# CS-341 Lecture 6

February 20, 2001
Dr. Vickery

## Assigning Numeric Values to 4-bit Numbers

| Binary | Unsigned | Sign-Magnitude | Biased (8) | One's Complement | Two's Complement |
|---|---|---|---|---|---|
| 0000 | 0 | +0 | -8 | +0 | 0 |
| 0001 | 1 | +1 | -7 | +1 | +1 |
| 0010 | 2 | +2 | -6 | +2 | +2 |
| 0011 | 3 | +3 | -5 | +3 | +3 |
| 0100 | 4 | +4 | -4 | +4 | +4 |
| 0101 | 5 | +5 | -3 | +5 | +5 |
| 0110 | 6 | +6 | -2 | +6 | +6 |
| 0111 | 7 | +7 | -1 | +7 | +7 |
| 1000 | 8 | -0 | 0 | -7 | -8 |
| 1001 | 9 | -1 | +1 | -6 | -7 |
| 1010 | 10 | -2 | +2 | -5 | -6 |
| 1011 | 11 | -3 | +3 | -4 | -5 |
| 1100 | 12 | -4 | +4 | -3 | -4 |
| 1101 | 13 | -5 | +5 | -2 | -3 |
| 1110 | 14 | -6 | +6 | -1 | -2 |
| 1111 | 15 | -7 | +7 | -0 | -1 |

## Two's Complement Encoding

- Using *n* bits to represent values, the range is $-2^{n-1}$ to $+2^{n-1}-1$
- Consider the leftmost bit to have a negative weight.
  - Useful when changing word size: *sign extension.*
  - Also makes it easy to evaluate numbers.
- Negate by subtracting from zero
  - Equivalent to "flip bits and add 1"
- Subtract by negating minuend and adding
- Carry is normal and usually ignored.
- Overflow is when correct value cannot be represented in *n* bits.

## The Leftmost Bit Has a Negative Weight

- $1101 = -2^3 + 2^2 + 2^0$
  - $= \underline{-8} + 4 + 1$
  - $= -3$
- $11101 = -2^4 + 2^3 + 2^2 + 2^0$
  - $= \underline{-16 + 8} + 4 + 1$
  - $= -3$
  - Because $-8 \equiv -16 + 8$

## Sign Extension

- $1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1001_2$
  - Same as $1001_2$ (-7)

BUT:

- $1011\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1001_2$
  - Equals $-2^{31} + 2^{29} + 2^{28} + 2^{27} + 2^{26} + 2^{25} + \ldots$

## Negate by Subtracting from Zero

- $(0 - X) = -X$
- In *n* bits, zero is the same as $2^n$
  - For example, $2^4$ is $1\underline{0000}_2$
- For any *n* $2^n-1$ is *n*-many ones.
  - For example, $2^4-1$ is 15, which is $1111_2$
- If you subtract from $2^n-1$, there will never be any borrows.
  - Only two cases: 1-0 is 1 and 1-1 is 0

## Two's Complement Negation

- When using two's complement encoding for numbers, "taking the two's complement" means to negate the value of a number.
- Subtract from zero, which is the same as subtracting from $2^n$-1 and adding 1, which is the same as flipping the bits and adding 1.
  - Positive numbers become negative
  - Negative numbers become positive
  - Zero becomes zero
  - But $-2^{n-1}$ becomes … $-2^{n-1}$

## Two's Complement Subtraction

- Negate the minuend (second operand), and add.
  - X-Y = X+(-Y)
- ALU design: just one circuit for adding, plus a small amount of logic for negating the minuend instead of a whole other circuit for subtracting.
  - Simpler to design *and* faster execution.
- A carry out of the leftmost position is normal, and may be ignored.
  - May be used for multiple-precision arithmetic.

## Two's Complement Overflow

- If the result of an addition is bigger than $+2^{n-1}-1$ or less than $-2^{n-1}$, the correct answer cannot be represented with *n* bits, and *overflow* has occurred.
- Carry into the leftmost position will be unequal to the carry out of the leftmost position *iff* overflow occurs.
  - Adding a positive value to a negative value cannot overflow.
  - Add two positives: sign bit of 1 indicates overflow
  - Add two negatives: sign bit of 0 indicates overflow

## Floating-Point Encoding

- Real numbers: range from -∞ to +∞
  - Infinitely many values between any two points on the number line.
- Many schemes, but virtually all computers now use IEEE-754 encoding.
- All schemes based on scientific notation
  - Signed fraction * $\text{base}^{\text{Signed exponent}}$
  - The base doesn't have to be encoded
    - It's always base two for IEEE-754

## IEEE-754 Encoding

- Two formats, using 32 and 64 bits
  - Correspond to Java's float and double data types.
    - Sign of fraction: 1 bit
    - Signed exponent: 8 bits (biased-127) or 11 bits (biased-1023) for 32 and 64-bit formats.
    - Value of fraction: 23 or 52 bits for 32 and 64-bit formats.