CIS 501: Computer Architecture

Unit 1: Introduction

Slides developed by Joe Devietti, Milo Martin & Amir Roth at UPenn with sources that included University of Wisconsin slides by Mark Hill, Guri Sohi, Jim Smith, and David Wood
Course Overview
Why Study Computer Architecture?

• **Understand where computers are going**
  • Future capabilities drive the (computing) world
  • Real world-impact: no computer architecture → no computers!

• **Understand high-level design concepts**
  • The best architects understand all the levels
    • Devices, circuits, architecture, compiler, applications

• **Understand computer performance**
  • learn valid experimental methodologies

• **Write better software**
  • The best software designers also understand hardware
  • Need to understand hardware to write fast software

• **Design hardware**
  • At Intel, AMD, IBM, ARM, Qualcomm, Oracle, NVIDIA, Samsung
Penn Legacy

- **ENIAC**: electronic numerical integrator and calculator
  - First operational general-purpose stored-program computer
  - Designed and built here by Eckert and Mauchly from 1943-6
  - Go see it (Moore building)

- **First seminars on computer design**
  - Moore School Lectures, 1946
  - "Theory and Techniques for Design of Electronic Digital Computers"
Course Goals

- **See the big ideas in computer architecture**
  - Pipelining, parallelism, caching, locality, abstraction, speculation, ...
  - Be able to understand Anandtech/Chipworks articles

- Exposure to examples of good (and some bad) engineering

- Understanding computer performance and metrics
  - Experimental evaluation/analysis (“science” in computer science)
  - Gain experience with simulators (architect’s tool of choice)

- Get exposure to research and cutting edge ideas
  - Read some research papers

- **My role: trick you into learning something**
Computer Science as an Estuary

**Engineering**
- Design
- Handling complexity
- Real-world impact
- Examples: Internet, microprocessor

**Science**
- Experiments
- Hypothesis
- Examples:
  - Internet behavior,
  - Protein-folding supercomputer
  - Human/computer interaction

**Mathematics**
- Limits of computation
- Algorithms & analysis
- Cryptography
- Logic
- Proofs of correctness

**Other Issues**
- Public policy, ethics, law, security

Where does architecture fit into computer science?
Engineering, some Science
Course Topics

• More depth on “undergraduate” architecture topics
  • Evaluation metrics and trends
  • ISAs (instruction set architectures)
  • Datapath & pipelining (including branch prediction)
  • Memory hierarchies, caches, & virtual memory

• Overview of semiconductor technology & energy/power

• Parallelism
  • Instruction: multiple issue, dynamic scheduling, speculation
  • Thread: cache coherence and synchronization, multicore
  • Data: vectors and GPUs

• More fun stuff if we get to it
CIS501: Administrivia

• Instructor: Prof. Joe Devietti (devietti@cis.upenn.edu)
  • TAs: Akshitha Sriraman and Liang Luo

• Three important web sites
  • Course website: syllabus, schedule, lecture notes
    • https://www.cis.upenn.edu/~cis501/
  • Piazza: announcements, questions & discussion
    • https://piazza.com/class/i4lg7532cd62ex
  • The way to ask questions/clarifications
    • Can post to just me & TAs or anonymous to class
    • As a general rule, no need to email me & TAs directly
  • Canvas: assignments
    • https://canvas.upenn.edu/

• Lectures are recorded
  • link on course website
Resources

• Readings
  • Research papers (linked from course web page)

• Optional readings
  • “Microprocessor Architecture: From Simple Pipelines to Chip Multiprocessors” by Jean-Loup Baer

• Previous course (to review lectures prior to class):
  • http://cis.upenn.edu/~devietti/classes/cis501-fall2013/

• Free resources
  • ACM digital library: http://www.acm.org/dl/
  • Computer architecture page: http://www.cs.wisc.edu/~arch/www/

• Local resources:
  • Architecture & Compilers Group: http://acg.cis.upenn.edu/
Prerequisites

• **Basic computer organization an absolute must**
  • Basic digital logic: gates, boolean functions, latches
  • Binary arithmetic: adders, hardware mul/div, floating-point
  • Basic datapath: ALU, register file, memory interface, muxes
  • Basic control: single-cycle control, microcode
• **Familiarity with assembly language**
  • “Computer Organization and Design: Hardware/Software Interface”
  • [https://www.cis.upenn.edu/~cis371/](https://www.cis.upenn.edu/~cis371/)

• **Significant programming experience**
  • No specific language required, C++/Java recommended
  • Why? assignments require writing code to **simulate hardware**
    • Not difficult if competent programmer; extremely difficult if not
For Non-CIS Students...

- Registration priority is given to CIS students, PhD students, and 2nd-year non-CIS master’s students
- For non-CIS students, email me (devietti@cis):
  1. Your name & Penn email address
  2. What program you’re enrolled in
  3. What year you are
  4. A transcript of your Penn courses with grades
  5. Description of prior courses on computer architecture
  6. A brief description of the largest programming project you’ve completed (lines of code, overall complexity, language used, etc.)
Coursework

• **Paper reviews (5 throughout the semester)**
  • Short response to papers we’ll read for class
  • Discuss and write up in **groups** of four
    • Twist: can’t work with the same group member twice

• **Homework assignments (6-7 throughout semester)**
  • Written questions and **programming**
  • two 48-hour “grace” periods, **max one per assignment**
    • Hand in late, no questions asked
  • **No assignments accepted after solutions posted**
  • **Individual work only**

• **Exams (2 exams)**
  • Midterm, **Wed 4 March 1:30-3pm**
    • in-class
  • **Cumulative** final (WPE I for PhD students)
    • **Tue 5 May noon-2pm**
    • location: TBD
    • Email me if you have 3+ exams scheduled for this day
Grading

• Tentative grade contributions:
  • Paper reviews: 5%
  • Homework assignments: 20%
  • Exams: 75%
    • Midterm: 30%
    • Final: 45%
  • Smiling: 0%

• Typical grade distributions
  • A: 40%
  • B: 40%
  • C/D/F: 20%
Academic Misconduct

- Cheating will **not** be tolerated

- General rule:
  - Anything with your name on it must be **YOUR OWN** work
  - Example: individual work on homework assignments

- Possible penalties
  - Zero on assignment (minimum)
  - Fail course
  - Note on permanent record
  - Suspension
  - Expulsion

- Penn’s Code of Conduct
  - [http://www.vpul.upenn.edu/osl/acadint.html](http://www.vpul.upenn.edu/osl/acadint.html)
Full Disclosure

- Potential sources of bias or conflict of interest

- Most of my funding is governmental (your tax $$$ at work)
  - National Science Foundation (NSF)
  - also some funding from Intel
What is Computer Architecture?
What is Computer Architecture?

• “Computer Architecture is the science and art of selecting and interconnecting hardware components to create computers that meet functional, performance and cost goals.” - WWW Computer Architecture Page

• An analogy to architecture of buildings…
What is Computer Architecture?

The role of a building architect:

- Materials
- Design
- Plans
- Goals
- Construction

Buildings
- Houses
- Offices
- Apartments
- Stadiums
- Museums
What is Computer Architecture?

The role of a *computer* architect:

"Technology" → Plans → Goals → Design → Manufacturing

**Computers**
- Desktops
- Servers
- Phones
- Supercomputers
- Game Consoles
- Embedded

**Important differences:** age (~60 years vs thousands), rate of change, automated mass production (magnifies design)
Computer Architecture Is Different...

- **Age of discipline**
  - 60 years (vs. five thousand years)

- **Rate of change**
  - All three factors (technology, applications, goals) are changing
  - Quickly!

- **Automated mass production**
  - Design advances magnified over millions of chips

- **Boot-strapping effect**
  - Better computers help design next generation
Design Goals & Constraints

- **Functional**
  - Needs to be correct
    - And unlike software, difficult to update once deployed
  - What functions should it support (Turing-completeness aside)

- **Reliable**
  - Does it *continue* to perform correctly?
  - Hard fault vs transient fault
  - Space satellites vs desktop vs server

- **High performance**
  - “Fast” is only meaningful in the context of a set of important tasks
  - Not just “Gigahertz” – truck vs sports car analogy
  - Impossible goal: fastest possible design for all programs
Design Goals & Constraints

• **Low cost**
  - Per unit manufacturing cost (wafer cost)
  - Cost of making first chip after design (mask cost)
  - Design cost (huge design teams, why? Two reasons...)

• **Low power/energy**
  - Energy in (battery life, cost of electricity)
  - Energy out (cooling and related costs)
  - Cyclic problem, very much a problem today

• **Challenge: balancing the relative importance of these goals**
  - And the balance is constantly changing
    - No goal is absolutely important at expense of all others
  - Our focus: *performance*, only touch on cost, power, reliability
Shaping Force: Applications/Domains

- Another shaping force: **applications** (usage and context)
  - Applications and application domains have different requirements
    - Domain: group with similar character
    - Lead to different designs

- **Scientific**: weather prediction, genome sequencing
  - First computing application domain: naval ballistics firing tables
  - Need: large memory, heavy-duty floating point
  - Examples: CRAY T3E, IBM BlueGene

- **Commercial**: database/web serving, e-commerce, Google
  - Need: data movement, high memory + I/O bandwidth
  - Examples: Sun Enterprise Server, AMD Opteron, Intel Xeon
More Recent Applications/Domains

- **Desktop**: home office, multimedia, games
  - Need: integer, memory bandwidth, integrated graphics/network?
  - Examples: Intel Core i*, AMD Athlon

- **Mobile**: laptops, tablets, phones
  - Need: **low power**, integer performance, integrated wireless
  - Laptops: Intel Core i*, Atom, AMD APUs
  - Smaller devices: ARM chips by Samsung, Qualcomm, Apple

- **Embedded**: microcontrollers in automobiles, door knobs
  - Need: low power, **low cost**
  - Examples: ARM chips, dedicated digital signal processors (DSPs)
  - 10 billion ARM chips sold in 2013

- **Deeply Embedded**: disposable “smart dust” sensors
  - Need: extremely low power, extremely low cost
Application Specific Designs

• This class is about general-purpose CPUs
  • Processor that can do anything, run a full OS, etc.
  • E.g., Intel Core i7, AMD Athlon, IBM Power, ARM, Intel Itanium

• In contrast to application-specific chips
  • Or ASICs (Application specific integrated circuits)
    • Also application-domain specific processors
  • Implement critical domain-specific functionality in hardware
    • Examples: video encoding, 3D graphics
  • General rules
    - Hardware is less flexible than software
    + Hardware more effective (speed, power, cost) than software
    + Domain specific more “parallel” than general purpose
      • But general mainstream processors becoming more parallel

• Trend: from specific to general (for a specific domain)
Next steps
First Assignment – Paper Review #1

• Read “Cramming More Components onto Integrated Circuits” by Gordon Moore

• As a group of four, meet and discuss the paper
  • can find group members via Piazza “Search for Teammates!”
  • Briefly answer the questions on the next slide
  • The goal of these questions is to get you reading, thinking about, and discussing the paper
  • Your answers should be short but insightful. For most questions, a single short paragraph will suffice

• Submit answers via Canvas
  • join “paper review #1 group N” group on Canvas
  • instructions on Piazza
  • submit one set of answers per group
  • include the names of all group members in the submission

• Due: Tue 27 Jan at 5pm
  • In-class discussion on Wed 28 Jan
For Next Week...

1. **Sign up for CIS 501 on Piazza**
   - instructions at the top of [http://cis.upenn.edu/~cis501/](http://cis.upenn.edu/~cis501/)

2. Assigned readings:
   - “Cramming More Components onto Integrated Circuits” by Moore
     - Use “group” on Piazza feature to find a group

3. If you’re a **non-CIS student** wanting to take 501
   - **send email** as discussed earlier

4. **Let’s talk now** if:
   - You have any other questions about prerequisites or the course