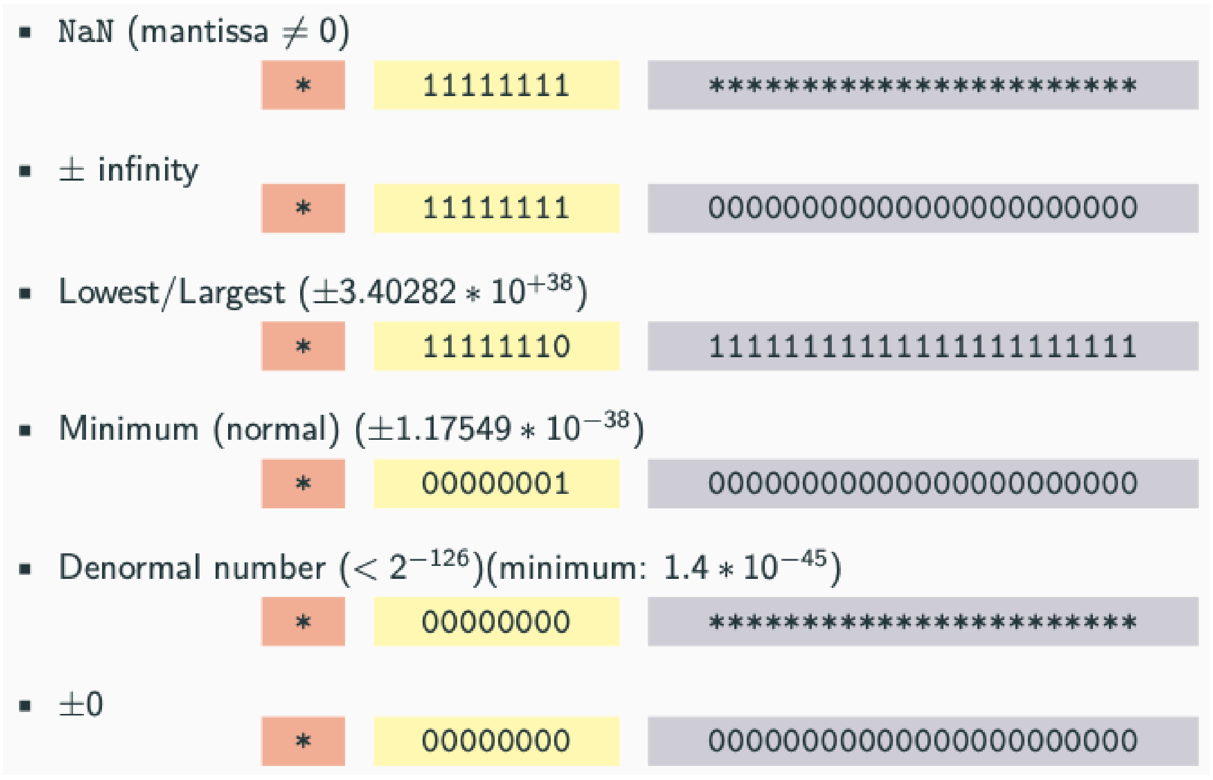
Basic_Concepts_2 - Cheatsheet



- NaN (mantissa $\neq 0$)

*	11111111	***********************

- $\pm$ infinity

*	11111111	00000000000000000000000

- Lowest/Largest ($\pm 3.40282 * 10^{+38}$)

*	11111110	11111111111111111111111

- Minimum (normal) ($\pm 1.17549 * 10^{-38}$)

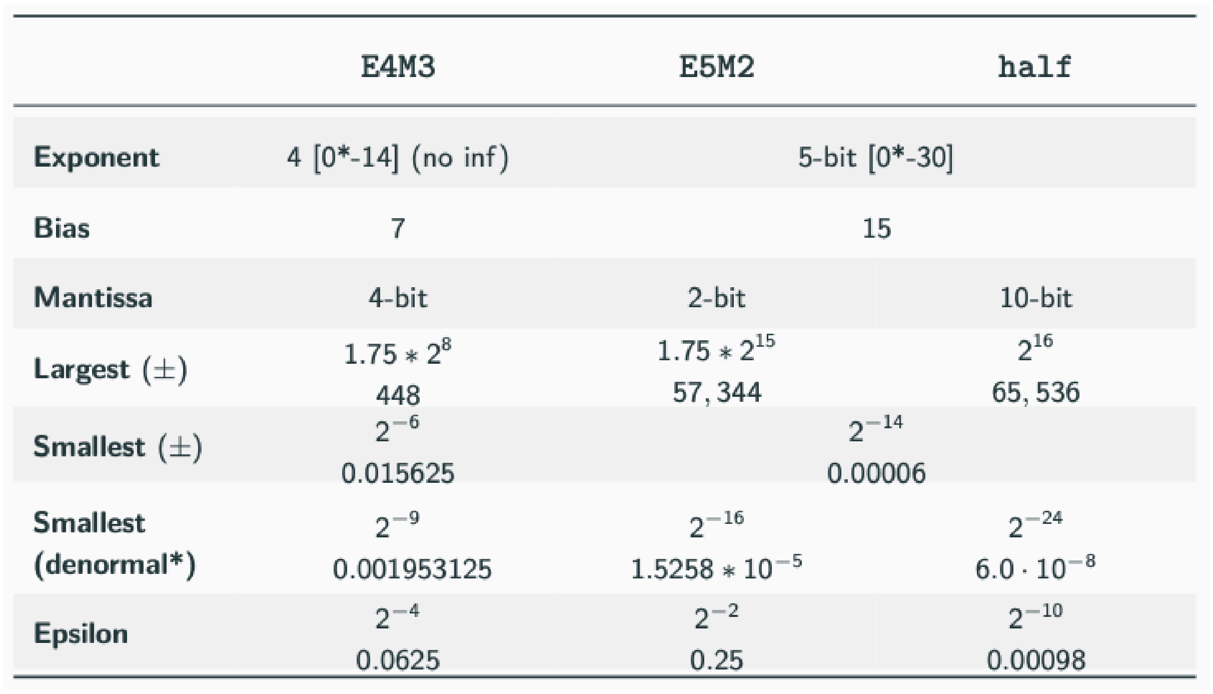
*	00000001	00000000000000000000000

- Denormal number ($< 2^{-126}$)(minimum: $1.4 * 10^{-45}$)

*	00000000	***********************

- ± 0

*	00000000	00000000000000000000000



	E4M3	E5M2	half
Exponent	4 [0*-14] (no inf)	5-bit [0*-30]	
Bias	7	15	
Mantissa	4-bit	2-bit	10-bit
Largest ($\pm$)	$1.75 * 2^{8}$ 448	$1.75 * 2^{15}$ 57, 344	2^{16} 65, 536
Smallest ($\pm$)	2^{-6} 0.015625	2^{-14} 0.00006	
Smallest (denormal*)	2^{-9} 0.001953125	2^{-16} $1.5258 * 10^{-5}$	2^{-24} $6.0 \cdot 10^{-8}$
Epsilon	2^{-4} 0.0625	2^{-2} 0.25	2^{-10} 0.00098

	bfloat16	float	double
Exponent	8-bit [0*-254]		11-bit [0*-2046]
Bias	127		1023
Mantissa	7-bit	23-bit	52-bit
Largest $(\pm)$	2^{128} $3.4 \cdot 10^{38}$		2^{1024} $1.8 \cdot 10^{308}$
Smallest $(\pm)$	2^{-126} $1.2 \cdot 10^{-38}$		2^{-1022} $2.2 \cdot 10^{-308}$
Smallest (denormal*)	/	2^{-149} $1.4 \cdot 10^{-45}$	2^{-1074} $4.9 \cdot 10^{-324}$
Epsilon	2^{-7} 0.0078	2^{-23} $1.2 \cdot 10^{-7}$	2^{-52} $2.2 \cdot 10^{-16}$

```
#include <limits>
// T: float or double

std::numeric_limits<T>::max();       // largest value

std::numeric_limits<T>::lowest();    // lowest value (C++11)

std::numeric_limits<T>::min();       // smallest value

std::numeric_limits<T>::denorm_min() // smallest (denormal) value

std::numeric_limits<T>::epsilon();   // epsilon value

std::numeric_limits<T>::infinity()   // infinity

std::numeric_limits<T>::quiet_NaN()  // NaN
```

```
#include <cmath> // C++11

bool std::isnan(T value)     // check if value is NaN
bool std::isinf(T value)     // check if value is ±infinity
bool std::isfinite(T value)  // check if value is not NaN
                             // and not ±infinity

bool std::isnormal(T value); // check if value is Normal

T    std::ldexp(T x, p)      // exponent shift x ∗ 2^p
int  std::ilogb(T value)     // extracts the exponent of value
```