Data III & Integers I

CSE 351 Winter 2024

Instructor:
Justin Hsia

Teaching Assistants:
Adithi Raghavan
Aman Mohammed
Connie Chen
Eyoel Gebre
Jiawei Huang
Malak Zaki
Naama Amiel
Nathan Khuat
Nikolas McNamee
Pedro Amarante
Will Robertson

http://xkcd.com/257/

Relevant Course Information

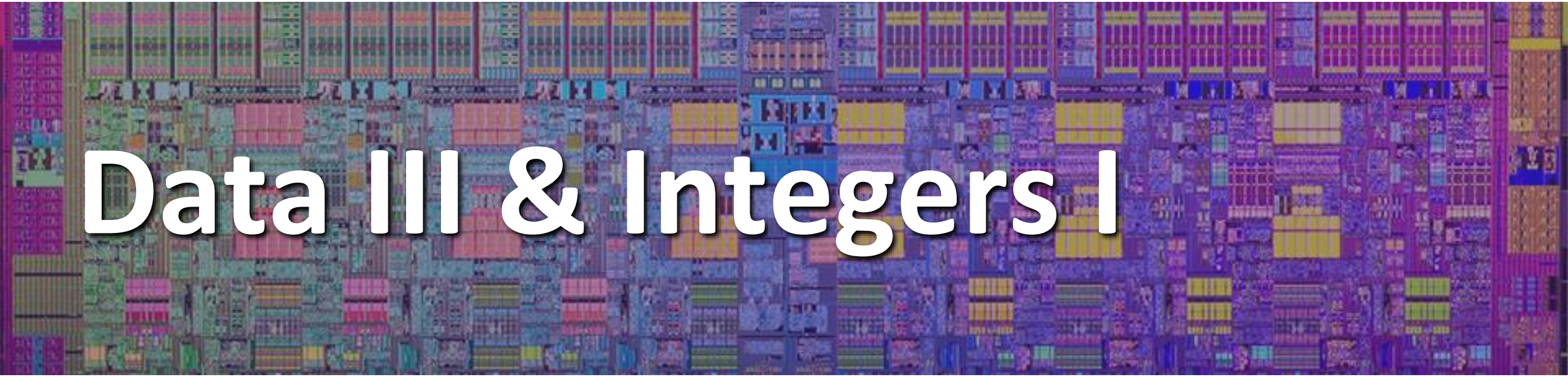
- HW2 due tonight, HW3 due Friday, HW4 due next Wednesday

- Lab 1a released
  - Some later functions require *bit shifting*, covered in Lesson 5
  - Workflow:
    1) Edit `pointer.c`
    2) Run the Makefile (`make clean` followed by `make`) and check for compiler errors & warnings
    3) Run ptest (`./ptest`) and check for correct behavior
    4) Run rule/syntax checker (`python3 dlc.py`) and check output
  - Due Monday 1/15, will overlap a bit with Lab 1b
    - We grade just your *last* submission
    - Don't wait until the last minute to submit – need to check autograder output

Lab Synthesis Questions

- All subsequent labs (after Lab 0) have a "synthesis question" portion
  - Can be found on the lab specs and are intended to be done *after* you finish the lab
  - You will type up your responses in a `.txt` file for submission on Gradescope
  - These will be graded "by hand" (read by TAs)

- Intended to check your understand of what you should have learned from the lab
  - Also great practice for short answer questions on the exams



Data III & Integers I

Lesson Summary (1/2)

- Bit-level operators allow for fine-grained manipulation
  - Bitwise AND (&), OR (|), XOR (^) and NOT (~) operate on the individual bits of the data
  - Especially useful with bitmasks, chosen bit vectors used with &, |, or ^
    - `b & 0 = 0, b & 1 = b` (set to zero or keep as-is)
    - `b | 0 = b, b | 1 = 1` (keep as-is or set to one)
    - `b ^ 0 = b, b & 1 = ~b` (keep as-is or flip the bit)

AND
Outputs 1 only when both input bits are 1:

&	0	1
0	**0**	**0**
1	**0**	**1**

OR
Outputs 1 when either input bit is 1:

|	0	1
0	**0**	**1**
1	**1**	**1**

XOR
Outputs 1 when either input is *exclusively* 1:

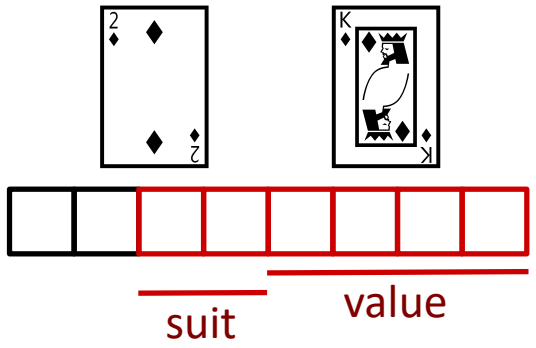
^	0	1
0	**0**	**1**
1	**1**	**0**

NOT
Outputs the opposite of its input:

~	
0	**1**
1	**0**

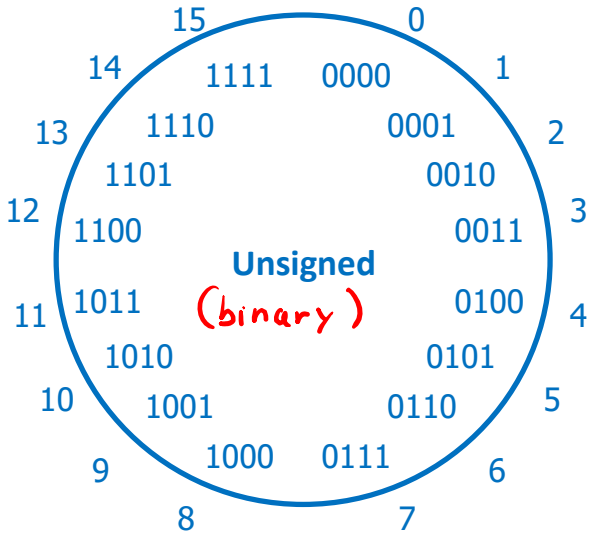
- Logical operators work on "truthiness" of data
  - 0 = False, anything else = True
  - Logical AND (&&), OR (||), and NOT (!) → always evaluate to 1 for True

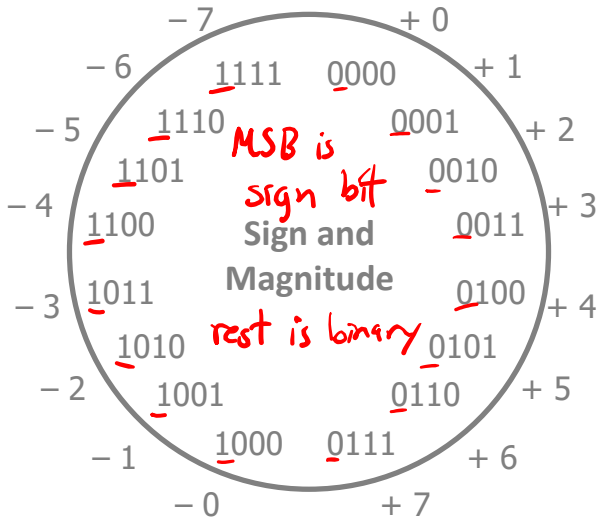
Lesson Summary (2/2)

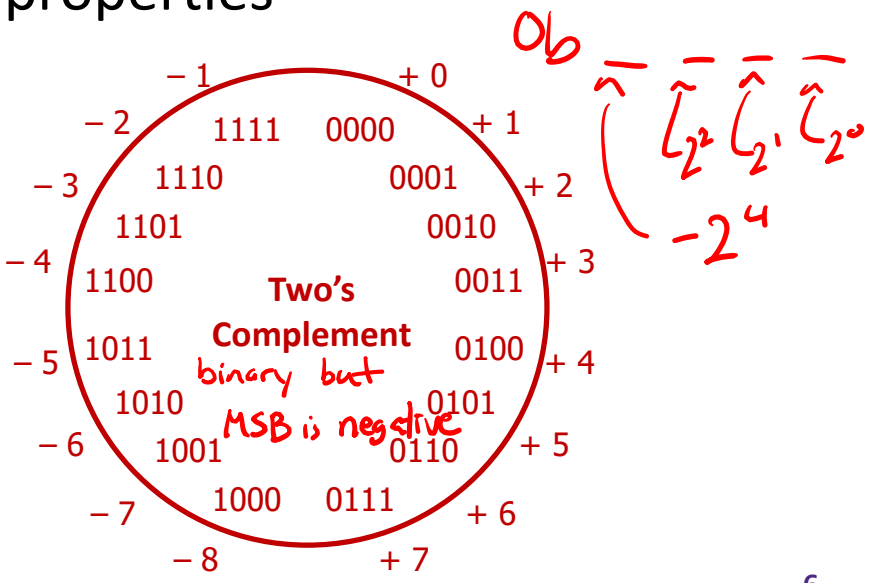


- Choice of *encoding scheme* is important
  - Tradeoffs based on size requirements and desired operations

- Integers represented using unsigned and two's complement representations (sign and magnitude not used in practice)
  - Limited by fixed bit width, satisfy desirable arithmetic properties



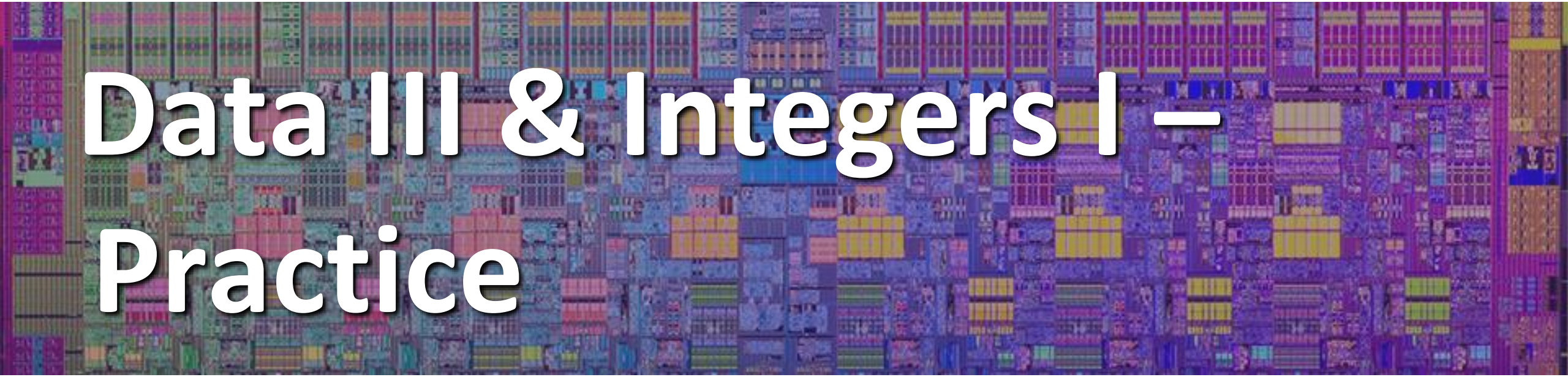




Lesson Q&A

- Learning Objectives:
  - Compute the effects of bit shifting, bitwise, logical, and arithmetic operations on integers.
  - Analyze the benefits and drawbacks of different integer representations (Unsigned, Sign and Magnitude, Two's Complement) and custom encoding schemes.

- What lingering questions do you have from the lesson?
  - Chat with your neighbors about the lesson for a few minutes to come up with questions



Data III & Integers I – Practice

Practice Questions (1/2)

- Compute the result of the following expressions for
  `char c = 0x81;` // 0b 1000 0001
  - `c ^ c` = 0x00 (1000 0001 ^ 1000 0001 = 0000 0000)
  - `~c & 0xA9` = 0x28 (0b0111 1110 & 0b1010 1001 = 0010 1000)
  - `c || 0x80` = 0x01 ("true" || "true" = "true")
  - `!(!c)` = 0x01 (!("false") = "true" = 0x1)

- Compute the value of `signed char sc = 0xF0;` (Two's Complement)

  = 0b 1111 0000
  = $-2^7+2^6+2^5+2^4$
  = -16

  $-sc = \sim sc + 1$ = 0b0000 1111 + 1 = 0b0001 0000 = +16

  sc = -16

Practice Questions (2/2)

- Take the 4-bit number encoding **`x = 0b1011`** (MSB)
- Which of the following numbers is NOT a valid interpretation of x using any of the number representation schemes discussed today?
  - Unsigned, Sign and Magnitude, Two's Complement

A. -4

B. -5

C. 11

D. -3

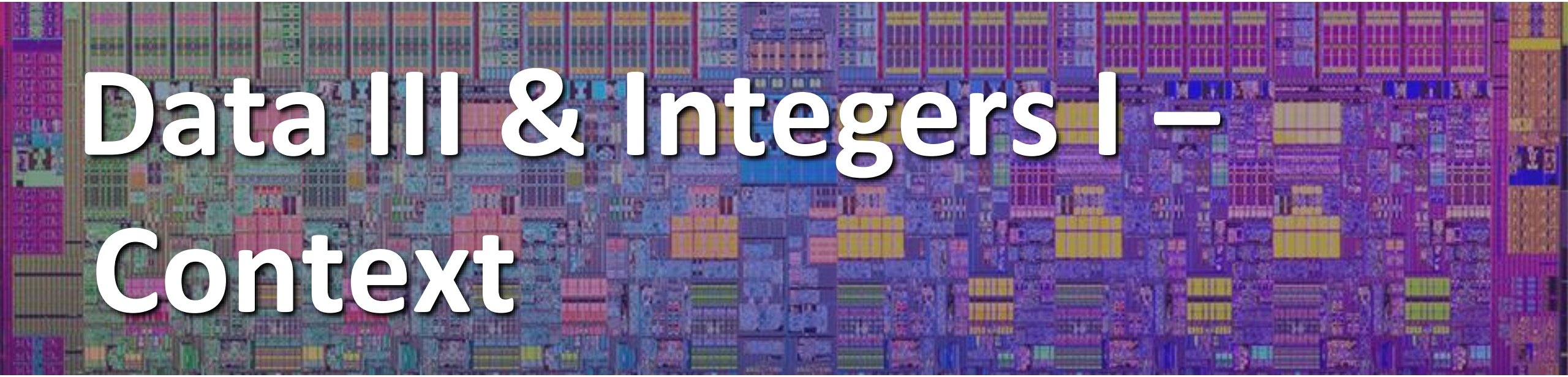
E. We're lost…

unsigned: $8+2+1=11$

sign + mag: $1011 \rightarrow -(2+1) = -3$

two's: $-8+2+1=-5$

$-x = 0b\ 0100+1 = 5 \rightarrow x = -5$



Data III & Integers I – Context

Integer Hardware

- In practice, all modern system use **unsigned** and **two's complement** encoding schemes for integers
  - Sign and magnitude for integers is a historical artifact, but useful context for design decision and for floating point (next unit)
  - Much of the same hardware can be used for both encoding schemes (*e.g.*, addition, subtraction)

- Fun fact: Java was designed to only support signed data types
  - *Assumed easier for beginners to understand* than having unsigned as well (*i.e.*, eliminate potential sources of error)
  - Unsigned operation support provided with Unsigned Integer API (starting with Java SE 8 in 2014)

Discussion Questions

- Discuss the following question(s) in groups of 3-4 students
  - I will call on a few groups afterwards so please be prepared to share out
  - Be respectful of others' opinions and experiences

- Thinking about the (implicit and explicit) design decisions for Two's Complement, what are some of the *advantages* and *disadvantages* of choosing to:
  - Represent consecutive (*i.e.*, no gaps) integers

    example, if only representing even integers, what should happen when we compute 6/2 ?
  - Represent the same number of positives and negatives

    the bias should make sense in the context of our application

    arithmetic might get weird again...
  - Positive number encodings match unsigned

    no need to convert anything when changing interpretations

Group Work Time

- During this time, you are encouraged to work on the following:
  1) If desired, continue your discussion
  2) Work on the homework problems
  3) Work on the lab (if applicable)

- Resources:
  - You can revisit the lesson material
  - Work together in groups and help each other out
  - Course staff will circle around to provide support