Representing Negative Integers

Ref. Ch. 4 and 5 of Goodwin & Miller.

A data type is a set of values and a set of operations or predicates on those values.

Mathematical integers: Infinite set, operations include $+, -, \text{ negation, } \ast \text{ and } \div$.

$A - B$ is the integer $X$ for which $B + X = A$. I.e., $A - B$ is the solution to the equation $B + X = A$ with unknown variable $X$.

The negation or additive inverse of $B$ is $0 - B$, so the negation of $B$, denoted by $-B$, satisfies $B + (-B) = 0$.

Other properties of Mathematical integers imply $A - B$ and $-B$ exist no matter what $A$ and $B$ are, and the solutions to the equations $B + X = A$ and $B + X = 0$ are unique.

(It just ain’t so for computer integers!) Only a finite subset of Math. integers can be represented by computer words with a fixed number of bits.

Alternate representations:

(a) Sign–Magnitude representation.

(b) One’s complement representation.

(c) Two’s complement representation.

(d) Biased representation.

(a) Sign–Magnitude form: (Not common)

Example: (with word size $n + 1 = 8$ bits)

\[
\begin{align*}
00101000 & \quad \text{represents} \ +40 \\
10101000 & \quad \text{represents} \ -40
\end{align*}
\]

Range: For a word size of $n + 1$,

Smallest integer = $-(2^n - 1)$
Largest integer = $+(2^n - 1)$

For example, with $n+1 = 8$, the range of integers representable using sign-magnitude form is $-127$ through $+127$.

Notes:

- Little Endian bit numbering (i.e., bits numbered right to left, with least significant bit numbered zero) has the most common usage and sense.

- BIT numbering order is unimportant except for hardware builders (PHY 454 students!). It’s useful for writing specifications and doing analyses like we will do.

- (BYTE address order IS VERY IMPORTANT for software people with portability concerns and network application jobs.)

Formula for value:

Bit string : $b_n b_{n-1} \ldots b_1 b_0$

Value : $(-1)^b_n \sum_{i=0}^{n-1} b_i \ast 2^i$

Disadvantage: The integer zero has two representations:

\[
\begin{align*}
00000000 & \quad \text{represents} \ +0 \\
10000000 & \quad \text{represents} \ -0
\end{align*}
\]

(b) One’s complement form: (Not common)

- Positive integers : Usual binary representation with sign bit = 0.

- Negative integers : Complement the binary representation of the corresponding positive value. (The sign bit is also complemented.)
Example 1:
0 0 0 0 1 1 1 1 -- represents +15
1 1 1 1 0 0 0 0 -- represents -15

Example 2: What decimal value does the 1’s complement binary number 10010010 represent?
Solution: When we complement all the bits, we get 01101101, which represents +109 decimal. So, the given integer = -109.

Range: For a word size of \( n + 1 \),
- Smallest integer = \(- (2^n - 1)\)
- Largest integer = \(+ (2^n - 1)\)

For example, with \( n+1 = 8 \), the range of integers representable using 1’s complement form is -127 through +127.

Formula for value:
- Bit string: \( b_n b_{n-1} \ldots b_1 b_0 \)
- Value: \( \sum_{i=0}^{n-1} b_i \cdot 2^i - b_n (2^n - 1) \)

Example 3: Find the 2’s complement representation of -15 using 8 bits.
+15: 0 0 0 0 1 1 1 1
1’s compl.: 1 1 1 1 0 0 0 0
Add 1:

\[
\begin{array}{c}
\text{Sum} \\
11110001 \\
\text{Carry} \\
0 \\
1 \\
1 \\
0 \\
1 \\
\end{array}
\]

Example 4: Find the 2’s complement representation of -64 using 8 bits.
+64: 0 1 0 0 0 0 0 0
1’s compl.: 1 0 1 1 1 1 1 1
Add 1:

\[
\begin{array}{c}
\text{Sum} \\
11111111 \\
\text{Carries} \\
11111111 \\
11000000 \\
\end{array}
\]

Disadvantage: The integer zero has two representations:
- 00000000 -- represents +0
- 11111111 -- represents -0

(c) Two’s complement form: (Most common)
- Positive integers: Usual binary representation, with sign bit = 0.
- Negative integers: Take 1’s complement and then add 1.

Recall: Binary addition table.

\[
\begin{array}{cccc}
\text{Inputs} & \text{Sum} & \text{Carry} \\
\hline
00 & 0 & 0 & 0 \\
01 & 1 & 0 & 0 \\
10 & 1 & 0 & 0 \\
11 & 0 & 0 & 1 \\
\end{array}
\]

Note: In 2’s complement arithmetic, the carry out of the sign bit is called the end around carry. It should be ignored. (It does not indicate overflow.)

Amazing Facts:
1. When the computer does the same bitwise calculations on any wordfull of bits and its 2’s complement as it does for addition of unsigned binary representations of non-neg. integers, the result is always 000000000000.
2. Why? \( X + (1\text{-s complement of } X) == 11111111\ldots1111\).
Add another 1, ignore the carry, and you get 0000000000000000.
3. Computer hardware ADD instructions do the unsigned binary addition calculation, so the same ADD instruction does the right job for both unsigned and 2-s complement represented integers.
4. Unsigned addition overflow if and only if carry = 1.
2-s complement addition overflow iff carry into sign bit \( \neq \) carry out of sign bit.
Another method of computing 2’s complement:

1. Start with the binary representation of the positive value.
2. Copy bits from right to left, until the first 1 has been copied.
3. Complement every bit thereafter.

**Example 5:** Find the 2’s complement representation of $-64$ using 8 bits.

$+64: \ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0$

2’s complement representation for $-64$:

$$
1\ 1\ 0\ 0\ 0\ 0\ 0\ 0
$$

(First 1 copied)

**Example 6:** What decimal value does the 2’s complement binary number 11111111 represent?

**Solution:** When we take the 2’s complement of the given number, we get 00000001, which represents $+1$ decimal. So, the given integer $= -1$ decimal.

**Example 7:** What is the 2’s complement of 00000000?

**Solution:**

Given: $0\ 0\ 0\ 0\ 0\ 0\ 0\ 0$

1’s compl.: $1\ 1\ 1\ 1\ 1\ 1\ 1\ 1$

Add 1:

$$
1\ 1\ 1\ 1\ 1\ 1\ 1\ 1
$$

---

$$
0\ 0\ 0\ 0\ 0\ 0\ 0\ 0
$$

---

1 --&gt; Carry out of sign bit

(should be ignored)

So, zero has a unique representation in 2’s complement form.

**Range:** For a word size of $n + 1$,

Smallest integer $= -2^n$

Largest integer $= +2^n - 1$

For example, with $n+1 = 8$, the range of integers representable using 2’s complement form is $-128$ through $+127$. (The range is asymmetric.)

**Formula for value:**

Bit string: $b_n b_{n-1} \ldots b_1 b_0$

Value: $\sum_{i=0}^{n-1} b_i \times 2^i - b_n \times 2^n$

In other words, 2’s complement representation is different from unsigned binary only in that the value contributed by the high order bit $b_n = 1$ is $-2^n$ instead of positive $2^n$. (Remember, $n = \# \ of \ bits - 1$.)

A minor disadvantage: Because of the asymmetry of the range, we cannot take the 2’s complement of the smallest (i.e., most negative) value.

**Example:** The 2’s complement representation for $-128$ decimal using 8 bits is 10000000. (You can verify that this represents $-128$ decimal using the formula given above.)

When we take the 2’s complement of 10000000, we get

10000000

itself. This cannot be correct.

The result is incorrect because the expected result, namely $+128$, cannot be represented using 8 bits (including the sign bit).

(d) Biased form: (Used only in special situations)

**Basic idea:**

- No sign bit (i.e., the resulting representation is unsigned).
• A suitable positive integer $B$ is chosen as the bias.
• integer $i$ (positive or negative) is represented by the unsigned binary representation of the value $B + i$.

**Example:** Consider 8-bit representations with bias $= 127$.

(a) Biased-127 representation of the decimal value $+9$ is the 8-bit unsigned representation of the integer $127 + 9 = 136$, namely $10001000$.

(b) Biased-127 representation of the decimal value $-21$ is the 8-bit unsigned representation of the integer $127 - 21 = 106$, namely $01101010$.

**Range:** For a word size of $n + 1$ and bias $B$

- Smallest integer $= -B$
- Largest integer $= +(2^{n+1} - 1 - B)$

For example, with $n + 1 = 8$ and bias $B = 127$, the range of integers representable using Biased-127 form is $-127$ through $+128$. (The range is asymmetric.)

**Formula for value:**

Bit string : $b_nb_{n-1} \ldots b_1b_0$

Bias : $B$

Value : $\sum_{i=0}^{n} b_i \times 2^i - B$

**Notes:**

• Used to store the exponents in the IEEE Floating Point Standard representation (to be discussed).

• For a word size of $n + 1$ bits, the normally chosen bias values are either $2^n$ or $2^n - 1$. (This is to keep the range of values reasonably symmetric around zero.)