DATA REPRESENTATION (CONTD)

Problem Solving with Computers-I

https://ucsb-cs16-wi17.github.io/

Recap: Representation of non-negative numbers

- Positional encoding
- External representation
- Internal representation
- Binary representation:
  - Only two symbols: 0 and 1
  - Each position is called a *bit*
  - 8 bits makes a byte
  - Bits take up space

$1\ 0\ 2\ A$

$16^3\ \ 16^2\ \ 16^1\ \ 16^0$

$1\ 1\ 0\ 1\ 0$

$16\ 8\ 4\ 2\ 1$

$= 16 + 8 + 2 = 26_{10}$

Converting between binary and decimal

Binary to decimal: $1\ 0\ 1\ 1\ 0_2 = ?_{10}$

16 8 4 2 1

$16 + 4 + 2 = 22_{10}$

Decimal to binary: $34_{10} = ?_2$

$-\ 32$

2

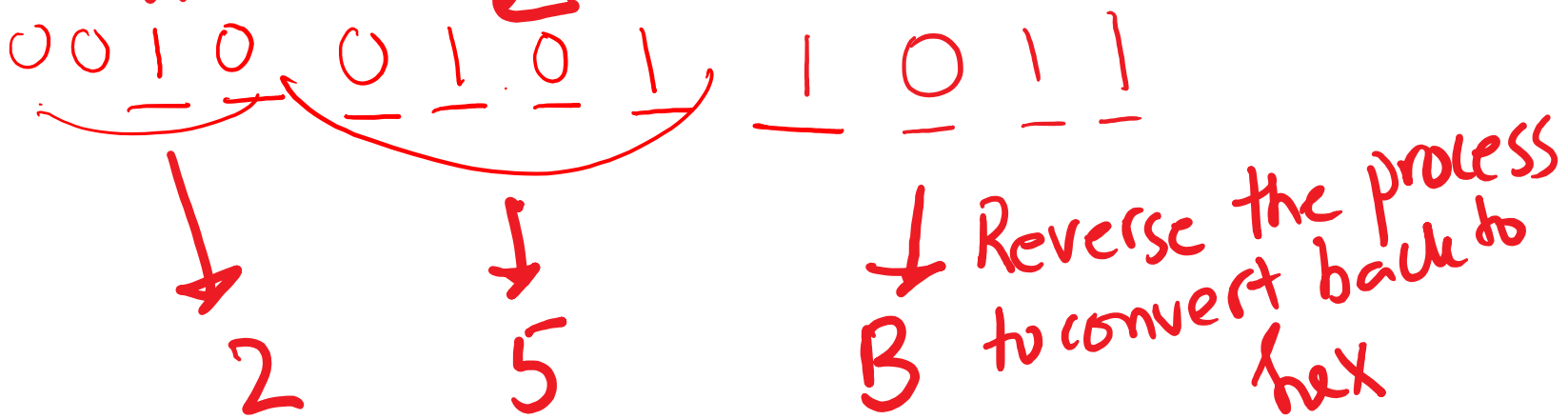
1 0 0 0 1 0

32 16 8 4 2 1

Hex to binary

- Each hex digit corresponds directly to four binary digits
- Programmers love hex, why?
  - * Compact representation for large numbers
  - * Easy conversion to binary

$25B_{16}$ = ? In binary



00	0	0000
01	1	0001
02	2	0010
03	3	0011
04	4	0100
05	5	0101
06	6	0110
07	7	0111
08	8	1000
09	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Hexadecimal to decimal

$25B_{16}$ = ? Decimal

11

256 16 1

2 * 256 + 5 * 16 + 11 * 1

= 603

Hexadecimal to decimal

- Use polynomial expansion

0X stands for hex representation

- $25B_{16}$ = 2*256 + 5*16 + 11*1 = 512 + 80 + 11

  = 603

- Decimal to hex: $36_{10}=?_{16}$

32
4

0 2 4
256 16 1

0X24

Decimal vs. Hexadecimal vs. Binary

Examples:

1010 1100 0011 (binary)
= 0xAC3

10111 (binary)
= 0001 0111 (binary)
= 0x17

0x3F9
= 11 1111 1001 (binary)

00	0	0000
01	1	0001
02	2	0010
03	3	0011
04	4	0100
05	5	0101
06	6	0110
07	7	0111
08	8	1000
09	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Binary to hex: 1000111100

00

2 3 C (12)

A. 8F0

B. 23C

Adding leading zeros is okay because it doesn't change the value of the number

C. None of the above

BIG IDEA: Bits can represent anything!!

Positional encoding scheme

Numbers	Binary Code
0	00
1	01
2	10
3	11

01 (1)
+10 (2)
11 (3)
Makes sense!

01 (1)
+ 01 (1)
10 (2)

Digit by digit addition works out. Good for making efficient circuits

Arbitrary scheme

Numbers	Code
0	11
1	10
2	00
3	01

10 (1)
+ 10 (1)
00 (2)

10 (1)
+ 00 (2)
10 (1)
Hz =1?

Lucky coincidence

How many (minimum) bits are required to represent the numbers 0 to 3?

BIG IDEA: Bits can represent anything!!

Arbitrary mapping is okay here

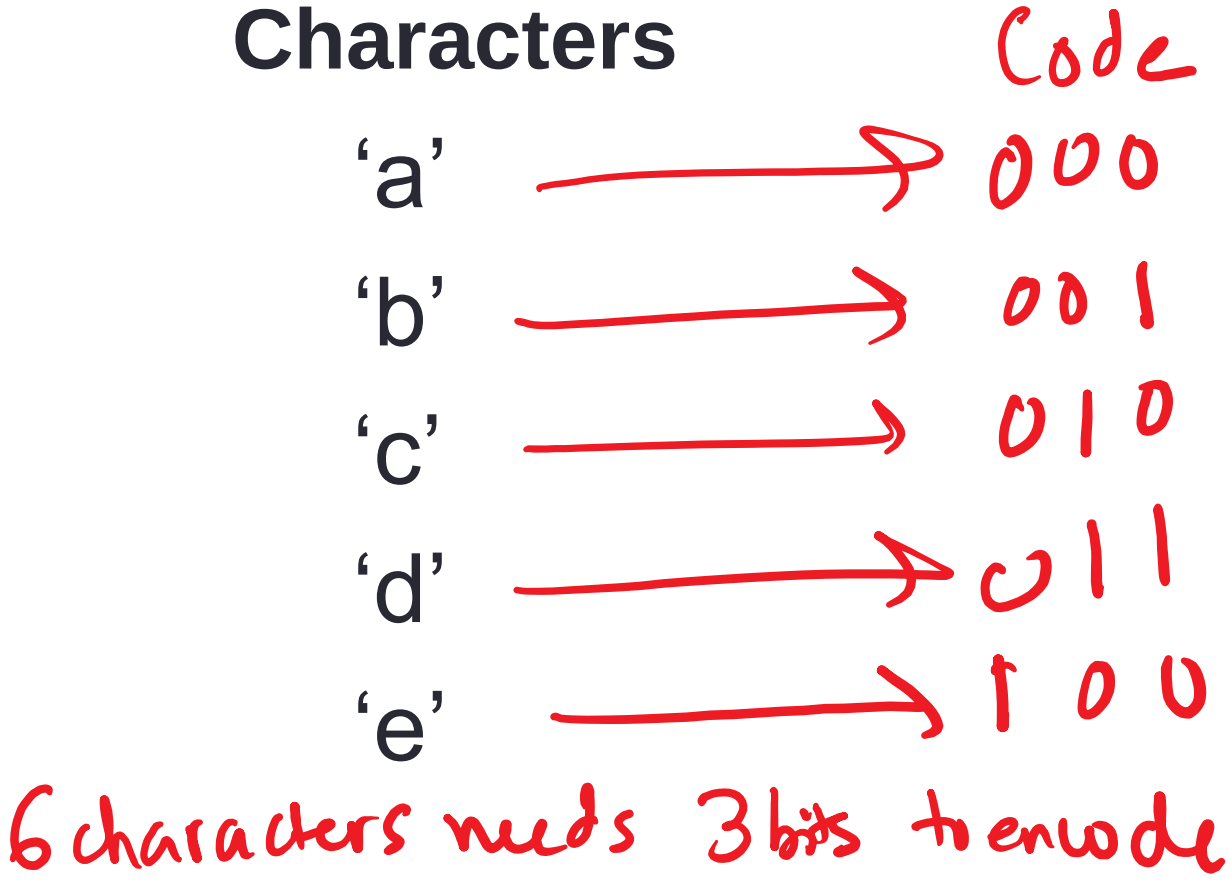
Colors	Binary code
Red	00
Green	01
Blue	10

11
01
00

How many (minimum) bits are required to represent the three colors? 2 bits

(can't have fractional bits)

BIG IDEA: Bits can represent anything!!



N bits can represent at most 2^N things

What is the minimum number of bits required to represent all the letters in the English alphabet?

A. 3
B. 4
C. 5
D. 6
E. 26

26 letters in the alphabet

$2^5 = 32$ (5 bits gives us enough unique bit patterns)

$2^4 = 16$ (4 bits is not enough)

BIG IDEA: Bits can represent anything!!

- Logical values?
  - 0 ⇒ False, 1 ⇒ True
- colors ?
- Characters?
  - 26 letters ⇒ 5 bits ($2^5 = 32$)
  - upper/lower case + punctuation
    ⇒ 7 bits (in 8) (“ASCII”)
  - standard code to cover all the world’s languages ⇒ 8,16,32 bits (“Unicode”)
    www.unicode.com
- locations / addresses? commands?

- MEMORIZE: N bits ⇔ at most 2^N things

What is the maximum positive value that can be stored in a byte?

A. 127

B. 128

C. 255

D. 256

1 byte → 8 bits

$2^8 = 256$

unsigned char x;

use 1 byte to store x

Intend to store only non-negative numbers

In that case, the range of values that can be stored are 0-255

Generalize to N bits

BIG IDEA: Bits can represent anything!!

[Positive & Negative nos]

Signed numbers	Binary Code
-3	101
-2	110
-1	111
0	000
1	001
2	010

-4 2 1

(This representation is called 2's complement)

1 110 (-2)

\+ 010 (2)

= 000 (0)

How many (minimum) bits are required to represent the numbers -3 to 2? (3)

Two's Compliment

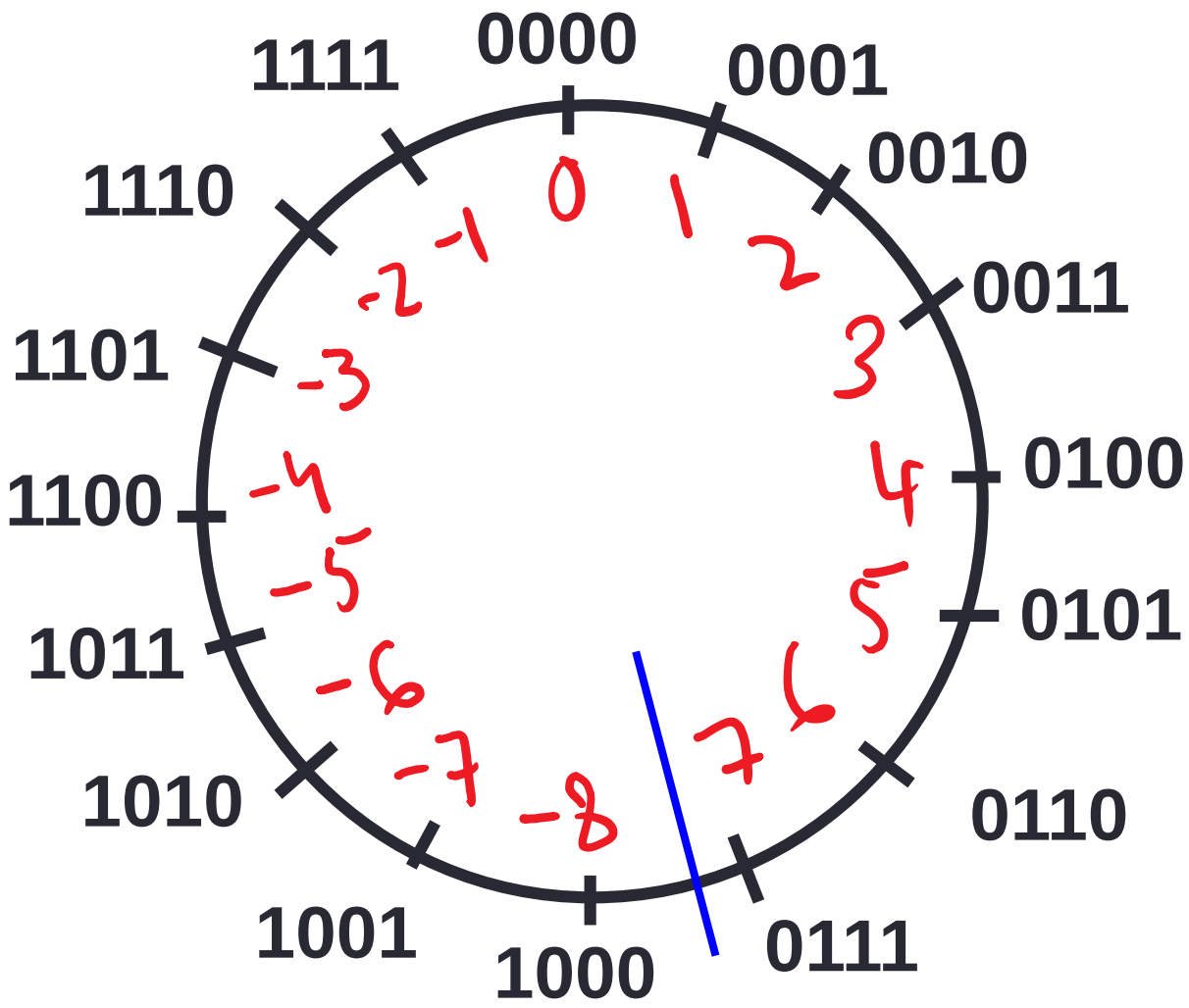
- Most significant bit represents a large negative weight:

$$\underset{-2^3}{1}\ \underset{2^2}{0}\ \underset{2^1}{0}\ \underset{2^0}{1} = -8+1 = -7$$

- To find the 2's complement representation
  - Write unsigned representation of the number saving one bit for sign
  - Flip all the bits
  - Add 1

Two's Complement

- Flip all the bits of unsigned representation and add 1



2 - 3 = ?

$$\begin{array}{rr} & 0010 \ (2) \\ + & 1101 \ (-3) \\ \hline & 1111 \ (-1) \end{array}$$

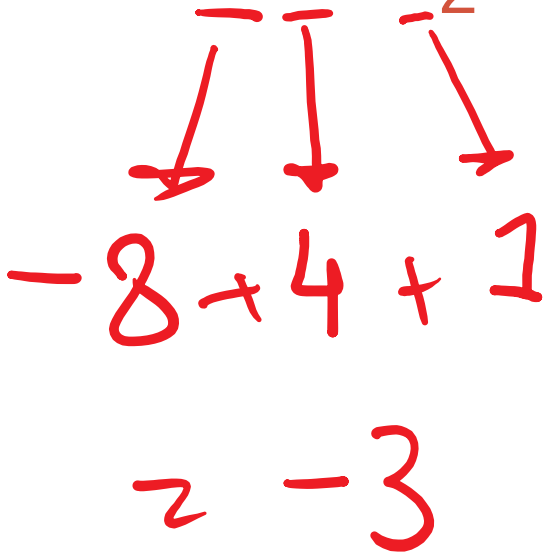
Two's Complement: $1101_2 = ?_{10}$

A. -2

B. -3

C. -4

D. -5



Addition and Subtraction

- Positive and negative numbers are handled in the same way.
- The carry out from the most significant bit is ignored.
- To perform the subtraction A - B, compute A + (two's complement of B)

Data types

Binary numbers in memory are stored using a finite, fixed number of bits typically:

- 8 bits (byte)
- 16 bits (half word)
- 32 bits (word)
- 64 bits (double word or quad)

Data type of a variable determines the:

- exact representation of variable in memory
- number of bits used (fixed and finite)
  - range of values that can be correctly represented

Next time

- Under the hood of program compilation
- Separate compilation with makefiles