

First Represent Unsigned Decimal Real Number 75.8515625 In Binary, Then Re-write The Binary Result More

First Represent Unsigned Decimal Real Number 75.8515625 In Binary, Then Re-write The Binary Result More

Understanding how to convert decimal real numbers into binary form is fundamental in computer science, digital electronics, and programming. Converting an unsigned decimal real number like 75.8515625 into binary involves a systematic approach, breaking down the number into its integer and fractional parts, converting each separately, and then combining them. This process not only helps in understanding number systems but also enhances skills in data representation and precision management in computing systems. In this article, we will explore step-by-step how to convert 75.8515625 into its binary equivalent, rewrite the binary form, and discuss various methods and tips for such conversions.

Understanding the Number: 75.8515625

Before diving into the conversion process, it's essential to understand the structure of the decimal number:

- Integer Part: 75
- Fractional Part: 0.8515625

Our goal is to convert both parts into binary and then combine them to get the full binary representation.

Converting the Integer Part to Binary

The integer part, 75, is straightforward to convert using successive division by 2.

Step-by-Step Conversion of 75 to Binary

1. Divide 75 by 2:
 - Quotient: 37
 - Remainder: 1
2. Divide 37 by 2:

- Quotient: 18
- Remainder: 1
- 3. Divide 18 by 2:
 - Quotient: 9
 - Remainder: 0
- 4. Divide 9 by 2:
 - Quotient: 4
 - Remainder: 1
- 5. Divide 4 by 2:
 - Quotient: 2
 - Remainder: 0
- 6. Divide 2 by 2:
 - Quotient: 1
 - Remainder: 0
- 7. Divide 1 by 2:
 - Quotient: 0
 - Remainder: 1

Now, reading the remainders from bottom to top gives the binary representation:

75 in binary = 1001011

Converting the Fractional Part to Binary

The fractional part, 0.8515625, is converted by successive multiplication by 2. Each step records the integer part (0 or 1) as the binary digit.

Step-by-Step Conversion of 0.8515625

1. Multiply 0.8515625 by 2:
 - Result: 1.703125
 - Binary digit: 1
2. Multiply 0.703125 by 2:
 - Result: 1.40625
 - Binary digit: 1
3. Multiply 0.40625 by 2:
 - Result: 0.8125
 - Binary digit: 0
4. Multiply 0.8125 by 2:
 - Result: 1.625
 - Binary digit: 1
5. Multiply 0.625 by 2:
 - Result: 1.25
 - Binary digit: 1
6. Multiply 0.25 by 2:

- Result: 0.5
 - Binary digit: 0
7. Multiply 0.5 by 2:
- Result: 1.0
 - Binary digit: 1

Since the fractional part reaches 0.0 after the last step, the conversion terminates.

Fractional part in binary = 1101101

Combining Integer and Fractional Parts

The complete binary representation of 75.8515625 is obtained by combining the binary form of the integer part and the fractional part, separated by a binary point.

Binary representation:

75.8515625 = 1001011.1101101

Re-writing the Binary Result in Different Forms

Once the binary representation is obtained, there are multiple ways to re-express or manipulate it depending on the context, such as normalization, scientific notation, or fixed/ floating-point formats.

Normalizing the Binary Number

Normalization involves expressing the binary number in the form:

$1.xxxxxx \times 2^n$

For our number:

1. Shift the binary point to just after the first '1':

1001011.1101101 becomes

$1.0010111101101 \times 2^6$

2. Explanation:

- The original binary point is moved 6 positions to the left.
- The exponent is adjusted accordingly (6 in this case).

Normalized form:

$$1.0010111101101 \times 2^6$$

Expressing in Scientific Notation

Similar to normalization, scientific notation in binary follows the form:

$$1.xxxxx \times 2^n$$

So, our number in scientific notation:

$$1.0010111101101 \times 2^6$$

This form is especially useful in floating-point representations in computers.

Converting Binary to Hexadecimal

For easier readability and compact representation, binary numbers are often converted to hexadecimal.

Step-by-step conversion:

1. Group the binary digits into groups of four, starting from the binary point:

- Integer part: 1001011 → pad with leading zeros if necessary: 0100 1011

- Fractional part: 1101101 → pad with trailing zeros: 1101 1010

2. Convert each group:

Binary	Hexadecimal
0100	4
1011	B
1101	D
1010	A

3. Final hexadecimal representation:

Integer part: 4B

Fractional part: D A

Putting it together:

75.8515625 in hexadecimal $\approx 0x4B.DA$

(Note: The fractional part can be extended further for more precision if needed.)

Understanding the Significance of Binary Representation

Binary representation of decimal numbers is fundamental in digital systems, where data is processed and stored in bits. Accurate conversion and understanding of binary forms enable:

- Efficient data storage
- Precise numerical computations
- Implementation of floating-point arithmetic
- Error detection and correction

Applications of Binary Number Conversion

Converting decimal real numbers to binary is crucial across various domains:

- Computer Programming: For data input/output, floating-point calculations, and debugging.
- Digital Electronics: In designing circuits that process binary signals.
- Data Transmission: Ensuring accurate data encoding and decoding.
- Scientific Computing: High-precision calculations often require binary representations.
- Embedded Systems: Where memory and processing power are limited, efficient data representation is essential.

Tips and Best Practices for Binary Conversion

- Always process the integer and fractional parts separately.
- Use successive division for integers and successive multiplication for fractions.
- Be mindful of rounding errors in fractional conversions; some fractions may have repeating binary representations.
- When converting to hexadecimal, group binary digits into four-bit chunks to simplify.

Conclusion

Converting an unsigned decimal real number like 75.8515625 into binary involves a clear, stepwise process that enhances understanding of number systems. The integer part is converted through successive division, while the fractional part is handled via successive multiplication. Combining these parts yields the full binary representation: 1001011.1101101. Furthermore, re-writing this binary number in normalized form, scientific notation, or hexadecimal makes it versatile for various applications, especially in computing and digital electronics. Mastering these conversions is vital for professionals working with low-level data processing, computer architecture, and digital design, ensuring precision and efficiency in digital computations.

Meta Description:

Learn how to convert the decimal number 75.8515625 into binary step-by-step, re-write the binary result in normalized, scientific, and hexadecimal forms, and understand its practical applications in computing systems.

Frequently Asked Questions

How do you convert the decimal number 75.8515625 into binary?

To convert 75.8515625 to binary, first convert the integer part (75) to binary, then convert the fractional part (0.8515625) to binary separately, and combine both results.

What is the binary representation of the decimal number 75.8515625?

The binary representation of 75.8515625 is 1001011.1101101101.

How is the fractional part 0.8515625 converted into binary?

The fractional part is converted by multiplying by 2 repeatedly and noting the integer parts: $0.8515625 \times 2 = 1.703125$ (1), then $0.703125 \times 2 = 1.40625$ (1), and so on, resulting in 1101101101.

Can you re-write the binary representation of 75.8515625 in normalized form?

Yes, in normalized binary form, 75.8515625 is written as $1.0010111101101101 \times 2^6$.

Why is rewriting the binary number in normalized form

useful?

Normalized form simplifies the representation for floating-point calculations, making it easier to perform arithmetic and maintain precision.

What are common applications of converting decimal real numbers to binary?

This conversion is fundamental in computer systems for storing, processing, and transmitting real numbers accurately in binary format.

Are there any tools or software that can automate this conversion process?

Yes, many scientific calculators, programming languages like Python, and software like MATLAB can automatically convert decimal real numbers to binary.

Additional Resources

First Represent Unsigned Decimal Real Number 75.8515625 In Binary, Then Re-write The Binary Result More: An In-Depth Exploration

Understanding how to convert decimal real numbers into binary form is fundamental in computer science and digital electronics. This process not only enhances comprehension of how computers interpret data but also serves as a cornerstone for more advanced topics like floating-point representations, digital signal processing, and computer architecture. In this article, we will meticulously analyze the conversion of the decimal number 75.8515625 into its binary equivalent, then explore methods to re-express and manipulate this binary result for various computational purposes.

Introduction to Binary Representation of Decimal Numbers

The binary number system, or base-2 system, uses only two digits: 0 and 1. Computers operate natively in binary because their hardware relies on two states, such as on/off or high/low voltage. Converting decimal numbers, especially real numbers, into binary involves two steps:

1. Converting the integer part into binary.
2. Converting the fractional part into binary.

For the decimal number 75.8515625, both parts need to be accurately translated into binary to ensure precise representation.

Step-by-Step Conversion of 75.8515625 to Binary

1. Converting the Integer Part (75) to Binary

The integer part, 75, is converted via successive division by 2, recording the remainders:

Division Step	Quotient	Remainder	Binary Digit (from bottom to top)
75 ÷ 2	37	1	1
37 ÷ 2	18	1	1
18 ÷ 2	9	0	0
9 ÷ 2	4	1	1
4 ÷ 2	2	0	0
2 ÷ 2	1	0	0
1 ÷ 2	0	1	1

Reading remainders from bottom to top, the binary integer part is: 1001011

Thus,
75 (decimal) = 1001011 (binary)

2. Converting the Fractional Part (0.8515625) to Binary

The fractional part is converted by successive multiplication by 2:

Multiplied Value	Integer Part	Fractional Part	Binary Digit
0.8515625 × 2	1.703125	0.703125	1
0.703125 × 2	1.40625	0.40625	1
0.40625 × 2	0.8125	0.8125	0
0.8125 × 2	1.625	0.625	1
0.625 × 2	1.25	0.25	1
0.25 × 2	0.5	0.5	0
0.5 × 2	1.0	0.0	1

The process stops because the fractional part reaches zero. Collecting the binary digits from the leftmost multiplication:

0.8515625 (decimal) = 0.11011101 (binary)

Complete Binary Representation of 75.8515625

Combining the integer and fractional parts:

75.8515625 (decimal) = 1001011.11011101 (binary)

This binary number precisely represents the decimal value in base-2 notation.

Re-writing the Binary Result for Different Purposes

Once the binary conversion is complete, there are multiple ways to re-express, optimize, or interpret the binary number depending on the context—be it for storage, computation, or visualization.

1. Fixed-Point Representation

In fixed-point format, the position of the binary point is fixed, and the number of bits allocated for integer and fractional parts is predetermined. For example, using an 8.8 fixed-point format (16 bits total), the number could be stored as:

- Integer part: 8 bits (e.g., 000001001)
- Fractional part: 8 bits (e.g., 11011101)

The full 16-bit fixed-point binary would be:

000001001.11011101

This allows for consistent arithmetic operations, especially in embedded systems, but requires careful handling of overflow and scaling.

Pros:

- Simple hardware implementation
- Fast arithmetic operations

Cons:

- Limited range and precision
- Fixed scaling factor

2. Floating-Point Representation (IEEE 754 Standard)

For more precise and wide-ranging representations, floating-point formats like IEEE 754 are used. The binary number is normalized, and the exponent and mantissa are stored separately.

Steps for representing 75.8515625 in IEEE 754 single-precision (32-bit):

- Normalize the binary number:

$$1001011.11011101 = 1.00101111011101 \times 2^6$$

- Exponent: $6 + \text{bias (127 for single-precision)} = 133$ (10000101 in binary)

- Mantissa: the fractional part after normalization (excluding the leading 1):

00101111011101 (pad with zeros to 23 bits)

The 32-bit IEEE 754 representation:

Sign	Exponent (8 bits)	Mantissa (23 bits)
0	10000101	00101111011101000000000

Benefits:

- Supports a vast range of values
- Suitable for scientific calculations

Drawbacks:

- Slight precision loss due to finite bits
- More complex hardware implementation

3. Re-Expressing in Hexadecimal

Binary numbers can also be grouped into 4-bit segments and converted into hexadecimal for more compact notation, useful in debugging and low-level programming.

Binary: 1001011.11011101

Groupings:

- Integer part: 0100 1011
- Fractional part: 1101 1101

Corresponding hex:

- 0100 1011 = 0x4B

- 1101 1101 = 0xDD

Therefore, the hexadecimal representation:

0x4B.DD

This form simplifies reading and copying binary data, especially in assembly language and hardware design.

Features and Considerations in Binary Representation

When representing decimal real numbers in binary, several features and considerations come into play:

- Precision: The number of bits allocated impacts how accurately the number can be represented.
- Range: The choice between fixed-point and floating-point determines the maximum and minimum values that can be stored.
- Efficiency: Binary formats optimized for specific applications can save memory or processing time.
- Compatibility: Standard formats like IEEE 754 ensure interoperability across systems.

Applications and Practical Implications

Understanding the binary representation of decimal numbers like 75.8515625 is crucial in various domains:

- Computer Programming: Accurate data storage and manipulation.
- Digital Signal Processing: Precise filter design and data analysis.
- Embedded Systems: Efficient fixed-point arithmetic for resource-constrained environments.
- Graphics and Multimedia: Color encoding and image processing often involve binary representations.

Conclusion

Converting the decimal number 75.8515625 into binary and re-expressing it in various formats is a foundational skill that bridges the gap between human-readable decimal notation and machine-level binary data. Whether viewed as a fixed-point binary string, a floating-point number, or a hexadecimal value, each representation serves specific purposes—balancing precision, range, and computational efficiency. Mastery of these conversion techniques enhances understanding of digital

systems, improves programming accuracy, and supports the development of efficient hardware and software solutions.

Summary of Key Points:

- The integer part 75 converts to 1001011 in binary.
- The fractional part 0.8515625 converts to 0.11011101 in binary.
- Complete binary representation: 1001011.11011101.
- Alternative representations include fixed-point, floating-point, and hexadecimal.
- Choosing the appropriate format depends on application requirements related to precision, range, and performance.

By comprehending these conversions and representations, technologists, students, and engineers can deepen their understanding of how numbers are handled within digital systems, leading to more efficient and accurate computational solutions.

Note: This detailed exploration emphasizes clarity and depth, ensuring that readers can follow each step of conversion and understand the implications of different binary representations.

First Represent Unsigned Decimal Real Number 75 8515625 In Binary Then Re Write The Binary Result More

First Represent Unsigned Decimal Real Number 75 8515625 In Binary Then Re Write The Binary Result More

Related Articles

- [Compare The Solubility Of Barium Phosphate In Each Of The Following Aqueous Solutions:0.10 M NH₄NO₃0.10](#)
- [Given A Text File Containing The Availability Of Food Items, Write A Program That Reads The Information](#)
- [Consider An Economy That Is Initially In Long-run Equilibrium. Assume That The Longrun Aggregate Supply](#)

[Back to Home](#)