Integer Representation

Representation of integers: unsigned and signed
Modular arithmetic and overflow
Sign extension
Shifting and arithmetic
Multiplication
Casting

modular arithmetic, overflow

\[
\begin{array}{c}
11 & 1011 \\
+ 2 & +0010 \\
\end{array}
\quad
\begin{array}{c}
13 & 1101 \\
+ 5 & +0101 \\
\end{array}
\]

\[x + y\] in \(n\)-bit unsigned arithmetic is \(\text{in math}

\[\text{unsigned overflow} = \]

Unsigned addition overflows if and only if

sign-magnitude

Most-significant bit (MSB) is sign bit
0 means non-negative 1 means negative
Remaining bits are an unsigned magnitude

8-bit sign-magnitude:

Anything weird here?

\[
\begin{array}{c}
00000000 \text{ represents } \underline{\text{______}} \\
01111111 \text{ represents } \underline{\text{______}} \\
10000101 \text{ represents } \underline{\text{______}} \\
10000000 \text{ represents } \underline{\text{______}} \\
\end{array}
\]

Arithmetic?

Example:

\[4 - 3 \neq 4 + (-3)\]

\[
\begin{array}{c}
00000100 \\
+ 10000011 \\
\end{array}
\]

Zero?

8-bit representations

\[
\begin{array}{c}
00001001 \\
10000001 \\
\end{array}
\]

n-bit two's complement numbers:

minimum = maximum =

ex
4-bit unsigned vs. 4-bit two’s complement

1011

\[1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0\]

\[1 \times -2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0\]

difference = \_\_\_ = 2 → -5

Another derivation

How should we represent 8-bit negatives?

• For all positive integers \(x\), we want the representations of \(x\) and \(-x\) to sum to zero.

• We want to use the standard addition algorithm.

\[
\begin{array}{c}
  \text{00000001} \\
  \text{00000010} \\
  \text{00000011}
\end{array}
\]

+ 00000000

+ 00000000

+ 00000000

• Find a rule to represent \(-x\) where that works...

unsigned shifting and arithmetic

\[x = 27;\]
\[y = x \ll 2;\]
\[y == 108\]

\[x = 237;\]
\[y = x \gg 2;\]
\[y == 59\]

unsigned logical shift left

11101101

0011101101

unsigned logical shift right

11101101

0011101101

two's complement shifting and arithmetic

\[x = -101;\]
\[y = x \ll 2;\]
\[y == 108\]

\[x = -19;\]
\[y = x \gg 2;\]
\[y == -5\]
**shift-and-add**

Available operations

- $x \ll k$ implements $x \cdot 2^k$
- $x + y$

Implement $y = x \cdot 24$ using only $\ll$, $+$, and integer literals

---

**What does this function compute?**

```c
unsigned puzzle(unsigned x, unsigned y) {
    unsigned result = 0;
    for (unsigned i = 0; i < 32; i++) {
        if (y & (1 << i)) {
            result = result + (x << i);
        }
    }
    return result;
}
```