FLOATING POINT NUMBER FROM DECIMAL TO BINARY

32-bit floating point number (jts6mq): 35.375 (base 10)

- Convert 35.375 to rational form: 283/8
    - The sign bit is 0
- Exponent:
    - Divide by 2^5, or 32-> 283/256
    - Exponent=132 (5+127)
    - 132 in binary = 1000 0100
- Mantissa:
    - 283/256 – 256/256 = 27/256
    - Subtracting increasing powers of ½
        - Cannot subtract 128/256 from 27/256 (1/2)
        - Cannot subtract 64/256 from 27/256 (1/4)
        - Cannot subtract 32/256 from 27/256 (1/8)
        - *Can* subtract 16/256 from 27/256 (1/16) = 11/256
        - *Can* subtract 8/256 from 11/256 (1/32) = 3/256
        - Cannot subtract 5/256 from 3/256 (1/64)
        - *Can* subtract 2/256 from 3/256 (1/128) = 1/256
        - *Can* subtract 1/256 from 1/256 (1/256) = 0
    - The parts of the mantissa are 1/16 + 1/32 + 1/128 + 1/256
    - Encoded into binary, this is equivalent to:
        - 0001 1011 0000 0000 0000 000

Altogether, this is equivalent to (in big-endian):
0100 0010 0000 1101 1000 0000 0000 0000

In little-endian, this is equivalent to:
0000 0000 1000 0000 0000 1101 0100 0010

The hex representation of this number is therefore:
0x00800d42

HEX TO FLOATING POINT

HEX NUMBER (little-endian): 0x00401ec3

- First, convert the little-endian format into big-endian
    - 0x00401ec3->0xc31e4000
- Next, convert each digit into a 4-bit chunk
    - 1100 0011 0001 1110 0100 0000 0000 0000
- The sign digit is 1, so this is a negative number
- The exponent portion is: 1000 0110
    - This is equal to 134 in binary
    - 134-127=7
    - The exponent is 7

- The mantissa portion is: 0011 1100 1000 0000 0000 000
    - Therefore, the set bits are:
    - 1/8, 1/16, 1/32, 1/64, 1/512
    - That's equal to .125 + .0625 + .03125 + .015625 + .00195312
    - = .23632812 + 1 = 1.23632812
    - multiplied by 128 (2^7) = 158.249999

RESULT: -158.249999