CS 61C Spring 2023

Number Representation

Discussion 1

1 Pre-Check

This section is designed as a conceptual check for you to determine if you conceptually understand and have any misconceptions about this topic. Please answer true/false to the following questions, and if false, correct the statement to make it true:

1.1 Depending on the context, the same sequence of bits may represent different things.

1.2 It is possible to get an overflow error in Two's Complement when adding numbers of opposite signs.

1.3 If you interpret a N bit Two's complement number as an unsigned number, negative numbers would be smaller than positive numbers.

1.4 If you interpret an N bit Bias notation number as an unsigned number (assume there are negative numbers for the given bias), negative numbers would be smaller than positive numbers.

1.5 We can represent fractions and decimals in our given number representation formats (unsigned, biased, and Two's Complement).

2 Unsigned Integers

2.1 If we have an n-digit unsigned numeral $d_{n-1}d_{n-2}\dots d_0$ 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 the three radices binary, decimal, and hex, we just let r be 2, 10, and 16, respectively.

Let's try this by hand.

(a) Convert the following numbers from their initial radix into the other two common radices:

1. 0b10010011
2. 63
3. 0b00100100
4. 0
5. 39

6. 437
7. 0x0123

(b) Convert the following numbers from hex to binary:

1. 0xD3AD
2. 0xB33F
3. 0x7EC4

2.2 Our preferred tool for writing large numbers is the IEC prefixing system, which is similar to scientific notation but with powers of 2 rather than 10:

Ki (Kibi) $= 2^{10}$	Gi (Gibi) $= 2^{30}$	Pi (Pebi) $= 2^{50}$	Zi (Zebi) $= 2^{70}$
Mi (Mebi) $= 2^{20}$	Ti (Tebi) $= 2^{40}$	Ei (Exbi) $= 2^{60}$	Yi (Yobi) $= 2^{80}$

For example, we would write 2^{81} as $2 * 2^{80} = 2$ Yi.

(a) Write the following numbers using IEC prefixes:

- 2^{16}
- 2^{27}
- 2^{43}
- 2^{36}
- 2^{34}
- 2^{61}
- 2^{47}
- 2^{59}

(b) Write the following numbers as powers of 2:

- 2 Ki
- 512 Ki
- 16 Mi
- 256 Pi
- 64 Gi
- 128 Ei

3 Signed Integers

3.1 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. Here are two common schemes:

Two's Complement:

- We can write the value of an n-digit two's complement number as $\sum_{i=0}^{n-2} 2^i d_i - 2^{n-1} d_{n-1}$.
- Negative numbers will have a 1 as their most significant bit (MSB). Plugging in $d_{n-1} = 1$ to the formula above gets us $\sum_{i=0}^{n-2} 2^i d_i - 2^{n-1}$.
- Meanwhile, positive numbers will have a 0 as their MSB. Plugging in $d_{n-1} = 0$ gets us $\sum_{i=0}^{n-2} 2^i d_i$, which is very similar to unsigned numbers.
- To negate 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.

Biased Representation:

- The number line is shifted so that the smallest number we want to be representable would be 0b0...0.
- To find out what the represented number is, read the representation as if it was an unsigned number, then add the bias.
- We can shift to any arbitrary bias we want to suit our needs. To represent (nearly) as much negative numbers as positive, a commonly-used bias for N bits is $-(2^{N-1} - 1)$.

For questions (a) through (c), assume an 8-bit integer and answer each one for the case of an unsigned number, biased number with a bias of -127, and two's complement number. Indicate if it cannot be answered with a specific representation.

(a) What is the largest integer? What is the result of adding one to that number?

1. Unsigned?
2. Biased?
3. Two's Complement?

(b) How would you represent the numbers 0, 1, and -1?

1. Unsigned?
2. Biased?
3. Two's Complement?

(c) How would you represent 17 and -17?

1. Unsigned?
2. Biased?
3. Two's Complement?

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

3.3 We now have three major radices (or bases) that allow us to represent numbers using a finite amount of symbols: binary, decimal, hexadecimal. Why do we use each of these radices, and why are each of them preferred over other bases in a given context?

4 Arithmetic and Counting

4.1 Addition and subtraction of binary/hex numbers can be done in a similar fashion as with decimal digits by working right to left and carrying over extra digits to the next place. However, sometimes this may result in an overflow if the number of bits can no longer represent the true sum. Overflow occurs if and only if two numbers with the same sign are added and the result has the opposite sign.

(a) Compute the decimal result of the following arithmetic expressions involving 6-bit Two's Complement numbers as they would be calculated on a computer. Do any of these result in an overflow? Are all these operations possible?

1. 0b011001 − 0b000111
2. 0b100011 + 0b111010
3. 0x3B + 0x06
4. 0xFF − 0xAA
5. 0b000100 − 0b001000

(b) What is the least number of bits needed to represent the following ranges using any number representation scheme?

1. 0 to 256
2. -7 to 56
3. 64 to 127 and -64 to -127
4. Address every byte of a 12 TiB chunk of memory

(c) How many distinct numbers can the following schemes represent? How many distinct *positive* numbers?

1. 10-bit unsigned
2. 8-bit Two's Complement
3. 8-bit One's Complement
4. 6-bit biased, with a bias of -30
5. 10-bit sign-magnitude