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
• Architecturally, these machines are more similar than different
  • Same parts
  • Different Scale
  • Different Constraints

Motherboards to scale
Orientation: A Server

Architecture begins about here.
Orientation: MacBook Air

Architecture begins about here.
Orientation: iPhone 4s

Architecture begins about here.
Nehalem Corei7 Quad-core Server processor

Nvidia Tegra 3 Five-core mobile processor
The processors go here...
The processors go here...
The processors go here…
The processors go here...
The processors go here...
The processors go here...
The processors go here...
The processors go here…
The processors go here...
The processors go here...
The processors go here...
Abstractions of the Physical World...

\[ \oint H \cdot dl = I + \varepsilon \frac{d}{dt} \iint E \cdot ds \]
\[ \oint E \cdot dl = -\mu \frac{d}{dt} \iint H \cdot ds \]
\[ \mu \iint H \cdot ds = 0 \]
\[ \varepsilon \iint E \cdot ds = \iint \iint q \cdot dv \]

Physics/Materials  Devices  Micro-architecture  Processors  Architectures
Abstractions of the Physical World…

Physics/Chemistry/Material Science

\[
\begin{align*}
\oint H \cdot dl &= I + \varepsilon \frac{d}{dt} \int \int E \cdot ds \\
\oint E \cdot dl &= -\mu \frac{d}{dt} \int \int H \cdot ds \\
\mu \oint H \cdot ds &= 0 \\
\varepsilon \oint E \cdot ds &= \int \int q_v dv
\end{align*}
\]

Physics/Materials Devices

This Course

Micro-architecture Processors Architectures
...for the Rest of the System
…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
Since 1940

50,000 x speedup
>1,000,000,000 x density
(Moore’s Law)
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
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.
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?
Where do We Get Performance?

Clock speed
Where do We Get Performance?

Clock speed

Golden age:
~40-50%/year
Where do We Get Performance?

Clock speed

Golden age: ~40-50%/year

Modern era: ~25%/year
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 doubles every 4 years
5-year projection: 200W total, 125 W/cm²!

From “New Microarchitecture Challenges in the Coming Generations of CMOS Process Technologies”
Power

Power doubles every 4 years
5-year projection: 200W total, 125 W/cm²!

From “New Microarchitecture Challenges in the Coming Generations of CMOS Process Technologies”
Power doubles every 4 years
5-year projection: 200W total, 125 W/cm²!

From “New Microarchitecture Challenges in the Coming Generations of CMOS Process Technologies”
Power

Power doubles every 4 years
5-year projection: 200W total, 125 W/cm²!

From “New Microarchitecture Challenges in the Coming Generations of CMOS Process Technologies”
Power

Power doubles every 4 years
5-year projection: 200W total, 125 W/cm²!

Power

Power doubles every 18 months - 5-year projections. W/cm²!

Watts/cm²

1.5µ 1µ 0.7µ 0.5µ 0.35µ 0.25µ 0.18µ 0.13µ 0.1µ 0.07µ

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.
- Seem 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
Intel P4 (2000)  
1 core

Intel Core 2 Duo (2006)  
2 cores

Intel Nahalem (2010)  
4 cores

SPARC T3 (2010)  
16 cores

Nvidia Tegra 3 (2011)  
5 cores

AMD Zambezi (2011)  
16 cores
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
cse141 Logistics
Course Staff

- **Instructor:** Steven Swanson
  - Lectures Tues + Thurs
  - Office hours TBA

- **TA:** Alice Liang
  - Discussion sec: Wednesday.

- See the course web page for contact information and office hours:
  - [http://cseweb.ucsd.edu/classes/sp13/cse141-a/](http://cseweb.ucsd.edu/classes/sp13/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.
Your Tasks

• Sign up for the mailing lists.
• Read the text!
  • Computer Organization and Design: The Hardware/Software Interface (4th Edition, revised) -- 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.
• Homeworks throughout the course. (20%)
• Weekly quizzes on Thursdays (20%)
• One midterm (25%)
• One cumulative final (35%)
Quizzes

- Every Thursday, online.
- Covers everything up to and including the previous class.
- 20 Minutes, 4-5 questions.
- Roughly 2% of your grade each.
- No make-ups.
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.
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
  - We will study the MIPS ISA in 141, and you will implement it in 141L
  - The discussions about processor implementation in 141 will be useful in 141L.
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.