Binary Arithmetic -- Negative numbers and Subtraction

Binary Mathematics

Binary Addition: 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

$$\begin{array}{rcr} 1\ \ 1\ 1\ \ \ \ \ \ \ \ & & \\ 0000\ 1110_2 & \text{which is} & 14_{10} \\ +\ 0000\ 0110_2 & \text{which is} & +\ \ 6_{10} \\ \hline 0001\ 0100_2 & & 20_{10} \end{array}$$

Not all integers are positive.

What about negative numbers?

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

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

$$\begin{array}{r} 14_{10} \\ -\ 2_{10} \\ \hline 12_{10} \end{array}$$

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

$$\begin{array}{r} 1\ 13\ \ \ \\ 1\ \cancel{2}\ 3_{10} \\ -\ \ 1\ 9_{10} \\ \hline 1\ 0\ 4_{10} \end{array}$$

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:

$$\begin{array}{rcr} 0000\ 1110_2 & \text{which is} & 14_{10} \\ -\ 0000\ 0110_2 & \text{which is} & -\ \ 6_{10} \\ \hline \end{array}$$

$0000\ 1000_2$ 8_{10}

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

Try this example:

$$\begin{array}{cccccccccc}
 & & & & 0 & 10 & \overset{1}{\cancel{10}} & 10 & & \\
 & 0 & 0 & 0 & \cancel{1} & \cancel{1} & 0 & 0 & 1_2 & \rightarrow \quad 25_{10} \\
- & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0_2 & \rightarrow \quad -14_{10} \\
\hline
 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1_2 & 11_{10}
\end{array}$$

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.

Step 1: 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 **sign** bit

|
This is the sign bit

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

0	0000		
1	0001	-1	1001
2	0010	-2	1010
3	0011	-3	1011
4	0100	-4	1100
5	0101	-5	1101
6	0110	-6	1110
7	0111	-7	1111

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 **reserved** 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 **positive**. 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:

0	1	1	1	1	1	1	1	=	+ 127_{10}
Sign	(64)	(32)	(16)	(8)	(4)	(2)	(1)		

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}

1	1	1	1	1	1	1	1	=	- 127_{10}
Sign	(64)	(32)	(16)	(8)	(4)	(2)	(1)		

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:

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

We'll first need to figure out how to write numbers in the 2's complement form?

An 8-bit positive number in 2's complement form,

0	X	X	X	X	X	X	X
Sign	(64)	(32)	(16)	(8)	(4)	(2)	(1)

An 8-bit negative numbers in 2's complement form,

1	X	X	X	X	X	X	X
Sign	(?)	(?)	(?)	(?)	(?)	(?)	(?)

O.K. Then, 14_{10} is: 0 0 0 0 1 1 1 0, but what about -6_{10}?

Negative numbers in the 2's complement form need to be obtained by:

(1) Finding its corresponding positive number (+6 is: 0 0 0 0 0 1 1 0)

(2) Flip all digits (1→0, 0→1):

0 0 0 0 0 1 1 0
1 1 1 1 1 0 0 1

(3) Add "1" to the flipped number:

1 1 1 1 1 0 0 1
+) 0 0 0 0 0 0 0 1
----
1 1 1 1 1 0 1 0 ← this is the 2's complement representation for -6$_{10}$

Therefore, since

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

we simply do:

1 1 1 1 1 1
0 0 0 0 1 1 1 0$_2$ 14$_{10}$
+)1 1 1 1 1 0 1 0$_2$ +) -6$_{10}$
--------------------------- ----------
~~1~~ 0 0 0 0 1 0 0 0$_2$ 8$_{10}$
|
Overflow (9th bit in our 8-bit system), IGNORE!

It is always a good habit to examine if the answer we obtained (00001000$_2$) is correct. We do so by converting it back to decimal numbers and check if its indeed 8$_{10}$.

Example 1:

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

Or

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

Step 1: Determine the 2's complement form for the numbers:

$12_{10} = 00001100_2$

Finding -9$_{10}$ would require the following three steps:

(1) Finding -9$_{10}$'s corresponding positive number (+9$_{10}$ is: 0 0 0 0 1 0 0 1)

(2) Flip all digits (1→0, 0→1):

0 0 0 0 1 0 0 1
1 1 1 1 0 1 1 0

(3) Add "1" to the flipped number:

1 1 1 1 0 1 1 0
+) 0 0 0 0 0 0 0 1
----
1 1 1 1 0 1 1 1 ← this is the 2's complement form for -9$_{10}$

Step 2: Add the numbers together.

$$\begin{array}{lll}
 & & \scriptstyle 1\ 1\ \ 1\quad\ 1 \\
\quad 12_{10} & & \ \ 0000\ \ 1100_2 \\
+\ (-9_{10}) & & +\ \ 1111\ \ 0111_2 \\
\text{--------------} & & \text{------------------------} \\
\quad\ 3_{10} & & \cancel{1}\ 0000\ \ 0011_2 \leftarrow \text{this is the positive number 3} \\
 & & | \\
 & & \text{IGNORE} \\
 & & \text{Overflow}
\end{array}$$

IT Worked!

Example 2:

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

Or

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

Step 1: Determine the 2's complement form for the numbers:

$25_{10} = 00011001_2$

Finding -14_{10} would require the following three steps:

(1) Finding -14_{10}'s corresponding positive number ($+14_{10}$ is: 0 0 0 0 1 1 1 0)

(2) Flip all digits (1→0, 0→1):

$$\begin{array}{l}
0\ 0\ 0\ 0\ 1\ 1\ 1\ 0 \\
1\ 1\ 1\ 1\ 0\ 0\ 0\ 1
\end{array}$$

(3) Add "1" to the flipped number:

$$\begin{array}{ll}
 & 1\ 1\ 1\ 1\ 0\ 0\ 0\ 1 \\
+) & 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1 \\
\text{---------------------------} & \\
 & 1\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ \leftarrow \text{this is the 2's complement form for } -14_{10}
\end{array}$$

Step 2: Add the numbers together.

$$\begin{array}{lll}
 & & \scriptstyle\ \ \ 1\ 1\ 1 \\
\quad 25_{10} & & \ \ 0001\ 1001_2 \\
+\ (-14_{10}) & & +\ \ 1111\ 0010_2 \\
\text{--------------} & & \text{------------------------} \\
\quad 11_{10} & & \cancel{1}\ 0000\ 1011_2 \leftarrow \text{this is the positive number 11} \\
 & & | \\
 & & \text{IGNORE} \\
 & & \text{Overflow}
\end{array}$$

IT Worked!

Example 3:

$9_{10} - 14_{10}$

Or

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

Step 1: Determine the 2's complement form for the numbers:

$9_{10} = 00001001_2$

Finding -14_{10} would require the following three steps:

(1) Finding -14_{10}'s corresponding positive number ($+14_{10}$ is: 0 0 0 0 1 1 1 0)

(2) Flip all digits (1→0, 0→1):

```
    0 0 0 0 1 1 1 0
    1 1 1 1 0 0 0 1
```

(3) Add "1" to the flipped number:

```
    1 1 1 1 0 0 0 1
+)  0 0 0 0 0 0 0 1
---------------------------
    1 1 1 1 0 0 1 0
```

← this is the 2's complement form for -14_{10}

Step 2: Add the numbers together.

$$\begin{array}{rcr} 9_{10} & \qquad & 0000\ 1001_2 \\ +\ (-14_{10}) & & +\ \ 1111\ 0010_2 \\ \hline -5_{10} & & 1111\ 1011_2 \end{array}$$

Ugh… Since the answer starts with a "1" in its sign bit, it is a negative number. However, since there is no direct meaning for each digit for a negative 2's complement number, we'll need to find its corresponding positive number in order to find out:

(Note that we do the "flipping/adding 1" thing to convert any positive 2's complement number to its corresponding negative number and vice versa):

$$\begin{array}{lr} & 1111\ 1011_2 \\ \text{flip all digits:} & 0000\ 0100_2 \\ \text{add 1} & +)\ 0000\ 0001_2 \\ \hline & 0000\ 0101_2 \end{array}$$

← this is $+5_{10}$ indeed, therefore $1111\ 1011_2 = -5_{10}$

Example 5:

-25 + 18

Step 1: Determine the 2's complement form for the numbers:

18_{10} = 00010010_2

Finding -25_{10} would require the following three steps:

(1) Finding -25_{10}'s corresponding positive number ($+25_{10}$ is: 0 0 0 1 1 0 0 1)

(2) Flip all digits (1→0, 0→1):

0 0 0 1 1 0 0 1
1 1 1 0 0 1 1 0

(3) Add "1" to the flipped number:

```
    1 1 1 0 0 1 1 0
+)  0 0 0 0 0 0 0 1
---------------------------
    1 1 1 0 0 1 1 1   ← this is the 2's complement form for -25₁₀
```

Step 2: Add the numbers together.

	1 1
(-25_{10})	$1110\ 0111_2$
$+\ \ 18_{10})$	$+\ 0001\ 0010_2$
--------------	------------------------
-7_{10}	$1111\ 1001_2$

Ugh… Since the answer starts with a "1" in its sign bit, it is a negative number. However, since there is no direct meaning for each digit for a negative 2's complement number, we'll need to find its corresponding positive number in order to find out:

(Note that we do the "flipping/adding 1" thing to convert any positive 2's complement number to its corresponding negative number and vice versa):

	$1111\ 1001_2$
flip all digits:	$0000\ 0110_2$
add 1	+) $0000\ 0001_2$

	$0000\ 0111_2$ ← this is $+7_{10}$ indeed, therefore $1111\ 1001_2 = -7_{10}$