Computer Science 146
Computer Architecture

Fall 2019

Harvard University

Instructor: Prof. David Brooks
dbrooks@eecs.harvard.edu

Course Outline

- Administrative details
- Why take CS146? What you will learn?
- What is Computer Architecture?

Instructor

- Instructor: David Brooks (dbrooks@eecs.harvard.edu)
  - Office Hours: M/W 11AM-Noon, MD141, stop by/email for other times
- Teaching Fellow: Iulian Brumar (ibrumar@g.harvard.edu)
  - Office Hours: T/Th , MD308, Tu 4-5pm or by appointment
  - Occasional "on-demand" TF teaching sessions… Exam review, homework, etc.
- Course Webpage
  - http://www.eecs.harvard.edu/cs146-246

Resources

- Text: "Computer Architecture: A Quantitative Approach," **Third Edition**, Hennessy and Patterson
- Some key research papers (will be available on the web or paper copies)
- SimpleScalar toolset
  - Widely used architectural simulator
  - SPEC2000 benchmarks
  - Will be used for some HW/Projects

Prerequisites

- CS141 (Computing Hardware) or similar
  - Logic Design (computer arithmetic)
  - Basic ISA (what is a RISC instruction)
  - Pipelining (control/data hazards, forwarding)
  - Will review the above during the first couple of weeks
- C Programming, UNIX
- Compilers, OS, Circuits/VLSI background is a plus, not needed

Course Expectations

- Homeworks (5-6 over the semester)
  - Will be a mix of paper/pencil and SimpleScalar based
- Midterm/Final
- Course project
  - Several possible ideas will be given and you may come up with your own
  - Project Goals:
    - Give you some design experience
    - Give you a flavor of architecture research
  - Small groups a possibility

Grading

- Grade Formula (+/- 5%)
  - Homework – 20%
  - Midterm – 25%
  - Final – 30%
  - Project – 25%
- Will follow the "standard" CS late policy
  - 5 late days per semester per student

Why take CS146?

- How does computing hardware work?
- Where is computing hardware going in the future?
  - And learn how to contribute to this future…
- How does this impact system software and applications?
  - Essential to understand OS/compilers/PL
  - For everyone else, it can help you write better code!
- How are future technologies going to impact computing systems?

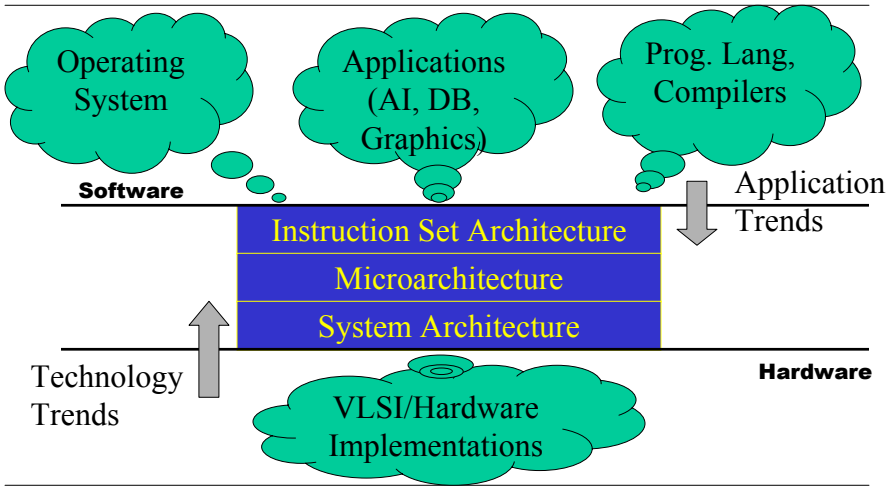
Topics of Study

- Focus on what modern computer architects worry about (both academia and industry)
- Get through the basics of modern processor design
- Understand the interfaces between architecture and system software (compilers, OS)
- System architecture and I/O (disks, memory, multiprocessors)
- Look at technology trends, recent research ideas, and the future of computing hardware

Estimated Schedule

- Introduction and Performance Metrics (1 week)
- ISA/Basic Pipelining Review (1 week)
- Hardware ILP (2 weeks)
- Software ILP (1 week)
- Caches/Memory (1.5 weeks)
- Modern Processor Case Studies (1.5 weeks)
- Multiprocessors/Multithreading (1-2 weeks)
- Input/Output and Interconnects (1 week)
- Research Trends (1 week)
- Technology Trends impact on architecture (1 week)

What is Computer Architecture?



Application Areas

- General-Purpose Laptop/Desktop
  - Productivity, interactive graphics, video, audio
  - Optimize price-performance
  - Examples: Intel Pentium 4, AMD Athlon XP
- Embedded Computers
  - PDAs, cell-phones, sensors => Price, Energy efficiency
  - Examples: Intel XScale, StrongARM (SA-110)
  - Game Machines, Network uPs => Price-Performance
  - Examples: Sony Emotion Engine, IBM 750FX

Application Areas

- Commercial Servers
  - Database, transaction processing, search engines
  - Performance, Availability, Scalability
  - Server downtime could cost a brokerage company more than $6M/hour
  - Examples: Sun Fire 15K, IBM p690, Google Cluster
- Scientific Applications
  - Protein Folding, Weather Modeling, CompBio, Defense
  - Floating-point arithmetic, Huge Memories
  - Examples: IBM DeepBlue, BlueGene, Cray T3E, etc.

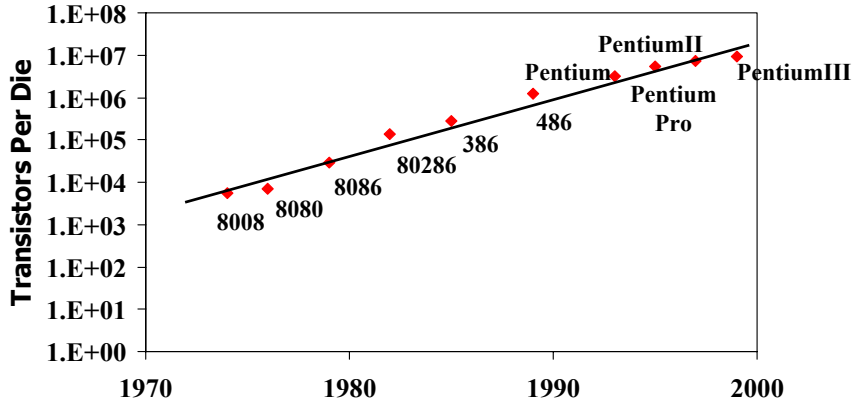
Software Trends

- No longer just executing C/FORTRAN code
- Object Oriented Programming
- Java
- Architectural features to assist security
- Middleware
  - Layer(s) between client and server applications
  - Hides complexity of client/server communications

Moore's Law

- Every 18-24 months
  - Feature sizes shrink by 0.7x
  - Number of transistors per die increases by 2x
  - Speed of transistors increases by 1.4x
- But we are starting to hit some roadblocks…
- Also, what to do with all of these transistors???

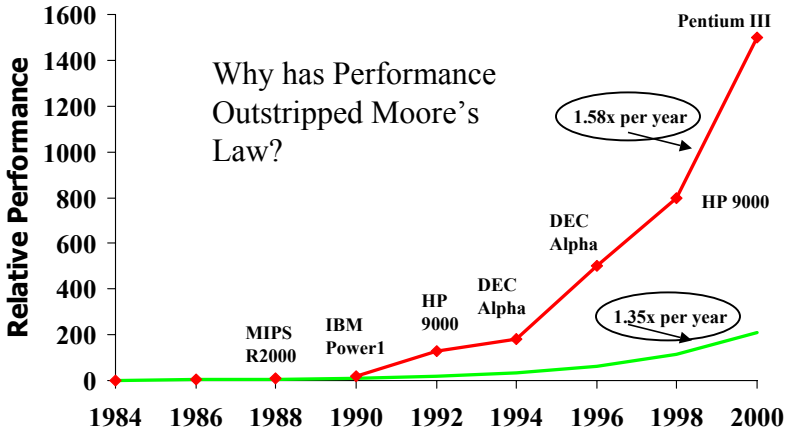
Moore's Law (Density)



How have we used these transistors?

- More functionality on one chip
  - Early 1980s – 32-bit microprocessors
  - Late 1980s – On Chip Level 1 Caches
  - Early/Mid 1990s – 64-bit microprocessors, superscalar (ILP)
  - Late 1990s – On Chip Level 2 Caches
  - Early 2000s – Chip Multiprocessors, On Chip Level 3 Caches
- What is next?
  - How much more cache can we put on a chip? (IRAM)
  - How many more cores can we put on a chip? (Piranha, etc)
  - What else can we put on chips?

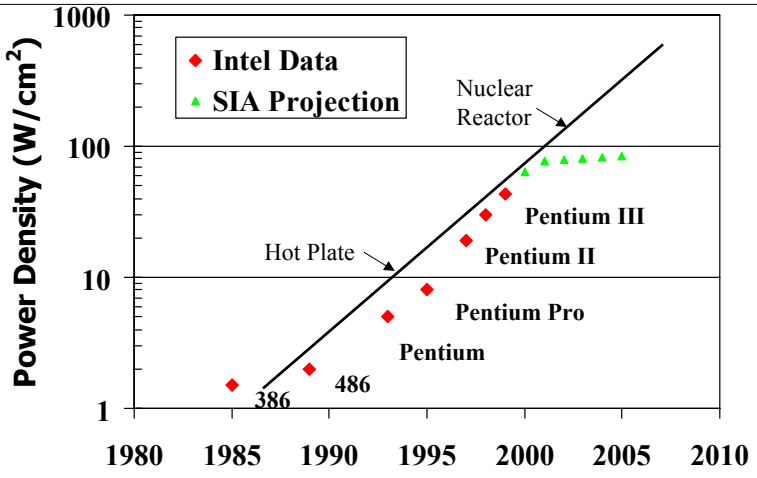
Moore's Law (Performance)



Performance vs. Technology Scaling

- Architectural Innovations
  - Massive pipelining (good and bad!)
  - Huge caches
  - Branch Prediction, Register Renaming, OOO-issue/execution, Speculation (hardware/software versions of all of these)
- Circuit Innovations
  - New logic circuit families (dynamic logic)
  - Better CAD tools
  - Advanced computer arithmetic

Moore's Law (Power)



Dealing with Complexity: Abstractions

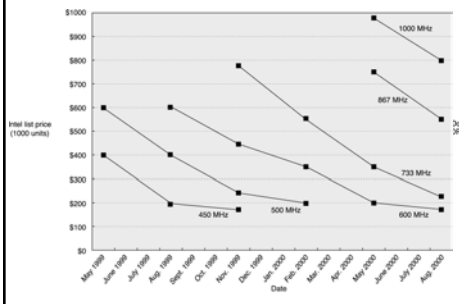
- As an architect, our main job is to deal with tradeoffs
  - Performance, Power, Die Size, Complexity, Applications Support, Functionality, Compatibility, Reliability, etc.
- Technology trends, applications… How do we deal with all of this to make real tradeoffs?
- Abstractions allow this to happen
- Focus is on metrics of these abstractions
  - Performance, Cost, Availability, Power

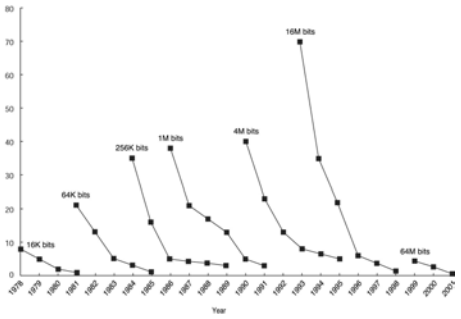
Metrics: Performance

- Reduce Response/Execution Time
  - Time between start and completion of an event
- Increase Throughput
  - Total amount of work in a given amount of time
- Execution Time is reciprocal of performance
- "X is N times faster than Y"
- N=Execution Time_Y/Execution Time_X
- Wall-Clock Time, CPU time (no I/O)
- More next Lecture

Metrics: Cost

- Again Moore's Law at work…





Metrics: Availability

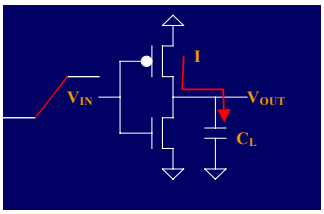
- Availability: Fraction of Time a system is available
- For servers, may be as important as performance
- Mean Time Between Failures (MTBF)
  - Period that a system is usable
  - Typically 500,000 hours for a PC Harddrive
- Mean Time to Repair (MTRR)
  - Recovery time from a failure
  - Should approach 0 for a big server (redundancy)

Metrics: Power Dissipation

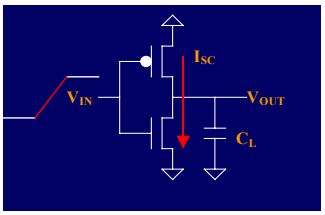
- We all understand why energy is important for embedded CPUs…
- High-end Server: ~130-170W
- High-end Desktop: ~70-100W
- High-end Laptop: ~30W
- Battery-Optimized Laptop: ~3-10W
- Embedded CPUs: ~.5-1W
- DSP: ~100mW
- Microcontrollers: ~10mW

Metrics: Power Dissipation

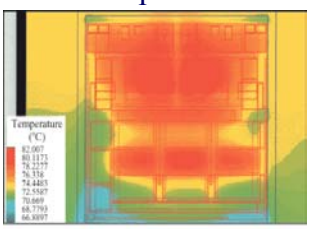
Capacitive (Dynamic) Power



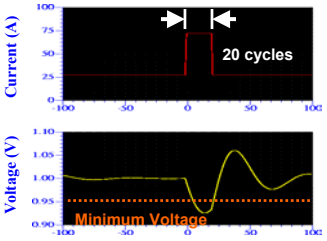
Static (Leakage) Power



Temperature



Di/Dt (Vdd/Gnd Bounce)



Summary

- Architecture is the “glue” between system software/applications and VLSI implementations
- Need to create abstractions to deal with complexity
- This course complements well with
  - CS148 (Design of VLSI Circuits and Systems)
  - CS161 (Operating Systems)
  - CS153 (Compilers)