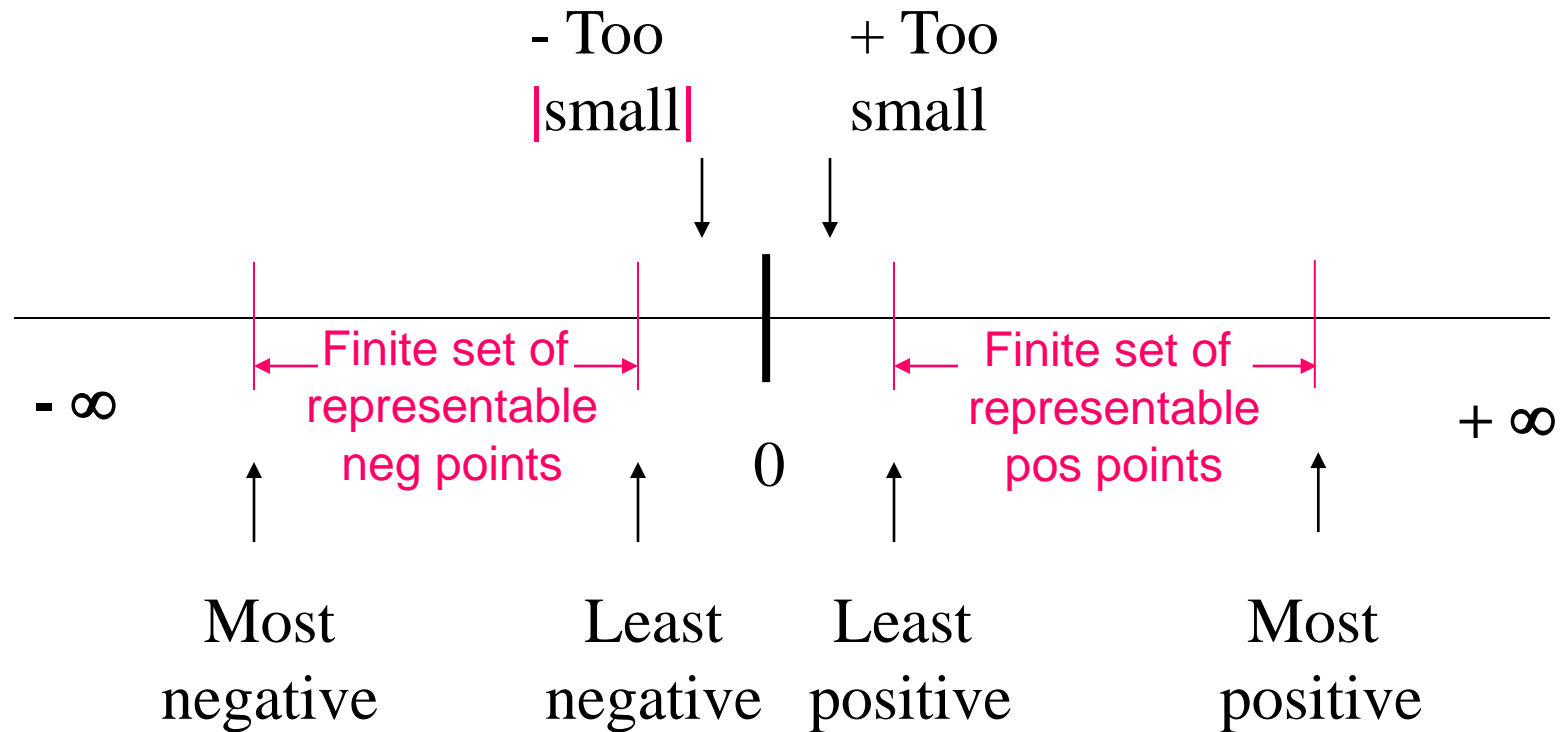
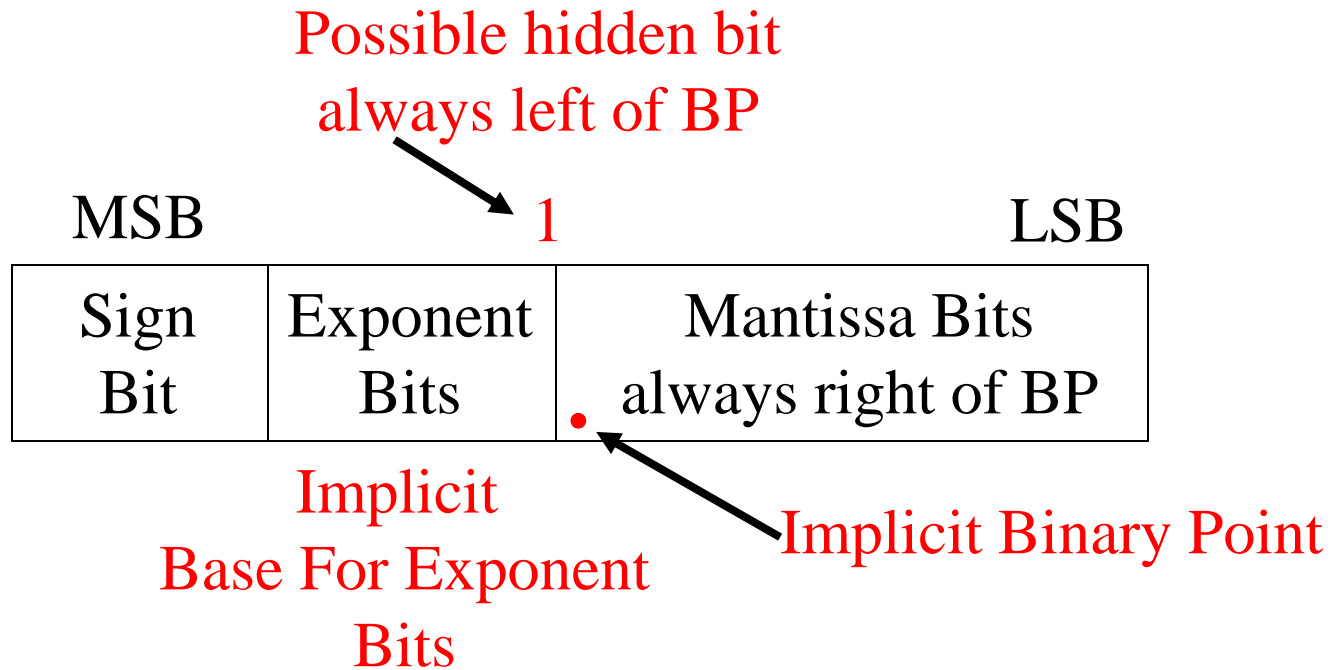


The number of points on the number line that can be represented with exact values is limited by the container size. A FP32 (32 bit float) has 32 bits in its container, so $2^{32} = 4 \text{ Gi}$ (4,294,967,296 possible patterns). Most of the patterns are used in the red zones below, but patterns are 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

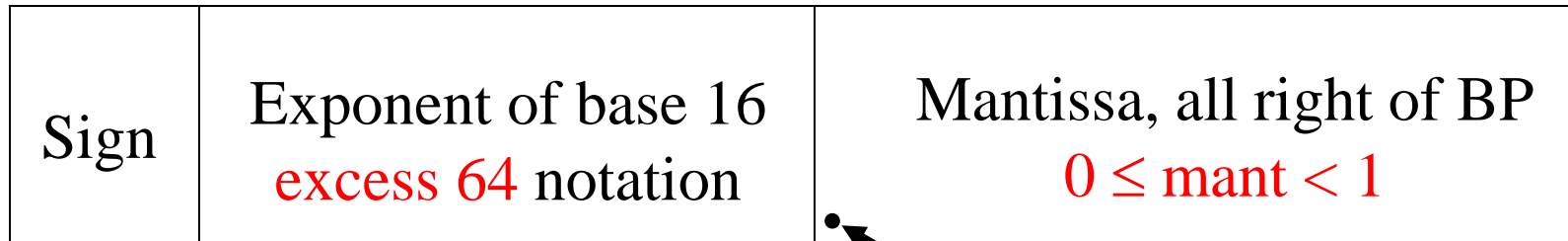


Maybe 16 so 16 exponent bits * mantissa

Maybe 2 so 2 exponent bits * mantissa

IBM Format

No hidden bit in IBM Format



0	100 0001	1100	...	0000
↑	↑	↑		
Positive sign bit	Excess 64, here is 65 so 16^{+1}	Mantissa has $1/2 + 1/4$ so $.75$		

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

IBM Format

0 100 0101 → 69 0110 ... 0000

$$16^{+5} \quad * \quad .375 \quad = \quad + \quad 393216.0$$

0 011 1111 → 63 1010 ... 0000

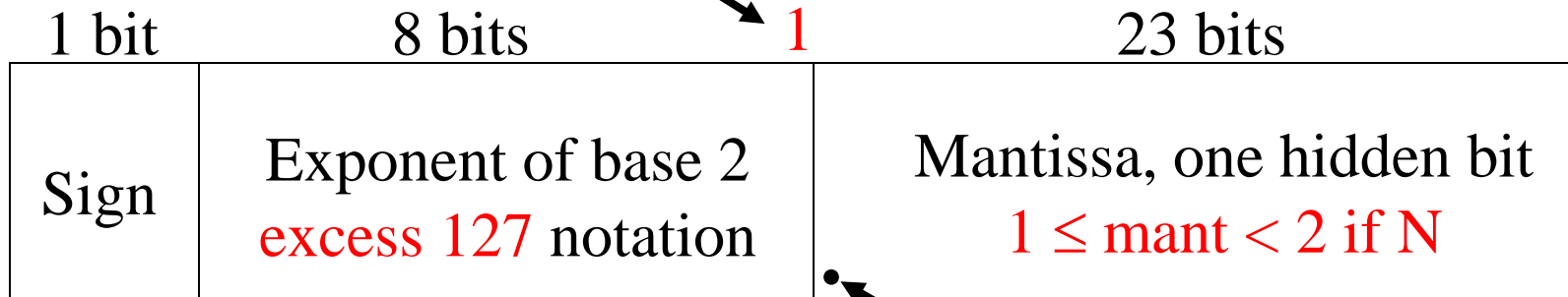
$$16^{-1} \quad * \quad .625 \quad = \quad + \quad .0390 \dots$$

1 011 1101 → 61 1100 ... 0000

$$16^{-3} \quad * \quad .75 \quad = \quad - \quad .000183 \dots$$

IEEE 754 Format (FP32)

hidden bit
always left of BP



0 is neg
1 is pos

exponents below 127 are negative

$2^7=128$ $2^0=1$

0

1000 0001

1100

... 0000

↑

↑

↑

Excess 127, here
is 129 so 2^{+2}

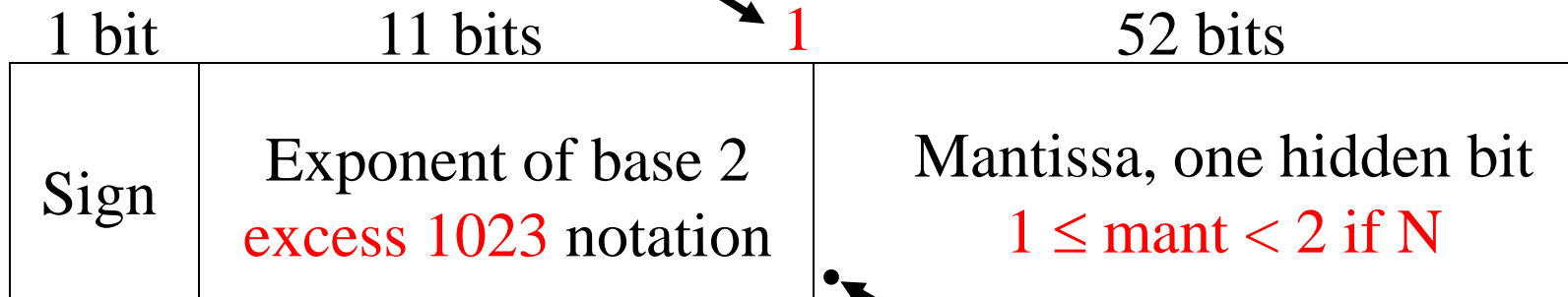
Mantissa is 1(HB) + 1/2 + 1/4
so 1.75

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

Implicit Binary Point

IEEE 754 Format, Double Precision

hidden bit
always left of BP



Implicit Binary Point

0 1000 0000 001 1100 ... 0000
 ↑ ↑ ↑

Excess 1023 here Mantissa is 1(HB) + 1/2 + 1/4
 is 1025 so 2^{+2} so 1.75

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

IEEE 754 Format

$$0 \quad \overset{128}{2^7} + \overset{4}{2^2} \overset{1}{2^0} \rightarrow 133 \quad \text{HB} \quad \overset{1/4}{2^{-2}} \downarrow \overset{1/8}{2^{-3}} \downarrow \quad 1.0110 \quad \dots \quad 0000$$

$$133 - 127 = 6 \quad * \quad 1 + 1/4 + 1/8 = 1.375 \quad = \quad + 88.0$$

$$0 \quad 0111 \ 1110 \rightarrow 126 \quad 1010 \quad \dots \quad 0000$$

$$2^{-1} \quad * \quad 1.625 \quad = \quad + .8125$$

$$\underline{1} \quad 0111 \ 1100 \rightarrow 124 \quad 1100 \quad \dots \quad 0000$$

$$2^{-3} \quad * \quad 1.75 \quad = \quad - .21875$$

Floating Point Possibilities, IEEE 754

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

Floating Point Possibilities, IEEE 754 (cont'd)

$$\begin{array}{rcl}
 0 & 1000\ 0101 & .0110 \\
 & 2^{+6} & * \quad 1.375 \\
 & & = \dots\ 0000 \\
 & & = +\ 88.0
 \end{array}$$

$$\begin{array}{rcl}
 0 & 0000\ 0001 & .0000 \\
 & 2^{-126} & * \quad 1.0 \\
 & & \cong \dots\ 0000 \\
 & & = +\ 1.175 * 10^{-38}
 \end{array}$$

$$\begin{array}{rcl}
 0 & 0000\ 0000 & .0000 \\
 & & = \dots\ 0000 \\
 & & = +\ 0.0
 \end{array}$$

$$\begin{array}{rcl}
 1 & 1111\ 1111 & .0110 \\
 & & = \dots\ 0000 \\
 & & = \text{NaN}
 \end{array}$$

$$\begin{array}{rcl}
 1 & 1111\ 1111 & .0000 \\
 & & = \dots\ 0000 \\
 & & = -\ \infty
 \end{array}$$

Un-normalized small

$$\begin{array}{rcl}
 0 & 0000\ 0000 & .00001 \\
 & 2^{-126} & * \quad 2^{-5} = +\ 2^{-131} \cong +\ 3.674 * 10^{-40} \\
 & & \dots\ 0000
 \end{array}$$

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 <u>0100</u>
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

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

0	0111 1110	2^{-1}	*	^{1 HB} .1010	=	... 0000
---	-----------	----------	---	-----------------------	---	----------

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

0	1000 0101	2^{+6}	*	.0000 00 11 0100 ... 0000	=	... 0000
+	0	1000 0101	*	^{1 HB} .0110 ... 0000	=	... 0000
0	1000 0101	2^{+6}	*	^{1 HB} .0110 00 11 0100 ... 0000	=	... 0000

Result is normalized as is:

$$1 + 1/4 + 1/8 + 1/128 + 1/256 + 1/1024$$

$$2^{+6} * 1.3876953 = + 88.8125$$

0000000

Adding 754 FP Numbers (cont'd)

0	1000 0101		^{1 HB} .1110	...	0000
	2^{+6}	*	1.875	=	+ 120.0
0	1000 0010		^{1 HB} .1011	...	0000
	2^{+3}	*	1.6875	=	+ 13.5

0	1000 0010		^{1 HB} .1011	...	0000
---	-----------	--	-----------------------	-----	------

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



+	0	1000 0101		.0011 0110	...	0000	000
	0	1000 0101		^{1 HB} .1110	...	0000	

0	1000 0101		^{10 HB + Carry Out} .0001 0110	...	0000
---	-----------	--	---	-----	------

Normalize result



0	1000 0110		^{1 HB} .0000 10110	...	0000	0
---	-----------	--	-----------------------------	-----	------	---

	2^{+7}	↑	*	1.04296875	=	+ 133.5
--	----------	---	---	------------	---	---------

Adding 754 FP Numbers (cont'd)

0	1000 0101	2^{+6}	*	^{1 HB} .1110 1.875	=	+ 120.0	...	0000	
+	0	1000 0101	2^{+6}	*	^{1 HB} .1011 1.6875	=	+ 108.0	...	0000

+	0	1000 0101	2^{+6}	*	^{1 HB} .1011 1.6875	=	+ 108.0	...	0000
	0	1000 0101	2^{+6}	*	^{1 HB} .1110 1.875	=	+ 120.0	...	0000

0	1000 0101	2^{+6}	*	^{11 HB + Carry Out} .1001 1.5625	=	+ 96.0	...	0000
---	-----------	----------	---	---	---	--------	-----	------

Normalize result

0	1000 0110	2^{+7}	*	^{1 HB} .1100 1000 1.78125	=	+ 228	...	0000
---	-----------	----------	---	--	---	-------	-----	------