Jon 112 Dis 1

Monday, August 27, 2018 2:27 PM

CS 61C Fall 2018

Number Representation

Discussion 1: August 27, 2018

Notes

1 Unsigned Integers

1.1 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 the three radices, binary, decimal, and hex, we just let r be 2, 10, and 16, respectively.

We don't have calculators during exams, so 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 $= 1\cdot 2^0 + 1\cdot 2^1 + 1\cdot 2^4 + 1\cdot 2^7 = 147 = 0x93$
2. 63 = 0b00111111 = 0x3F
3. 0b00100100
4. 0
5. 39
6. 437
7. 0x0123

$63/2 = 31$ R1
$31/2 = 15$ R1
$15/2 = 7$ R1
$7/2 = 3$ R1
$3/2 = 1$ R1
$1/2 = 0$ R1

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

1. 0xD3AD = 0b 1101 0011 1010 1101
2. 0xB33F
3. 0x7EC4

(c) Write the following numbers using IEC prefixes:

group by powers of 10

- 2^{16} — $2^6 \cdot 2^{10}$ = 2^6Ki = 64Ki
- 2^{27} — $2^7 \cdot 2^{20}$ = 128 Mi
- 2^{43}
- 2^{36}
- 2^{34}
- 2^{61}
- 2^{47}
- 2^{58}

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

- 2 Ki — $2 \cdot 2^{10} = 2^{11}$
- 512 Ki — $2^9\, 2^{10} = 2^{19}$
- 16 Mi
- 256 Pi
- 64 Gi
- 128 Ei

binary ↔ decimal
binary ↔ hex
decimal ↔ hex X

B→D
$\sum \text{digit} \cdot 2^{\text{place}}$

11010 ← lowest: 0
$0\cdot 2^0 + 1\cdot 2^1 + 0\cdot 2^2 + 1\cdot 2^3 + 1\cdot 2^4$
$= 0 + 2 + 0 + 8 + 16$
$= 26$

D→B
75 div by 2, find R

$75/2 =$	37	R 1
$37/2 =$	18	R 1
$18/2 =$	9	R 0
$9/2 =$	4	R 1
$4/2 =$	2	R 0
$2/2 =$	1	R 0
$1/2 =$	0	R 1

1001011

B↔H
group binary into 4
1101 0011
0011 → 3
1101 → D
0xD3

Binary	Hex
0000	0
0001	1
⋮	⋮
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Sign + mag ✗
Two's comp
Bias

2 Signed Integers

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

Two's Comp
+: 0b0[mag]
→ + value
−: 0b1[~~]
flip bits, add 1
→ new_mag
− new_mag

Bias
Unsigned shifted (pos)
Unsignd # − Bias

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? The largest integer's representation + 1? 0b1

1. Unsigned? 0b1111 1111 = 255 +1 → 0b1 0000 0000 0b0000 0000 → 0
2. Biased? -127 small, 255 - 127 = 128
3. Two's Complement? 0b0111 1111 = 127 0b1000 0000 -128

(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?

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

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

$0b10110111 = x \quad \overline{x}\,?$

$+\ 0b01001000 = \overline{x}$

$0b11111111$

$+\ 0b\quad 1$

$1\underbrace{00000000} = 0b0000\,0000$

$x + (\overline{x} + 1) = 0$

$= \text{"} - x \text{"}$

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

3 Counting

3.1 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.

(a) How many bits do we need to represent a variable that can only take on the values 0, π or e?

(b) 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?

(c) If the only value a variable can take on is e, how many bits are needed to represent it?