**64-bit IEEE 754 double-precision floating-point format**

To represent 0.10.1 in the **64-bit IEEE 754 double-precision floating-point format**, we go through the same steps as in 32-bit, but this time with a larger mantissa (52 bits instead of 23) and a different exponent bias.

### **Step-by-Step Conversion of** 0.10.1 **to 64-bit Floating Point**

1. **Convert 0.10.1 to Binary**  
   0.10.1 in binary is:

0.110=0.00011001100110011001100110011001100110011001100110011…20.1\_{10} = 0.00011001100110011001100110011001100110011001100110011\ldots\_2

It’s a repeating binary fraction: 0.000110011‾20.\overline{000110011}\_2.

1. **Normalize the Binary**  
   Normalize 0.10.1 to the form 1.M×2E1.M \times 2^E:

0.1=1.1001100110011001100110011001100110011001100110011010…2×2−40.1 = 1.1001100110011001100110011001100110011001100110011010\ldots\_2 \times 2^{-4}

* + The normalized mantissa is 1.1001100110011001100110011001100110011001100110011010…1.1001100110011001100110011001100110011001100110011010\ldots.
  + The exponent is −4-4.

1. **Compute the Biased Exponent**  
   In IEEE 754 double precision, the exponent bias is 10231023:

Ebiased=E+bias=−4+1023=1019E\_{\text{biased}} = E + \text{bias} = -4 + 1023 = 1019

* + In binary: Ebiased=0111111011E\_{\text{biased}} = 0111111011.

1. **Truncate the Mantissa**  
   The mantissa stores the fractional part of 1.M1.M after the leading 11 (implicit). Double precision allows for 52 bits:

M=1001100110011001100110011001100110011001100110011010M = 1001100110011001100110011001100110011001100110011010

1. **Combine Components**  
   Now, assemble the final 64-bit representation:
   * **Sign bit** (SS): 00 (positive number)
   * **Exponent** (EbiasedE\_{\text{biased}}): 01111110110111111011
   * **Mantissa**: 10011001100110011001100110011001100110011001100110101001100110011001100110011001100110011001100110011010

Final representation:

0 0111111011 10011001100110011001100110011001100110011001100110100 \, 0111111011 \, 1001100110011001100110011001100110011001100110011010

### **Reconstructing the Value**

Using the IEEE 754 formula:

N=(−1)S×1.M×2E−biasN = (-1)^S \times 1.M \times 2^{E - \text{bias}}

Substitute the values:

* S=0S = 0
* M=1.1001100110011001100110011001100110011001100110011010M = 1.1001100110011001100110011001100110011001100110011010
* E−bias=−4E - \text{bias} = -4

Reconstruct:

1. M≈1.600000000000000088817841970012523233890533447265625M \approx 1.600000000000000088817841970012523233890533447265625
2. N=M×2−4≈0.1000000000000000055511151231257827021181583404541015625N = M \times 2^{-4} \approx 0.1000000000000000055511151231257827021181583404541015625

### **Approximation Stored**

The value stored in 64-bit floating-point representation for 0.10.1 is:

0.10000000000000000555111512312578270211815834045410156250.1000000000000000055511151231257827021181583404541015625

### **Final 64-bit Representation**

Binary: 0 0111111011 1001100110011001100110011001100110011001100110011010\text{Binary: } 0 \, 0111111011 \, 1001100110011001100110011001100110011001100110011010 Hexadecimal: 3FB999999999999A\text{Hexadecimal: } 3FB999999999999A

To represent 0.30.3 in the **64-bit IEEE 754 double-precision floating-point format**, follow these steps:

### **Step-by-Step Conversion of** 0.30.3 **to 64-bit Floating Point**

#### 1. **Convert** 0.30.3 **to Binary**

The decimal 0.30.3 in binary is:

0.310=0.010011001100110011001100110011001100110011001100110011…20.3\_{10} = 0.010011001100110011001100110011001100110011001100110011\ldots\_2

It is a repeating binary fraction: 0.0100110011‾20.\overline{0100110011}\_2.

#### 2. **Normalize the Binary**

Normalize 0.30.3 to the form 1.M×2E1.M \times 2^E:

0.3=1.0011001100110011001100110011001100110011001100110011…2×2−20.3 = 1.0011001100110011001100110011001100110011001100110011\ldots\_2 \times 2^{-2}

* The normalized mantissa is 1.0011001100110011001100110011001100110011001100110011…1.0011001100110011001100110011001100110011001100110011\ldots.
* The exponent is −2-2.

#### 3. **Compute the Biased Exponent**

In IEEE 754 double precision, the exponent bias is 10231023:

Ebiased=E+bias=−2+1023=1021E\_{\text{biased}} = E + \text{bias} = -2 + 1023 = 1021

* In binary: Ebiased=0111111101E\_{\text{biased}} = 0111111101.

#### 4. **Truncate the Mantissa**

The mantissa stores the fractional part of 1.M1.M after the leading 11 (implicit). Double precision allows for 52 bits:

M=0011001100110011001100110011001100110011001100110011M = 0011001100110011001100110011001100110011001100110011

#### 5. **Combine Components**

Now, assemble the final 64-bit representation:

* **Sign bit** (SS): 00 (positive number)
* **Exponent** (EbiasedE\_{\text{biased}}): 01111111010111111101
* **Mantissa**: 00110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011

The final 64-bit representation is:

0 0111111101 00110011001100110011001100110011001100110011001100110 \, 0111111101 \, 0011001100110011001100110011001100110011001100110011

### **Reconstructing the Value**

Using the IEEE 754 formula:

N=(−1)S×1.M×2E−biasN = (-1)^S \times 1.M \times 2^{E - \text{bias}}

Substitute the values:

* S=0S = 0
* M=1.0011001100110011001100110011001100110011001100110011M = 1.0011001100110011001100110011001100110011001100110011
* E−bias=−2E - \text{bias} = -2

Reconstruct:

1. M≈1.1875M \approx 1.1875
2. N=M×2−2=1.1875×0.25=0.296875N = M \times 2^{-2} = 1.1875 \times 0.25 = 0.296875

### **Approximation Stored**

The value stored in 64-bit floating-point representation for 0.30.3 is:

0.2999999999999999888977697537484345957636833190917968750.299999999999999988897769753748434595763683319091796875

### **Final 64-bit Representation**

Binary: 0 0111111101 0011001100110011001100110011001100110011001100110011\text{Binary: } 0 \, 0111111101 \, 0011001100110011001100110011001100110011001100110011 Hexadecimal: 3FD3333333333333\text{Hexadecimal: } 3FD3333333333333