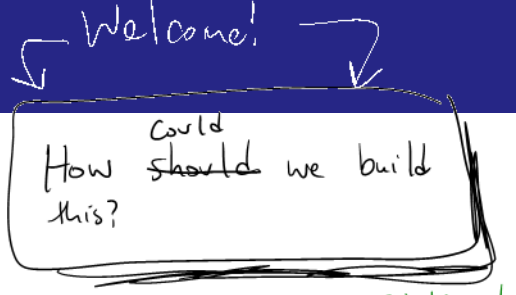
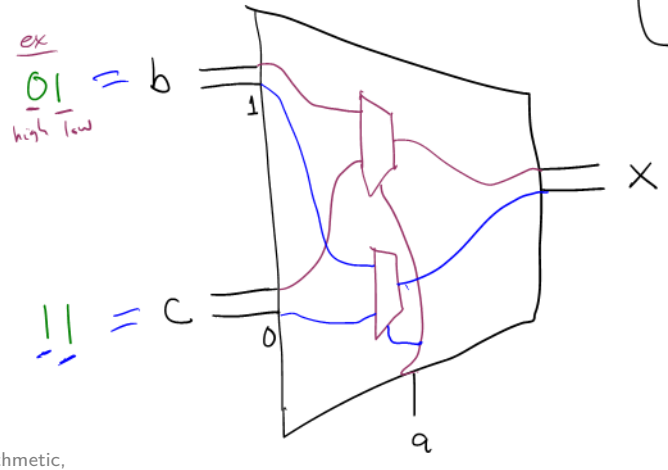
# 2-bit Multiplexer (mux)




2-bit values instead of 1-bit values




2 bits each

$x = a\ ?\ b : c$

| a | b | c | x |
|---|---|---|---|
| 0 | ? | [?] | 11 |

# Binary Arithmetic, Bitwise Operations

CS 2130: Computer Systems and Organization 1
September 3, 2025

# Announcements

- My Office Hours
  - Wednesdays 2:30-4pm
  - Fridays 11am-12pm
- TA Office Hours starting today
- Homework 1 available Friday, due September 15, 2025

# Multi-bit Values

- So far, only talking about 2 things
- Numbers, strings, objects, ...

# Numbers

From our oldest cultures, how do we mark numbers?

- **unary** representation: make marks, one per "thing"
  - Awkward for large numbers, ex: CS 2130?
  - Hard to tell how many marks there are
- Update: group them!
- Romans used new symbols: V, X, L, C, M

# Numbers

From our oldest cultures, how do we mark numbers?

- Arabic numerals
  - Positional numbering system
  - The 10 is significant:
    - 10 symbols, using 10 as base of exponent
  - The 10 is *arbitrary*
  - We can use other bases! $\pi, 2130, 2,$ ...

# Bases

We will discuss a few in this class

- Base-10 (decimal) - talking to humans
- Base-8 (octal) - shows up occasionally
- Base-2 (binary) - most important! (we've been discussing 2 things!)
- Base-16 (hexadecimal) - nice grouping of bits

# Binary

2 digits: 0, 1

7 bits

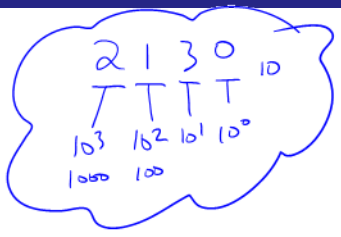
Try to turn $1100101_2$ into base-10:

$2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

$64\ 32\ 16\ 8\ 4\ 2\ 1$

101

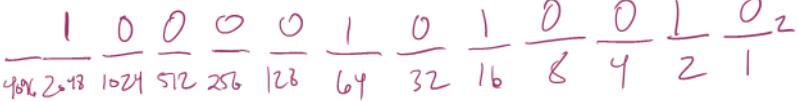
$64 + 32 + 4 + 1 = 101_{10}$

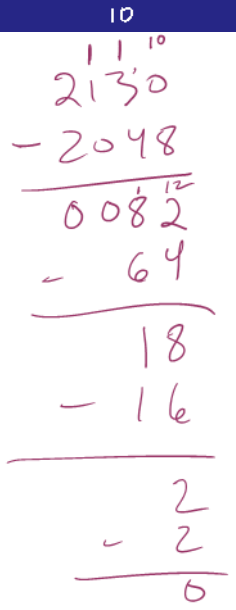


# Binary

Any downsides to binary?

Turn $2130_{10}$ into base-2:
*hint: find largest power of 2 and subtract*



# Long Numbers

How do we deal with numbers too long to read?

# Long Numbers

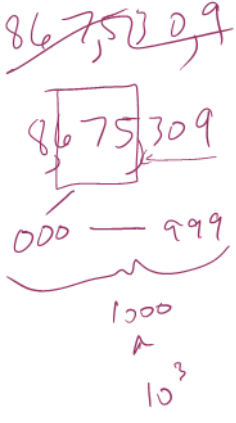
How do we deal with numbers too long to read?

- Group them by 3 (right to left)

# Long Numbers

How do we deal with numbers too long to read?

- Group them by 3 (right to left)
- In decimal, use commas: ,
- Numbers between commas: 000 - 999



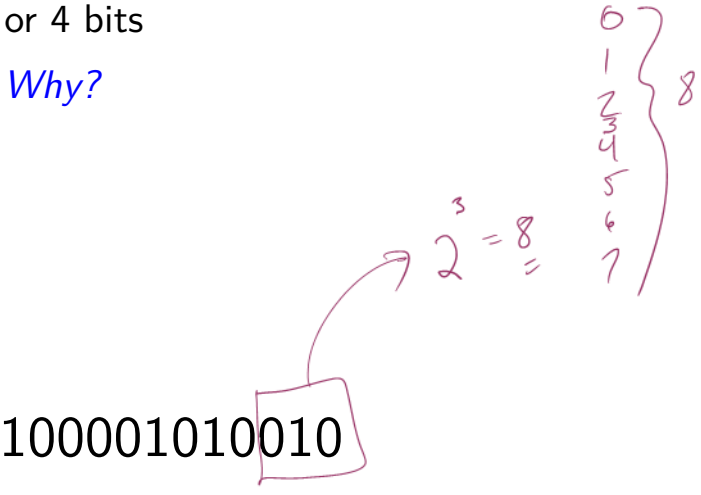
# Long Numbers

How do we deal with numbers too long to read?

- Group them by 3 (right to left)
- In decimal, use commas: ,
- Numbers between commas: 000 - 999
- Effectively base-1000

# Long Numbers in Binary - Readability

- Typical to group by 3 or 4 bits
- No need for commas *Why?*



# Long Numbers in Binary - Readability

- Typical to group by 3 or 4 bits
- No need for commas *Why?*
- We can use a separate symbol per group
- How many do we need for groups of 3?

100001010010

# Long Numbers in Binary - Readability

- Typical to group by 3 or 4 bits
- No need for commas *Why?*
- We can use a separate symbol per group
- How many do we need for groups of 3?
- Turn each group into decimal representation


100001010010

# Long Numbers in Binary - Readability

- Typical to group by 3 or 4 bits
- No need for commas *Why?*
- We can use a separate symbol per group
- How many do we need for groups of 3?
- Turn each group into decimal representation
- Converts binary to **octal**

$4122_8$

100001010010

# Long Numbers in Binary - Readability

- Groups of 4 more common
- How many symbols do we need for groups of 4?

$2^4 = 16$

100001010010

# Long Numbers in Binary - Readability

- Groups of 4 more common
- How many symbols do we need for groups of 4?
- Converts binary to **hexadecimal**
- Base-16 is very common in computing


100001010010

8 5 2 $_{16}$

# Hexadecimal

Need more than 10 digits. What next?

1110

e E

0
1
2
3
4
5
6
7
8
9
a = 10
b = 11
c = 12
d = 13
e = 14
f = 15

# Hexadecimal Exercise

Consider the following hexadecimal number:

852dab1e

Is it even or odd?

# Finally, Numbers!

Storing Integers

- Use binary representation of decimal numbers
- Usually have a limited number of bits (ex: 32, 64)
  - Depending on language
  - Depending on hardware

# Finally, Numbers!

Storing Integers

- Use binary representation of decimal numbers
- Usually have a limited number of bits (ex: 32, 64)
  - Depending on language
  - Depending on hardware
- Is there something missing?

# Negative Integers

Representing negative integers

# Negative Integers

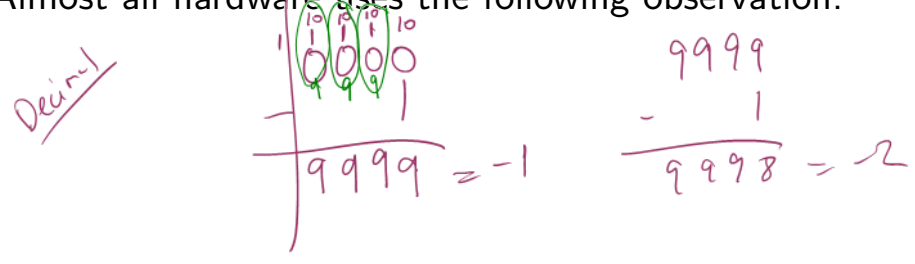
Representing negative integers

- Can we use the minus sign?

# Negative Integers

Representing negative integers

- Can we use the minus sign?
- In binary we only have 2 symbols, must do something else!
- Almost all hardware uses the following observation:



# Representing Negative Integers

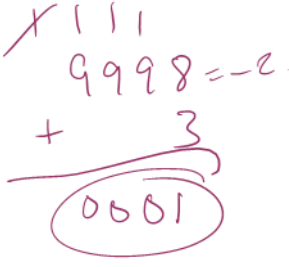
Computers store numbers in fixed number of wires

- Ex: consider 4-digit decimal numbers

# Representing Negative Integers

Computers store numbers in fixed number of wires

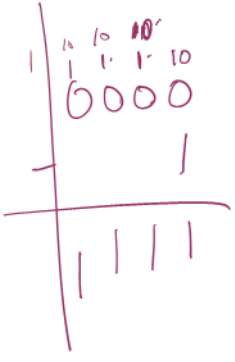
- Ex: consider 4-digit decimal numbers
- Throw away the last borrow:
  - 0000 - 0001 = 9999 == -1
  - 9999 - 0001 = 9998 == -2
  - Normal subtraction/addition still works
  - Ex: -2 + 3



# Representing Negative Integers

Computers store numbers in fixed number of wires

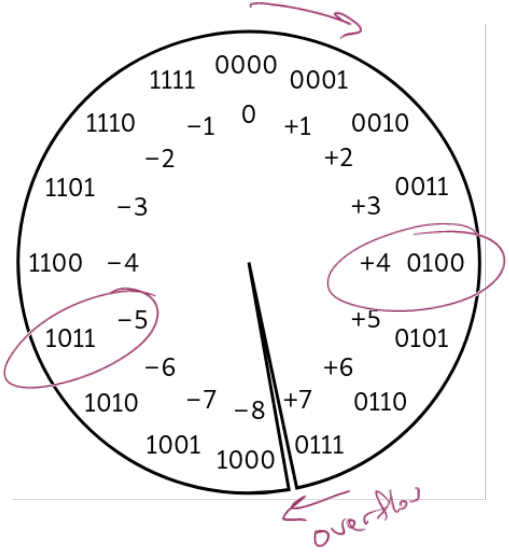
- Ex: consider 4-digit decimal numbers
- Throw away the last borrow:
  - 0000 - 0001 = 9999 == -1
  - 9999 - 0001 = 9998 == -2
  - Normal subtraction/addition still works
  - Ex: -2 + 3
- This works the same in binary



# Two's Complement

This scheme is called **Two's Complement**

- More generically, a *signed* integer
- There is a break as far away from 0 as possible
- First bit acts vaguely like a minus sign
- Works as long as we do not pass number too large to represent



# Two's Complement

# Values of Two's Complement Numbers

Consider the following 8-bit binary number in Two's Complement:

$$11010011$$

What is its value in decimal?