Two's Complement

Based on slides by Jared Moore

Two's Complement

Key: Most-significant bit has a weight of -2^{N-1} rather than 2^{N-1}

For 1101_2, instead of

$2^3 + 2^2 + 2^0 = 13$

we have

$-2^3 + 2^2 + 2^0 = -3$

Two's Complement

Negation Algorithm:

1. Flip all bits in the number
2. Add 1 to the least-significant bit
3. (Ignore overflow)

- Convert the following to their 2's Complement Negative (4-bits):
  - 1_{10}
  - 5_{10}
  - 0_{10}
  - 7_{10}

Two's Complement

Negation Algorithm:

1. Flip all bits in the number
2. Add 1 to the least-significant bit
3. (Ignore overflow)

Note on terminology:

The number system is referred to as ”Two's complement numbers”.

The negation algorithm above is called “taking the two's complement”.

Two’s Complement -- why bother?

One representation for 0. (How?)

What implication on the range of possible values does this have?
Think about the 4-bit case.

Adding Two's Complement

$$\begin{array}{r} 6_{10} \\ +\ 1_{10} \\ \hline \end{array} \qquad \begin{array}{r} -4_{10} \\ +\ -2_{10} \\ \hline \end{array} \qquad \begin{array}{r} 4_{10} \\ +\ -\ 2_{10} \\ \hline \end{array} \qquad \begin{array}{r} -6_{10} \\ +\ 2_{10} \\ \hline \end{array}$$

Adding Two's Complement

$$\begin{array}{r} 0110_2 \\ +\,0001_2 \\ \hline 0111_2 \end{array} \qquad \begin{array}{r} 1100_2 \\ +\,1110_2 \\ \hline 1010_2 \end{array} \qquad \begin{array}{r} 0100_2 \\ +\,1110_2 \\ \hline 0010_2 \end{array} \qquad \begin{array}{r} 1010_2 \\ +\,0010_2 \\ \hline 1100_2 \end{array}$$

Should be

7 ✅ -6 ✅ 2 ✅ -4 ✅

Two's Complement -- why bother?

One representation for 0.

- This is nice, but *not* the biggest reason we use two's complement

Identical to unsigned in terms of addition!

Subtracting Two's Complement

$$\begin{array}{r} -14_{10} \\ -\ 12_{10} \\ \hline \end{array} \qquad \begin{array}{r} -10_{10} \\ -\ 1_{10} \\ \hline \end{array} \qquad \begin{array}{r} 14_{10} \\ -\ 12_{10} \\ \hline \end{array} \qquad \begin{array}{r} -7_{10} \\ -\ 3_{10} \\ \hline \end{array}$$

First, need to decide on number of bits. Assume 4-bit numbers for this

Then, (1) negate the second number and (2) add the numbers

Two's Complement

Another way to think of two's complement numbers is to remember that

$$X + -X = 0$$

To negate 6-bit two's complement number 110110_2 , find number that results in sum of 0

$$\begin{array}{r} 110110_2 \\ +\ 001001_2 \\ \hline 111111_2 \\ +\ 000001_2 \\ \hline 000000_2 \end{array}$$

Number we want is thus $001001_2 + 000001_2 = 001010_2$

This is the same as algorithm learned earlier, just from a different perspective

Sign Extension

Given that -4 in 4-bit two's complement is 1100_2 , what is -4 in 5-bit two's complement?

Turns out, it is simply 11100_2 (You should check if you are not sure)

Just prepended a 1 to bit version

Sign Extension

Similar idea works for positive numbers:

$$4 = 0100_2 \quad \text{(4-bit)}$$
$$4 = 00100_2 \quad \text{(5-bit)}$$

To represent the same number with more bits:

For negative numbers, prepend 1

For positive numbers, prepend 0

In general, prepend the sign bit to the number

Two's Complement

- Magnitude of negative numbers works opposite of intuition
  - 1111 is the smallest in magnitude negative number
  - 1000 is the largest in magnitude negative number

- Positive numbers work exactly as expected
  - 0111 is the largest positive number
  - 0001 is the smallest positive number

- First bit can still be considered the sign. Why?

System	Range
Unsigned	$[0, 2^N - 1]$
Sign/Magnitude	$[-2^{N-1} + 1, 2^{N-1} - 1]$
Two's Complement	$[-2^{N-1}, 2^{N-1} - 1]$