Integer Representation

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

Wellesley CS 240

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

x+y in \( n \)-bit unsigned arithmetic is
\[
\text{unsigned overflow} = \text{math answer too big to fit}
\]

Unsigned addition overflows if and only if a carry bit is dropped.

(4-bit) two's complement signed integer representation

\[
\begin{array}{cccc}
1 & 0 & 1 & 1 \\
-2^3 & 2^2 & 2^1 & 2^0
\end{array}
\]

\[
= 1 \times (-2^3) + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0
\]

4-bit two's complement integers:
minimum =
maximum =
two’s complement vs. unsigned

\[
\begin{array}{cccccc}
2^{n-1} & 2^{n-2} & \ldots & 2^1 & 2^0 \\
-2^{n-1} & 2^{n-2} & \ldots & 2^1 & 2^0 \\
\end{array}
\]

What's the difference?

n-bit unsigned numbers:
minimum = \(2^{n-1}\)
maximum = \(2^n - 1\)

n-bit two's complement numbers:
minimum = \(-2^{n-1}\)
maximum = \(2^n - 1\)

8-bit representations

\[
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\
\end{array}
\]

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.

\[
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
+ & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
+ & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

- Find a rule to represent \(-x\) where that works...
unsigned **shifting** and **arithmetic**

\[
\begin{align*}
\text{unsigned } &\quad x = 27; \\
&\quad 0 0 0 1 1 0 1 1 \\
y &\quad y = x << 2; \\
&\quad 0 0 0 1 1 0 1 1 0 0 \\
y == 108 &\quad 0 0 0 1 1 0 1 1 0 0 \\
\end{align*}
\]

logical shift left

\[
\begin{align*}
\text{unsigned } &\quad x = 237; \\
&\quad 1 1 1 0 1 1 0 1 1 \\
y &\quad y = x >> 2; \\
&\quad 0 1 1 1 0 1 1 0 1 \\
y == 59 &\quad 0 1 1 1 0 1 1 0 1 \\
\end{align*}
\]

logical shift right

---

**two's complement shifting** and **arithmetic**

\[
\begin{align*}
\text{signed } &\quad x = -101; \\
&\quad 1 0 0 1 1 0 1 1 \\
y &\quad y = x << 2; \\
&\quad 1 0 0 1 1 0 1 1 0 0 \\
y == 108 &\quad 1 0 0 1 1 0 1 1 0 0 \\
\end{align*}
\]

logical shift left

\[
\begin{align*}
\text{signed } &\quad x = -19; \\
&\quad 1 1 1 1 0 1 1 0 1 \\
y &\quad y = x >> 2; \\
&\quad 0 0 1 1 1 0 1 1 \\
y == -5 &\quad 0 0 1 1 1 0 1 1 \\
\end{align*}
\]

arithmetic shift right

---

**shift-and-add**

**Available operations**

\[
\begin{align*}
x &\quad x << k \\
&\quad \text{implements } x \times 2^k \\
x + y &\quad x + y
\end{align*}
\]

**Implement** \( y = x \times 2^4 \) **using only** \(<, +, \text{ 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;
}
```