2's-Complement Representation

RECALL FROM THE LAST LECTURE:

n-bit 2's-complement representation of D: \[ [D]_2 = 2^n - D \]

How to compute it?

\[ [D]_2 = (2^n - 1 - D) + 1 \]

1. Complement the bits
2. Add 1 to the Least Significant Bit
3. Discard carry out from Most Significant Bit
Addition with 2’s Complement

- Added by ordinary binary addition, ignoring any carries beyond the MSB.
- The result must be inside the range of the numbers represented by n-bits. Otherwise overflow occurs, and the result is not correct.

Example, number of bits limited to $n = 5$

Then, the range is $-2^{5-1} = -16 \ldots 2^{5-1} - 1 = +15 \sim 32$ numbers

<table>
<thead>
<tr>
<th>+5_{10} 00101</th>
<th>+9 01001</th>
<th>+12 01100</th>
</tr>
</thead>
<tbody>
<tr>
<td>+7_{10} 00111</td>
<td>+8 11000</td>
<td>+7 00111</td>
</tr>
<tr>
<td>+12_{10} 01100</td>
<td>+1 100001</td>
<td>+19 10011</td>
</tr>
</tbody>
</table>

2's complement of +8: $8_{10} = 01000 \rightarrow 10111$

ignore carries beyond MSB

negatives number resulted from adding 2 positive numbers $\rightarrow$ overflow

---

Carries and Overflow

- We ignore carries beyond MSB because we are adding two’s complement numbers as if they were unsigned numbers.
- A carry beyond MSB is an artifact of adding the sign bits and does not indicate overflow:
  - For example, every time we add two negative numbers, a carry beyond MSB occurs, but not necessarily an overflow.
  - On the other hand, in a previous-slide example, overflow occurred without a carry beyond MSB.
Overflow Detection Rule (1)

Overflow: If the sign of the addends is the same but different from the sign of the result.

\[ \begin{array}{c|c|c|c}
-14 & 14_{10} = 01110 & 10001 & -7_{10} = 11001 \\
-7 & 10010 & = -14_{10} & \Rightarrow 10010 \\
-21 < -16 & 101011 & \text{(2's complement)} & 11001 \\
\end{array} \]

If \( n = 6 \) bits, no overflow, range of numbers \(-32 \ldots +31\):

\[ \begin{array}{c|c|c|c|c|c|c}
& 110010 & + & 111001 & \rightarrow & \text{verify result:} & 010100 \\
\hline
\text{ignore carries} & 1 & \text{beyond MSB} & | & 101011 & + & 1 \\
\text{ignore carries} & & & & & & 010101 = 21_{10} \text{ magnitude} \\
\end{array} \]

Overflow Detection Rule (2)

- Overflow occurs when the value affects the sign bit:
  - adding two positives yields a negative
  - adding two negatives gives a positive
  - subtract a negative from a positive and get a negative
  - subtract a positive from a negative and get a positive
- No overflow when adding a positive and a negative number
- No overflow when subtracting two numbers of same sign
- Consider the operations \( A + B \), and \( A - B \)
  - Can overflow occur if \( B \) is 0 ? cannot occur !
  - Can overflow occur if \( A \) is 0 ? can occur !
    (for \( A - B \) if \( B = -2^{n-1} \))
    [ e.g., for \( n=5 \): \( 0 - (-16) = +16 \) ]
Subtraction with 2’s Complement

- \( A - B = A + (-B) = A + [B]_2 \)
- Subtraction identical to addition, the sign absorbed by the representation

- Again, the result must be inside the range of the numbers represented by n-bits. Otherwise overflow occurs, and the result is not correct.

Example, \( n = 5 \), the range is \(-2^{5-1} = -16 \) \( \ldots \) \( 2^{5-1} - 1 = 15 \)

\[
\begin{array}{c|c|c|c|c}
+5 & 00101 & +9 & 01001 & -8 & 11000 \\
-3 & 11101 & -9 & 10111 & \hline
0 & \text{\textcolor{red}{\textbf{00000 = 0}}}_{10} & -17 & 101111 & \\
\end{array}
\]

\( 2 \text{'s complement of } -8 \)
\( 9_{10} = 10110 \)
\( 11000 = -8 \)
\( \text{ignore carries beyond MSB} \)

\[
\begin{array}{c|c}
+5 & 00101 \\
-8 & 11000 \\
\hline
-3 & 11101 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c|c}
9_{10} & 01001 \rightarrow 10110 & -8 & 11000 & \text{positive number } \rightarrow \text{ overflow} \\
10111 & \rightarrow -8_{10} & \hline
0 & \text{\textcolor{red}{\textbf{00000 = 0}}}_{10} & -17 & 101111 & \\
\end{array}
\]

Multiplication in Decimal

- An example in decimal:

\[
214 \times 45 = 9630
\]

(1) For each digit of Multiplier, multiply Multiplicand by it.
(2) Multiply the product by the order of the digit \((\times 10^i)\), i.e., shift it by one to the left:

\[
\begin{array}{c|c}
\times & \text{ multiplicand } \times \text{ multiplier} \\
\hline
\text{aaa} & \text{zzz} \\
\text{bbb} & \text{aaaa} \\
\text{ccc} & \text{bbbb} \\
\text{ddd} & \text{cccc} \\
\text{eee} & \text{ddddd} \\
cetc... & \text{eedd} \\
\hline
\text{9630} & \text{product} \\
\end{array}
\]

\( 214 \times 5 = 1070 \) and then add to it the result of \( 214 \times 4 = 856 \) right-shifted by one column.
Multiplication in Binary

- Multiplying in binary follows the same form as in decimal:

  Product $P$ is composed purely of selecting, shifting and adding multiplicand $A$. The $i^{th}$ bit of multiplier $B$ indicates whether a shifted version of $A$ is to be selected in the $i^{th}$ row of the sum.

Multiplication in Binary

- Because there are only two digits in binary (0 and 1). The multiplication algorithm becomes only:
  1. Shift Multiplicand
  2. Multiply Shifted Multiplicand by 1 or 0
  3. Add the Shifted Multiplicands

- So we can perform multiplication using just full adders and a little logic for selection, in a layout which performs the shifting.
**Multiplication with Partial Products**

- In digital systems, more convenient to work with **partial products**, instead of listing all shifted multiplicands and then adding them.

\[
\begin{array}{c}
10_{10} \\
\downarrow \\
11_{10}
\end{array} \times 
\begin{array}{c}
\downarrow \\
1011
\end{array} = 110_{10}
\]

\[
\begin{array}{c}
00000000 \\
+ \ 1010 \\
00011110 \\
+ \ 1010000
\end{array} = 01101110
\]

**Multiplication with 2’s Complement (1)**

- Two’s complement multiplication works the same as unsigned multiplication: shifted multiplicand is weighted by the multiplier bit, except for the MSB which, when “1” (i.e., **negative** multiplier), has a **negative** weight.

\[
\begin{array}{c}
-6 \\
\downarrow \\
-5
\end{array} \times 
\begin{array}{c}
\downarrow \\
-1011
\end{array} = 30
\]

\[
\begin{array}{c}
00000000 \\
+ \ 11010 \\
111010 \\
+ \ 110100
\end{array} = 00011110
\]
Multiplication with 2's Complement (2)

- Two's complement multiplication works the same as unsigned multiplication:
  - when multiplier is positive, its MSB has zero weight:

\[
-6 \times +5 = -30
\]

\[
\begin{align*}
1010 & \times 0101 \\
00000 & + 11010 \\
111010 & + 000000 \\
0000000 & + 1101000 \\
11100010 & + 00000000 \\
= 11100010
\end{align*}
\]

\[
[11100010]_2 = 00011101 \quad \frac{1}{8} 00011110 = 30_{10}
\]

Decimal Division

\[
\begin{align*}
&\quad \frac{827_{10}}{21_{10}} \\
&= 39_{10} \\
&\quad \text{quotient} \\
&\quad \text{remainder}
\end{align*}
\]

1. Select most-significant digit from Dividend to compare to Divisor
   - \(8 < 21\)

2. It's smaller than Divisor, so, consider two digits
   - \(82 > 21\times1 = 21\)
   - \(82 > 21\times2 = 42\)
   - \(82 > 21\times3 = 63\)
   - \(82 < 21\times4 = 84\)

3. Find greatest \(d\) (from 1 to 9) that satisfies:
   - \(82 \geq 21 \times d\)
   - Intuition (guessing) when done by human
   - Algorithm that increases \(d\) until
     - either \(d \times 21 > 82\); use \((d-1)\)
     - or \(d = 9\)

4. Determine \(d\) using:
   - Intuition (guessing) when done by human
   - Algorithm that increases \(d\) until
     - either \(d \times 21 > 82\); use \((d-1)\)
     - or \(d = 9\)
Binary Division

Many steps before finding a number > Divisor. Presence of leading 0s disturbs the conventional algorithm.

Extract digits from Dividend and shift them to align them with Divisor.

In binary, \( d \) can only take the value 0 or 1. Means:
Divisor \( \times d \leq \) Extracted Digits from Dividend \( \Rightarrow d = 1 \)

Quotient: Shift left serial input from LSB.

Every step the Extracted Digits are compared to the Divisor:
If Divisor \( \times 1 > \) Extracted Digits \( \Rightarrow \) Shift in 0 in the Quotient
If Divisor \( \times 1 \leq \) Extracted Digits \( \Rightarrow \) Shift in 1 in the Quotient

\[
\begin{align*}
827_{10} & \quad \text{dividend} \\
+ & \quad \text{divisor} \\
= & \quad \text{quotient} \\
8_{10} & \quad \text{remainder}
\end{align*}
\]

\[
827_{10} \div 21_{10} = 39_{10}
\]

\[
8_{10} \quad \text{dividend} \quad 001100111011_{2} \quad \text{divisor} \quad + 000000010101_{2} \quad \text{quotient} \quad = 000001000111_{2} \quad \text{remainder} \quad 000000001000_{2}
\]

82710 \div 2110 = 3910

\[
\begin{align*}
001100111011_{2} \quad \text{dividend} \\
- 000000010101_{2} \quad \text{divisor} \\
\hline
000000010001_{2} \quad \text{quotient} \\
- 000000010101_{2} \quad \text{remainder}
\end{align*}
\]