CS61C Spring 2015 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 large numbers, and our preferred tool for doing that is the IEC prefixing system:

- Kilo- (Ki) $= 2^{10} \approx 10^3$
- Mega- (Mi) $= 2^{20} \approx 10^6$
- Giga- (Gi) $= 2^{30} \approx 10^9$
- Tera- (Ti) $= 2^{40} \approx 10^{12}$
- Peta- (Pi) $= 2^{50} \approx 10^{15}$
- Exa- (Ei) $= 2^{60} \approx 10^{18}$
- Zetta- (Zi) $= 2^{70} \approx 10^{21}$
- Yotta- (Yi) $= 2^{80} \approx 10^{24}$

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:

   (a) 0b10010011 = 147 = 0x93

   (b) 0xD3AD = 0b1101 0011 1010 1101 = 54189

   (c) 63 = 0b0011 1111 = 0x3F

   (d) 0b00100100 = 36 = 0x24

   (e) 0xB33F = 0b1011 0011 0011 1111 = 45887

   (f) 0 = 0b0 = 0x0

   (g) 0x7EC4 = 0b0111 1110 1100 0100 =32452

   (h) 437 = 0b0001 1011 0101 = 0x1B5

2. Write the following numbers using IEC prefixes:

   (a) 2^{16}= 64 Ki

   (b) 2^{34}= 16 Gi

   (c) 2^{27}= 128 Mi

   (d) 2^{61}= 2 Ei

   (e) 2^{43}= 8 Ti

   (f) 2^{47}= 128 Ti

   (g) 2^{36}= 64 Gi

   (h) 2^{58}= 256 Pi

3. Write the following numbers as powers of 2:

   (a) 2 Ki = 2^{11}

   (b) 256 Pi = 2^{58}

   (c) 512 Ki = 2^{19}

   (d) 64 Gi = 2^{36}

   (e) 16 Mi = 2^{24}

   (f) 128 Ei = 2^{67}

2 Signed Integers w/ Two's Complement

- Two's complement is the standard solution for representing signed integers.
  - Most significant bit has a negative value, all others have positive.
  - 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.

2.1 Exercises

For the following questions assume an 8 bit integer. Answer each question for the case of a two's complement number and an unsigned number.

1. What is the largest integer? The largest integer + 1?
   (a) [Unsigned:] 255, 0
   (b) [Two's Complement:] 127, -128

2. How do you represent the numbers 0, 1, and -1?
   (a) [Unsigned:] 0b0000 0000, 0b0000 0001, N/A
   (b) [Two's Complement:] 0b0000 0000, 0b0000 0001, 0b1111 1111

3. How do you represent 17, -17?
   (a) [Unsigned:] 0b0001 0001, N/A
   (b) [Two's Complement:] 0b0001 0001, 0b1110 1111

4. What is the largest integer that can be represented by *any* encoding scheme that only uses 8 bits? There is no such integer. For example, an arbitrary 8-bit mapping could choose to represent the numbers from 1 to 256 instead of 0 to 255.

5. Prove that the two's complement inversion trick is valid (i.e. that x and $\overline{x}+1$ sum to 0). Note that for any x we have $x+\overline{x}=$ 0b1...1. A straightforward hand calculation shows that $0\text{b}1\ldots1+0\text{b}1=0$.

6. Explain where each of the three radices shines and why it is preferred over other bases in a given context. Decimal is the preferred radix for human hand calculations, likely related to the fact that humans have 10 fingers.

   Binary numerals are particularly useful for computers. Binary signals are less likely to be garbled than higher radix signals, as there is more "distance" (voltage or current) between valid signals. Additionally, binary signals are quite convenient to design circuits with, as we'll see later in the course.

   Hexadecimal numbers are a convenient shorthand for displaying binary numbers, owing to the fact that one hex digit corresponds exactly to four binary digits.

3 Counting

Bitstrings can be used to represent more than just numbers. In fact, we use bitstrings to represent *everything* inside a computer. Two key facts to remember about binary counting:

- With n bits, we can count or represent 2^n things. [Why? Because we have 2 possible values for the each bit (0 or 1) $\implies$ $2 \cdot 2 \cdots 2$ (n times) $= 2^n$ different bitstrings]
- Equivalently, to count or represent k things, we need $n = \lceil \log_2 k \rceil$ bits.

3.1 Exercises

1. If the only values a variable can take on are $0, \pi$ or e, what is the minimum number of bits needed to represent it?

   2. there are 3 things to represent, so we need $n = \lceil \log_2 3 \rceil = 2$ bits

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? 42 bits.

   3 TiB $= 3 \cdot 2^{40}$

   $\lceil \log_2(3 \cdot 2^{40}) \rceil$

   $= \lceil \log_2 3 + \log_2 2^{40} \rceil$

   $= \lceil 1.58 + 40 \rceil = 42$

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

   0. If there's only 1 thing to represent, $\log_2 1 = 0$. Intuitively, we interpret this to mean the variable's value is fixed, so we don't need additional bits to label any different states of that variable (because there are none).