# CS61C Spring 2017 Discussion 0 – Number Representation

## 1 Unsigned Integers

If we have an $n$-digit unsigned numeral $d_{n-1}d_{n-2}\ldots d_0$ in radix (or base) $r$, then the value of that numeral is $\sum_{i=0}^{n-1} r^i d_i$, which is just fancy notation to say that instead of a 10's or 100's place we have an $r$'s or $r^2$'s place. For binary, decimal, and hex we just let $r$ be 2, 10, and 16, respectively.

Recall also that we often have cause to write down unreasonably large numbers, and our preferred tool for doing that is the IEC prefixing system: Ki = $2^{10}$, Mi = $2^{20}$, Gi = $2^{30}$, Ti = $2^{40}$, Pi = $2^{50}$, Ei = $2^{60}$, Zi = $2^{70}$, Yi = $2^{80}$.

### 1.1 We don't have calculators during exams, so let's try this by hand

1. Convert the following numbers from their initial radix into the other two common radices: 0b10010011, 0xD3AD, 63, 0b00100100, 0xB33F, 0, 39, 0x7EC4, 437

2. Write the following numbers using IEC prefixes: $2^{16}$, $2^{34}$, $2^{27}$, $2^{61}$, $2^{43}$, $2^{47}$, $2^{36}$, $2^{58}$.

3. Write the following numbers as powers of 2: 2 Ki, 256 Pi, 512 Ki, 64 Gi, 16 Mi, 128 Ei

## 2 Signed Integers

Unsigned binary numbers work for natural numbers, but many calculations use negative numbers as well. To deal with this, a number of different schemes have been used to represent signed numbers, but we will focus on two's complement, as it is the standard solution for representing signed integers.

### 2.1 Two's complement

- Most significant bit has a negative value, all others are positive. So the value of an $n$-digit two's complement number can be written as $\sum_{i=0}^{n-2} 2^i d_i - 2^{n-1} d_n$.
- Otherwise exactly the same as unsigned integers.
- A neat trick for flipping the sign of a two's complement number: flip all the bits and add 1.
- Addition is exactly the same as with an unsigned number.
- Only one 0, and it's located at 0b0.

### 2.2 Exercises

For questions 1 – 3, assume an 8 bit integer and answer each one for the case of a two's complement number and unsigned number, indicating if it cannot be answered with a specific representation.

1. What is the largest integer? The largest integer + 1?

2. How do you represent the numbers 0, 1, and -1?

3. How do you represent 17, -17?

4. What is the largest integer that can be represented by *any* encoding scheme that only uses 8 bits?

5. Prove that the two's complement inversion trick is valid (i.e. that $x$ and $\overline{x} + 1$ sum to 0).

6. Explain where each of the three radices shines and why it is preferred over other bases in a given context.

## 3 Counting

Bitstrings can be used to represent more than just numbers. In fact, we use bitstrings to represent *everything* inside a computer. And, because we don't want to be wasteful with bits it is important that to remember that $n$ bits can be used to represent $2^n$ distinct things. For each of the following questions, answer with the minimum number of bits possible.

### 3.1 Exercises

1. How many bits do we need to represent a variable that can only take on the values $0, \pi$ or $e$?

2. If we need to address 3 TiB of memory and we want to address every byte of memory, how long does an address need to be?

3. If the only value a variable can take on is $e$, how many bits are needed to represent it.