1 Integer

- Interpretation of binary representations

$$B2U_w(\overrightarrow{x}) = \sum_{i=0}^{w-1} x_i 2^i \qquad UMin_w = 0 \qquad UMax_w = 2^w - 1$$

$$B2T_w(\overrightarrow{x}) = -x_{w-1}2^{w-1} + \sum_{i=0}^{w-2} x_i 2^i \quad TMin_w = -2^{w-1} \quad TMax_w = 2^{w-1} - 1$$

- Transformation between unsigned and signed

$$T2U_w(x) = \begin{cases} x & 0 \le x \le TMax_w \\ 2^w + x & TMin_w \le x < 0 \end{cases}$$

$$U2T_w(u) = \begin{cases} u & 0 \le u \le TMax_w \\ u - 2^w & TMax_w < u \le UMax_w \end{cases}$$

- Expansion

$$\overrightarrow{u} = [u_{w-1}, \ldots, u_0] \to \overrightarrow{u}' = [0, \ldots, 0, u_{w-1}, \ldots, u_0] \qquad B2U_w(\overrightarrow{u}) = B2U_{w'}(\overrightarrow{u}')$$
$$\overrightarrow{x} = [x_{w-1}, \ldots, x_0] \to \overrightarrow{x}' = [x_{w-1}, \ldots, x_{w-1}, x_{w-1}, \ldots, x_0] \quad B2T_w(\overrightarrow{x}) = B2T_{w'}(\overrightarrow{x}')$$

- Truncation

$$\overrightarrow{u} = [u_{w-1}, \ldots, u_0] \to \overrightarrow{u}' = [u_{k-1}, \ldots, u_0] \quad B2U_k(\overrightarrow{u}') = B2U_w(\overrightarrow{u}) \mod 2^k$$
$$\overrightarrow{x} = [x_{w-1}, \ldots, x_0] \to \overrightarrow{x}' = [x_{k-1}, \ldots, x_0] \quad B2T_k(\overrightarrow{x}') = U2T_k\left(B2U_w(\overrightarrow{x}) \mod 2^k\right)$$

- Addition

$$x +_w^u y = \begin{cases} x + y & 0 \le x + y < 2^w \\ x + y - 2^w & 2^w \le x + y \le 2^{w+1} - 2 \end{cases} \qquad 0 \le x, y \le 2^w - 1$$

$$x +_w^t y = \begin{cases} x + y + 2^w & -2^w \le x + y < -2^{w-1} \\ x + y & -2^{w-1} \le x + y \le 2^{w-1} - 1 \\ x + y - 2^w & 2^{w-1} \le x + y \le 2^w - 2 \end{cases} \quad -2^{w-1} \le x, y \le 2^{w-1} - 1$$

- Negation

$$-_w^t x = \begin{cases} TMin_w & x = TMin_w \\ -x & TMin_w < x \le TMax_w \end{cases}$$

- Multiplication

$$x *_w^u y = (x \cdot y) \mod 2^w \qquad u \ll k = u *_w^u 2^k$$
$$x *_w^t y = U2T_w((x \cdot y) \mod 2^w) \quad x \ll k = x *_w^t 2^k$$

- Division

$$x \gg k = \lfloor x/2^k \rfloor \qquad (x + (1 \ll k) - 1) \gg k = \lceil x/2^k \rceil \qquad x/y \equiv \begin{cases} \lfloor x/y \rfloor & x \ge 0, y > 0 \\ \lceil x/y \rceil & x < 0, y > 0 \end{cases}$$

2 Floating Point Number

- Definition $V = (-1)^s \times M \times 2^E$. 1-bit s encodes the sign s, k-bit `exp` encodes the exponent E, and n-bit `frac` encodes the significand M. float: 1, 8, 23; double: 1, 11, 52.

  Normalized `exp` is neither 0 or $2^k - 1$ (all 1).
    - $E = e - Bias$, in which $Bias = 2^{k-1} - 1$.
    - $M = 1.f_{n-1} \dots f_1 f_0$.

  Denormalized `exp = 0`
    - $E = 1 - Bias$.
    - $M = 0.f_{n-1} \dots f_1 f_0$.

  Infinity `exp`$=2^k - 1$ (all 1), `frac` $= 0$.

  NaN `exp`$=2^k - 1$ (all 1), `frac` $\neq 0$.

- Use round to even.

- $x +^f y \equiv Round(x+y)$. FP addition lacks associativity but features monotonicity: $x+a \geq x+b$ if $a \geq b$.

- $x *^f y \equiv Round(x \times y)$. FP Multiplication is not associative, and it does not distribute over addition. It features monotonicity:

$$a \geq b, c \geq 0 \Rightarrow a *^f c \geq b *^f c$$
$$a \geq b, c \leq 0 \Rightarrow a *^f c \leq b *^f c$$
$$a \neq NaN \Rightarrow a *^f a \geq 0$$

- Type conversions:
    - `int` to `float`: no overflow. Possible to be rounded.
    - `int/float` to `double`: precise conversion.
    - `double` to `float`: possible to overflow to infinity. Possible to be rounded.
    - `float/double` to `int`: round to 0. Possible to overflow.