CSCI 220: Computer Architecture I  
Instructor: Pranava K. Jha

Representation of Signed Integers

There are three important methods of representing signed integers in digital computers:

1. Sign/magnitude method
2. One’s complement method, and
3. Two’s complement method.

Assuming that a total of \( n \) bits is reserved for storing integers, the range of numbers that can be represented in each of the three methods is as follows:

<table>
<thead>
<tr>
<th>Method</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sign/magnitude method</td>
<td>(- (2^{n-1} - 1)) to (+ (2^{n-1} - 1))</td>
</tr>
<tr>
<td>One’s complement method</td>
<td>(- (2^{n-1} - 1)) to (+ (2^{n-1} - 1))</td>
</tr>
<tr>
<td>Two’s complement method</td>
<td>(- 2^{n-1}) to (+ (2^{n-1} - 1))</td>
</tr>
</tbody>
</table>

**Sign/magnitude method:** Given an integer \( r \) in the range of \(- (2^{n-1} - 1)\) to \(+ (2^{n-1} - 1)\), its \( n \)-bit representation in this system is as follows:

- Represent the magnitude of \( r \) using \( n-1 \) bits.
- If \( r \) is positive, then put a 0 at the leftmost bit position.
- If \( r \) is negative, then put a 1 at the leftmost bit position.

Note that (i) if \( r \) is positive, then its representation in this method is the usual binary representation using \( n \) bits, and (ii) if \( r \) is negative, then its representation in this method is the binary representation of \( 2^{n-1} + \lfloor r \rfloor \) using \( n \) bits.

It is easy to see that \(-0\) and \(+0\) (that are the same quantity) have different representations in this method.

**One’s complement method:** Given a number \( r \) in the range of \(- (2^{n-1} - 1)\) to \(+ (2^{n-1} - 1)\), its \( n \)-bit representation in this system is as follows:

- Represent the magnitude of \( r \) using \( n \) bits.
- If \( r \) is positive, then the string obtained at Step 1 is the correct representation of \( r \).
- If \( r \) is negative, then obtain a bit-wise complementation of the string obtained at Step 1.
Note that (i) if $r$ is positive, then its representation in this method is its usual binary representation using $n$ bits, and (ii) if $r$ is negative, then its representation in this method is the binary representation of $2^n - 1 - |r|$ using $n$ bits.

In the one’s complement method also, $-0$ and $+0$ have different representations.

**Two’s complement method:** Given a number $r$ in the range of $-2^{n-1}$ to $+(2^{n-1} - 1)$, its $n$-bit representation in this system is as follows:

- Represent the magnitude of $r$ using $n$ bits.
- If $r$ is positive, then the string obtained at Step 1 is the correct representation of $r$.
- If $r$ is negative, then obtain a bit-wise complementation of the string obtained at Step 1, and then add a 1 to the resulting string.

Note that (i) if $r$ is positive, then its representation in this method is its usual binary representation using $n$ bits, and (ii) if $r$ is negative, then its representation in this method is the binary representation of $2^n - |r|$ using $n$ bits.

It is remarkable to note that 0 has a unique representation in the two’s complement method. In fact, this method has several other advantages. As a result, this is the most widely used method of representing signed integers in digital computers.

**Remark:** Each of the three methods has the characteristic that (i) If the given number is positive, then the leftmost bit of its representation is 0, and (ii) If the given number is negative, then the leftmost bit of its representation is 1.

**Addition and subtraction of numbers**

**Sign/magnitude method:** Adding two numbers that are both positive or both negative is straightforward. Simply perform the addition of the magnitudes, and assign the same sign as that of the original operands. When the signs of the two operands are different, the task becomes complex. In this case, smaller magnitude is subtracted from the larger. The sign is the same as that of the number with the larger magnitude. This is what makes arithmetic operations in this method so cumbersome: Any adder circuit must also include a subtractor and a comparator.

**One’s complement method:** Operation performed is simple addition of the two operands, both of which are assumed to be in the one’s complement form. If there is a carry out of the high-order bit position, then the carry bit is added to the result of the sum. The final result is the answer in one’s complement form.

**Two’s complement method:** Operation performed is simple addition of the two operands, both of which are assumed to be in the two’s complement form. Any carry out of the high-order bit position is simply discarded. The final result is the answer in two’s complement form.
Q. Using four bits, decimal +0 is denoted by binary 0000 in one’s-complement system. Determine what signed integers are represented by the remaining four-bit binary strings in this system. Do a similar exercise with respect to two’s-complement system.
Signed integers in one’s-complement system (using four bits)

Signed integers in two’s-complement system (using four bits)
Q. Perform the following arithmetic operations in binary using two’s-complement representations for negative numbers:

(a) $+42 + (-18)$.

(b) $(-42) - (-18)$.

The given (decimal) numbers are such that seven bits are necessary for correct addition/subtraction. The following table is relevant.

<table>
<thead>
<tr>
<th>Decimal integer</th>
<th>2’s complement representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>+42</td>
<td>0101010</td>
</tr>
<tr>
<td>−18</td>
<td>1101110</td>
</tr>
<tr>
<td>−42</td>
<td>1010110</td>
</tr>
<tr>
<td>+18</td>
<td>0010010</td>
</tr>
</tbody>
</table>

(a) $+42 + (-18)$

\[
\begin{array}{c}
0101010 \\
\underline{+1101110} \\
10011000
\end{array}
\]

Discard the carry-out of 1. Result: 0011000.

(b) $-42 - (-18) = -42 + 18$

\[
\begin{array}{c}
1010110 \\
\underline{+0010010} \\
1101000
\end{array}
\]

There is no carry-out. Result is correct in 2’s complement form.