COMPUTER ARCHITECTURE

Ronald A. Thisted
Departments of Statistics, Health Studies, and Anesthesia & Critical Care
The University of Chicago

7 April 1997

To appear, *Encyclopedia of Biostatistics*.

A *computer architecture* is a detailed specification of the computational, communication, and data storage elements (hardware) of a computer system, how those components interact (machine organization), and how they are controlled (instruction set). A machine's architecture determines which computations can be performed most efficiently, and which forms of data organization and program design will perform optimally.

The term *architecture* as applied to computer design, was first used in 1964 by Gene Amdahl, G. Anne Blaauw, and Frederick Brooks, Jr., the designers of the IBM System/360. They coined the term to refer to those aspects of the instruction set available to programmers, independent of the hardware on which the instruction set was implemented. The System/360 marked the introduction of *families* of computers, that is, a range of hardware systems all executing essentially the same basic machine instructions.

The System/360 also precipitated a shift from the preoccupation of computer designers with computer arithmetic, which had been the main focus since the early 1950s. In the 1970s and 1980s, computer architects focused increasingly on the instruction set. In the current decade, however, designers' main challenges have been to implement processors efficiently, to design communicating memory hierarchies, and to integrate multiple processors in a single design.

It is instructive to examine how this transition from concentrating on instruction sets to high-level integration has taken place.

Instruction-set architecture

In the late 1970s, statisticians often had to be skilled FORTRAN programmers. Many were also sufficiently conversant with assembly language programming for a particular computer that they wrote subprograms directly using the computer's basic instruction set.

The Digital VAX 11/780 was a typical scientific computer of the era. The VAX had over 300 different machine-level instructions, ranging in size from 2 to 57 bytes in length, and 22 different addressing modes. Machines such as the VAX, the Intel 80x86 family of processors (the processors on which the IBM PC and its successors are based), and the Motorola 680x0 processors (on which the Apple Macintosh is based) all had multiple addressing modes, variable-length instructions, and large instruction sets. By the middle of the 1980s, such machines were described as "complex instruction-set computers" (CISC). These architectures did have the advantage that each instruction/addressing-mode combination performed its special task efficiently, making it possible to fine-tune performance on large tasks with very different characteristics and computing requirements.

In CISC computers, 80% or more of the computation time typically is spent executing

only a small fraction (10–20%) of the instructions in the instruction set, and many of the cycles are spent performing operating system services. Patterson and Ditzel [3] proposed the "reduced instruction-set computer" (RISC) in 1980. The RISC idea is to design a small set of instructions which make implementation of these most frequently performed tasks maximally efficient. The most common features of RISC designs are a single instruction size, a small number of addressing modes, and no indirect addressing.

RISC architectures became popular in the middle to late 1980s. Examples of RISC architectures include Digital's Alpha processor family, Hewlett-Packard's PA series, the Motorola/Apple/IBM PowerPC family, and Sun's Ultra-SPARC family. RISC architectures are particularly suitable for taking advantage of compiler-based optimization, which means that programs such as computationally-intensive statistical procedures written in compiled languages are likely to see best performance in RISC environments.

The *operating system* for a computer is a program that allocates and schedules the computer's computational resources such as processors and memory. For instance, the operating system controls how much of the computer's main memory is available to programs and how much is used to cache portions of files being read from or written to disk storage. Operating systems also determine which programs will be processed at any given time, and how long the processor will "belong" to the program currently claiming it. Computers such as the VAX and System/360 were designed in conjunction with an architecture-specific operating system (VMS and OS/360, respectively). Indeed, until the late 1980s, most operating systems were written specifically for computers with a particular architecture, and only a small number of these operating systems were ever converted to control machines based on other designs.

Early RISC architectures were designed to make Unix operating systems perform efficiently, and today most major Unix-based computer manufacturers rely on a RISC architecture. However, it is important to note that many different operating systems can run on a single computer, whether that computer's architecture is a CISC or a RISC design. For instance, Unix, Linux, MacOS, Windows NT, OS/2, and other operating systems all have versions that operate on PowerPC processors. Increasingly, architectures are designed to run multiple operating systems efficiently, and the most common operating systems do operate on many different processors. This means that choice of operating system is becoming less closely tied to the processor family on which a computer is based.

Hardware and Machine Organization

Up to this point we have talked largely about instruction-set aspects of computer architecture. The architectural advances of primary interest to statisticians today involve hardware and machine organization.

The hardware architecture consists of low-level details of a machine, such as timing requirements of components, layouts of circuit boards, logic design, power requirements, and the like. Fortunately, few of these details affect the day-to-day work of statisticians aside from their consequences — processors continue to get smaller and faster, and memory continues to get larger and less expensive.

At the level of machine organization, the computers we use are built of inter-dependent systems, of which the processor itself is just one. Others include memory and memory management systems, specialized instruction processors, busses for communication within and

between systems and with peripheral devices, and input/output controllers. In *multiprocessing* architectures, the protocols for interaction between multiple processors (the multiprocessing control system) is included as well.

In a typical computer based on Intel's Pentium Pro processor, the processor itself is tightly linked to a memory cache consisting of synchronous static random access memory (SRAM). This memory component is very fast, but because it is part of the same chip as the processor itself, it must also be quite small. This level-one cache memory is the first element of the Pentium Pro's *memory system.* This system in the Pentium architecture consists of a hierarchy of successively larger (but slower) memory layers: a level-two cache, coupled to main memory (dynamic RAM, or extended data-out (EDO) DRAM). The main memory bank is backed up in turn by virtual memory residing on disk; a memory management chipset (separate from the Pentium processor) controls transactions between these two layers of the memory system.

Input and output are handled via plug-in slots for processor cards attached to the system bus, which also serves the graphics controller. Typically, graphics output has its own, separate, graphics memory for preparing and displaying the contents of graphics displays. In addition, the graphics subsystem may incorporate specialized processors that accelerate graphical computations as well as specialized memory modules linked to the graphics processors.

For most statisticians using a computer on the desktop (or increasingly, on the laptop), then, improving performance can often be achieved by increasing the amount of level-two cache memory, by increasing the amount of main memory, by acquiring faster and larger disk drives, by adding graphics accelerators and graphics memory, or by adding a high-speed input/output controller.

On more powerful workstation systems, many of these features are integrated directly on the main processor chip. The UltraSPARC-1, for instance, has the memory management unit, the floating-point processor, as well as graphics and imaging support on the same chip as the instruction processor.

Floating-point computation

From the earliest days of digital computers, statistical computation has been dominated by floating-point calculations. "Scientific computers" are often defined as those which deliver high performance for floating-point operations. The aspects of computer design that make these operations possible have followed the same evolutionary path as the rest of computer architecture.

Early computer designs incorporated no floating-point instructions. Since all numbers were treated as integers, programmers working with nonintegers had to represent a number using one integer to hold the significant digits coupled to a second integer to record a scaling factor. In effect, each programmer had to devise his or her own floating-point representation. By the 1960s, some designs introduced instructions for floating-point operations, but many had none. (The IBM 1620 computer, for example, had fixed-precision integer arithmetic, but the programmer could control the number of digits of precision.) By the mid-1970s, virtually all scientific computers had floating-point instructions. Unfortunately for the practical statistician, the representation of floating-point numbers, the meaning of floating point operations, and the results of an operation differed substantially from one machine to the

next. The burden of knowing numerical analysis and good numerical algorithms fell heavily on the data analyst's shoulders.

In the early 1980s the Institute of Electrical and Electronics Engineers (IEEE) developed a standard for floating-point arithmetic [1]. To implement these standards, computer architects of that time developed a floating-point architecture separate from that of the principal processor — in effect, moving the floating-point issues from the level of machine-specific definition of arithmetic to a common set of operations (an "instruction set") whose output could be strictly defined. Examples include the Motorola 6888x and the Intel 80x87 floating-point processors (FPPs), which were designed in parallel with the 680x0 and 80x86 central processors, respectively.

In later RISC-based architectures, floating point processes are tightly coupled to the central instruction processor. The UltraSPARC-1 design incorporates an IEEE-compliant FPP which performs all floating-point operations (including multiplies, divides, and square roots). The PowerPC family also includes an integrated IEEE-compliant FPP. These processors illustrate the increased role of hardware and machine organization. The FPPs are logically external to the basic instruction processor. Therefore the computer design must include channels for communication of operands and results, and must incorporate machine-level protocols to guarantee that the mix of floating-point and non-floating-point instructions are completed in the correct order. Today, many of these FPPs are part of the same integrated-circuit package as the main instruction processor. This keeps the lines of communication very short and achieves additional effective processor speed.

Parallel and Vector Architectures

The discussion above concerns the predominant computer architecture used by practicing statisticians, one based on the sequential single processor. The notion that speed and reliability could be enhanced by coupling multiple processors in a way that enabled them to share work is an obvious extension, and one that has been explored actively since at least the 1950s.

Vector computers include instructions (and hardware!) that make it possible to execute a single instruction (such as an `add`) simultaneously on a vector of operands. In a standard scalar computer, computing the inner product of two vectors x and y of length p requires a loop within which the products $x_i y_i$ are calculated. The time required is that of p multiplications, together with the overhead of the loop. On a vector machine, a single instruction would calculate all p products at once.

The highest performance scientific computers available since 1975 incorporate vector architecture. Examples of early machines with vector capabilities include the CRAY-1 machine and its successors from Cray Research and the CYBER-STAR computers from CDC.

Vector processors are special cases of *parallel architecture,* in which multiple processors cooperatively perform computations. Vector processors are examples of machines which can execute a single instruction on multiple data streams (SIMD computers). In computers with these architectures, there is a single queue of instructions which are executed in parallel (that is, simultaneously) by multiple processors, each of which has its own data memory cache. Except for special purposes (such as array processing), the SIMD model is neither sufficiently flexible nor economically competitive for general-purpose designs.

Computers with multiple processors having individual data memory, and which fetch and

execute their instructions independently (MIMD computers), are more flexible than SIMD machines and can be built by taking regular scalar microprocessors and organizing them to operate in parallel. Such *multiprocessors* are rapidly approaching the capabilities of the fastest vector machines, and for many applications already have supplanted them.

Additional Reading and Summary

An elementary introduction to computer architecture is contained in Patterson and Hennessy [4]; more advanced topics are covered in [5].

Thisted [6] gives an overview of floating-point arithmetic and the IEEE standard with a view toward common statistical computations. Goldberg [2] presents an overview of floating-point issues for computer scientists.

Patterson and Hennessy [5] contains a lengthy chapter on parallel computers based on multiprocessors, as well as a short appendix on vector computers. Both sections contain historical notes and detailed pointers to the literature for additional reading.

In the early days of digital computers, few useful computations could be accomplished without a thorough understanding of the computer's instruction set and a detailed knowledge of its architecture. Advances in computer architecture, coupled with advances in programming languages, compiler technology, operating systems, storage, and memory, have made such specific knowledge of much reduced importance for producing high-quality scientific work.

References

[1] Institute of Electrical and Electronics Engineers (IEEE). (1985). *IEEE Standard for Binary Floating-Point Arithmetic (Standard 754-1985).* IEEE, New York. Reprinted in *SIGPLAN Notices,* **22(2)**, 5–48.

[2] Goldberg, D. (1991) "What every computer scientist should know about floating-point arithmetic," *Computing Surveys,* **23(1)**, 5–48.

[3] Patterson, David A., and Ditzel, D. R. (1980). "The case for the reduced instruction set computer," *Computer Architecture News,* **8(6)**, 25–33.

[4] Patterson, David A., and Hennessy, John L. (1994). *Computer Organization and Design: The Hardware/Software Interface.* Morgan Kaufmann: San Francisco.

[5] Patterson, David A., and Hennessy, John L. (1996). *Computer Architecture: A Quantitative Approach, Second Edition.* Morgan Kaufmann: San Francisco.

[6] Thisted, Ronald A. (1988). *Elements of Statistical Computing: Numerical Computation.* Chapman and Hall: London.