CS61c Midterm Review (fa06)

Number representation and Floating points

☺ From your friendly reader ☺

Number representation (See: Lecture 2, Lab 1, HW#1)

- KNOW: Kibi (2^{10}), Mebi(2^{20}), Gibi(2^{30}), Tebi(2^{40}), Pebi(2^{50}), Exbi(2^{60}), Zebi(2^{70}), Yobi (2^{80})
- Know the Binary (0b), Octal (0), Decimal, Hexadecimal (0x) representation of numbers.
  - Example: $1111_2 = 17_8 = 15_{ten} = F_{hex}$
    or 0b1111 = 017 = 15 = 0xF
- Know how to convert from one base to another. Example: $104_5 = 4*5^0+0*5^1+1*5^2 = 29_{ten}$
- Know what is a bit, a Nibble, a Byte, a Word
  - Example: above example uses a Nibble!
- The lecture presents the evolution of number representation: sign and magnitude, one's complement, two's complement

  - **Sign and Magnitude:**

  *Left most bit represents the sign (0 == positive, 1 == negative), the rest represents magnitude.
  *In order to negate, flip the left most bit.
  * Can represent from $-(2^{N-1}-1)$ to $2^{N-1}-1$

  Cons:

  *We have two zeros, namely 0x00000000 and 0x80000000
  *Arithmetic is non-trivial
  *Number comparison is non trivial (-2 > -1)

  - **One's Complement**

  *Left most bit represents the sign.
  *In order to negate, must flip all the bits.
  * Can represent from $-(2^{N-1}-1)$ to $2^{N-1}-1$

  Cons:

  * Still have two zeros.
  * (x + (-x)) != 0, thus arithmetic is still complicated.

  - **Two's Complement** (standard)

  *Left most bit represents the sign.
  *In order to negate must flip bits and add 1. ➔ -x = flip_bits(x) +1
  * Can represent from -2^{N-1} to $2^{N-1}-1$
- Unsigned numbers can represent from 0 to 2^N-1
- Know overflow: there isn't enough bits to express a number. E.g: MAX+1 causes overflow
- Big-Endian vs Little-Endian
  - Refers to the order in which BYTES are stored in memory
  - bits of the byte are stored as usual
  - **Big-Endian:** Big Units first (are on the left)
    Example: today's date representation in big-endian is 06/10/15

- **Little-Endian:** Little Units first.
  Example: today's date representation in little-endian is 15/10/06

Floating points (See: Lecture 15 and 16, Lab 6)

- S Exponent Significand | rounding_bits
  - Significand also known as the Mantissa
  - **Bias:** in order to be able to represent tiny and huge values, usually is $2^{e-1}-1$ for normalized numbers, where e is number of bits for E
  - **Rounding bits:** there are usually extra bits tagged on after the Significand in order to correct for rounding errors when doing arithmetic
  - **Single Precision** (32 bits): S is **1** bit, E is **8** bits, S is **23** bits, bias = 2^7-1 = **127**
  - **Double Precision** (64 bits): S is **1** bit, E is **11** bits, S is **52** bits, bias = 2^8-1 = **1023**

- Exponent size vs. Signficand size ≡ range vs. precision

- Important to know all types (Norm, zero, Denorm, ± ∞ , NaN)
  Hint: Green sheet can be very useful!
  Single Precision discussed below: (* ≡ anything, S is Sign, E is Exponent, M is Mantissa)
  - **Norm**:
    - S = *, $0 < E < 2^8-1$, M = *
    - Form: $(-1)^S*2^{(E-127)}$ *(1+Mantissa)
  - **Zero:**
    - S = *, E = 0, M = 0
  - **Denorm**
    - Exponent (implied bias): -126
    - S = *, E = 0, M>0
    - Form: $(-1)^S*2^{(-126)}$ *(0+ Mantissa)
  - ± ∞
    - S = *, $E = 2^8-1$ (all ones), M = 0
  - **NaN:**
    - S = *, $E = 2^8-1$ (all ones), M > 0

  Double Precision is analogous to above, with exception of E range and the bias (implied bias for Denorm is -1022)

- **Floating point addition:**
  - See ***http://www.ecs.umass.edu/ece/koren/arith/simulator/FPAdd/*** for a good simulation for floating point addition/subtraction. Make sure you understand what is going on!
  - Steps:
    - Align exponents (biggest exponent wins!)
    - Normalize the result
    - Round
      - Extra bits tagged on at the end of the 32bits (64bits) for rounding error

- Types: Round to zero, Round to nearest even, Round to $+\infty$, etc.

Somewhat-trivial Questions:

1) What is 0xffff ffff in decimal form (you can use $*2^y$ in your answer)
   - Unsigned int: _____
   - Sign and magnitude: _____
   - One's complement: _____
   - Two's complement: _____

2) J-Lo, who is thirty-six of age (as of fa05), whispers her age to you: "Thirty six". Was that big-endian or little-endian (mt fa05)

3) Let x = 0xffff ffff. What is the decimal value (if any) using Single Precision floats?

4) Let x = 0x0040 0000. What is the decimal value (if any) using Single Precision floats?

5) What is a mebi?

Conceptual/Midterm type Questions:

6) Let x = 0x4000 0000 and y = 0x0040 0000 be *floats*. What is x+y?

7) What is minimum positive *float x* such that x + 1 = x (You've done this before in lab, and this *has* been on the Midterm before!)

8) True/False: *Float* arithmetic is associative. i.e. (x+y)+z = x+(y+z) . If not, show a counterexample.
9) Let *float* x=2^{20}, find *float* y such that x+y have 0xb11 in the rounding bits on the alignment step (assume there are 2 such bits)
   i.e. x = 1.********** ********** *** | 00 * 2^n
   y = *.********** ********** *** | 00 * 2^n
   x + y = *.********** ********** *** | 11 * 2^n

- Compute x+y, using truncation.

- Compute x+y using “Round to zero”

- Compute x+y using “round to Plus Infinity”

From Dan’s previous Midterm (fa05)
(Perhaps *b* should be done before *a* ☺)

b) What is min, max, and positive min (greater than 0) for each of the types specified above?

Type:	Min	max	positive (normalized) min (>0)
Unsigned int			
Sign-Magnitude			
2s complement			
Float			

c) What is the largest integer that a float/double can store?

d) What is the largest integer that float/double can’t store?