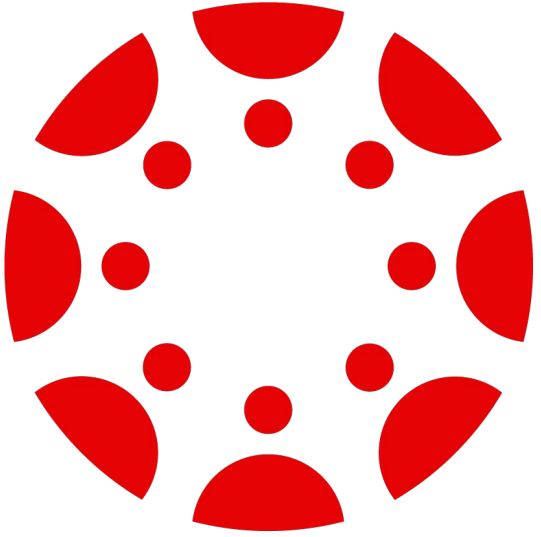
Quiz 1 Review

- Canvas is fun.



Two's Compliment

- Reserve MSbit for signedness

PRINCIPLE: Definition of two's-complement encoding

For vector $\vec{x} = [x_{w-1}, x_{w-2}, \ldots, x_0]$:

$$B2T_w(\vec{x}) \doteq -x_{w-1}2^{w-1} + \sum_{i=0}^{w-2} x_i 2^i \tag{2.3}$$

The most significant bit x_{w-1} is also called the *sign bit*. Its "weight" is -2^{w-1}, the negation of its weight in an unsigned representation. When the sign bit is set to 1, the represented value is negative, and when set to 0, the value is nonnegative. As examples, Figure 2.13 shows the mapping, given by $B2T$, from bit vectors to integers for the following cases:

$$\begin{array}{lcllcll}
B2T_4([0001]) & = & -0\cdot 2^3 + 0\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 & = & 0+0+0+1 & = & 1 \\
B2T_4([0101]) & = & -0\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 & = & 0+4+0+1 & = & 5 \\
B2T_4([1011]) & = & -1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0 & = & -8+0+2+1 & = & -5 \\
B2T_4([1111]) & = & -1\cdot 2^3 + 1\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0 & = & -8+4+2+1 & = & -1
\end{array} \tag{2.4}$$

Two’s Compliment: Continued

- Wikipedia Example:
  - Convert -6 to binary
  - base10 6 → base2 0110
  - Flip MSbit all bits to represent negative number
    - 1001 → -8 + 1 == -7
  - Add 1
    - 1010 → -8 + 2 = -6

Bits:	1	0	1	0
Decimal bit value:	−8	4	2	1
Binary calculation:	$-(\mathbf{1}\times 2^3)$	$(\mathbf{0}\times 2^2)$	$(\mathbf{1}\times 2^1)$	$(\mathbf{0}\times 2^0)$
Decimal calculation:	$-(\mathbf{1}\times 8)$	**0**	$\mathbf{1}\times 2$	**0**

Two’s Compliment: Continued

- Wikipedia Example:
  - Convert -6 to binary
  - base10 6 → base2 0110
  - Flip MSbit all bits to represent negative number
    - 1001 → -8 + 1 == -7
  - Add 1
    - 1010 → -8 + 2 = -6

Bits:	1	0	1	0
Decimal bit value:	−8	4	2	1
Binary calculation:	$-(\mathbf{1}\times 2^3)$	$(\mathbf{0}\times 2^2)$	$(\mathbf{1}\times 2^1)$	$(\mathbf{0}\times 2^0)$
Decimal calculation:	$-(\mathbf{1}\times 8)$	**0**	$\mathbf{1}\times 2$	**0**

Casting in C/C++

- Implicit casting
  - int x
  - short c
  - if (x == c)
- What is the logical result for values less than 32767 and greater than 32767? (2^15 - 1)
- How does truncation affect these checks?

Casting in C/C++

- Messy behavior.

```
1    int x = 53191;
2    short sx = (short) x;   /* -12345 */
3    int y = sx;             /* -12345 */
```

Signed to Unsigned

- Positive values are easy right?
  - 6 = 6
  - Int == u_int
- Casting works as you would think, mostly.
  - int x = int(u_int);

Signed to Unsigned : Continued

- What to do with two’s complement to unsigned?
  - 1010 == -6 “signed”
  - 1010 == 10 “signed”
- Which is correct?

Addition Overflow

- Signed Overflow

PRINCIPLE: Two's-complement addition

For integer values x and y in the range $-2^{w-1} \leq x, y \leq 2^{w-1} - 1$:

$$x +_w^t y = \begin{cases} x + y - 2^w, & 2^{w-1} \leq x + y \quad \text{Positive overflow} \\ x + y, & -2^{w-1} \leq x + y < 2^{w-1} \quad \text{Normal} \\ x + y + 2^w, & x + y < -2^{w-1} \quad \text{Negative overflow} \end{cases} \tag{2.13}$$

Addition Overflow: Off the end

- Overflow bit lost to the wind

x	y	$x + y$	$x +_4^t y$	Case
−8 [1000]	−5 [1011]	−13 [10011]	3 [0011]	1
−8 [1000]	−8 [1000]	−16 [10000]	0 [0000]	1
−8 [1000]	5 [0101]	−3 [11101]	−3 [1101]	2
2 [0010]	5 [0101]	7 [00111]	7 [0111]	3
5 [0101]	5 [0101]	10 [01010]	−6 [1010]	4

Multiplication Overflow: Off the end

- Truncation is similar
- Machine is not aware of signedness

Mode	x		y		$x \cdot y$		Truncated $x \cdot y$	
Unsigned	5	[101]	3	[011]	15	[001111]	7	[111]
Two's complement	−3	[101]	3	[011]	−9	[110111]	−1	[111]
Unsigned	4	[100]	7	[111]	28	[011100]	4	[100]
Two's complement	−4	[100]	−1	[111]	4	[000100]	−4	[100]
Unsigned	3	[011]	3	[011]	9	[001001]	1	[001]
Two's complement	3	[011]	3	[011]	9	[001001]	1	[001]

Division by bit shifting

- Each shift right is a power of 2
- 500 >> 2 == ?
  - 500 == 1 F 4 == 0001 1111 0100
  - 0000 0111 1101 == 0 7 D == 125
  - >> 2 == 1/4

Signed Division by bitshifting

- Each shift right is a power of 2
- -500 >> 2 == ?
  - -500 == E 0 C == **1**110 0000 1100
  - **11**11 1000 0011 == F 8 3 == -125
  - >> 2 == ¼
- 1s padding?