Two's Complement

How can negative numbers be represented using only binary 0's and 1's so that a computer can "read" them accurately?

The concept is this: Consider the binary numbers from 0000 to 1111 (i.e., 0 to 15 in base ten).

$\underline{0}001 \rightarrow \underline{0}111$ will represent the positive numbers $1 \rightarrow 7$ respectfully

and, $\underline{1}001 \rightarrow \underline{1}111$ will represent the negative numbers $-7 \rightarrow -1$, respectfully.

In a computer, numbers are stored in **registers** where there is reserved a designated number of bits for the storage of numbers in binary form. Registers come in different sizes. This handout will assume a register of size 8 for each example.

It is easy to change a negative integer in base ten into binary form using the method of two's complement.

First make sure you choose a register that is large enough to accommodate all of the bits needed to represent the number.

Step 1: Write the absolute value of the given number in binary form. Prefix this number with 0 indicate that it is positive.
Step 2: Take the complement of each bit by changing zeroes to ones and ones to zero.
Step 3: Add 1 to your result. This is the two's complement representation of the negative integer.

EXAMPLE: *Find the two's complement of –17*

Step 1: $17_{10} = 0001\ 0001_2$

Step 2: Take the complement: 1110 1110

Step 3: Add 1: 1110 1110 + 1 = 1110 1111.

Thus the two's complement for -17 is $1110\ 1111_2$. It begins on the left with a 1, therefore we know it is negative.

Now you try some:

Find the two's complement for

a. –11

b. –43

c. –123

To translate a number in binary back to base ten, the steps are reversed:

Step 1: Subtract 1: $\therefore$ 1110 1111 – 1 = 1110 1110

Step 2: Take the complement of the complement: 0001 0001

Step 3: Change from base 2 back to base 10 $\therefore$ 16 + 1 = 17

Step 4: Rewrite this as a negative integer: –17

This suggests a new way to subtract in binary due to the fact that subtraction is defined in the following manner:

X – Y = X + (-Y)

EXAMPLE 1: *Subtract 17 from 23, as a computer would, using binary code.*

Given a register of size 6, 23 – 17 = 23 + (-17) becomes

0001 0111 + 1110 1111 = 10000 0110. (Verify both the binary form of 23 and the addition.) Since this result has 9 bits, which is too large for the register chosen, the leftmost bit is truncated, resulting in the binary representation of the *positive* (it starts with a 0) integer 00000110. When this is changed to a decimal number, note that 4 + 2 = 6 which is the answer expected.

Note that a register of size eight can only represent decimal integers between $-2^{(8-1)}$ and $+2^{(8-1)}$ and, in general, a register of size *n* will be able to represent decimal integers between $-2^{(n-1)}$ and $+2^{(n-1)}$

EXAMPLE 2: *Subtract 29 from 23, as a computer would, using binary code.*

Again we use a register of size 8, so that 23 – 29 = 23 + (-29) becomes

0001 0111 + 1110 0011 = 1111 1010. (Verify both the binary form of –29 and the addition.) Note that no truncation of the leftmost bit is necessary here. The result is the *negative* (it starts with a 1) integer 1111 1010. This needs to be "translated" to change it back to a decimal (see the steps on how to do this in the box above). Hence, going backwards, 1111 1010 – 1 = 1111 1001. The complement of which is 0000 0110 which is 6 in decimal. Negating this we get -6 as expected.

Now you try some:

Subtract each, as a computer out, using binary code using registers of size 8.

a) 26 – 15

b) –31 – 6

c) 144 – 156

d) Make up your own exercises as needed.

ANSWERS

$-11 = 1111\ 0101_2$

$-43 = 1101\ 0101_2$

$-123 = 1000\ 0101_2$

26 – 15 = 26 + (-15) = 0001 1010 + 1111 0001 = 10000 1011, and truncating the leftmost 1 to remain within a register of 8, the answer is $0000\ 1011_2$

–31 – 6 = (–31) + (–6) = 1110 0001 + 1111 1010 = 11101 1011, and truncating the leftmost 1 to remain within a register of 8, the answer is $1101\ 1011_2$

144 – 156 = 144 + (–156) = 1001 0000 + 0110 0100 = 1111 0100, which remains within the register of 8 bits (so nothing gets truncated), thus the answer is $1111\ 0100_2$.