

# CS 199: Lecture 2

Data Representation:  
Numbers and Text

# Review

- Binary numbers
- Binary addition

# Why Binary?

- Because 2 is special! It's the smallest number that's not 1.
- So simple a computer could do it!
- On/off, long/short (Morse Code), light/dark (CDs), high/low (pitches, voltage)

5 1000

1 100

3 10

7 1



decimal digits



binary digits



1 0 1 1 1

$$16 + 4 + 2 + 1 = 23$$

1 1 0 1

$$8 + 4 + 1 = 13$$

0

16

8

4

2

1

1

16

8

4

2

1

2

16

8

4

2

1

3

16

8

4

2

1

4

16

8

4

2

1

5

16

8

4

2

1

6

16

8

4

2

1

7

16

8

4

2

1

8

16

8

4

2

1

9

16

8

4

2

1

10

16

8

4

2

1

11

16

8

4

2

1

12

16

8

4

2

1

13

16

8

4

2

1

14

16

8

4

2

1

15

16

8

4

2

1

16

16

8

4

2

1

17

16

8

4

2

1

18

16

8

4

2

1

19

16

8

4

2

1

20

16

8

4

2

1

21

16

8

4

2

1

22

16

8

4

2

1

23

16

8

4

2

1

24

16

8

4

2

1

25

16

8

4

2

1

26

16

8

4

2

1

27

16

8

4

2

1

28

16

8

4

2

1

29

16

8

4

2

1

30

16

8

4

2

1

31

16

8

4

2

1

0

16

8

4

2

1

# Other Bases

$$\begin{aligned}\text{Base 3: } 2012 &= 2 \times 3^3 + 0 \times 3^2 + 1 \times 3^1 + 2 \times 3^0 \\ &= 2 \times 27 + 0 \times 9 + 1 \times 3 + 2 \times 1 \\ &= 54 + 0 + 3 + 2 \\ &= 59\end{aligned}$$

# Other Bases

Base 16... what are the numerals ?

Base 2: {0,1}

Base 3: {0,1,2}

Base 10: {0,1,2,3,4,5,6,7,8,9}

Base 16: {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}

$$\begin{aligned} 1CF &= 1 \times 16^2 + 12 \times 16^1 + 15 \times 16^0 \\ &= 256 + 192 + 15 = 463 \end{aligned}$$

# Binary and Hexadecimal

0	0000	8	1000
1	0001	9	1001
2	0010	10 A	1010
3	0011	11 B	1011
4	0100	12 C	1100
5	0101	13 D	1101
6	0110	14 E	1110
7	0111	15 F	1111

- How do we write 463 in Binary?
- $463 = 256 + 128 + 64 + 8 + 4 + 2 + 1$
- $463 = 111001111$
- $463 = 1 \ 1100 \ 1111$
- $463 = 1 \ C \ F$  in Hex

# Binary Subtraction

Decimal Subtraction

8423  
6915

-----

Binary Subtraction

1101011  
1001101

-----

What about  $1001101 - 1101011$ ?  
How do we represent negative numbers?

# Representations are Arbitrary!



When I use a word, it means just what I choose it to mean, neither more nor less.

'When I use a word,' Humpty Dumpty said, in a rather scornful tone, 'it means just what I choose it to mean, neither more nor less.'

'The question is,' said Alice, 'whether you *can* make words mean so many different things.'

'The question is,' said Humpty Dumpty, 'which is to be master - that's all.'

# Properties of a Good Representation

- Should be simple
- We should be able to do useful things with it.
- Should be efficient
- Recovering from (minor) errors should be easy.

# Negative Numbers

- How would we represent -5 in binary?

- Sign-magnitude

$$+ 5 = (+) 101$$

$$- 5 = (-) 101$$

- What about 0?

# Using 1 digit

- Suppose we had the symbols 0 through 9. How many (positive and negative) numbers can we represent?
- Which symbol gets which number?

# 10's complement

-5	-4	-3	-2	-1	0	1	2	3	4
<hr/>									
5	6	7	8	9	0	1	2	3	4

Now, let's bend the rules and engage in circular reasoning....

# 10's complement

- Addition is moving forward along the number line, subtraction is moving back
- $A - B = A + (-B)$
- Clock Diagram

# Using 3 digits

- If we had 3 digits, which positive and negative numbers could we represent?
- What would the representation of -7 be?
- 1000's complement

# Moving to Binary

- Using 4 bits, which positive and negative numbers can we represent?
- What is the representation of -5?
- Given a binary number, how can we quickly find out what number it represents?
- 2's Complement

# Real Numbers

16

8

4

2

1

$1/2$

$1/4$

$1/8$

$1/16$

$1/32$

# Scientific Notation

- $351200000 = 3.512 \times 10^8$
- $0.0002075 = 2.075 \times 10^{-4}$
  
- $350000 = 3.5 \times 10^5$
- $3.5 = 11.1$  in binary
  
- We represent the *mantissa* (sign/mag) and the *exponent* (sign/mag)

# Text

- How many symbols can we represent using  $n$  binary digits?
- How many text symbols do we need to represent?
- 7-digit ASCII, EBCDIC

# ASCII

- American Standard Code for Information Interchange
- A-Z = 65-90                      (65=100 0001)
- a-z = 97-122                      (97=110 0001)
- 0-9 = 48-57                      (011 0000) to (011 1001)
- What about that 8th bit (to get a *byte*) ??
- *parity check*

# Extending ASCII: Unicode

- With more bits, we can represent more symbols.
- Hangul, Zhuyin, Devanagari,
- Mongolian, Cherokee, Canadian Aboriginal Syllabics, Tifinagh (Berbers), Osmanya (Somali)
- Ogham (Irish), Cuneiform, Klingon, Tolkien