## Number Systems

Any number of the form:

\[(IJKL)_b\]

can be evaluated by the following equation:

\[L \times B^0 + K \times B^1 + J \times B^2 + I \times B^3\]

## Base Conversion

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
<th>Octal</th>
<th>Hexadecimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
<td>00</td>
<td>0x00</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
<td>01</td>
<td>0x01</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
<td>02</td>
<td>0x02</td>
</tr>
<tr>
<td>3</td>
<td>0011</td>
<td>03</td>
<td>0x03</td>
</tr>
<tr>
<td>4</td>
<td>0100</td>
<td>04</td>
<td>0x04</td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
<td>05</td>
<td>0x05</td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
<td>06</td>
<td>0x06</td>
</tr>
<tr>
<td>7</td>
<td>0111</td>
<td>07</td>
<td>0x07</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>10</td>
<td>0x08</td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
<td>11</td>
<td>0x09</td>
</tr>
<tr>
<td>10</td>
<td>1010</td>
<td>12</td>
<td>0x0A</td>
</tr>
<tr>
<td>11</td>
<td>1011</td>
<td>13</td>
<td>0x0B</td>
</tr>
<tr>
<td>12</td>
<td>1100</td>
<td>14</td>
<td>0x0C</td>
</tr>
<tr>
<td>13</td>
<td>1101</td>
<td>15</td>
<td>0x0D</td>
</tr>
<tr>
<td>14</td>
<td>1110</td>
<td>16</td>
<td>0x0E</td>
</tr>
<tr>
<td>15</td>
<td>1111</td>
<td>17</td>
<td>0x0F</td>
</tr>
<tr>
<td>16</td>
<td>10000</td>
<td>20</td>
<td>0x10</td>
</tr>
</tbody>
</table>
Binary to Decimal Conversion

Example – Convert \((1101)_2\) to Decimal

\[
1101 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\
= 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 \\
= 8 + 4 + 0 + 1 \\
= (13)_{10}
\]

Hexadecimal to Decimal Conversion

Example – Convert \((E42)_{16}\) to Decimal

\[
E42 = E \times 16^2 + 4 \times 16^1 + 2 \times 16^0 \\
= 14 \times 256 + 4 \times 16 + 2 \times 1 \\
= 3584 + 64 + 2 \\
= 3650
\]
Decimal to Binary Conversion

Example – Convert \((25)_{10}\) to Binary

<table>
<thead>
<tr>
<th>Quotient</th>
<th>Remainder</th>
</tr>
</thead>
<tbody>
<tr>
<td>(25)</td>
<td>(12) + 1</td>
</tr>
<tr>
<td>(12)</td>
<td>(6) + 0</td>
</tr>
<tr>
<td>(6)</td>
<td>(3) + 0</td>
</tr>
<tr>
<td>(3)</td>
<td>(1) + 1</td>
</tr>
<tr>
<td>(1)</td>
<td>(0) + 1</td>
</tr>
</tbody>
</table>

\[11001\]

Decimal to "Hex" Conversion

Example – Convert \((79)_{10}\) to Hexadecimal

<table>
<thead>
<tr>
<th>Quotient</th>
<th>Remainder</th>
</tr>
</thead>
<tbody>
<tr>
<td>(79)</td>
<td>(4) + 15 (\Rightarrow F)</td>
</tr>
<tr>
<td>(4)</td>
<td>(0) + 4</td>
</tr>
</tbody>
</table>

\[(4F)_{16}\]
**Binary to "Hex" Conversion**

*Example - Convert \(10101110\) to Hex*

\[
\begin{align*}
1010 & \quad 1110 \\
(10_{10} & \quad 14_{10}) \\
\downarrow & \quad \downarrow \\
(A & \quad E)_{16}
\end{align*}
\]

**"Hex" to Binary Conversion**

*Example - Convert \((3B7)_{16}\) to Binary*

\[
\begin{align*}
3 & \quad B & \quad 7 \\
\overline{0011} & \quad \overline{1011} & \quad \overline{0111}
\end{align*}
\]

**Fractional Conversion**

*Example - Convert \(0.1011\) binary to decimal*

\[
\begin{align*}
0.1011 &= 1 \times 2^{-1} = 0.5 \\
&= 0 \times 2^{-2} = 0.0 \\
&= 1 \times 2^{-3} = 0.125 \\
&= 1 \times 2^{-4} = 0.0625 \\
&= 1 \times 2^{-6} = 0.03125 \\
& \quad 0.71875 \text{ decimal}
\end{align*}
\]
Fractional Conversion

Example – Convert 0.4E1 Hex to decimal

$$0.4E_1 = 4 \times 16^{-1} = 0.25$$
$$E \times 16^{-2} = 14 \times 16^{-2} = 0.0546875$$
$$1 \times 16^{-3} = 0.0002441$$
$$= 0.3049316 \text{ decimal}$$

Fractional Conversion

Example – Convert 0.642 Decimal to Binary

$$0.642 \times 2 = 1.284 \quad 1$$
$$0.284 \times 2 = 0.568 \quad 0$$
$$0.568 \times 2 = 1.136 \quad 1$$
$$0.136 \times 2 = 0.272 \quad 0$$
$$0.272 \times 2 = 0.544 \quad 0$$
$$0.544 \times 2 = 1.088 \quad 1$$
$$\vdots$$
Binary Addition Examples

\[
\begin{array}{ccc}
\text{+} & 00101110 & 01000111 \\
\text{+} & 00011101 & 01101101 \\
\text{+} & 01001011 & 10110100 \\
\end{array}
\]

\[
10110110 \\
+ 11011100 \\
\underline{110010010}
\]

Sign-Magnitude Notation

We represent negative numbers by using a "sign-plus-magnitude" notation. Binary values can also be represented this way, by appending a 0 for positive, 1 for negative.

\[
\begin{array}{c}
+57 \Rightarrow 001111001 \\
-57 \Rightarrow 101111001 \\
\end{array}
\]

Sign-magnitude is clumsy for the computer!
Rules for Sign-Magnitude Arithmetic

1. If both numbers are positive, then add them, and prepend a 0 (for positive).

2. If both numbers are negative, then add the absolute values and then prepend a 1 (for negative).

3. If one number is positive and the other negative, then determine which has the larger absolute value. Subtract the larger absolute value from the smaller one. Then if the larger number was positive, prepend a 0, otherwise prepend a 1 (for negative).

WHEW!

One’s Complement Representation

A negative binary number can be represented by taking the positive representation of the number and changing all the 0’s to 1’s and all the 1’s to 0’s.

Example 1 - Express -5₁₀ as an 8-bit 1’s complement number
-5₁₀ = -(00000101) = 1111010

Example 2 - Express -13₁₀ as an 8-bit 1’s complement number
-13₁₀ = -(00001101) = 11110010

Example 3 - Express +14₁₀ as an 8-bit 1’s complement number
+14₁₀ = +(00001110) = 00001110

Example 4 - Express -0₁₀ as an 8-bit 1’s complement number
-0₁₀ = -(00000000) = 11111111!
One's Complement Values

What values do the following bit patterns represent in the one’s complement system?

Example 1 – 00101101
    00101101 ⇒ positive number, or +45

Example 2 – 11101101
    11101101 ⇒ negative number, or −(00010110), or −18

Example 3 – 11111110
    11111110 ⇒ negative number, or −(00000010), or −1

Finding the One's Complement

To “find the one’s complement” of a number means to find the negative of a number:

Example 1 – Find the one’s complement of 00101101
    −(00101101) = 11010010

Example 2 – Find the one’s complement of 11101101
    −(11101101) = 00010010
One's Complement Arithmetic

To add:

1. Add the two numbers, including sign bits.
2. If a carry out of the sign bit occurs, then add 1 to the sum ("end-around carry")

To subtract:

1. Take the 1's complement of the second number and then add.

One's Complement Examples

Perform the following 1's complement arithmetic, and show the decimal equivalents of each problem

\[
\begin{align*}
0010 & \ 1101 \Rightarrow \ (+45) \\
+ & \ 0001 \ 1011 \Rightarrow \ (+27) \\
\hline
0100 & \ 1000 \Rightarrow \ (+72)
\end{align*}
\]

\[
\begin{align*}
0110 & \ 1101 \Rightarrow \ (+109) \\
+ & \ 1111 \ 0010 \Rightarrow \ (-13) \\
\hline
0101 & \ 1111 \ \Rightarrow \ (+96)
\end{align*}
\]
One's Complement Examples

Perform the following 1's complement arithmetic, and show the decimal equivalents of each problem.

\[
\begin{align*}
0010\ 1001 &\rightarrow (+41) \rightarrow 0010\ 1001 \\
-0001\ 0111 &\rightarrow (-23) \rightarrow +1110\ 1000 \\
&\quad \quad +1 \\
&\quad \quad 0001\ 0010 \quad (+18)
\end{align*}
\]

\[
\begin{align*}
1110\ 1010 &\rightarrow (-21) \rightarrow 1110\ 1010 \\
-1101\ 1000 &\rightarrow (-39) \rightarrow +0010\ 0111 \\
&\quad \quad +1 \\
&\quad \quad 0001\ 0010 \quad (+18)
\end{align*}
\]

Two's Complement Representation

A negative binary number can be represented by taking the one's complement of the number and then adding 1. This is called the two's complement representation.

Example 1: Express \(-5\) as an 8-bit 1's complement number
\[
-5_{10} = -(00000101) = 11111010 \\
\quad +1 \\
\quad 11111011
\]

Example 2: Express \(-13\) as an 8-bit 1's complement number
\[
-13_{10} = -(00001101) = 11110010 \\
\quad \quad \Rightarrow \\
\quad +1 \\
\quad 11110011
\]

Example 3: Express \(-0\) as an 8-bit 1's complement number
\[
-0_{10} = -(00000000) = 11111111 \\
\quad +1 \\
\quad 00000000
\]
Two’s Complement Values

What values do the following bit patterns represent in the two’s complement system?

Example 1 – 00101101
  00101101 ⇒ positive number, or +45

Example 2 – 11101101
  11101101 ⇒ negative number, so -(00010010), or −19

Example 3 – 11111111
  11111111 ⇒ negative number, so -(00000000), or −1

Finding the Two’s Complement

To “find the two’s complement” of a number means to find the negative of a number:

Example 1 – Find the two’s complement of 00101101
  −(00101101) = 11010010 + 1 = 11010011

Example 2 – Find the two’s complement of 1101101
  −(1101101) = 00100100 + 1 = 0010011
Two’s Complement Arithmetic

To add:
1. Add the two numbers, including sign bits. Ignore any carry out of the sign bit.

To subtract:
1. Take the 2’s complement of the second number and then add.

Two’s Complement Examples

Perform the following 2’s complement arithmetic, and show the decimal equivalents of each problem

\[ \begin{align*}
0010 \ 1101 & \Rightarrow \ (+55) \\
+ \ 0001 \ 1011 & \Rightarrow \ (+17) \\
\hline
0100 \ 1000 & \Rightarrow \ (+72)
\end{align*} \]

\[ \begin{align*}
0110 \ 1101 & \Rightarrow \ (+109) \\
+ \ 1111 \ 0010 & \Rightarrow \ (+00011110) \Rightarrow \ (-14) \\
0101 \ 1111 & \Rightarrow \ (+95)
\end{align*} \]
Two's Complement Examples

Perform the following 2's complement arithmetic, and show the decimal equivalents of each problem

\[
\begin{align*}
0010 \ 1001 & \Rightarrow ( +41 ) & \Rightarrow & 0010 \ 1001 \\
-0001 \ 0111 & \Rightarrow -( +23 ) & \Rightarrow & +1110 \ 1001 \\
& & & 0001 \ 0010 \ (+18)
\end{align*}
\]

\[
\begin{align*}
1110 \ 1010 & \Rightarrow ( -22 ) & \Rightarrow & 1110 \ 1010 \\
-1101 \ 1000 & \Rightarrow -( -40 ) & \Rightarrow & +0010 \ 1000 \\
& & & 0001 \ 0010 \ (+18)
\end{align*}
\]

Note: throw away the carry out

---

Overflow

In a real computer, a finite number of bits are used in calculations.

Whenever an attempt is made to represent or compute a number that is too large to be represented in the number of bits available, overflow occurs.

This is analogous to trying to represent a number larger than 1000 in only three decimal digits.

Example - Represent the decimal value 343 as an unsigned 8-bit value

ANS - The unsigned representation requires 9 bits, 101010111, so cannot be represented as an 8-bit number.
Overflow in 1’s and 2’s Complement

Since the MSB is reserved for the sign, one less bit is available for magnitudes.

Example – Represent the value +209 decimal as an 8-bit 2’s complement number.

Ans – The binary representation for +209 is 1101 0001. However, this pattern represents a negative number, since the MSB or sign bit is a 1. This value cannot be representable as an 8-bit 2’s complement number.

Maximum/Minimum Values

For unsigned integers:

\[ 0 - 2^{n-1} - 1, \text{ where } n \text{ is number of bits} \]

For 8 bits: 0 – 255
For 16 bits: 0 – 65535
For 32 bits: 0 – 4294967295

For 2’s complement (ints):

\[ -2^{n-1} - 2^{n-1} - 1, \text{ where } n \text{ is number of bits:} \]

For 8 bits: -128 - +127
For 16 bits: -32768 - +32767
For 32 bits: -2147483648 - +2147483647
Detecting Overflow

For unsigned integers:

Overflow occurs when the value doesn’t fit within the
number of bits allotted the number.

For 1’s/2’s complement (two ways):

1. Determine the answer. Overflow if it is larger/smaller than limits.

2. Check carries into and out of sign bit – if neither or both occur,
then OK. If only one occurs, overflow