Signed number representations

- We might also want to represent signed integers in different bases.
- One simple way to do this would be to use a **sign bit**; Given the representation of a number in n digits, we could add another digit:
  - If the extra digit is 0, the number is positive;
  - If the extra digit is 1, the number is negative.
- Example... Representation of 9 and -9 using 8 digits in base-2:
  - The value 9 with 8 bits is 00001001_2.
  - The value $+9$ would be represented as 000001001_2. Note the extra leading bit with value 0.
  - The value -9 would be represented as 100001001_2. Note the extra leading bit with value 1.
- The "problem" with using a sign bit is that we introduce the concept of "$+0$" and "-0". Also (not evident now) is that when we build circuits to perform numerical operations, the need for a sign bit complicates the circuitry unnecessarily.

Radix complements

- The solution to our dilemma of representing negative numbers is to consider the idea of r-complements (or radix complements).
- The r-complement of a positive value V in base-r using n digits is defined as

$$\begin{array}{rl} r^n - V & \text{if } V \neq 0 \\ 0 & \text{if } V = 0 \end{array}$$

Radix complements

- Example: Represent the value 546700 in base-10. Then, find the 10s complement of the result. Assume only 7 digits are available.
  - The value 546700 has representation 0546700_{10}.
  - The 10s complement of 0546700_{10} is $10^7 - 0546700 = 10000000 - 0546700 = 0453300_{10}$.
- Example: Represent the value 88 in base-2. Then, find the 2s complement of the result. Assume only 8 digits are available.
  - The value 88 has representation 01011000_2.
  - The 2s complement of 01011000_2 is $2^8 - 01011000 = 100000000 - 01011000 = 10101000_2$.

Radix complements in base-2 (2s complements)

- In base-2, the 2s complement of a number can be found very quickly in other ways (i.e., other than performing a subtraction):
  1. You can simply "flip the bits" and add 1.
  2. You can copy bits right to left until the "first" 1 is encountered and then start flipping bits.

Signed numbers

- Turns out that representing negative numbers using radix complements is a really good idea (vs. using a sign bit).
- In other words, if we want to represent a negative value in base-r, we find the rs complement of its absolute value and whatever we get represents the negative number.
- Example: Represent -546700 in base-10 using 7 digits.
  - We previously found that the 10s complement if $|-546700| = 546700$ was 0453300_{10}.
  - Therefore, -546700 will be represented by 0453300_{10}.

  Example: Represent -88 in base-2 using 8 bits.
  - We previously found that the 2s complement if $|-88| = 88$ was 01011000_2.
  - Therefore, -88 will be represented by 10101000_2.
-

Signed numbers and 2s-complements

- Note that, in base-2 a positive value will always have *leading zeros* while a negative value will always have *leading ones*.
- If we are given a number in base-2 and we are told that it represents a signed integer *and* and the leading bits are 0, we can simply find the (+ve) value it represents.
- If we are given a number in base-2 and we are told that it represents a signed integer *and* and the leading bits are 1, we need to first find the 2s complement and then add a negative sign to figure out the value.

Signed numbers and 2s-complements

- Example: What value does 11101010_2 represent if the number is represented using 2s complement?
  - There are leading ones, so we know the number is negative...
  - Take the 2s complement of 11101010_2 to get 00010110_2.
  - The representation 00010110_2 equals a value of 22.
  - This means the absolute value of the number is 22.
  - Therefore, 11101010_2 must be the representation of -22.

Signed numbers and 2s-complements

- We should consider the range of numbers we can represent in n bits in base-2.
- Example: With 3 bits we have the following patterns available and the corresponding values:

0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	-4
1	0	1	-3
1	1	0	-2
1	1	1	-1

- We can represent the numbers $-2^{n-1} \cdots 2^{n-1} - 1$.
- Note we have only a single representation of the value 0.

Signed addition

- Turns out that if negative numbers are represented using rs complement, then addition works! We perform the addition and ignore the carry out.
- Example 1...

$$\begin{array}{rr|rr} & 00000110 & & 6 \\ + & 00001101 & + & 13 \\ \hline & & & \end{array} \rightarrow \begin{array}{rr|rr} & 00000110 & & 6 \\ + & 00001101 & + & 13 \\ \hline 0 & \underbrace{00010011}_{\text{result}} & & 19 \end{array}$$

- Example 2...

$$\begin{array}{rr|rr} & \text{-}00000110 & & \text{-}6 \\ + & 00001101 & + & 13 \\ \hline & & & \end{array} \rightarrow \begin{array}{rr|rr} & 11111010 & & \text{-}6 \\ + & 00001101 & + & 13 \\ \hline 1 & \underbrace{00000111}_{\text{result}} & & 7 \end{array}$$

Signed addition

- Example 3...

$$
\begin{array}{cr|cr}
 & -00000110 & & -6 \\
+ & -00001101 & + & -13 \\
\hline
 & & &
\end{array}
\quad \rightarrow \quad
\begin{array}{cr|cr}
 & 11111010 & & -6 \\
+ & 11110011 & + & -13 \\
\hline
1 & \underbrace{11101101}_{\text{result}} & & -19
\end{array}
$$

- Example 4...

$$
\begin{array}{cr|cr}
 & 00000110 & & 6 \\
+ & -00001101 & + & -13 \\
\hline
 & & &
\end{array}
\quad \rightarrow \quad
\begin{array}{cr|cr}
 & 00000110 & & 6 \\
+ & 11110011 & + & -13 \\
\hline
0 & \underbrace{11111001}_{\text{result}} & & -7
\end{array}
$$

Signed subtraction

- If we represent negative numbers using complements, then subtraction becomes much easier. We avoid borrows and simply use the idea of "adding the opposite" or $(\pm M) - (\pm N) = (\pm M) + (\mp N)$.
- In other words, instead of subtracting the subtrahead N from the minuend M, we add the 2's complement of the subtrahend N to the minuend M.
- Since we know that addition works, so to will subtraction.

Subtraction using radix complements

- Done by using complements and addition instead of subtraction...
- Example 1...

$$\begin{array}{rr|rr} & \text{-00000110} & & \text{-6} \\ \text{-} & \text{-00001101} & \text{-} & \text{-13} \\ \hline & & & \end{array} \rightarrow \begin{array}{rr|rr} & \text{-00000110} & & \text{-6} \\ + & 00001101 & + & 13 \\ \hline & & & \end{array}$$

$$\rightarrow \begin{array}{rr|rr} & 11111010 & & \text{-6} \\ + & +00001101 & + & 13 \\ \hline 1 & \underbrace{00000111}_{\text{result}} & & +7 \end{array}$$

Overflow and signed addition

- We need to be concerned about *overflow* and *underflow* when performing signed addition. This can only happen if we are adding either:
  1. Two positive numbers that exceed the upper limit of what we can represent; or
  2. Two negative numbers that exceed the lower limit of what we can represent.

Overflow and signed addition

- Consider...

$$
\begin{array}{cccccccccc}
 & {}^{0} & {}^{1} & {}^{0} & {}^{0} & {}^{0} & {}^{0} & {}^{0} & {}^{0} & \\
 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 70 \\
+ & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & +80 \\
\hline
 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 150
\end{array}
$$

- The value 150 is too large to represent in 8-bits. Notice that the result is *obviously wrong* since it indicates the result of the addition is a negative number even though we are adding two positive numbers.

Overflow and signed addition

- Consider...

$$
\begin{array}{c|cccccccc|r}
 & ^1 & ^0 & ^1 & ^1 & ^0 & ^0 & ^0 & ^0 & \\
 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & -70 \\
+ & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & -80 \\
\hline
 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & -150
\end{array}
$$

- The value -150 is too large to represent in 8-bits. Notice that the result is *obviously wrong* since it indicates the result of the addition is a postive number even though we are adding two negative numbers.

Overflow and signed addition

- Turns out that an observation can be made: If the carry in to the most significant digit is *different* than the carry out from the most significant digit, then either overflow or underflow has occurred.
    - This is true for signed arithmetic using 2s complement representation of negative numbers.
    - If there is no overflow (e.g., consider adding a +ve number and a -ve number where overflow cannot happen), the carry in and out at the most significant bit will *always* be the same.
- Summary:
    - For unsigned addition, a carry out from the most significant bit indicates overflow;
    - For signed addition, if the carry in and out at the most significant bit are different, this indicates overflow.