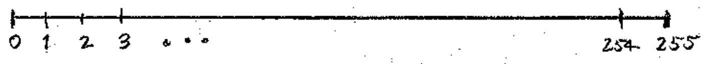
A Note on 2's Complement Arithmetic

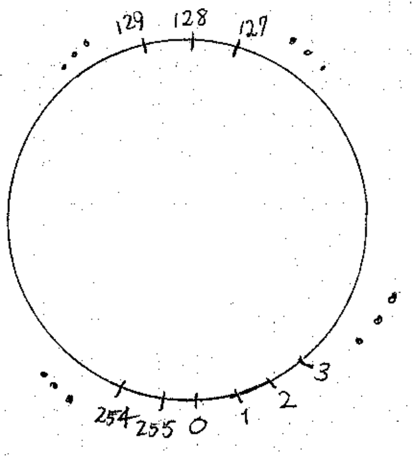
JCoonen 10Sept80

In an eight bit byte one can represent the whole numbers 0 through 255, that is 00_{16} through FF_{16}.

A natural question is how best to implement signed arithmetic on bytes, with extensions to 16 and 32 bits. First consider the set of representable magnitudes as a subset of the real line:



In eight bit arithmetic, adding 1 to 255 results in 0, with a carry out from the leading bit. Thus the representable set is more accurately depicted as a circle:



On the circle counterclockwise movement corresponds to addition (ignoring carries out of the lead bit), and clockwise movement corresponds to subtraction (ignoring borrows).

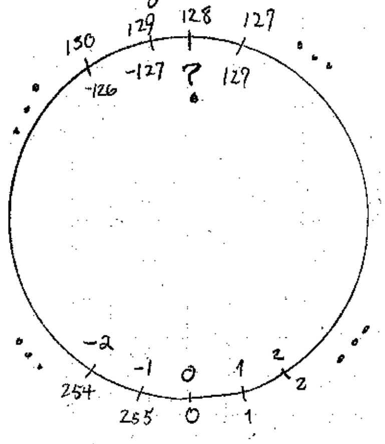
Ideally, the signed number system should have the properties:

1. Zero is represnted by 00_{16}.

2. Addition and subtraction of a number by 1 is achieved by the obvious 8-bit magnitude arithmetic, ignoring carries and borrows from the lead bit.

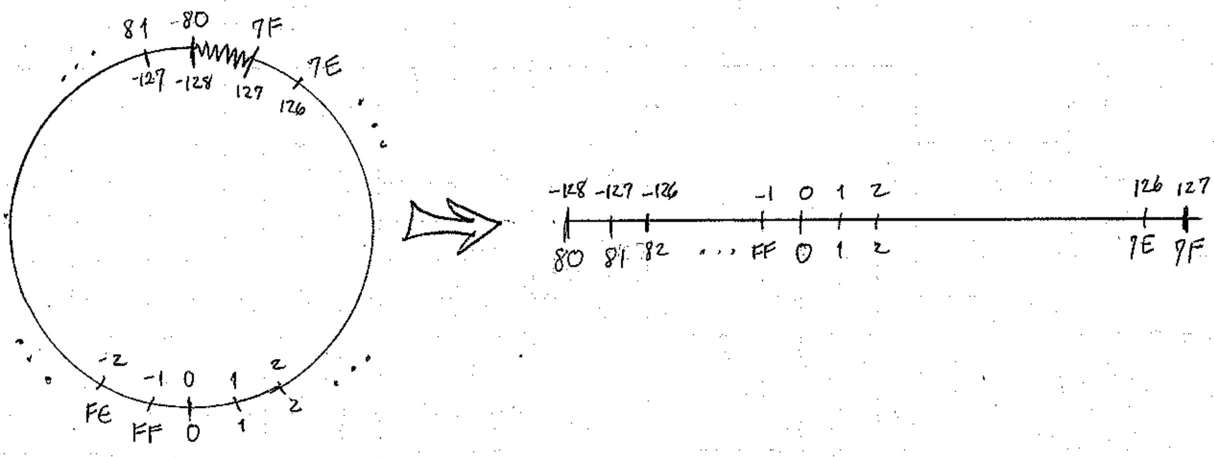
3. If a nonzero number is representable then so is its negative.

These properties imply the encoding:



In terms of arithmetic and its implied order, a "linear" subset of the signed integers is being represented by a "circular" set, namely the numbers 0 through 255 with their 8-bit magnitude arithmetic. At some point the circle must break as it is mapped onto the linear segment.

By convention the break is made between $7F_{16}$ and 80_{16}:



Property 3 above is satisfied for all bit patterns except 80_{16}. This situation must arise since there are 256 numbers and 0 has no negative. 80_{16} is arbitrarily set to represent -128 (rather than $+128$) so that the lead bit of the byte is 1 precisely when the represented number is negative.

Two equivalent statements describe the representation:

1) If $80_{16} \le X \le FF_{16}$ then X represents the number

$$x = X - 2^8 = -(2^8 - X).$$

2) If $1 \le x \le 7F_{16}$ then $-x$ is represented by $X = 2^8 - x$,

where $80_{16} \le X \le FF_{16}$.

The name "2's complement" dates back to a number system described by John von Neumann in the mid-40's for the EDVAC and similar machines. Von Neumann, who argued against use of floating-point arithmetic, used a fixed-point number system normalized so that $1 \le x < 2$ for positive representable number x. The negative of x is represented by $2 - x$, a true <u>two's</u> <u>complement</u>!

The following pages describe various 2's-complement operations.

If x and y are representable numbers, the sum $x+y$ computed by simply adding $X+Y$, where X, Y are the 8-bit strings representing x and y.

$z = x+y$ is accomplished by $Z = X+Y$, a simple magnitude add of 8-bit numbers.

Case 1: $x \geq 0 \Rightarrow x = X$ $\quad y \geq 0 \Rightarrow y = Y$
$Z = X+Y = x+y$

1.1) $0 \leq x+y < 2^7$
$Z = x+y$
Z rep. $x+y$

1.2) $2^7 \leq x+y < 2^8$
$Z = 2^8 - (2^8 - (x+y))$
Z rep. $-(2^8-(x+y)) = (x+y) - 2^8$

Comment: Arithmetic overflow causes a "large" positive result to "wrap around" to a negative number.

Case 2: $x < 0 \Rightarrow x = -(2^8 - X)$ $\quad y \geq 0 \Rightarrow y = Y$
$Z = X+Y = 2^8 + x + y$

2.1) $x+y \geq 0$
$Z = x+y$ + carry-out
Z rep. $x+y$

2.2) $x+y < 0$
$Z = 2^8 - (-(x+y))$
Z rep. $x+y$ ... the usual 2's-complement notation for a negative number

Case 3: $x \geq 0$, $y < 0$ is similar to Case 2.

Case 4: $x < 0 \Rightarrow x = -(2^8 - X)$ $\quad y < 0 \Rightarrow y = -(2^8 - Y)$
$Z = X+Y = 2^8 + 2^8 + x + y$

4.1) $-2^7 \leq x+y < 0$
$Z = 2^8 + \left[2^8 - (-(x+y))\right]$
$= 2^8 - (-(x+y))$ + carry-out
Z rep. $x+y$

4.2) $-2^8 < x+y < -2^7$
$Z = 2^8 + \left[2^8 + (x+y)\right]$
$= 2^8 + (x+y)$ + carry-out
Z rep. $2^8 + (x+y)$

Comment: A negative number of large magnitude has "wrapped around" to a positive number.

Two representable numbers x and y are subtracted or compared by adding X and the 2's-complement of Y. (Comparison is simply subtraction with no result delivered.)

$z = x-y$ is accomplished by $Z = X + (2^8 - Y)$

Case 1: $x \geq 0 \Rightarrow x = X$ $\quad y \geq 0 \Rightarrow y = Y$
$Z = X + (2^8 - Y) = 2^8 + (x-y)$

1.1) $x \geq y$
$Z = x-y$ + carry-out ... a carry-out but no borrow!
Z rep. $x-y$

1.2) $x < y$
$Z = 2^8 - (y-x)$ ... no carry-out but a borrow!
Z rep. $x-y$

Case 2: $x < 0 \Rightarrow x = -(2^8 - X)$ $\quad y \geq 0 \Rightarrow y = Y$
$Z = X + (2^8 - Y) = 2^8 + 2^8 + (x-y)$

2.1) $-2^7 \leq x-y < 0$
$Z = 2^8 - (y-x)$ + carry ... carry but no borrow!
Z rep. $x-y$

2.2) $-2^8 < x-y < -2^7$
$Z = 2^8 + (x-y)$ + carry ... carry but no borrow!
Z rep $2^8 + (x-y)$ ... overflow wraps $x-y$ to a positive number.

Case 3: $x > 0 \Rightarrow x = X$ $\quad y < 0 \Rightarrow -(2^8 - Y)$
$Z = X + (2^8 - Y) = x-y$

3.1) $0 \leq x-y < 2^7$
$Z = x-y$
Z rep. $x-y$

3.2) $2^7 \leq x-y < 2^8$
$Z = x-y = 2^8 - (2^8 - (x-y))$
Z rep. $-(2^8 - (x-y)) = (x-y) - 2^8$ ... overflow wraps positive $x-y$ to a negative number

Case 4: $x<0 \Rightarrow x = -(2^8-X)$ $y<0 \Rightarrow y = -(2^8-Y)$

$Z = X + (2^8-Y) = 2^8 + (x-y)$

4.1) $x-y>0$

$Z = x-y + \text{carry}$

Z rep. $x-y$

4.2) $x-y<0$

$Z = 2^8-(y-x)$

Z rep $x-y$

Since negation is effected by 2's complementation, $X-Y$ is computed as $X+(2^8-Y)$ or $2^8+(X-Y)$. There is carry-out of this quantity if and only if $X \geq Y$, if and only if there is no borrow. Since the Carry bit is used to chain "borrows" across multiple-precision operations, after a subtraction or comparison operation, the C bit is set to the complement of the carry-out of $X+(2^8-Y)$.

If x and y are representable numbers the double-width product $x \cdot y$ never overflows. The product is computed from bit strings X and Y according to the signs of x and y:

$z = x \cdot y$ $Z =$ ______

Case 1: $x \geq 0 \Rightarrow x = X$ $y \geq 0 \Rightarrow y = Y$
$Z = X \cdot Y$
Z rep. $x \cdot y$

Case 2: $x < 0 \Rightarrow x = -(2^8 - X)$ $y \geq 0 \Rightarrow y = Y$
$Z = 2^{16} - ((2^8 - X) \cdot Y)$
$= 2^{16} - 2^8 \cdot Y + X \cdot Y$
Z rep. $x \cdot y$

Case 3: $x \geq 0$, $y < 0$ similar to Case 2.

Case 4: $x < 0 \Rightarrow x = -(2^8 - X)$ $y < 0 \Rightarrow y = -(2^8 - Y)$
$Z = (2^8 - X) \cdot (2^8 - Y)$
$= 2^{16} - 2^8 \cdot X - 2^8 \cdot Y + XY$
Z rep. $x \cdot y$

These details show how a 2's-complement product can be transformed to a magnitude product. Given bit strings X and Y, if the 16-bit product $X \cdot Y$ is sought then compute

$Z = X \cdot Y$ ... 2's complement product

Corrections:

Case 1: No correction.
Case 2: Add Y into the high order byte of Z
Case 3: Add X into the high order byte of Z
Case 4: Add X and Y into the high order byte of Z

Note that the corrections 2-4 will cause the "2^{16}" to carry out of the lead bit of Z.