The Past is Prologue: A New Golden Age for Computer Architecture

David Patterson
UC Berkeley and Google
July 16, 2018
Lessons of last 50 years of Computer Architecture

1. Software advances can inspire architecture innovations
2. Raising the hardware/software interface creates opportunities for architecture innovation
3. Ultimately the marketplace settles architecture debates
By early 1960’s, *IBM had 4 incompatible lines of computers!*

701  7094
650  7074
702  7080
1401 7010

Each system had its own:

- Instruction set architecture (ISA)
- I/O system and Secondary Storage:
  - magnetic tapes, drums and disks
- Assemblers, compilers, libraries,...
- Market niche: business, scientific, real time, ...

*IBM System/360 – one ISA to rule them all*
Control versus Datapath

- Processor designs split between *datapath*, where numbers are stored and arithmetic operations computed, and *control*, which sequences operations on datapath.

- Biggest challenge for computer designers was getting control correct.

Maurice Wilkes invented the idea of *microprogramming* to design the control unit of a processor.*

- Logic expensive vs. ROM or RAM
- ROM cheaper and faster than RAM
- *Control design now programming*

## Microprogramming in IBM 360

<table>
<thead>
<tr>
<th>Model</th>
<th>M30</th>
<th>M40</th>
<th>M50</th>
<th>M65</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datapath width</td>
<td>8 bits</td>
<td>16 bits</td>
<td>32 bits</td>
<td>64 bits</td>
</tr>
<tr>
<td>Microcode size</td>
<td>4k x 50</td>
<td>4k x 52</td>
<td>2.75k x 85</td>
<td>2.75k x 87</td>
</tr>
<tr>
<td>Clock cycle time (ROM)</td>
<td>750 ns</td>
<td>625 ns</td>
<td>500 ns</td>
<td>200 ns</td>
</tr>
<tr>
<td>Main memory cycle time</td>
<td>1500 ns</td>
<td>2500 ns</td>
<td>2000 ns</td>
<td>750 ns</td>
</tr>
<tr>
<td>Price (1964 $)</td>
<td>$192,000</td>
<td>$216,000</td>
<td>$460,000</td>
<td>$1,080,000</td>
</tr>
<tr>
<td>Price (2018 $)</td>
<td>$1,560,000</td>
<td>$1,760,000</td>
<td>$3,720,000</td>
<td>$8,720,000</td>
</tr>
</tbody>
</table>
IC Technology, Microcode, and CISC

- Logic, RAM, ROM all implemented using same transistors
- Semiconductor RAM ≈ same speed as ROM
- With Moore’s Law, memory for control store could grow
- Since RAM, easier to fix microcode bugs
- Allowed more complicated ISAs (CISC)
- Minicomputer (TTL server) example:
  - Digital Equipment Corp. (DEC)
  - VAX ISA in 1977
- 5K x 96b microcode
Writable Control Store

- If Control Store is RAM, then could tailor “firmware” to application: “Writable Control Store”
- Microprogramming became popular in academia
  - Patterson PhD thesis*
- Xerox Alto (Bit Slice TTL) in 1973**
  - 1st personal computer with 1st Graphical User Interface and 1st Ethernet
  - BitBlt and Ethernet controller in microcode

* Verification of microprograms, David Patterson, UCLA, 1976
** “The design of a system for the synthesis of correct microprograms,” David Patterson, Proc. 8th Annual Workshop of Microprogramming, 1975
Microprocessor Evolution

- Rapid progress in 1970s, fueled by advances in MOS technology, imitated minicomputers and mainframe ISAs
- “Microprocessor Wars”: compete by adding instructions (easy for microcode), justified given assembly language programming
- Intel iAPX 432: Most ambitious 1970s micro, started in 1975
  - 32-bit capability-based, object-oriented architecture, custom OS written in Ada
  - Severe performance, complexity (multiple chips), and usability problems; announced 1981
- Intel 8086 (1978, 8MHz, 29,000 transistors)
  - “Stopgap” 16-bit processor, 52 weeks to new chip
  - ISA architected in 3 weeks (10 person weeks) assembly-compatible with 8 bit 8080
- IBM PC 1981 picks Intel 8088 for 8-bit bus (and Motorola 68000 was late)
- Estimated PC sales: 250,000
- Actual PC sales: 100,000,000 ⇒ 8086 “overnight” success
- Binary compatibility of PC software ⇒ bright future for 8086
Analyzing Microcoded Machines 1980s

- **HW/SW interface rises from assembly to HLL programming**
  - Compilers now source of measurements

- **John Cocke group at IBM**
  - Worked on a simple pipelined processor, 801 minicomputer (ECL server), and advanced compilers inside IBM
  - Ported their compiler to IBM 370, only used simple register-register and load/store instructions (similar to 801)
  - Up to 3X faster than existing compilers that used full 370 ISA!

- **Emer and Clark at DEC in early 1980s***
  - Found VAX 11/780 average clock cycles per instruction (CPI) = 10!
  - Found 20% of VAX ISA ⇒ 60% of microcode, but only 0.2% of execution time!

- **Patterson after ’79 DEC sabbatical: repair microcode bugs in microprocessors?****
  - What’s magic about ISA interpreter in Writable Control Store? Why not other programs?

---

** “RISCy History,” David Patterson, May 30, 2018, Computer Architecture Today Blog
From CISC to RISC

• Use RAM for instruction *cache* of user-visible instructions
  • Software concept: Compiler vs. Interpreter
  • Contents of fast instruction memory change to what application needs now vs. ISA interpreter

• Use simple ISA
  • Instructions as simple as microinstructions, but not as wide
  • Enable pipelined implementations
  • Compiled code only used a few CISC instructions anyways

• Further benefit with chip integration
  • In early ‘80s, could finally fit 32-bit datapath + small caches on a single chip
  • Chaitin’s register allocation scheme* benefits load-store ISAs

Berkeley and Stanford RISC Chips

RISC-I (1982) Contains 44,420 transistors, fabbed in 5 µm NMOS, with a die area of 77 mm², ran at 1 MHz

RISC-II (1983) contains 40,760 transistors, was fabbed in 3 µm NMOS, ran at 3 MHz, and the size is 60 mm²

Stanford MIPS (1983) contains 25,000 transistors, was fabbed in 3 µm & 4 µm NMOS, ran at 4 MHz (3 µm), and size is 50 mm² (4 µm)

Microprocessor without Interlocked Pipeline Stages


"Iron Law" of Processor Performance: How RISC can win

\[
\text{Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Time}}{\text{Clock cycle}}
\]

- CISC executes fewer instructions / program (≈ 3/4X instructions) but many more clock cycles per instruction (≈ 6X CPI)
  ⇒ RISC ≈ 4X faster than CISC

“Performance from architecture: comparing a RISC and a CISC with similar hardware organization,”
CISC vs. RISC Today

PC Era
- Hardware translates x86 instructions into internal RISC instructions *(Compiler vs Interpreter)*
- Then use any RISC technique inside MPU
- > 350M / year !
- x86 ISA eventually dominates servers as well as desktops

PostPC Era: Client/Cloud
- IP in SoC vs. MPU
- Value die area, energy as much as performance
- > 20B total / year in 2017
  - x86 in PCs peaks in 2011, now decline ~8% / year (2016 < 2007)
  - x86 servers ⇒ Cloud ~10M servers total* (0.05% of 20B)
- 99% Processors today are RISC
- Marketplace settles debate

*A Decade of Mobile Computing*, Vijay Reddi, 7/21/17, Computer Architecture Today
VLIW: Very Long Instruction Word (Josh Fisher)

- Multiple operations packed into one instruction (like a wide microinstruction)
- Each operation slot for a fixed function, constant operation latencies
From RISC to Intel/HP Itanium, EPIC IA-64

- Time for Intel to increase address to 64 bits
- EPIC is Intel’s name for their VLIW architecture
  - “Explicitly Parallel Instruction Computing” architecture, chip was “Itanium”
  - Developed jointly with HP starting 1994
- AMD wouldn’t be able to make Itanium, so had to make 64-bit x86
- Many companies gave up RISC for Itanium since it was widely believed to be inevitable (Microsoft, SGI, Hitachi, Bull, …)
VLIW Issues and an “EPIC Failure”

▪ “The Itanium approach...was supposed to be so terrific—until it turned out that the wished-for compilers were basically impossible to write.”
  - Donald Knuth, Stanford

▪ Code size explosion
▪ Unpredictable branches
▪ Variable memory latency (unpredictable cache misses)
▪ Pundits noted delays and under performance of Itanium product ridiculed by the chip industry

Marketplace rejects VLIW for general purpose computers

Itaniumum ⇒ “Itanic” (like infamous ship Titanic)
Lessons of last 50 years of Computer Architecture

1. Software advances can inspire architecture innovations
   - Microprogramming - control as code
   - RISC: Compile vs. ISA interpreter in microcode
   - Pentium Pro x86: Hardware translator vs. interpreter

2. Raising the HW/SW interface creates arch. opportunities
   - Assembly to High Level Language → RISC

3. Ultimately the marketplace settles architecture debates
   - 432, Itanium lost
   - IBM S/360, 8086 (PC Era), RISC (PostPC Era) won
Outline

Part I: History of Architecture - Mainframes, Minicomputers, Microprocessors, RISC vs CISC, VLIW

Part II: Current Architecture Challenges - Ending of Dennard Scaling and Moore’s Law, Security

Part III: Future Architecture Opportunities - Domain Specific Languages and Architecture, Open Architectures, Agile Hardware Development
Fundamental Changes in Technology

• Technology
  • End of Dennard scaling: power becomes the key constraint
  • Ending of Moore’s Law: transistors improvement slows

• Architectural
  • Limitation and inefficiencies in exploiting instruction level parallelism end the uniprocessor era in 2004
  • Amdahl’s Law and its implications end “easy” multicore era

• Products
  • PC/Server ⇒ IoT, Mobile/Cloud
Moore’s Law Slowdown in Intel Processors

Moore, Gordon E. "No exponential is forever: but ‘Forever’ can be delayed!"
Energy scaling for fixed task is better, since more and faster transistors

Power consumption based on models in “Dark Silicon and the End of Multicore Scaling,” Hadi Esmaelizadeh, ISCA, 2011
End of Growth of Single Program Speed?

CISC
2X / 3.5 yrs
(22%/yr)

RISC
2X / 1.5 yrs
(52%/yr)

Multicore
2X / 3.5 yrs
(23%/yr)

End of Dennard Scaling
⇒
End of Amdahl’s Law
⇒
2X / 6 yrs
(12%/yr)

End of the Line?
2X / 20 yrs
(3%/yr)

Based on SPECintCPU. Source: John Hennessy and David Patterson, Computer Architecture: A Quantitative Approach, 6/e. 2018
Current Security Challenge

- Spectre: speculation $\Rightarrow$ timing attacks that leak $\geq 10$ kb/s
- More microarchitecture attacks on the way*
- Spectre is bug in computer architecture definition vs chip
- Need Computer Architecture 2.0 to prevent timing leaks**
- Software not yet secure $\Rightarrow$ how can hardware help?

* “A Survey of Microarchitectural Timing Attacks and Countermeasures on Contemporary Hardware,” Qian Ge, Yuval Yarom, David Cock, and Gernot Heiser, Journal of Cryptographic Engineering, April, 2018
** “A Primer on the Meltdown & Spectre Hardware Security Design Flaws and their Important Implications”, Mark Hill, 2/15/18, Computer Architecture Today
Part II: Challenges Summary

- Performance improvements are at a standstill
  - Slowing Moore’s Law
  - No more Dennard Scaling
    - Microarchitecture techniques: ILP, multicore, etc. are inefficient, hence burn energy

- State of computer security is embarrassing for all of us in the computing field
  - Seems unlikely systems will ever become secure using software only solutions

"What we have before us are some breathtaking opportunities disguised as insoluble problems." - John Gardner, 1965
Outline

Part I: History of Architecture -
Mainframes,
Minicomputers,
Microprocessors,
RISC vs CISC, VLIW

Part II: Current Architecture Challenges -
Ending of Dennard Scaling and Moore’s Law, Security

Part III: Future Architecture Opportunities -
Domain Specific Languages and Architecture,
Open Architectures, Agile Hardware Development
What Opportunities Left?

▪ **SW-centric**
  - Modern scripting languages are interpreted, dynamically-typed and encourage reuse
  - Efficient for programmers but not for execution

▪ **HW-centric**
  - Only path left is *Domain Specific Architectures*
  - Just do a few tasks, but extremely well

▪ **Combination:**
  - Domain Specific Languages & Architectures
  - Raises level of HW/SW Interface
What’s the Opportunity?

Matrix Multiply: relative speedup to a Python version (18 core Intel)

from: “There’s Plenty of Room at the Top,” Leiserson, et. al., to appear.
Domain Specific Architectures (DSAs)

• Achieve higher efficiency by tailoring the architecture to characteristics of the domain
  • Not one application, but a domain of applications
    - Different from strict ASIC since still runs software
  • Requires more domain-specific knowledge then general purpose processors need

• Examples:
  • Neural network processors for machine learning
  • GPUs for graphics, virtual reality
  • Programmable network switches and interfaces
Why DSAs Can Win (no magic)
Tailor the Architecture to the Domain

• More effective parallelism for a specific domain:
  • SIMD vs. MIMD
  • VLIW vs. Speculative, out-of-order
• More effective use of memory bandwidth
  • User controlled versus caches
• Eliminate unneeded accuracy
  • IEEE replaced by lower precision FP
  • 32-64 bit integers to 8-16 bit integers
• Domain specific programming language provides path for software
Deep learning is causing a machine learning revolution

ML Training Trends

Since 2012, AI training compute demand doubles every 3.5 months!
(Moore’s Law doubled “only” every 18 months)

From “AI and Compute.”
Dario Amodei and Danny Hernandez, May 16, 2018
Tensor Processing Unit v1

Google-designed chip for neural net inference

In production use for 36 months: used by billions on search queries, for neural machine translation, for AlphaGo match, …

In-Datacenter Performance Analysis of a Tensor Processing Unit, Jouppi, Young, Patil, Patterson et al., ISCA 2017,
TPU: High-level Chip Architecture

- The Matrix Unit: 65,536 (256x256) 8-bit multiply-accumulate units
- 700 MHz clock rate
- Peak: 92T operations/second
  - 65,536 * 2 * 700M
- >25X as many MACs vs GPU
- >100X as many MACs vs CPU
- 4 MiB of on-chip Accumulator memory
- 24 MiB of on-chip Unified Buffer (activation memory)
- 3.5X as much on-chip memory vs GPU
- Two 2133MHz DDR3 DRAM channels
- 8 GiB of off-chip weight DRAM memory
Measure performance of Machine Learning?

See MLPerf.org ("SPEC for ML")

- Benchmark suite being developed by
  ≥13 companies and
  ≥6 universities
TPUv2 (announced May 2017)

Charter: Do **training**, so always **bigger problems**.
   More general, more flexible.
   Supports backprop, activation storage.

Hardware: Still systolic arrays for matrix operations.
   Much better “everything else”: vector, scalar
   Multi-chip parallelism and interconnect.

Numerics: bfloat16 (same exponent size as float32).

High-Bandwidth Memory (HBM) unlocks peak compute.

Compute, Memory, Network, System ⇒ Supercomputer (for ML).
TPUv3 (announced May 2018)

Liquid cooling: more heat.

Bigger pods: more scale.

Rapid iteration!
Current Neural Network Architecture Debate

- Nvidia enhances many thread approach of GPU
- Google enhances single large 2D multiplier of TPU chip
- Intel enhances SIMD instructions of CPU
  - Also buys Nervana to offer custom chip solutions
- Microsoft enhances FPGAs that customize “hardware”
- > 45 startups with their own architecture bets
- *Ultimately the marketplace settles architecture debates*
Part III: DSL/DSA Summary

- Lots of opportunities
  - Make Python run like C with compiler + HW
    - Deja vu: make HLLs fast on RISC
    - Domain Specific Architectures
- But, new approach to computer architecture is needed.
- The Renaissance computer architecture team is vertically integrated. Understands:
  - Applications
  - DSLs and related compiler technology
  - Principles of architecture
  - Implementation technology
- Everything old is new again!
Part III: Open Architectures

- Software advances can inspire architecture innovations
- Why open source compilers and operating systems but not ISAs?
RISC-V Origin Story

- UC Berkeley Research using x86 & ARM?
  - Impossible – too complex and IP issues
- 2010 started “3-month project” to develop own clean-slate ISA
  - Krste Asanovic, Andrew Waterman, Yunsup Lee, Dave Patterson
- 4 years later, released frozen base user spec

Why are outsiders complaining about changes of RISC-V in Berkeley classes?
What’s Different About RISC-V?

- **Simple**
  - Far smaller than proprietary ISAs
  - 2700 pages for ARMv8 manual vs 200 for RISC-V manual

- **Clean-slate design**
  - 25 years later, so can learn from mistakes of predecessors
  - Avoids microarchitecture or technology-dependent features

- **Modular**
  - Small standard base ISA*
  - Multiple standard extensions

- **Supports DSAs**
  - Vast opcode space reserved

- **Community designed**
  - RISC-V Foundation
  - Grow via optional extensions vs. incremental required features
  - Optional extensions for technical reasons vs. for marketing reasons by private corporation

*“How close is RISC-V to RISC-I?”*
David Patterson, 9/19/17, ASPIRE Blog
Foundation Members since 2015
Foundation Working Groups (partial list)

- Bit Manipulation
- Compliance
- Debug
- Memory Model
- Privileged Spec
- Vector
- Security
- Base ISA Opcode
NVDLA: An Open DSA and Implementation

- NVDLA: NVIDIA Deep Learning Accelerator for DNN Inference
- Free & Open: All SW, HW, and documentation on GitHub
- Scalable, configurable design
  - Each block operates independently or in pipeline to bypass memory
  - Data type configurable: int8, int16, fp16,
  - 2D MAC array configurable: 8 to 64 x 4 to 64
  - Size scales 6X (0.5 - 3mm²), power scales 15X (20 - 300 mW)
- RISC-V core as host (optional)
Security and Open Architecture

- Security community likes simple, verifiable (no trap doors), alterable, free and open architecture and implementations
- Equally important is number of people and organizations performing architecture experiments
  - Want all the best minds to work on security
- Plasticity of FPGAs + open source RISC-V implementations and SW ⇒ novel architectures can be deployed online, evaluated, & iterated in weeks vs years (even 100 MHz OK)
- RISC-V may become security exemplar via HW/SW codesign by architects and security experts
Part III: Open Architecture Summary

- Typically simpler as not marketing driven (helps verification, security) > area/power/performance at low end and = at high end
- More organizations designing processors (open source) ⇒ more competitive marketplace ⇒ faster innovation
- Will become primary experimental vehicle of security experts?

Open Architecture Goal

Create industry-standard open ISAs for all computing devices

“Linux for processors”
Agile Hardware Development

▪ **Software advances can inspire innovations**

▪ Agile: small teams do short development between working but incomplete prototypes and get customer feedback per step

▪ Scrum team organization
  - 5 - 10 person team size
  - 2 - 4 week sprints for next prototype iteration

▪ New CAD enables SW Dev techniques to make small teams productive via abstraction & reuse
## Reuse: Shared Lines of RTL Code (Chisel)

<table>
<thead>
<tr>
<th>RISC-V Core</th>
<th>Z-scale</th>
<th>Rocket</th>
<th>BOOM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>32-bit 3-stage pipeline in-order 1-instruction issue L1 caches (≈ ARM Cortex-M0)</td>
<td>64-bit, FPU, MMU 5-stage pipeline in-order 1-instruction issue L1 &amp; L2 caches (≈ ARM Cortex-A5)</td>
<td>64-bit, FPU, MMU 5-stage pipeline out-of-order 2-, 3-, or 4- instruction issue L1 &amp; L2 caches (≈ ARM Cortex-A9)</td>
</tr>
<tr>
<td><strong>Unique LOC</strong></td>
<td>600 (40%)</td>
<td>1,400 (10%)</td>
<td>9,000 (45%)</td>
</tr>
<tr>
<td><strong>LOC all 3 share</strong></td>
<td>500 (30%)</td>
<td>500 (5%)</td>
<td>500 (5%)</td>
</tr>
<tr>
<td><strong>LOC Z-scale &amp; Rocket share</strong></td>
<td>500 (30%)</td>
<td>500 (5%)</td>
<td>---</td>
</tr>
<tr>
<td><strong>LOC Rocket &amp; BOOM share</strong></td>
<td>---</td>
<td>10,000 (80%)</td>
<td>10,000 (50%)</td>
</tr>
<tr>
<td><strong>Total LOC</strong></td>
<td>1,600</td>
<td>12,400</td>
<td>19,500</td>
</tr>
</tbody>
</table>
Agile Hardware Development Methodology

Small chip tape-out 100 chips 1x1mm @ 28nm is affordable at $14,000!

AWS FPGA F1 instance ⇒ develop new prototypes using cloud (nothing to buy)

Four 28nm & Six 45nm RISC-V Chips taped out in 5 years

Raven-1
Raven-2
Raven-3

EOS14
EOS16
EOS18
EOS20
EOS22
EOS24

2011
2012
2013
2014
2015

May
Apr
Aug
Feb
Jul
Sep
Mar
Nov
Mar

1 core + vector coprocessor
1.0 GHz (adaptive-clocking)
34 DP GFLOPS / Watt

2 cores, 1.7 GHz,
15 DP GFLOPS / Watt

Raven: ST 28nm FDSOI
EOS: IBM 45nm SOI
Lessons of last 50 years of Computer Architecture

1. **Software advances can inspire architecture innovations**
   - Microprogramming - control as software
   - RISC, x86 ISA - (Hardware) translator vs. interpreter
   - Open Architectures and Implementations
   - Agile Hardware Development

2. **Raising the HW/SW interface enables arch. opportunities**
   - Assembly to HLL ⇒ RISC
   - HLL to Domain Specific Language ⇒ DSA

3. **Ultimately the marketplace settles architecture debates**
   - 432, Itanium lost
   - IBM S/360, 8086 (PC Era), RISC (Post PC) Era won
   - CPU vs GPU vs TPU vs FPGA: DSA winner is ???