Hexadecimal to Binary

To perform conversions between hex & binary, it is necessary to know the four-bit binary numbers (0000 – 1111), and their equivalent hex digits.

Hex	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111
key	value

Digital Systems: Principles and Applications, 11/e
Ronald J. Tocci, Neal S. Widmer, Gregory L. Moss

Hexadecimal to Binary

$$
\begin{aligned}
9F2_{16} &= \quad 9 \qquad\qquad F \qquad\qquad 2 \\
&\qquad \downarrow \qquad\qquad \downarrow \qquad\qquad \downarrow \\
&= 1\ 0\ 0\ 1 \quad 1\ 1\ 1\ 1 \quad 0\ 0\ 1\ 0 \\
&= 100111110010_2
\end{aligned}
$$

Practice:

$BA6_{16} = (?)_2$
$3A5_{16} = (?)_2$
$2E7C_{16} = (?)_2$

Binary to Hexadecimal

To perform conversions between hex & binary, it is necessary to know the four-bit binary numbers (0000 - 1111), and their equivalent hex digits.

Binary	Hex
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F
key	value

Digital Systems: Principles and Applications, 11/e
Ronald J. Tocci, Neal S. Widmer, Gregory L. Moss

Binary to Hexadecimal

- The binary number is grouped into groups of four bits
  - each group is converted to its equivalent hex digit.

- If needed, leading zeros can be padded left of the MSB (zfill) to form a group of four bits

$$1\,1\,1\,0\,1\,0\,0\,1\,1\,0_2 = \underbrace{\boxed{0\;0}\;1\;1}_{3}\;\underbrace{1\;0\;1\;0}_{A}\;\underbrace{0\;1\;1\;0}_{6}$$
$$= 3A6_{16}$$

```
Practice:
```

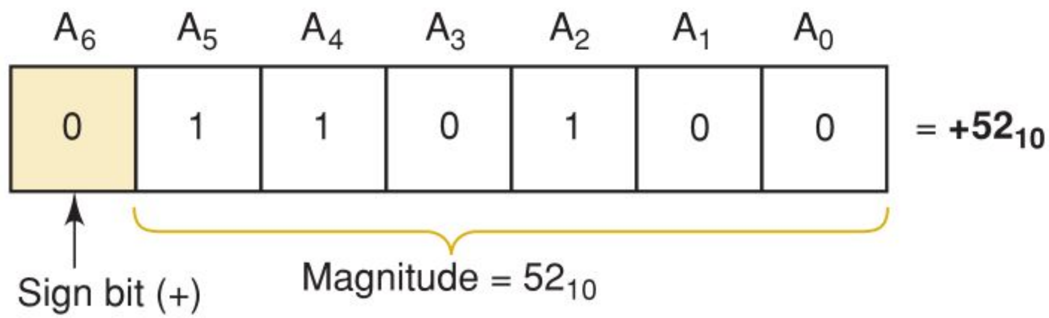
$101011111_2 = (?)_{16}$

$11011101_2 = (?)_{16}$

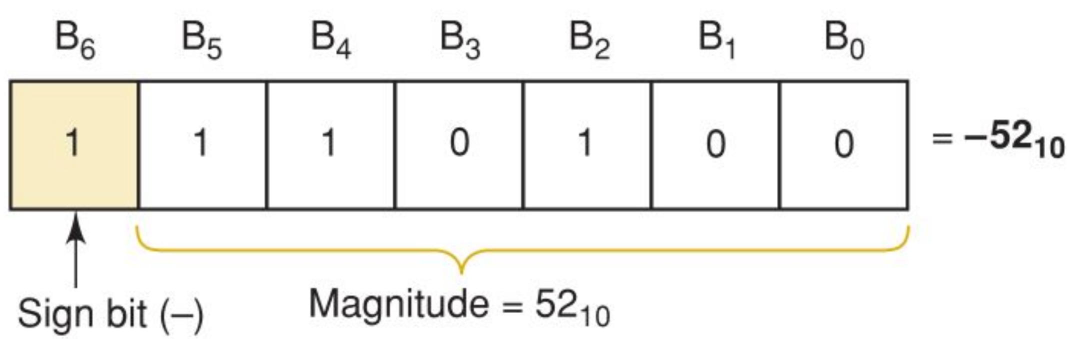
$100011010011_2 = (?)_{16}$

Signed binary integers: Sign-Magnitude representation

Sign-bit:
sign bit of **0** is placed in front of the MSB to indicate **positive** signed binary number.



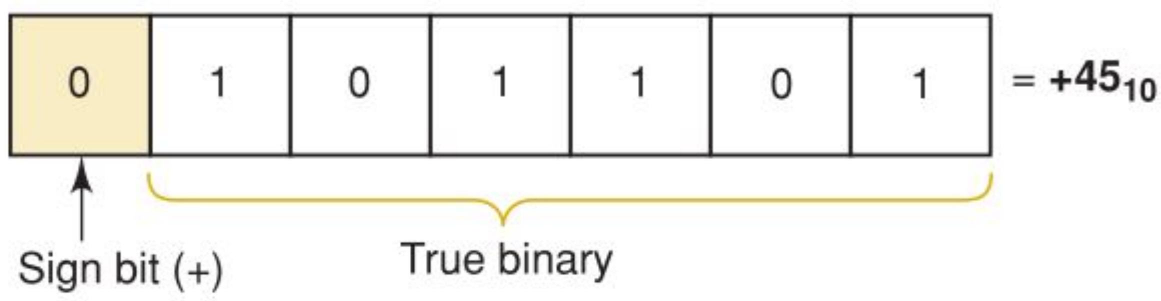
Sign-bit:
sign bit of **1** is placed in front of the MSB to indicate **negative** signed binary number.



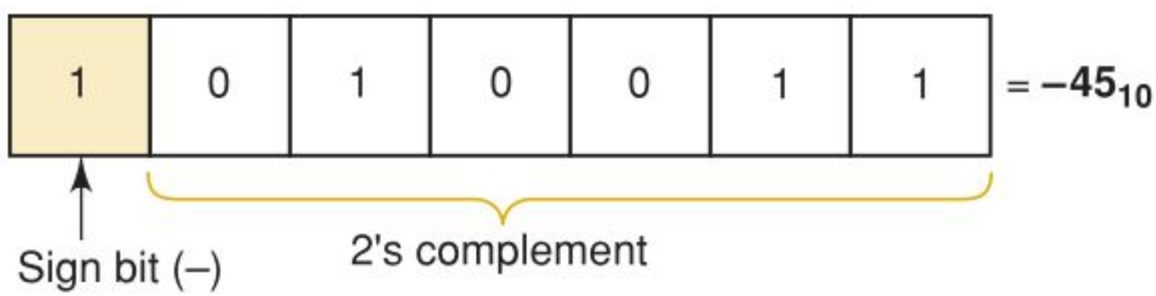
The magnitude bits are the *true binary* equivalent of the decimal value being represented

Signed binary integers: Two's Complement representation

Sign-bit:
sign bit of **0** is placed in front of the MSB to indicate positive signed binary number.



Sign-bit:
sign bit of **1** is placed in front of the MSB to indicate negative signed binary number.



If the number is negative, the magnitude is represented in its 2's- complement form, and a sign bit of **1** is placed in front of the MSB

Signed binary integers: Two's Complement representation

Two's complement of binary integer is formed by

1. inverting each bit
2. adding one

Given signed value:	**0101101**	$+45_{10}$
Invert bits:	1010010	
Add 1 to inverted value:	1010010 +0000001	
Two's complement representation:	**1010011**	-45_{10}

Note: two's complement operation is **reversible**

Given signed value:	**1010011**	-45_{10}
Invert bits:	0101100	
Add 1 to inverted value:	0101100 +0000001	
Two's complement representation:	**0101101**	$+45_{10}$

Signed binary integers: Two's Complement representation

Two's complement of binary integer is formed by

1. inverting each bit
2. adding one

Given signed value:	0111
Invert bits:	1000
Add 1 to inverted value:	1000 +0001
Two's complement representation	**1001**

Note: two's complement operation is **reversible**

Given signed value:	**1001**
Invert bits:	0110
Add 1 to inverted value:	0110 +0001
Two's complement representation	**0111**

Signed binary integers: Two's Complement representation

Practice

Two's complement of binary integer is formed by

1. inverting each bit
2. adding one

Given signed value:	`0110100`
Invert bits:	
Add 1 to inverted value:	
Two's complement representation:	

Note: two's complement operation is **reversible**

Given signed value:	
Invert bits:	
Add 1 to inverted value	
Two's complement representation:	

Signed binary integers: Two's Complement representation

Practice

Two's complement of binary integer is formed by

1. inverting each bit
2. adding one

Given signed value:	01100
Invert bits:	
Add 1 to inverted value:	
Two's complement representation:	

Note: two's complement operation is **reversible**

Given signed value:	
Invert bits:	
Add 1 to inverted value:	
Two's complement representation:	

Signed binary integers: Two's Complement representation

Sign Extension

Special Case

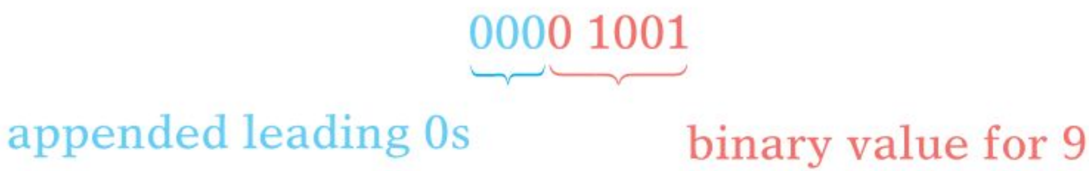
- Most digital systems today store numbers in registers sized in even multiples of four bits:
    - registers are usually 4, 8, 12, 16, 32, or 64 bits in length.

- The size of a register determines the length of the binary data that is stored in it.

Signed binary integers: Two's complement representation

Sign Extension

- If we need to store a **positive** five-bit number in an eight-bit register:
  Example: signed binary representation for +9 is `01001`



The MSB (sign bit) is still 0, indicating a positive value

- If we need to store **negative** five-bit numbers in an eight-bit register:
  Example: signed binary representation for -9 is `10111`

The proper way to extend a **negative** number is to append leading 1's.

Signed binary integers: Two's complement representation

Special Case

- Whenever a signed number has a 1 in the sign bit and all 0s for the magnitude bits, for example:

  1000 = 2^3 (-8 in decimal)
  10000 = 2^4 (-16 in decimal)
  100000 = 2^5 (-32 in decimal)
  ...

- **Special Case**: taking the two's complement of these numbers produces the value we started with!

- Because we are at the limit of the range of negative numbers that can be represented by this many bits.

- **FIX**: extend the sign of these special numbers:

  Given 1000 (-8)
  Extend the sign 11000
  Take 2's complement of 11000 we get the correct 01000 (+8)

Signed binary integers limit for the range of values

- Signed integer of n bits uses only $n - 1$ bits to represent the magnitude.

Example: Let $n = 4$ bits, only $n - 1 = 3$ bits to represent the magnitude.

two's complement	decimal
0111	7
0110	6
0101	5
0100	4
0011	3
0010	2
0001	1
0000	0
1111	−1
1110	−2
1101	−3
1100	−4
1011	−5
1010	−6
1001	−7
1000	−8

Signed binary integers limit for the range of values

- Signed integer of *n* bits uses only *n - 1* bits to represent the magnitude.

Therefore we have the following range of values for our signed data types:

Type	Range (low-to-high)	Power of 2
signed nibble	–8 to +7	-2^{3} to $(2^{3}-1)$
signed byte	–128 to +127	-2^{7} to $(2^{7}-1)$
signed word	–32,768 to +32,767	-2^{15} to $(2^{15}-1)$
signed double word	–2,147,483,648 to +2,147,483,647	-2^{31} to $(2^{31}-1)$
signed quadword	–9,223,372,036,854,775,808 to +9,223,372,036,854,775,807	-2^{63} to $(2^{63}-1)$

Signed binary integers: adding can result in overflow

<u>overflow for signed integers may occur if</u>:
The numbers being added have the same sign-bit, but the sign-bit of their sum differs

- two positives sum to a negative (clearly wrong)
- two negatives sum to a positive (clearly wrong)

Note: Adding a positive number and negative number (or vice-versa) cannot result in overflow

Suppose we are working in four-bit two's-complement representation:

0111 + 0001 = 1000 (overflow)

Example with two positives: 7 + 1 ≠ -8

1000 + 1111 = (1)0111 (overflow)

Example with two negatives: -8 + -1 ≠ 7

two's complement	decimal
0111	7
0110	6
0101	5
0100	4
0011	3
0010	2
0001	1
0000	0
1111	−1
1110	−2
1101	−3
1100	−4
1011	−5
1010	−6
1001	−7
1000	−8

signed decimal to signed binary

Algorithm

1. convert the absolute value of decimal integer to binary
2. If original number was negative, create the two's complement of the binary number

Example: -33 to signed binary

Given value:	`-33`
absolute value:	`33`
binary representation of 33	`00100001`

because original number was negative we obtain Two's complement representation	`11011110 invert the bits`
	`00000001 add one`
	`--------`
	`11011111 result`

`11011111 is the two's complement representation of -33`

signed binary to signed decimal

For a signed binary the algorithm for converting to decimal

1. if the MSB is 1:
   we know we have a negative binary integer in two's complement representation. Therefore we apply two's complement a second time to get its positive binary integer.

   else:
   we already have a positive binary integer
2. Then convert positive binary integer to decimal as if it was unsigned binary

Given value:	`11110000`

create two's complement representation	`00001111 invert bits`
	`00000001 add 1`
	`--------`
	`00010000`

convert to decimal	`16`
since the original number was negative the final answer is -16	`-16`

Two’s complement notation of Hexadecimal

Extra

Two's complement notation of Hexadecimal

Two's complement of hexadecimal is formed by

1. invert each digit
2. adding one

Note: an easy way to reverse the bits of a hexadecimal digit is to subtract the digit from 15

you can review older video on invert()

Given value:	`6A3D`
Invert digits:	`95C2`
Add 1	`95C2` `+0001`
Two's complement representation	`95C3`

signed hexadecimal integers:

- if MSD of hexadecimal >= 8:
  then our number is negative
- if MSD of hexadecimal <= 7
  then our number is positive
- **Example**:
  8A22 is negative
  73D1 is positive

```
Note: two's complement operation is reversible

95C3 → 6A3C + 0001 → 6A3D
```