



[Courseware \(/courses/UTAustinX/UT.6.01x/1T2014/courseware\)](/courses/UTAustinX/UT.6.01x/1T2014/courseware)

[Course Info \(/courses/UTAustinX/UT.6.01x/1T2014/info\)](/courses/UTAustinX/UT.6.01x/1T2014/info)

[Discussion \(/courses/UTAustinX/UT.6.01x/1T2014/discussion/forum\)](/courses/UTAustinX/UT.6.01x/1T2014/discussion/forum)

[Wiki \(/courses/UTAustinX/UT.6.01x/1T2014/course_wiki\)](/courses/UTAustinX/UT.6.01x/1T2014/course_wiki)

[Progress \(/courses/UTAustinX/UT.6.01x/1T2014/progress\)](/courses/UTAustinX/UT.6.01x/1T2014/progress)

[Questions \(/courses/UTAustinX/UT.6.01x/1T2014/a3da417940af4ec49a9c02b3eae3460b/\)](/courses/UTAustinX/UT.6.01x/1T2014/a3da417940af4ec49a9c02b3eae3460b/)

[Syllabus \(/courses/UTAustinX/UT.6.01x/1T2014/a827a8b3cc204927b6efaa49580170d1/\)](/courses/UTAustinX/UT.6.01x/1T2014/a827a8b3cc204927b6efaa49580170d1/)

Help

Observation: If the least significant binary bit is zero, then the number is even.

Observation: If the right-most n bits (least sign.) are zero, then the number is divisible by 2^n .

Observation: Consider an 8-bit unsigned number system. If bit 7 is low, then the number is between 0 and 127, and if bit 7 is high then the number is between 128 and 255.

One of the first schemes to represent signed numbers was called **one's complement**. It was called one's complement because to negate a number, we complement (logical not) each bit.

For example, if 25 equals 00011001_2 in binary, then -25 is 11100110_2 . An 8-bit one's complement number can vary from -127 to $+127$. The most significant bit is a sign bit, which is 1 if and only if the number is negative. The difficulty with this format is that there are two zeros $+0$ is 00000000_2 , and -0 is 11111111_2 . Another problem is that one's complement numbers do not have basis elements. These limitations led to the use of two's complement.

The **two's complement** number system is the most common approach used to define signed numbers. It is called two's complement because to negate a number, we complement each bit (like one's complement), then add 1.

For example, if 25 equals 00011001_2 in binary, then -25 is 11100111_2 . If a byte is used to represent a signed two's complement number, then the value of the number is

$$N = -128 \cdot b_7 + 64 \cdot b_6 + 32 \cdot b_5 + 16 \cdot b_4 + 8 \cdot b_3 + 4 \cdot b_2 + 2 \cdot b_1 + b_0$$

Observation: The most significant bit in a two's complement signed number will specify the sign.

Observation: To take the negative of a two's complement signed number we first complement (flip) all the bits, then add 1.

To practice, you can enter an 8-bit binary number in the following field and press "show" to see its value if interpreted as signed or unsigned integer. For convenience, you can also enter hexadecimal input with '0x' prefix.

Unsigned value = $1 * 2^7 + 0 * 2^6 + 1 * 2^5 + 0 * 2^4 + 0 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0 = 165$

Signed Value = $-1 * 2^7 + 0 * 2^6 + 1 * 2^5 + 0 * 2^4 + 0 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0 = -91$

CHECKPOINT 2.12

Give the representations of the decimal 45 in 8-bit binary and hexadecimal.

Combine binary basis elements to create the desired value. $45 = 32 + 8 + 4 + 1$, so $45 = 00101101_2 = 0x2D$.

CHECKPOINT 2.13

Give the representations of the decimal 200 in 8-bit binary and hexadecimal.

Combine binary basis elements to create the desired value. $200 = 128 + 64 + 8$, so $200 = 11001000_2 = 0xC8$.

CHECKPOINT 2.14

Convert the signed binary number 11011010_2 .


Combine signed binary basis elements to create the desired value. $-128 + 64 + 16 + 8 + 2 = -38$.

CHECKPOINT 2.15

Are the signed and unsigned decimal representations of the 8-bit hex number 0x95 the same or different?

They are different, because bit 7 is one.



 <https://courses.edx.org/courses/UTAustinX/UT...>
(<https://plus.google.com/108235383044095082735/posts>)

 (<http://youtube.com/user/edxonline>)
© 2013 edX, some rights reserved.

Terms of Service and Honor Code -
Privacy Policy (<https://www.edx.org/edx-privacy-policy>)