CS 61C Summer 2020

Number Representation

Discussion 1: June 22th, 2020

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 include an explanation:

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

True

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

False

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

False

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.

True

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. Recall that our preferred tool for writing large numbers is the IEC prefixing system:

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) $= 2^{80}$

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

1. 0b10010011 0x93 147
2. 63 0b00111111 0x3F
3. 0b00100100 0x24 36
4. 0 0x00 0b00000000
5. 39 0b00100111 0x27
6. 437 0b000110110101 0x1B5
7. 0x0123 0b0000000100100011 1+2+32+256= 291

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

1. 0xD3AD 0b 1101 0011 1010 1101
2. 0xB33F 0b 1011 0011 0011 1111
3. 0x7EC4 0b 0111 1110 1100 0100

(c) Write the following numbers using IEC prefixes:

- 2^{16} 64 Ki
- 2^{27} 128 Mi
- 2^{43} 16 Ti
- 2^{36} 64 Gi
- 2^{34} 16 Gi
- 2^{61} 2 Ei
- 2^{47} 128 Ti
- 2^{59} 512 Pi

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

- 2 Ki 2 2^20 = 2^21
- 512 Ki 2^9* 2^10 = 2^19
- 16 Mi 2^4 * 2^20 = 2^24
- 256 Pi 2^8 * 2^50 = 2^58
- 64 Gi 2^6 * 2^10 = 2^16
- 128 Ei 2^7 * 2^60 = 2^67

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, but we will focus on two's complement, as it is the standard solution for representing signed integers.

- 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 $\sum_{i=0}^{n-2} 2^i d_i - 2^{n-1} d_{n-1}$.
- 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.

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? 2^8 = 256 Adding one overflow's from 0b11111111 -> 0b00000000 = 0
2. Biased? 2^8 - 127 = 129 Adding one overflow's from 0b11111111 -> 0b00000000 = -127
3. Two's Complement? 0b 01111111 = 127 Adding one overflow's from 0b01111111 -> 0b10000000 = -128

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

1. Unsigned? 0b 00, 0b 01, you can't represent -1
2. Biased? 0b 01111111 = 127 - 127 =0, 0b10000000 = 128-127 = 1, 0b01111110 = 126-127 = -1
3. Two's Complement? 0b 00, 0b 01, 0b 11111111 = 127 - 128 = -1

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

1. Unsigned? 0b 00010001, you can't
2. Biased? 144 - 127 = 128 + 16 - 127 = 0b00101000
3. Two's Complement? 0b00010001, 0b11101111

(d) What is the largest integer that can be represented by *any* encoding scheme that only uses 8 bits?

255, but actually "no such integer" as bias can be however large you want

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

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

Decimals, good for natural numbers, readable and intuitive to anyone
Binary is great for on/off switches, so useful for electrical circuits and CS
Hexadecimal, more readable binary shorthand

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 Won't overflow, pos + neg never overflows in 2's complement
2. 0b100011 + 0b111010 may overflow, as 2 negatives sometimes overflow in 2's complement
3. 0x3B + 0x06 This overflows
4. 0xFF − 0xAA Not possible, can't be represented in 6 bit. Does not overflow in 8 bit

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

1. 0 to 256 9 bits, 8 bits only gets 256 possible numbers, 0-256 is 257 numbers
2. -7 to 56 64 possible numbers, so 6 bits with a bias of -7 is the easiest
3. 64 to 127 and -64 to -127 64 possible unsigned numbers, so 6 bits if representing 1, 7 bits if you want to add a sign
4. Address every byte of a 12 TiB chunk of memory

12 needs 2^4, and Tebi needs 2^40, so we need 2^44 bytes