Number Systems

• General form of a number in base \( b \) is
  \[
  \text{where } a_d a_{d-1} \ldots a_0 \text{ are the positional coefficients.}
  \]

Modern computers use binary arithmetic, i.e., base 2.

Conversions

• To convert from decimal to binary, divide by 2 repeatedly, read reminders up.
• Easier to convert to octal than to binary.

Addition

• Addition in base 2:
  \[
  \begin{array}{c}
  \text{11001101} \\
  + \text{10011001} \\
  \hline
  \text{11000110}
  \end{array}
  \]

  The sum might have \( \geq \) one more digit than the largest operand.

Addition in base \( p \):

\[
\begin{array}{c}
\text{Addition in base } p:
\end{array}
\]

Multiplication

• Multiplication in base 2:
  \[
  \begin{array}{c}
  \text{00101101} \\
  \times \text{10111001} \\
  \hline
  \text{00000000} \\
  \text{00101101} \\
  \text{00101101} \\
  \text{00101101} \\
  \text{00101101} \\
  \hline
  \text{010000010000101}
  \end{array}
  \]

  The product has \( \approx \) as many digits as the two operands combined.

Number Systems

General form of a number in base \( b \) is
Computers usually have a fixed number of binary digits ("bits"), e.g., 32 bits.

For example, using 6 bits, numbered 0 to 5 from the right:

- The largest number is $2^5 = 32$.
- The smallest number is $0$.

What is $50 + 20$?

$$
\begin{array}{c}
\begin{array}{c}
00110001 \\
+ 01010100 \\
\hline
10001101
\end{array}
\end{array}
$$

The highest bit doesn't fit, so we get:

- Spilling over the lefthand side is considered an error.
- The unsigned result is $166$.

The sign-magnitude notation is:

- The most significant bit (MSB) is the sign bit; $0$ for $+$, $1$ for $-$.

Addition and subtraction are complicated when signs differ.

The one's complement notation:

- $-k = (2^n - 1) - k = 11111...1 - k$.

Addition and subtraction are easy, but there are 2 representations for $0$.

Two's complement notation:

- $-k = 2^n - k = (2^n - 1) - k + 1$.

Two's complement is easily used to negate a 2's complement number:

- First complement all the bits, then add 1.

Computers usually have a fixed number of binary digits ("bits").
### Sign Extension

- To convert from a small signed integer to a larger one, copy the sign bit.
- To convert a large signed integer to a smaller one, check truncated bits:
  - Check the truncated bits for a small signed integer to a larger one, copy the sign bit.

### Floating Point Numbers

- Floating point numbers are like scientific notation.
- Significand restricted to range, e.g., \( m \) and fixed number of digits.
- Floating point is an approximation for infinitely many real numbers.
- Normalized floating point numbers make the representation unique.
- Example:
  - For base 2, the significand is 1 and fixed number of digits.
  - NaN (signaling/quiet) (denormalized)

### IEEE Floating Point

- IEEE format uses a hidden bit to increase precision by 1 bit.
- Single precision (32-bit) format.
- Values from \( 1.1754943508222875 \times 10^{-38} \) to \( 3.4028234663852886 \times 10^{38} \).
IEEE Floating Point, cont'd

- Double precision (double) format

<table>
<thead>
<tr>
<th>Value: 2.2250738585072014E-308</th>
<th>1.7976931348623157E+308</th>
</tr>
</thead>
</table>

Biased exponents in the most-significant bits are useful because

- Integer compare instructions can be used to compare floating point values

- A bit string of 0's represents the value 0.0

- NaN (signaling/quiet)

- Denormalized

\[
\text{Double precision (double) format}
\]