Positional Number Systems

• A number system consists of an order set of symbols (digits) with relations defined for +, -, *, /

• The radix (or base) of the number system is the total number of digits allowed in the number system.

  — Example, for the decimal number system:
    • Radix, \( r = 10 \), Digits allowed = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

• In positional number systems, a number is represented by a string of digits, where each digit position has an associated weight.

• The value of a number is the weighted sum of the digits.

• The general representation of an unsigned number \( D \) with whole and fraction portions number in a number system with radix \( r \):

\[
D_r = d_{p-1} d_{p-2} \ldots d_1 d_0 d_{-1} d_{-2} \ldots D_{-n}
\]

• The number above has \( p \) digits to the left of the radix point and \( n \) fraction digits to the right.

• A digit in position \( i \) has as associated weight \( r^i \)

• The value of the number is the sum of the digits multiplied by the associated weight \( r^i \):

\[
D = \sum_{i=-n}^{p-1} d_i \times r^i
\]
Positional Number Systems

Number: \( D_r = d_{p-1} d_{p-2} \ldots d_1 d_0 d_{-1} d_{-2} \ldots D_{-n} \)

Value: \( D = \sum_{i=-n}^{p-1} d_i \times r^i \)

- For example in the decimal number system:
  \[
  5185.68_{10} = 5 \times 10^3 + 1 \times 10^2 + 8 \times 10^1 + 5 \times 10^0 + 6 \times 10^{-1} + 8 \times 10^{-2} \\
  = 5 \times 1000 + 1 \times 100 + 8 \times 10 + 5 \times 1 + 6 \times 0.1 + 8 \times 0.01 
  \]

- For the binary number system with radix = 2, digits 0, 1
  \[
  D_2 = d_{p-1} \times 2^{p-1} \ldots d_1 \times 2^1 + d_0 \times 2^0 + d_{-1} \times 2^{-1} + d_{-2} \times 2^{-2} \ldots 
  \]

- For Example:
  \[
  10011_2 = 1 \times 16 + 0 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1 = 19_{10} \\
  \text{MSB} \quad \text{LSB} \quad \text{(least significant bit)}
  \]

  
  (most significant bit)

  \[
  101.001_2 = 1 \times 4 + 0 \times 2 + 1 \times 1 + 0 \times 0.5 + 0 \times 0.25 + 1 \times 0.125 = 5.125_{10}
  \]

  Binary Point
### Number Systems Used in Computers

<table>
<thead>
<tr>
<th>Name of Radix</th>
<th>Radix</th>
<th>Set of Digits</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimal</td>
<td>r=10</td>
<td>{0,1,2,3,4,5,6,7,8,9}</td>
<td>(255_{10})</td>
</tr>
<tr>
<td>Binary</td>
<td>r=2</td>
<td>{0,1}</td>
<td>(11111111_2)</td>
</tr>
<tr>
<td>Octal</td>
<td>r=8</td>
<td>{0,1,2,3,4,5,6,7}</td>
<td>(377_8)</td>
</tr>
<tr>
<td>Hexadecimal</td>
<td>r=16</td>
<td>{0,1,2,3,4,5,6,7,8,9,A, B, C, D, E, F}</td>
<td>(\text{FF}_{16})</td>
</tr>
</tbody>
</table>

- **Decimal**: Radix 10, Set of Digits: \(\{0,1,2,3,4,5,6,7,8,9\}\). Example: \(255_{10}\)
- **Binary**: Radix 2, Set of Digits: \(\{0,1\}\). Example: \(11111111_2\)
- **Octal**: Radix 8, Set of Digits: \(\{0,1,2,3,4,5,6,7\}\). Example: \(377_8\)
- **Hexadecimal**: Radix 16, Set of Digits: \(\{0,1,2,3,4,5,6,7,8,9,A, B, C, D, E, F\}\). Example: \(\text{FF}_{16}\)

<table>
<thead>
<tr>
<th>Decimal</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hex</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
</tr>
<tr>
<td>Binary</td>
<td>0000</td>
<td>0001</td>
<td>0010</td>
<td>0011</td>
<td>0100</td>
<td>0101</td>
<td>0110</td>
<td>0111</td>
<td>1000</td>
<td>1001</td>
<td>1010</td>
<td>1011</td>
<td>1100</td>
<td>1101</td>
<td>1110</td>
<td>1111</td>
</tr>
</tbody>
</table>
Radix-\( r \) to Decimal Conversion

- The decimal value of a number in any radix \( r \) is found by converting each digit to its radix 10 equivalent and expanding the value using radix arithmetic:

\[
D = \sum_{i=-n}^{p-1} d_i \times r^i
\]

- Examples:

\[
\begin{align*}
1101.101_2 &= 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-3} \\
&= 8 + 4 + 1 + 0.5 + 0.125 = 13.625_{10} \\
572.6_8 &= 5 \times 8^2 + 7 \times 8^1 + 2 \times 8^0 + 6 \times 8^{-1} \\
&= 320 + 56 + 16 + 0.75 = 392.75_{10} \\
2A.8_{16} &= 2 \times 16^1 + 10 \times 16^0 + 8 \times 16^{-1} \\
&= 32 + 10 + 0.5 = 42.5_{10} \\
132.3_4 &= 1 \times 4^2 + 3 \times 4^1 + 2 \times 4^0 + 3 \times 4^{-1} \\
&= 16 + 12 + 2 + 0.75 = 30.75_{10} \\
341.24_5 &= 3 \times 5^2 + 4 \times 5^1 + 1 \times 5^0 + 2 \times 5^{-1} + 4 \times 5^{-2} \\
&= 75 + 20 + 1 + 0.4 + 0.16 = 96.56_{10}
\end{align*}
\]
Decimal-to-Binary Conversion

• Separate the decimal number into whole and fraction portions.
• To convert the whole number portion to binary, use successive division by 2 until the quotient is 0. The remainders form the answer, with the first remainder as the least significant bit (LSB) and the last as the most significant bit (MSB).
• Example: Convert $179_{10}$ to binary:

  $179 / 2 = 89$ remainder 1 (LSB)
  
  $/ 2 = 44$ remainder 1
  
  $/ 2 = 22$ remainder 0
  
  $/ 2 = 11$ remainder 0
  
  $/ 2 = 5$ remainder 1
  
  $/ 2 = 2$ remainder 1
  
  $/ 2 = 1$ remainder 0
  
  $/ 2 = 0$ remainder 1 (MSB)

  $179_{10} = 10110011_2$
Decimal-to-Binary Conversion

- To convert decimal fractions to binary, repeated multiplication by 2 is used, until the fractional product is 0 (or until the desired number of binary places). The whole digits of the multiplication results produce the answer, with the first as the MSB, and the last as the LSB.
- Example: Convert $0.3125_{10}$ to binary

<table>
<thead>
<tr>
<th>Result Digit</th>
<th>.3125 × 2 = 0.625</th>
<th>0 (MSB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>.625 × 2</td>
<td>= 1.25</td>
<td>1</td>
</tr>
<tr>
<td>.25 × 2</td>
<td>= 0.50</td>
<td>0</td>
</tr>
<tr>
<td>.5 × 2</td>
<td>= 1.0</td>
<td>1 (LSB)</td>
</tr>
</tbody>
</table>

$0.3125_{10} = .0101_2$
Decimal-to-Binary Conversion

Sum-of-Weights Method

- Determine the set of binary weights whose sum is equal to the decimal number.

Examples:

\[ 9_{10} = 8 + 1 = 2^3 + 2^0 = 1001_2 \]

\[ 18_{10} = 16 + 2 = 2^4 + 2^1 = 10010_2 \]

\[ 58_{10} = 32 + 16 + 8 + 2 = 2^5 + 2^4 + 2^3 + 2^1 = 111010_2 \]

\[ 0.625_{10} = 0.5 + 0.125 = 2^{-1} + 2^{-3} = 0.101_2 \]
Decimal to Radix-r Conversion

- Separate the decimal number into whole and fraction portions.
- To convert the whole number portion to binary, use successive division by the radix (r) until the quotient is 0. The remainders form the answer, with the first remainder as the least significant digit (LSD) and the last as the most significant digit (MSD).
- To convert decimal fractions to radix-r, repeated multiplication by the radix (r) is used, until the fractional product is 0 (or until the desired number of binary places). The whole digits of the multiplication results produce the answer, with the first as the MSD, and the last as the LSD.
- Example: Convert $467_{10}$ to octal
  
  $467 / 8 = 58$ remainder 3 (LSD)
  
  $/ 8 = 7$ remainder 2
  
  $/ 8 = 0$ remainder 7 (MSD)
  
  $467_{10} = 723_8$
Binary to Octal Conversion

- Separate the whole binary number portion into groups of 3 beginning at the binary point and working to the left. Add leading zeroes as necessary.

- Separate the fraction binary number portion into groups of 3 beginning at the binary point and working to the right. Add trailing zeroes as necessary.

- Convert each group of 3 to the equivalent octal digit.

- Example:

\[ 3564.875_{10} = 110111\ 101\ 100.111_2 \]
\[ = (6 \times 8^3) + (7 \times 8^2) + (5 \times 8^1) + (4 \times 8^0) + (7 \times 8^{-1}) \]
\[ = 6754.7_8 \]
Binary to Hexadecimal Conversion

• Separate the whole binary number portion into groups of 4 beginning at the binary point and working to the left. Add leading zeroes as necessary.

• Separate the fraction binary number portion into groups of 4 beginning at the binary point and working to the right. Add trailing zeroes as necessary.

• Convert each group of 4 to the equivalent hexadecimal digit.

• Example:

\[3564.875_{10} = 1101\ 1110\ 1100.1110_2\]

\[= (D \times 16^2) + (E \times 16^1) + (C \times 16^0) + (E \times 16^{-1})\]

\[= \text{DEC.E}_{16}\]
Conversion between Number Systems Summary

- Radix-r to decimal:
  - Multiply digits with their corresponding weights and add

- Decimal to binary (radix 2)
  - Whole numbers: repeated division by 2
  - Fractions: repeated multiplication by 2

- Decimal to radix-r
  - Whole numbers: repeated division by r
  - Fractions: repeated multiplication by r

- Binary to Octal
  - Substitute groups of three bits with corresponding octal digit.

- Binary to Hexadecimal
  - Substitute groups of four bits with corresponding hexadecimal digit.
Binary Arithmetic Operations

Addition

• Similar to decimal number addition, two binary numbers are added by adding each pair of bits together with carry propagation.

• Addition Example:

\[
\begin{align*}
X & \quad 190 \\
Y & \quad + 141 \\
X + Y & \quad 331
\end{align*}
\]

\[
\begin{align*}
1011111000 & \quad \text{Carry} \\
101111110 & \\
10001101 & + \\
10100101111 & \\
\end{align*}
\]
Binary Arithmetic Operations

Subtraction

• Two binary numbers are subtracted by subtracting each pair of bits together with borrowing, where needed.

• Subtraction Example:

\[
\begin{array}{cccccccc}
& & & & 0 & 0 & 1 & 1 \\
& & & X & & & & 229 \\
& & & - & Y & & & - 46 \\
& & & & & 183 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
& & & & 0 & 0 & 1 & 1 \\
& 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & \text{Borrow} \\
& 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
& 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\
& 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\
\end{array}
\]
Negative Binary Number Representations

- Signed-Magnitude Representation:
  - For an $n$-bit binary number:
    - Use the first bit (most significant bit, MSB) position to represent the sign where 0 is positive and 1 is negative.
    - Remaining $n-1$ bits represent the magnitude which may range from: $-2^{(n-1)} + 1$ to $2^{(n-1)} - 1$
    - Ex. $1 1 1 1 1 1 1 1_2 = -127_{10}$
  - This scheme has two representations for 0; i.e., both positive and negative 0: for 8 bits: 00000000, 10000000
  - Arithmetic under this scheme uses the sign bit to indicate the nature of the operation and the sign of the result, but the sign bit is not used as part of the arithmetic.
Negative Binary Number Representations

- Two’s complement representation:
- MSB is the sign \((\text{MSB} = 1 \text{ indicates a negative number})\)
- To negate a number complement all bits and add 1

ex. \(119_{10} = 01110111\) complement bits

\[
10001000
\]

+1 add 1

\[
10001001_2 = -119_{10}
\]
Properties of Two's Complement Numbers

- X plus the complement of X equals 0.
- There is one unique 0.
- Positive numbers have 0 as their leading bit (MSB); while negatives have 1 as their MSB.
- The range for an n-bit binary number in 2’s complement representation is:
  \[ \text{from } -2^{(n-1)} \text{ to } 2^{(n-1)} - 1 \]
- The complement of the complement of a number is the original number.
- Subtraction is done by addition to the 2’s complement of the number.
Value of Two's Complement Numbers

- For an n-bit 2’s complement number the weights of the bits is the same as for unsigned numbers except of the MSB or sign bit where the weight is $-2^{n-1}$, thus the value of the n-bit 2’s complement number is given by:

$$D_{\text{2's-complement}} = d_{n-1} \times -2^{n-1} + d_{n-2} \times 2^{n-2} \ldots d_1 \times 2^1 + d_0$$

For example:
the value of the 4-bit 2’s complement number 1011 is given by:

$$\text{value} = d_3 \times -2^3 + d_2 \times 2^2 + d_1 \times 2^1 + d_0$$
$$= 1 \times -2^3 + 0 \times 2^2 + 1 \times 2^1 + 1$$
$$= -8 + 0 + 2 + 1$$
$$= -8 + 3 = -5$$
Extending Two's Complement Numbers: Sign Extension

• An n-bit 2’s complement number can be converted to an m-bit number where m>n by appending m-n copies of the sign bit to the left of the number. This process is called sign extension.

• Example: To convert the 4-bit 2’s complement number 1011 to an 8-bit representation, the sign bit (here = 1) must be extended by appending four 1’s to the left of the number:

\[
\begin{align*}
1011 \text{ 4-bit 2’s-complement} & = \ 11111011 \text{ 8-bit 2’s-complement} \\
\text{To verify that the value of the 8-bit number is still -5} & \\
\text{value of 8-bit number} & = -2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2 + 1 \\
& = -128 + 64 + 32 + 16 + 8 + 2 + 1 \\
& = -128 + 123 = -5
\end{align*}
\]
Two’ complement addition/subtraction

Examples:

\begin{align*}
4 & \quad 0100 & -2 & \quad 1110 \\
+ & -7 & 1001 & + & -6 & 1010 \\
\hline
-3 & \quad 1101 & -8 & 1 & 1000
\end{align*}

- Overflow occurs if signs (MSBs) of both operands are the same and the sign of the result is different.
- Overflow can also be detected if the carry in the sign position is different from the carry out of the sign position.

Ignore carry out from MSB
Negative Binary Number Representations

- One’s-Complement representation
- MSB is the sign (MSB = 1 indicates a negative number)
- Negative numbers are found by complementing all bits

- ex. \(119_{10} = 01110111\)
  \(-119_{10} = 10001000\)

- The range of values for an \(n\)-bit binary number in 1’s complement representation is:

\[-2^{(n-1)} + 1 \text{ to } 2^{(n-1)} - 1\]

- One’s-complement addition/subtraction:
  If there is a carry out of the sign position add 1

Ex. \(-2 \quad 1101\)

\(+ -5 \quad 1010\)

\[\underline{+ \quad 1} \quad 10111\]

\[-7 \quad 1000\]
Value of One's Complement Numbers

- For an n-bit 2’s complement number the weights of the bits is also the same as for unsigned numbers except of the MSB or sign bit where the weight is \(-(2^{n-1} +1)\), thus the value of the n-bit 1’s complement number is given by:

\[D_{1\text{'s-complement}} = d_{n-1} \times -(2^{n-1} +1) + d_{n-2} \times 2^{n-2} + \ldots + d_1 \times 2^1 + d_0\]

For example:

the value of the 4-bit 1’s complement number 1011 is given by:

\[
\text{value} = d_3 \times -(2^3 +1) + d_2 \times 2^2 + d_1 \times 2^1 + d_0 \\
= 1 \times -(2^3 +1) + 0 \times 2^2 + 1 \times 2^1 + 1 \\
= -7 + 0 + 2 + 1 \\
= -7 + 3 = -4
\]
## Comparison of Signed-Magnitude & Complements

**Example: 4-bit signed number (positive values)**

<table>
<thead>
<tr>
<th>Value</th>
<th>Signed-Magnitude</th>
<th>1s Comp.</th>
<th>2s Comp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>+7</td>
<td>0111</td>
<td>0111</td>
<td>0111</td>
</tr>
<tr>
<td>+6</td>
<td>0110</td>
<td>0110</td>
<td>0110</td>
</tr>
<tr>
<td>+5</td>
<td>0101</td>
<td>0101</td>
<td>0101</td>
</tr>
<tr>
<td>+4</td>
<td>0100</td>
<td>0100</td>
<td>0100</td>
</tr>
<tr>
<td>+3</td>
<td>0011</td>
<td>0011</td>
<td>0011</td>
</tr>
<tr>
<td>+2</td>
<td>0010</td>
<td>0010</td>
<td>0010</td>
</tr>
<tr>
<td>+1</td>
<td>0001</td>
<td>0001</td>
<td>0001</td>
</tr>
<tr>
<td>+0</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
</tbody>
</table>
## Comparison of Signed-Magnitude & Complements

**Example: 4-bit signed number (negative values)**

<table>
<thead>
<tr>
<th>Value</th>
<th>Sign-and-Magnitude</th>
<th>1s Comp.</th>
<th>2s Comp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0</td>
<td>1000</td>
<td>1111</td>
<td>-</td>
</tr>
<tr>
<td>-1</td>
<td>1001</td>
<td>1110</td>
<td>1111</td>
</tr>
<tr>
<td>-2</td>
<td>1010</td>
<td>1101</td>
<td>1110</td>
</tr>
<tr>
<td>-3</td>
<td>1011</td>
<td>1100</td>
<td>1101</td>
</tr>
<tr>
<td>-4</td>
<td>1100</td>
<td>1011</td>
<td>1100</td>
</tr>
<tr>
<td>-5</td>
<td>1101</td>
<td>1010</td>
<td>1011</td>
</tr>
<tr>
<td>-6</td>
<td>1110</td>
<td>1001</td>
<td>1010</td>
</tr>
<tr>
<td>-7</td>
<td>1111</td>
<td>1000</td>
<td>1001</td>
</tr>
<tr>
<td>-8</td>
<td>-</td>
<td>-</td>
<td>1000</td>
</tr>
</tbody>
</table>

MSB = 1 indicates a negative number in either notation