Overview of the ECE Computer Software Curriculum

David O'Hallaron

Associate Professor of ECE and CS

Carnegie Mellon University

The Fundamental Idea of Abstraction

Human beings
Applications
Software systems
Architecture
Logic
Circuits
Devices
Physics

Systems of all kinds control complexity using layers of abstractions.

Why Study Software?

Human beings	Software tools
Applications	
Software systems	
Architecture	
Logic	
Circuits	
Devices	
Physics	

1. Engineers working at all levels need to build and use software tools.

Why Study Software? (cont)

Human beings
Applications
Software systems
Architecture
Logic
Circuits
Devices
Physics

2. Mediocre engineers understand one level

Why Study Software? (cont)

Human beings
Applications
Software systems
Architecture
Logic
Circuits
Devices
Physics

2. Mediocre engineers understand one level

Good engineers understand a level above and below

Why Study Software? (cont)

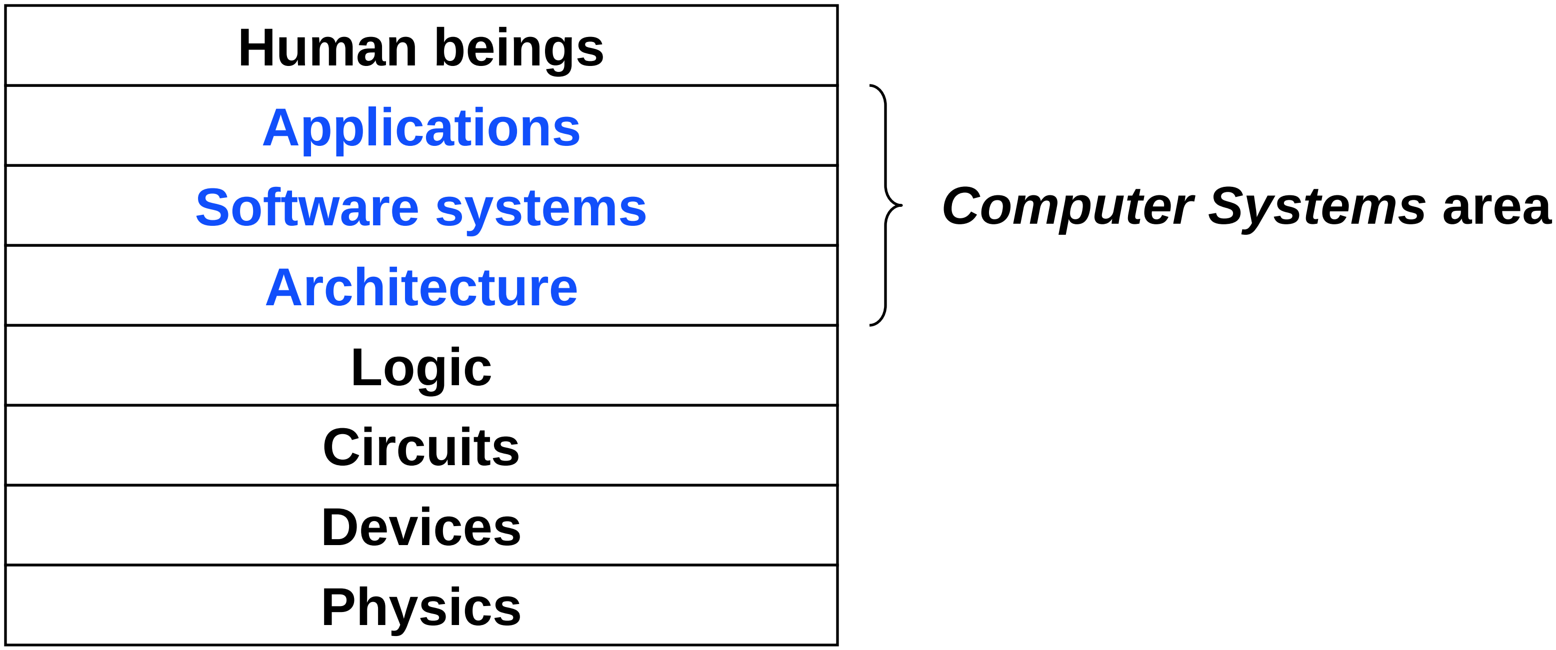
Human beings
Applications
Software systems
Architecture
Logic
Circuits
Devices
Physics

2. Mediocre engineers understand one level

Good engineers understand a level above and below

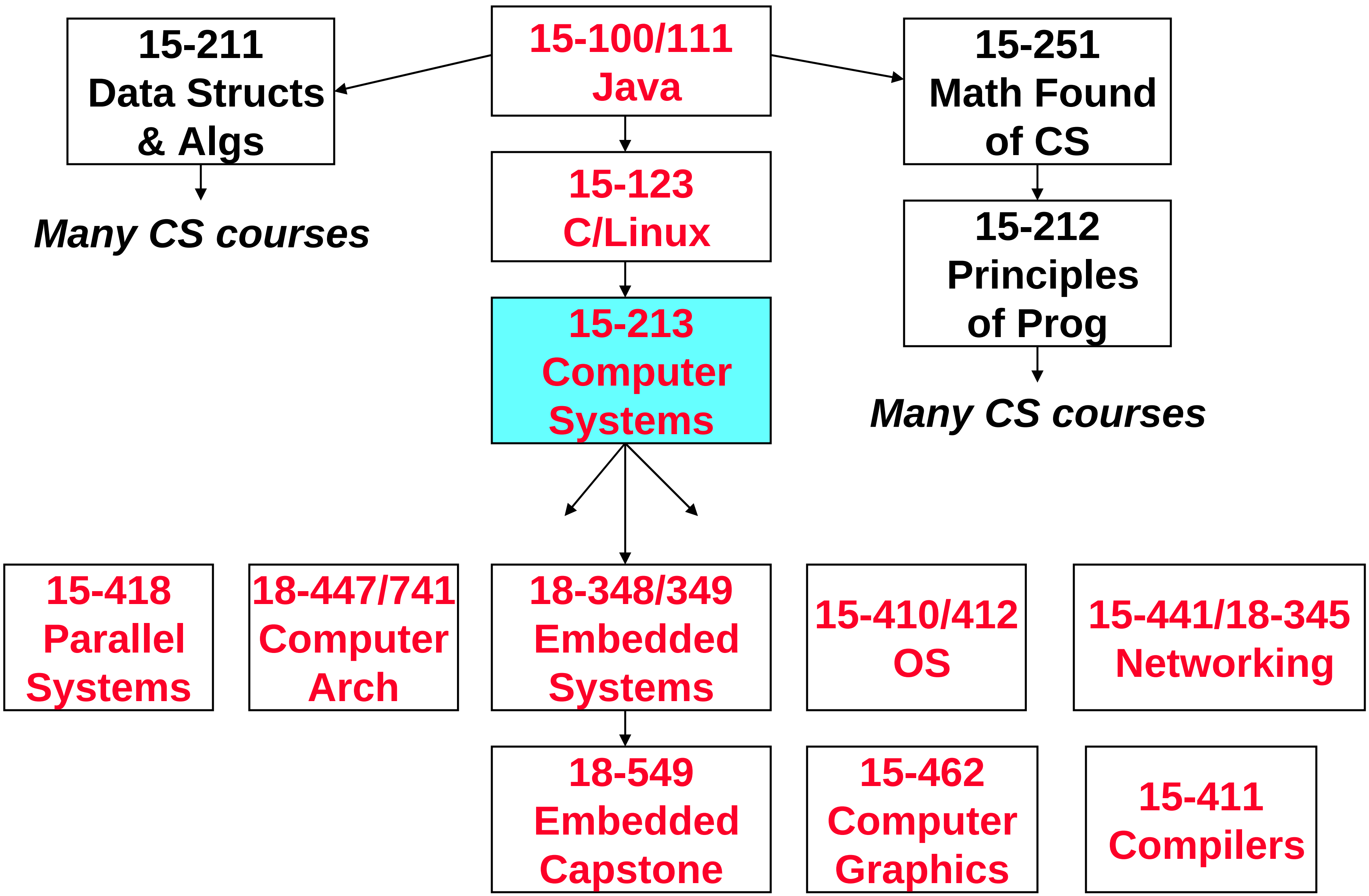
The best engineers understand all levels!

Computer Systems



The ECE "software" track introduces you to the intellectual area of Computer Systems.

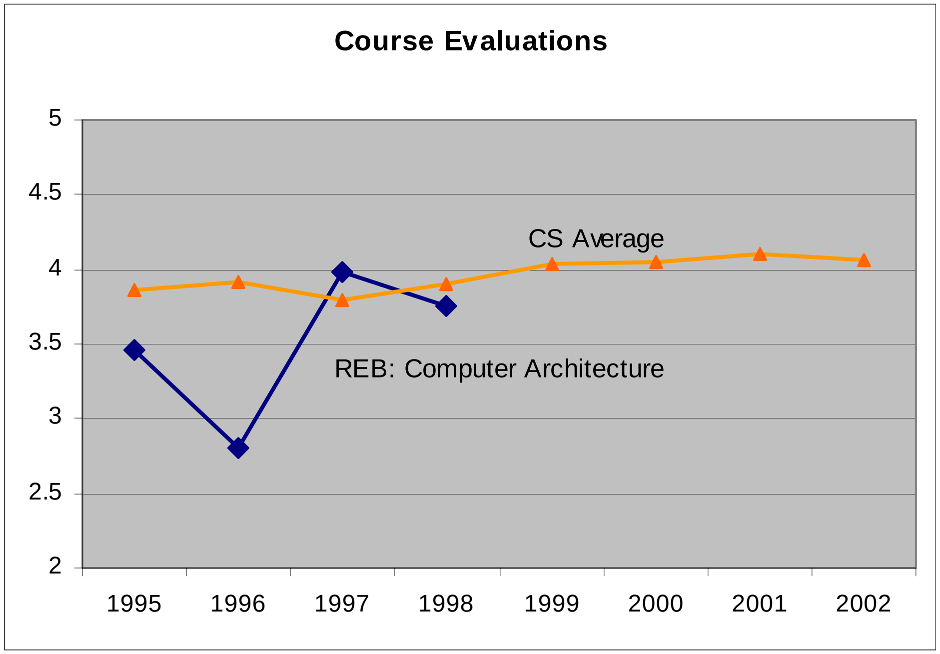
Computer Systems Courses



15-213: Intro to Computer Systems (ICS)

1995-1997: REB/DROH teaching computer architecture/organization course at CMU.

- **Good material, dedicated teachers, but students hate it**
- **Don't see how it will affect their lives as programmers**



ICS Background (cont)

1997: OS instructors complain about lack of preparation

- **Students don't know machine-level programming well enough**
  - » **What does it mean to store the processor state on the run-time stack?**
- **Our architecture course was not part of prerequisite stream**

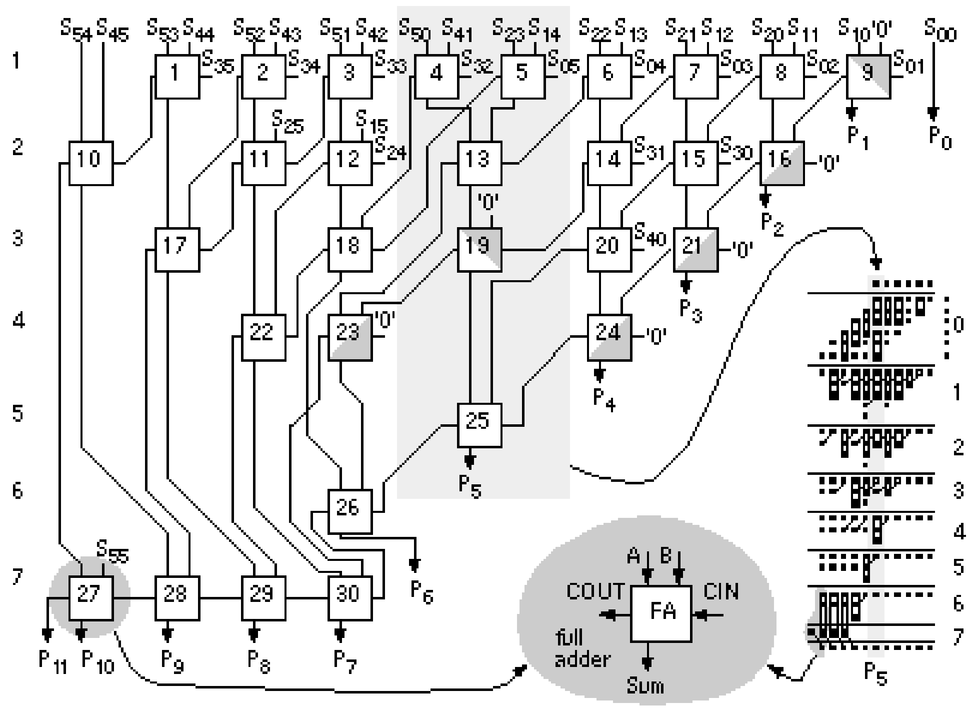
ICS Background

1997: REB/DROH pursue new idea:

Introduce students to computer systems from a programmer's perspective rather than a system designer's perspective.

Topic Filter: What parts of a computer system affect the correctness, performance, and utility of my C programs?

Computer Arithmetic
Builder's Perspective



– **How to design high performance arithmetic circuits**

Computer Arithmetic
Programmer's Perspective

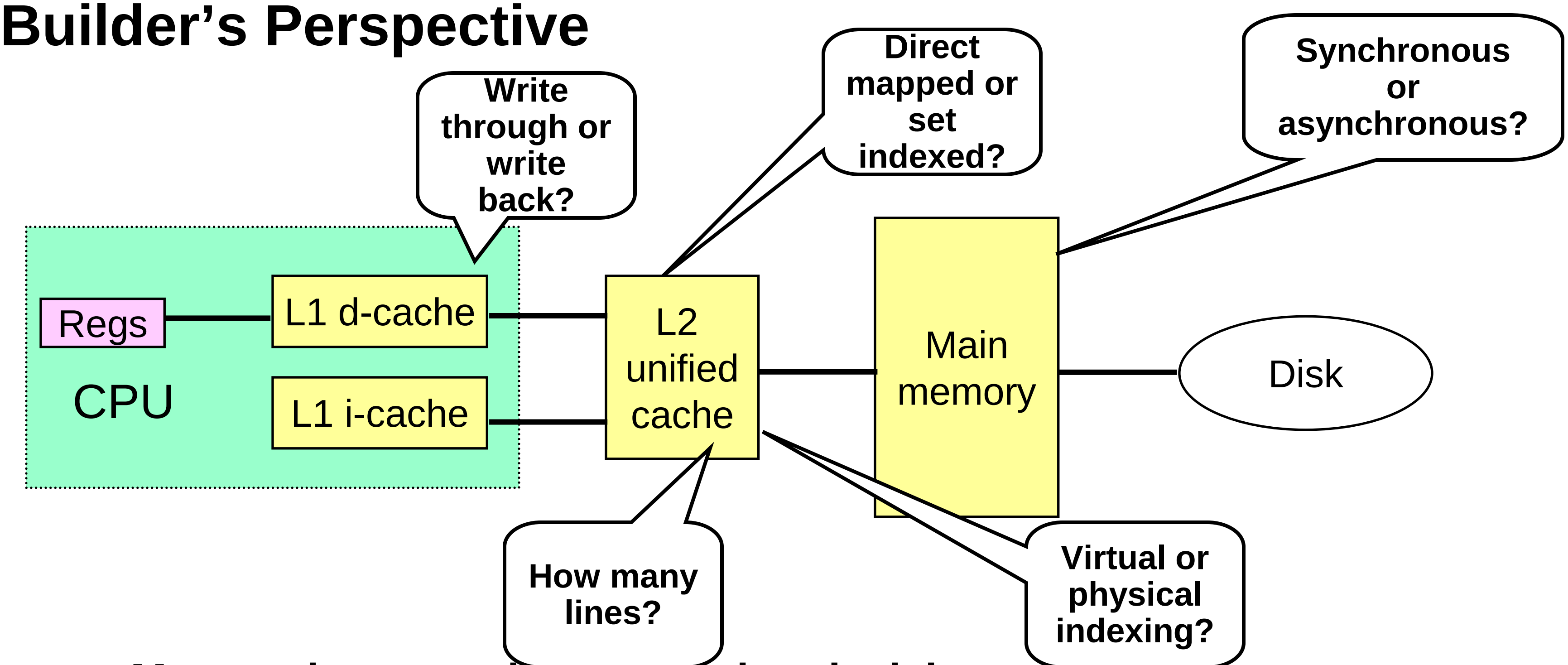
```
void show_squares()
{
  int x;
  for (x = 5; x <= 5000000; x*=10)
    printf("x = %d x^2 = %d\n", x, x*x);
}
```

```
x =       5   x² =          25
x =      50   x² =        2500
x =     500   x² =      250000
x =    5000   x² =    25000000
x =   50000   x² = -1794967296
x =  500000   x² =   891896832
x = 5000000   x² = -1004630016
```

– **Numbers are represented using a finite word size**
– **Operations can overflow when values too large**
  » **But behavior still has clear, mathematical properties**

Memory System
Builder's Perspective

Builder's Perspective



- Must make many difficult design decisions
- Complex tradeoffs and interactions between components

Memory System
Programmer's Perspective

```
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];
}
```

59,393,288 clock cycles

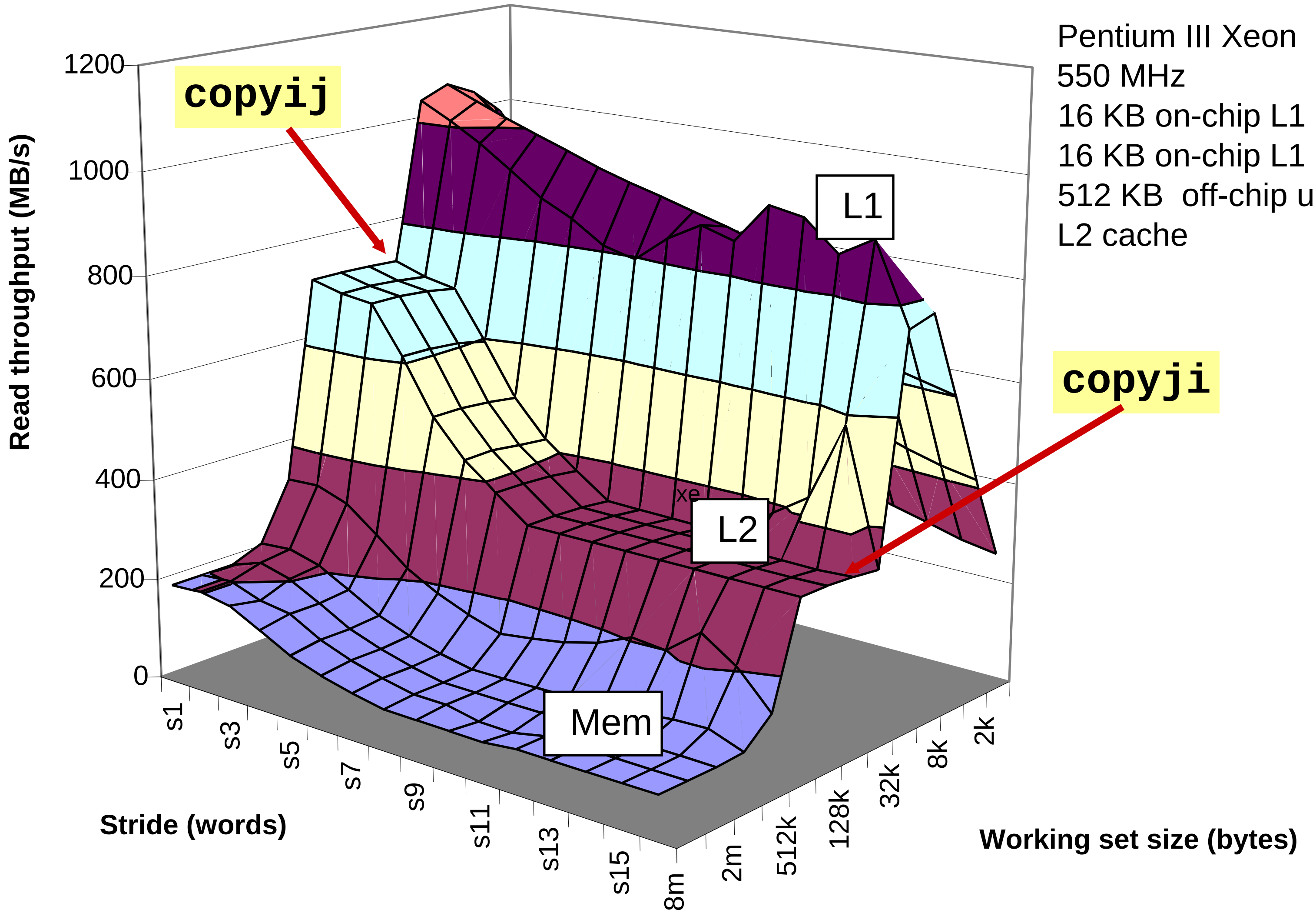
1,277,877,876 clock cycles

21.5 times slower!

(Measured on 2GHz Intel Pentium 4)

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

The Memory Mountain



Pentium III Xeon
550 MHz
16 KB on-chip L1 d-cache
16 KB on-chip L1 i-cache
512 KB off-chip unified
L2 cache

15-213: Intro to Computer Systems

Goals

- Introduced in 1998
- Teach students to be sophisticated application programmers
- Prepare students for upper-level systems courses

Taught every semester to 150 students

- 50% CS, 40% ECE, 10% other.

Part of the 4-course CMU CS core:

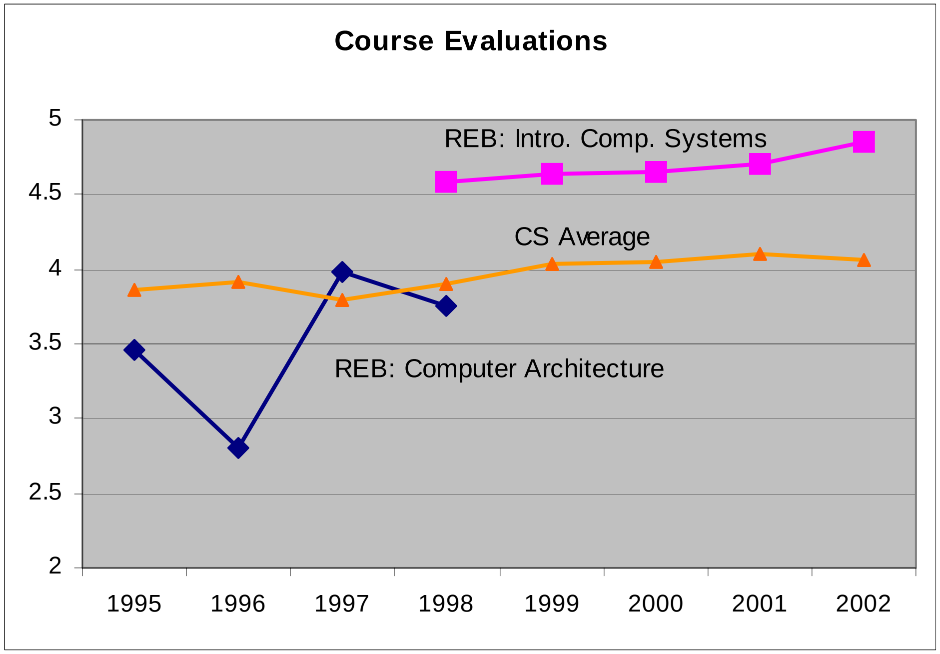
- Data structures and algorithms (Java) (15-211)
- Computer systems (C) (15-213)
- Fundamentals of Programming (ML) (15-212)
- Mathematical foundations of CS (15-251)

Will (likely) become part of new ECE core in Fall’07

- Circuits, Logic design, Computer systems, Signal processing

ICS Feedback

Students



Faculty

- Prerequisite for most upper level CS systems courses
- Also required for ECE embedded systems, architecture, and network courses. Added to ECE required core in Fall 2007.

Lecture Topics

Data representations [3]

- It's all just bits.
- `int`'s are not integers and `float`'s are not reals.

IA32 machine language [5]

- Analyzing and understanding compiler-generated machine code.

Program optimization [2]

- Understanding compilers and modern processors.

Memory Hierarchy [3]

- Caches matter!

Linking [1]

- With DLL's, linking is cool again!

Lecture Coverage (cont)

Exceptional Control Flow [2]

- The system includes an operating system that you must interact with.

Measuring performance [1]

- Accounting for time on a computer is tricky!

Virtual memory [4]

- How it works, how to use it, and how to manage it.

I/O and network programming [4]

- Programs often need to talk to other programs.

Application level concurrency [2]

- Processes, I/O multiplexing, and threads.

Total: 27 lectures, 14 week semester.

Labs

Key teaching insight:

- Cool Labs $\Rightarrow$ Great Course

A set of 1 and 2 week labs define the course.

Guiding principles:

- Be hands on, practical, and fun.
- Be interactive, with continuous feedback from automatic graders
- Find ways to challenge the best while providing worthwhile experience for the rest
- Use healthy competition to maintain high energy.

Fostering "Friendly Competition"

Desire

- Challenge the best without blowing away everyone else

Method

- Web-based submission of solutions
- Server checks for correctness and computes performance score
  - » How many stages passed, program throughput, …
- Keep updated results on web page
  - » Students choose own nickname

Relationship to Grading

- Students get full credit once they reach set threshold
- Push beyond this just for own glory/excitement

Lab Exercises

Data Lab (2 weeks)

- Manipulating bits.

Bomb Lab (2 weeks)

- Defusing a binary bomb.

Buffer Lab (1 week)

- Exploiting a buffer overflow bug.

Performance Lab (2 weeks)

- Optimizing kernel functions.

Shell Lab (1 week)

- Writing your own shell with job control.

Malloc Lab (2-3 weeks)

- Writing your own malloc package.

Proxy Lab (2 weeks)

- Writing your own concurrent Web proxy.

Bomb Lab

– Idea due to Chris Colohan, TA during inaugural offering

Bomb: **C program with six *phases*.**

Each phase expects student to type a specific string.

- Wrong string: bomb *explodes* by printing BOOM! (- 1/4 pt)
- Correct string: phase *defused* (+10 pts)
- In either case, bomb sends a message to a grading server
- Grading server posts current scores anonymously and in real time on Web page

Goal: Defuse the bomb by defusing all six phases.

The kicker:

- Students get only the binary executable of a *unique* bomb
- To defuse their bomb, students must disassemble and reverse engineer this binary

The Beauty of the Bomb

Get a deep understanding of machine code in the context of a fun game

Learn about machine code in the context they will encounter in their professional lives

- Working with compiler-generated code

Learn concepts and tools of debugging

- Forward vs backward debugging
- Students *must* learn to use a debugger to defuse a bomb

Summary and Conclusions

Claim: The best engineers understand computer systems at all levels of abstraction, *including the software levels.*

Carnegie Mellon ECE students take courses in computer systems that are offered by both the CS and ECE departments

***15-213 – Introduction to Computer Systems* is the prereq for all upper level systems courses.**