
WELCOME
TO
15-213
14-513
18-213
18-613
15-513

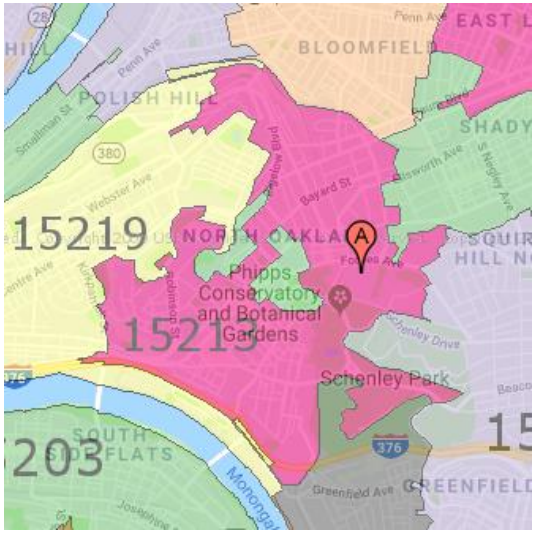
Course Overview

18-213/18-613: Computer Systems
An 15-213/18-213/15-513/14-513/18-613 Ecosystem Course

1st Lecture, May 14th, 2024

Instructors:

Gregory Kesden



18-213/18-613
In A Nutshell

Why Are We Here?

- "This course provides a **programmers view** of how computer systems execute programs, store information, and communicate."

- "It enables students to become **more effective programmers**, especially in dealing with issues of performance, portability and robustness."

- **"It also serves as a foundation** for courses on compilers, networks, operating systems, and computer architecture, where a deeper understanding of systems-level issues is required."

  - From the official course description, emphasis added.

Why Are We Here?

- Topics covered include:
  - machine-level code and its generation by optimizing compilers,
  - performance evaluation and optimization,
  - computer arithmetic,
  - memory organization and management,
  - networking technology and protocols, and
  - supporting concurrent computation.

- Also from the official course description

Who Are You?
(And, Who Is Everyone Else?)

Who Are You?

- **18-213**
  - ECE Undergrads

- **18-613**
  - MS in Electrical and Computer Engineering (Pittsburgh and Silicon Valley)
  - MS in Software Engineering (Silicon Valley)
  - MS in Artificial Intelligence Engineering

- **Interlopers** ☺
  - Others who happened to want the flexibility of a remote class this summer, since the other related offerings are Pittsburgh-centric.
    - You are most welcome and invited!

Where Is Everyone Else? Sibling Offering

- **15-213**
  - Undergrads in CS (Majors, 2nd majors, minors)
- **14-513/15-513**
  - Other Masters Students (Outside of the College of Engineering)

Who Am I?

I'm Here To Help!

Who Am I? Gregory Kesden

andrew.cmu.edu/user/gkesden/

GREGORY KESDEN

Electrical and Computer Engineering (ECE)
- Associate Teaching Professor
- MS ECE Faculty Advisor

- Office: HH A205
- Voice: 412-268-8647 (Forwards to cell, 24x7)
- Fax: 877-225-2329 (Who uses FAXes these days?)
- Email: gkesden@andrew.cmu.edu
- Hangouts, Messenger, Skype, Etc: gkesden

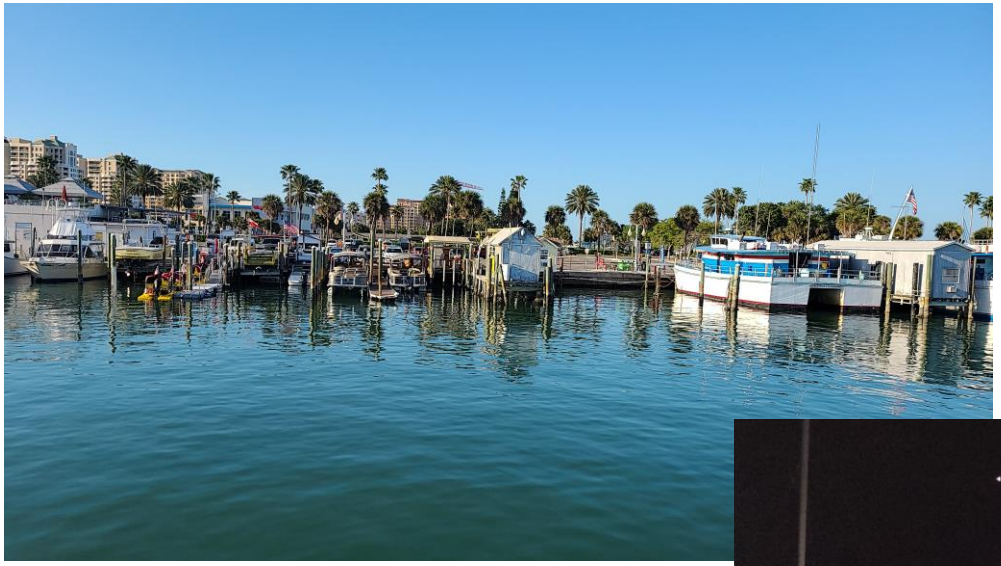
- Virtual Office Hours: I'm not online right now. See schedule below or send email. Thanks!
- Spring 2023 Schedule: Office Hours, Class Times, Recurring Meetings, Etc.

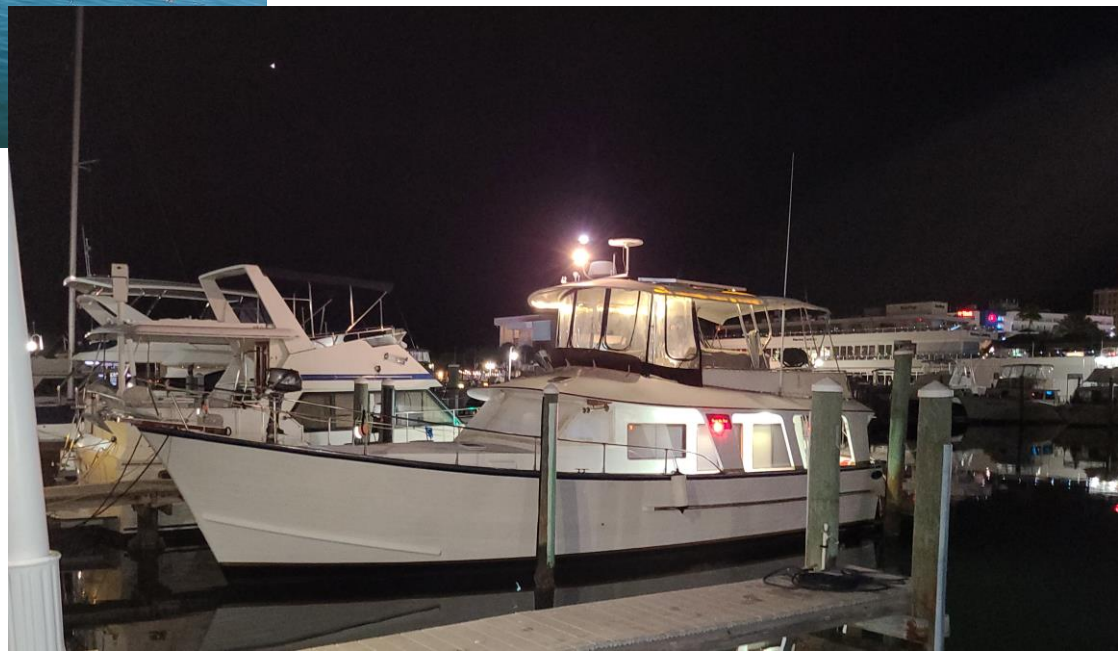
SUMMER-1 2023 CLASSES
- 18-213/613: Introduction to Computer Systems (Summer 2023)
- 18-537: C and UNIX Primer (Summer 2023)

RECENT CLASSES
- 18-100: Introduction to ECE (Fall 2020, Canvas) (Spring 2021, Canvas) (Spring 2022, Canvas) (Fall 2022, Canvas)(Summer 2023, Canvas)
- 18-349/14-642: Introduction to Embedded Systems: (Fall 2019) (Fall 2020) (Fall 2021)
- 14-848: Cloud Infrastructure: (Fall 2017) / (Fall 2018) / (Fall 2019)
- 14-760: Adv. Real-World Data Networks (Spring 2019) / (Spring 2020)
- 14-740: Networks (Spring 2018)
- 14-736: Distributed Systems (Spring 2018) (Spring 2019) / (Spring 2020)
- 14-712: Operating Systems (Spring 2020)
- 18-537-C6: An Introduction to C and UNIX (Summer-2/Mini-6 2022)
- 18-537-D6: An Introduction to Python and ML(Summer-2/Mini-6)
- 14-513/18-613/18-600: Computer Systems: (Fall 2017) / (Fall 2018) / (Fall 2019) / (Summer 2020) / (Spring 2021) / (Fall 2021) / (Spring 2022) / (Summer-All 2022) / (Fall 2022) / (Spring 2023)
- 15-418/618: Parallel Architecture and Programming (Spring 2018)

QUICK LINKS
- MS ECE Requirements AY2020
- Academic Calendar
- Career and Professional Development Center Calendar
- Schedule of Classes

Where Am I? Working Remote (Sometimes)





About Me

I'm Here To Help!

About The Course
Topical Coverage In Detail

Programs and Data

- **Topics**
  - Bit operations, arithmetic, assembly language programs
  - Representation of C control and data structures
  - Includes aspects of architecture and compilers

- **Assignments**
  - L0 (C programming Lab): Test/refresh your C programming abilities
  - L1 (datalab): Manipulating bits
  - L2 (bomblab): Defusing a binary bomb
  - L3 (attacklab): The basics of code injection attacks

The Memory Hierarchy

- **Topics**
  - Memory technology, memory hierarchy, caches, disks, locality
  - Includes aspects of architecture and OS

- **Assignments**
  - L4 (cachelab): Building a cache simulator and optimizing for locality.
    - Learn how to exploit locality in your programs.

Virtual Memory

- **Topics**
  - Virtual memory, address translation, dynamic storage allocation
  - Includes aspects of architecture and OS

- **Assignments**
  - L5 (malloclab): Writing your own malloc package
    - Get a real feel for systems-level programming

Exceptional Control Flow

- **Topics**
  - Hardware exceptions, processes, process control, Unix signals, nonlocal jumps
  - Includes aspects of compilers, OS, and architecture

- **Assignments**
  - L6 (tshlab): Writing your own Unix shell.
    - A first introduction to concurrency

Networking, and Concurrency

- **Topics**
  - High level and low-level I/O, network programming
  - Internet services, Web servers
  - concurrency, concurrent server design, threads
  - I/O multiplexing with select
  - Includes aspects of networking, OS, and architecture

- **Assignments**
  - L7 (proxylab): Writing your own Web proxy
    - Learn network programming and more about concurrency and synchronization.

What Makes This Course Different?
Course Perspective and Overarching Theme

Course Perspective

- **Most Systems Courses are Builder-Centric**
  - Computer Architecture
    - Design pipelined processor in Verilog
  - Operating Systems
    - Implement sample portions of operating system
  - Compilers
    - Write compiler for simple language
  - Networking
    - Implement and simulate network protocols

Course Perspective (Cont.)

- **Our Course is Programmer-Centric**
  - By knowing more about the underlying system, you can be more effective as a programmer
  - Enable you to
    - Write programs that are more reliable and efficient
    - Incorporate features that require hooks into OS
      - E.g., concurrency, signal handlers
  - Cover material in this course that you won't see elsewhere
  - Not just a course for dedicated hackers
    - **We bring out the hidden hacker in everyone!**

Course Theme: (Systems) Knowledge is Power!

- **Systems Knowledge**
  - How hardware (processors, memories, disk drives, network infrastructure) plus software (operating systems, compilers, libraries, network protocols) combine to support the execution of application programs
  - How you as a programmer can best use these resources
- **Useful outcomes**
  - Become more effective programmers
    - Able to find and eliminate bugs efficiently
    - Able to understand and tune for program performance
  - Prepare for later “systems” classes in CS, ECE, INI, ...
    - Compilers, Operating Systems, Networks, Computer Architecture, Embedded Systems, Storage Systems, Computer Security, etc.

It's Important to Understand How Things Work

- **Why do I need to know this stuff?**
  - Abstraction is good, but don't forget reality
- **Most CS courses emphasize abstraction**
  - **(CE courses less so)**
  - Abstract data types
  - Asymptotic analysis
- **These abstractions have limits**
  - Especially in the presence of bugs
  - Need to understand details of underlying implementations
  - Sometimes the abstract interfaces don't provide the level of control or performance you need

About The Course
How We Get There

Course Components

- **Lectures**
  - Higher level concepts
- **Labs (8)**
  - The heart of the course
  - 1-2+ weeks each
  - Provide in-depth understanding of an aspect of systems
  - Programming and measurement
- **Exams (midterm + final)**
  - Test your understanding of concepts & mathematical principles

Lab Rationale

- **Each lab has a well-defined goal such as solving a puzzle or winning a contest**
- **Doing the lab should result in new skills and concepts**
- **We try to use competition in a fun and healthy way**
  - Set a reasonable threshold for full credit
  - Post intermediate results (anonymized) on Autolab scoreboard for glory!

Course Resources

Textbooks

- **Randal E. Bryant and David R. O'Hallaron,**
  - *Computer Systems: A Programmer's Perspective*, **Third Edition** (CS:APP3e), Pearson, 2016
  - http://csapp.cs.cmu.edu
  - This book really matters for the course!
    - How to solve labs
    - Practice problems typical of exam problems
  - Electronic editions available (*Don't get paperback version*!)
  - On reserve in Sorrells Library
- **Brian Kernighan and Dennis Ritchie,**
  - *The C Programming Language*, Second Edition, Prentice Hall, 1988
  - Still the best book about C, from the originators
  - Even though it does not cover more recent extensions of C
  - On reserve in Sorrells Library

Online Resources

- **Web site: https://www.cs.cmu.edu/~18213**
  - Contains the schedule, deadlines, course policies, and many other resources.

- **Autolab: https://ics.autolabproject.com/**
  - Used to distribute labs, let you test your solutions, compete against each other, and for final grading

Online Resources

- **Piazza: https://piazza.com/class/lhn33vtkftw2oj/**
  - Q&A among your peers and privately with course staff
  - Can be anonymous with peers.

- **Canvas: https://www.canvas.cmu.edu**
  - Video links are automatically uploaded here after processing
    - Look for “Panopto”
  - Access is automatically managed by the University, updating each night to sync with the official course roster.

Getting Help

- **Email, Call, or Zoom the Instructor**
  - See web site for coordinates. Cell phone is 412-818-7813
- **Piazza**
  - https://piazza.com/class/lhn33vtkftw2oj/post/9
- **Drop in office hours via Zoom**
  - TBA
- **Small Group Sessions**
  - We'll organize those this week
- **Appointments**
  - By request

Facilities

- **Labs will use the Intel Computer Systems Cluster**
  - The "shark machines"
  - `linux> ssh shark.ics.cs.cmu.edu`

  - 21 servers donated by Intel for 213/513/613
    - 10 student machines (for student logins)
    - 1 head node (for instructor logins)
    - 10 grading machines (for autograding)
  - Each server: Intel Core i7: 8 Nehalem cores, 32 GB DRAM, RHEL 6.1
  - Rack-mounted in Gates machine room
  - Login using your Andrew ID and password

Course Policies

Policies: Grading

Final grades based on a straight scale (90/80/70/60)

- **Labs (50%):**
  - weighted according to effort
- **Final Exam (25%):**
  - Comprehensive, proctored, at the end.
  - No note sheet, we'll provide a reference sheet.
- **Homework (20%)**
  - Weekly Problem sets, mostly former exam questions
- **Small Group/Participation (5% or coefficient)**
  - Small groups are mandatory.
  - If you can't attend you'll be dropped, less than 60% attendance it'll count as a coefficient for your average, otherwise 5%
  - Via Zoom (Camera on). Weekly for ~1hr.

Timeliness

- **Grace days**
  - **5 grace days for the semester**
  - **Limit of 0, 1, or 2 grace days per lab used automatically**
  - Covers scheduling crunch, out-of-town trips, illnesses, minor setbacks
- **Lateness penalties**
  - Once grace day(s) used up, get penalized **15% per day**
  - No handins later than **3 days after due date**
- **Catastrophic events**
  - Major illness, death in family, …
  - Formulate a plan (with your academic advisor) to get back on track
- **Advice**
  - Once you start running late, it's really hard to catch up
  - Try to save your grace days until the last few labs

A Personal Touch

- **You'll soon be organized into groups of 5 students and a TA facilitator**
  - Each TA facilitator only has 2 groups.
  - The goal is to give you personal attention and to help you enjoy getting to know and working with some of your wonderful colleagues.
- **If you need anything, your TA should be your first stop**
  - They focus on only 10 students. Their goal is to know you well.
  - They have time allocated each week for 1:1 meetings with students in their group – if you need help, don't let it go to waste, just ask
- **They are very empowered.**
  - If you need anything, e.g. schedule adjustments, just ask. They'll work with you in a personal way to adjust deadlines to your needs.
  - But they aren't allowed to "kick the can". They need to make adjustments that lead to getting caught up. They are asked to consider the whole schedule and ensure a good path to success.

Academic Integrity, and Violations Thereof

- **Doing one's own work and giving credit where credit is due for the ideas and work of others are very important in the academic culture at CMU and across the United States, among many other places**

- **Even what may seem to some to be small failures to do one's own work or credit others are taken *very* seriously here**
  - Course penalties ranging from -100% on the assignment to an R in the class
  - Program penalties ranging from probation to dismissal
  - Other consequences, such as loss of eligibility for scholarships, fellowships, awards, etc

Consider "Speeding" on the Highway

- **The posted sign says, "65 MPH"**
  - In some places this means, "Go 60 MPH to be safe, because if you cross 65 MPH, you'll likely be pulled over, fined, and your insurance will cost a lot more. And, be careless too often and you'll lose your license."
    - The speed limit is set to protect lives. It makes no sense to risk killing a person, widowing someone, orphaning someone, or disabling someone for life just to get somewhere a few minutes faster.
  - In some places this means, "Don't go less than 65 MPH or over 79 MPH"
    - Cars are pretty safe, speed limits are set very conservatively, I'm an especially good driver, and if I go slower, I'll slow down everyone behind me and they'll be annoyed, people will honk at me, and my friends won't let me drive."
- **Conditions – and culture – define one's view.**
  - It varies *drastically* from place to place.

Our Culture

- **Credit must be given for other people's intellectual contributions (.)**
  - Ideas
  - Work product
- **Using other people's intellectual property when it isn't allowed is like trespassing – but worse.**
  - It isn't allowed (.)
- **Using other people's intellectual property without proper citation isn't allowed.**
  - It is like stealing a rental care.

Know the Policies

- **Look at the syllabus, course web site, University Web site, and any departmental or program information:**
  - Read the policies
  - Ask questions
  - Follow the policies

- **Work that isn't yours can't influence your project work**
  - Things you google, or find on git, or your seniors tell you, or your friends tell you.

- **You can't give unauthorized assistance**
  - Even after you leave the class.

- **The policy is retroactive**
  - You can be punished in accordance with the policy – even after graduation.

How to Avoid AIVs

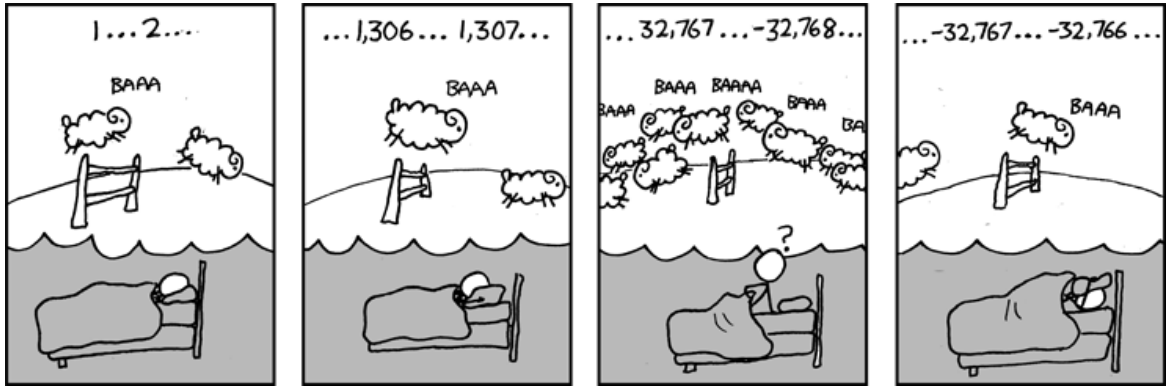
- **Start early**
- **Don't rely on marathon programming sessions**
  - Your brain works better in small bursts of activity
  - Ideas / solutions will come to mind while you're doing other things
- **Plan for stumbling blocks**
  - Assignment is harder than you expected
  - Code doesn't work
  - Bugs hard to track down
  - Life gets in the way
    - Minor health issues
    - Unanticipated events

Diving In:
Great Realities

Great Reality #1:

Ints are not Integers, Floats are not Reals

- **Example 1: Is $x^2 \geq 0$?**
  - Float's: Yes!
  - Int's:
    - 40000 * 40000 --> 1600000000
    - 50000 * 50000 --> ?



- **Example 2: Is (x + y) + z = x + (y + z)?**
  - Unsigned & Signed Int's: Yes!
  - Float's:
    - (1e20 + -1e20) + 3.14 --> 3.14
    - 1e20 + (-1e20 + 3.14) --> ??

Source: xkcd.com/571

Computer Arithmetic

- **Does not generate random values**
  - Arithmetic operations have important mathematical properties
- **Cannot assume all "usual" mathematical properties**
  - Due to finiteness of representations
  - Integer operations satisfy "ring" properties
    - Commutativity, associativity, distributivity
  - Floating point operations satisfy "ordering" properties
    - Monotonicity, values of signs
- **Observation**
  - Need to understand which abstractions apply in which contexts
  - Important issues for compiler writers and serious application programmers

Great Reality #2:
You've Got to Know Assembly

- **Chances are, you'll never write programs in assembly**
  - Compilers are much better & more patient than you are
- **But: Understanding assembly is key to machine-level execution model**
  - Behavior of programs in presence of bugs
    - High-level language models break down
  - Tuning program performance
    - Understand optimizations done / not done by the compiler
    - Understanding sources of program inefficiency
  - Implementing system software
    - Compiler has machine code as target
    - Operating systems must manage process state
  - Creating / fighting malware
    - x86 assembly is the language of choice!

Great Reality #3: Memory Matters

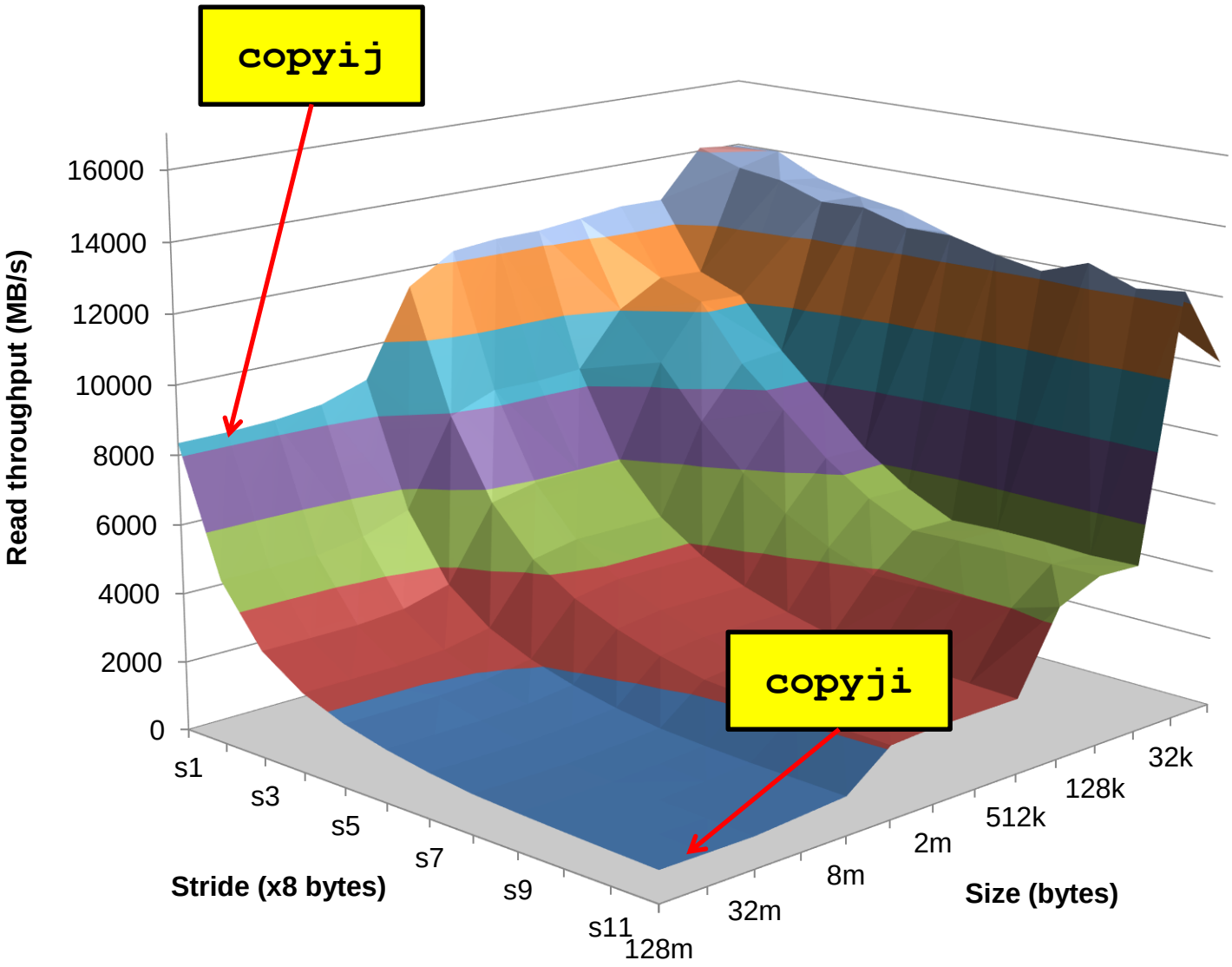
Random Access Memory Is an Unphysical Abstraction

- **Memory is not unbounded**
  - It must be allocated and managed
  - Many applications are memory dominated
- **Memory referencing bugs especially pernicious**
  - Effects are distant in both time and space
- **Memory performance is not uniform**
  - Cache and virtual memory effects can greatly affect program performance
  - Adapting program to characteristics of memory system can lead to major speed improvements

Great Reality #4: There's more to performance than asymptotic complexity

- **Constant factors matter too!**
- **And even exact op count does not predict performance**
  - Easily see 10:1 performance range depending on how code written
  - Must optimize at multiple levels: algorithm, data representations, procedures, and loops
- **Must understand system to optimize performance**
  - How programs compiled and executed
  - How to measure program performance and identify bottlenecks
  - How to improve performance without destroying code modularity and generality

Why The Performance Differs



Great Reality #5:
Computers do more than execute programs

- **They need to get data in and out**
  - I/O system critical to program reliability and performance

- **They communicate with each other over networks**
  - Many system-level issues arise in presence of network
    - Concurrent operations by autonomous processes
    - Coping with unreliable media
    - Cross platform compatibility
    - Complex performance issues

Coming Up

Next class: From Signals to Bits to Integer Arithmetic

Soon: Autolab will come online for testing

Soon: Group sign-up

Soon: Office Hours posting.

Always: How can we be of service? Please reach out!

Welcome
and Enjoy!