cse141: Introduction to Computer Architecture

Steven Swanson
Andiry Xu
Qi Li
Today’s Agenda

• What is architecture?
• Why is it important?
• What’s in this class?
Computer Architecture
What is architecture?

• How do you build a machine that computes?
  • Quickly, safely, cheaply, efficiently, in technology X, for application Y, etc.

• Architects develop new mechanism for performing and organizing “mechanical” computation
Why is architecture important?

• For the world
  • Computer architecture provides the engines that power all of computing

Civilization advances by extending the number of important operations which we can perform without thinking about them.

-- Alfred North Whitehead
Why is architecture important?

• For the world
  • Computer architecture provides the engines that power all of computing

Civilization advances by extending the number of important operations which we can perform without thinking about them.

  -- Alfred North Whitehead

• For you
  • As computer scientists, software engineers, and sophisticated users, understanding how computers work is essential
  • The processor is the most important piece of this story
  • Many performance (and efficiency) problems have their roots in architecture.
Orientation

The internet
Form Factors (to scale)

- Architecturally, these machines are more similar than different
  - Same parts
  - Different Scale
  - Different Constraints
Orientation: A Server

Architecture begins about here.
Architecture begins about here.
Orientation: iPhone 4s

Flash Memory on the back

Peripheral

Sim Card

CPU + DRAM

Architecture begins about here.
You are here

Nehalem Corei7 Quad-core Server processor

Nvidia Tegra 3 Five-core mobile processor
Processors are everywhere!
Abstractions of the Physical World...

\[ \int \mathbf{H} \cdot d\mathbf{l} = I + \varepsilon \frac{d}{dt} \iint \mathbf{E} \cdot d\mathbf{s} \]
\[ \int \mathbf{E} \cdot d\mathbf{l} = -\mu \frac{d}{dt} \iint \mathbf{H} \cdot d\mathbf{s} \]
\[ \mu \oint \mathbf{H} \cdot d\mathbf{s} = 0 \]
\[ \varepsilon \oint \mathbf{E} \cdot d\mathbf{s} = \iiint \mathbf{q} \cdot d\mathbf{v} \]
for the Rest of the System

cse121  cse131  cse130  cseEverythingElse

Architectures
Processor Abstraction
Compilers
Languages
Software Engineers/Applications
Current state of architecture
Moore's Law

- The number of transistors we can build in a fixed area of silicon doubles (roughly) every two years.

Moore's Law is the most important driver for historic CPU performance gains.
Since 1940

50,000 x speedup
>1,000,000,000 x density
(Moore’s Law)

Plug boards -> Java
Hand assembling -> GCC
No OS -> Windows 7

We have used this performance to make computers easier to use, easier to program, and to solve ever-more complicated problems.
Where do We Get Performance?

Golden age: ~40-50%/year

Modern era: ~25%/year

Clock speed

Relative Performance or Clock speed (Mhz)

Year

specINT95 Perf
specINT2000 Perf
specINT2006 Perf
specINT2000 Mhz
specINT2006 Mhz
The End of Clock Speed Scaling

• Clock speed is the biggest contributor to power
  • Chip manufactures (Intel, esp.) pushed clock speeds very hard in the 90s and early 2000s.
  • Doubling the clock speed increases power by 2-8x
  • Clock speed scaling is essentially finished.

• Most future performance improvements will be due to architectural and process technology improvements
Power and heat

Figure 1. In CPU architecture today, heat is becoming an unmanageable problem. 
(Courtesy of Pat Gelsinger, Intel Developer Forum, Spring 2004)
The Rise of Parallelism

• Multi-processors
  • If one CPU is fast, two must be faster!
  • They allow you to (in theory) double performance without changing the clock speed.

• Seems simple, so why are becoming so important now
  • Speeding up a single CPU makes everything faster!
    • An application’s performance double every 18 months with no effort on the programmer’s part.
  • Getting performance out of a multiprocessor requires work.
    • Parallelizing code is difficult, it takes (lots of) work
    • There aren’t that many threads
    • Remember or look forward to cse120
Why This Class?
The Goal of a Degree in CS or CE (My $0.02)

• To understand the components and abstractions that make up a modern computing system
• To understand how they impact a system’s performance, efficiency, and usefulness
• To be able to harness, modify, and extend them to solve problems effectively
Goals for this Class

• Understand how CPUs run programs
  • How do we express the computation the CPU?
  • How does the CPU execute it?
  • How does the CPU support other system components (e.g., the OS)?
  • What techniques and technologies are involved and how do they work?
• Understand why CPU performance (and other metrics) vary.
  • How does CPU design impact performance?
  • What trade-offs are involved in designing a CPU?
  • How can we meaningfully measure and compare computer systems?
• Understand why program performance varies
  • How do program characteristics affect performance?
  • How can we improve a program’s performance by considering the CPU running it?
  • How do other system components impact program performance?
What’s in this Class

- Instruction sets
  - MIPS
  - x86
  - ISAs and the compiler
- The processor pipeline
  - Basic design
  - Pipelining
  - Dealing with hazards
  - Speculation and control
- Measuring performance
  - Amdahl’s Law
  - Performance measurement
  - Metrics
- The memory system
  - Memory technologies
  - Caching
  - Operating system support
    - Virtual memory
    - Exceptions, interrupts
    - IO
  - Introduction to multiprocessors
Performance and You!

• Live Demo

cd demos/
make
java  -server -Xmx$[1024*1024*1024]  -Xmx$[1024*1024*1024]  LoopNest 1000 ij
java  -server -Xmx$[1024*1024*1024]  -Xmx$[1024*1024*1024]  LoopNest 1000 ji
cse141 Logistics
Course Staff

- Instructor: Steven Swanson
  - Lectures Tues + Thurs
  - Office hours TBA
- TA: Andiry Xu and Qi Li
  - Discussion sec: Wednesday 4-4:50 (Center 214).
  - Office ours TBA
- See the course web page for contact information and office hours:
  - http://cseweb.ucsd.edu/classes/sp14/cse141-a/
Academic Honesty

• Don’t cheat.
  • Cheating on a test will get you an F in the class and no option to drop, and a visit with your college dean.
  • Cheating on homeworks means you don’t have to turn them in any more, but you don’t get points either. You will also take at least 25% penalty on the exam grades.

• Copying solutions of the internet or a solutions manual is cheating.
• Review the UCSD student handbook
• When in doubt, ask.
Course Structure
Conventional Lectures

• Traditional class structures often look like:

  Lecture → Textbook → Homework → Exam

  - First Exposure
  - Read Hard Stuff
  - See if You Know Hard Stuff
  - Show Knowledge Mastery

• This structure is backwards
  • You see things for the first time in class.
  • Then you wrestle with the hard parts on your own.
Peer Instruction-Based Lectures

- Better structure
  - First exposure on your own (i.e., read the book)
  - I help with the hard stuff in class.
- Greater opportunity for expert feedback and interaction!
Science to Back it Up

Crouch, C., Mazur, E. Peer Instruction: Ten years of experience and results

![Bar chart showing normalized gain over years with bars for Traditional Instruction and Peer Instruction]
Science to Back it Up

• Everyone constructs their own understanding
  • I can’t dump understanding into your brain
• To learn YOU must actively work with a problem and construct your own understanding of it
  • I’m going to help you do this *in class.*
  • You must do it *outside* of class as well.

It’s like muscle development!

Strenuous, repeated effort ->
New Muscle Cells

Strenuous, repeated effort ->
New Neurons, Links!

Development of new neurons in response to difficult learning task
T. Shors, Sci. Amer. Mar 09
Students say discussion is helpful...

- It really makes you realize exactly what mistakes you are making and sometimes you don’t feel as bad if you are wrong because you can see that fellow classmates think the same way.
- Discussion is really helpful as sometimes when you get lost in other classes, you are lost for the rest of the lecture. Discussion and clicker questions help make students realize when they are getting confused before it is too late and the discussions with classmates helps get us back on track.

*From an upper division computer science course*
Students at UCSD LIKE *Peer Instruction* with clickers!

<table>
<thead>
<tr>
<th></th>
<th>Lower Division Non-Majors Computing</th>
<th>Upper Division Math-Based Computing</th>
<th>Upper Division Computing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valuable for my learning</td>
<td>91%</td>
<td>80%</td>
<td>79%</td>
</tr>
<tr>
<td>Recommend other instructors use</td>
<td>87%</td>
<td>91%</td>
<td>90%</td>
</tr>
</tbody>
</table>
Lecture: Peer Instruction

• Are you prepared?
  • Quick quiz at beginning of class, using clickers
  • Correct answers count toward your grade!
• Pose carefully designed question
  • Participation counts toward your grade (not correctness)
  • Solo vote: Think for yourself and select answer
  • Discuss: Analyze problem in teams of 3
    • Practice analyzing, talking about challenging concepts
    • Reach consensus
    • If you have questions, raise your hand and I or the TAs will come around
• Group vote: Everyone in group votes
  • You must all vote the same to get your point
• Class wide discussion:
  • Led by YOU (students) – tell us what you talked about in discussion that everyone should know!
Your Tasks

• Sign up for the mailing lists.
• Read the text!
  • Computer Organization and Design: The Hardware/Software Interface (5th Edition) -- previous editions are not supported
  • I’m not going to cover everything in class, but you are responsible for all the assigned text.

• Come to class!
  • I will cover things not in the book.
  • You are responsible for them too.

• Using your clicker (10%)
• Reading quizzes (10%)
• Homeworks throughout the course. (20%)
• One midterm (25%)
• One cumulative final (35%)
Homeworks

• Assigned on Thursday, due one week later
• Partly from the book.
• These are the best way to prepare for the tests.
• Due in a TA’s box, 15 minutes before class starts.
  • Check the assignment for which TA to turn it in to.
  • The mailboxes are located in the grad student mail room on the second floor of the CSE building.
Peer instruction

• I’ll bring in activities to ENGAGE you in exploring your understanding of the material
  • Let you practice
  • Bring out misconceptions
  • Let us LEARN from each other about difficult parts.
• You will be GET CREDIT for your efforts to learn in class
  • By answering questions with a clicker (iClicker)
  • Answer 80% of the clicker questions in class, get 10% of your final grade
  • Process: Individual Answer, Group Discussion, Group Answer
• Register your i-clicker
  • Log into Ted.
  • Choose “Course tools” from the blue menu on the left.
  • Choose “i>Clicker Remote Registration”
• Set your channel to “DC”
Grading

• Grading is on a 13 point scale -- F through A+
  • You will get a letter grade on each assignment
  • Your final grade is the weighted average of the assignment grades.

• An excel spreadsheet calculates your grades
  • We will post a sanitized version online once a week.
  • It will tell you exactly where you stand.
  • It specifies the curves used for the exams etc.

• OpenOffice doesn’t run it properly.
The Link to 141L

• You do not need to take 141L along with 141, but you may need both to get your degree.
• The classes are mostly independent, except
  • You will use what we learn about ISA design in 141L
  • You will use what we learn about pipeline in 141L.
• Questions specific to 141L should go to John Eldon (the instructor).