Module 1 - Lesson 03: 2's Complement and Arithmetic Operations

Complement Math

Unfortunately, computers do not have the capabilities to subtract binary numbers. Therefore, something like:

$x = 5 – 2$ becomes $x = 5 + (-2)$

The **TENS** complement **X**, of an n-digit number **N** is **X** = 10^n – **N**

- ➢ TENS complement of 8 is 2 $(10^1 – 8 = 2)$
- ➢ TENS complement of 90 is 10 $(10^2 – 90 = 10)$
- ➢ TENS complement of 350 is 650 $(10^3 – 350 = 650)$

The **NINES** complement is determined by subtracting each digit in the decimal number from the decimal digit 9.

Therefore:

NINES complement = **TENS** complement – 1 _OR_ **TENS** complement = **NINES** complement + 1

Examples:

The **TENS** complement of$(7)_{10}$, $(319)_{10}$, and $(14827)_{10}$:

$$\begin{array}{r} 10 \\ \underline{-\ 7} \\ 3 \end{array} \qquad \begin{array}{r} 1000 \\ \underline{-\ 319} \\ 681 \end{array} \qquad \begin{array}{r} 100000 \\ \underline{-\ 14827} \\ 85173 \end{array}$$

The **NINES** complement of$(7)_{10}$, $(319)_{10}$, and $(14827)_{10}$:

$$\begin{array}{r} 9 \\ \underline{-\ 7} \\ 2 \end{array} \qquad \begin{array}{r} 999 \\ \underline{-\ 319} \\ 680 \end{array} \qquad \begin{array}{r} 99999 \\ \underline{-\ 14827} \\ 85172 \end{array}$$

Consider the following:

$$\begin{array}{rl} 23456 & \textit{minuend} \\ \underline{-\ 14827} & \textit{subtrahend} \\ 8629 & \end{array}$$

We can also achieve the same result by adding the **TENS** complement of the subtrahend and then subtracting 10^n:

$$23456 + (100000 - 14827) - 100000 = 8629$$

The advantage of using this method is that you never need to borrow a place value. **Complement addition replaces binary subtraction in computer arithmetic**.

Binary Complements

The **TWOS** complement, **X**, of an n-bit binary number **N** is $\mathbf{X} = 2^n - \mathbf{N}$

Example: If **N** = 101010, the **TWOS** complement is $2^6 - 101010 = 1000000 - 101010 = 010110$

The **ONES** complement is determined by subtracting each bit in the binary number from 1. Therefore the **ONES** complement of 101010 is:

$$\begin{array}{r} 111111 \\ \underline{-\ 101010} \\ 010101 \end{array}$$

Notice that the **ONES** complement can be found easily by "flipping" or "switching" each bit.

Since **TWOS** complement is equivalent to the **ONES** complement + 1, we can use the following algorithm to get the **TWOS** complement of a binary number:

1. Get the **ONES** complement by "flipping" each bit in the binary number (change all 0's to 1's and all 1's to 0's).
2. Add 1 to the **ONES** complement.

Example: Find the **TWOS** complement of 10010101.

01101010 + 1 = 01101011

Exercise 3: Find the **TWOS** complement of the following binary numbers:

110111011 0110001000

Negative Binary Numbers

How are negative numbers represented and used in a computer?

We must consider the following:

1. How do we represent the negative sign?
2. When performing calculations with negative numbers, extra care is required with regards to bit field length (how many bits represent each number)
3. Bit field extension may be required (performing an operation that results in the bit field length being too small for the answer)

Representing the Negative Sign in Binary

a) **We can dedicate the Most Significant Bit (left-most bit) to indicate the sign of the number**

   The bits to the left of the most significant bit indicate the number's ***magnitude***. This is known as ***sign-magnitude***. This method makes it easy to identify whether a number is negative, BUT there are two representations of zero and addition does not always work properly.

b) **Use the ONES complement of a number to represent its negative value**

   This method also gives two representations of the number zero.

c) **** Use the TWOS complement of a number to represent its negative value ****

   This is the most common method to represent negative numbers.

   The most significant bit (left-most bit):

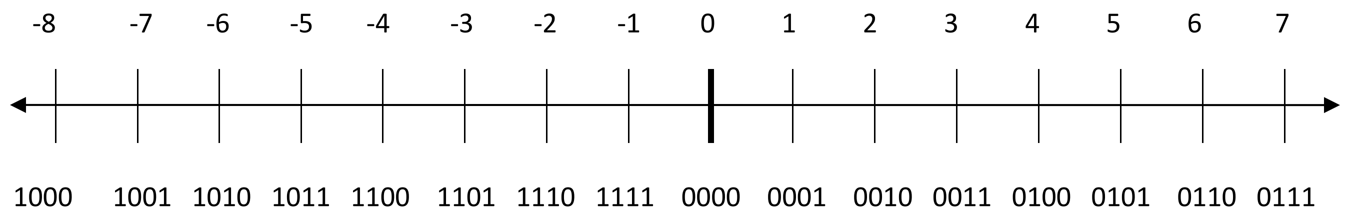
   - **ALWAYS 1 if the number is NEGATIVE**
   - **ALWAYS 0 when a number is NON-NEGATIVE**

1. Bit Field Length

When working with negative binary numbers it is imperative to pick a bit field length and use that same length throughout the calculation. Bit field length ***must*** be the same for both numbers in the calculation.

Example:

We decide to use a bit field length of 4 to represent the numbers from -8 (**TWOS** complement of 8 is 1000) to 7 (0111). The number line for a **4-bit** field looks like the following:



When finding the **TWOS** complement of a number, we use the *entire* bit field length to calculate the complement.

TWOS complement of $(3)_{dec}$ with 4-bit field length is:

$$\begin{array}{rl} 0011 & \\ 1100 & [\ \textbf{ONES}\text{ complement }] \\ \underline{+\quad 1} & \\ 1101 & [\ \textbf{TWOS}\text{ complement } \textbf{or } (-3)_{dec}\ ] \end{array}$$

Working with fixed bit lengths is fine when **adding two numbers of opposite signs**. The resulting sum is **ALWAYS** correct within the fixed bit length.

Example: $(1)_{dec} + (-1)_{dec}$

$$\begin{array}{r} 0001 \\ \underline{+1111} \\ 10000 \end{array}$$

The answer is 10000**bin**. Since we are using a fixed bit length of 4, we just **discard** the left-most bit to get an answer of 0000 or $(0)_{dec}$.

****NOTE** The answer is NOT $(16)_{dec}$!**

Subtracting in the binary number system:

To subtract a binary number, we "add the opposite" (the **TWOS** complement).

1. Convert the ***subtrahend*** to its TWOS complement.
2. Add the **TWOS** complement to the ***minuend***.
3. If the result extends beyond the bit field length of the minuend and subtrahend, the extra bit can be discarded.

Examples: Evaluate 1001 – 0011 Evaluate $(25)_{\text{dec}} - (37)_{\text{dec}}$

Evaluate $(-3)_{\text{dec}} - (4)_{\text{dec}}$

Exercise 4: Evaluate the following:

$(67)_{10} - (32)_{10}$ $(-9)_{10} - (5)_{10}$ 011001 – 010010

2. Bit Field Extension

When numbers are the same sign, using a fixed bit length may cause issues. It may be that an extra bit is required for the correct answer. This is called ***overflow***.

Examples:

$$\begin{array}{r} 0\ 1\ 0\ 1 \\ +\ 0\ 1\ 1\ 1 \\ \hline \boxed{1}\ 1\ 0\ 0 \end{array} \quad \begin{array}{c} = \\ = \\ \neq \end{array} \quad \begin{array}{r} 5 \\ +\ 7 \\ \hline 12 \end{array} \qquad\qquad \begin{array}{r} 1\ 1\ 0\ 0 \\ +\ 1\ 0\ 0\ 1 \\ \hline \boxed{0}\ 1\ 0\ 1 \end{array} \quad \begin{array}{c} = \\ = \\ \neq \end{array} \quad \begin{array}{r} -\ 4 \\ -\ 7 \\ \hline -\ 11 \end{array}$$

Note that the sign of the result is ***different*** than the sign of both addends!

General Rules for Bit Field Extension:

1. If two numbers have different signs, their sum will never overflow.
2. If the numbers have the same sign, they might overflow.
3. To deal with overflow we extend the bit field length of the numbers **BEFORE** the calculation.

If when calculating overflow is occurring, "promote" values to larger bit field lengths. This process is known as ***sign extension***.

Sign Extension:

- Duplicate the value of the most significant bit to the left of the existing bit field
- The sign of the number is retained
- The value of the number does not change
- **All values MUST be promoted to the same bit field length BEFORE any calculations**.

Example: Evaluate $(-4)_{dec} - (7)_{dec}$

Exercise 5:

Subtract the following binary numbers using complement math. Watch out for negative results!

00110 – 01111

01000 – 01110

Exercises

1. Subtract the following binary numbers using the complement method and calculate the decimal equivalent (watch out for negative results!)

a) $00101 - 01111$ b) $01011 - 01110$

2. Add the following pairs of ***signed*** binary numbers:

a) $010111 + 01011$ b) $01101 + 01100$

3. Steve bought the following from the Sheridan Department Store:
   Shoes for $111011**bin**
   Jacket for $110111**bin**
   Gloves for $1110**bin**
   Socks for $111**bin**

   Calculate the total bill in both binary dollars and decimal dollars.

4. Subtract to negative using the complement method: 0 1 1 1 1 1 – 1 0 0 0 0 0