A signed number in binary code is a number that can be either positive or negative. It is represented using a single bit to indicate the sign of the number, and the remaining bits to represent the magnitude of the number.

The most common way to represent signed numbers in binary code is using **two's complement notation**. In two's complement notation, the sign bit is the most significant bit (MSB), and it is 0 for positive numbers and 1 for negative numbers. The remaining bits represent the magnitude of the number, which is calculated by taking the two's complement of the unsigned magnitude.

To calculate the two's complement of a number, we first invert all of the bits in the number, and then add 1. For example, the two's complement of the unsigned number 0101 is 1010, and the two's complement of the unsigned number 1010 is 0101.

The following table shows the two's complement representation of some signed numbers:

Decimal	Two's complement
0	0000
1	0001
2	0010
3	0011
-1	1111

Decimal	Two's complement
-2	1110
-3	1101

Two's complement notation is a very efficient way to represent signed numbers in binary code. It allows us to perform arithmetic operations on signed numbers without having to worry about the sign of the numbers.

Here is an example of how to add two signed numbers in binary code using two's complement notation:

```
0001 + 0010 = 0011
```

The sign bit of the result is 0, so the result is a positive number.

Two's complement notation is used in most computers and microprocessors to represent signed numbers. It is also used in many other applications, such as telecommunications and signal processing.