CIS 501 Computer Architecture

Unit 0: Introduction

Slides developed by Milo Martin & Amir Roth at the University of Pennsylvania with sources that included University of Wisconsin slides by Mark Hill, Guri Sohi, Jim Smith, and David Wood.

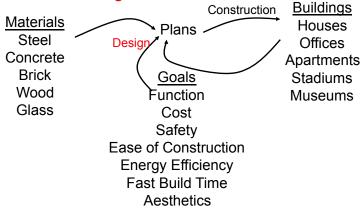
What is Computer Architecture?

- "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...

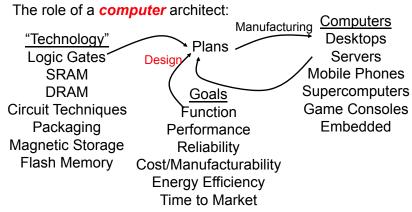
CIS 501 (Martin): Introduction 1 CIS 501 (Martin): Introduction 2

What is **Computer** Architecture?

The role of a **building** architect:



What is Computer Architecture?



Important differences: age (~60 years vs thousands), rate of change, automated mass production (magnifies design)

CIS 501 (Martin): Introduction 4

CIS 501 (Martin): Introduction 4

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

CIS 501 (Martin): Introduction

5

Design Goals

- 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...)
 - (Dime/dollar joke)
- 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

Design Goals

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
- Google story memory errors and sun spots
- Space satellites vs desktop vs server reliability

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

CIS 501 (Martin): Introduction

6

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

CIS 501 (Martin): Introduction 7 CIS 501 (Martin): Introduction 8

More Recent Applications/Domains

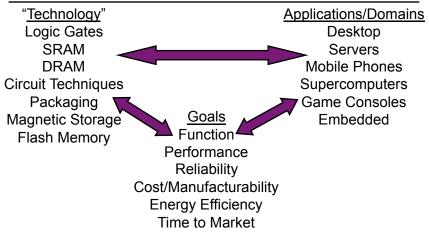
- **Desktop**: home office, multimedia, games
 - Need: integer, memory bandwidth, integrated graphics/network?
 - Examples: Intel Core 2, Core i7, AMD Athlon
- Mobile: laptops, mobile phones
 - Need: **low power**, integer performance, integrated wireless
 - Laptops: Intel Core 2 Mobile, Atom, AMD Turion
 - Smaller devices: ARM chips by Samsung and others, Intel Atom
- **Embedded**: microcontrollers in automobiles, door knobs
 - Need: low power, low cost
 - Examples: ARM chips, dedicated digital signal processors (DSPs)
 - Over 1 billion ARM cores sold in 2006 (at least one per phone)
- **Deeply Embedded**: disposable "smart dust" sensors
 - Need: extremely low power, extremely low cost

CIS 501 (Martin): Introduction

9

11

Constant Change: Technology



- Absolute improvement, different rates of change
- New application domains enabled by technology advances CIS 501 (Martin): Introduction

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)

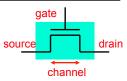
CIS 501 (Martin): Introduction 10

Technology Trends

CIS 501 (Martin): Introduction 12

"Technology"

- Basic element
 - Solid-state **transistor** (i.e., electrical switch)
 - Building block of integrated circuits (ICs)



- What's so great about ICs? Everything
 - + High performance, high reliability, low cost, low power
 - + Lever of mass production
- Several kinds of IC families
 - **SRAM/logic**: optimized for speed (used for processors)
 - **DRAM**: optimized for density, cost, power (used for memory)
 - Flash: optimized for density, cost (used for storage)
 - Increasing opportunities for integrating multiple technologies
- Non-transistor storage and inter-connection technologies
 - Disk, optical storage, ethernet, fiber optics, wireless

CIS 501 (Martin): Introduction 13

Technology Change Drives Everything

- Computers get 10x faster, smaller, cheaper every 5-6 years!
 - A 10x quantitative change is qualitative change
 - Plane is 10x faster than car, and fundamentally different travel mode
- New applications become self-sustaining market segments
 - Recent examples: mobile phones, digital cameras, mp3 players, etc.
- Low-level improvements appear as discrete high-level jumps
 - · Capabilities cross thresholds, enabling new applications and uses

Technology Trends

Moore's Law

- Continued (up until now, at least) transistor miniaturization
- Some technology-based ramifications
 - Absolute improvements in density, speed, power, costs
 - SRAM/logic: density: ~30% (annual), speed: ~20%
 - DRAM: density: ~60%, speed: ~4%
 - Disk: density: ~60%, speed: ~10% (non-transistor)
 - Big improvements in flash memory and network bandwidth, too

• Changing quickly and with respect to each other!!

- Example: density increases faster than speed
- Trade-offs are constantly changing
- Re-evaluate/re-design for each technology generation

CIS 501 (Martin): Introduction

Revolution I: The Microprocessor

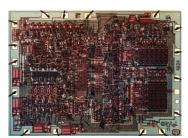
Microprocessor revolution

- One significant technology threshold was crossed in 1970s
- Enough transistors (~25K) to put a 16-bit processor on one chip
- Huge performance advantages: fewer slow chip-crossings
- Even bigger cost advantages: one "stamped-out" component
- Microprocessors have allowed new market segments
 - Desktops, CD/DVD players, laptops, game consoles, set-top boxes, mobile phones, digital camera, mp3 players, GPS, automotive
- And replaced incumbents in existing segments
 - Microprocessor-based system replaced supercomputers, "mainframes", "minicomputers", etc.

CIS 501 (Martin): Introduction 15 CIS 501 (Martin): Introduction 16

First Microprocessor

- Intel 4004 (1971)
 - Application: calculators • Technology: 10000 nm
 - · 2300 transistors
 - 13 mm²
 - 108 KHz
 - 12 Volts
 - · 4-bit data
 - · Single-cycle datapath



CIS 501 (Martin): Introduction

17

Pinnacle of Single-Core Microprocessors

- Intel Pentium4 (2003)
 - Application: desktop/server
 - Technology: 90nm (1/100x)
 - 55M transistors (20,000x)
 - 101 mm² (10x)
 - 3.4 GHz (10,000x)
 - 1.2 Volts (1/10x)
 - 32/64-bit data (16x)
 - 22-stage pipelined datapath
 - 3 instructions per cycle (superscalar)
 - Two levels of on-chip cache
 - data-parallel vector (SIMD) instructions, hyperthreading



CIS 501 (Martin): Introduction

18

Tracing the Microprocessor Revolution

- How were growing transistor counts used?
- Initially to widen the datapath
 - 4004: 4 bits → Pentium4: 64 bits
- ... and also to add more powerful instructions
 - To amortize overhead of fetch and decode
 - To simplify programming (which was done by hand then)

Revolution II: Implicit Parallelism

- Then to extract implicit instruction-level parallelism
 - Hardware provides parallel resources, figures out how to use them
 - Software is oblivious.
- Initially using pipelining ...
 - Which also enabled increased clock frequency
- ... caches ...
 - Which became necessary as processor clock frequency increased
- ... and integrated floating-point
- Then deeper pipelines and branch speculation
- Then multiple instructions per cycle (superscalar)
- Then dynamic scheduling (out-of-order execution)
- We will talk about these things

Pinnacle of Single-Core Microprocessors

- Intel Pentium4 (2003)
 - Application: desktop/server
 - Technology: 90nm (1/100x)
 - 55M transistors (20,000x)
 - 101 mm² (10x)
 - 3.4 GHz (10,000x)
 - 1.2 Volts (1/10x)
 - 32/64-bit data (16x)
 - 22-stage pipelined datapath
 - 3 instructions per cycle (superscalar)
 - Two levels of on-chip cache
 - data-parallel vector (SIMD) instructions, hyperthreading



CIS 501 (Martin): Introduction

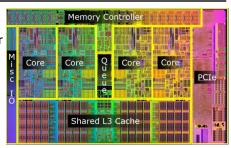
21

Revolution III: Explicit Parallelism

- Then to support explicit data & thread level parallelism
 - Hardware provides parallel resources, software specifies usage
 - Why? diminishing returns on instruction-level-parallelism
- First using (subword) vector instructions..., Intel's SSE
 - One instruction does four parallel multiplies
- ... and general support for multi-threaded programs
 - Coherent caches, hardware synchronization primitives
- Then using support for multiple concurrent threads on chip
 - First with single-core multi-threading, now with multi-core
- Graphics processing units (GPUs) are highly parallel
 - Converging with general-purpose processors (CPUs)?

Modern Multicore Processor

- Intel Core i7 (2009)
 - Application: desktop/server
 - Technology: 45nm (1/2x)
 - 774M transistors (12x)
 - 296 mm² (3x)
 - 3.2 GHz to 3.6 Ghz (~1x)
 - 0.7 to 1.4 Volts (~1x)



- 128-bit data (2x)
- 14-stage pipelined datapath (0.5x)
- 4 instructions per cycle (~1x)
- Three levels of on-chip cache
- data-parallel vector (SIMD) instructions, hyperthreading
- Four-core multicore (4x)

CIS 501 (Martin): Introduction 22

To ponder...

Is this decade's "multicore revolution" comparable to the original "microprocessor revolution"?

CIS 501 (Martin): Introduction 23 CIS 501 (Martin): Introduction 24

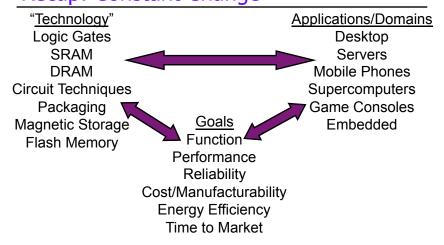
Technology Disruptions

- Classic examples:
 - The transistor
 - Microprocessor
- More recent examples:
 - · Multicore processors
 - Flash-based solid-state storage
- Near-term potentially disruptive technologies:
 - Phase-change memory (non-volatile memory)
 - Chip stacking (also called 3D die stacking)
- Disruptive "end-of-scaling"
 - "If something can't go on forever, it must stop eventually"
 - Can we continue to shrink transistors for ever?
 - Even if more transistors, not getting as energy efficient as fast

CIS 501 (Martin): Introduction

25

Recap: Constant Change



CIS 501 (Martin): Introduction 26

Managing This Mess

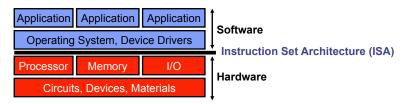
- Architect must consider all factors
 - Goals/constraints, applications, implementation technology
- Questions
 - How to deal with all of these inputs?
 - How to manage changes?
- Answers
 - Accrued institutional knowledge (stand on each other's shoulders)
 - Experience, rules of thumb
 - Discipline: clearly defined end state, keep your eyes on the ball
 - Abstraction and layering

Pervasive Idea: Abstraction and Layering

- **Abstraction**: only way of dealing with complex systems
 - Divide world into objects, each with an...
 - Interface: knobs, behaviors, knobs → behaviors
 - **Implementation**: "black box" (ignorance+apathy)
 - Only specialists deal with implementation, rest of us with interface
 - Example: car, only mechanics know how implementation works
- Layering: abstraction discipline makes life even simpler
 - Divide objects in system into layers, layer *n* objects...
 - Implemented using interfaces of layer n-1
 - Don't need to know interfaces of layer n-2 (sometimes helps)
- Inertia: a dark side of layering
 - Layer interfaces become entrenched over time ("standards")
 - Very difficult to change even if benefit is clear (example: Digital TV)
- Opacity: hard to reason about performance across layers

CIS 501 (Martin): Introduction 27 CIS 501 (Martin): Introduction 28

Abstraction, Layering, and Computers

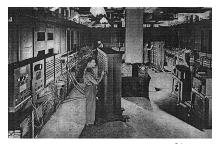


- Computer architecture
 - Definition of **ISA** to facilitate implementation of software layers
- This course mostly on **computer micro-architecture**
 - Design Processor, Memory, I/O to implement ISA
- Touch on compilers & OS (n +1), circuits (n -1) as well

Penn Legacy

CIS 501 (Martin): Introduction

- ENIAC: electronic numerical integrator and calculator
 - First operational general-purpose stored-program computer
 - Designed and built here by Eckert and Mauchly
 - Go see it (Moore building)
- · First seminars on computer design
 - Moore School Lectures, 1946
 - "Theory and Techniques for Design of Electronic Digital Computers"



29

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
 - · Writing well-tuned (fast) software requires knowledge of hardware
- Get a (design or research) hardware job
 - Intel, AMD, IBM, ARM, Motorola, Sun/Oracle, NVIDIA, Samsung
- Get a (design or research) software job
 - Best software designers understand hardware
 - Need to understand hardware to write fast software

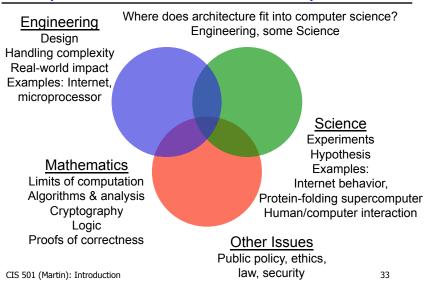
CIS 501 (Martin): Introduction 30

Course Goals

- See the "big ideas" in computer architecture
 - Pipelining, parallelism, caching, locality, abstraction, etc.
- 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)
 - Understanding quantitative data and experiments
- Get exposure to "research" and cutting edge ideas
 - Read some research literature (i.e., papers)
 - Course project
- My role: trick you into learning something

CIS 501 (Martin): Introduction 31 CIS 501 (Martin): Introduction 32

Computer Science as an Estuary



CIS501: Administrivia

- Instructor: **Prof. Milo Martin** (milom@cis.upenn.edu)
 - TAs: Christian DeLozier & Abhishek Udupa
- Lectures
 - Please do not be disruptive (I'm easily distracted as it is)
- Three different web sites
 - Course website: syllabus, schedule, lecture notes, assignments
 - http://www.cis.upenn.edu/~cis501/
 - "Piazza": announcements, questions & discussion
 - http://www.piazza.com/upenn/fall2011/cis501
 - 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 directly
 - "Blackboard": grade book, turning in some assignments
 - https://courseweb.library.upenn.edu/

Course Topics

- Revisiting "undergraduate" computer architecture topics
 - Evaluation metrics and trends
 - ISAs (instruction set architectures)
 - Datapaths and pipelining
 - Memory hierarchies & virtual memory
- Parallelism
 - · Instruction: multiple issue, dynamic scheduling, speculation
 - Data: vectors and streams
 - Thread: cache coherence and synchronization, multicore
- More fun stuff if we get to it

CIS 501 (Martin): Introduction

34

Resources

- Readings
 - "Microprocessor Architecture: From Simple Pipelines to Chip Multiprocessors" by Jean-Loup Baer
 - Penn Bookstore or Amazon (\$68) or Kindle (\$54)
 - Research papers (online)
- 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://www.cis.upenn.edu/acg/

CIS 501 (Martin): Introduction 35 CIS 501 (Martin): Introduction 36

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"
- http://www.cis.upenn.edu/~cis371/

• Significant programming experience

- · No specific language required
- Why? assignments require writing code to simulate hardware
 - Not difficult if competent programmer; extremely difficult if not

CIS 501 (Martin): Introduction 37

For Non-CIS Students...

- Registration priority is given to CIS students
- For non-CIS students
 - As the class is already extremely large...
 - · I'll only consider admitting students not in their first semester
- For non-CIS students *not* in their first semester, if you
 want to be considered, send me via email (milom@cis):
 - 1. Your name & Penn email address
 - 2. What program you're enrolled in
 - 3. A transcript of all your Penn courses with grades
 - 4. Description of prior courses on computer architecture
 - A brief description of the largest programming project you've completed (lines of code, overall complexity, language used, etc.)

The Students of CIS501

• Three different constituencies, different backgrounds

PhD students

- · More research focused
- · WPE-I PhD qualifying exam

• MSE students (CIS, EMBS, Robotics, others)

- Expand on undergraduate coursework
 - Which, unfortunately, varies widely

• BSE (undergraduate) students

- Expand on undergraduate coursework (CIS371)
- · For those considering graduate school
- Extremely difficult to tailor course for all three constituencies

CIS 501 (Martin): Introduction

38

Coursework

Homework assignments

- · Written questions and programming
- Due at beginning of class
- 2 total "grace" periods (next class period), max one per assignment
 - Hand in late, no questions asked
- · No assignments accepted after solutions posted
- Individual work

Paper reviews

- 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

Exams

- Midterm, in class, Thursday, October 27th
- Cumulative final
 - Thursday, December 15th 12-2pm
 - WPE I for PhD students

CIS 501 (Martin): Introduction 39 CIS 501 (Martin): Introduction 40

Coursework

- Mini-research project
 - Topic
 - Validate data in some paper studied in class (default)
 - Examine modest extension to paper (more ambitious)
 - Your own idea (great!)
 - Use simulation tools
 - · Homework will help you get ready
 - Groups of four (keep an eye out for potential partners)
 - Proposal + final report
 - · More detail later

CIS 501 (Martin): Introduction

41

CIS 501 (Martin): Introduction

42

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

Grading

- Tentative grade contributions:
 - Homework assignments: 20%
 - Paper reviews: 5%
 - Mini-research group project: 15%
 - Exams: 60%Midterm: 25%Final: 35%
- Typical grade distributions
 - A: 40%, B: 40%, C/D/F: 20%

Full Disclosure

- Potential sources of bias or conflict of interest
- Most of my funding governmental (your tax \$\$\$ at work)
 - National Science Foundation (NSF)
 - DARPA & ONR
- My non-governmental sources of research funding
 - NVIDIA (sub-contract of large DARPA project)
 - Inte
 - Sun/Oracle (hardware donation)
- Collaborators and colleagues
 - Intel, IBM, AMD, Oracle, Microsoft, Google, VMWare, ARM, etc.
 - (Just about every major computer hardware company)

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
 - 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
- E-mail the answers to me:
 - Text only, **no html or attachments**, please
 - Send to: cis501+reviews@cis.upenn.edu
 - The "+reviews" is important, don't leave it out
 - Carbon copy (CC) all group members
 - Include the names of all group member at the start of the e-mail
- Due: "last thing" Wednesday, Sept 14th

CIS 501 (Martin): Introduction

For Next Week...

- Read Chapter 1 for Thursday
- Read "Cramming More Components onto Integrated Circuits" by Moore, 1965
 - Group discussion responses for "last thing" Wednesday
- If you're a non-CIS student wanting to take this course
 - Send me email as discussed earlier
- See me *right now* if:
 - You're an undergraduate taking this course
 - Any other questions about prerequisites or the course

Paper Review #1 Questions

- Q1: The figure on page 2 graphs relative manufacturing cost per component against the number of components per integrated circuit. Why do the chips become less cost effective per component for both very large and very small numbers of components per chip?
- Q2: One of the potential problems which Moore raises (and dismisses) is heat. Do you agree with Moore's conclusions? Either justify or refute Moore's conclusions.
- Q3: A popular misconception of Moore's law is that it states that the speed of computers increases exponentially, however, that is not what Moore foretells in this paper. Explain what Moore's law actually says based on this paper.

CIS 501 (Martin): Introduction 46

45