# **Binary Arithmetic -- Negative numbers and Subtraction**

#### **Binary Mathematics**

<u>Binary Addition:</u> this is performed using the same rules as decimal except all numbers are limited to combinations of zeros (0) and ones (1).

Using 8-bit numbers

## Not all integers are positive.

What about negative numbers?

What if we wanted to do:  $14_{10} + (-2_{10}) = 12_{10}$ 

So,: 
$$14_{10} + (-2_{10}) = 12_{10}$$
  $\longleftrightarrow$   $14_{10} - (+2_{10}) = 12_{10}$   $14_{10}$   $-2_{10}$   $\cdots$   $12_{10}$ 

This example is a deceptively easy because while there no need to borrow. Let's look at another example:

If we are using a paper and pencil, binary subtraction "can" be done using the same principles as decimal subtraction.

Binary Subtraction: Use standard mathematical rules:

This is rather straightforward. In fact no borrowing was even required in this example.



Try this example:

#### THIS WAS PAINFUL!

When so much borrowing is involved the problem is very error prone.

Not only was the example above complex because of all the borrowing required but computers have additional problems with signed or negative numbers. Given the nature of the machine itself, how do we represent a negative number? What makes this even worse is that computers are fixed-length or finite-precision machines.

There are two common ways to represent negative numbers within the computer. Remember, the minus sign does not exist. The computer world is made up entirely of zeros (0) and ones (1). These two techniques are called signed magnitude representation and two's complement.

Let's explore sign-magnitude representation first. In the sign-magnitude number system, the most significant bit, the leftmost bit, holds the sign (positive or negative). A zero (0) in that leftmost bit means the number is positive. A one (1) in that leftmost bit means the number is negative.

<u>Step 1</u>: Decide how many bits the computer has available for your operations. Remember computers are fixed-length (or finite-precision) machines.

For example: if we use 4-bits, the leftmost bit is the sign bit and all the rest are used to hold the binary numbers. In a 4-bit computer world, this leaves only 3 bits to hold the number.

This limits our numbers to only very small ones.

A 4-bit number would look like X X X X the left-most bit is considered the <u>sign</u> bit

|
This is the sign bit

Using four bits, these are the ONLY binary numbers a computer could represent.

0	0000		
1	0001	-1	<b>1</b> 001
2	<b>0</b> 010	-2	<b>1</b> 010
3	<b>0</b> 011	-3	<b>1</b> 011
4	<b>0</b> 100	-4	<b>1</b> 100
5	<b>0</b> 101	-5	<b>1</b> 101
6	<b>0</b> 110	-6	<b>1</b> 110
7	<b>0</b> 111	-7	<b>1</b> 111

If we were using 8-bits the left-most bit will contain the sign. This would leave 7 bits to hold the number.

XXXX XXXX |
This is the sign bit



This sign bit is <u>reserved</u> and is no longer one of the digits that make up the binary number. Remember if the sign bit is **zero (0)** the binary number following it is <u>positive</u>. If the sign bit is **one (1)** the binary number following it is **negative**.

Using the sign-magnitude system the largest positive number that can be stored by an 8-bit computer is:

This is: 
$$64 + 32 + 16 + 8 + 4 + 2 + 1 = 127_{10}$$

If there were a one (1) in the first bit, the number would be equal to  $-127_{10}$ 

Over time it has become obvious that a system that even further reduces the number of available bits while meaningful, is not especially useful.

Then of course there is still the problem of how to deal with these positive and negative numbers. While this representation is simple, arithmetic is suddenly impossible. The standard rules of arithmetic don't apply. Creating a whole new way to perform arithmetic isn't overly realistic.

Fortunately another technique is available.

#### Two's Complement

Two's complement is an alternative way of representing negative binary numbers. This alternative coding system also has the unique property that subtraction (or the addition of a negative number) can be performed using addition hardware. Architects of early computers were thus able to build arithmetic and logic units that performed operations of addition and subtraction using only adder hardware. (As it turns out since multiplication is just successive addition and division is just successive subtraction it was possible to use simple adder hardware to perform all of these operations.

Let's look at an example:

$$14_{10} - 6_{10} = 14_{10} + (-6_{10}) = 8_{10}$$

0000 1110
$$_2$$
 + 1000 0110 $_2$  =  $?_2$ 

left-most digit is 0
number is positive

<u>Step 1</u>: Decide how many bits you are going to use for all your operations. For our purposes we will always use 8 bits.

If we were using 8-bits the left-most bit will contain the sign. This would leave 7 bits to hold the number.



This sign bit is <u>reserved</u> and is no longer one of the digits that make up the binary number. Using two's complement, the computer recognizes the presence of a one (1) in the leftmost bit which tells the machine that before it does mathematics it needs to convert negative numbers into their two's compliment equivalent.

the sign bit is 0 so the number is positive. The binary number is 7-digits long,
 the sign bit is 1 so the number is negative. The binary number is only 7-digits long,

# Example 1:

$$14_{10}$$
 -  $6_{10}$  =  $14_{10}$  + (-  $6_{10}$ ) =  $8_{10}$ 
 $0000$  1110<sub>2</sub> + 1000 0110<sub>2</sub> = ?<sub>2</sub>

| left-most digit is 0 | left-most digit is 1, number is negative number is positive

Step 2: Strip the sign bits off the numbers.

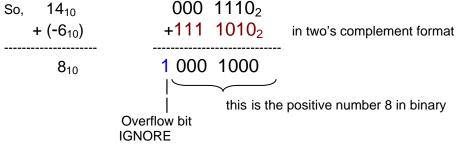
Step 3: Convert the negative number into it's two's complement form.

**Note**: If neither of the number were negative we would be doing simple addition and this would not be necessary.

How do we find the two's complement of -6?

Write down the number without the sign bit ------  $000 \ 0110_2$  a) Flip all the digits The  $1 \rightarrow 0$ , the  $0 \rightarrow 1$   $111 \ 1001_2$  b) Add 1 to this number  $111 \ 1010_2$  c) This is now - 6 in the two's complement format

Step 4: **Add** the two's complement in place of the negative number.



IT Worked!



# Example 2:

$$12_{10} - 9_{10} = 3_{10}$$

Or

$$12_{10} + (-9_{10}) = 3_{10}$$

$$0000 \ 1100_2 + 1000 \ 1001_2 = ?_2$$
Positive 12 Negative 9

Step 1: Determine the number of bits we are using. Choose 8 bits

$$12_{10} = 0000 1100_2$$

$$-9_{10} = 1000 1001_2$$

Step 2: Strip off the sign bits.

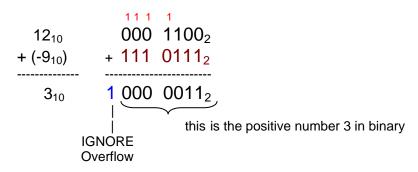
Step 3: Determine the 2's complement of the negative number, or the number to be subtracted.

Find the 2's complement of - 9<sub>10</sub>

Write down the number  $000 \ 1001_2$  without the sign bit

A) Flip all the digits  $0 \rightarrow 1, 1 \rightarrow 0$ B) Add One(1) + 1C) This is Two's Complement  $111 \ 0111_2$  this is the 1's complement  $111 \ 0111_2$ 

**Step 4**: Add the numbers together. In this case, add  $12_{10}$  in binary (000  $1100_2$ ) and the two's complement of -  $9_{10}$  in binary (111 0111). Ignore any overflow.



IT Worked!



#### Example 3:

$$25_{10} - 14_{10} = 11_{10}$$

Or

$$25_{10} + (-14_{10}) = 11_{10}$$

$$0001 \ 1001_2 + 1000 \ 1110_2 = 0000 \ 1011$$
Positive 25 Negative 14 Positive 11

Step 1: Determine the number of bits we are using. Choose 8 bits

Step 2: Strip off the sign bits.

Step 3: Determine the 2's complement of the negative number, or the number to be subtracted.

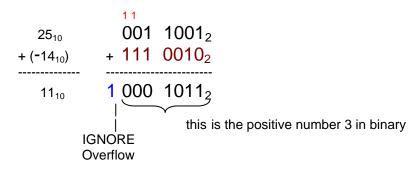
Find the 2's complement of - 14<sub>10</sub>

Write down the number without the sign bit

A) Flip all the digits  $0 \rightarrow 1, 1 \rightarrow 0$ B) Add One(1)

C) This is the Two's Complement form of -9  $000 \ 1110_2$ this is the 1's complement this is the 1's complement on the sign bit is 1, negative number 111  $0010_2$ Note: left-most bit is 1, negative number 111  $0010_2$ 

**Step 4**: Add the numbers together. In this case, add 12<sub>10</sub> in binary (000 1100<sub>2</sub>) and the two's complement of 9<sub>10</sub> in binary (111 0111). Ignore any overflow.



In all of the above examples, the resulting number, the answer was positive.

What happens if we are working with numbers where the result is negative (where the negative number is larger than the positive number).

#### **NOTE THIS:**



IT Worked!

The basic computational process is the same. Negative numbers are converted to their two's complement equivalent. But since the resulting number is negative, rather than positive, twos complement is used again to convert the number back to standard binary.

### Example 4:

Step 1: Decide on the number of binary digits -- Choose 8

Step 2: Strip off the sign bits.

**Step 3**: Convert the negative number to it's twos complement equivalent.

 $9_{10} - 14_{10} = 9_{10} + (-14_{10}) = -5$ 

Write down the negative number without sign bit	000 1110 <sub>2</sub>
A) Flip the digits	111 0001 <sub>2</sub>
B) Add one (1)	+ 1
C) This is two's complement	111 0010 <sub>2</sub>

Step 4: Add the two numbers together.

$$9_{10}$$
  $000 \ 1001_2$   $+ (-14_{10})$   $+ \ 111 \ 0010_2$  in two's complement format  $000 \ 1001_2$  in two's complement format  $000 \ 1001_2$  in two's complement format  $000 \ 1001_2$   $000 \ 1001_2$  in two's complement format  $000 \ 1001_2$   $0000 \ 1001_2$   $0000 \ 1001_2$   $0000 \ 1001_2$   $0000 \ 1001_2$   $0000 \ 1001_2$   $0000 \ 1001_2$ 

Step 5: Convert from two's complement by doing the two's complement process again.

Write down the number	111 1011 <sub>2</sub>
A) Flip the digits	000 01002
B) Add one (1)	+ 1
C) This is two's complement D) We know the result	000 0101 <sub>2</sub> positive 5, in standard binary
Is negative, change sign bit	1000 0101 <sub>2 negative 5, in standard binary</sub>



# Example 5:

$$-25 + 18 = -7$$

Proceed the same way, converting the negative number to it's two's complement equivalent.

Decimal	Standard Binary		
- 25 <sub>10</sub> + 18 <sub>10</sub>	1001 1001 <sub>2</sub> + 0001 0010 <sub>2</sub>	left-most digit says negative left-most digit says positive	
+ 10 <sub>10</sub>		leit-most digit says positive	
<b>-</b> 7 <sub>10</sub>			

Step 1: Decide on the number of binary digits -- Choose 8

Step 2: Strip off the sign bit.

**Step 3**: Convert the negative number to it's twos complement equivalent.

Write down the negative number Without the sign bit	001 1001 <sub>2</sub>
A) Flip the digits B) Add one (1)	110 0110 <sub>2</sub> + 1
C) This is two's complement	110 0111 <sub>2</sub>

**Step 4**: Add the two numbers together.

Step 5: Convert from two's complement by doing the two's complement process again.

Write down the number	111 1001 <sub>2</sub>
A) Flip the digits B) Add one (1)	000 0110 <sub>2</sub> + 1
, , , ,	000 0111
C) This is two's complement D) We know the result	000 0111 <sub>2</sub> positive 7, in standard binary
Is negative, change sign bit	1000 1110 <sub>2 negative 7, in standard binary</sub>



Try these problems:

1) 
$$9_{10} - 7_{10} = 2_{10}$$
  
 $9_{10} + (-7_{10}) = 2_{10}$  0000 1001 + 1000 0111 =

2) 
$$17_{10} - 12_{10} = 5_{10}$$
  $17_{10} + (-12_{10}) = 5_{10}$  0001 0001 + 1000 1100 =

3) 
$$18_{10} - 22_{10} = -4_{10}$$
  $18_{10} + (-22_{10}) = -4_{10}$  0001 0010 + 1001 0110 =

4) 
$$7_{10} - 19_{10} = -12_{10}$$
  $7_{10} + (-19_{10}) = -12_{10}$  0000 0111 + 1001 0011 =

5) 
$$-26_{10} + 10_{10} = -16_{10}$$
 1001 1010 + 0000 1010 =

6) 
$$-19_{10} + 6_{10} = -13_{10}$$
 1001 0011 + 0000 0110 =



# **Solutions:**

1) 
$$9_{10} - 7_{10} = 2_{10}$$
  
This is the same as  $9_{10} + (-7_{10}) = 2_{10}$ 

Strip the sign bit.

Convert the negative number to it's 2's complement equivalent

Now add the numbers

$$000 \ 1001 + 111 \ 1001$$

$$1 \ 000 \ 0010$$

$$= 2_{10}$$

ignore overflow

2) 
$$17_{10} - 12_{10} = 5_{10}$$
  
This is the same as  $17_{10} + (-12_{10}) = 5_{10}$ 

Strip the sign bit.

Convert the negative number to it's 2's complement equivalent

Now add the numbers

ignore overflow



3) 
$$18_{10} - 22_{10} = -4_{10}$$
  
This is the same as

$$18_{10} + (-22_{10}) = -4_{10}$$

0001 0010 + 1001 0110 =

Strip the sign bit.

Convert the negative number to it's 2's complement equivalent

This is 2's comp 110 1010

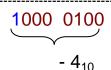
Now add the numbers

No Overflow

This is still in 2's complement Apply 2's complement to number again

111 1100
Flip the digits 000 0011
Add one(1) + 1

Put the sign bit back on we know the number is negative



4) 
$$7_{10} - 19_{10} = -12_{10}$$
  
This is the same as  $7_{10} + (-19_{10}) = -12_{10}$ 

Strip the sign bits.

Convert the negative number to it's 2's complement equivalent



Now add the numbers

000 0111
+ 110 1101

No Overflow
This is still in 2's complement
Apply 2's complement to number again
111 0100
Flip the digits
O00 1011
Add one(1) + 1

Put the sign bit back on
we know the number

5)  $-26_{10} + 10_{10} = -16_{10}$  1001 1010 + 0000 1010 = Strip the sign bits. Convert the negative number to it's 2's complement equivalent.

- 12<sub>10</sub>

1001 1010 \_\_\_\_\_\_ 001 1010 Flip the digits 110 0101 Add one(1) + 1 This is 2's comp 110 0110

is negative

Now add the numbers

110 0110
+ 000 1010

111 0000

No Overflow
This is still in 2's complement
Apply 2's complement to number again
111 0000
Flip the digits
O00 1111
Add one(1) + 1

Put the sign bit back on we know the number is negative

1001 0000

- 16<sub>10</sub>



6) 
$$-19_{10} + 6_{10} = -13_{10}$$
 1001 0011 + 0000 0110 =

Convert the negative number to it's 2's complement equivalent

Now add the numbers

No Overflow

This is still in 2's complement Apply 2's complement to number again

Tip the digits 000 1100
Add one(1) + 1

Put the sign bit back on we know the number is negative - 13<sub>10</sub>

