Outline

- Computer Science at a Crossroads
- Computer Architecture v. Instruction Set Arch.
- How would you like your CS252?
- What Computer Architecture brings to table

Crossroads: Conventional Wisdom in Comp. Arch

- Old Conventional Wisdom: Power is free, Transistors expensive
- New Conventional Wisdom: “Power wall” Power expensive, Xtors free (Can put more on chip than can afford to turn on)
- Old CW: Sufficiently increasing Instruction Level Parallelism via compilers, innovation (Out-of-order, speculation, VLIW, …)
- New CW: “ILP wall” law of diminishing returns on more HW for ILP
- Old CW: Multiplies are slow, Memory access is fast
- New CW: “Memory wall” Memory slow, multiplies fast (200 clock cycles to DRAM memory, 4 clocks for multiply)
- Old CW: Uniprocessor performance 2X / 1.5 yrs
- New CW: Power Wall + ILP Wall + Memory Wall = Brick Wall
  - Uniprocessor performance now 2X / 5(?) yrs
  - Sea change in chip design: multiple “cores” (2X processors per chip / ~ 2 years)
  - More simpler processors are more power efficient

Crossroads: Uniprocessor Performance


- VAX : 25%/year 1978 to 1986
- RISC + x86: 52%/year 1986 to 2002
- RISC + x86: ??%/year 2002 to present
Sea Change in Chip Design

- Intel 4004 (1971): 4-bit processor, 2312 transistors, 0.4 MHz, 10 micron PMOS, 11 mm² chip

- RISC II (1983): 32-bit, 5 stage pipeline, 40,760 transistors, 3 MHz, 3 micron NMOS, 60 mm² chip

- 125 mm² chip, 0.065 micron CMOS = 2312 RISC II+FPU+Icache+Dcache
  - RISC II shrinks to ~ 0.02 mm² at 65 nm
  - Caches via DRAM or 1 transistor SRAM (www.t-ram.com)?
  - Proximity Communication via capacitive coupling at > 1 TB/s?

  (Ivan Sutherland @ Sun / Berkeley)

• Processor is the new transistor?

Déjà vu all over again?

- Multiprocessors imminent in 1970s, ‘80s, ‘90s, …
- “… today’s processors … are nearing an impasse as technologies approach the speed of light…”

- Transputer was premature
  ⇒ Custom multiprocessors strove to lead uniprocessors
  ⇒ Procrastination rewarded: 2X seq. perf. / 1.5 years

- “We are dedicating all of our future product development to multicore designs. … This is a sea change in computing”
  Paul Otellini, President, Intel (2004)

- Difference is all microprocessor companies switch to multiprocessors (AMD, Intel, IBM, Sun; all new Apples 2 CPUs)
  ⇒ Procrastination penalized: 2X sequential perf. / 5 yrs
  ⇒ Biggest programming challenge: 1 to 2 CPUs

Problems with Sea Change

- Algorithms, Programming Languages, Compilers, Operating Systems, Architectures, Libraries, … not ready to supply Thread Level Parallelism or Data Level Parallelism for 1000 CPUs / chip,

- Architectures not ready for 1000 CPUs / chip
  - Unlike Instruction Level Parallelism, cannot be solved by just by computer architects and compiler writers alone, but also cannot be solved without participation of computer architects


Outline

- Computer Science at a Crossroads
- Computer Architecture v. Instruction Set Arch.
- How would you like your CS252?
- What Computer Architecture brings to table
Instruction Set Architecture: Critical Interface

- Properties of a good abstraction
  - Lasts through many generations (portability)
  - Used in many different ways (generality)
  - Provides convenient functionality to higher levels
  - Permits an efficient implementation at lower levels

Example: MIPS

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>r0, r1, ..., r31, PC, lo, hi</td>
<td>32 x 32-bit GPRs (R0=0), 32 x 32-bit FP regs (paired DP)</td>
</tr>
</tbody>
</table>

Arithmetic logical
- Add, AddU, Sub, SubU, And, Or, Xor, Nor, SLT, SLTU, AddI, AddIU, SLTI, SLTIU, AndI, OrI, XorI, LUI, SLL, SRL, SRA, SLLV, SRLV, SRAV

Memory Access
- LB, LBU, LH, LHU, LW, LWL, LWL, SB, SH, SW, SWL, SWR

Control
- J, JAL, JR, JALR, BEq, BNE, BLEZ, BGTZ, BLTZ, BGEZ, BLTZAL, BGEZAL

32-bit instructions on word boundary

Instruction Set Architecture

"... the attributes of a [computing] system as seen by the programmer, i.e. the conceptual structure and functional behavior, as distinct from the organization of the data flows and controls the logic design, and the physical implementation."

- Amdahl, Blaauw, and Brooks, 1964

ISA vs. Computer Architecture

- Old definition of computer architecture = instruction set design
  - Other aspects of computer design called implementation
  - Insinuates implementation is uninteresting or less challenging
- Our view is computer architecture >> ISA
- Architect's job much more than instruction set design; technical hurdles today more challenging than those in instruction set design
- Since instruction set design not where action is, some conclude computer architecture (using old definition) is not where action is
  - We disagree on conclusion
  - Agree that ISA not where action is (ISA in CA:AQA 4/e appendix)
Comp. Arch. is an Integrated Approach

- What really matters is the functioning of the complete system
  - hardware, runtime system, compiler, operating system, and application
  - In networking, this is called the “End to End argument”
- Computer architecture is not just about transistors, individual instructions, or particular implementations
  - E.g., Original RISC projects replaced complex instructions with a compiler + simple instructions

Computer Architecture is Design and Analysis

Architecture is an iterative process:
- Searching the space of possible designs
- At all levels of computer systems

Creativity

Cost / Performance Analysis

Good Ideas

Bad Ideas

Mediocre Ideas

Outline

- Computer Science at a Crossroads
- Computer Architecture v. Instruction Set Arch.
- How would you like your CS252?
- What Computer Architecture brings to table
- Technology Trends

CS252: Administrivia

Instructor: Prof David Patterson
  Office: 635 Soda Hall, pattrsn@cs
  Office Hours: Tue 11 - noon or by appt.
  (Contact Cecilia Pracher; cpracher@eecs)
T. A: Archana Ganapathi, archanag@eecs
Class: M/W, 11:00 - 12:30pm  203 McLaughlin (and online)
Web page: http://www.cs/~pattsrn/courses/cs252-S06/
  Lectures available online <9:00 AM day of lecture
Wiki page: ??
First reading assignment: Chapter 1 (handout) for today, Monday
Appendix A (handout) A for Wed 1/24
Typical Class format (after week 2)

- Bring questions to class
- 1-Minute Review
- 20-Minute Lecture
- 5-Minute Administrative Matters
- 25-Minute Lecture/Discussion
- 5-Minute Break (water, stretch)
- 25-Minute Discussion based on your questions

- I will come to class early to answer questions, can stay after on Wednesdays

Quizzes

- Preparation causes you to systematize your understanding
- Reduce the pressure of taking exam
  - 2 Graded quizzes: dates TBA
  - goal: test knowledge vs. speed writing
    » 3 hrs to take 1.5-hr quiz (5:30-8:30 PM, TBA location)
  - Both quizzes can bring summary sheet
    » Transfer ideas from book to paper

- Students/Faculty meet over free pizza/drinks at La Val’s after exam

CS 252 Course Focus

Understanding the design techniques, machine structures, technology factors, evaluation methods that will determine the form of computers in 21st Century

Computer Architecture:
- Organization
- Hardware/Software Boundary

Programming Languages

Parallelism

Measurement & Evaluation

History

Compilers

Interface Design (ISA)

Technology

Applications

Operating Systems

Your CS252

- Computer architecture is at a crossroads
  – Institutionalization and renaissance
  – Power, dependability, multi CPU vs. 1 CPU performance
- Mix of lecture vs. discussion
  – Depends on how well reading is done before class
- Goal is to learn how to do good systems research
  – Learn a lot from looking at good work in the past
  – At commit point, you may chose to pursue your own new idea instead.
Research Paper Reading

- As graduate students, you are now researchers
- Most information of importance to you will be in research papers
- Ability to rapidly scan and understand research papers is key to your success

So: you will read a few papers in this course
- Quick 1 paragraph summaries and question will be due in class
- Important supplement to book.
- Will discuss papers in class

Papers will be scanned and on web page

Related Courses

- CS 152: How to build it
- CS 252: Strong Prerequisite
- CS 258: Parallel Architectures, Languages, Systems

- CS 250: Integrated Circuit Technology from a computer-organization viewpoint

Coping with CS 252

- Undergrads must have taken CS152
- Grad Students with too varied background?
  - In past, CS grad students took written prelim exams on undergraduate material in hardware, software, and theory
  - 1st 5 weeks reviewed background, helped 252, 262, 270
  - Prelims were dropped => some unprepared for CS 252?
- Grads without CS152 equivalent may have to work hard; Review: Appendix A, B, C; CS 152 home page, maybe Computer Organization and Design (COD) 3/e
  - Chapters 1 to 8 of COD if never took prerequisite
  - If took a class, be sure COD Chapters 2, 6, 7 are familiar
  - I can loan you a copy
- Will spend 2 lectures on review of Pipelining and Memory Hierarchy, and in class quiz to be sure everyone is up to speed

Grading

- 15% Homeworks (work in pairs) and reading writeups
- 35% Examinations (2 Quizzes)
- 35% Research Project (work in pairs)
  - Transition from undergrad to grad student
  - Berkeley wants you to succeed, but you need to show initiative
  - pick topic (more on this later)
  - meet 3 times with faculty to see progress
  - give oral presentation or poster session
  - written report like conference paper
  - 3 weeks work full time for 2 people
  - Opportunity to do “research in the small” to help make transition from good student to research colleague
- 15% Class Participation
New Project opportunity this semester

- FPGAs as New Research Platform
- As ~ 25 CPUs can fit in Field Programmable Gate Array (FPGA), 1000-CPU system from ~ 40 FPGAs?
  - 64-bit simple “soft core” RISC at 100MHz in 2004 (Virtex-II)
  - FPGA generations every 1.5 yrs; 2X CPUs, 2X clock rate
- HW research community does logic design (“gate shareware”) to create out-of-the-box, Massively Parallel Processor runs standard binaries of OS, apps
  - Gateware: Processors, Caches, Coherency, Ethernet Interfaces, Switches, Routers, … (IBM, Sun have donated processors)
  - E.g., 1000 processor, IBM Power binary-compatible, cache-coherent supercomputer @ 200 MHz; fast enough for research

RAMP

- Since goal is to ramp up research in multiprocessing, called Research Accelerator for Multiple Processors
  - Web page ramp.eecs.berkeley.edu

Why RAMP Good for Research?

<table>
<thead>
<tr>
<th></th>
<th>SMP</th>
<th>Cluster</th>
<th>Simulate</th>
<th>RAMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost (1000 CPUs)</td>
<td>F ($40M)</td>
<td>C ($2M)</td>
<td>A+ ($0M)</td>
<td>A ($0.1M)</td>
</tr>
<tr>
<td>Cost of ownership</td>
<td>A</td>
<td>D</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>Scalability</td>
<td>C</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>Power/Space (kilowatts, racks)</td>
<td>D (120 kw, 12 racks)</td>
<td>D (120 kw, 12 racks)</td>
<td>A+ (.1 kw, 0.1 racks)</td>
<td>A (1.5 kw, 0.3 racks)</td>
</tr>
<tr>
<td>Community</td>
<td>D</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>Observability</td>
<td>D</td>
<td>C</td>
<td>A+</td>
<td>A+</td>
</tr>
<tr>
<td>Reproducibility</td>
<td>B</td>
<td>D</td>
<td>A+</td>
<td>A+</td>
</tr>
<tr>
<td>Flexibility</td>
<td>D</td>
<td>C</td>
<td>A+</td>
<td>A+</td>
</tr>
<tr>
<td>Credibility</td>
<td>A+</td>
<td>A+</td>
<td>F</td>
<td>A</td>
</tr>
<tr>
<td>Perform. (clock)</td>
<td>A (2 GHz)</td>
<td>A (3 GHz)</td>
<td>F (0 GHz)</td>
<td>C (0.2 GHz)</td>
</tr>
<tr>
<td>GPA</td>
<td>1/21/2006</td>
<td>CS252-a06, Lec 01-intro</td>
<td>B</td>
<td>A+</td>
</tr>
</tbody>
</table>

RAMP 1 Hardware

- Completed Dec. 2004 (14x17 inch 22-layer PCB)
- Module:
  - FPGAs, memory, 10GigE conn.
  - Compact Flash
  - Administration/maintenance ports:
    - 10/100 Enet
    - HDMI/DVI
    - USB
  - ~4K/module w/o FPGAs or DRAM

Called “BEE2” for Berkeley Emulation Engine 2
Multiple Module RAMP 1 Systems

- 8 compute modules (plus power supplies) in 8U rack mount chassis
  - 500-1000 emulated processors
- Many topologies possible
- 2U single module tray for developers
- Disk storage: disk emulator + Network Attached Storage

Vision: Multiprocessing Watering Hole

- RAMP attracts many communities to shared artifact
  ⇒ Cross-disciplinary interactions
  ⇒ Accelerate innovation in multiprocessing
- RAMP as next Standard Research Platform?
  (e.g., VAX/BSD Unix in 1980s, x86/Linux in 1990s)

Supporters (wrote letters to NSF) & Participants

- Gordon Bell (Microsoft)
- Ivo Bolsens (Xilinx CTO)
- Norm Jouppi (HP Labs)
- Bill Kramer (NERSC/LBL)
- Craig Mundie (MS CTO)
- G. Papadopoulos (Sun CTO)
- Justin Rattner (Intel CTO)
- Ivan Sutherland (Sun Fellow)
- Chuck Thacker (Microsoft)
- Kees Vissers (Xilinx)
- Doug Burger (Texas)
- Bill Dally (Stanford)
- Carl Ebeling (Washington)
- Susan Eggers (Washington)
- Steve Keckler (Texas)
- Greg Morrisett (Harvard)
- Scott Shenker (Berkeley)
- Ion Stoica (Berkeley)
- Kathy Yelick (Berkeley)

RAMP Participants: Arvind (MIT), Krste Asanovic (MIT), Derek Chiou (Texas), James Hoe (CMU), Christos Kozyrakis (Stanford), Shih-Lien Lu (Intel), Mark Oskin (Washington), David Patterson (Berkeley), Jan Rabaey (Berkeley), and John Wawrzynek (Berkeley)

RAMP Summary

- RAMP as system-level time machine: preview computers of future to accelerate HW/SW generations
  - Trace anything, Reproduce everything, Tape out every day
  - FTP new supercomputer overnight and boot in morning
  - Clone to check results (as fast in Berkeley as in Boston?)
  - Emulate Massive Multiprocessor, Data Center, or Distributed Computer
- Carpe Diem
  - Systems researchers (HW & SW) need the capability
  - FPGA technology is ready today, and getting better every year
  - Stand on shoulders vs. toes: standardize on multi-year Berkeley effort on FPGA platform Berkeley Emulation Engine 2 (BEE2)
  - Architecture researchers get opportunity to immediately aid colleagues via gateware (as SW researchers have done in past)
  - See ramp.eecs.berkeley.edu

- Vision “Multiprocessor Research Watering Hole” accelerate research in multiprocessing via standard research platform
  ⇒ hasten sea change from sequential to parallel computing
RAMP projects for CS 252

- Design a of guest timing accounting strategy
  - Want to be able specify performance parameters (clock rate, memory latency, network latency, …)
  - Host must accurately account for guest clock cycles
  - Don’t want to slow down host execution time very much

- Build a disk emulator for use in RAMP
  - Imitates disk, accesses network attached storage for data
  - Modeled after guest VM/driver VM from Xen VM?

- Build a cluster using components from opencores.org on BEE2
  - Open source hardware consortium

- Build an emulator of an “Internet in a Box”
  - (Emulab/Planetlab in a box is closer to reality)

Other projects

- Recreate results from research paper to see
  - If they are reproducible
  - If they still hold

- Performance evaluation of Niagara, new 8 core, 4 threads per core chip from Sun

- Propose your own research project that is related to computer architecture

Outline

- Computer Science at a Crossroads
- Computer Architecture v. Instruction Set Arch.
- How would you like your CS252?
- What Computer Architecture brings to table

What Computer Architecture brings to Table

- Other fields often borrow ideas from architecture
- Quantitative Principles of Design
  1. Take Advantage of Parallelism
  2. Principle of Locality
  3. Focus on the Common Case
  4. Amdahl’s Law
  5. The Processor Performance Equation
- Careful, quantitative comparisons
  - Define, quantity, and summarize relative performance
  - Define and quantify relative cost
  - Define and quantify dependability
  - Define and quantify power
- Culture of anticipating and exploiting advances in technology
- Culture of well-defined interfaces that are carefully implemented and thoroughly checked
1) Taking Advantage of Parallelism

- Increasing throughput of server computer via multiple processors or multiple disks
- Detailed HW design
  - Carry lookahead adders uses parallelism to speed up computing sums from linear to logarithmic in number of bits per operand
  - Multiple memory banks searched in parallel in set-associative caches
- Pipelining: overlap instruction execution to reduce the total time to complete an instruction sequence.
  - Not every instruction depends on immediate predecessor ⇒ executing instructions completely/partially in parallel possible
  - Classic 5-stage pipeline:
    1) Instruction Fetch (Ifetch),
    2) Register Read (Reg),
    3) Execute (ALU),
    4) Data Memory Access (Dmem),
    5) Register Write (Reg)

2) The Principle of Locality

- The Principle of Locality:
  - Program access a relatively small portion of the address space at any instant of time.
- Two Different Types of Locality:
  - Temporal Locality (Locality in Time): If an item is referenced, it will tend to be referenced again soon (e.g., loops, reuse)
  - Spatial Locality (Locality in Space): If an item is referenced, items whose addresses are close by tend to be referenced soon (e.g., straight-line code, array access)
- Last 30 years, HW relied on locality for memory perf.
Levels of the Memory Hierarchy

- **CPU Registers**
  - 100s Bytes
  - 300-500 ps (0.3-0.5 ns)

- **L1 and L2 Cache**
  - 10s-100s K Bytes
  - ~1 ns - ~10 ns
  - $1000s/ GByte

- **Main Memory**
  - G Bytes
  - 80ns-200ns
  - ~$100/ GByte

- **Disk**
  - 10s T Bytes, 10 ms (10,000,000 ns)
  - ~$1 / GByte

**3) Focus on the Common Case**

- **Common sense guides computer design**
  - Since its engineering, common sense is valuable

- **In making a design trade-off, favor the frequent case over the infrequent case**
  - E.g., Instruction fetch and decode unit used more frequently than multiplier, so optimize it 1st
  - E.g., If database server has 50 disks / processor, storage dependability dominates system dependability, so optimize it 1st

- **Frequent case is often simpler and can be done faster than the infrequent case**
  - E.g., overflow is rare when adding 2 numbers, so improve performance by optimizing more common case of no overflow
  - May slow down overflow, but overall performance improved by optimizing for the normal case

- **What is frequent case and how much performance improved by making case faster => Amdahl's Law**

**4) Amdahl's Law**

\[
\text{ExTime}_{\text{new}} = \text{ExTime}_{\text{old}} \times \left[ (1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}} \right]
\]

\[
\text{Speedup}_{\text{overall}} = \frac{\text{ExTime}_{\text{old}}}{\text{ExTime}_{\text{new}}} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}
\]

**Amdahl’s Law example**

- **New CPU 10X faster**
- I/O bound server, so 60% time waiting for I/O

\[
\text{Speedup}_{\text{overall}} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}} = \frac{1}{(1 - 0.4) + \frac{0.4}{10}} = \frac{1}{0.64} = 1.56
\]

- Apparently, its human nature to be attracted by 10X faster, vs. keeping in perspective its just 1.6X faster
5) Processor performance equation

CPU time = Seconds = Instructions x Cycles x Seconds

<table>
<thead>
<tr>
<th></th>
<th>Program</th>
<th>Compiler</th>
<th>Inst. Set.</th>
<th>Organization</th>
<th>Technology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inst Count</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>CPI</td>
<td></td>
<td>(X)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clock Rate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

What’s a Clock Cycle?

- Old days: 10 levels of gates
- Today: determined by numerous time-of-flight issues + gate delays
  - clock propagation, wire lengths, drivers

And in conclusion …

- Computer Architecture >> instruction sets
- Computer Architecture skill sets are different
  - 5 Quantitative principles of design
  - Quantitative approach to design
  - Solid interfaces that really work
  - Technology tracking and anticipation
- CS 252 to learn new skills, transition to research
- Computer Science at the crossroads from sequential to parallel computing
  - Salvation requires innovation in many fields, including computer architecture
- RAMP is interesting and timely CS 252 project opportunity given CS is at the crossroads
- Read Chapter 1, then Appendix A, record bugs!