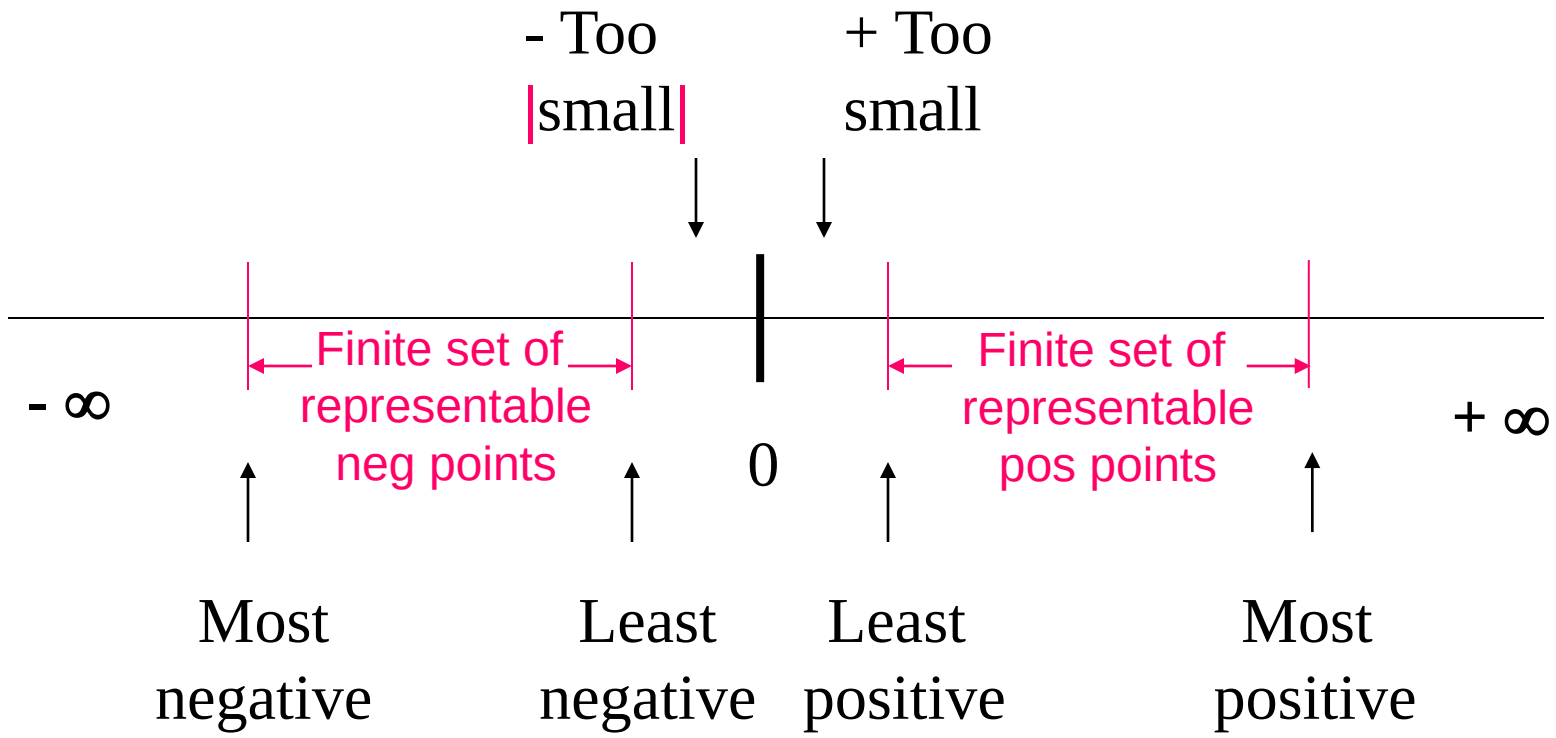
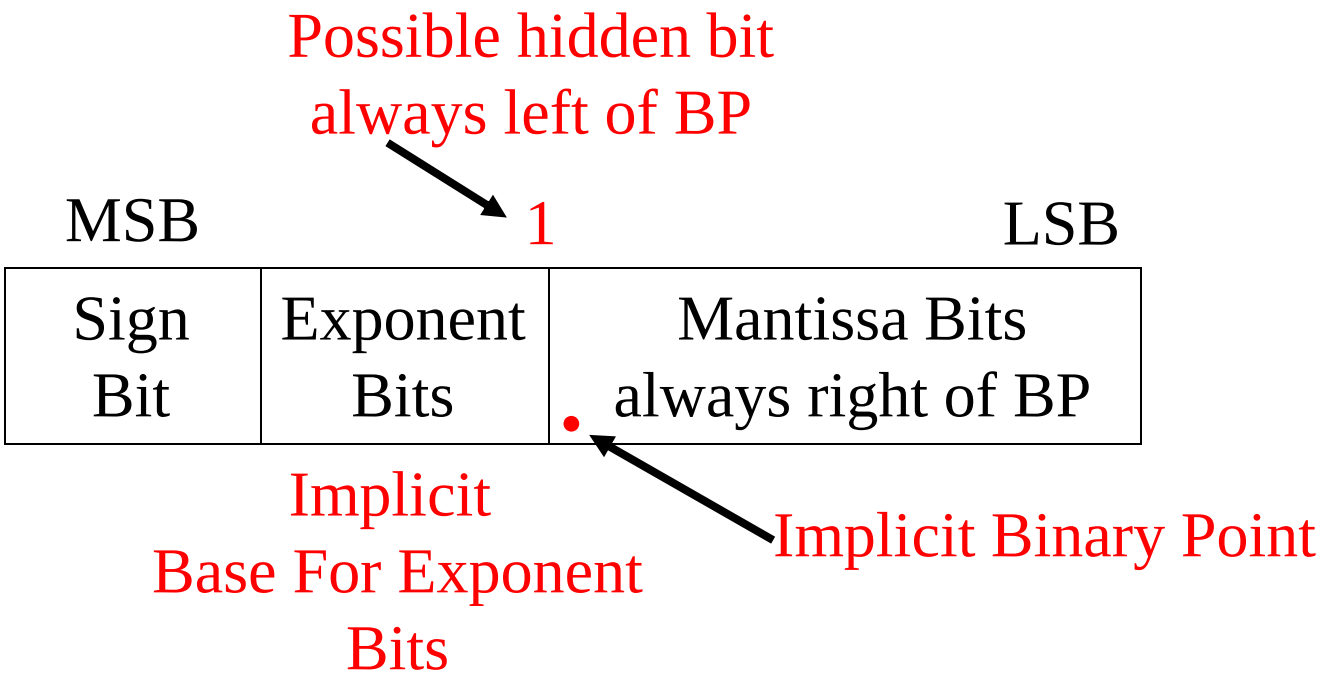
The number of points on the number line that can be represented with exact values is limited by the container size. A FP32 (32 bit float) has 32 bits in its container, so 2^{32} = 4 Gi (4,294,967,296 possible patterns). Most of the patterns are used in the red zones below, but patterns are reserved To indicate + or – infinity, NaN (not a number) and zero (32 bits of all zeroes). Any value in the "Too small" range (between the red zones) has to be represented as zero and any number outside the red zones has to be represented as + or - ∞



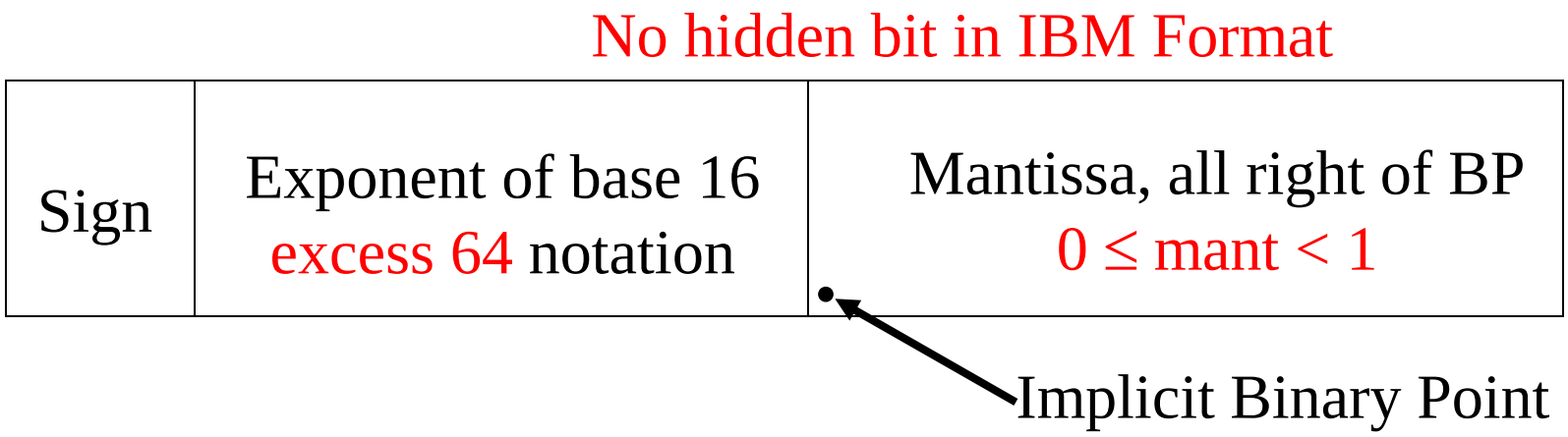
General Floating Format



Maybe 16 so 16 exponent bits * mantissa

Maybe 2 so 2 exponent bits * mantissa

IBM Format



So value is $16^{+1} * .75 = \mathbf{12.0}$

IBM Format

0 100 0101 → 69 0110 … 0000

16^{+5} * .375 = + 393216.0

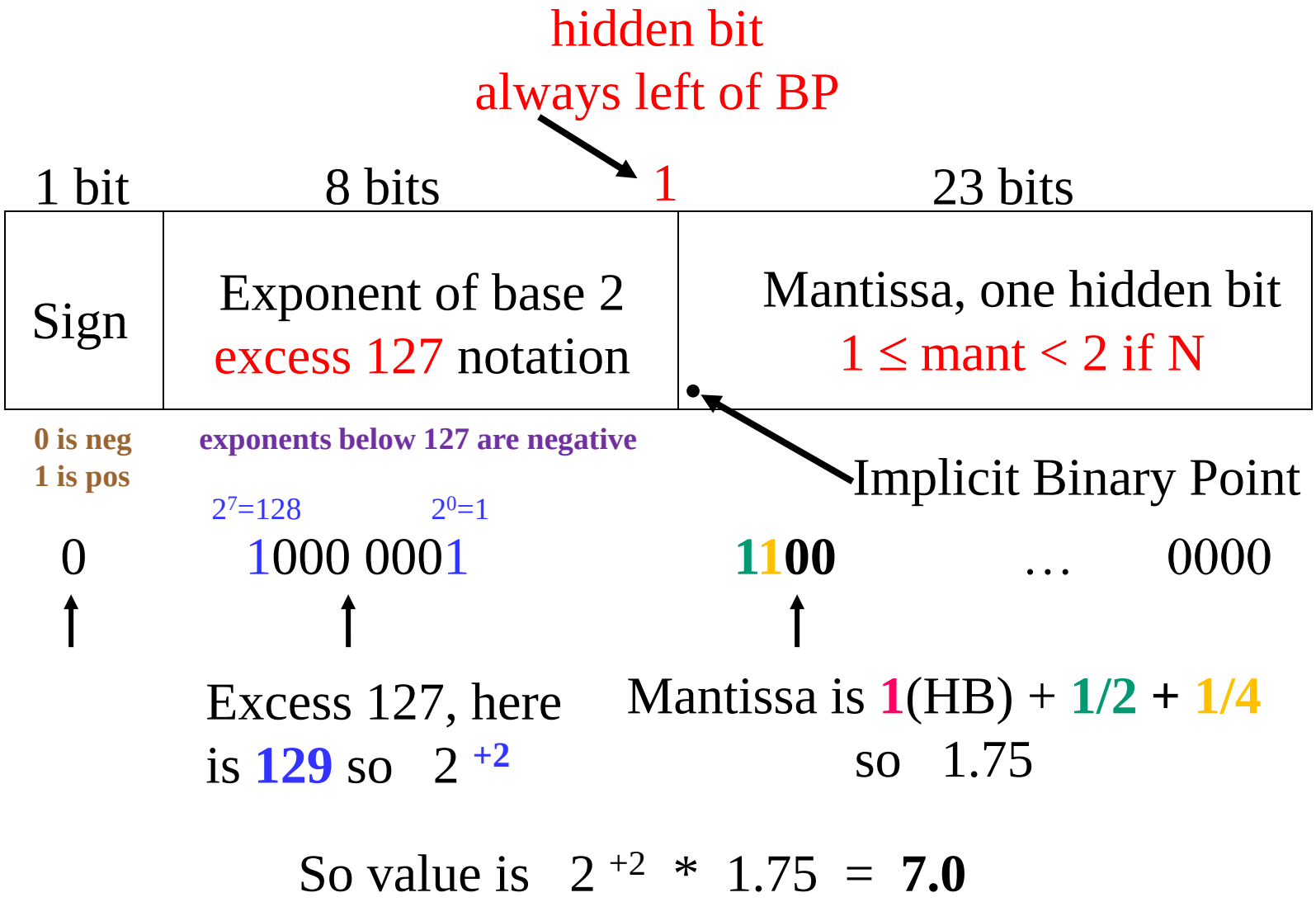
0 011 1111 → 63 1010 … 0000

16^{-1} * .625 = + .0390 ...

1 011 1101 → 61 1100 … 0000

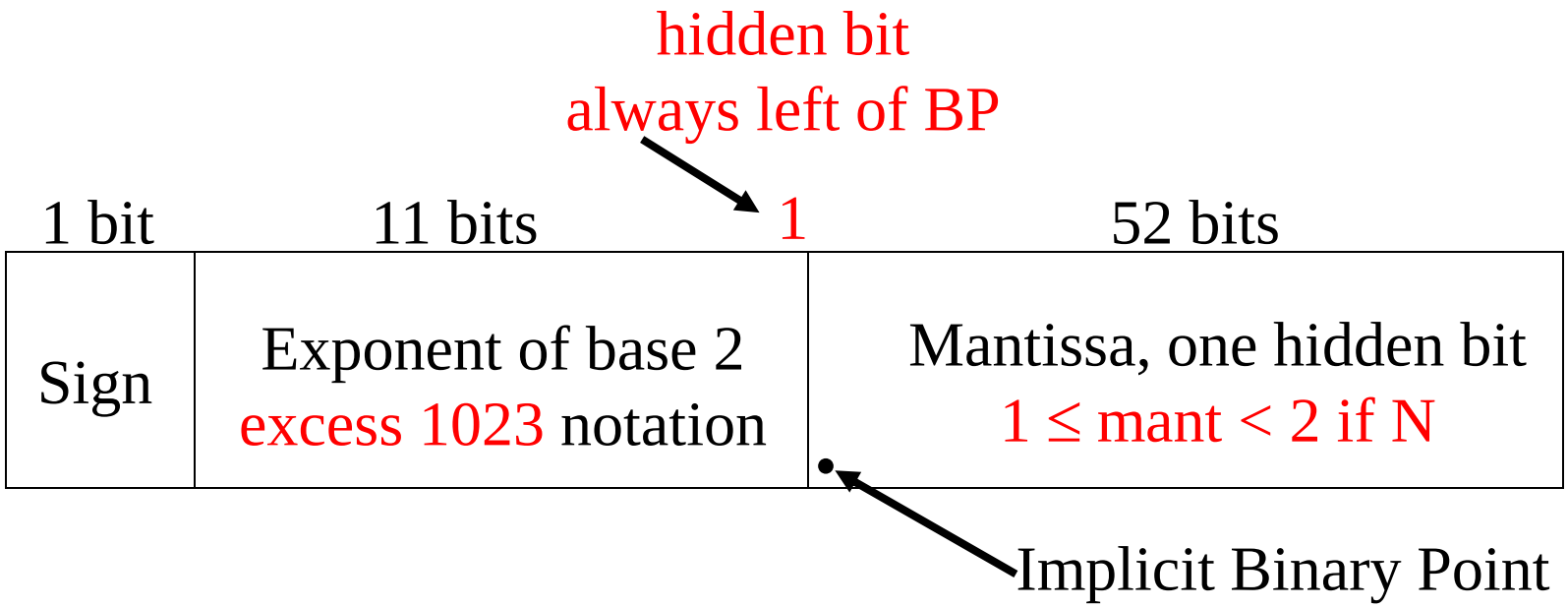
16^{-3} * .75 = - .000183 ...

IEEE 754 Format (FP32)



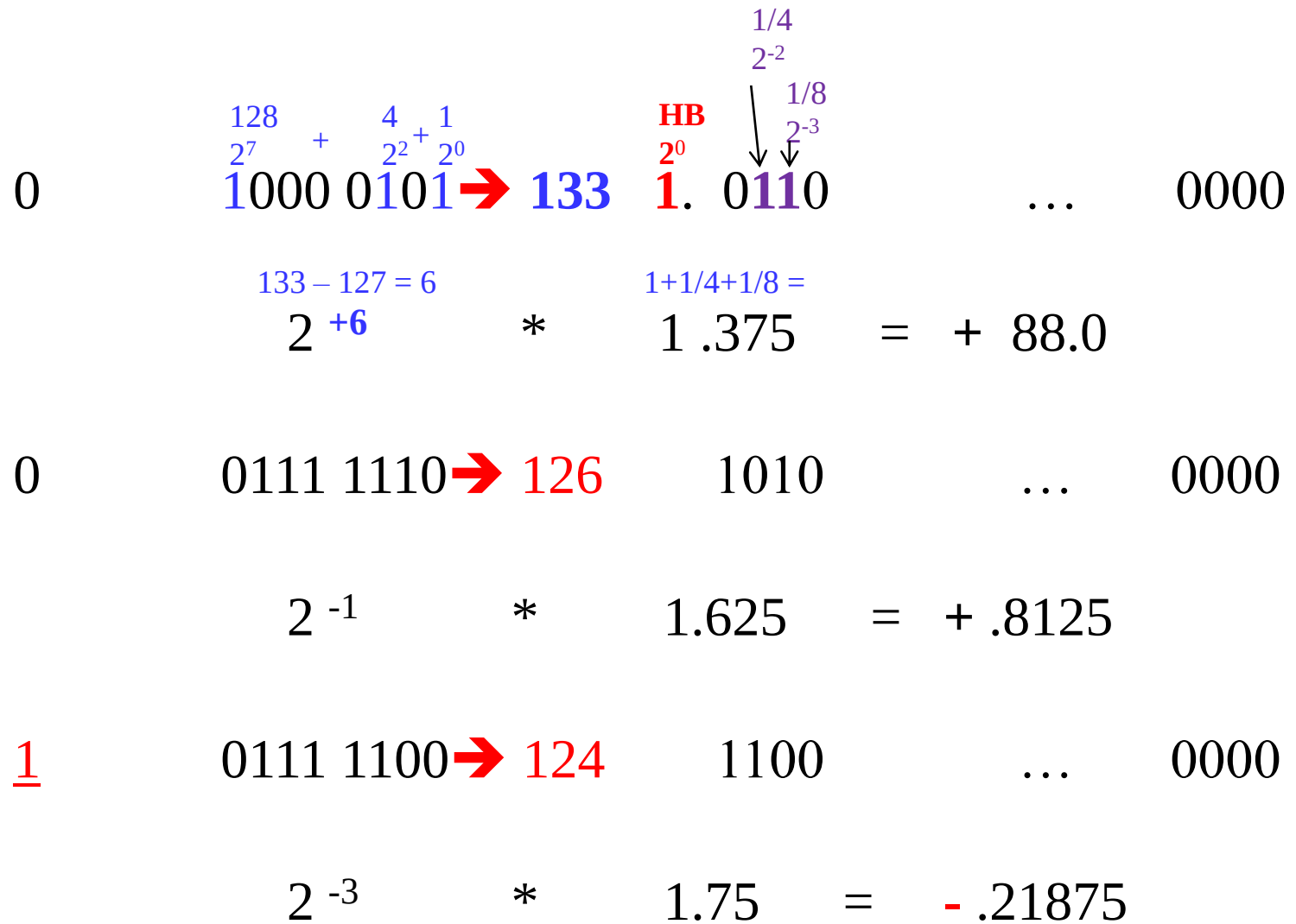
So value is $2^{+2} * 1.75 = \mathbf{7.0}$

IEEE 754 Format, Double Precision



So value is $2^{+2} * 1.75 = \mathbf{7.0}$

IEEE 754 Format



Floating Point Possibilities, IEEE 754

Least positive N	0 0000 0001	0000 … 0000 = 2^{-126}, ≅ 10^{-38}
Most positive	0 1111 1110	1111 … 1111 ≅ 2^{+128} , ≅ 10^{+38}
Least negative N	1 0000 0001	0000 … 0000 = -2^{-126}, ≅ -10^{-38}
Most negative	1 1111 1110	1111 … 1111 ≅ -2^{+128} , -10^{+38}
Least pos/neg DN	$^1/_0$ 0000 0000	0000 … 0001 = $^+/-\ 2^{-149}$, ≅ 10^{-45}
Zero	0 0000 0000	0000 … 0000
Pos/neg infinity	$^1/_0$ 1111 1111	0000 … 0000 = $^+/-$ ∞
NAN	$^1/_0$ 1111 1111	Any non-zero pattern = NAN

Floating Point Possibilities, IEEE 754 (cont'd)

0 1000 0101 .0110 … 0000

2^{+6} * 1.375 = + 88.0

0 0000 0001 .0000 … 0000

2^{-126} * 1.0 ≅ $+ 1.175 * 10^{-38}$

0 0000 0000 .0000 … 0000

= + 0.0

1 1111 1111 .0110 … 0000

= **NAN**

1 1111 1111 .0000 … 0000

= - ∞

Un-normalized small

0 0000 0000 .00001 … 0000

2^{-126} * $2^{-5} = + 2^{-131} \cong + 3.674 * 10^{-40}$

```
please enter a floating point number and new-line: 234765.579335

  mantissa:     0x654365        or:              110 0101 0100 0011 0110 0101
  exponent:     0x90            or:    1001 0000
      sign:     0x0             or: 0
in base 10:     234765.6        or: 0 1001 0000 110 0101 0100 0011 0110 0101

please enter a floating point number and new-line: 234765.579

  mantissa:     0x654365        or:              110 0101 0100 0011 0110 0101
  exponent:     0x90            or:    1001 0000
      sign:     0x0             or: 0
in base 10:     234765.6        or: 0 1001 0000 110 0101 0100 0011 0110 0101

please enter a floating point number and new-line: 234765.571

the floating value for INPUT_NUMBER is broken out as:
  mantissa:     0x654365        or:              110 0101 0100 0011 0110 0101
  exponent:     0x90            or:    1001 0000
      sign:     0x0             or: 0
in base 10:     234765.6        or: 0 1001 0000 110 0101 0100 0011 0110 0101

please enter a floating point number and new-line: 234765.57

  mantissa:     0x654364        or:              110 0101 0100 0011 0110 0100
  exponent:     0x90            or:    1001 0000
      sign:     0x0             or: 0
in base 10:     234765.6        or: 0 1001 0000 110 0101 0100 0011 0110 0100
```

```
-bash-4.1$ ./subtract

please enter a floating point number and new-line: 234765.579335
please enter a floating point number and new-line: 234765.571
the difference between the two numbers entered is:  0.000000


-bash-4.1$ ./subtract

please enter a floating point number and new-line: 234765.579335
please enter a floating point number and new-line: 234765.57
the difference between the two numbers entered is:  0.015625
```

Adding 754 FP Numbers

			1 HB			
0	1000 0101		.0110	…	0000	
	2^{+6}	*	1.375	=	+ 88.0	
+			1 HB			
0	0111 1110		.1010	…	0000	
	2^{-1}	*	1.625	=	+ .8125	

		1 HB			
0	0111 1110	.1010	…	0000	

Shift mantissa of smaller number right 7 places, note hidden bit

	0	1000 0101	.0000 0011	0100	… 0000	0 0 0 0 0
+			1 HB			
	0	1000 0101	.0110		… 0000	
			1 HB			
	0	1000 0101	.0110 0011	0100	… 0000	

Result is normalized as is: **1 + 1/4 + 1/8 + 1/128 + 1/256 + 1/1024**

2^{+6} * 1 .3876953 = + 88.8125

Adding 754 FP Numbers (cont'd)

1 HB

0 1000 0101 .1110 … 0000

2^{+6} * 1 .875 = + 120.0

\+

1 HB

0 1000 0010 .1011 … 0000

2^{+3} * 1.6875 = + 13.5

1 HB

0 1000 0010 .1011 … 0000

Shift mantissa of smaller number right 3 places, note hidden bit

0 1000 0101 .0011 0110 … 0000 0 0 0

\+

1 HB

0 1000 0101 .1110 … 0000

---

10 HB + Carry Out

0 1000 0101 .0001 0110 … 0000

Normalize result

1 HB

0 1000 0110 .0000 10110 … 0000 0

2^{+7} * 1 .04296875 = + 133.5

Adding 754 FP Numbers (cont'd)

			1 HB		
0	1000 0101		.1110	…	0000
	2^{+6}	*	1 .875	=	+ 120.0
+			1 HB		
0	1000 0101		.1011	…	0000
	2^{+6}	*	1.6875	=	+ 108.0

		1 HB		
0	1000 0101	.1011	…	0000
+		1 HB		
0	1000 0101	.1110	…	0000
		11 HB + Carry Out		
0	1000 0101	.1001	…	0000

Normalize result →

			1 HB		
0	1000 0110		.1100 1000	…	0000 0
	2^{+7} ↑	*	1 .78125	=	+ 228