Positive and Negative Infinity

Positive and negative infinity are special floating-point values that represent the maximum positive and negative real numbers, respectively. They are used to represent results of operations that would otherwise overflow, such as dividing by zero or adding a very large number to a very small number.

NaNs are bit patterns that do not represent any valid real number. They are used to represent results of invalid floating-point operations, such as dividing zero by zero or taking the square root of a negative number.

The following table shows the specific encodings for positive infinity, negative infinity, quiet NaNs, and signaling NaNs in the IEEE floating-point standard:

Type	Sign bit	Exponent	Significand
Positive infinity	0	11111111	00000000
Negative infinity	1	11111111	00000000
Quiet NaN	x	11111111	xxxxxxx...
Signaling NaN	x	11111110	xxxxxxx...

Note that the significand field for NaNs can be any value.

Floating-point units (FPUs) handle positive infinity, negative infinity, and NaNs in special ways.

For example, if the FPU attempts to perform an operation that would result in positive infinity or negative infinity, it will instead return the corresponding infinity value.

If the FPU attempts to perform an operation that would result in a NaN, it will instead return a NaN value and may also generate an exception.

The specific behavior of the FPU when handling positive infinity, negative infinity, and NaNs is defined by the IEEE floating-point standard.

This ensures that all floating-point units behave in a consistent manner, regardless of the manufacturer or architecture.

TABLE 12-6 Specific Single-Precision Encodings.

Value	Sign, Exponent, Significand
Positive zero	0 00000000 00000000000000000000000
Negative zero	1 00000000 00000000000000000000000
Positive infinity	0 11111111 00000000000000000000000
Negative infinity	1 11111111 00000000000000000000000
QNaN	x 11111111 1xxxxxxxxxxxxxxxxxxxxxx
SNaN	x 11111111 0xxxxxxxxxxxxxxxxxxxxxx[a]

[a] SNaN significand field begins with 0, but at least one of the remaining bits must be 1.