# 1.4 50 years of Computer Architecture: From the Mainframe CPU to the Domain-Specific TPU and the Open RISC-V Instruction Set

David Patterson, Google and UC Berkeley

## 1. Our Story Begins in the Early 1960s

IBM had four incompatible computer lines. Each had its own unique *instruction-set architecture* (*ISA*); I/O system; system software (assemblers, compilers, libraries); and market niches (business, scientific, real time). IBM engineers bet that they could invent a single ISA that would work for customers of all four lines. Moreover, the same program would run correctly on any implementation of that ISA, though at different speed and cost. That vision required a new way to build computers that would be *binary compatible* from the cheapest 8-bit model to the fastest 64-bit version.

While datapaths were straightforward, the hardest part of computer design then and now is control. Maurice Wilkes, a computing pioneer, proposed that control be built from Read Only Memory (ROM) [1]. At the time, Random Access Memory (RAM) was much less expensive than logic, and ROM was much less expensive than RAM. Wilkes dubbed it *microprogramming*, in that designing control resembled programming at a low, detailed level, and he called each word of the control ROM a *microinstruction*.

IBM engineers embraced microprogramming to deliver their grand goal. Each model of the new computer family would have its own ISA interpreter written in microinstructions customized to that model. The more hardware to control, the wider the microinstruction, and wider datapaths generally needed fewer microinstructions for the ISA interpreter since they computed more quickly. The cheapest model used 4000 microinstructions 50 bits wide and the fastest one used 2800 microinstructions 87 bits wide.

On April 7, 1964, IBM announced the most important technical event in the company's history:

> *"System/360 represents a sharp departure from concepts of the past in designing and building computers. …*
>
> *System/360 is a single system spanning the performance range of virtually all current IBM computers. …*
>
> *System/360 purchase prices range from $133,000 to $5,500,000."*

In today's dollars, the range was $1,054,000 to $43,570,000.

IBM bet the company that binary compatibility would work, and it won that bet. Correspondingly, *mainframe computers* dominated the high end of the information technology (IT) field for decades. IBM still sells a descendant of the System/360 more than 50 years later, making it the longest lasting ISA.

## 2. Complex-Instruction-Set Computers (CISC)

Built from small-scale and medium-scale integrated circuits, *minicomputers* were next on the IT scene. Logic, RAM, and ROM were all made from the same transistor in the 1970s. Moore's Law meant that the control store could be much larger, yet still affordable and fast. That increase led to larger microprograms, which supported larger ISAs with many sophisticated instructions. But, as microprograms grew, they were more likely to have bugs. To make microprograms easier to repair, control store became RAM as it was about the same speed as semiconductor ROM.

A shining example of minicomputers and complex ISAs was the VAX-11/780, which Digital Equipment Corporation (DEC) announced October 25, 1977. It had 5120 microinstructions that were 96 bits wide. The VAX line of minicomputers was a workhorse of the IT industry for the next decade.

## 3. The Dominant Microprocessor ISA

Moore's Law suggested that microprocessors would eventually compete with minicomputers in performance but at much lower cost. Just as IBM still sells an offshoot of the System/360, Intel founder Gordon Moore believed that the ISA that succeeded the Intel 8080 8-bit microprocessor would last the lifetime of the company.

In 1975, a year after Intel completed the 8080, Moore started a skunk works in Oregon to invent an ISA worthy of Intel's future. He hired many new PhDs in computer science to deliver that goal.

Their ISA was a complete break from the 8080: It was a stack computer with no general-purpose registers; instruction lengths could be any number of bits; addresses were 32-bit capabilities; and they wrote a custom operating system for it in the esoteric programming language Ada. With considerable fanfare, in 1981 Intel announced the iAPX 432 [2]:

*"The vacuum tube, the transistor, the microprocessor—at least once in a generation an electronic device arises to shock and strain designers' understanding. The latest such device is the iAPX 432 micromainframe processor, a processor as different from the current crop of microprocessors (and indeed, mainframes) as those devices are from the early electromechanical analog computers of the 1940s."*

Alas, its large microprogram could not fit into a single chip, so the iAPX 432 was expensive, slow, and late. When the Oregon engineers told Moore in 1977 that their project would take much longer than he wanted, Intel launched an emergency project to develop a 16-bit successor to the 8080. The small team was given in only 52 weeks to invent an ISA, and design the chip. Given the abrupt schedule, the team designed the ISA in only 3 weeks of elapsed time with a total effort of 10 person weeks. With the goal of being assembly language compatible with the 8080, the ISA widened the 8080 accumulators and added a 20-bit segmented address space. Intel released the 8086 in 1978 with neither high hope or hype.

Around the same time, IBM started a group in Florida to develop a consumer product that would compete with the Apple I computer. They seriously considered using the Motorola 68000 microprocessor, whose ISA resembled the IBM System/360, but the 68000 was late. IBM went instead with a cost-reduced variant of the Intel 8086. IBM announced the resulting Personal Computer on August 12, 1981:

*"This is the computer for just about everyone who has ever wanted a personal system at the office, on the university campus or at home .... We believe its performance, reliability, and ease of use, make it the most advanced, affordable personal computer in the marketplace."*

IBM projected sales of 250,000 but sold 100,000,000 PCs, turning the Intel 8086 into an "overnight" success. When binary compatibility for PC software was factored in, the IBM PC also gave the 8086 a very bright future.

Gordon Moore correctly predicted that Intel's next ISA would dominate its future, but it was the emergency substitute 8086, not the anointed iAPX 432, that fulfilled the promise. (Intel discontinued that ISA in 1986.) Subsequently in 1985, Intel extended the 8086 ISA address size to 32 bits with the 80386, enabling the 80x86 ISA to dominate microprocessors for the next 15 years.

## 4. Reduced-Instruction-Set Computers (RISC)

Returning to larger computers, DEC engineers found 20% of VAX instructions were responsible for 60% of the microcode, but only accounted for 0.2% of execution time [3]. Contemporaneously, John Cocke and his group at IBM Research ported an experimental compiler to IBM System/370 that only used simple register-register and load/store instructions. These programs ran much faster than those generated from existing compilers that utilized the full ISA. Such results called into question the big microcoded interpreters in their larger control stores and the complex ISAs they enabled.

It was realized that an alternative was having an ISA so simple that it did not require a microcoded interpreter. Another way to think of it was that such instructions were as simple as microinstructions, but not as wide. The next insight was to convert the fast RAM of control store into an instruction cache of user-visible instructions. The contents of fast instruction memory could change to hold what the executing application needs rather than always containing an ISA interpreter.

This simple ISA also made it easy to have pipelined implementations, which led to faster clock rates. The advances of Moore's Law meant that in the early 1980s, a 32-bit datapath with small caches could fit in a single chip. The lack of chip crossings of an integrated design also meant faster operation. This ISA style became known as a *Reduced-Instruction-Set Computer* (*RISC*) in contrast to CISC [4].

One skepticism directed at RISC was that it would execute more instructions than a CISC, so how could it be better? The following formula resolved the dilemma:

$$\frac{Intructions}{Program} \times \frac{Clock\ cycles}{Instruction} \times \frac{Time}{Clock\ cycle} = \frac{Time}{Program}$$

CISC implementations executed fewer instructions per program, but RISC computers executed on average many fewer clock cycles per instruction (CPI). To their credit, DEC engineers published a paper 10 years later comparing a CISC to a RISC with a similar datapath. They found about a factor of three advantage for RISC [5]: the VAX executed roughly half as many instructions as the RISC, but its CPI was about six times higher.

RISC research projects at IBM, Berkeley, and Stanford paved the way for the commercial success of ARM [6], MIPS, POWER, SPARC [7], and many more. Correspondingly, Reduced Instruction Set Computers (RISC) became the dominant ISA of workstations, minicomputers, and servers from the mid 1980s to the turn of the century.

## 5. RISC versus CISC Today

The 80x86 eventually surpassed the RISC architectures by using hardware to translate the 80x86 ISA into internal RISC-like instructions, which allowed the 80x86 to copy any performance enhancements developed for RISC processors. (Intel calls them *micro operations* and AMD calls them *RISC operations*.) Examples include deep pipelines, fetching multiple instructions per clock cycle, and branch prediction. Given superior semiconductor processing and circuit design, the 80x86 eventually had the fastest processors, and overtook the market in small servers from RISCs in addition to dominating the PC market.

As Moore's Law increased the transistor budgets, the extra overhead in area and energy of hardware translation was affordable for PCs and servers, but it was too costly for the embedded market. For example, 100% of Android and Apple phones and tablets use RISC processors. The size of the embedded market means billions of chips are shipped each year using RISC processors from ARM (Advanced RISC Machine), Synopsis ARC (Argonaut RISC Core), Cadence Tensilica, and MIPS. Figure 1.4.1 shows that RISC shipments grew about 24% annually recently, providing a sevenfold increase since 2007.

The 80x86 CISC ISA dominated chip shipments in the PC era, but RISC architectures control the PostPC era. We demark that era with the introduction of the iPhone in 2007. Annual 80x86 shipments peaked in 2011 at 365M, but have been declining about 8% annually since; Intel and AMD made fewer 80x86 chips in 2016 than in 2007. While 80x86 dominates the cloud portion of the PostPC Era, Reddi [8] estimates that the entire installed base for the Amazon, Google, and Microsoft clouds is 10M servers. While these chips are expensive, their volume is negligible; 10M RISC chips ship every 4 hours!

## 6. Very-Long-Instruction-Word (VLIW) Computers

Some researchers believed that Very-Long-Instruction-Word (VLIW) computers would replace CISC and RISC as the dominant ISA style. Similar in concept to the wide microinstructions, a single wide VLIW instruction commands multiple operations. Figure 1.4.2 gives an example of an instruction that could perform two integer operations, two memory accesses, and two floating-point operations. The supporting hardware would then include two integer units, two load-store units, and two floating-point units. Instead of the 32-bit instructions of RISC architectures, VLIW instructions were more than 100 bits.

In pure VLIW, there were no interlocks between instructions. If a VLIW instruction calculated a result need by a subsequent instruction, and the latency for that operation was three clock cycles, the dependent instruction had to execute exactly three clock cycles later to get the correct results. It was up to the software to schedule the operations properly, which made the hardware easier to design but upped the demands on compilers.

Figure 1.4.2 shows three distinct latencies for different operations within one VLIW instruction, which was typically true. While such a task would be challenging for the assembly-language programmer, VLIW advocates believed that advanced compiler technology could successfully map high-level language programs to the software-scheduled hardware.

In the mid 1990s, Intel faced a decision about the fate of the 80x86 ISA. Its 32-bit address space would soon be insufficient for many applications. While one choice was to extend the old ISA to 64 bits, similar to how it transitioned from 16 to 32 bits with the 80386, Intel had business and technical reasons to change the ISA. A business reason was to eject AMD as an ISA partner, as AMD also had the right to make microprocessors that were compatible with the 80x86 ISA. The RISC architectures also posed a business opportunity to Intel, as a new ISA might capture that market. The technical reason was that the 80x86 was an antiquated ISA that had relatively few registers, was missing some useful operations, and had legacy features that were maintained only for backwards binary-compatibility.

Consequently, Intel joined forces with Hewlett Packard in 1994 to promote *Explicitly-Parallel-Instruction Computing* (*EPIC*) as their 64-bit address successors to the 80x86, and HP PA-RISC ISAs. EPIC was essentially a binary-compatible variation of VLIW. The initial goal was to ship the first chip in 1997. Intel announced the official name of the processor, *Itanium*, on October 4, 1999:

*"EPIC is the old term for what is now known as the Itanium processor family architecture, co-developed by HP and Intel. This design philosophy will one day replace RISC and CISC. It is a gateway into the 64-bit future..."*

There was considerable publicity about the Itanium even before the first chips were available, with dozens of companies (HP, Microsoft, Silicon Graphics, Bull, Hitachi, ...) giving up their RISC products to embrace Itanium, which was widely viewed as the inevitable winner given the backing of HP and Intel.

Since AMD was not invited to participate, it had no choice but to develop its own 64-bit ISA. The AMD64 ISA extended 80x86 to 64-bit addresses by widening all registers plus adding a few more registers to help compilers, following the precedent of the 80386.

Alas, the Itanium was an "EPIC" failure! The first version did not ship until 2001. The performance difficulties included scheduling VLIW code for unpredictable branches and for variable memory latency due to unpredictable cache misses, and for the explosion in code size that increased instruction cache misses due to wider instructions and their low utilization. As the Stanford computer scientist Donald Knuth summarized [10]:

*"The Itanium approach...was supposed to be so terrific—until it turned out that the wished-for compilers were basically impossible to write."*

Considering the billions invested and the extensive promotion, the chip delays and under performance caused online forums to rechristen it the *Itanic* after the infamous "unsinkable" Titanic"!

The VLIW Itanium architecture was heralded as the 64-bit address successor to the 80x86, but instead it was AMD64 that Intel was eventually forced to adopt. (Intel ended the Itanium line in May 2017.) Ironically, the emergency replacement 8086 ISA and its descendants beat both of Intel's anointed and trumpeted successors: the iAPX 432 and the Itanium.

VLIW failed for general-purpose computing, but found a home in Digital Signal Processors (DSPs), which avoid three of the weaknesses of VLIW: DSP programs are small so code size matters less; its branches are usually very predictable; and the hardware provides program-controlled memories instead of caches, which offers fixed memory latency.

## 7. Domain-Specific Architectures

Architects rode Moore's Law and Dennard Scaling to turn increased resources into performance. They designed sophisticated processors and memory hierarchies that exploited parallelism between instructions without the knowledge of the programmer. Architects eventually ran out of techniques for instruction-level parallelism that could be exploited efficiently. The end of

Dennard scaling and the lack of greater (efficient) instruction-level parallelism in 2004 forced the industry to switch from a single energy-hogging processor per microprocessor to multiple efficient processors or *cores* per chip.

A law that is as true today as when Gene Amdahl stated it in 1967, demonstrates the diminishing returns to increasing the number of processors: Amdahl's Law states that the sequential part of the task limits the theoretical speedup from parallelism; if ⅛ of the task is serial, the maximum speedup is 8 even if one adds 100 processors, and the rest of the task is easily parallel.

Figure 1.4.3 shows how the impact of these three laws on processor performance for the past 40 years. If the trends continue, single-program performance using standard benchmarks will not double for 20 years! At the present state-of-the-art,

- transistors are not getting much better (due to the ending of Moore's Law),
- the peak power per mm$^2$ of chip is increasing (due to the end of Dennard scaling), but the power budget per chip is limited (due to electromigration, mechanical and thermal limits), and
- we have already played the multicore card (limited by Amdahl's Law).

In view of these inevitable limitation, architects now widely believe that the only path left for major improvements in performance-cost-energy is *domain-specific architectures* (*DSAs*).

## 8. An Example DSA: the Google TPU

A trailblazing example of DSA concept is the Google *Tensor Processing Unit* (*TPU*), first deployed in 2015, which serves billions of people [11]. It runs *deep neural network* (*DNN*) inference 15-to-30 times faster with 30-to-80 times better energy efficiency than contemporary CPUs and GPUs in similar semiconductor technologies.

The block diagram of the TPU in Figure 1.4.4 shows that the host sends instructions over the PCIe bus to an instruction buffer. The internal blocks are typically connected together by 256-byte-wide paths. Starting in the upper-right corner, the *Matrix Multiply Unit* is the heart of the TPU. It contains 256×256 multiply-accumulators (*MACs*) that can perform 8-bit multiply-and-adds on integers. The 16-bit products are collected in the 4MiB of 32-bit *Accumulators* below the matrix unit. The 4MiB represents 4096, 256-element, 32-bit accumulators. The matrix unit produces one 256-element partial sum per clock cycle.

The weights for the matrix unit are staged through an on-chip Weight FIFO that reads from an off-chip 8GiB DRAM called *Weight Memory*. The weight FIFO holds up to four 256×256 tiles of 8-bit weights. The intermediate results are held in the 24MiB on-chip *Unified Buffer*, which serves as inputs to the Matrix Unit. A programmable DMA controller transfers data to-or-from CPU Host memory and the Unified Buffer. To function dependably at Google scale, the TPUs internal and external memories have error detection and correction hardware.

The philosophy of the TPU microarchitecture is to keep the large matrix unit busy. Toward that end, the instruction that reads the weights follows the *decoupled-access/execute* philosophy [12], in that it can complete after sending its address, but before the weight is fetched from Weight Memory. The matrix unit will stall if the input activation or weight data is not ready.

Since readout of a large SRAM uses much more power than arithmetic, the matrix unit uses *systolic execution* [13] to save energy by reducing reads and writes of the Unified Buffer. It relies on data from two directions arriving at cells in an array at regular intervals where they are combined. A given 256-element multiply-accumulate operation moves through the matrix as a diagonal wave front. The weights are preloaded, and take effect with the advancing wave alongside the first data of a new block. Control and data are pipelined to give the illusion that the 256 inputs are read at once, and that they instantly update one location of each of all 256 accumulators.

The TPU can be compared to two large chips: an 18-core Intel Haswell CPU (662 mm$^2$) and the Nvidia Kepler K80 GPU (561 mm$^2$). The TPU is about half the area of the Haswell or Kepler, and half the power, yet packs 25 times the MACs (65,536 8-bit vs. 2,496 32-bit), and 3.5 times the on-chip memory (28MiB vs. 8MiB) as the GPUs.

Figure 1.4.5 illustrates Roofline Performance models for six neural network applications that represent 95% of the TPUs datacenter inference workload in 2016. The Roofline Performance model [14] for high-performance computer modelling illustrates the effectiveness of the applications. This simple visual model offers insights into the causes of performance bottlenecks. The peak computation rate forms the "flat" part of the roofline, and memory bandwidth is the "slanted" part of the roofline.

The six DNN applications are generally further below their ceilings for Haswell CPU and K80 GPU than is the case for the TPU. Response time is the reason! Many of these DNN applications are parts of end-user-facing services. For example, the 99th-percentile response-time limit of one application was 7ms. Haswell and the K80 run at just 42% and 37%, respectively, of the highest throughput achievable if the response-time limit is relaxed. These bounds affect the TPU as well, but at 80%, it is operating much closer to its highest application throughput. On average, the TPU is 29 and 15 times faster than the CPU and GPU.

In datacenters, cost-performance trumps mere performance. Since Google does not disclose its costs, but power is correlated with total cost of ownership, Figure 1.4.6 plots performance/Watt, showing GPU vs. CPU (blue), TPU vs. CPU (red), and TPU vs. GPU (orange). Quantitatively, the relative performance/Watt (ignoring host power) is 83 for TPU vs. CPU, and 29 for TPU vs. GPU. Figure 1.4.6 also shows a hypothetical TPU (TPU') using the same GDDR5 memory as in the GPU. The relative performance/Watt (ignoring host power) of TPU' leaps to an amazing 196× over the Haswell CPU (green bar), and 68× over the K80 GPU (purple bar).

But, more importantly, Google's Tensor Processing Unit for deep-neural-network inference has been deployed successfully in the cloud since 2015, and is used regularly by billions of people. Seven factors explain its energy-performance advantages, given in priority order:

1. The TPU has 1 very large, 2-dimensional multiply units, while the CPU and GPU have 18 and 13 smaller, 1-dimensional multiply units. The matrix multiplies inherent to DNNs benefit from 2-dimensional hardware.
2. The TPUs DNN inference applications uses 8-bit integers rather than 32-bit floating point to improve efficiency of computation, memory bandwidth, and memory capacity.
3. The 2-dimensional organization enables systolic arrays, which reduce register accesses and energy.
4. The TPU drops features required by CPUs and GPUs that DNNs do not use, which makes the TPU cheaper, saves energy, and allows transistors to be repurposed for domain-specific optimizations.
5. The TPU has 1 program thread while the K80 has 13 and the CPU has 18. A single thread makes it easier to stay within a fixed latency limit that the neural network applications demand, as well as saving energy.
6. The TPU has enough flexibility to implement the DNNs of 2017, as well as the lesser demands of 2013.
7. The original production applications were written using TensorFlow, making them easy to port to the TPU with high-performance, rather than requiring rewriting on the very different TPU hardware.

## 9. RISC-V

Despite the promise of DSAs, we still need general-purpose processors to run the programs that are not domain specific, such as operating systems, user interfaces, compilers, and so on. Amazingly, 30 years after the first commercial RISC architectures hit the market, the consensus is still that RISC is the best ISA style for general-purpose processors. No one has proposed a new CISC architecture since 1985, nor a new general-purpose VLIW since 2000.

This ISA consensus plus the move to DSAs have led to a new take on ISAs, called *RISC-V* [15]. (It is pronounced "RISC five" since it is the fifth RISC architecture from UC Berkeley.) RISC-V is unconventional not only because it is a recent ISA (born this decade when most alternatives date from the 1970s or 1980s), but also because it is an *open* ISA. Unlike practically all prior ISAs, its future is free from the decisions and ultimate fate of a single corporation. It belongs instead to an open nonprofit foundation (riscv.org) with 100 members.

The goal of the RISC-V Foundation is to maintain the stability of RISC-V, evolve it slowly and carefully for technological requirements, and to try to make it as popular for processors as Linux is for operating systems. RISC-V benefits from starting 25 years later than other popular ISAs, which allowed its architects to borrow the good ideas but to not repeat the mistakes of the past [15].

Keeping with its heritage, RISC-V is a minimalist ISA; in fact, the base ISA is remarkably similar to its great-great-grandparent RISC-I [9]. One indication of complexity is the size of the documentation. The ISA manual for x86-32 is 2198 pages or 2,186,259 words [16]. The RISC-V equivalents are 236 pages and 76,702 words [17]. If one read manuals as a (terribly boring) full-time job (8 hours a day for 5 days a week), it would take a month for one pass over the x86-32 but less than a day for the RISC-V. Using this common-sense metric, RISC-V is 1/30th the complexity.

Beyond being minimalist open and recent, RISC-V is unusual since, again unlike almost all prior ISAs, it is *modular*. At the core is a base ISA that runs a full software stack (operating system, libraries, debuggers, compilers). The base is frozen and will never change, which gives compiler writers, operating system developers, and assembly language programmers a stable target. The modularity comes from optional standard extensions that hardware can include or not, depending on the needs of the application. Example optional extensions are multiply and divide, single- and double-precision floating-point arithmetic, atomic instructions, compact instruction encoding, and vector instructions [18]. Even including all optional extensions, a summary of the minimalist RISC-V ISA fits into only two pages [19].

To achieve the software-desirable goal of a single ISA that works from the smallest to the largest computers, it must lead to efficient designs both for edge devices and the cloud. To empower large-scale computers, RISC-V offers 64-bit, as well as 32-bit address versions. Minimalism and modularity enable small and low energy implementations of RISC-V, which helps embedded applications. While some argue that ISA complexity doesn't matter for high-end processors, it does matter for low-cost applications, which the lack of success of the 80x86 illustrates. A universal ISA by definition must work well everywhere.

A modern ISA must also reserve opcode space to support the closest possible coupling between general-purpose cores and domain-specific accelerators. In the 1970s and 1980s, when Moore's Law was in full force, there was little thought of saving opcode space for future accelerators. Architects instead valued larger address and immediate fields to reduce the number of instructions executed per program. RISC-V enables DSAs simply by reserving opcodes for custom accelerators.

An open ISA also enables sharing of implementations of RISC-V either for free, or for profit by any organization. Competition, a free market, and open implementations may lower costs and increase innovation, similar to the benefits of open-source software. Open designs also reduce the odds of unwanted malicious secrets being hidden in a processor.

## 10. Conclusion

For at least the past decade, computer architecture researchers have been publishing innovations based on simulations using limited benchmarks claiming improvements for general-purpose processors *of 10 percent or less,* while the DSAs like the TPU reports gains for a domain-specific architecture deployed in real hardware running genuine production applications *of more than a factor of 10* [20]. Order-of-magnitude differences between commercial products are rare in computer architecture, which explains architects' enthusiasm for DSAs. DSAs need ISA support, which RISC-V offers.

The goal of RISC-V is to be effective for all computing devices, from the smallest to the fastest; to be modular to allow tailoring of implementations to the needs of the application; to be long-lived by having a non-profit foundation as its owner that evolves it slowly in response to technology change; and to preserve opcode space to support DSAs.

Ironically, the end of Dennard scaling and Moore's Law may rejuvenate computer architecture research and development, as advances must now come from innovation visible in ISAs. DSAs and RISC-V may play leading roles in this renaissance!

*References:*

[1] Wilkes, Maurice, William Renwick, David Wheeler, "The Design of the Control Unit of An Electronic Digital Computer", *Proceedings of the IEE-Part B: Radio and Electronic Engineering* 105:20, pp.121-128, 1958.

[2] Hemenway, Jack, Robert Grappel, "Understand the Newest Processor to Avoid Future Shock", *Electronic Design News*, pp. 129-36, April 29, 1981.

[3] Emer, Joel, Douglas Clark, "A Characterization of Processor Performance in the VAX-11/780", *Proc. International Symposium on Computer Architecture*, pp, 301-310, 1984.

[4] Patterson, David, David Ditzel, "The Case for the Reduced Instruction Set Computer", *ACM SIGARCH Computer Architecture News* 8, no. 6, pp. 25-33, 1980.

[5] Bhandarkar, Dileep, Douglas Clark, "Performance from Architecture: Comparing a RISC and a CISC with Similar Hardware Organization", In *Proc. Architectural Support for Prog. Languages and Operating Systems Symposium*, pp. 310-319, 1991.

[6] Furber, Steve. "Microprocessors: The Engines of the Digital Age." *Proc. Royal Society Series A* 473:2199. The Royal Society, 2017.

[7] "Chip Hall of Fame: Sun Microsystems SPARC Processor", https://spectrum.ieee.org/tech-history/silicon-revolution/chip-hall-of-fame-sun-microsystems-sparc-processor, IEEE Spectrum June 2017.

[8] Reddi, Vijay, "A Decade of Mobile Computing", *Computer Architecture Today*, July 21, 2017.

[9] Patterson, David, "How Close is RISC-V to RISC-I?", *ASPIRE Blog*, June 19, 2017

[10] Knuth, Donald E., Andrew Binstock. Interview with Donald Knuth. *InformIT,* 04-01, 2010

[11] Jouppi, Norman P., Cliff Young, Nishant Patil, David Patterson, *et al*, "In-Datacenter Performance Analysis of a Tensor Processing Unit", *Proc. International Symposium on Computer Architecture*, 2017.

[12] Smith, James, "Decoupled Access/Execute Computer Architectures", *Proc. International Symposium on Computer Architecture*, 1982

[13] Kung, H.T., Charles Leiserson. "Algorithms for VLSI processor Arrays", *Introduction to VLSI systems*, 1980.

[14] Williams, Samuel, Andrew Waterman, David Patterson, "Roofline: An Insightful Visual Performance Model for Multicore Architectures." *Communications of the ACM* 52, no. 4, pp. 65-76, 2009

[15] Patterson, David, "Reduced Instruction Set Computers Then and Now", *IEEE Computer*, December 2017.

[16] Baumann, Andrew, "Hardware is the New Software", *Proc. 16th Workshop on Hot Topics in Operating Systems*, pp. 132–137, 2017.

[17] Waterman, Andrew, Krste Asanovic, editors, "*The RISC-V Instruction Set Manual, Volume I: User-Level ISA*, Version 2.2", https://riscv.org/specifications/. "*The RISC-V Instruction Set Manual Volume II: Privileged Architecture* Version 1.10", https://riscv.org/specifications/privileged-isa/ May 2017.

[18] Patterson, David, Andrew Waterman, "SIMD Considered Harmful", *Computer Architecture Today*, September 18, 2017

[19] Patterson, David and Andrew Waterman, *The RISC-V Reader: An Open Architecture Atlas"*, Strawberry Canyon, 1st edition, 200 pages, www.riscvbook.com, 2017.

[20] Hennessy, John L., David A. Patterson, "*Computer Architecture: A Quantitative Approach"*, 6th edition, Elsevier, 2018.

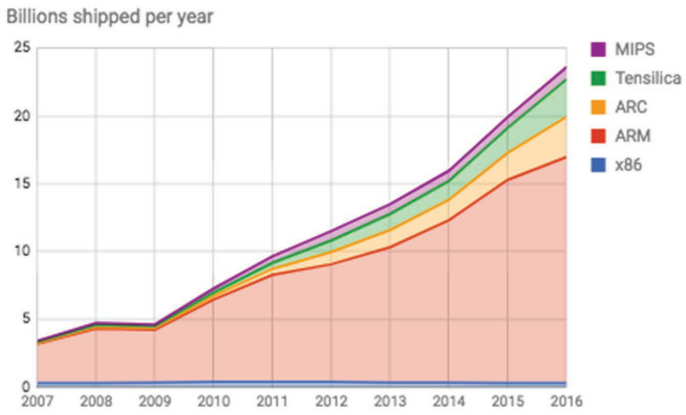


Figure 1.4.1: Billions of chips shipped 2007 to 2016; 99% of 2016 chips are RISC [9].

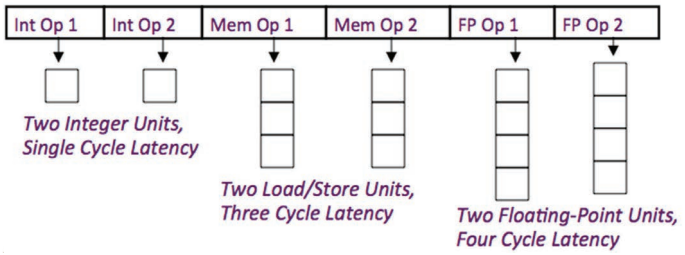


Figure 1.4 2: Hypothetical VLIW Instruction.

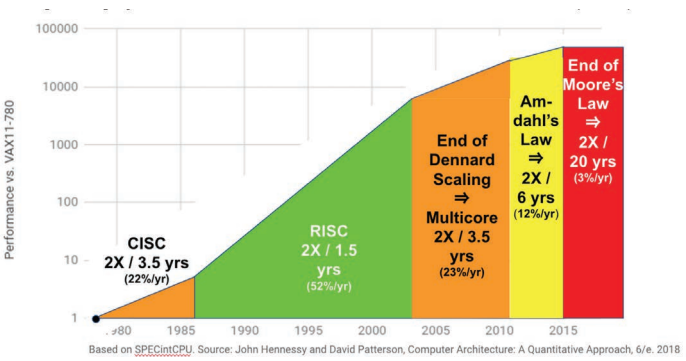


Figure 1.4.3: Average performance gain for a single program over time versus VAX 11/780 using SPECintCPU [20].

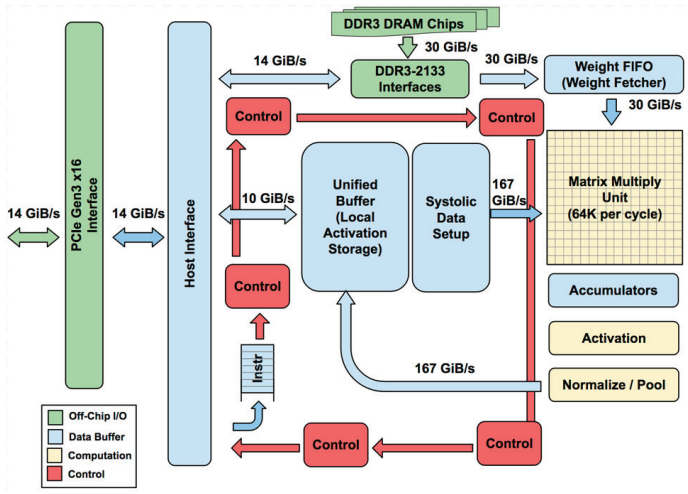


Figure 1.4.4: Block diagram of the TPU [11].

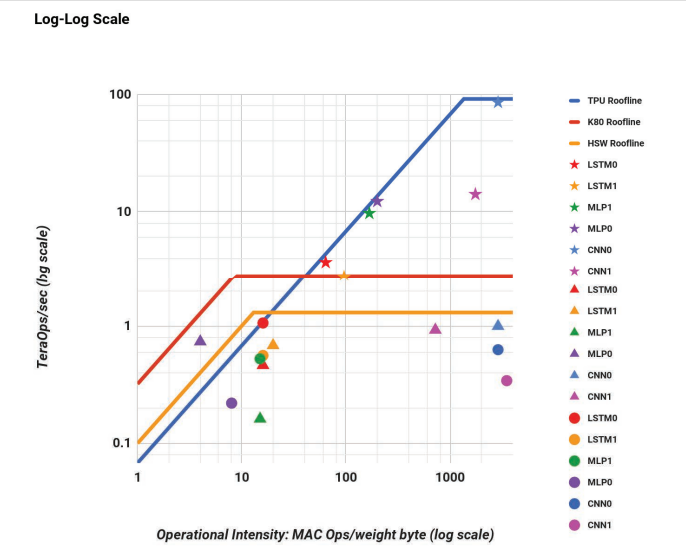


Figure 1.4.5: Roofline Performance Model of the CPU, GPU, and TPU [11].

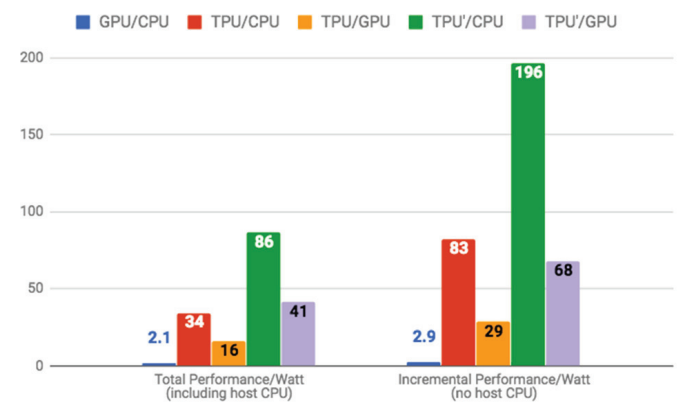


Figure 1.4.6: Relative Performance per Watt of a CPU, GPU, the original TPU, and a revised TPU' [11].