U. Wisconsin CS/ECE 752 Advanced Computer Architecture I

Prof. David A. Wood

Unit 0: Introduction

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

Slides enhanced by Milo Martin, Mark Hill, and David Wood with sources that included Profs. Asanovic, Falsafi, Hoe, Lipasti, Shen, Smith, Sohi, Vijaykumar, and 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…

What is ~~Computer~~ Architecture?

The role of a building architect:

Construction

Materials
Steel
Concrete
Brick
Wood
Glass

Design

Plans

Buildings
Houses
Offices
Apartments
Stadiums
Museums

Goals
Function
Cost
Safety
Ease of Construction
Energy Efficiency
Fast Build Time
Aesthetics

What is Computer Architecture?

The role of a *computer* architect:

"Technology"
Logic Gates
SRAM
DRAM
Circuit Techniques
Packaging
Magnetic Storage
Flash Memory

Design

Plans

Manufacturing

Computer
PCs
Servers
PDAs
Mobile Phones
Supercomputers
Game Consoles
Embedded

Goals
Function
Performance
Reliability
Cost/Manufacturability
Energy Efficiency
Time to Market

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

Design Goals

- Functional
  - Needs to be correct
  - 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 probe vs PC reliability
- High performance
  - “Fast” is only meaningful in the context of a set of important tasks
  - Not just “Gigahertz”
  - Impossible goal: fastest possible design for all programs

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…)

- Low power
  - Energy in (battery life, cost of electricity)
  - Energy out (cooling and related costs)
  - Static vs dynamic power, sleep modes, peak vs average
  - Cyclic problem, very much a problem today

- Challenge: balancing the relative importance of these goals
  - And the balance is constantly changing

Constant Change

“Technology”

Logic Gates
SRAM
DRAM
Circuit Techniques
Packaging
Magnetic Storage
Flash Memory

Applications/Domains

PCs
Servers
PDAs
Mobile Phones
Supercomputers
Game Consoles
Embedded

Goals

Function
Performance
Reliability
Cost/Manufacturability
Energy Efficiency
Time to Market

- Absolute improvement, **different rates of change**, **cyclic**
- Better computers help design the next generation (CAD)

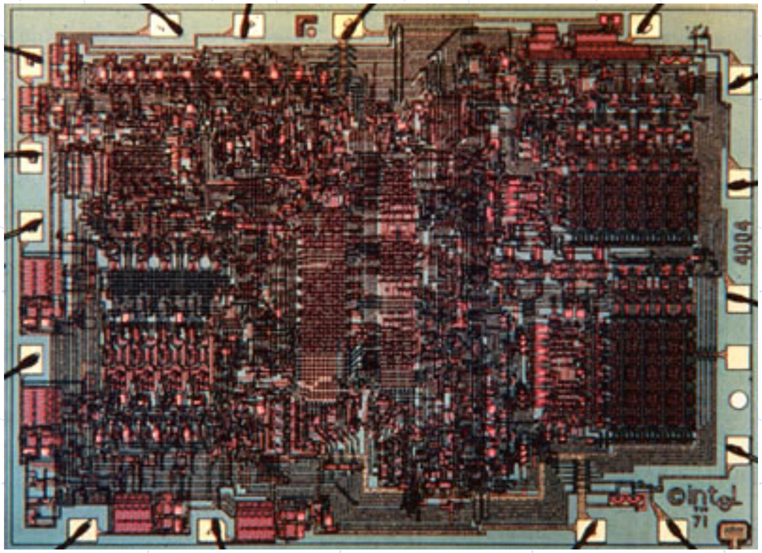
Rapid Change

	1971–1980	1981–1990	1991–2000	2008	2015
Transistors (M)	0.01–0.1	0.1–1	1–100	300-1000	10000?
Clock (MHz)	0.2–2	2–20	20–1000	3500	10000?
MIPS	<0.2	0.2–20	20–2000	16000	100000?

- Exciting: perhaps the fastest moving field … ever
  - Processors vs. cars
    - 1985: processors = 1 MIPS, cars = 60 MPH
    - 2000: processors = 500 MIPS, cars = 30,000 MPH?
- Another exciting field? Genomics
  - Guess what: many genomics advances are computational

First Microprocessor

- Connect a few transistors together to make…



- Intel 4004
  - 1971 (first microprocessor)
  - 4-bit data
  - 2300 transistors
  - 10 μm PMOS
  - 108 KHz
  - 12 V
  - 13 mm^2

Not-so-Recent Microprocessor

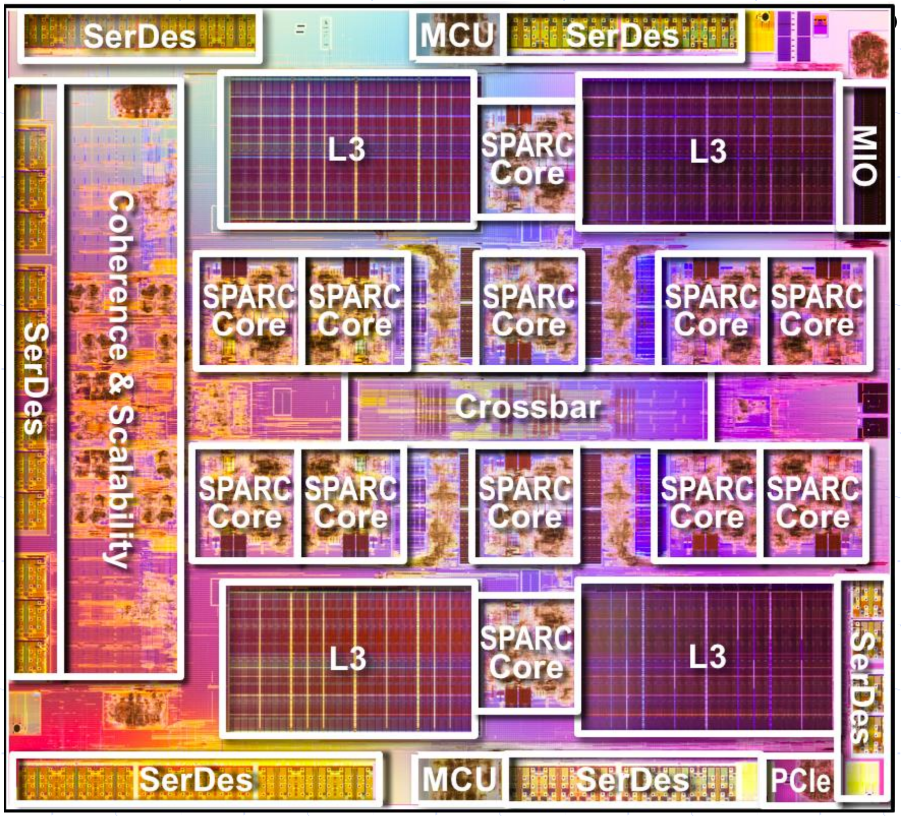
- Or a few more to form…

- Intel Pentium4 + HT
  - 2003
  - 32/64-bit data
  - 55M transistors
  - 0.90 μm CMOS
  - 3.4 GHz
  - 1.2 V
  - 101 mm^2

By end of course, this will make sense!

- Pentium 4 specifications: what do each of these mean?
  - Technology
    - 55M transistors, 0.90 μm CMOS, 101 mm^2, 3.4 GHz, 1.2 V
  - Performance
    - 1705 SPECint, 2200 SPECfp
  - ISA
    - X86+MMX/SSE/SSE2/SSE3 (X86 translated to RISC uops inside)
  - Memory hierarchy
    - 64KB 2-way insn trace cache, 16KB D$, 512KB–2MB L2
    - MESI-protocol coherence controller, processor consistency
  - Pipeline
    - 22-stages, dynamic scheduling/load speculation, MIPS renaming
    - 1K-entry BTB, 8Kb hybrid direction predictor, 16-entry RAS
    - 2-way hyper-threading

A recently announced processor



Oracle M6

- 12 Cores, 96 threads
- 1-8 threads per core
- 16 KB L1I$ and L1D$/core
- 128 KB L2$/core
- 48 MB L3$
- 1 TB memory per socket
- Upto 8 sockets glueless MP
- 4.1 Tbps link bandwidth
- 28 nm
- 3.6 GHz
- 643 mm^2
- 4.27 billion transistors

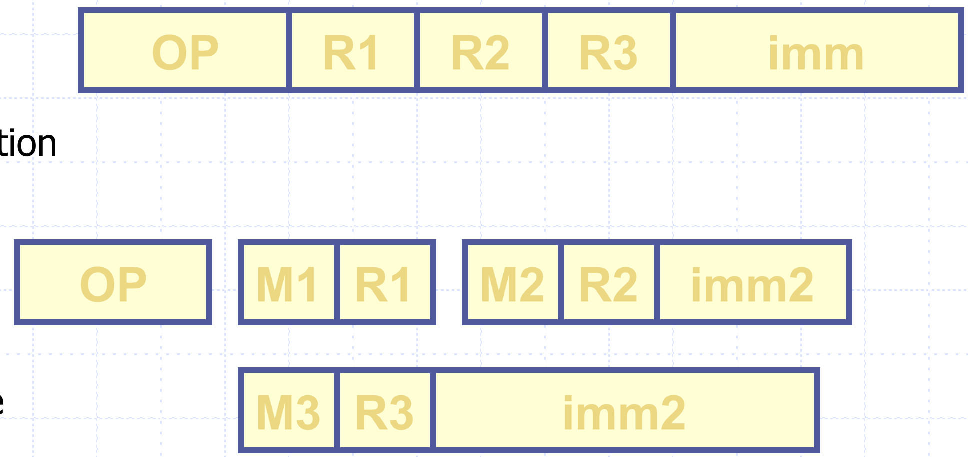
Layers of abstraction

- Architects need to understand computers at many levels
  - Instruction Set Architecture
  - Microarchitecture
  - Systems Architecture
  - Technology
  - Applications

- Good architects are "Jacks of most trades"

Instruction Set Architecture

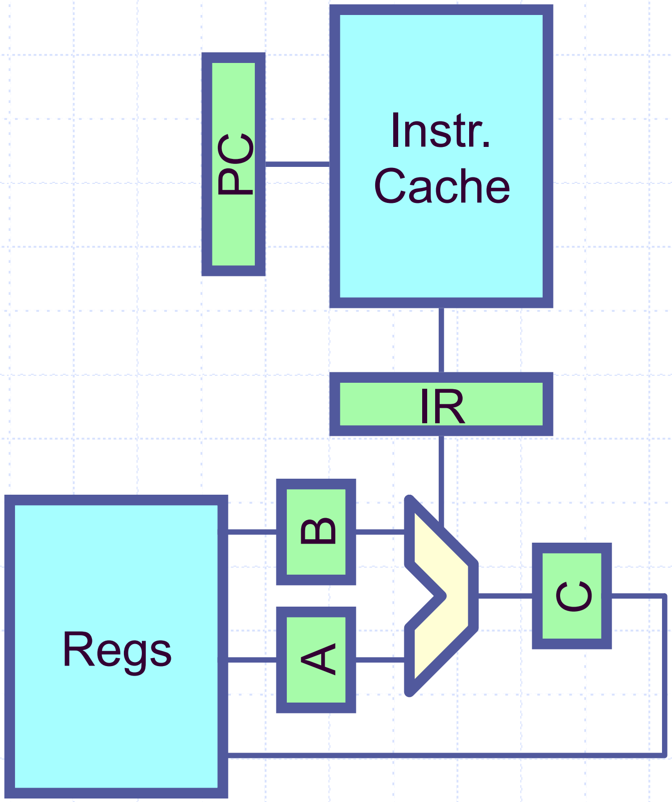
- Hardware/Software interface
  - Software impact
    - support OS functions
      - restartable instructions
      - memory relocation and protection
    - a good compiler target
      - simple
      - orthogonal
    - Dense
      - Improve memory performance



  - Hardware impact
    - admits efficient implementation
      - across generations
    - admits parallel implementation
      - no ‘serial’ bottlenecks
  - Abstraction without interpretation

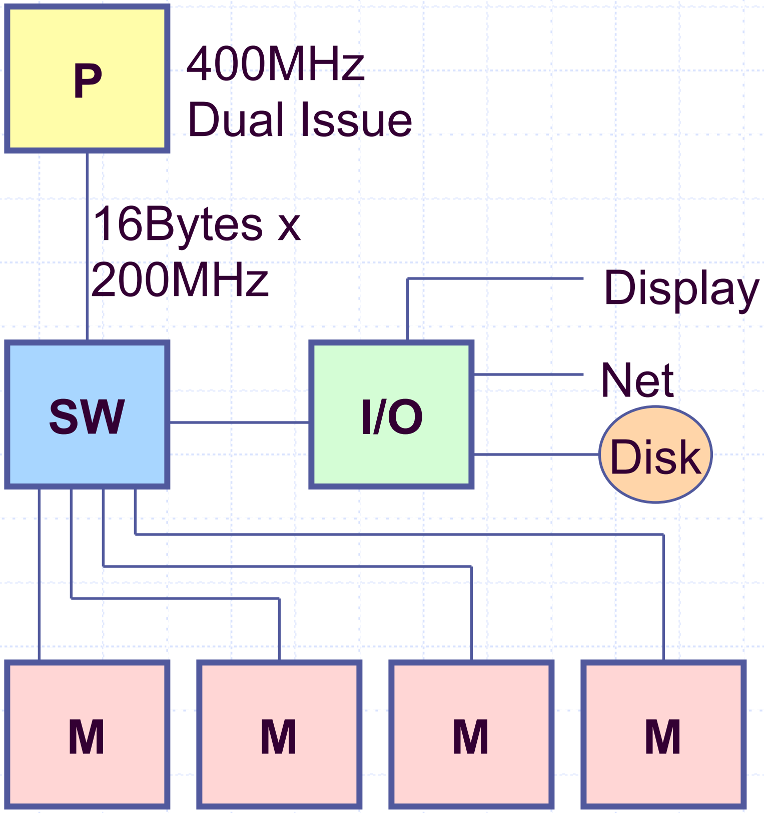
Microarchitecture

- Register-transfer-level (RTL) design
- Implement instruction set
- Exploit capabilities of technology
  - locality and concurrency
- Iterative process
  - generate proposed architecture
  - estimate cost
  - measure performance
- Still emphasis is on overcoming sequential nature of programs
  - deep pipelining
  - multiple issue
  - dynamic scheduling
  - branch prediction/speculation



System-Level Design

- Design at the level of processors, memories, and interconnect.
- More important to application performance, cost and power than CPU design
- Feeds and speeds
  - constrained by IC pin count, module pin count, and signaling rates
- System balance
  - for a particular application
- Driven by
  - performance/cost goals
  - available components (cost/perf)
  - technology constraints



Large-system example

- Google Project 02: Oregon/Washington border
- Cheap electricity: Columbia river
- Cheap cooling: Columbia river
- Cheap b/w: surplus optic network from tech-boom era
- Google, Yahoo, and Microsoft

Technology Trends

- Processor (SRAM)
  - Density: ~30%, Speed: ~20%
- Memory (DRAM)
  - Density: ~60%, Speed: ~4%
- Disk
  - Density: ~60%, Speed: ~10%

- **Changing quickly and with respect to each other!!**
  - Fundamentally changes design
  - Different tradeoffs

- Exciting: constant re-evaluation and re-design

Shaping Force: Applications/Domains

- Another shaping force: **applications**
  - 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
  - Need: data movement, high memory + I/O bandwidth
  - Examples: Sun Enterprise Server, AMD Opteron, Intel Xeon, IBM Power 7

More Recent Applications/Domains

- **Desktop**: home office, multimedia, games
  - Need: integer, memory bandwidth, integrated graphics/network
  - Examples: Intel Core i7, AMD Phenom
- **Mobile**: laptops
  - Need: **low power**, integer performance, integrated wireless
  - Examples: Intel i5, AMD APUs
- **Embedded**: PDAs, cell phones, automobiles, door knobs
  - Need: low power, **low cost**, integrated DSP?
  - Examples: Intel Atom and ARM SOCs
- **Sensors**: disposable "smart dust"
  - Need: extremely low power, extremely low cost

Application Specific Designs

- This course is mostly about **general-purpose CPUs**
  - CPU that can do anything, specifically run a full OS
  - E.g., Intel Core 2, IBM Power6, AMD Opteron, Intel Itanium

- Large, profitable segment of **application-specific CPUs**
  - Implement some critical domain-specific functionality in hardware
    - Graphics engines, physics engines

  \+ Much more effective (performance, power, cost) than software

      + General rule: hardware is better than software
  - Examples: may or may not get to
    - Emotion engine (PS2/PSX), Samsung phone, HDTV, iPod…

Why Study Computer Architecture?

- **Understand where computers are going**
  - Future capabilities drive the computing world
  - Forced to think 5+ years in the future
- **Exposure to high-level design**
  - Less about “design” than “what to design”
  - Engineering, science, art
  - Architects paint with broad strokes
  - The best architects understand all the levels
    - Devices, circuits, architecture, compilers, applications
- **Understand hardware for software tuning**
- **Real-world impact**
  - no computer architecture → no computers!
- **Get a job** (design or research)

Course Prerequisites

- CS/ECE 552 Basic Architecture
  - Logic: gates, boolean functions, latches, memories
  - Datapath: ALU, register file, memory interface, muxes
  - Control: single-cycle control, micro-code
  - Caches & pipelining (will go into these in more detail here)
  - Some familiarity with assembly language
  - Hennessy & Patterson's "Computer Organization and Design"
- Also
  - CS 537 Operating Systems (processes, threads, & virtual memory)
  - C/Unix programming

Some Course Goals

- **Exposure to the "big ideas" in computer architecture**
- Exposure to examples of good (and some bad) engineering
- Understanding computer performance and metrics
  - Empirical evaluation
  - Understanding quantitative data and experiments
- "Research" exposure
  - Read research literature (i.e., papers)
  - Course project
  - Cutting edge proposals
- Non-goals: "how computers work", detailed design
- Let's look at course home page…