Converter to 32 Bit Single Precision IEEE 754 Binary Floating Point Standard System: Converting Base 10 Decimal Numbers

Convert to 32 bit single precision IEEE 754 binary floating point standard

Decimal number: allowed: + - . digits

Convert to 32 bit single precision IEEE 754 binary floating point standard

convert

A number in 32 bit single precision IEEE 754 binary floating point standard representation requires three building elements: sign (it takes 1 bit and it's either 0 for positive or 1 for negative numbers), exponent (8 bits) and mantissa (23 bits)

Latest decimal numbers converted from base ten to 32 bit single precision IEEE 754 floating point binary standard representation

-0.857 to 32 bit single precision IEEE 754 binary floating point = ?	*Nov 26 21:02 UTC (GMT)*
0.333 333 333 333 333 333 333 333 32 to 32 bit single precision IEEE 754 binary floating point = ?	*Nov 26 21:02 UTC (GMT)*
-0.848 to 32 bit single precision IEEE 754 binary floating point = ?	*Nov 26 21:02 UTC (GMT)*
0.888 888 888 888 887 to 32 bit single precision IEEE 754 binary floating point = ?	*Nov 26 21:02 UTC (GMT)*
-55.611 111 111 4 to 32 bit single precision IEEE 754 binary floating point = ?	*Nov 26 21:02 UTC (GMT)*
-0.842 to 32 bit single precision IEEE 754 binary floating point = ?	*Nov 26 21:02 UTC (GMT)*
33 554 433 to 32 bit single precision IEEE 754 binary floating point = ?	*Nov 26 21:02 UTC (GMT)*
11 000 000 111 000 000 000 000 000 000 002 to 32 bit single precision IEEE 754 binary floating point = ?	*Nov 26 21:02 UTC (GMT)*
-0.833 6 to 32 bit single precision IEEE 754 binary floating point = ?	*Nov 26 21:02 UTC (GMT)*
76.375 to 32 bit single precision IEEE 754 binary floating point = ?	*Nov 26 21:02 UTC (GMT)*
12 342.14 to 32 bit single precision IEEE 754 binary floating point = ?	*Nov 26 21:02 UTC (GMT)*
-2 018.125 to 32 bit single precision IEEE 754 binary floating point = ?	*Nov 26 21:02 UTC (GMT)*
-0.833 4 to 32 bit single precision IEEE 754 binary floating point = ?	*Nov 26 21:01 UTC (GMT)*

All base ten decimal numbers converted to 32 bit single precision IEEE 754 binary floating point

How to convert decimal numbers from base ten to 32 bit single precision IEEE 754 binary floating point standard

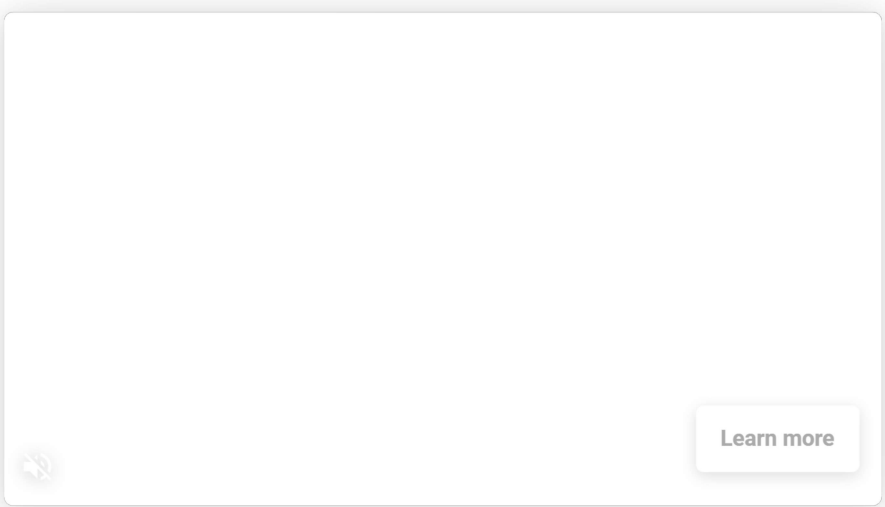
Follow the steps below to convert a base 10 decimal number to 32 bit single precision IEEE 754 binary floating point:

1. If the number to be converted is negative, start with its the positive version.

2. First convert the integer part. Divide repeatedly by 2 the base ten positive representation of the integer number that is to be converted to binary, until we get a quotient that is equal to zero, keeping track of each remainder.

3. Construct the base 2 representation of the positive integer part of the number, by taking all the remainders of the previous dividing operations, starting from the bottom of the list constructed above. Thus, the last remainder of the divisions becomes the first symbol (the leftmost) of the base two number, while the first remainder becomes the last symbol (the rightmost).

4. Then convert the fractional part. Multiply the number repeatedly by 2, until we get a fractional part that is equal to zero, keeping track of each integer part of the results.



5. Construct the base 2 representation of the fractional part of the number by taking all the integer parts of the previous multiplying operations, starting from the top of the constructed list above (they should appear in the binary representation, from left to right, in the order they have been calculated).

6. Normalize the binary representation of the number, by shifting the decimal point (or if you prefer, the decimal mark) "n" positions either to the left or to the right, so that only one non zero digit remains to the left of the decimal point.

7. Adjust the exponent in 8 bit excess/bias notation and then convert it from decimal (base 10) to 8 bit binary, by using the same technique of repeatedly dividing by 2, as shown above:
Exponent (adjusted) = Exponent (unadjusted) + $2^{(8-1)}$ - 1

8. Normalize mantissa, remove the leading (leftmost) bit, since it's allways '1' (and the decimal sign if the case) and adjust its length to 23 bits, either by removing the excess bits from the right (losing precision...) or by adding extra '0' bits to the right.

9. Sign (it takes 1 bit) is either 1 for a negative or 0 for a positive number.

Example: convert the negative number -25.347 from decimal system (base ten) to 32 bit single precision IEEE 754 binary floating point:

1. Start with the positive version of the number:

|-25.347| = 25.347

2. First convert the integer part, 25. Divide it repeatedly by 2, keeping track of each remainder, until we get a quotient that is equal to zero:

division = quotient + **remainder**;

25 ÷ 2 = 12 + **1**;

12 ÷ 2 = 6 + **0**;

6 ÷ 2 = 3 + **0**;

3 ÷ 2 = 1 + **1**;

1 ÷ 2 = 0 + **1**;

We have encountered a quotient that is ZERO => FULL STOP

---

3. Construct the base 2 representation of the integer part of the number by taking all the remainders of the previous dividing operations, starting from the bottom of the list constructed above:

$25_{(10)} = 1\ 1001_{(2)}$

4. Then convert the fractional part, 0.347. Multiply repeatedly by 2, keeping track of each integer part of the results, until we get a fractional part that is equal to zero:

#) multiplying = **integer** + fractional part;

1) 0.347 × 2 = **0** + 0.694;

2) 0.694 × 2 = **1** + 0.388;

3) 0.388 × 2 = **0** + 0.776;

4) 0.776 × 2 = **1** + 0.552;

5) 0.552 × 2 = **1** + 0.104;

6) 0.104 × 2 = **0** + 0.208;

7) 0.208 × 2 = **0** + 0.416;

8) 0.416 × 2 = **0** + 0.832;

9) 0.832 × 2 = **1** + 0.664;

10) 0.664 × 2 = **1** + 0.328;

11) 0.328 × 2 = **0** + 0.656;

12) 0.656 × 2 = **1** + 0.312;

13) 0.312 × 2 = **0** + 0.624;

14) 0.624 × 2 = **1** + 0.248;

15) 0.248 × 2 = **0** + 0.496;

16) 0.496 × 2 = **0** + 0.992;

17) 0.992 × 2 = **1** + 0.984;

18) 0.984 × 2 = **1** + 0.968;

19) 0.968 × 2 = **1** + 0.936;

20) 0.936 × 2 = **1** + 0.872;

21) 0.872 × 2 = **1** + 0.744;

22) 0.744 × 2 = **1** + 0.488;

23) 0.488 × 2 = **0** + 0.976;

24) 0.976 × 2 = **1** + 0.952;

We didn't get any fractional part that was equal to zero. But we had enough iterations (over Mantissa limit = 23) and at least one integer part that was different from zero => FULL STOP (losing precision...).

5. Construct the base 2 representation of the fractional part of the number, by taking all the integer parts of the previous multiplying operations, starting from the top of the constructed list above:

$0.347_{(10)} = 0.0101\ 1000\ 1101\ 0100\ 1111\ 1101_{(2)}$

6. Summarizing - the positive number before normalization:

$25.347_{(10)} = 1\ 1001.0101\ 1000\ 1101\ 0100\ 1111\ 1101_{(2)}$

7. Normalize the binary representation of the number, shifting the decimal point 4 positions to the left so that only one non-zero digit stays to the left of the decimal point:

$25.347_{(10)} =$
$1\ 1001.0101\ 1000\ 1101\ 0100\ 1111\ 1101_{(2)} =$
$1\ 1001.0101\ 1000\ 1101\ 0100\ 1111\ 1101_{(2)} \times 2^0 =$
$1.1001\ 0101\ 1000\ 1101\ 0100\ 1111\ 1101_{(2)} \times 2^4$

8. Up to this moment, there are the following elements that would feed into the 32 bit single precision IEEE 754 binary floating point:

Sign: 1 (a negative number)

Exponent (unadjusted): 4

Mantissa (not-normalized): 1.1001 0101 1000 1101 0100 1111 1101

9. Adjust the exponent in 8 bit excess/bias notation and then convert it from decimal (base 10) to 8 bit binary (base 2), by using the same technique of repeatedly dividing it by 2, as already demonstrated above:

Exponent (adjusted) = Exponent (unadjusted) + $2^{(8-1)}$ - 1 = $(4 + 127)_{(10)} = 131_{(10)} =$
$1000\ 0011_{(2)}$

10. Normalize the mantissa, remove the leading (leftmost) bit, since it's allways '1' (and the decimal point) and adjust its length to 23 bits, by removing the excess bits from the right (losing precision...):

Mantissa (not-normalized): 1.1001 0101 1000 1101 0100 1111 1101

Mantissa (normalized): 100 1010 1100 0110 1010 0111

Conclusion:

Sign (1 bit) = 1 (a negative number)

Exponent (8 bits) = 1000 0011

Mantissa (23 bits) = 100 1010 1100 0110 1010 0111

Number -25.347, converted from the decimal system (base 10) to 32 bit single precision IEEE 754 binary floating point =

1 - 1000 0011 - 100 1010 1100 0110 1010 0111