# SOFTWARE DEMAND, HARDWARE SUPPLY

DO THE DEMANDS OF NEW SOFTWARE OUTPACE DEVELOPMENTS IN HARDWARE? EXPERIMENTS WITH THE BEHAVIOR OF SPEC CPU ON-CHIP CACHES AND DATA COLLECTION FROM A WIDE RANGE OF PROCESSORS OVER TIME ADDRESS THIS QUESTION AND ILLUMINATE TRENDS IN SOFTWARE AND HARDWARE EVOLUTION.

Jesús Alastruey
José Luis Briz
Pablo Ibáñez
Victor Viñals
University of Zaragoza

For nearly 40 years, Moore's law has successfully predicted that the number of IC components will double periodically (every year beginning in 1965, every two years since 1975).[1, 2] This evolution has induced similar growth rates in other technological parameters and related economic indicators. Technology forecasters frequently seek to extrapolate by examining the factors behind Moore's law.[3, 4]

Decreasing the size of transistors and wires lets designers place more functional units and additional memory on chip while increasing the operating frequency. Advances in compilers, instruction set architecture (ISA), and microarchitecture help engineers make the most of instruction-level parallelism (ILP). Software seems to be growing exponentially as well. Programs sometimes demand more operations and greater memory capacity than hardware can provide. As Myhrvold's first law of software states, "Software is a gas. It expands to fit the container it is in."[5]

It's a common belief that software and hardware evolve together and adjust to each other, but there's little quantitative evidence to support this view. This article examines this hypothesis, comparing the hardware-software relationship with that of the producer-consumer balance in the general marketplace. According to this hypothesis, what hardware technology can offer (supply) and what software can request (demand) are intimately related, influencing each other's evolution.

To fully understand the supply-demand relationship, a researcher would have to analyze a broad range of software and hardware components for a substantial period of time, running each program on the proper hardware and harvesting comparable performance measures. We focused instead on a modest but feasible scenario: the evolution of the SPEC CPU benchmarks[6] running on a set of leading high-performance microprocessors for comparable time periods.[7] This leads to a partial understanding of the whole issue. We chose the SPEC CPU suite because we needed a benchmark that covers enough time to mirror general market evolution.

## Gathering data

We gathered information about high-end microprocessors from 1989 to 2005 and identified those representing the appearance of a new microarchitecture. We experimented with all the SPEC CPU program suites developed since the benchmark was first initiated, both CINT (integer benchmarks) and CFP (floating-point benchmarks). On the hardware side, we focused on the number of tran-

Published by the IEEE Computer Society

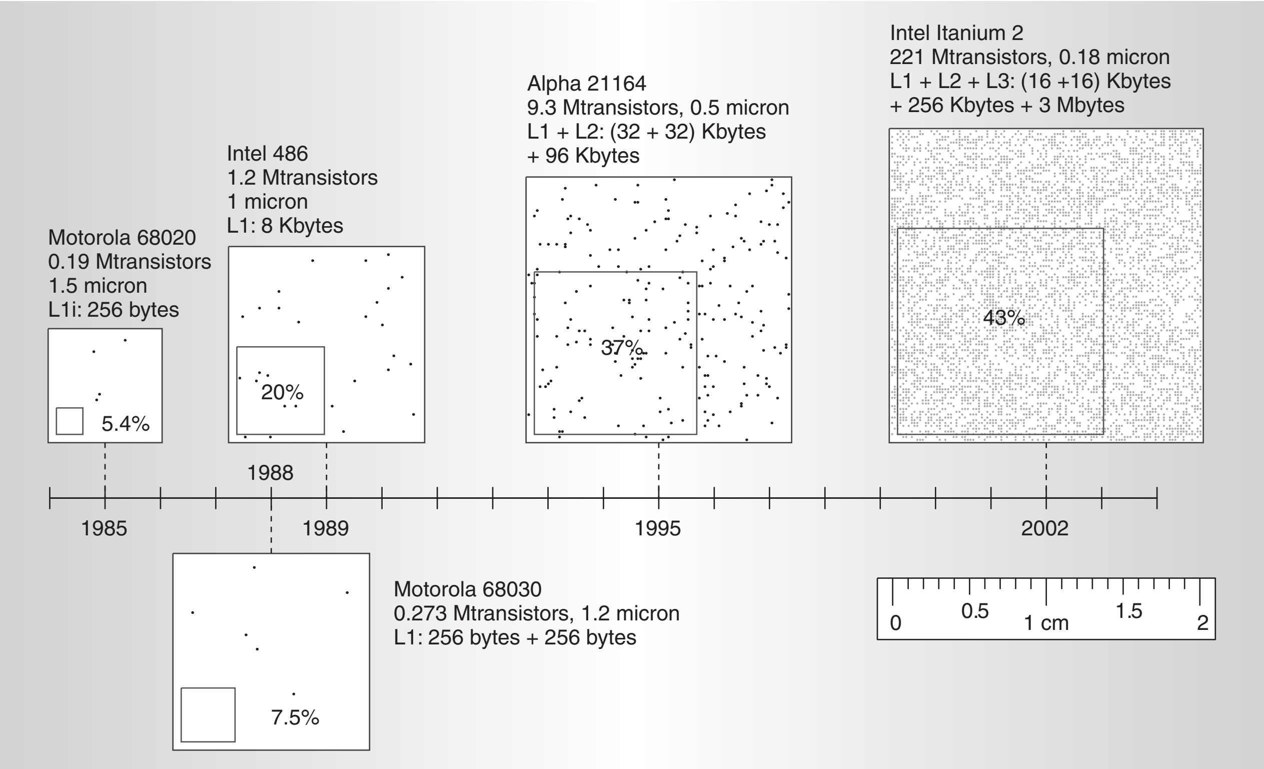

Figure 1. Milestones in the evolution of on-chip caches. Left to right: Motorola 68020, first on-chip cache (instructions only); Motorola 68030, first on-chip split cache; Intel 486, first on-chip cache with a significant size; Alpha 21164, first second-level (L2) on-chip cache; and Intel Itanium 2, first third-level (L3) on-chip cache. Dot density is proportional to actual transistor density. Interior frames approximate the area devoted to on-chip cache.

sistors per chip, the clock rate, and on-chip cache allocation. On the software side, we watched memory footprint, instruction count, and first-level-cache miss rates.

On-chip memory allows data delivery and instruction fetch to be as fast and as wide as needed. It has evolved from 0 bytes to megabytes in less than 15 years, as Figure 1 illustrates. On-chip memory's influence on microprocessor performance suggests that its evolution is directly linked to software demands. For this reason, we examine on-chip memory, asking different questions. For example, since the appearance of on-chip cache in 1985, has its growth rate surpassed or fallen short of the growth rate of the number of transistors on a chip?

Our analysis of quantitative evidence led to several interesting findings. First, what we call software demand and hardware supply continually adjust to each other. Second, the percentage of chip transistors devoted to cache memory remains stable in each microprocessor family, considered separately. Additionally, on-chip-cache size grows according to the amount of data memory that programs demand. We note also that first-level on-chip caches grow at a slower pace, but they do not yield an increase in miss rates for all codes.

## Evolution of supply

Multiplying chip transistor count by frequency (TransFreq) gives us a figure closely related to peak performance. Setting all other elements (compiler, ISA, and microarchitecture) aside and doubling either transistor count or frequency can double processing speed. If we simultaneously double the frequency, the number of functional units, and the on-chip storage capacity, programs with a lot of ILP can speed up roughly by a factor of 4, as long as off-chip communication is not a bottleneck. Inefficiencies in extracting ILP prevent a given increase in the transistor budget from translating directly into performance, but TransFreq can serve as a comparative

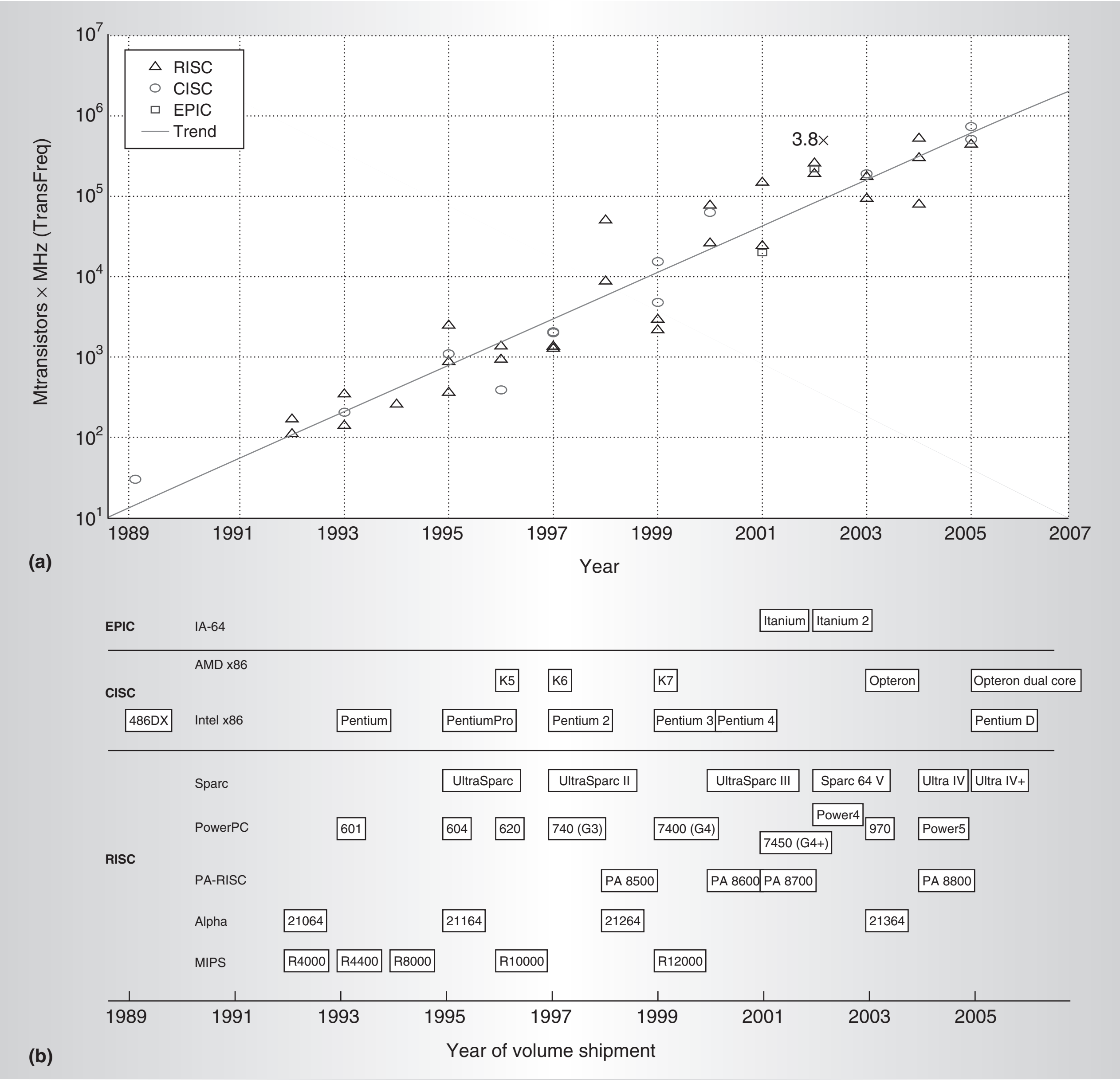

Figure 2. Performance potential of leading microprocessors, focusing on the version released in the volume shipment year. Evolution of TransFreq (transistor count × frequency) in the microprocessor architectures sampled (a); detail of microprocessor sample (b).

measure to help us understand the trends among competing microprocessors.

## Leading microprocessors and peak performance

Figure 2a shows the TransFreq numbers for the three types of processor architectures, and Figure 2b shows our sample of leading processors. Even though only two EPIC (explicitly parallel instruction computing) microprocessors had been released at the time of our analysis, we include them because of their outstanding technological meaning; however, we do not compute tendencies separately for EPIC. Values cluster around the exponential interpolation trend line. This fact isn't apparent if we consider frequency and transistor count separately. TransFreq almost quadruples every two years, contributing both transistor count and frequency in similar shares (2.2× and 1.7× respectively). In other words,

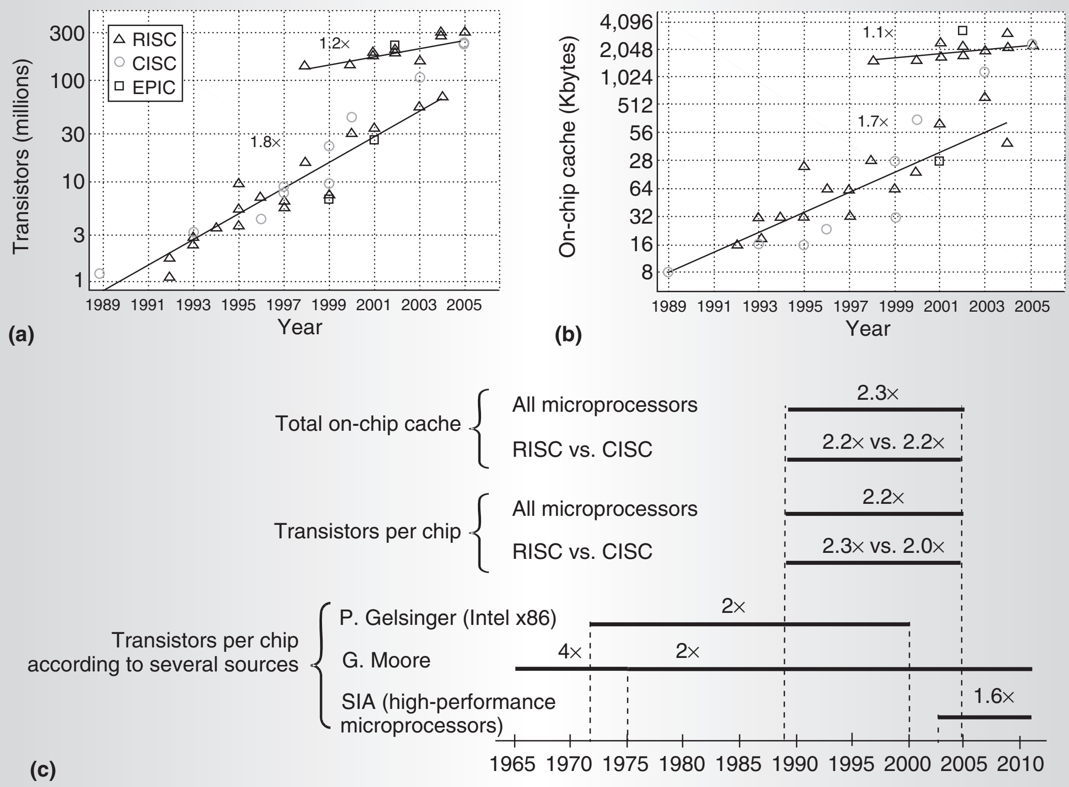

Figure 3. Evolution of microprocessor architectures in our sample and their growth factors for two-year periods: overall transistor count (a) and aggregate on-chip-cache size (b). Each plot shows two trend lines, corresponding to on-chip caches below or above 1 Mbyte. Our computed trends for total on-chip cache and number of transistors per chip appear in comparison with reports from other sources (c).

new microprocessors have twice as many transistors (room for functional units and storage capacity) and operate at a twofold frequency. The challenge for the compiler-ISA-microarchitecture complex in a high-performance environment is just to exploit this fact.

## Transistor count and on-chip caches

Figure 3a shows transistor count and Figure 3b shows total on-chip-cache size for our microprocessor sample. For microprocessors developed in the past eight years, we distinguish two groups, depending on whether the total on-chip-cache size is below or above 1 Mbyte (transistor count below or above 100 million). The group on the lower trend line includes microprocessors with only a first-level cache or a moderate-size second-level cache—that is, just enough for the performance of low-cost systems. The group on the upper trend line, above 1 Mbyte (Power4 and Power5, Sparc64 V, UltraSparcIV+, Alpha 21364, Opteron, Opteron dual core, and Pentium D), has either a second-level cache of about 1 Mbyte, with a third-level cache (Itanium 2), or a first-level cache several orders of magnitude larger than the rest. HP's PA-RISC processors exemplify the latter case. The exponential trend seems reasonable for each group, although scattering is higher than for the TransFreq measure, especially for cache allocation. The cache allocation growth factor is lower for the group above 1 Mbyte (1.1×, compared with 1.7× every two years). The transistor count growth trend (Figure 3a) is 1.2× for microprocessors with more than 100 million transistors and 1.8× for those with fewer transistors.

Figure 3c shows the resulting trends for all microprocessors (except EPIC) as a single group and by architecture. Note that the growth rates for transistor count and on-chip-cache allocation are similar. The 2.2× factor for the transistor count for the whole set of microprocessors is much higher than the Semiconductor Industry Association (SIA) estimates for the forthcoming years (1.6×).[8] The 2.0× factor for the CISC architecture matches both Moore's law[1,2] and Gelsinger's estimates for the Intel x86 family.[7]

Because transistor count and cache allocation have similar growth rates, caches represent a fairly constant percentage of the overall transistor count in leading microprocessors. This fact doesn't necessarily apply to any particular leading microprocessor's evolution, because CMOS process improvements are also a factor. Shrinking the feature size lets engineers design another generation of essentially the same microarchitecture by including extra transistors that are used almost entirely to enlarge the cache hierarchy. A good example is the Itanium 2 processor's evolution from the McKinley core (3 Mbytes) to the Madison core (9 Mbytes).

### Family and cache-level trends

Figure 4 shows the evolution of on-chip-cache levels and sizes for each leading microprocessor family. As we have indicated, most families use an approximately constant percentage of the transistor count for allocating caches. AMD and Sparc microprocessors exceed the general trend, whereas MIPS processors fall below. The first-level-cache size—instructions plus data—grows more slowly than the transistor count in all cases except HP PA-RISC microprocessors, until it reaches a value that differs from family to family. In the Intel Pentium 4, the data cache even shrinks. This smaller cache might be a response to the need to serve several simultaneous requests at processor speed by using several ports or memory banks. The instruction cache size also stabilizes, except in the Pentium 4 (estimated size 96 Kbytes). Interestingly, the cache size required by code for the SPEC CPU suites hardly grows at all, suggesting that stabilization of the instruction cache size doesn't adversely affect SPEC-like codes.

In most processor families, having a second-level cache compensates for the effects of a frozen first-level-cache size. Traditionally, microprocessors targeted for high-end servers have been endowed with generous second-level off-chip caches (from 1.5 Mbytes to 2 Mbytes). These second-level caches can now be integrated into the chip (examples include Alpha, Sparc64 V, and Power5). A third-level on-chip cache appears for the first time in the Intel Itanium 2. This design takes advantage of the technological integration capacity to include several megabytes of cache on chip, with a latency lower than that of external memory at any level. We include the not-yet-released Itanium 3 (Figure 4h) but without a tendency line, because having only three points could be misleading.

## Evolution of demand

Our study of software evolution considers all SPEC suites from 1989 through 2000. To make the comparison coherent, we generated always-optimized Sparc binaries. To account for each microprocessor generation's temporal scenario, we compiled SPEC89 and SPEC92 for the Sparc V7, SPEC95 for the Sparc V8, and SPEC 2000 for the Sparc V8+A (-fast -xO5 in all cases).

Quantifying the memory behavior of the entire SPEC series means running all the reference inputs of each of the 74 benchmarks, totaling 256 runs (210 integer and 46 floating point). To obtain experimental data in a reasonable time, we concentrated on three simple metrics. To evaluate software evolution, we measured the number of executed instructions, the memory footprint of data, and the memory footprint of instructions. For software-cache interaction, we measured the miss rates for all SPEC series, obtained by simulation using the Shade tool for all of these experiments.[9]

We obtained miss rates by simulating typical first-level caches for every time period. To define typical first-level caches, we assigned a three-year life to each microprocessor (covering its volume shipment year plus the following two years). Then, for every year, we applied a median filter and computed the mean of each parameter (cache size, block size, and associativity), rounding to the nearest power of 2. This method provided a conservative estimate of cache and block sizes and set associativity. Blocks were always 32 bytes. The resulting sizes and associativities appear in Figure 5.

### Memory footprint

For every program, we measured the number of references issued to every 32-byte memory block, and we ordered the blocks

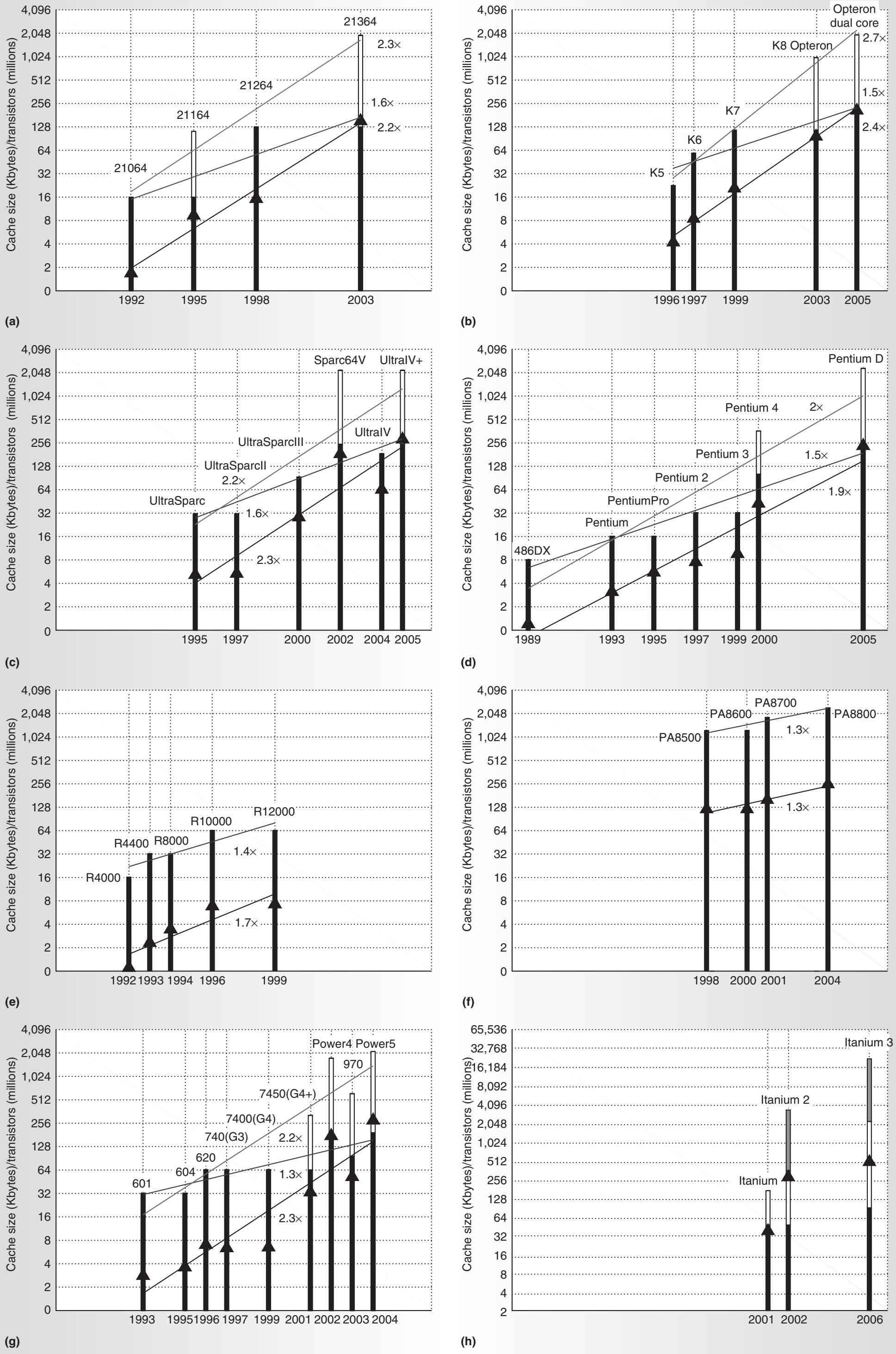

Figure 4. On-chip-cache size (bars) and chip transistor count (triangles) for leading microprocessor families: Alpha (a), AMD-86 (b), Sparc (c), Intel x86 (d), MIPS (e), HP PA (f), PowerPC (g), and Itanium (h). Cache levels are stacked: first (black), second (white), and third (gray), from bottom to top. The trend lines are exponential interpolations: transistor count (solid line), first-level-cache size (dashed line), and total on-chip-cache size (dot-dashed line). Numbers near these trend lines show the growth factor every two years. The *y*-axis for the Itanium (h) has a different scale.

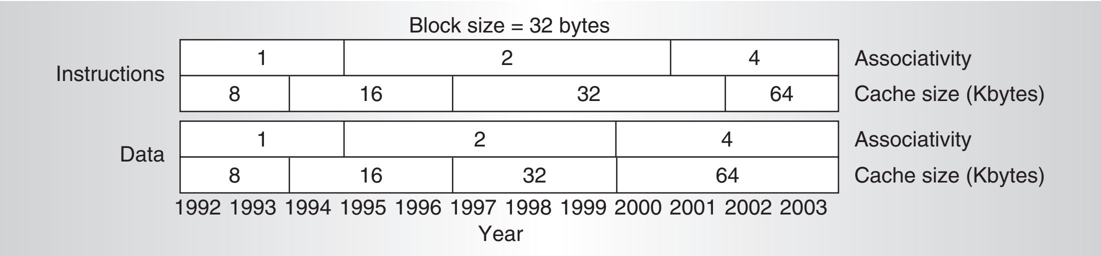

Figure 5. Typical first-level-cache parameters for instructions and data.

according to that number. The entire list of referenced blocks makes up the 100 percent memory footprint—namely, the memory accessed by the program. From the ordered list, we selected the minimum number of blocks that accumulated—for example, 50 percent or 90 percent of the memory references to instructions or data, thereby obtaining a 50 percent or 90 percent memory footprint for instructions or data. According to Thiebaut and Stone, this metric is like the cache footprint for an infinite cache.[10] It doesn't consider locality strictly and therefore doesn't let us draw precise conclusions about cache performance, but it provides an intuitive idea of the pressure the memory hierarchy holds.

Actually, a program with good temporal locality yields a very low miss rate when executing with a cache size approaching a 90 percent memory footprint. Moreover, the memory footprint is much simpler to obtain than the working set, which would require much greater simulation time (currently a year of CPU time). Figure 6 plots the arithmetic mean of a 100 percent, a 90 percent, and a 50 percent memory footprint for every SPEC suite, separating instruction footprints (Figures 6a and 6b) from data footprints (Figures 6c and 6d).

### Instructions

Figures 6a and 6b show that there is no growth in the 100 percent memory footprint for instructions. For CINT programs, this footprint falls between 128 Kbytes and 256 Kbytes, and for CFP programs it's about 128 Kbytes. The 50 percent memory footprint is very small for both CINT and CFP programs (about 2 Kbytes). The 90 percent memory footprint is moderate in size: 16 Kbytes for CINT (at or below the typical instruction cache size of each time period), and 8 Kbytes for CFP (always below the typical instruction cache size).

CINT programs have high locality, well above the old 90/10 rule-of-thumb. Fifty percent of references to instructions point to between 0.6 percent and 1.2 percent of instruction blocks, and 90 percent of references to instructions point to about 7 percent of instruction blocks. CFP program behavior is similar, but the percentages of referenced blocks are slightly higher. In the SPEC 2000 benchmark, the reference count increases, but memory footprints decrease. This might indicate that the total count of executed instructions rises because the processed data sets are larger, not because the benchmark code is more complex than in previous SPEC CPU editions.

### Data

The 100 percent memory footprint for data is much larger (1 Mbyte to 64 Mbytes) for the whole set of suites from 1989 through 2000, and it's reasonable to assume that this metric follows an exponential trend. Nevertheless, SPEC92 programs clearly split that period in two. The rates of growth for 100 percent and 90 percent memory footprints are 2.3× and 2.1× in CINT and CFP, respectively, every two years. These trends demonstrate an important experimental fact: Memory footprint growth is comparable to that of on-chip caches (Figures 3b and 3c). These findings agree with those of Hennessy and Patterson,[11] who stated that the software-required memory growth factor is between 2.25× and 4×.

Today's chips cannot capture a 90 percent memory footprint for data over the entire SPEC 2000 suite, but this doesn't necessarily imply poor on-chip-cache performance. We can say that the 50 percent memory footprint is small for CINT (8 Kbytes to 64 Kbytes, or less than

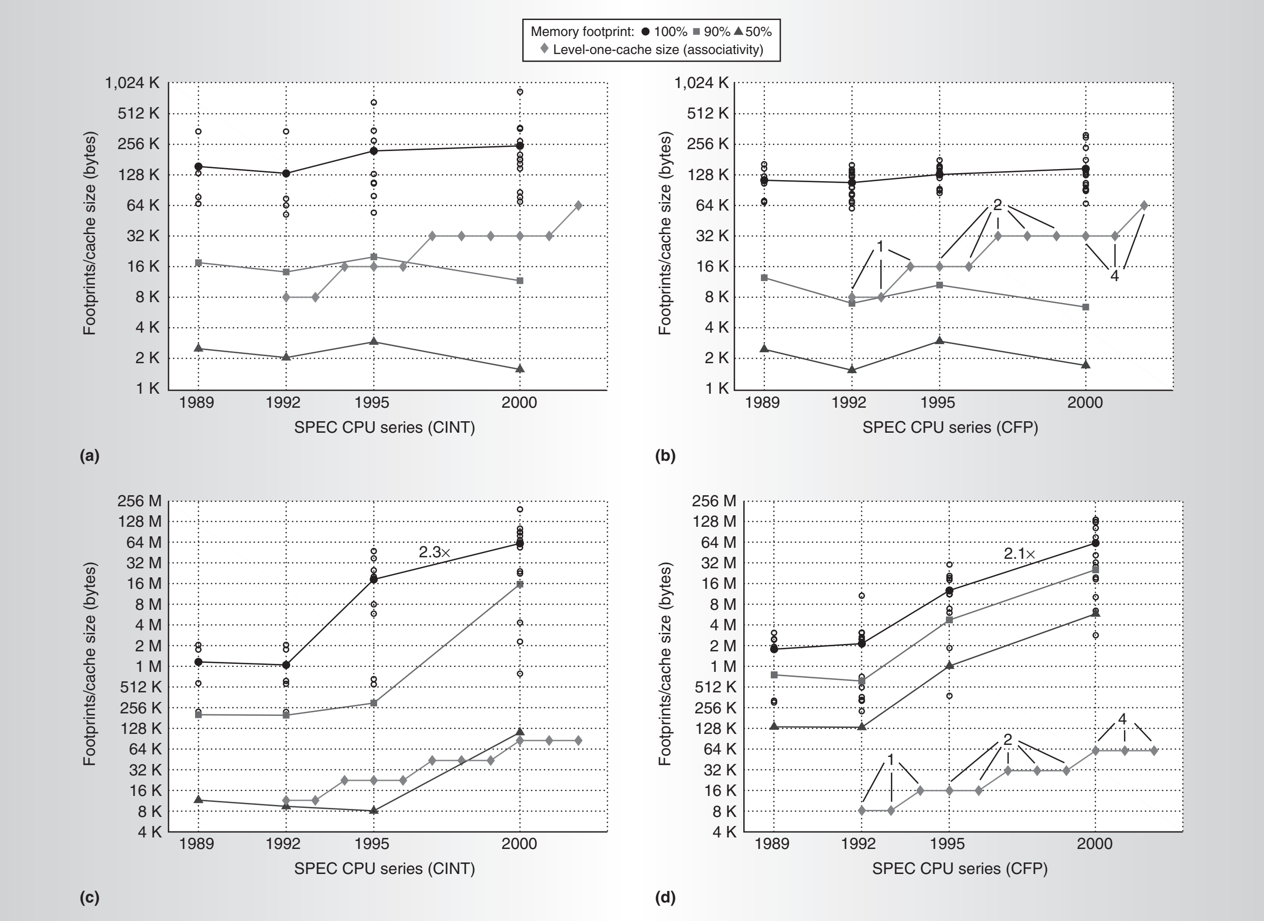

Figure 6. Memory footprints of 100 percent, 90 percent, and 50 percent for CINT instructions (a), CFP instructions (b), CINT data (c), and CFP data (d). The diamond icons indicate the size of the typical cache (as previously defined) for every period, scaled on the *y*-axis; the numbers 1, 2, and 4 indicate the typical cache's associativity by year. Circles above and below the 100 percent memory footprint lines represent the datum of each program, to highlight dispersion. In the 90 percent and 50 percent memory footprints, we omit per-program data because dispersion is very low.

1 percent of referenced blocks), and that it grows by a moderate factor (1.6× every two years). The 50 percent memory footprint for CFP is much larger (128 Kbytes to 8 Mbytes, 6 percent to 10 percent of referenced blocks), and growth matches that of the 100 percent memory footprint (2.1× every two years).

### Number of instructions executed

Figure 7 plots the count of instructions executed for each SPEC series, and an exponential trend seems reasonable. The resulting growth factor (2.8× every two years for the entire SPEC suite) is comparable to the 3.8× TransFreq growth factor shown in Figure 2a, considering that the latter is an optimistic performance upper bound (real performance grows at a slower pace). Therefore, a second important experimental outcome is that software demand (instruction count) seems to keep pace with hardware supply. But let's assume the SPEC CPU suite actually measures the maximum real performance. This is a reasonable target for a benchmark; ideally, every SPEC edition should execute on a computer of its own era in constant time. If this assumption holds true, the peak computational power (TransFreq) suffers from an inefficiency factor of exactly 0.7× every two years.

Assuming the next SPEC generation follows the growth factors in Figure 7, executing both CINT 2006 and CFP 2006 will mean running about 200 tera-instructions overall—a nice challenge for the computer industry.

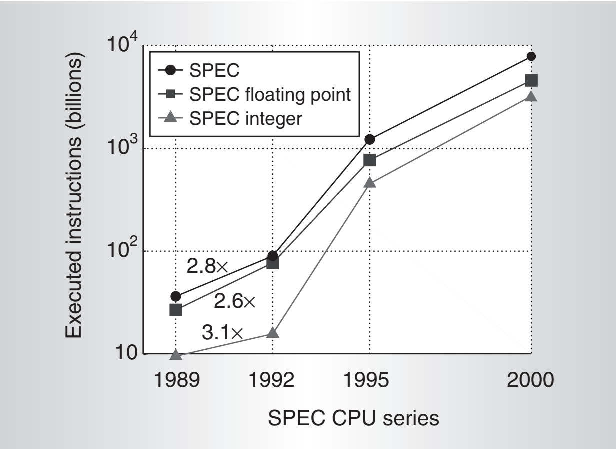

Figure 7. Executed instruction count for each CINT suite (triangles), each CFP suite (squares), and the entire SPEC suite (circles).

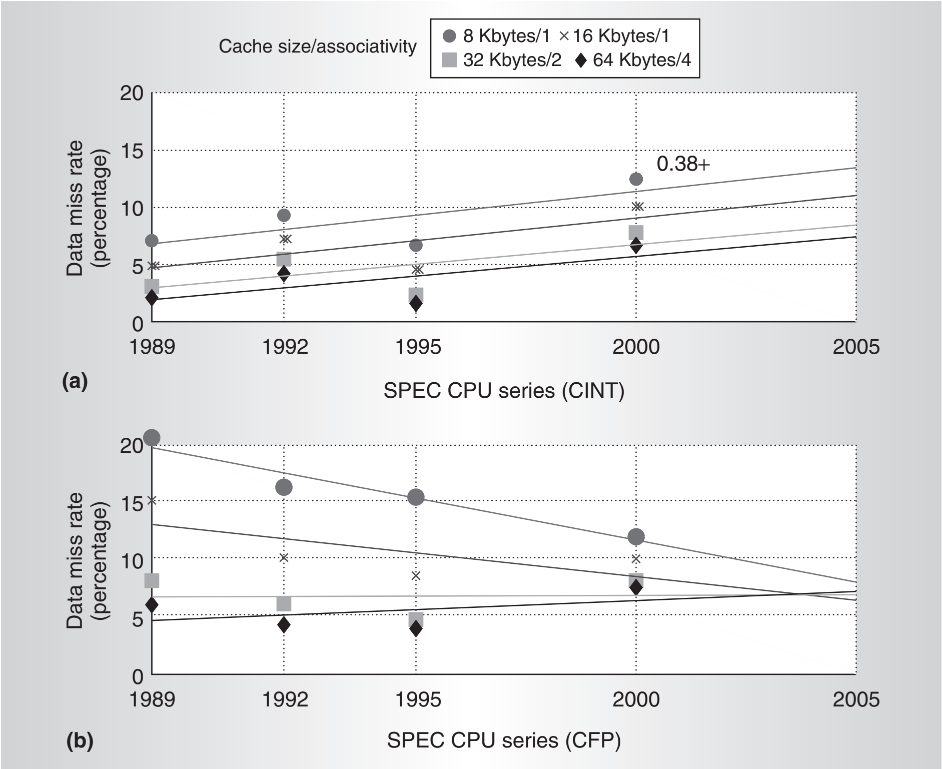

Figure 8. Data miss rates for each SPEC CPU release (projected to 2005), using the typical first-level data caches defined in Figure 6: CINT miss rates (a) and CFP miss rates (b). Every data point on the graph shows the arithmetic mean of the miss rates for an entire CINT or CFP release.

## Miss rate trends in first-level caches

Accurately measuring cache memory performance means computing the average memory access time and calculating its contribution to program execution time. To do this, we must consider the different features of each microprocessor and cache hierarchy along the time period we analyze. This includes cores issuing in-order and out-of-order instructions, and memory caches ranging from the simplest to those supporting several pending misses, several ports or banks, and prefetching mechanisms. Considering the workload, this cycle-by-cycle simulation isn't really feasible. So we opted for a simpler approach, recognizing its limits. The miss rate (number of cache misses for every 100 references) is easier to compute and is sufficient to highlight a first-level cache's performance resulting from software evolution, which is our goal. Therefore, we first defined a typical first-level cache for every year from 1992 through 2003; then, using Shade, we simulated the execution of every program in all the SPEC CPU series (1989, 1992, 1995, and 2000) on every typical cache.

Instruction miss rates seem to correlate with the data appearing in Figures 6a and 6b, where the 90 percent instruction memory footprint falls below the size of the typical corresponding cache. Integer code miss rates are low and nearly stable over time (averaging 0.1 percent for the typical 2002-2003 cache). Floating-point-code miss rates are even lower and decrease over time, yielding negligible values for the typical 2002-2003 cache (about 0.001 percent for SPEC 2000).

Data miss rates have a more interesting outcome: First-level data cache performance degrades for CINT codes but seems to stabilize for CFP codes. Integer code data miss rates (Figure 8a) increase linearly by 0.38+ every year, very close to the 0.40+ value that Flynn cited for 1975 through 1990.[12] According to this trend, the 2002-2003 typical cache should have a data miss rate of about 7.8 percent when executing CINT 2006. Thus, for integer codes, caches suffer increasing miss rates as the memory footprint grows.

Data miss rates for floating-point codes (Figure 8b), on the other hand, do not increase with the memory footprint. Experimental data for the four typical data caches shows data miss rates converging toward 10 percent, even though memory footprints have been growing exponentially since 1989. For example, the 50 percent memory footprint for floating-point data grows 2.1× every two years, remaining far larger than typical first-level data caches. In 1992, the data cache was roughly 16 times smaller than the 50 percent memory footprint, and in 2000 it was 128 times smaller. As the memory footprint grows over time, the smallest data caches (8 Kbytes/1, 16 Kbytes/1) experience a decrease in their miss rates, meaning that the locality captured by small caches is unrelated to the overall memory footprint.

Our performance simulation, along with hardware data collected from data sheets and published articles, provided quantitative evidence leading to two major conclusions:

- Microprocessors nearly quadruple their peak computing power every two years, whereas the SPEC CPU benchmarks increase the number of executed instructions by a factor of 2.8×. The trends are comparable, if we assume that real computing power always falls well below the theoretical peak. But this also implies that peak and real performance are moving increasingly further apart.
- The growing memory footprint of SPEC CPU applications is keeping pace with total on-chip-cache size.

The leading ISA microprocessors devote a constant percentage of chip transistors to cache allocation. Thus, Moore's law also applies to on-chip-cache size. We cannot yet say whether this holds true for new chips with multiple cores or multithreading capabilities. Unlike the total size of on-chip caches, first-level data cache size seems to stabilize, probably to feed increasingly faster and wider CPU cores in a timely fashion.

The moderate rise in first-level data cache miss rates for CINT codes is an optimistic performance degradation, because integer commercial workloads (Web servers, online transaction processing, and so on) make greater demands on the data cache than CINT does. First-level-cache size and associativity are unlikely to grow, owing to delay and bandwidth requirements. This increases the importance of traditional and new mechanisms for hiding latency.

Surprisingly, first-level data cache miss rates seem to stabilize for CFP codes over time, reducing the pressure on first-level caches. But is this a feature of the new floating-point codes or should the selection process for CFP codes be revised? Because of its implications in forthcoming SPEC CPU releases, this question deserves further investigation. MICRO

## Acknowledgments

This work was supported by Spain's Ministry of Education and Science and European RDF under contract TIN-2004-07739-C02-02, by Diputación General de Aragón (Grupo Consolidado de Investigación GAZ, Grupo de Arquitectura de Zaragoza), and by the European Network of Excellence on High-Performance Embedded Architecture and Compilation. We are also indebted to Oscar Blasco (Department of Education of the Aragonian government) for his invaluable help in the early stages of the work, and to Dianne Rice (SPEC) for supplying a copy of SPEC89, and to editor Bob Carlson for his valuable suggestions for improving the article.

## References

1. G.E. Moore, "Progress in Digital Integrated Electronics," *Proc. IEEE Int'l Electron Devices Meeting* (IEDM 75), IEEE Press, pp. 11-13.
2. G.E. Moore, "Cramming More Components onto Integrated Circuits," 1965; ftp://download.intel.com/museum/Moores_Law/Articles-Press_Releases/Gordon_Moore_1965_Article.pdf.
3. G.E. Moore, "No Exponential Is Forever: But 'Forever' Can Be Delayed!" *Proc. IEEE Int'l Solid-State Circuits Conf.* (ISSCC 03), IEEE Press; http://sscs.org/History/MooresLaw.htm.
4. R.R. Schaller, "Moore's Law: Past, Present, and Future," *IEEE Spectrum*, vol. 34, no. 6, June 1997, pp. 52-59.

5. N. Myhrvold, "The Next Fifty Years of Software," ACM talk, 1997; http://research.microsoft.com/acm97/.
6. SPEC, Sept. 2000; http://www.spec.org.
7. P.P. Gelsinger, "Microprocessors for the New Millennium: Challenges, Opportunities, and New Frontiers," *Proc. IEEE Int'l Solid-State Circuits Conf.* (ISSC 01), IEEE Press, 2001, pp. 22-25.
8. Semiconductor Industry Assn., *2003 Int'l Technology Roadmap for Semiconductors*.
9. R.F. Cmelik and D. Keppel, "Shade: A Fast Instruction-Set Simulator for Execution Profiling," *Proc. 1994 ACM Int'l Conf. Measurement and Modeling of Computer Systems*, ACM Press, 1994, pp 128-137.
10. D. Thiebaut and H.S. Stone, "Footprints in the Cache," *ACM Trans. Computer Systems*, vol. 5, no. 4, Nov. 1987, pp. 305-329.
11. J.L. Hennessy and D.A. Patterson, *Computer Architecture: A Quantitative Approach*, 2nd ed., Morgan Kaufman, 1996.
12. M.J. Flynn, *Computer Architecture: Pipelined and Parallel Processor Design*, Jones and Bartlett, Boston, 1995.

**Jesús Alastruey** is an assistant professor and a PhD candidate in the Informática e Ingeniería de Sistemas (IIS) Department at the University of Zaragoza (UZ), Spain. His research interests include memory hierarchy and processor microarchitecture. Alastruey has an MS in telecommunications from UZ.

**José Luis Briz** is an associate professor in the IIS Department at UZ. His research interests include memory hierarchy and processor microarchitecture. He has MS degrees in geology and computer science and a PhD in computer engineering, all from UZ. Briz is a member of the IEEE, the ACM, and UZ's Instituto de Investigación en Ingeniería de Aragón (I3A).

**Pablo Ibáñez** is an associate professor in the IIS Department at UZ. His research interests include memory hierarchy, processor microarchitecture, and parallel computer architecture. He has an MS in computer science from Universitat Politècnica de Catalunya and a PhD in computer engineering from UZ. Ibáñez is a member of the IEEE, the ACM, and the I3A.

**Victor Viñals** is a professor in the IIS Department at UZ. His research interests include processor microarchitecture, memory hierarchy, and parallel computer architecture. He has an MS in telecommunications and a PhD in computer science, both from Universitat Politècnica de Catalunya, where he also served on the faculty. Viñals is a member of the IEEE, the ACM, and the I3A.

Direct questions and comments about this article to J.L. Briz, Departamento de Informática e Ing. de Sistemas, University of Zaragoza, Maria de Luna 1, 50018 Zaragoza, Spain; briz@unizar.es.

For further information on this or any other computing topic, visit our Digital Library at http://www.computer.org/publications/dlib.