Bitwise 1’s and 2’s complement

**Decimal:** 10  
**Binary (8-bit):** 00001010

**1's Complement**

1's complement means **inverting all the bits** (changing 0s to 1s and 1s to 0s).

**Steps:**

Original (8-bit) = 00001010  
1's Complement = 11110101

**2's Complement**

2's complement means **taking the 1's complement and adding 1** to the result.

**Steps:**

1's Complement = 11110101  
Add 1 = 11110101 + 00000001 = 11110110

**Final 2's complement:** 11110110

**Summary**

| **Type** | **Binary** |
| --- | --- |
| Original (10) | 00001010 |
| 1's Complement | 11110101 |
| 2's Complement | 11110110 |

The **minus sign** appears in the result because of how **binary numbers are represented in computers**, especially using **2's complement** representation.

**Computers use 2's complement to represent negative numbers**

In an **8-bit system**, the **most significant bit (MSB)**—the leftmost bit—is used as the **sign bit**:

* 0 = **positive**
* 1 = **negative**

So, for example:

| **Binary** | **MSB** | **Decimal** |
| --- | --- | --- |
| 00001010 | 0 | +10 |
| 11110110 | 1 | -10 |

In your earlier example:

* You started with **+10**: 00001010
* Then you calculated the **2's complement**: 11110110
* That result is how **-10** is stored in binary using 2's complement.

**Why 2's complement ?**

Because it allows you to **do subtraction as addition**.  
For example:

5 - 3 → 5 + (-3)

Using 2's complement, negative numbers can be added just like positive numbers—no special subtraction logic needed.

The minus sign appears because in **2's complement**, when the MSB is 1, the number is interpreted as **negative**.