precision system $(10, 3, -1, 2)$,

$$fl(99.95) = \begin{cases} 99.9 & \text{in } \textit{round down} \text{ and } \textit{round towards zero}; \\ 100 & \text{in } \textit{round to nearest}; \end{cases}$$

Rounding in finite precision

Example: In the finite precision system $(10, 3, -1, 2)$,

$$fl(99.95) = \begin{cases} 99.9 & \text{in } \textit{round down} \text{ and } \textit{round towards zero}; \\ 100 & \text{in } \textit{round to nearest}; \end{cases}$$

For any x such that $N_{\min} \leq |x| \leq N_{\max}$,

$$\frac{|fl(x) - x|}{|x|} \leq \frac{\beta^{1-p}}{2} := u$$

in the *round to nearest* rounding mode.

Rounding in finite precision

Example: In the finite precision system $(10, 3, -1, 2)$,

$$fl(99.95) = \begin{cases} 99.9 & \text{in } \textit{round down} \text{ and } \textit{round towards zero}; \\ 100 & \text{in } \textit{round to nearest}; \end{cases}$$

For any x such that $N_{\min} \leq |x| \leq N_{\max}$,

$$\frac{|fl(x) - x|}{|x|} \leq \frac{\beta^{1-p}}{2} := u$$

in the *round to nearest* rounding mode.

$u \rightarrow$ **maximum relative rounding error or unit roundoff in 'round to nearest' rounding mode.**

Rounding in finite precision

Example: In the finite precision system $(10, 3, -1, 2)$,

$$fl(99.95) = \begin{cases} 99.9 & \text{in } \textit{round down} \text{ and } \textit{round towards zero}; \\ 100 & \text{in } \textit{round to nearest}; \end{cases}$$

For any x such that $N_{\min} \leq |x| \leq N_{\max}$,

$$\frac{|fl(x) - x|}{|x|} \leq \frac{\beta^{1-p}}{2} := u$$

in the *round to nearest* rounding mode.

$u \rightarrow$ **maximum relative rounding error or unit roundoff in 'round to nearest' rounding mode.**

In the finite precision system $(10, 3, -1, 2)$, $u = \frac{10^{-2}}{2} = 0.005$.

Rounding in finite precision

Example: In the finite precision system $(10, 3, -1, 2)$,

$$fl(99.95) = \begin{cases} 99.9 & \text{in } \textit{round down} \text{ and } \textit{round towards zero}; \\ 100 & \text{in } \textit{round to nearest}; \end{cases}$$

For any x such that $N_{\min} \leq |x| \leq N_{\max}$,

$$\frac{|fl(x) - x|}{|x|} \leq \frac{\beta^{1-p}}{2} := u$$

in the *round to nearest* rounding mode.

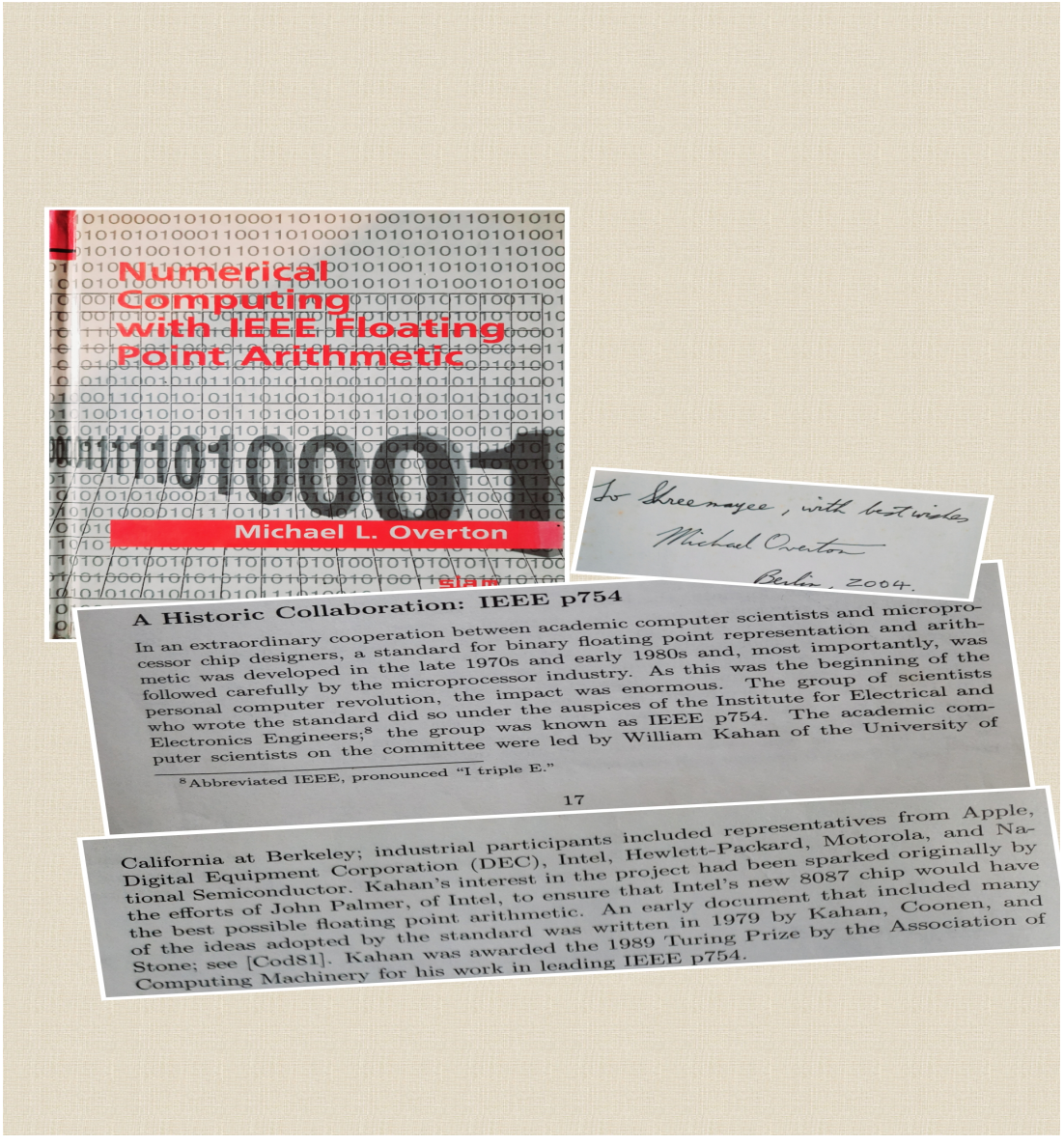
$u \rightarrow$ **maximum relative rounding error or unit roundoff in 'round to nearest' rounding mode.**

In the finite precision system $(10, 3, -1, 2)$, $u = \frac{10^{-2}}{2} = 0.005$.

Therefore for all x with $N_{\min} \leq |x| \leq N_{\max}$,

$$fl(x) = x(1 + \delta) \text{ where } |\delta| \leq u.$$

IEEE finite precision arithmetic



IEEE finite precision arithmetic

- This is a binary ($\beta = 2$) system.
- In single precision there are 32 bits to store a number divided into 1 bit for the sign, 23 bits for s and 8 bits for E.
- In double precision there are 64 bits to store a number divided 1 bit for the sign, 52 bits for s and 11 bits for E.

IEEE finite precision arithmetic

- This is a binary ($\beta = 2$) system.
- In single precision there are 32 bits to store a number divided into 1 bit for the sign, 23 bits for s and 8 bits for E.
- In double precision there are 64 bits to store a number divided 1 bit for the sign, 52 bits for s and 11 bits for E.

As $b_0 = 1$ for all normalized floating point numbers. Therefore a bit is not wasted for this and all 23(52) bits are used to store b_1 to b_{23} (b_1 to b_{52}) in single (double) precision.

IEEE finite precision arithmetic

- This is a binary ($\beta = 2$) system.
- In single precision there are 32 bits to store a number divided into 1 bit for the sign, 23 bits for s and 8 bits for E.
- In double precision there are 64 bits to store a number divided 1 bit for the sign, 52 bits for s and 11 bits for E.

As $b_0 = 1$ for all normalized floating point numbers. Therefore a bit is not wasted for this and all 23(52) bits are used to store b_1 to b_{23} (b_1 to b_{52}) in single (double) precision.

This is called *hidden bit technique* and it effectively increases the precision from (23 to 24) (52 to 53) in single (double) precision.

IEEE finite precision arithmetic

- This is a binary ($\beta = 2$) system.
- In single precision there are 32 bits to store a number divided into 1 bit for the sign, 23 bits for s and 8 bits for E.
- In double precision there are 64 bits to store a number divided 1 bit for the sign, 52 bits for s and 11 bits for E.

As $b_0 = 1$ for all normalized floating point numbers. Therefore a bit is not wasted for this and all 23(52) bits are used to store b_1 to b_{23} (b_1 to b_{52}) in single (double) precision.

This is called *hidden bit technique* and it effectively increases the precision from (23 to 24) (52 to 53) in single (double) precision.

To care of both positive and negative exponents, the exponent string stores the binary representation of $E + 127$ in single and $E + 1023$ in double precision. This is called *biased representation*.

IEEE finite precision arithmetic

- This is a binary ($\beta = 2$) system.
- In single precision there are 32 bits to store a number divided into 1 bit for the sign, 23 bits for s and 8 bits for E.
- In double precision there are 64 bits to store a number divided 1 bit for the sign, 52 bits for s and 11 bits for E.

As $b_0 = 1$ for all normalized floating point numbers. Therefore a bit is not wasted for this and all 23(52) bits are used to store b_1 to b_{23} (b_1 to b_{52}) in single (double) precision.

This is called *hidden bit technique* and it effectively increases the precision from (23 to 24) (52 to 53) in single (double) precision.

To care of both positive and negative exponents, the exponent string stores the binary representation of $E + 127$ in single and $E + 1023$ in double precision. This is called *biased representation*.

For example if the exponent field stores 1000001 the exponent stored is $129 - 127 = 2$ and if the exponent field stores 0000011, then the exponent stored is -124.

IEEE finite precision arithmetic

- This is a binary ($\beta = 2$) system.
- In single precision there are 32 bits to store a number divided into 1 bit for the sign, 23 bits for s and 8 bits for E.
- In double precision there are 64 bits to store a number divided 1 bit for the sign, 52 bits for s and 11 bits for E.

As $b_0 = 1$ for all normalized floating point numbers. Therefore a bit is not wasted for this and all 23(52) bits are used to store b_1 to b_{23} (b_1 to b_{52}) in single (double) precision.

As $b_0 = 1$ for all normalized floating point numbers. Therefore a bit is not wasted for this and all 23(52) bits are used to store b_1 to b_{23} (b_1 to b_{52}) in single (double) precision.

This is called *hidden bit technique* and it effectively increases the precision from (23 to 24) (52 to 53) in single (double) precision.

To care of both positive and negative exponents, the exponent string stores the binary representation of $E + 127$ in single and $E + 1023$ in double precision. This is called *biased representation*.

IEEE finite precision arithmetic

- This is a binary ($\beta = 2$) system.
- In single precision there are 32 bits to store a number divided into 1 bit for the sign, 23 bits for s and 8 bits for E.
- In double precision there are 64 bits to store a number divided 1 bit for the sign, 52 bits for s and 11 bits for E.

As $b_0 = 1$ for all normalized floating point numbers. Therefore a bit is not wasted for this and all 23(52) bits are used to store b_1 to b_{23} (b_1 to b_{52}) in single (double) precision.

This is called *hidden bit technique* and it effectively increases the precision from (23 to 24) (52 to 53) in single (double) precision.

To care of both positive and negative exponents, the exponent string stores the binary representation of $E + 127$ in single and $E + 1023$ in double precision. This is called *biased representation*.

How is zero stored?

IEEE storage tables: Single precision

Table 4.1: IEEE Single Format

$\pm$	$a_1 a_2 a_3 \ldots a_8$	$b_1 b_2 b_3 \ldots b_{23}$

If exponent bitstring $a_1 \ldots a_8$ is	Then numerical value represented is
$(00000000)_2 = (0)_{10}$	$\pm(0.b_1 b_2 b_3 \ldots b_{23})_2 \times 2^{-126}$
$(00000001)_2 = (1)_{10}$	$\pm(1.b_1 b_2 b_3 \ldots b_{23})_2 \times 2^{-126}$
$(00000010)_2 = (2)_{10}$	$\pm(1.b_1 b_2 b_3 \ldots b_{23})_2 \times 2^{-125}$
$(00000011)_2 = (3)_{10}$	$\pm(1.b_1 b_2 b_3 \ldots b_{23})_2 \times 2^{-124}$
$\downarrow$	$\downarrow$
$(01111111)_2 = (127)_{10}$	$\pm(1.b_1 b_2 b_3 \ldots b_{23})_2 \times 2^{0}$
$(10000000)_2 = (128)_{10}$	$\pm(1.b_1 b_2 b_3 \ldots b_{23})_2 \times 2^{1}$
$\downarrow$	$\downarrow$
$(11111100)_2 = (252)_{10}$	$\pm(1.b_1 b_2 b_3 \ldots b_{23})_2 \times 2^{125}$
$(11111101)_2 = (253)_{10}$	$\pm(1.b_1 b_2 b_3 \ldots b_{23})_2 \times 2^{126}$
$(11111110)_2 = (254)_{10}$	$\pm(1.b_1 b_2 b_3 \ldots b_{23})_2 \times 2^{127}$
$(11111111)_2 = (255)_{10}$	$\pm\infty$ if $b_1 = \cdots = b_{23} = 0$, NaN otherwise

Figure : Image courtesy *Numerical Computing with IEEE Floating Point Arithmetic by M. L. Overton*, page 19.

IEEE storage tables: Double precision

Table 4.2: IEEE Double Format

$\pm$	$a_1 a_2 a_3 \ldots a_{11}$	$b_1 b_2 b_3 \ldots b_{52}$

If exponent bitstring is $a_1 \ldots a_{11}$	Then numerical value represented is
$(00000000000)_2 = (0)_{10}$	$\pm(0.b_1 b_2 b_3 \ldots b_{52})_2 \times 2^{-1022}$
$(00000000001)_2 = (1)_{10}$	$\pm(1.b_1 b_2 b_3 \ldots b_{52})_2 \times 2^{-1022}$
$(00000000010)_2 = (2)_{10}$	$\pm(1.b_1 b_2 b_3 \ldots b_{52})_2 \times 2^{-1021}$
$(00000000011)_2 = (3)_{10}$	$\pm(1.b_1 b_2 b_3 \ldots b_{52})_2 \times 2^{-1020}$
$\downarrow$	$\downarrow$
$(01111111111)_2 = (1023)_{10}$	$\pm(1.b_1 b_2 b_3 \ldots b_{52})_2 \times 2^{0}$
$(10000000000)_2 = (1024)_{10}$	$\pm(1.b_1 b_2 b_3 \ldots b_{52})_2 \times 2^{1}$
$\downarrow$	$\downarrow$
$(11111111100)_2 = (2044)_{10}$	$\pm(1.b_1 b_2 b_3 \ldots b_{52})_2 \times 2^{1021}$
$(11111111101)_2 = (2045)_{10}$	$\pm(1.b_1 b_2 b_3 \ldots b_{52})_2 \times 2^{1022}$
$(11111111110)_2 = (2046)_{10}$	$\pm(1.b_1 b_2 b_3 \ldots b_{52})_2 \times 2^{1023}$
$(11111111111)_2 = (2047)_{10}$	$\pm\infty$ if $b_1 = \cdots = b_{52} = 0$, NaN otherwise

Figure : Image courtesy *Numerical Computing with IEEE Floating Point Arithmetic by M. L. Overton*, page 22.

Rounding in IEEE finite precision

- The gap between 1 and the next floating point number denoted by ϵ is called *machine epsilon*.
- The gap between any other pairs of normalized floating point numbers with exponent E is $\epsilon \times \beta^E$.
- The rounding mode is *round to nearest* and $u = \epsilon/2$.
- *Overflow* occurs for $|x| > N_{\max}$ as $fl(x) = \pm\infty$.
- *Underflow* occurs if $x \neq 0$ but $fl(x) = 0$.
- Subnormal numbers ensure very gradual underflow to 0.

IEEE single and double precision

Single precision (32 bit):

$$p = 24;$$
$$\epsilon = 2^{-23} \approx 1.2 \times 10^{-7}; \quad u \approx 0.6 \times 10^{-7};$$
$$E_{\min} = -126; \quad E_{\max} = 127;$$
$$N_{\min} = 2^{-126} \approx 1.2 \times 10^{-38};$$
$$N_{\max} = (2 - 2^{-23})2^{127} \approx 3.4 \times 10^{38}.$$

Double precision (64 bit):

$$p = 53;$$
$$\epsilon = 2^{-52} \approx 2.2 \times 10^{-16}; \quad u \approx 1.1 \times 10^{-16};$$
$$E_{\min} = -1022; \quad E_{\max} = 1023;$$
$$N_{\min} = 2^{-1022} \approx 2.2 \times 10^{-308};$$
$$N_{\max} = (2 - 2^{-52})2^{1023} \approx 1.8 \times 10^{308}.$$

IEEE finite precision system

Exercise: Consider the finite precision system $(2, 3, -1, 1)$. Suppose that the hidden bit technique is in use. Additionally assume that there is also a means of indicating that the hidden bit is zero and the exponent value is -1 so that it is a miniature version of the IEEE systems.

(a) List all the numbers in the system including the subnormal numbers in both binary and decimal forms.

(b) Find ϵ, u, $N_{\min}$ and $N_{\max}$.

(c) Noting that for any three numbers a, b, and c in the system,

$$fl(a+b+c) = fl(fl(a+b)+c) \text{ and } fl(c+b+a) = fl(fl(c+b)+a),$$

find three numbers x, y, z such that $fl(x+y+z) \neq fl(z+x+y)$ in round to nearest rounding mode.