Jon Dis 1 Notes

Monday, August 27, 2018 2:27 PM

CS 61C Fall 2018

Number Representation

Discussion 1: August 27, 2018

$\square$ = individual / groups

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 = 0x93 = $2^0 + 2^1 + 2^4 + 2^7$ = 147 (1, 2, 16, 128)
2. $\boxed{2.}$ 63 = 0b111111 = 0b0011 1111 = 0x3F
3. $\boxed{3.}$ 0b00100100 = 0x24 = $2^2 + 2^5$ = 36 (4, 32)
4. 0 0b0 0x0
5. 39
6. 437
7. 0x0123

} home

2)

63/2	31	1
31/2	15	1
15/2	7	1
7/2	3	1
3/2	1	1
1/2	0	1

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

1. 0xD3AD 0x 1101 0011 1010 1101
2. 0xB33F
3. 0x7EC4

} home

Group by powers of 10 →

(c) Write the following numbers using IEC prefixes:

- 2^{16} = $2^6 \cdot 2^{10}$ 64Ki
- 2^{27} = $2^7 \cdot 2^{20}$ 128Mi
- $\boxed{\cdot}$ 2^{43} = $2^3 \cdot 2^{40}$ 8Ti
- $\boxed{\cdot}$ 2^{36} = $2^6 \cdot 2^{30}$ 64Gi
- 2^{34}
- 2^{61}
- 2^{47}
- 2^{58}

home

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

- 2 Ki $2 \cdot 2^{10} = 2^{11}$
- $\boxed{\cdot}$ 512 Ki $2^9 \cdot 2^{10} = 2^{19}$
- $\boxed{\cdot}$ 16 Mi $2^4 \cdot 2^{20} = 2^{24}$
- 256 Pi
- 64 Gi
- 128 Ei

home

With n bits, we can represent 2^n unique things.

Notes

1) Binary ⟷ Decimal

B → D

digit $\cdot 2^{\text{place}}$

ex. 11010

$= 0 \cdot 2^0 + 1 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3 + 1 \cdot 2^4$

$= 0 + 2 + 0 + 8 + 16$

$= 26$

D → B

divide by 2 + Remainder

ex. 75

	#	R
75/2	37	1
37/2	18	1
18/2	9	0
9/2	4	1
4/2	2	1
2/2	1	0
1/2	0	1

1001011

Dec ⟷ Hex :(

Hex ⟷ Binary :)

Group by 4 → convert

ex 11010011

1101 → D

0011 → 3

} 0xD3

or reverse

0xD3

D → 1101

3 → 0011

} 0b11010011

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

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.

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?

1. Unsigned? 255 , 0
2. Biased? 128 , -127
3. Two's Complement? 127 , -128

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

1. Unsigned? 0b00000000 0b00000001 N/A
2. Biased? 0b01111111 0b10000000 0b01111110
3. Two's Complement? 0b00000000 0b00000001 0b11111111

0b00000001
0b11111110 ← flip
0b11111111 ← add 1

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

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

} home

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

anything all up to interpretation

Can do (−) now!

~~Sign + value~~
~~0 111 ⇒ +7~~
~~1 111 ⇒ −7~~
~~two zeros~~
~~0000~~
~~1000~~
~~Bad!~~

Bias
Unsigned value − Bias
Bias typically $\left(\frac{X}{2} - 1\right)$
for representable range X
ex. 4 bits, bias = 7 $\left(\frac{2^4}{2} - 1\right)$
→ 0000 = −7
0111 = 0
1111 = 8

Two's Complement
0b0 1011001
"sign" "value"
+ → + value
− → flip bits, add 0b00...01 ← 1 in bin
− new_magnitude

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

$x + (\overline{x} + 1) = 0$ → 0b1001001

$$
\begin{array}{r}
0b1001\,0001 \\
+\ 0b0110\,1110 \\
\hline
0b1111\,1111 \\
+\qquad\qquad 1 \\
\hline
\cancel{1}0000\,0000
\end{array}
$$

overflow

→ 0b0000 0000

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

home

3 Counting

cut

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?