CSE 30: Computer Organization and Systems Programming

Lecture 3: Number Representation and Overflow

Diba Mirza
University of California, San Diego
Choosing a representation for signed numbers

• What do we do with numbers? $+, -, \times, \div, <, >$

Unsigned representation

Addition

7
+ 3
10
1010 (10)

-7
1111

+ 3
0011

(−4) (expected) 0010
(−4) (actual) 0010

Sign
↓
MSB
Magnitude
Shortcomings of sign and magnitude?

• Arithmetic circuit complicated
  – Special steps depending whether signs are the same or not

• Also, two zeros
  – $0x00000000 = +0_{\text{ten}}$
  – $0x80000000 = -0_{\text{ten}}$

• Therefore sign and magnitude abandoned!
One’s Complement

• To make a number negative, just flip all its bits!
What is the range of numbers we can represent in One’s complement format using N bits

A. \(-2^{N-1} \text{ to } 2^{N-1}\)
B. \(-2^{N-1} \text{ to } 2^{N-1} - 1\)
C. \(-(2^{N-1} - 1) \text{ to } 2^{N-1} - 1\)
D. \(-2^N \text{ to } 2^N\)
Short comings of One’s Complement

• Still have two zeros (11111111 and 00000000)

• Need an extra step in addition if there is a carry out – see example 1.6 in ARM book
Two’s Compliment

- Two’s’s compliment encoding

\[-2^{n-1}d_{n-1} + 2^{n-2}d_{n-2} + \ldots + 2^1d_1 + 2^0d_0\]

What is the range of values that can be represented in 2’s complement using N bits?

-2^{n-1} to 2^{n-1} - 1
Two’s Complement

- Flip all the bits of unsigned representation and add 1

\[ \begin{align*}
0000 & \rightarrow 1111 \\
0001 & \rightarrow 1110 \\
0010 & \rightarrow 1101 \\
0011 & \rightarrow 1100 \\
0100 & \rightarrow 1011 \\
0101 & \rightarrow 1010 \\
0110 & \rightarrow 1001 \\
0111 & \rightarrow 1000 \\
1000 & \rightarrow 0111 \\
1001 & \rightarrow 0110 \\
1010 & \rightarrow 0101 \\
1011 & \rightarrow 0100 \\
1100 & \rightarrow 0011 \\
1101 & \rightarrow 0010 \\
1110 & \rightarrow 0001 \\
1111 & \rightarrow 0000 
\end{align*} \]

\[2 - 3 = \ ?\]
Two’s Complement: $1101_2 = ?_{10}$

A. -2

B. -3

C. -4

D. -5
Negating Two’s Compliment

- To negate any number, flip all the bits and add 1

\[ -x \rightarrow \text{2's comp} \rightarrow x \]

How would you subtract two numbers that are both represented in 2’s complement?

\[ A \ - \ B \]
\[ A + (-B) \]
\[ A \ (\text{+}) \ B \]

\[ \text{2's comp on } B \]
The negation of $11110001_2$ is $\underline{\blacksquare}2$

A. $00001110$

B. $00001111$

C. $00011110$

D. $01110001$
Sign Extension

• Given: Number represented in 4 bits using 2’s complement format
• Find: 8 bit representation of the same number in 2’s complement format
Addition and Subtraction

• Positive and negative numbers are handled in the same way.
• The carry out from the most significant bit is ignored.
• To perform the subtraction $A - B$, compute $A + \text{(two's complement of B)}$
Overflow

- Overflow occurs when an addition or subtraction results in a value which cannot be represented using the number of bits available.

Unsigned:

\[
\begin{array}{c}
1111 \\
0010 \\
\hline
0001
\end{array}
\]

Signed overflow occurs when adding two numbers:

\[
\begin{array}{c}
0110 (-6) \\
+0111 (7) \\
\hline
10101 \text{ Overflow} \\
0101 \text{ Correct Result}
\end{array}
\]
Is overflow a problem in modern programs?

A. We have totally solved this business!

B. Yep, still a problem.
Handling Overflow

- Hardware can detect when overflow occurs

- Software may or may not check for it
  - C and Java don’t!
How To Detect Overflow?

Q: In 2’s complement representation, overflow occurs on addition if there is a carry out of the most significant bit (sign bit).

A. True
B. False
How To Detect Overflow

• Contention: On addition, an overflow occurs if and only if the carry into the sign bit differs from the carry out from the sign bit.
Will $0111_2 + 0101_2$ result in overflow?

A. Yes

B. No

C. It depends
Will $011111111011_2 + 111111101101_2$ result in overflow (assume 2’s comp)?

A. Yes

B. No

C. It depends