Sure! Let's break down the for loop with the statement cout << ((n >> i) & 1); using an example to see how it works step by step.

**Example**

Let's take an example where n = -5.

**Binary Representation of -5**

In a 32-bit system, -5 is represented in two's complement form. First, let's see the binary representation of 5, then take its two's complement to get -5:

1. **5 in binary** (32-bit representation):

Copy code

00000000 00000000 00000000 00000101

1. **Inverting the bits** (one's complement):

Copy code

11111111 11111111 11111111 11111010

1. **Adding 1 to the inverted bits** (two's complement):

Copy code

11111111 11111111 11111111 11111011

Thus, -5 in a 32-bit binary representation is:

Copy code

11111111 11111111 11111111 11111011

**Loop Explanation**

The loop for (int i = 31; i >= 0; i--) iterates from the most significant bit (31) to the least significant bit (0).

**Step-by-Step Execution:**

1. **Initialization**: i starts at 31.
2. **Condition Check**: i >= 0.
3. **Shift Operation**: n >> i shifts the bits of n right by i positions.
4. **Bitwise AND Operation**: (n >> i) & 1 extracts the least significant bit after the shift.
5. **Print the Bit**: cout prints the extracted bit.

Let's go through a few iterations for n = -5:

1. **i = 31**:
   * n >> 31: Shift all bits of n right by 31 positions.

Copy code

11111111 11111111 11111111 11111011 >> 31 = 1

* + (1 & 1): Extract the least significant bit.

Copy code

1 & 1 = 1

* + Print 1.

1. **i = 30**:
   * n >> 30: Shift all bits of n right by 30 positions.

less

Copy code

11111111 11111111 11111111 11111011 >> 30 = 11 (in binary: 11)

* + (11 & 1): Extract the least significant bit.

Copy code

11 & 1 = 1

* + Print 1.

1. **i = 29**:
   * n >> 29: Shift all bits of n right by 29 positions.

less

Copy code

11111111 11111111 11111111 11111011 >> 29 = 111 (in binary: 111)

* + (111 & 1): Extract the least significant bit.

Copy code

111 & 1 = 1

* + Print 1.

This process continues for each bit position down to i = 0.

**Full Execution for -5**

The full binary representation of -5 would be printed as:

Copy code

11111111111111111111111111111011

This matches the two's complement representation of -5.

**Summary**

The loop for (int i = 31; i >= 0; i--) iterates through each bit position from 31 to 0:

* n >> i shifts the bits of n right by i positions.
* (n >> i) & 1 extracts the least significant bit of the shifted value.
* The extracted bit is printed, building the binary representation of n bit by bit from the most significant bit to the least significant bit.