The ontogeny of RISC

Steve Steinberg

1. Introduction

> "It is a consistent theme that each generation of computers obsoletes the performance evaluation techniques of the prior generation."
>
> *Computer Architecture: A Quantitative Approach,* Hennessy & Patterson.

When I read this I thought of Thomas Kuhn, of incommensurable paradigms, and the path to relativism. This traitorous sentence seemed to undermine the thrust of the entire text. How could a quantitative approach help to decide between two competing architectures if no single metric could be applied to both? Progress, from generation to generation, could not be proven monotonic. I imagined a race with as many finishing lines as competitors; I imagined a world with arbitrary and constantly changing units. I recalled Kuhn and Feyerabend's argument that scientists who subscribe to different theories speak different and fundamentally untranslatable languages: "mass" does not — can not — mean the same thing to an Einsteinian physicist as it does to a Newtonian. This, now cast in terms of my own field, seemed unassailable, but its implication, that controversial design decisions are settled by happenstance and politics rather than with quantitative evidence, did not.

Although I was unconvinced by the textbook's examples and problem sets where simple arithmetic magically determined all decisions, neither could I believe that underdetermined design decisions could consistently lead to (what felt like) rapid improvements in computer performance. This uncertainty nagged at me throughout my course that semester; I wanted to resolve the question and restore my faith in the implacable and deterministic trajectory of computer science. I decided it would be possible to trace a debate within computer architecture and see how facts were marshaled, evidence gathered, sides chosen; and from this, determine how scientists with different metrics, different world-views, were able to communicate across the chasm and reach consensus.

Philosophers of science have many historical examples to test their theories against; I had only a few to choose from but was compensated by the abundance and easy availability of documentation. I chose to examine RISC — Reduced Instruction Set Computing — because it marked a clear shift in computer architecture, was recent, has reached near-consensus, and was largely developed by the textbook's authors. The idea behind RISC, that a processor with only a minimal set of simple instructions could lead to better performance than a processor that included instructions for complex high-level tasks, was first studied at IBM in the mid-1970's. Little came of this work, but the rumors that surfaced were tantalizing enough to convince first Patterson at UC Berkeley and then Hennessy at Stanford to initiate their own projects during the early 1980's to build prototype RISC processors. While scientists at the two largest microprocessor vendors, Motorola and Intel, steadfastly argued against RISC, by 1990 sixty percent of all workstations were using RISC microprocessors.

My first instinct, to obtain all relevant papers and then tabulate their arguments to determine how RISC developed, was quickly exposed as impractical. Not only was the body of literature enormous, but, more importantly, arguments were often implicit in the text: what questions were asked said more than the endlessly qualified answers, things not said were as relevant as what was. Tabulation appeared intractable. I became convinced I needed a tool — an automated tool — which could handle the papers in bulk, which could sort, summarize, and spit out a result (perhaps some sort of neural net, trained to sniff out arguments, I thought). I wanted to be empirical, quantitative — to apply

the same scientific rigor I was searching for.

Groping through related research I eventually found the technique of co-word analysis. It promised not a single definitive number, but something almost as appealing: a map of the field, a visualization of the mechanisms, a battle chart of the debate. If I could *see* the dynamics of a technical debate, I thought, perhaps I could understand them.

> "Take the heterogeneous. Homogenize it to two dimensions. Juxtapose this homogeneity in a forceful textual structure. Send the structure out to act on others who will be likewise homogenized. This is the secret of the laboratory as an agent of change, a secret that involves building complex but enforceable worlds on paper."
>
> *Laboratories and Texts*, John Law.

2. Method[1]

Co-word analysis was developed by proponents of social constructivism and the technique is as theory-laden as any scientific tool. Co-word analysis implicitly depends upon three beliefs. First, that technology is hopelessly heterogeneous; that there is no way to separate the scientific from the social and political. The scientist is an entrepreneur who uses whatever resources are available, from social influence to raw data, to reach some profit (whether in prestige or funding). Second, that technology is an emergent phenomena without any autonomously set trajectory. Rather, it is shaped by a web of social and technical forces every step of the way. Third, that the primary tool of scientists, the weapon used to exert force on the socio-technical web, is the text. A successful scientific paper is one which links entities in new ways and exerts enough force to convince other scientists to believe in this linkage. (The constructivists refer to this process of imposing a structure upon others as *interessement*). Of these beliefs, I find only the second, that technology is not path-dependent, less than convincing, and my interpretation of results will reflect this.

From the above tenets our method of analysis becomes clear: the technique must examine scientific papers in aggregate and the objective is to expose attempts at interessement. We can think of a scientific paper as a network of important words, the network defines the author's world-view. For example, Patterson and Ditzel's 1980 paper, "The Case for RISC," was one of the first to link "RISC" with "pipelining." It is a sign of successful interessment that five years later these architectural concepts were routinely linked together. We can extend this type of analysis so that it is quantitative and works on collections of papers as follows: obtain all the abstracts for papers on RISC, choose a set of keywords k, and then construct a square, symmetric co-word matrix c where c_{ij} = {number of abstracts which contain keywords k_i and k_j}. When this is graphed so that each keyword is positioned near the keywords it is most commonly associated with, we can begin to make out the dynamics of the struggle for the hearts and minds of scientists.

Co-word analysis sounds at first both unconvincing and tantalizing, possible problems and improvements immediately come to mind. But to focus on method at this date is neither interesting or necessary. For too long those interested in scientometrics have concentrated on validating and fine-tuning quantitative methods while largely ignoring the significance of their results. I wish to use the tool, not extend or justify it. I have implemented the core subset of the co-word method described above without debating over the unproved or contentious improvements which have been proposed, and in this paper I largely ignore implementation details. I give the following review of my methodology only as proof of my seriousness.

After collecting all abstracts with the keyword "RISC" for the years 1980-1993 from the Inspec database, I set about determining the most important words. I did this by filtering out the 200 most common English words from the abstracts and then ranking the remaining words by frequency. Lists of the top 300 words were then given to three experts in the field who chose those words they felt were most important (most central to RISC research). This resulted in a total of 76 keywords. Finally, those words which were synonyms were grouped together to form 45 keyword "buckets." From these I constructed a similarity matrix using the inclusion index, $I_{ij} = c_{ij}/\min(c_i, c_j)$. I chose the non-symmetrical inclusion index because it emphasizes the hierarchy of connections between less-frequently occurring words ("mediator words") and the central poles of research. This matrix

was then fed to ALSCAL, a multi-dimensional scaling routine which translates a similarity matrix into Euclidean distances, and the resulting coordinates were used to create the map. Keywords were displayed in a font-size proportional to the word's frequency (words so infrequent as to be illegible were eliminated) and lines were drawn between strongly-linked keywords.

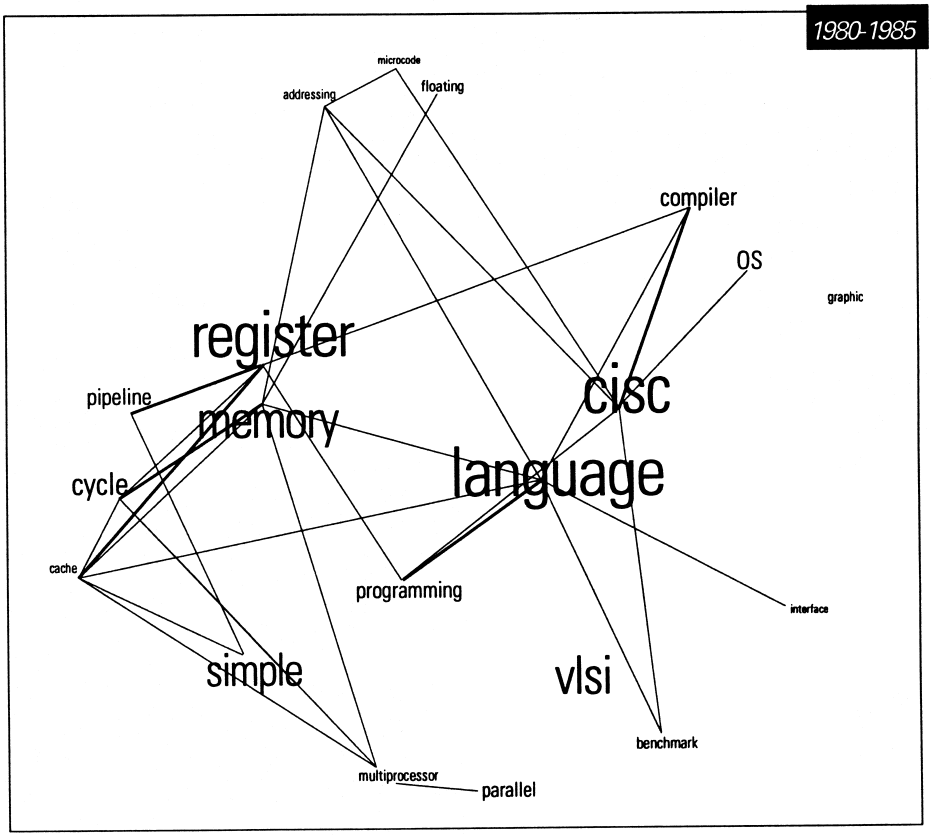
> "Once the technical means of control have reached a certain size, a certain degree of *being connected* one to another, the chances for freedom are over for good. The word has ceased to have meaning."
>
> *Gravity's Rainbow*, Thomas Pynchon.

3. Results

A topographic map appears to the first-time user as a meaningless blur of lines and letters. With experience, ravines can be made out and trails evaluated; the map defines the terrain with a precision undiminished by the abstractness of its representation. Co-word maps are similar, but more treacherous. It is easy to find meaning where there is only noise, or to imagine patterns in randomness. I guard against these dangers by charting familiar territory and by constantly consulting the raw abstracts the map was generated from. But we can not expect co-word maps to give up answers without a fight; we must provide both the narrative and interpretation. In the following guided tour of five chronologically ordered co-word maps, I hug the shore of empiricism while still reading (between) the lines.

A. Schism (1980-1985)

1980. Intel's 8086 has just been released, 64k DRAMs are now available, VLSI technology is becoming well established, the VAX 11/780 is state-of-the-art, and the first papers to explicitly examine and define RISC are beginning to appear. Because the design philosophy of RISC was so opposed to the existing research tradition, every paper was as much a defense and justification as it was a report of research done. The initial battle lines of the debate were drawn by Patterson and Ditzel's 1980 paper, "The Case for RISC", and these lines show-up clearly in the period's co-word map. The map contains two main clusters: on the left, surrounding register and memory, are those architectural



terms which uniquely define RISC; on the right, orbiting about language, are the terms of contention—the words hurled between the RISC and CISC camps.

Language is the most frequent term on the map, and the one which most clearly encapsulates the differences between RISC and CISC. CISC proponents believed that a processor's instruction set should closely correspond to high-level languages such as FORTRAN and COBOL. This, they argued, made compiler writing a simple matter of one-to-one translation, made programming directly in assembly manageable, and, because the instructions were implemented in hardware, lead to better performance. RISC proponents, however, argued that compiler writers usually relied on just a few simple and well-understood instructions, that no one programmed in assembly anymore, and that high-level instructions often ran slower than an equivalent sequence of simple instructions due to the complexity of implementation. This debate is shown in the co-word map with the connections between language, CISC, compiler and programming.

The lack of support for high-level languages was largely used as an attack against RISC. The main argument in favor of RISC is represented by the term VLSI, still located on the right-side of the map but below and separate from language. Both the capabilities and limitations of VLSI, the technology which allows complex circuits to fit on a single chip, were enlisted in support for RISC's minimalist philosophy. In Patterson's words: "If VLSI technology continues to at least double chip density roughly every two years, a design that takes only two years to design and debug can potentially use a much superior technology and hence be more effective than a design that takes four years." This attempt to tie RISC to the inexorable progress of VLSI technology is a classic example of hetrogeneous engineering and is still sometimes held up as the deciding argument. Yet it makes a number of implicit assumptions—for example, that CISC development takes longer than the discrete time between VLSI generations—which have not held up in practice.

On the left-side of the co-word map are the terms and connections which define RISC and delineate its world-view. At the cluster's center lies registers and memory, terms which reflect the paradigmatic shift from a very large set of addressing modes to the load-store architecture of RISC. While not a new idea (the original Cray, for example, was load-store), it marked an abrupt turn-around from contemporary trends. Registers were also closely associated with RISC because both the Berkeley and Stanford prototypes supported what were, for the time, very large register sets, and this led to new research on the register-allocation problem. It also led to a further incomensurability according to some critics, who claimed that RISC's performance was due more to these large register sets than any advantage gained through simplicity. Other defining terms in this cluster include cycle (cycles-per-instruction, a largely irrelevant metric for CISC machines became an overriding one for RISC), simple (an objective made no less desirable by its lack of a technically meaningful definition), and pipelining (a technique which, while not inherently part of RISC, was initially seen as nearly impossible to apply to CISC processors).

While a detailed examination of the co-word map provides reassurance that it reflects the real issues and structure of the research field, the map's meaning is clearest—and most telling—when examined at a higher-level. The dense, tightly-connected web indicates that RISC began as a narrow and cohesive field, one distinct from the rest of computer architecture and not yet divided by further specialization. The bi-polar nature of the map reflects the gap between researchers, the interregnum between paradigms. Around one pole lies the battle lines of the debate, around the other are the terms which define RISC. Now, by mapping subsequent research in two-year snapshots, we can see how this debate unfolded and how the definition evolved.

B. To market (1986-1987)

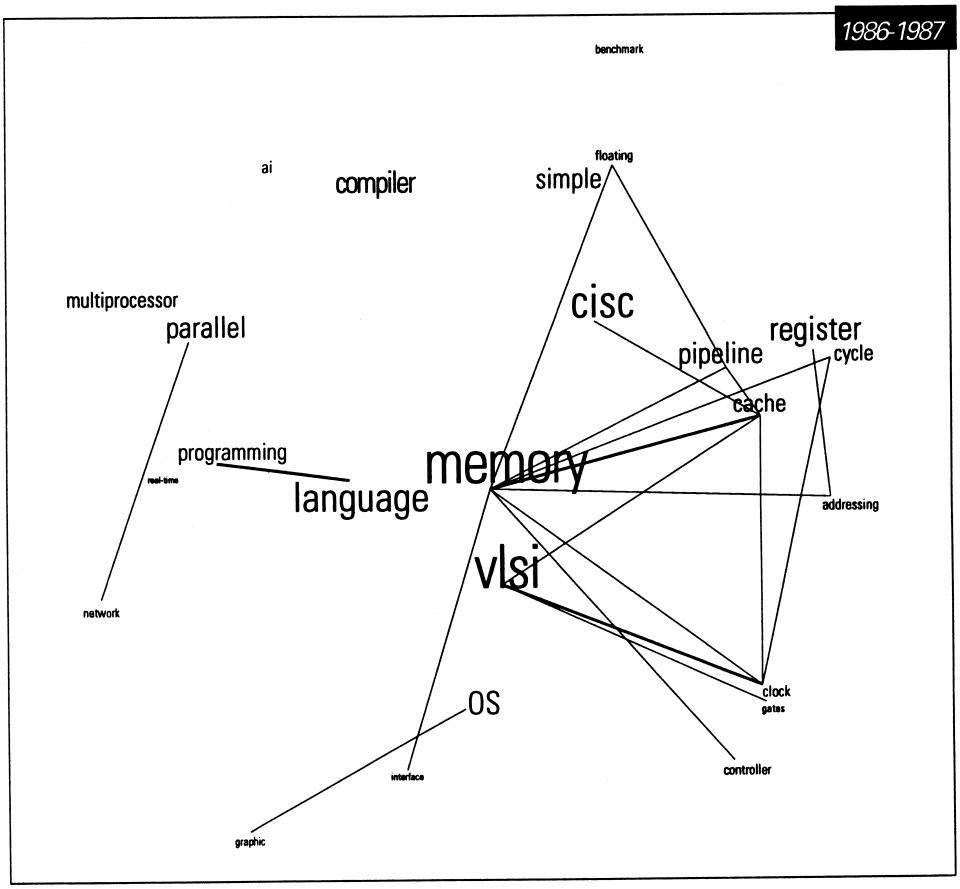
Sun introduced the SPARC, the first commercially important RISC microprocessor, in 1986. In explaining their decision to use RISC, Sun cited the argument that it would allow for extremely rapid design cycles (this, however, has not been historically borne out by Sun who has been notoriously slow in introducing new generations of their processor). The SPARC was quickly followed by the MIPS processor (from the eponymous workstation vendor co-founded by Hennessy), the clunky RT PC from IBM, and the first draft

of Hewlett-Packard's PA-RISC.

RISC was no longer just a paper tiger, it was a tangible product with money invested in ensuring its success, and this transition is plainly apparent in the period's co-word map. The map, formerly bi-polar, now appears as if it had been tilted to one side. The cluster representing the debate is in disarray, while the cluster concerned with defining and implementing RISC has grown to dominate the map. VLSI—the technology of implementation—has become larger and more central. Compiler and language, terms which pointed to perceived flaws, have become separated and less frequent.

The cluster of terms which define RISC remains tightly connected (reflecting the fields continued cohesiveness), with the most noticeable change being cache's increased prominence. (While proponents spoke early on of how RISC would leave room for on-chip caches, they neglected to mention that RISC's faster cycle times made memory stalls so expensive as to make caches mandatory.) On the left side of the co-word map, clearly separated from the definition cluster, we see various trends in computer science—networks, parallelism, AI—slowly being draw into the web as RISC research began to diffuse through the field.

While the reconfiguration of the co-word map from bi-polar to lop-sided indicates that the high-level language argument had been resolved, the map is frustratingly mute on explaining how. Even through hand-examining the literature I could find no dénouements, no compelling proofs that high-level assembly languages are intrinsically slower. In fact, during this period FORTRAN compilers for the VAX were steadily increasing in performance and they were no longer using only a small subset of available instructions, refuting one of the cornerstones of the RISC argument. But this was an iterative development within the industry and not one that generated many research papers. The RISC research community appears to have answered the high-level argument by ignoring it, by deciding to go ahead and build machines and let the benchmarks sort it out. However, the belief that two very different architectures can be quantitatively compared is refuted by the quote that begins this paper. While by 1988 it was clear that industry and academia



had chosen two separate and diverging paths, the reasons—and possible outcomes—remained murky.

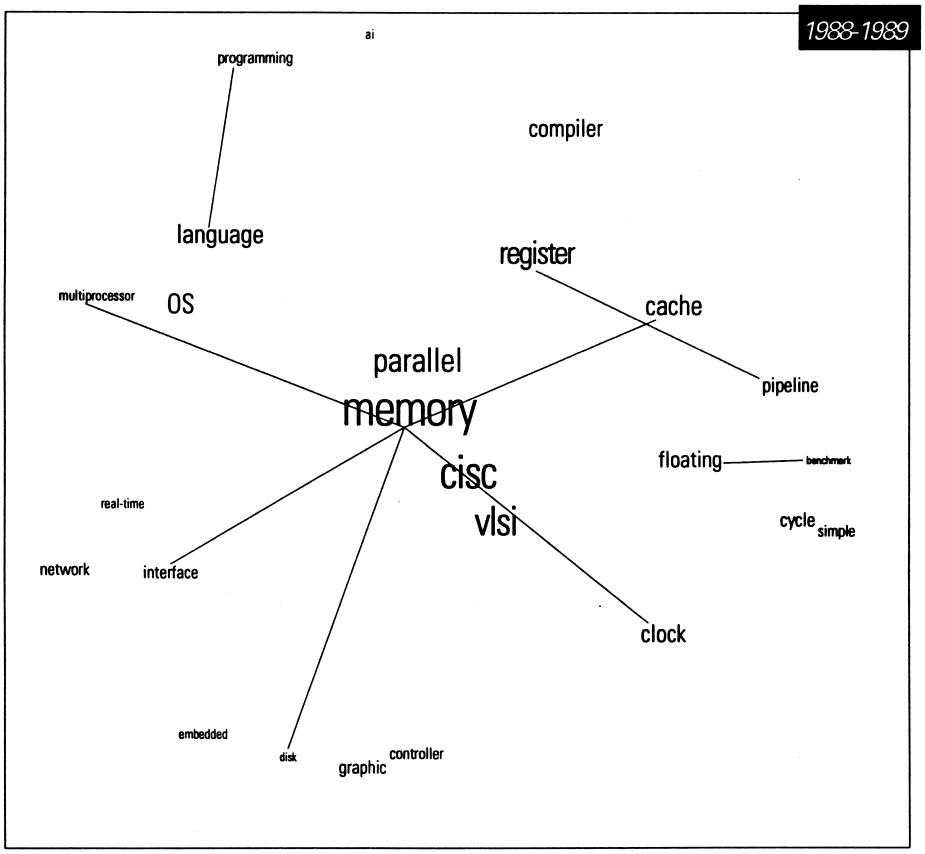
C. Embedded pursuits (1988-1989)

The co-word map for 1988-1989 shows a field settling down and reaching some sort of technical maturity. Terms have dispersed to form small clusters of specialization about the periphery, and the size of terms and their connections have become relatively static. The interesting part of the map is now the large cluster of emerging terms at the bottom left. These terms—real-time, interface, graphic, controller, embedded—give warning of the first beachhead in the RISC vs. CISC battle.

RISC processors were originally designed for use in workstations, a category of computers which was largely created by the same companies who pioneered RISC (e.g., Sun and MIPS). Although a few workstations with processors from the x86 and 680x0 and families were developed, it represented a tiny percentage of Intel's and Motorola's sales. To give some sort of context, in 1989 about 200,00 workstations were sold. In that same year, a few million processors were sold for use in PCs, and a billion or so processors were sold for embedded applications. So, although RISC now had the majority of the workstation market, it was a very small pond indeed.

That RISC needed to expand and find new markets is obvious, that the embedded marketplace would be chosen first is less so. Embedded applications require reliability more than performance, and they are usually written in assembly language—a tedious process with RISC. However, there were two other aspects in RISC's favor. First, price. Saving even a few cents is critical in mass-market embedded applications, and simple processors require less die area which means greater yield. Second, embedded applications are usually custom-written and highly-specific. Therefore, switching to a new processor would not require the conversion of hundreds of popular applications, as would be the case with PCs. This lack of "network externalities and bandwagon effects" undoubtedly made the market appear more approachable.[3]

Yet, despite the determined push in research indicated by



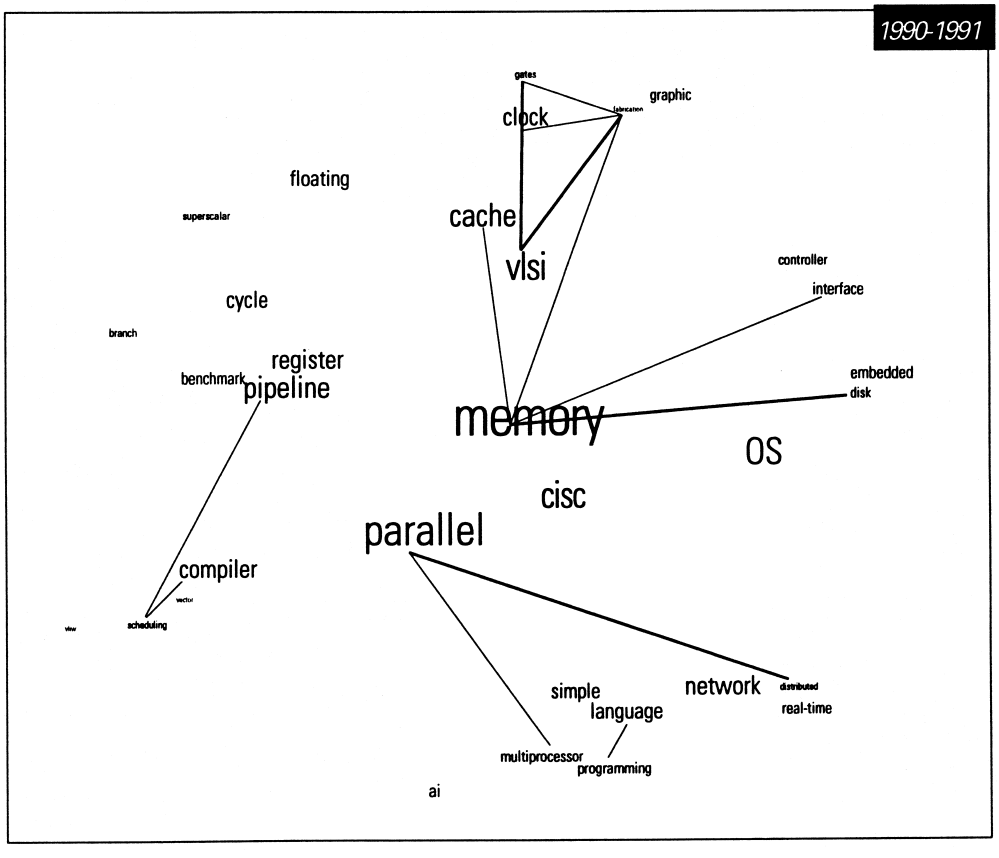
the co-word map, RISC has still not captured the embedded market. Twelve-year-old CISC processors (now available for under a dollar) dominate the market's low-end, while Motorola continues to introduce new generations of the 680x0 for the high-end. Ironically, it has been Intel— last bastion of CISC, creator of the enormously byzantine Pentium — who has had the most success with RISC processors for embedded applications. Intel's i860 RISC processor, introduced in 1989, quickly became popular thanks to rather optimistic benchmarking claims and aggressive pricing. (Clearly the economies of scale is an important reason for Motorola's and Intel's dominance in this market, but the success of ARM, a RISC processor developed by Acorn in England, suggests it is not the only one.) Subsequent co-word maps of RISC show continued interest in embedded applications but little growth. It was not the quick route toward RISC dominance some had hoped.

D. Subsumption (1990-1991)

An invasion of the PC market now appeared imminent due to plunging workstation prices, and this brought the RISC vs. CISC battle (often described in ludicrously inaccurate terms) to the attention of a much larger audience. It was also during this period that two researchers at DEC published a paper that compared the MIPS and the VAX 8700 and ended with the heretical conclusion that RISC offered a fundamental advantage.[4] This paper, although filled with caveats and focusing solely on the poorly-designed VAX, was widely seen as a public surrender by CISC's largest supporter. But underneath this increasingly hyped battle, as marketing departments began to enter the fray in earnest, the co-word map exposes a subtle but important development.

The map for 1990-1991 contains four distinct clusters: at top is a group of implementation terms (clock, VLSI), on the right is the embedded cluster, below is a parallel-processing cluster, and to the left are the traditional RISC terms along with new microprocessor design terms (superscalar, scheduling). It is the latter two clusters, representing macro and micro new technologies, which truely signal the arrival of RISC's success.

The parallel-processing cluster shows that both the uses of



RISC (as building blocks for multiprocessors) and new extensions to RISC (instruction-level parallelism) had become central aspects of research. Similarly, in the left cluster, the traditional RISC terms—register, pipeline, cycle—are now linked with new, emerging microprocessor techniques such as dynamic branch prediction, superscalar techniques, and aggressive scheduling. Yet these techniques have no inherent connection to RISC, in fact, they contradict some of RISC's key tenets. Multiple instruction units lead to a huge increase in complexity and in number of required gates, largely nullifying RISC's cost advantage. And, while complex instructions had originally been derided because their cost-model was difficult for compiler writers to understand, scheduling places an even greater burden on compiler writers.

RISC was now the starting point for research, not a destination. It had become part of the firmament, but made firm by familiarity rather than any provable correctness.

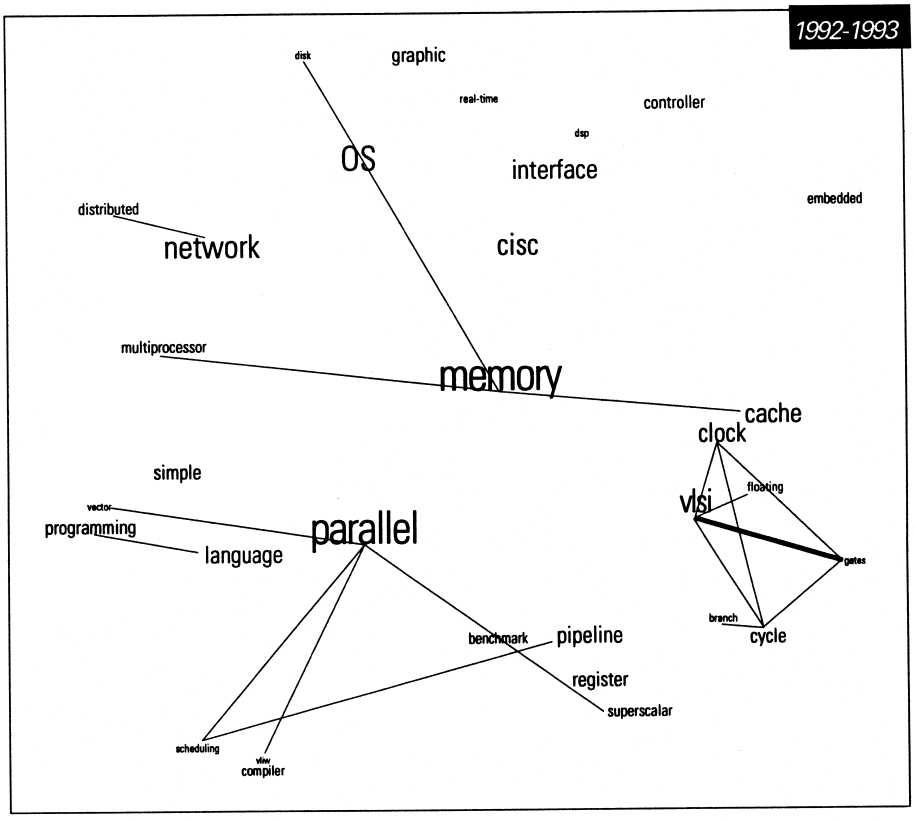
But by 1991 almost every new technology, from superscalar processing to networks, had become routinely, casually, linked to RISC. It was now the starting point for research, not a destination. It had become part of the firmament on which researchers stood, but made firm by familiarity rather than any provable correctness. While RISC had still not won in the marketplace, its near-total hegemony of research was beginning to endanger the symbiotic relationship between research and industry and the influx of new ideas industry depends on. It was during this period that Motorola and IBM began to talk about "second-generation RISC", a concept which seemed to encompass all possible architectural techniques except for a few central RISC ones (like design simplicity), but which offered a syntactical concession to researchers.

The scenario I have sketched is somewhat reminiscent of Planck's assertion that a scientific idea comes to be accepted not by convincing opponents, but by opponents dying off and being replaced by a new generation which has grown up with the idea. The difference with RISC lies not only in the accelerated time-scale, but also in the complex interplay between research and industry. RISC was first developed at IBM in the mid-1970's, was then adopted by researchers (some of whom went off to form a new niché market), and became a convenient base for further research. The resulting developments were then incorporated by the industry monoliths. When seen this way, RISC vs. CISC appears more like an intricate dance or highly-developed eco-system than a war of attrition.

E. Consensus and death (1992-1993)

This final co-word map represents the logical conclusion of RISC's trajectory toward dominance: the map of RISC is now indistinguishable from the total map of computer architecture. Like a Russian doll which contains and is enclosed by its own image, RISC is now a subset of a larger but structurally similar field. At the map's center lies memory, solitary and ponderous in its generality. The keywords which have represented RISC since the beginning are now small and subsumed by parallel (remember that the inclusion index is not symmetrical; it represents a hierarchy between infrequent and more common terms). Other keywords are arranged in clusters that represent standard specialties of computer architecture: implementation to the right, I/O at top, parallelism below. RISC is no longer a distinct field with its own unique connections, it has become the dominant research tradition, it has become what CISC once was: an increasingly challenged status quo.

1994. Motorola and IBM introduce the PowerPC, a microprocessor with a large set of instructions and addressing modes which only a marketing department could describe as RISC. Digital Equipment, whose Alpha 21064 may represent RISC's apotheosis, hints of a new direction for the next generation of Alphas. Similarly, HP and Intel speak of moving away from the RISC-CISC dichotomy and toward VLIW where the goal is to do as much as possible per clock-cycle rather than to decrease the cycle time. RISC is still used as a marketing term; it is new enough to sound important, old enough that most people have heard of it. But, technically, the word simply refers to the status quo; it has ceased to have meaning.



> "Pourquoi n'y aurait-il pas, en quelque sorte, une science nouvelle par object? Une *Mathesis singularis* (et non plus *universalis*)?"
>
> *La Chambre claire,* Ronald Barthes.

4. Conclusion

Like the Kabbalists, I will be accused of constructing theories from random confluences of words. Like the Kabbalists, I find justification in the process. The value of co-word maps is of description and encapsulation rather than explanation. They illustrate the development of RISC and provide the solid, empirical posts a narrative can be built on. Because of them I now understand my error.

If CISC had been pursued intently it could have resulted in performance as good or better than what we now enjoy. However, it would have first required a rapid improvement in compiler technology. RISC, on the other hand, because of its simplicity and comparative ease of fabrication, offered computer architects a convenient test-bed for new research. Therefore, knowing that inactivity leads to death, that one must publish or perish, architects chose an unblocked path. The question of whether RISC or CISC was intrinsically faster was ignored: "room to grow," and not some arbitrary notion of performance, was the determining criteria. This is reflected by the co-word maps in which development terms, rather than comparative terms, quickly came to dominate. In the case of RISC, then, the problem of incommensurable performance metrics did not arise.

Once this is understood, the subsequent developments appear inevitable. Donald MacKenzie has claimed that what appears to be a technological trajectory is really a self-fulfilling prophecy.[4] (We believe computer performance will double every three years, so we invest whatever resources are necessary to make it happen.) Yet, as decisions are made which influence later decisions, an autonomous trajectory begins to take hold. As more resources were devoted to RISC within the architecture community, as other research became subsumed under the RISC label, the choice became irrevocable. The decision could not be undone; it could only be nullified by contrary decisions made further along the arc.

My error, in believing that incommensurability must lead to relativism, was in believing there to be a single path to improved performance, in imagining a divine arc which could not be deviated from. Because there are actually many paths, relatively *ad hoc* methods of theory choice will usually iterate toward increased performance, just as the random permutations of genetic algorithms will usually converge at a global optimum. What we call progress is an emergent property of interaction between distinct and incommensurable theories and objects. A comparison to natural, biological systems is facile, but apt: the self-similar nature of research was graphically shown by the co-word maps in which the structure of RISC grew to resemble that of computer architecture. (I am not the first to say this. A. F. Van Raan has already shown that co-citations of scientific papers form a self-organizing system that can be graphed as a fractal landscape.[5])

Yet, in the end, I am no closer to articulating the algorithm by which one architecture is chosen over another than when I began. My attempt to open the black box did not reveal the well-oiled deterministic machinery or coldly-rational blinking lights I had hoped for. Instead, it exposed a controlled randomness, a few heuristics which seem to work. To call this process emergent is, perhaps, just a new way to black-box the infinite and unknown. But, as tentative and limited as my findings are, they have altered my perception of technology and its progress. I still believe technology to be scientific and empirical, but, now stripped of its omniscience and singularity, it somehow seems less sacred.

[1] This discussion draws heavily on *Mapping the Dynamics of Science and Technology,* ed. Michel Callon, John Law, and Arie Rip.

[2] Khazam, J. and Mowery, D. The commericialization of RISC. *Research Policy,* 23, 1994.

[3] Bhandarkara, D. and Clark, W. Performance from Architecure: Comparing a RISC and a CISC. *ASPLOS '91.*

[4] My thinking on this topic is influenced by, on one hand, D. MacKenzie's *Inventing Accuracy*, and on the other, by N. Rosenberg's *Exploring the black box.*

[5] Van Raan, A. Fractal dimensions of co-citations, *Nature*, 347, 1990.