Number Systems

A number of the form:

\[(IJKL)_{B}\]

can be evaluated in the following way:

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

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>00000</td>
<td>00</td>
<td>0x00</td>
</tr>
<tr>
<td>1</td>
<td>00001</td>
<td>01</td>
<td>0x01</td>
</tr>
<tr>
<td>2</td>
<td>00010</td>
<td>02</td>
<td>0x02</td>
</tr>
<tr>
<td>3</td>
<td>00011</td>
<td>03</td>
<td>0x03</td>
</tr>
<tr>
<td>4</td>
<td>00100</td>
<td>04</td>
<td>0x04</td>
</tr>
<tr>
<td>5</td>
<td>00101</td>
<td>05</td>
<td>0x05</td>
</tr>
<tr>
<td>6</td>
<td>00110</td>
<td>06</td>
<td>0x06</td>
</tr>
<tr>
<td>7</td>
<td>00111</td>
<td>07</td>
<td>0x07</td>
</tr>
<tr>
<td>8</td>
<td>01000</td>
<td>10</td>
<td>0x08</td>
</tr>
<tr>
<td>9</td>
<td>01001</td>
<td>11</td>
<td>0x09</td>
</tr>
<tr>
<td>10</td>
<td>01010</td>
<td>12</td>
<td>0x0A</td>
</tr>
<tr>
<td>11</td>
<td>01011</td>
<td>13</td>
<td>0x0B</td>
</tr>
<tr>
<td>12</td>
<td>01100</td>
<td>14</td>
<td>0x0C</td>
</tr>
<tr>
<td>13</td>
<td>01101</td>
<td>15</td>
<td>0x0D</td>
</tr>
<tr>
<td>14</td>
<td>01110</td>
<td>16</td>
<td>0x0E</td>
</tr>
<tr>
<td>15</td>
<td>01111</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>(1)</td>
</tr>
<tr>
<td>(12)</td>
<td>(0)</td>
</tr>
<tr>
<td>(6)</td>
<td>(0)</td>
</tr>
<tr>
<td>(3)</td>
<td>(1)</td>
</tr>
<tr>
<td>(1)</td>
<td>(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>(15)</td>
</tr>
<tr>
<td>(4)</td>
<td>(4)</td>
</tr>
</tbody>
</table>

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

Example – Convert \(10101110_2\) to Hex

\[
\begin{array}{c}
1010 \\
1110 \\
\downarrow \\
\downarrow \\
(10_{10} \\
14_{10}) \\
\downarrow \\
\downarrow \\
(A \\
E)_{16}
\end{array}
\]

"Hex" to Binary Conversion

Example – Convert \(3B7_{16}\) to Binary

\[
\begin{array}{ccc}
3 & B & 7 \\
\downarrow & \downarrow & \downarrow \\
(0011 & 1011 & 0111)_2
\end{array}
\]

Fractional Conversion

Example – Convert 0.10111 binary to decimal

\[
\begin{align*}
0.10111 &= 1 \times 2^{-1} = 0.5 \\
&= 0 \times 2^{-2} = 0.0 \\
&= 1 \times 2^{-3} = 0.125 \\
&= 1 \times 2^{-4} = 0.0625 \\
&= 1 \times 2^{-5} = \underline{0.03125} \\
\text{0.71875 decimal}
\end{align*}
\]
Fractional Conversion

Example – Convert 0.4E1 Hex to decimal

\[ 0.4E1 = 4 \times 16^{-1} = 0.25 \]
\[ E \times 16^{-2} = 14 \times 16^{-2} = 0.0546875 \]
\[ 1 \times 16^{-3} = 0.0002441 \]
\[ = 0.3049316 \text{ decimal} \]

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 Arithmetic Examples

Addition

\[
\begin{array}{c}
01100110 \\
+ 00011101 \\
\hline
01001011
\end{array}
\quad
\begin{array}{c}
01001111 \\
+ 01101101 \\
\hline
10110100
\end{array}
\quad
\begin{array}{c}
10110110 \\
+ 11011100 \\
\hline
110010010
\end{array}
\]

Note: Requires 3 bits

Subtraction

\[
\begin{array}{c}
001011011 \\
- 00110101 \\
\hline
00010010
\end{array}
\quad
\begin{array}{c}
10110010 \\
- 00011100 \\
\hline
10010110
\end{array}
\]

Hex Arithmetic Examples

Addition

\[
\begin{array}{c}
3B4 \\
+ 6A5 \\
\hline
A59
\end{array}
\quad
\begin{array}{c}
1AF \\
+ 345 \\
\hline
4F4
\end{array}
\]

Subtraction

\[
\begin{array}{c}
3B4 \\
- 195 \\
\hline
21F
\end{array}
\quad
\begin{array}{c}
A30 \\
- 1D8 \\
\hline
2F9
\end{array}
\]

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.

+57 ⇒ 0 0 1 1 1 0 0 1

-57 ⇒ 1 0 1 1 1 0 0 1

We use sign-magnitude as our "normal" system.
But 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).

WHHEW!
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 $-5_{10}$ as an 8-bit 1's complement number

$-5_{10} = -(00000101) \Rightarrow 11111010$

Example 2 - Express $-13_{10}$ as an 8-bit 1's complement number

$-13_{10} = -(00001101) \Rightarrow 11110010$

Example 3 - Express $+14_{10}$ as an 8-bit 1's complement number

$+14_{10} = + (00001110) \Rightarrow 00001110$

Example 4 - Express $0_{10}$ as an 8-bit 1's complement number

$0_{10} = -(00000000) \Rightarrow 11111111$

One's Complement Values

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

Example 1 - 00101101

00101101 \Rightarrow positive number, or +45

Example 2 - 11101101

11101101 \Rightarrow negative number, so -(00010010), or -18

Example 3 - 11111110

11111110 \Rightarrow negative number, so -(00000001), 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

\[-(11101101) = 00010010\]

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

\[-5_{10} = -(00000101) = 11111010\]

\[+1 \quad 11111011\]

Example 2 – Express \(-13\) as an 8-bit 1’s complement number

\[-13_{10} = -(00001101) = 11110010\]

\[+1 \quad 11110011\]

Example 3 – Express \(-0\) as an 8-bit 1’s complement number

\[-0_{10} = -(00000000) = 11111111\]

\[+1 \quad 00000000\]
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 -(00010011), or -19

Example 3 - 11111111  
11111111 ⇒ negative number, so -(00000001), 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 11101101  
-(11101101) = 00010010 + 1 = 00010011
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

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

\[
\begin{align*}
0110 \ 1101 & \Rightarrow \ 0110 \ 1101 \ \ (+109) \\
+ \ 1111 \ 0010 & \Rightarrow \ +(00001110) \ = \ ( -14 ) \\
0101 \ 1111 & \Rightarrow \ ( +95 )
\end{align*}
\]
Two's Complement Examples

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

\[ 0010 \ 1001 \Rightarrow ( +41 ) \Rightarrow 0010 \ 1001 \]
\[ -0001 \ 0111 \Rightarrow - ( +23 ) \Rightarrow \underline{+}1110 \ 1001 \]
\[ 0001 \ 0010 \quad (+18) \]

\[ 1110 \ 1010 \Rightarrow ( -22 ) \Rightarrow 1110 \ 1010 \]
\[ -1101 \ 1000 \Rightarrow -( -40 ) \Rightarrow \underline{+}0010 \ 1000 \]
\[ 1 \quad 0001 \ 0010 \quad (+18) \]

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, 10101011, so it 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.

Answer - 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 represented as an 8 bit 2's complement number.

Maximum/Minimum Values

For unsigned integers:

$0 - 2^n - 1$, where $n$ is number of bits

For 8 bits: $0 - 255$
For 16 bits: $0 - 65535$ ($0 - 64K$)
For 32 bits: $0 - 4294967295$ ($0 - 4G$)

For 2's complement (ints):

$-2^{n-1} - 2^{n-1} - 1$, where $n$ is number of bits:

For 8 bits: $-128 - +127$
For 16 bits: $-32768 - +32767$ ($-32K$)
For 32 bits: $-2147483648 - +2147483547$ ($-2G$)
Detecting Overflow

For unsigned integers:

Overflow occurs when the value doesn’t fit with in 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 carry into and out of sign bit — if neither or both occur, than OK. If only one occurs, overflow