Computer Architecture: Fundamentals, Tradeoffs, Challenges

Chapter 1
Introduction, Focus, Overview

Yale Patt
The University of Texas at Austin

Austin, January 27, 2025

Some Topics

- ***Lecture 1. Introduction, Focus, Overview***
- ***Parallelism (Single thread, multithread)***
- ***Mechanisms (Run-time, Compile time)***
- ***Meganonsense***
- ***Research thrusts, moving forward***
- ***Paradigms***
- ***The Microprocessor of 2035***
- ***Random Thoughts***

Lecture 1. Introduction, Focus, Overview

Outline

- *The Computer System*
- *A science of tradeoffs*
- *Several Perspectives (Note the transformation hierarchy*
- *Architecture vs Microarchitecture*
- *Performance – A low-level perspective (inst, data, fn_unit)*
- *A few more words on data supply*
- *Performance – A high-level perspective (George Michael’s triangle)*
- *The Role of Numbers*
- *Speculation*

Outline Continued)

- ***Moore’s Law***
- ***The von Neumann Model***
- ***Intro to Nonsense: Is hardware parallel or sequential***
- ***Do it in hardware or do it in software***
- ***Design Principles***
- ***Design Points***
- ***Design Methodology***
- ***Role of the Architect***
- ***Thinking outside the box***
- ***Finally, a few questions***

The Computer System

- ***The Processor***
  - ***Manages the computer system, processes the instructions***
  - ***Accesses information (loads/stores) from memory and I/O***
  - ***Computes (operate instructions) with functional units***
  - ***Maintains instruction flow (control instructions)***
- ***The Memory System***
  - ***Multiple levels of cache***
  - ***Main memory***
- ***Input/output devices***
  - ***Some very simple (keyboard, monitor***
  - ***Some more complex (disk)***
  - ***Some are like a processor in their own right***
- ***Access Mechanisms***
  - ***CAM (cache), Memory (RAM), Disk (DASD), Tape (Sequential)***

Computer Architecture: A Science of Tradeoffs

- ***ISA***
- ***Microarchitecture***
- ***System***

Computer Architecture: A Science of Tradeoffs

- *ISA*
- *Microarchitecture*
- *System*

ISA Tradeoffs (with examples)

- *Dynamic static interface (The semantic gap)*
- *Some example instructions*
  - *EDITPC, INDEX, AOBLEQ, LDCTX, CALL, FF,*
  - *INSQUE/REMQUE, Triads, CHMD, PROBE*
- *Security (at ISA: capability based ISAs)*
  - *I432, IBM System 38, Data General Fountainhead*
- *Predication*
  - *X86: CMOV*
  - *ARM: inst[31:28]*
  - *THUMB: IT block*
- *Support for multiple ISAs*
  - *ARM: T bit in the status register*
  - *VAX: Compatibility mode bit in the PSL*

ISA Tradeoffs (continued)

- ***Register set and size***
  - ***Many machine have 32 32-bit registers***
  - ***x86 now has 512-bit registers***
  - ***Itanium has one-bit predicate registers***
- ***Condition codes vs using a general purpose register***
  - ***MIPS, CDC6600***
- ***Rich instruction set vs Lean instruction set***
  - ***Hewlett Packard's RISC: HPPA has 140 instructions***
  - ***Orthogonal to RISC vs CISC***
- ***Load/Store vs Operate in the same instruction***
  - ***LC-3b, x86 are load/store***
  - ***x86 is not load/store***

ISA Tradeoffs (continued)

- ***Memory address space (keeps growing!)***
- ***Memory addressability***
  - *Most memories: byte addressable (Data processing)*
  - *Scientific machines: 64 bits (size of normal fl.pt. operands)*
  - *Burroughs 1700: one bit (virtual machines)*
- ***Page Size (4KB vs more than one)***
  - *Wasted space*
  - *Longer access time*
  - *Too many PTEs*
- ***I/O architecture***
  - *Most today use memory mapped I/O*
  - *Old days, special I/O instructions*
  - *x86 still has both*

ISA Tradeoffs (continued)

- ***Compile time vs run time***
  - ***MIPS initially had NO hardware interlocks***
- ***Instruction format***
  - ***Most have fixed length, uniform decode***
  - ***x86 has variable length, with prefixes***
  - ***i432 had different bit size opcode***
- ***Word length***
  - ***VAX: 32 bits***
  - ***x86: initially 16 bits, then 32 bits, today 64 bits***
  - ***CRAY 1: 64 bits***
  - ***DEC System 20: 36 bits (LISP car, cdr for AI processing)***

ISA Tradeoffs (continued)

- ***Help for the programmer vs help for the uarchitect***
  - ***Who gets the cushy job?***
  - ***Unaligned accesses***
  - ***Data Types***
  - ***Addressing modes***
- ***Unaligned access***
  - ***LC-3b does not allow unaligned access***
  - ***DEC: PDP-11 (no), VAX (yes), Alpha (no)***
- ***Data types (rich or lean)***
  - ***Integers, floats of various sizes***
  - ***Doubly-linked list, character string***
- ***Addressing modes (rich or lean)***
  - ***Indirect addressing***
  - ***Autoincrement, postdecrement***
  - ***SIB byte in x86***

ISA Tradeoffs (continued)

- ***VLIW vs …***
  - ***VLIW: compiler does it***
  - ***Superscalar: part of the microarchitecture***
- ***0,1,2,3 address machine (how many EXPLICIT)***
  - ***LC-3b: 3 address***
  - ***x86: 2 address***
  - ***VAX: both 2 and 3***
  - ***Old days: one address (registers were expensive)***
  - ***Stack machine: 0 address***
- ***Precise exceptions vs …***
  - ***Precise exceptions: today, everyone***
  - ***IBM 360/91: NO***
- ***Privilege modes***
  - ***Most ISA have two – supervisor and user***
  - ***VAX had four***

Computer Architecture: A Science of Tradeoffs

- ***ISA***
- ***Microarchitecture***
- ***System***

Microarchitecture Tradeoffs

- ***CPI vs. cycle time (or, IPC vs. frequency)***
- ***in-order vs. out-of-order execution***
- ***Speculate vs. stand around and wait***
- ***Issue-width***
- ***ASIC vs. programmed control***
- ***Pipeline depth***
- ***Cache characteristics***
- ***Use of chip real estate***
  - ***Better branch predictor, accelerators, microcode, cache***
- ***Unified vs separate instruction and data caches***

Microarchitecture Tradeoffs (continued)

- ***Fault tolerance***
  - ***Yesterday: Tandem (Two cores in lock step)***
  - ***Tomorrow: repeat to remove soft errors (or two cores in lock step)***
- ***On-chip latency – near neighbor communication***
- ***Vertical vs Horizontal Microcode***
  - ***Selective (generally one micro-op per microinstruction***
  - ***Constructive (many fields, generally each specifies a micro-op)***
- ***Vertical vs Horizontal Microcode***
- ***One level vs. two-level microcode (e.g., Nanodata QM-1)***
- ***On-chip interconnect (contention, cost, latency)***
- ***Prefetching***
- ***Branch target buffer – saves a bubble on taken branch***
- ***Partitioning – until when?***

Computer Architecture: A Science of Tradeoffs

- ***ISA***
- ***Microarchitecture***
- ***System***

System Tradeoffs

- ***Homogeneous vs heterogeneous cores***
- ***One interface to software or more than one***
- ***Interconnection structure***
- ***Inter-core cache organization***
- ***Implications of the memory controller***
- ***Prefetching in a multicore environment***
- ***MP granularity (loosely coupled vs. tightly)***
- ***Distributed shared memory versus centralized***
- ***Use of commodity vs tailored parts***
- ***What do we integrate on chip***
  - ***Memory controller***
  - ***Analog devices***

Various Perspectives

- ***A Global Perspective***
  - ***Relevance** of the Transformation Hierarchy*
- ***A Microarchitecture Perspective***
  - ***The three ingredients to performance***
- ***A Physical Perspective in a changing world***
  - ***Wire delay (recently relevant) – Why? (frequency)***
  - ***Bandwidth (recently relevant) – Why? (multiple cores)***
  - ***Power, energy (recently relevant) – Why? (cores, freq)***
  - ***Soft errors (recently relevant) – Why? (freq)***
  - ***Partitioning (since the beginning of time)***

Various Perspectives

- ***A Global Perspective***
  - ***Relevance** of the Transformation Hierarchy*
- ***A Microarchitecture Perspective***
  - ***The three ingredients to performance***
- ***A Physical Perspective in a changing world***
  - ***Wire delay (recently relevant) – Why? (frequency)***
  - ***Bandwidth (recently relevant) – Why? (multiple cores)***
  - ***Power, energy (recently relevant) – Why? (cores, freq)***
  - ***Soft errors (recently relevant) – Why? (freq)***
  - ***Partitioning (since the beginning of time)***

Problem

Algorithm

Program

ISA (Instruction Set Arch)

Microarchitecture

Circuits

Electrons

Architecture vs Microarchitecture

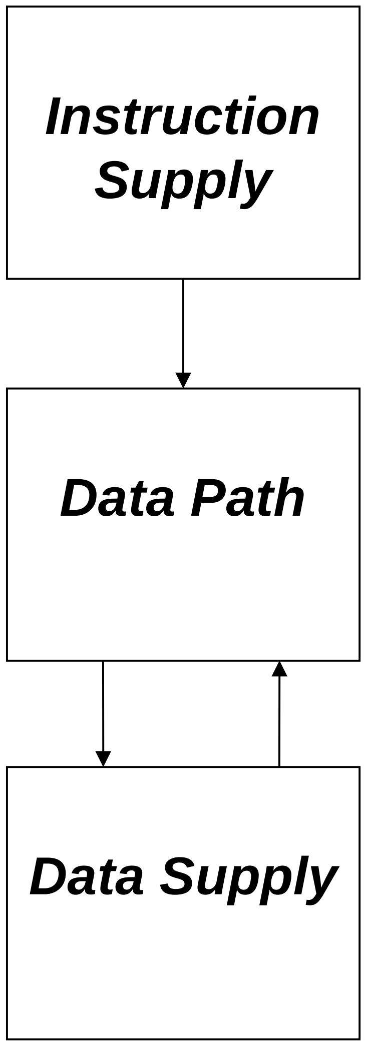
- ***Architecture***
  - ***Visible to the software***
  - ***Address Space, Addressability***
  - ***Opcodes, Data Types, Addressing Modes***
  - ***Support for Multiprocessors (e.g., TSET)***
  - ***Support for Multiprogramming (e.g., LDCTX)***
- ***Microarchitecture***
  - ***Not Visible to the software***
  - ***Caches (although this has changed, …sort of)***
  - ***Branch Prediction***
  - ***The instruction cycle***
  - ***Pipelining***

DIGRESSION (nugget): You have a brilliant idea, and
It requires a change to the ISA or to the uarchitecture.

Various Perspectives

- ***A Global Perspective***
  - ***Relevance** of the Transformation Hierarchy*
- ***A Microarchitecture Perspective***
  - ***The three ingredients to performance***
- ***A Physical Perspective in a changing world***
  - ***Wire delay (recently relevant) – Why? (frequency)***
  - ***Bandwidth (recently relevant) – Why? (multiple cores)***
  - ***Power, energy (recently relevant) – Why? (cores, freq)***
  - ***Soft errors (recently relevant) – Why? (freq)***
  - ***Partitioning (since the beginning of time)***

Microarchitecture (The Requirement)

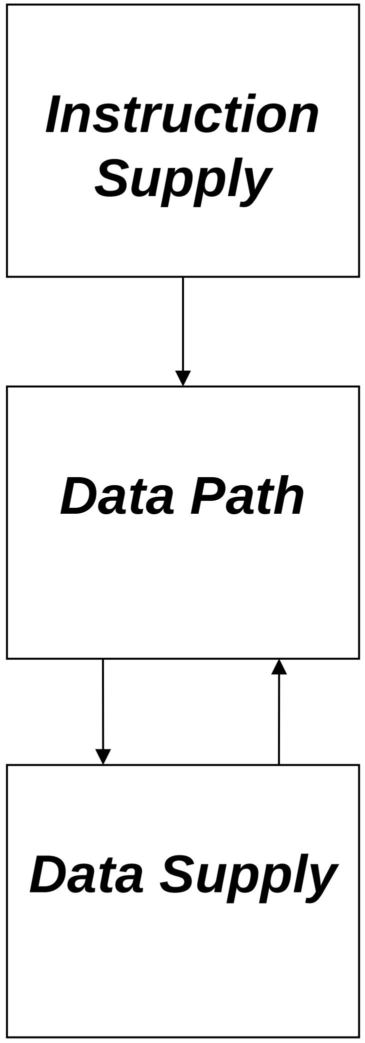


- *Perfect Instruction Cache*
- *No packet breaks*
- *100% branch prediction*

- *Perfect data flow*
- *Irregular parallelism*
- *Enough functional units*
- *Perfect interconnect*

- *Infinite capacity*
- *Single cycle access time*

Microarchitecture (The Problem)

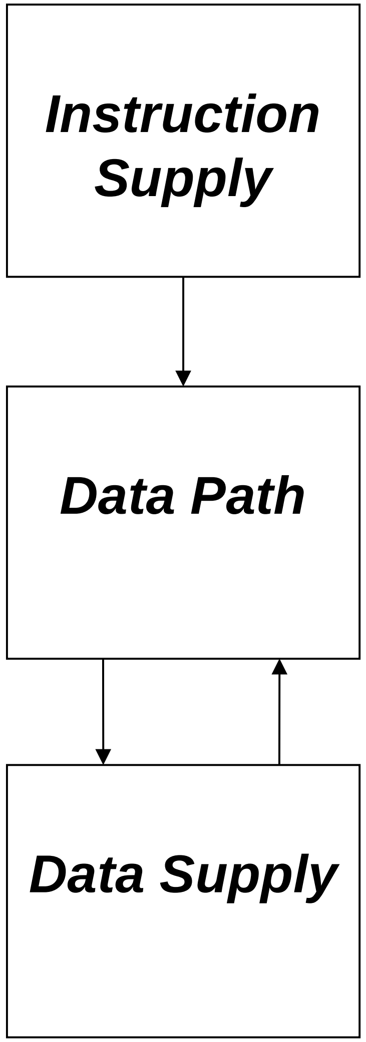


- ***Bandwidth (Pins)***
- ***Latency***
- ***Conditional branches***

- ***Data Dependencies***
- ***Sequential bottleneck***

- ***Bandwidth***
- ***Latency***

Microarchitecture (The Solution)



- ***Block –Structure***
- ***Bandwidth (microcode)***
- ***Latency (prefetch)***
- ***Branch Prediction***

- ***Multiple Issue***
- ***Deep pipelines***
- ***Data flow***

- ***Bandwidth (data organization)***
- ***Latency (prefetch, poststore)***

A few more words on Data Supply

- ***Memory is particularly troubling***
  - ***Off-chip latency (hundreds of cycles, and getting worse)***
  - ***What can we do about it?***
    - ***Larger caches***
    - ***Better replacement policies***
    - ***Predict what is in memory (value prediction)***

- ***Is MLP (Memory level parallelism) the answer***
  - ***Wait for two accesses at the same time***
  - ***Do parallel useful work while waiting (Runahead)***

George Michael’s Three Elements for Performance

- ***Only the programmer knows the ALGORITHM***
  - ***Pragmas***
  - ***Pointer chasing***
  - ***Partition code, data***
- ***Only the COMPILER knows the future (sort of ??)***
  - ***Predication***
  - ***Prefetch/Poststore***
  - ***Block-structured ISA***
- ***Only the HARDWARE knows the past***
  - ***Branch directions***
  - ***Cache misses***
  - ***Functional unit latency***

Various Perspectives

- ***A Global Perspective***
  - ***Relevance** of the Transformation Hierarchy*
- ***A Microarchitecture Perspective***
  - ***The three ingredients to performance***
- ***A Physical Perspective in a changing world!***
  - ***Wire delay (recently relevant) – Why? (frequency)***
  - ***Bandwidth (recently relevant) – Why? (multiple cores)***
  - ***Power, energy (recently relevant) – Why? (cores, freq)***
  - ***Soft errors (recently relevant) – Why? (freq)***
  - ***Partitioning (since the beginning of time)***

Numbers (because comp arch is obsessed with numbers)

- ***The Baseline – Make sure it is the best***
  - ***Superlinear speedup (Are you evil, or just confused?)***
  - ***Recent example, one core vs. 4 cores with ability to fork***
- ***The Simulator you use – Is it bug-free?***
- ***Understanding vs "See, it works!"***
  - ***16/64***
- ***You get to choose your experiments***
  - ***SMT: If throughput is your metric, run the idle process!***
  - ***Combining cores: what should each core look like***
- ***You get to choose the data you report***
  - ***Wrong path detection: WHEN was the wrong path detected***
- ***Never gloss over anomalous data***

Speculation

- ***Why good? – improves performance***
- ***How? – we guess***
  - ***Starting with the design of ALUs, many years ago!***
  - ***Branch prediction – enables parallelism***
  - ***Way prediction***
  - ***Data prefetching – enables parallelism***
  - ***Value prediction – enables parallelism***
  - ***Address prediction – enables parallelism***
  - ***Memory disambiguation – enables parallelism***
- ***Why bad? – consumes energyl***

Moore's Law

- ***What is it?***
  - ***The law itself: ??***
  - ***A law of Physics? Microarchitecture? Psychology?***
- ***Why has it been important?***
  - ***Everyone knows: chip resources (2300 transistors initially)***
    - ***5 billion transistors today***
  - ***Just as important: frequency (106 Kilohertz initially)***
    - ***Gigahertz today***
  - ***i.e., We can do more computing concurrently and faster!***
- ***Why all the attention today?***
  - ***Too expensive to continue making smaller transistors***
    - ***7 nanometers = 70 Angstroms***
  - ***Charles Leiserson et al at MIT: Plenty of room at the top***
  - ***I say: True, but still plenty to do at the bottom***

The von Neumann Model

- ***The classical model of computing***
  - ***Not really von Neumann***
  - ***Von Neumann was about co-locating inst, data in same mem***
  - ***The model has been declared dead for the future***
- ***Digression**: Wrong!*
  - ***Future machines will include lots of accelerators***
  - ***The von Neumann machine will be needed to maintain order***

Hardware: Sequential or Parallel?

- ***First the nonsense: Hardware is Sequential, cycle by cycle***
- ***Hardware is inherently parallel***
  - ***It has been since time began***
  - ***Then why the sudden interest?***
- ***Why is parallel important?***
  - ***A simple example: factorial***
  - ***It allows us to compute faster than the speed of light!***
- ***The key idea is Synchronization***
  - ***It can be explicit***
  - ***It can be implicit***
- ***Pipelining***
  - ***Parallelism at its most basic level***
  - ***Everyone in the world understands that (e.g., factories)***
- ***Speculation (formerly a no-brainer, today it depends)***
- ***Single thread vs. multiple threads***
- ***Single core vs. multiple cores***

Hardware or Software

- ***Do it in hardware (e.g., ASIC)***
  - ***Takes time, not easily changed***
  - ***Generally higher performance***
- ***Do it in software (Programmed control)***
  - ***Easier, faster to implement, easily changed***
  - ***Generally lower performance***
- ***Which is better?***
  - ***It depends***
  - ***What about FPGAs?***
    - ***LUTs (Look-up Tables)***
    - ***Pass transistors***

Design Principles

- ***Critical path design***
- ***Bread and Butter design***
- ***Balanced design***

Critical path design

- ***The ill-advised performance equation states:***
  - ***Performance = 1/(Time) = 1/(length x CPI x cycle time)***
  - ***CPI varies depending on what can hide the latency***
- ***Likely paths***
  - ***ALU***
  - ***On-chip storage access***
  - ***Microsequencer function***
- ***Methodology***
  - ***Pick the longest***
  - ***Work on shortening it until it no longer is***
  - ***Iterate***
- ***Bad design example: Removing ucode had no effect***

Bread and Butter design

- ***What does your machine have to do real fast?***
  - ***Or what if optimized will be a BIG win?***
- ***Concentrate your efforts there***
- ***The other stuff? Just be sure it not done too badly***
- ***Bad design example: The DEC 2080***

Balanced Design

- ***Front end, back end should make sense together***
- ***Bad design #1: 6-wide issue, four functional units***
- ***Bad design #2: A supercomputer with one result bus***
- ***Bad design #3: Multi-threading with one ALU***

Design Methodology

- ***Specify your design point***
- ***Identify the Bread and Butter***
- ***Optimize the Bread and Butter***
- ***Cover the rest***

Design Points

-- Performance

-- Reliability

-- Availability

-- Cost

-- Power

-- Time to Market

Role of the Architect

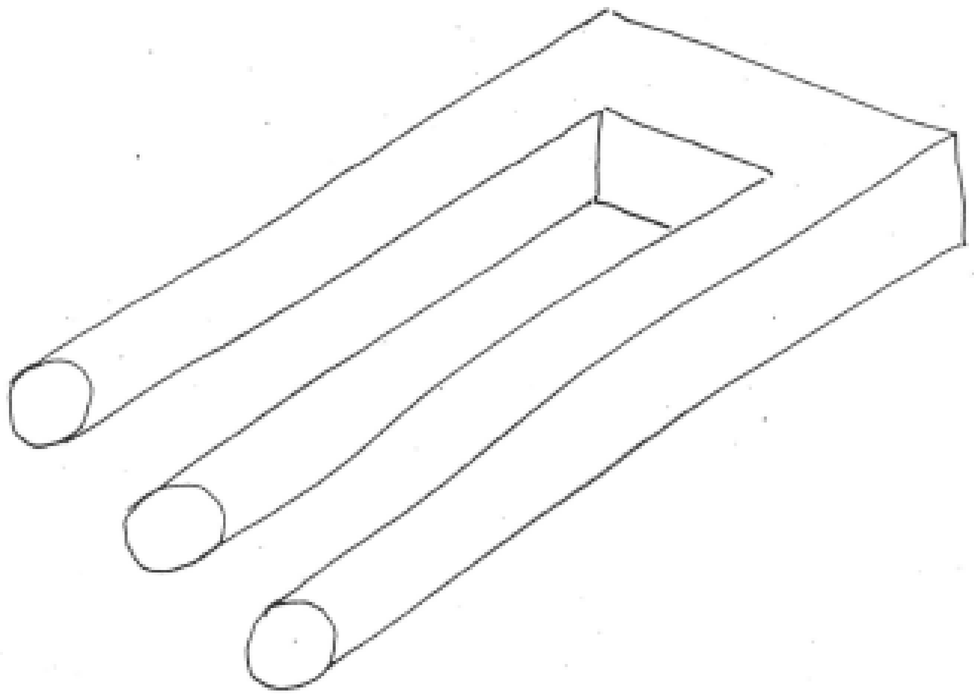
-- Look Backward (Examine old code)

-- Look forward (Listen to the dreamers)

-- Look Up (Nature of the problems)

-- Look Down (Predict the future of technology)

Finally, people are always telling you: Think outside the box



I prefer: Expand the box

A Few Specifics

* *HPS – expanded on Tomasulo*

* *SMT – expanded on Burton*

* *Perceptron predictor – expanded on Widrow/Rosenblatt/etc.*

Something you are all familiar with: Look-ahead Carry Generators

- ***They speed up ADDITION***

- ***But why do they work?***

Addition

$$\begin{array}{cc} 1 & 2 \\ & 9 \\ \hline 2 & 1 \\ \uparrow & \uparrow \end{array} \qquad\qquad \begin{array}{cccccc} 1 & 8 & 2 & 3 & 7 & 8 \\ 6 & 4 & 5 & 2 & 5 & 9 \\ \hline 8 & 2 & 7 & 6 & 3 & 7 \\ \uparrow & & & & & \uparrow \end{array}$$

Question:

- ***What is computer architecture?***

- ***It is a contract between***
  - ***The software (what it demands)***
  - ***The hardware (what it agrees to deliver)***

Question:

- ***What is microarchitecture?***

- ***It is a science of tradeoffs:***
  - ***What functionality we will deliver***
  - ***At what performance***
  - ***At what cost***

Question:

- ***How do we compute faster than the speed of light?***
- ***We do things concurrently.***

Question:

- ***Should we add a feature if the clock slows by 10% ?***

- ***Normally no, unless***
  - ***The benefit of the feature taking a single cycle outweighs***
  - ***the fact that everything else takes 10% more time.***

Question: Is computer architecture dead?

Answer: Computer Architecture will always be alive
and healthy as long as people can dream.

(Dreamers are not the architects, they are those who
want to use machines in new and interesting ways)

Computer Architecture is about the interface between
what technology can provide and the market demands

Tack!