# 15-213: Introduction to Computer Systems
# Written Assignment 1

This written homework covers Bits, Bytes, Integers.

## Directions

Complete the question(s) on the following pages with single paragraph answers. These questions are not meant to be particularly long! Once you are done, submit this assignment on Canvas.

Below is an example question and answer.

Q: Please describe benefits of two's-complement signed integers versus other approaches.

A: Other representations of signed integers (ones-complement and sign-and-magnitude) have two representations of zero (+0 and −0), which makes testing for a zero result more difficult. Also, addition and subtraction of two's complement signed numbers are done exactly the same as addition and subtraction of unsigned numbers (with wraparound on overflow), which means a CPU can use the same hardware and machine instructions for both.

## Grading

Each assignment will be graded in two parts:

1. Does this work indicate any effort? (e.g. it's not copied from a homework for another class or from the book)
2. Three peers will provide short, constructive feedback.

## Due Date

This assignment is due on September 3rd by 11:59 pm Pittsburgh time (currently UTC−4). Remember to convert this time to the timezone you currently reside in. Because we immediately release a solution grading rubric 12 hours past the deadline, no late submissions will be accepted after this 12 hour grace period. Also note that Canvas will generally accept only one submission.

# Question #1

Before the two's-complement representation of signed binary numbers was invented, some computers used *sign-and-magnitude* representation. In this representation, the most significant bit is *only* the sign of the number (0 for positive, 1 for negative); it has no place value. The remaining bits are interpreted as an unsigned number. For instance, the four-bit number 1011 in sign-and-magnitude represents -3, and 0101 represents 5.

a. If you add the four-bit sign-and-magnitude numbers 0100 and 0101 using unsigned addition, what is the value of the result, interpreted as sign-and-magnitude?

```
-1
```

b. If you add the four-bit sign-and-magnitude numbers 0010 and 1001 using unsigned addition, what is the value of the result, interpreted as sign-and-magnitude?

```
-3
```

c. If you add the four-bit sign-and-magnitude numbers 1010 and 1001 using unsigned addition, what is the value of the result, interpreted as sign-and-magnitude?

```
3
```

d. Describe an algorithm for adding sign-and-magnitude numbers which will make sums involving negative numbers come out correctly. For instance 0011 + 1001 should produce 0010 (2), and 1010 + 1001 should produce 1011 (-3). Don't worry about what happens on overflow in either direction.

```
- Check the signs first.
- If the signs are the same:
  1. Keep the sign bit (the highest bit).
  2. Add the remaining bits like unsigned addition.
  3. Apply the sign to the result.
   For example:
      0101(5) + 0010(2) = 0111 (7)
      1101(-5) + 1010(-2) = 1111(-7).
- If the signs are different:
  1. Compare the absolute value of the numbers
  2. Hold the signs ((the highest bit)
  2. Subtract the smaller value from the larger one
  3. Apply the sign of the number with the larger abs value.
   For example:
      1101(-5) + 0010(2)
      -> Compare absolute value: 5 > 2
      -> 5-2 = 3
      -> Apply sign of -5
      -> Result: -3
```

## Question #2

Explain the process of casting a signed integer to an unsigned integer of the same width, and vice versa. Do so by answering the following questions. How does the value change? Does the bit pattern change? As a hint, there should be two cases to consider.

**signed → unsigned**

Bit pattern: stays the same.
Value:
- If the signed value is non-negative (0 to Tmax), the unsigned value is the same, and vice versa.

- If the signed value is negative (Tmin to -1), the unsigned value is interpreted as a large number, specifically value + 2^n, where n is the bit-width. This maps the negative range Tmin … -1 onto the unsigned range Tmax+1 … Umax.

**unsigned → signed**

Bit pattern: stays the same.

Value:

- If the unsigned value is ≤ Tmax, it is interpreted as the same integer.

- If the unsigned value is > Tmax, it is interpreted as a negative number, specifically value - 2^n.

**The above cases show that, in the overlapping range between signed and unsigned, values can be converted without change. Outside of that range, the mapping is one-to-one: the signed range Tmin … -1 maps onto the unsigned range Tmax+1 … Umax.**