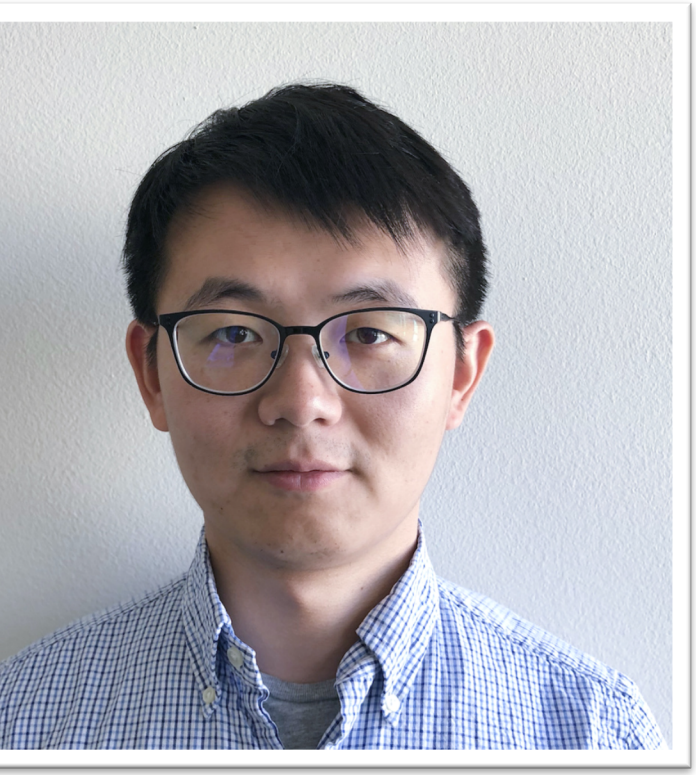
# Course Overview

CSC3501: Computer Organization & Design
1st Lecture, Spring 2021

**Instructors:**

Dr. Hao Wang (https://www.haow.ca/)

# Instructors: Hao Wang



Ph.D. University of Toronto

- Distributed computing
- Cloud computing
- Machine learning systems

A full-stack developer

A watercolor enthusiast

A photographer

A runner

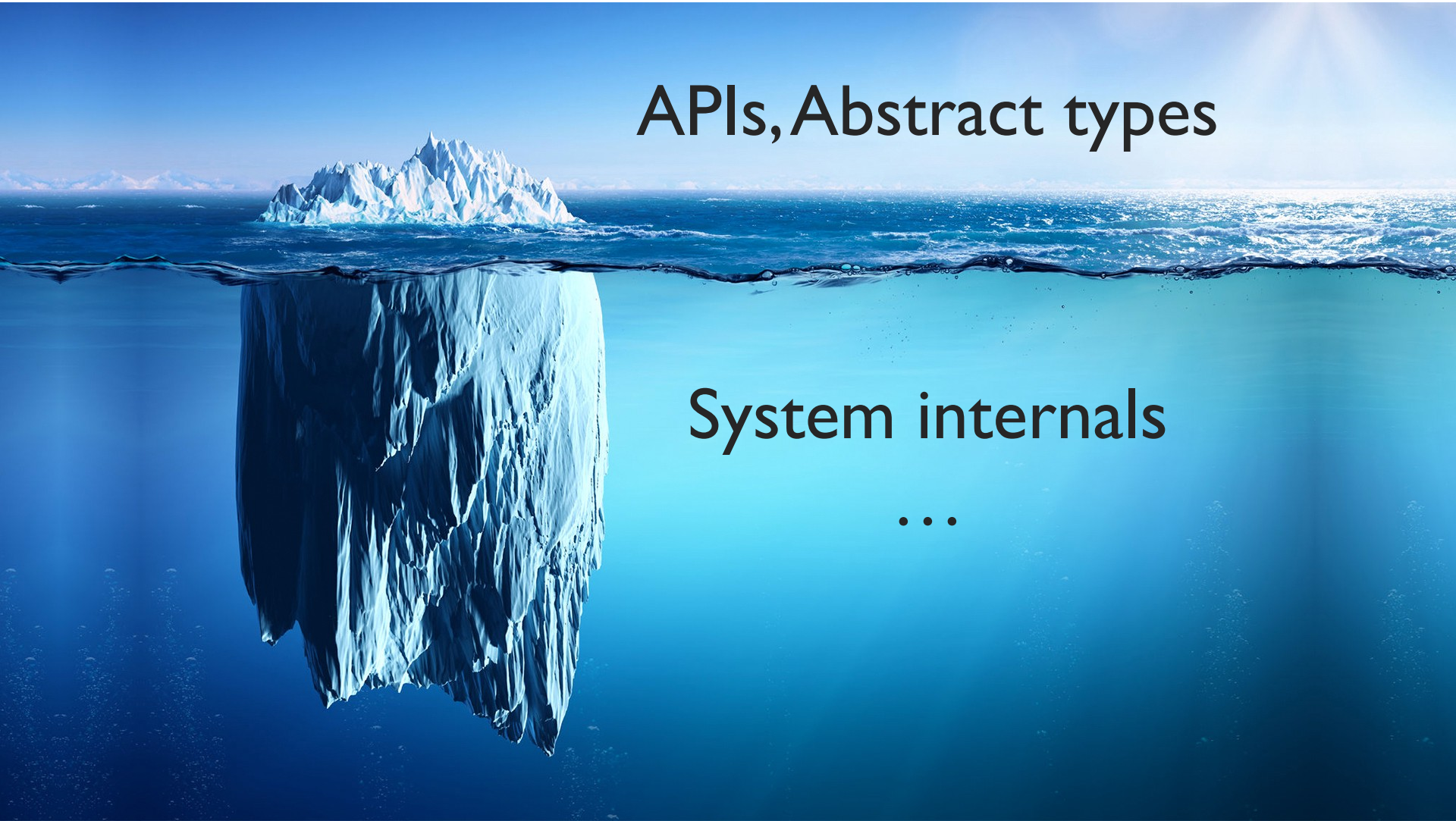
A video gamer

…

# Overview

- Course theme
- Five realities
- How the course fits into the CS/ECE curriculum
- Academic integrity

# Course Theme:
# Abstraction Is Good But Don't Forget Reality



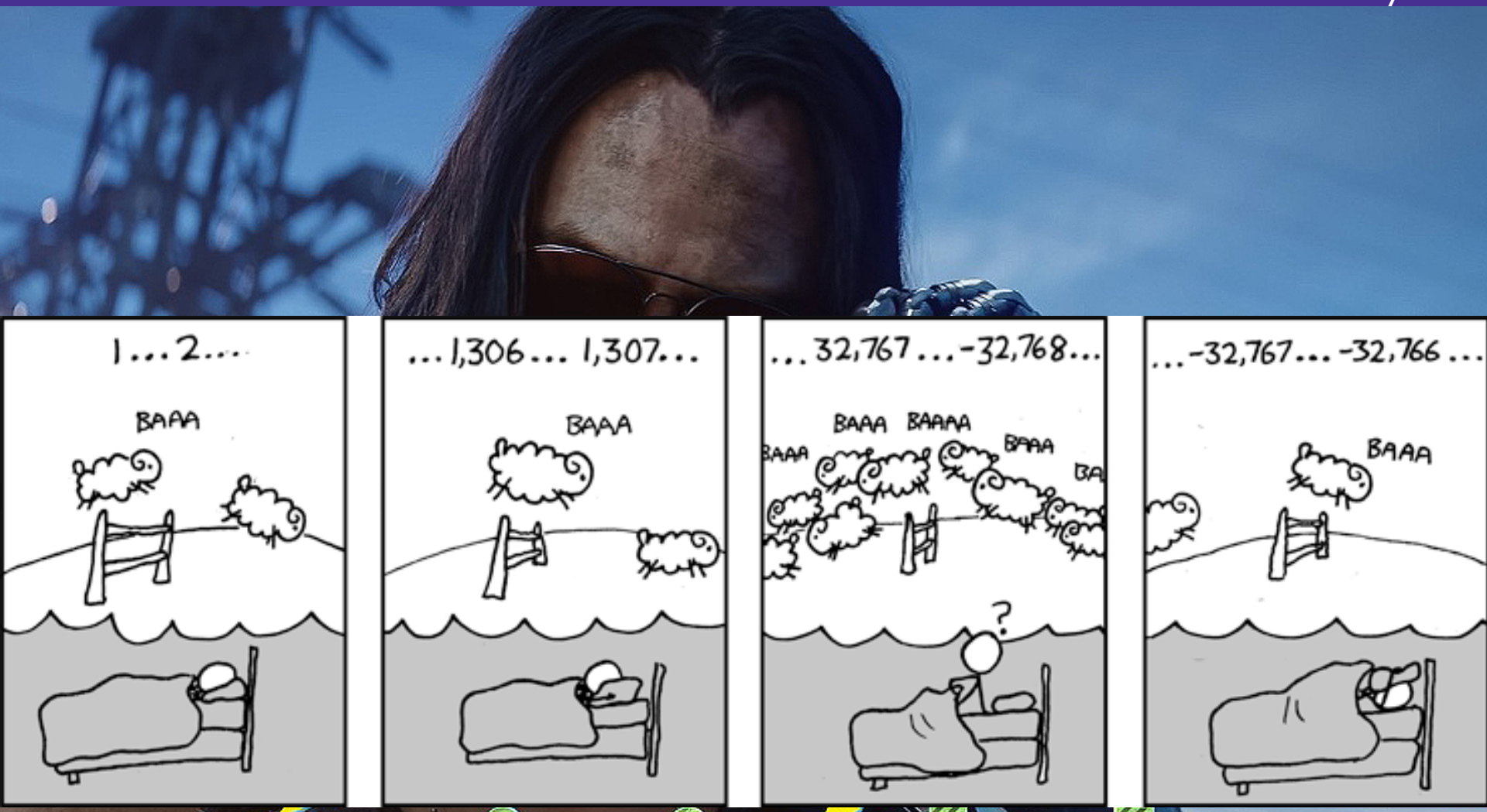
# Course Theme: Abstraction Is Good But Don't Forget Reality

- **Most CS and CE courses emphasize abstraction**
  - Abstract data types
  - Asymptotic analysis
- **These abstractions have limits**
  - Especially in the presence of bugs
  - Need to understand details of underlying implementations
- **Useful outcomes from taking CSC3501**
  - 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
    - Compilers, Operating Systems, Networks, Computer Architecture, Embedded Systems, Storage Systems, etc.

# 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


1...2...
BAAA
...1,306... 1,307...
BAAA
...32,767...-32,768...
BAAA BAAAA
BAAA
BAAA
BA
?
...-32,767... -32,766...
BAAA


2 0 7 7

# 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

# Memory Referencing Bug Example

```
typedef struct {
  int a[2];
  double d;
} struct_t;

double fun(int i) {
  volatile struct_t s;
  s.d = 3.14;
  s.a[i] = 1073741824; /* Possibly out of bounds */
  return s.d;
}
```

```
fun(0)  →  3.14
fun(1)  →  3.14
fun(2)  →  3.1399998664856
fun(3)  →  2.00000061035156
fun(4)  →  3.14
fun(6)  →  Segmentation fault
```

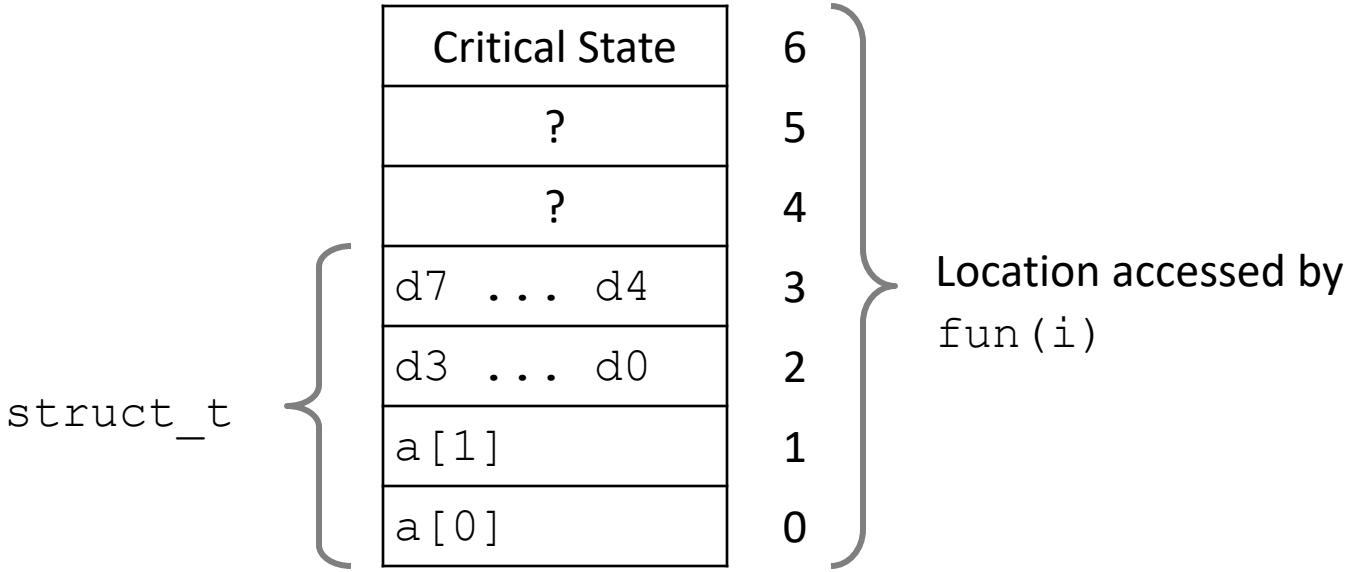
- Result is system specific

# Memory Referencing Bug Example

```
typedef struct {
  int a[2];
  double d;
} struct_t;
```

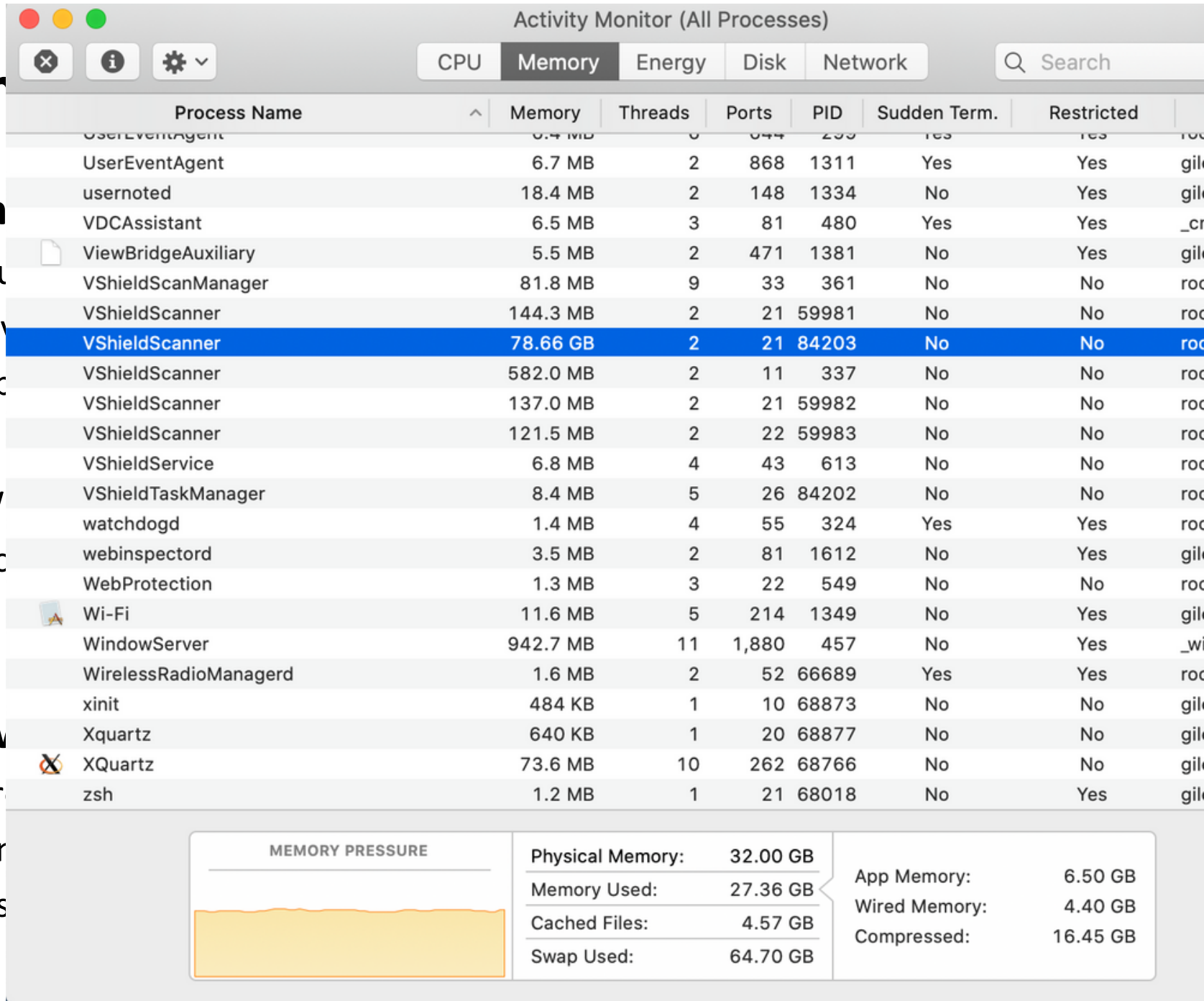
```
fun(0)  →  3.14
fun(1)  →  3.14
fun(2)  →  3.1399998664856
fun(3)  →  2.0000061035156
fun(4)  →  3.14
fun(6)  →  Segmentation fault
```

Explanation:



# Mer

- C an
  - Ou
  - Inv
  - Ab
- Can
  - W
  - Ac
    - 
    - 
- How
  - Pr
  - Ur
  - Us



# 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

# Memory System Performance Example

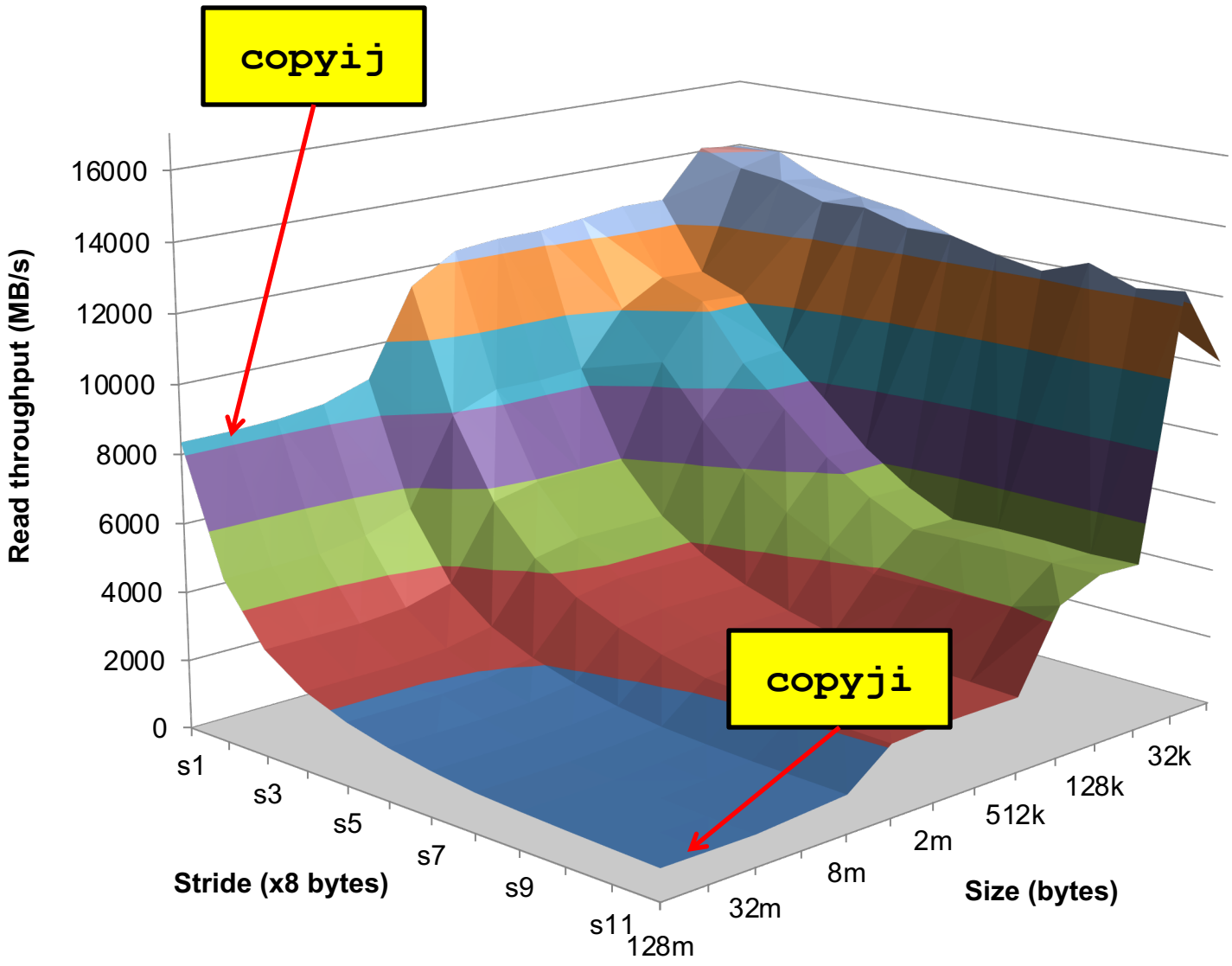
```
void copyij(int src[2048][2048],
            int dst[2048][2048])
{
  int i,j;
  for (i = 0; i < 2048; i++)
    for (j = 0; j < 2048; j++)
      dst[i][j] = src[i][j];
}
```

```
void copyji(int src[2048][2048],
            int dst[2048][2048])
{
  int i,j;
  for (j = 0; j < 2048; j++)
    for (i = 0; i < 2048; i++)
      dst[i][j] = src[i][j];
}
```

4.3ms 2.0 GHz Intel Core i7 Haswell 81.8ms

- Hierarchical memory organization
- Performance depends on access patterns
  - Including how step through multi-dimensional array

# Why The Performance Differs



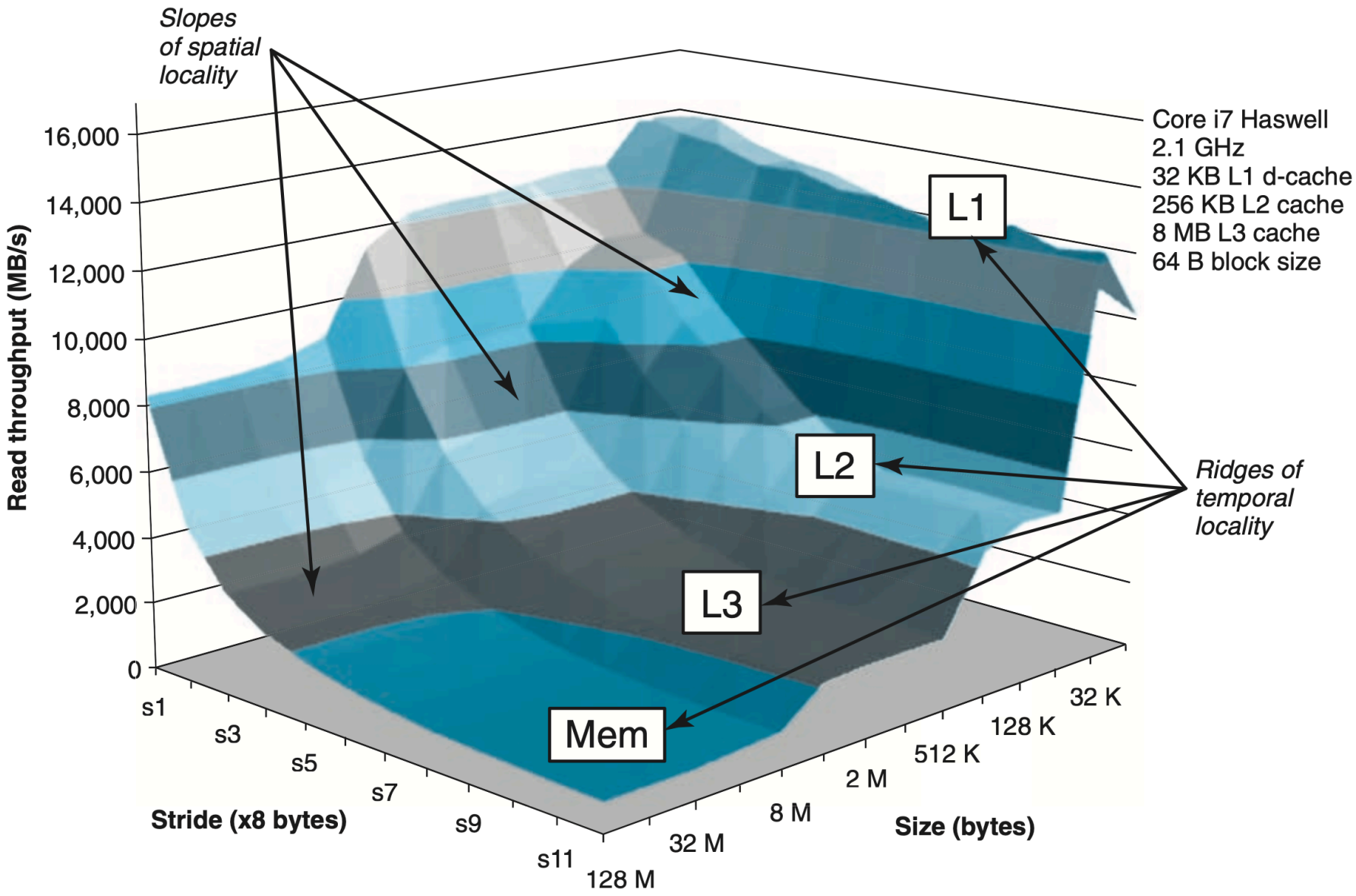



**Figure 6.41** **A memory mountain.** Shows read throughput as a function of temporal and spatial locality.

# 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

# 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
  - Purpose is to show that by knowing more about the underlying system, one 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!**

# Cheating: Description

- What is cheating?
  - Sharing code: by copying, retyping, **looking at**, or supplying a file
  - Describing: verbal description of code from one person to another.
  - Coaching: helping your friend to write a lab, line by line
  - Searching the Web for solutions
  - Copying code from a previous course or online solution
    - You are only allowed to use code we supply, or from the CS:APP website
- What is NOT cheating?
  - Explaining how to use systems or tools
  - Helping others with high-level design issues
- See the course syllabus for details.
  - Ignorance is not an excuse

# Cheating: Consequences

- Penalty for cheating:
  - Removal from course with failing grade (no exceptions!)
  - Permanent mark on your record
  - Your instructors' personal contempt
- Detection of cheating:
  - We have sophisticated tools for detecting code plagiarism
  - We also collect code from GitHub
- Don't do it! But…
  - Start early
  - Ask the staff for help when you get stuck

# Textbooks

- Randal E. Bryant and David R. O'Hallaron,
  - *Computer Systems: A Programmer's Perspective*, **Third Edition** (CS:APP3e), Pearson, 2016
  - https://www.vitalsource.com/referral?term=9780134092997
  - This book really matters for the course!
    - How to solve labs
    - Practice problems typical of exam problems

- Brian Kernighan and Dennis Ritchie,
  - *The C Programming Language*, Second Edition, Prentice Hall, 1988
  - Still the best book about C, from the originators

# Course Components

- Lectures
  - Higher level concepts
- Quizzes
- Labs (4)
  - 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

# Getting Help

- Moodle
  - Complete schedule of lectures, exams, and assignments
  - Copies of lectures, assignments, exams, solutions
  - Clarifications to assignments

# Getting Help

- Email to the instructor or TA using your official school email
  - No Last minute questions
- Office hours (starting Wed Jan 13):
  - Wednesday 1PM-3PM via Zoom meeting
- 1:1 Appointments
  - You can schedule 1:1 Zoom meeting with any of the instructor and TA

# Policies: Labs And Exams

- No work groups
  - You must work alone on all lab assignments
- Exams
  - Exams will be online in network-isolated clusters
  - Held over multiple days. Self-scheduled; just sign up!

# Policies: Grading

- Exams (75%): midterm (30%), final (45%)
- Quizzes (5%)
- Assignments (20%)
  - Assignment 1: 2%, Assignments 2-4: 6% each

# Timeliness

- Lateness penalties
  - Penalized **20% per day**
  - No hand-ins 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

# Other Rules of the Lecture Hall

- Laptops: permitted
- Electronic communications: forbidden
  - No email, instant messaging, cell phone calls, etc
  - Mute your devices
- Check the COVID-19 statement

# Data Representations and Operations

- Topics
  - Bits operations, arithmetic, data types
  - Includes aspects of architecture and compilers

- Assignments
  - A1 (datalab): Manipulating bits

# Assembly Programming

- Topics
  - Representation of C control and data structures
  - Includes aspects of architecture and compilers

- Assignments
  - A2 (attacklab): The basics of code injection attacks

# The Memory Hierarchy and Cache

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

- Assignments
  - A3 (cachelab): Building a cache simulator and optimizing for locality
    - Learn how to exploit locality in your programs

# Processor Architecture

- Topics
  - Processor instruction sets, logic design, instruction execution
  - Includes aspects of architecture, hardware design

- Assignments
  - A4 (archlab): Optimizing a Y86 array copying function and a working pipelined Y86 processor design
    - Understand the interactions between hardware and software

# 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

# Enjoy and Happy Hacking!