K

Historical Perspectives and References

If … history … teaches us anything, it is that man in his quest for knowledge and progress, is determined and cannot be deterred.

John F. Kennedy
address at Rice University
(1962)

Those who cannot remember the past are condemned to repeat it.

George Santayana
The Life of Reason (1905),
vol. 2, ch 3.

K.1 Introduction

This appendix provides historical background on some of the key ideas presented in the chapters. We may trace the development of an idea through a series of machines or describe significant projects. If you're interested in examining the initial development of an idea or machine or interested in further reading, references are provided at the end of each section.

Section K.2 starts us off with the invention of the digital computer and corresponds to Chapter 1. Section K.3 on instruction set architecture covers Appendices B, I, and J. Section K.4 on pipelining and instruction level parallelism corresponds to Chapters 2 and 3, and Appendices A and G. Section K.5 on multiprocessors and parallel programming covers Chapter 4 and Appendices E, F, and H. Section K.6 on memory hierarchy corresponds to Chapter 5 and Appendix C. Finally, Section K.7 on I/O corresponds to Chapter 6.

K.2 The Early Development of Computers (Chapter 1)

In this historical section, we discuss the early development of digital computers and the development of performance measurement methodologies.

The First General-Purpose Electronic Computers

J. Presper Eckert and John Mauchly at the Moore School of the University of Pennsylvania built the world's first fully operational electronic general-purpose computer. This machine, called ENIAC (Electronic Numerical Integrator and Calculator), was funded by the U.S. Army and became operational during World War II, but it was not publicly disclosed until 1946. ENIAC was used for computing artillery firing tables. The machine was enormous—100 feet long, $8^1/_2$ feet high, and several feet wide. Each of the 20 ten-digit registers was 2 feet long. In total, there were 18,000 vacuum tubes.

Although the size was three orders of magnitude bigger than the size of the average machines built today, it was more than five orders of magnitude slower, with an add taking 200 microseconds. The ENIAC provided conditional jumps and was programmable, which clearly distinguished it from earlier calculators. Programming was done manually by plugging up cables and setting switches and required from a half hour to a whole day. Data were provided on punched cards. The ENIAC was limited primarily by a small amount of storage and tedious programming.

In 1944, John von Neumann was attracted to the ENIAC project. The group wanted to improve the way programs were entered and discussed storing programs as numbers; von Neumann helped crystallize the ideas and wrote a memo proposing a stored-program computer called EDVAC (Electronic Discrete Variable Automatic Computer). Herman Goldstine distributed the memo and put von Neumann's name on it, much to the dismay of Eckert and Mauchly, whose

names were omitted. This memo has served as the basis for the commonly used term *von Neumann computer.* Several early inventors in the computer field believe that this term gives too much credit to von Neumann, who conceptualized and wrote up the ideas, and too little to the engineers, Eckert and Mauchly, who worked on the machines. Like most historians, your authors (winners of the 2000 IEEE von Neumann Medal) believe that all three individuals played a key role in developing the stored-program computer. Von Neumann's role in writing up the ideas, in generalizing them, and in thinking about the programming aspects was critical in transferring the ideas to a wider audience.

In 1946, Maurice Wilkes of Cambridge University visited the Moore School to attend the latter part of a series of lectures on developments in electronic computers. When he returned to Cambridge, Wilkes decided to embark on a project to build a stored-program computer named EDSAC (Electronic Delay Storage Automatic Calculator). (The EDSAC used mercury delay lines for its memory; hence the phrase "delay storage" in its name.) The EDSAC became operational in 1949 and was the world's first full-scale, operational, stored-program computer [Wilkes, Wheeler, and Gill 1951; Wilkes 1985, 1995]. (A small prototype called the Mark I, which was built at the University of Manchester and ran in 1948, might be called the first operational stored-program machine.) The EDSAC was an accumulator-based architecture. This style of instruction set architecture remained popular until the early 1970s. (Appendix B starts with a brief summary of the EDSAC instruction set.)

In 1947, Mauchly took the time to help found the Association of Computing Machinery. He served as the ACM's first vice-president and second president.

That same year, Eckert and Mauchly applied for a patent on electronic computers. The dean of the Moore School, by demanding the patent be turned over to the university, may have helped Eckert and Mauchly conclude they should leave. Their departure crippled the EDVAC project, which did not become operational until 1952.

Goldstine left to join von Neumann at the Institute for Advanced Study at Princeton in 1946. Together with Arthur Burks, they issued a report based on the 1944 memo [Burks, Goldstine, and von Neumann 1946]. The paper led to the IAS machine built by Julian Bigelow at Princeton's Institute for Advanced Study. It had a total of 1024 40-bit words and was roughly 10 times faster than ENIAC. The group thought about uses for the machine, published a set of reports, and encouraged visitors. These reports and visitors inspired the development of a number of new computers, including the first IBM computer, the 701, which was based on the IAS machine. The paper by Burks, Goldstine, and von Neumann was incredible for the period. Reading it today, you would never guess this landmark paper was written more than 50 years ago, as most of the architectural concepts seen in modern computers are discussed there (e.g., see the quote at the beginning of Chapter 5).

In the same time period as ENIAC, Howard Aiken was designing an electromechanical computer called the Mark-I at Harvard. The Mark-I was built by a team of engineers from IBM. He followed the Mark-I with a relay machine, the

Mark-II, and a pair of vacuum tube machines, the Mark-III and Mark-IV. The Mark-III and Mark-IV were built after the first stored-program machines. Because they had separate memories for instructions and data, the machines were regarded as reactionary by the advocates of stored-program computers. The term *Harvard architecture* was coined to describe this type of machine. Though clearly different from the original sense, this term is used today to apply to machines with a single main memory but with separate instruction and data caches.

The Whirlwind project [Redmond and Smith 1980] began at MIT in 1947 and was aimed at applications in real-time radar signal processing. Although it led to several inventions, its overwhelming innovation was the creation of magnetic core memory, the first reliable and inexpensive memory technology. Whirlwind had 2048 16-bit words of magnetic core. Magnetic cores served as the main memory technology for nearly 30 years.

Important Special-Purpose Machines

During the Second World War, there were major computing efforts in both Great Britain and the United States focused on special-purpose code-breaking computers. The work in Great Britain was aimed at decrypting messages encoded with the German Enigma coding machine. This work, which occurred at a location called Bletchley Park, led to two important machines. The first, an electromechanical machine, conceived of by Alan Turing, was called BOMB [see Good in Metropolis, Howlett, and Rota 1980]. The second, much larger and electronic machine, conceived and designed by Newman and Flowers, was called COLOSSUS [see Randall in Metropolis, Howlett, and Rota 1980]. These were highly specialized cryptanalysis machines, which played a vital role in the war by providing the ability to read coded messages, especially those sent to U-boats. The work at Bletchley Park was highly classified (indeed some of it is still classified), and so its direct impact on the development of ENIAC, EDSAC, and other computers is hard to trace, but it certainly had an indirect effect in advancing the technology and gaining understanding of the issues.

Similar work on special-purpose computers for cryptanalysis went on in the United States. The most direct descendent of this effort was a company, Engineering Research Associates (ERA) [see Thomash in Metropolis, Howlett, and Rota 1980], which was founded after the war to attempt to commercialize on the key ideas. ERA built several machines, which were sold to secret government agencies, and was eventually purchased by Sperry-Rand, which had earlier purchased the Eckert Mauchly Computer Corporation.

Another early set of machines that deserves credit was a group of special-purpose machines built by Konrad Zuse in Germany in the late 1930s and early 1940s [see Bauer and Zuse in Metropolis, Howlett, and Rota 1980]. In addition to producing an operating machine, Zuse was the first to implement floating point, which von Neumann claimed was unnecessary! His early machines used a mechanical store that was smaller than other electromechanical solutions of the

time. His last machine was electromechanical but, because of the war, was never completed.

An important early contributor to the development of electronic computers was John Atanasoff, who built a small-scale electronic computer in the early 1940s [Atanasoff 1940]. His machine, designed at Iowa State University, was a special-purpose computer (called the ABC—Atanasoff Berry Computer) that was never completely operational. Mauchly briefly visited Atanasoff before he built ENIAC, and several of Atanasoff's ideas (e.g., using binary representation) likely influenced Mauchly. The presence of the Atanasoff machine, together with delays in filing the ENIAC patents (the work was classified, and patents could not be filed until after the war) and the distribution of von Neumann's EDVAC paper, were used to break the Eckert-Mauchly patent [Larson 1973]. Though controversy still rages over Atanasoff's role, Eckert and Mauchly are usually given credit for building the first working, general-purpose, electronic computer [Stern 1980]. Atanasoff, however, demonstrated several important innovations included in later computers. Atanasoff deserves much credit for his work, and he might fairly be given credit for the world's first special-purpose electronic computer and for possibly influencing Eckert and Mauchly.

Commercial Developments

In December 1947, Eckert and Mauchly formed Eckert-Mauchly Computer Corporation. Their first machine, the BINAC, was built for Northrop and was shown in August 1949. After some financial difficulties, the Eckert-Mauchly Computer Corporation was acquired by Remington-Rand, later called Sperry-Rand. Sperry-Rand merged the Eckert-Mauchly acquisition, ERA, and its tabulating business to form a dedicated computer division, called UNIVAC. UNIVAC delivered its first computer, the UNIVAC I, in June 1951. The UNIVAC I sold for $250,000 and was the first successful commercial computer—48 systems were built! Today, this early machine, along with many other fascinating pieces of computer lore, can be seen at the Computer History Museum in Mountain View, California. Other places where early computing systems can be visited include the Deutsches Museum in Munich and the Smithsonian in Washington, D.C., as well as numerous online virtual museums.

IBM, which earlier had been in the punched card and office automation business, didn't start building computers until 1950. The first IBM computer, the IBM 701 based on von Neumann's IAS machine, shipped in 1952 and eventually sold 19 units [see Hurd in Metropolis, Howlett, and Rota 1980]. In the early 1950s, many people were pessimistic about the future of computers, believing that the market and opportunities for these "highly specialized" machines were quite limited. Nonetheless, IBM quickly became the most successful computer company. The focus on reliability and a customer- and market-driven strategy was key. Although the 701 and 702 were modest successes, IBM's follow-up machines, the 650, 704, and 705 (delivered in 1954 and 1955) were significant successes, each selling from 132 to 1800 computers.

Several books describing the early days of computing have been written by the pioneers [Wilkes 1985, 1995; Goldstine 1972], as well as Metropolis, Howlett, and Rota [1980], which is a collection of recollections by early pioneers. There are numerous independent histories, often built around the people involved [Slater 1987], as well as a journal, *Annals of the History of Computing,* devoted to the history of computing.

Development of Quantitative Performance Measures: Successes and Failures

In the earliest days of computing, designers set performance goals—ENIAC was to be 1000 times faster than the Harvard Mark-I, and the IBM Stretch (7030) was to be 100 times faster than the fastest machine in existence. What wasn't clear, though, was how this performance was to be measured. In looking back over the years, it is a consistent theme that each generation of computers obsoletes the performance evaluation techniques of the prior generation.

The original measure of performance was time to perform an individual operation, such as addition. Since most instructions took the same execution time, the timing of one gave insight into the others. As the execution times of instructions in a machine became more diverse, however, the time for one operation was no longer useful for comparisons. To take these differences into account, an *instruction mix* was calculated by measuring the relative frequency of instructions in a computer across many programs. The Gibson mix [Gibson 1970] was an early popular instruction mix. Multiplying the time for each instruction times its weight in the mix gave the user the *average instruction execution time*. (If measured in clock cycles, average instruction execution time is the same as average CPI.) Since instruction sets were similar, this was a more accurate comparison than add times. From average instruction execution time, then, it was only a small step to MIPS (as we have seen, the one is the inverse of the other). MIPS had the virtue of being easy for the layperson to understand.

As CPUs became more sophisticated and relied on memory hierarchies and pipelining, there was no longer a single execution time per instruction; MIPS could not be calculated from the mix and the manual. The next step was benchmarking using kernels and synthetic programs. Curnow and Wichmann [1976] created the Whetstone synthetic program by measuring scientific programs written in Algol 60. This program was converted to FORTRAN and was widely used to characterize scientific program performance. An effort with similar goals to Whetstone, the Livermore FORTRAN Kernels, was made by McMahon [1986] and researchers at Lawrence Livermore Laboratory in an attempt to establish a benchmark for supercomputers. These kernels, however, consisted of loops from real programs.

As it became clear that using MIPS to compare architectures with different instruction sets would not work, a notion of relative MIPS was created. When the VAX-11/780 was ready for announcement in 1977, DEC ran small benchmarks that were also run on an IBM 370/158. IBM marketing referred to the 370/158 as a 1 MIPS computer, and since the programs ran at the same speed, DEC market-

ing called the VAX-11/780 a 1 MIPS computer. Relative MIPS for a machine M was defined based on some reference machine as

$$\text{MIPS}_{\text{M}} = \frac{\text{Performance}_{\text{M}}}{\text{Performance}_{\text{reference}}} \times \text{MIPS}_{\text{reference}}$$

The popularity of the VAX-11/780 made it a popular reference machine for relative MIPS, especially since relative MIPS for a 1 MIPS computer is easy to calculate: If a machine was five times faster than the VAX-11/780, for that benchmark its rating would be 5 relative MIPS. The 1 MIPS rating was unquestioned for four years, until Joel Emer of DEC measured the VAX-11/780 under a time-sharing load. He found that the VAX-11/780 native MIPS rating was 0.5. Subsequent VAXes that run 3 native MIPS for some benchmarks were therefore called 6 MIPS machines because they run six times faster than the VAX-11/780. By the early 1980s, the term MIPS was almost universally used to mean relative MIPS.

The 1970s and 1980s marked the growth of the supercomputer industry, which was defined by high performance on floating-point-intensive programs. Average instruction time and MIPS were clearly inappropriate metrics for this industry, hence the invention of MFLOPS (millions of floating-point operations per second), which effectively measured the inverse of execution time for a benchmark. Unfortunately customers quickly forget the program used for the rating, and marketing groups decided to start quoting peak MFLOPS in the supercomputer performance wars.

SPEC (System Performance and Evaluation Cooperative) was founded in the late 1980s to try to improve the state of benchmarking and make a more valid basis for comparison. The group initially focused on workstations and servers in the UNIX marketplace, and that remains the primary focus of these benchmarks today. The first release of SPEC benchmarks, now called SPEC89, was a substantial improvement in the use of more realistic benchmarks. SPEC2006 still dominates processor benchmarks almost two decades later.

References

Amdahl, G. M. [1967]. "Validity of the single processor approach to achieving large scale computing capabilities," *Proc. AFIPS 1967 Spring Joint Computer Conf.* 30 (April), Atlantic City, N.J., 483–485.

Atanasoff, J. V. [1940]. "Computing machine for the solution of large systems of linear equations," Internal Report, Iowa State University, Ames.

Bell, C. G. [1984]. "The mini and micro industries," *IEEE Computer* 17:10 (October), 14–30.

Bell, C. G., J. C. Mudge, and J. E. McNamara [1978]. *A DEC View of Computer Engineering*, Digital Press, Bedford, Mass.

Burks, A. W., H. H. Goldstine, and J. von Neumann [1946]. "Preliminary discussion of the logical design of an electronic computing instrument," Report to the U.S. Army Ordnance Department, p. 1; also appears in *Papers of John von Neumann,* W. Aspray and A. Burks, eds., MIT Press, Cambridge, Mass., and Tomash Publishers, Los Angeles, Calif., 1987, 97–146.

Curnow, H. J., and B. A. Wichmann [1976]. "A synthetic benchmark," *The Computer J.,* 19:1, 43–49.

Flemming, P. J., and J. J. Wallace [1986]. "How not to lie with statistics: The correct way to summarize benchmarks results," *Comm. ACM* 29:3 (March), 218–221.

Fuller, S. H., and W. E. Burr [1977]. "Measurement and evaluation of alternative computer architectures," *Computer* 10:10 (October), 24–35.

Gibson, J. C. [1970]. "The Gibson mix," Rep. TR. 00.2043, IBM Systems Development Division, Poughkeepsie, N.Y. (Research done in 1959.)

Goldstine, H. H. [1972]. *The Computer: From Pascal to von Neumann,* Princeton University Press, Princeton, N.J.

Gray, J., and C. van Ingen [2005]. "Empirical measurements of disk failure rates and error rates," Microsoft Research, MSR-TR-2005-166, December.

Jain, R. [1991]. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling,* Wiley, New York.

Kembel, R. [2000]. "Fibre Channel: A comprehensive introduction," *Internet Week* (April).

Larson, E. R. [1973]. "Findings of fact, conclusions of law, and order for judgment," File No. 4-67, Civ. 138, *Honeywell v. Sperry-Rand and Illinois Scientific Development,* U.S. District Court for the State of Minnesota, Fourth Division (October 19).

Lubeck, O., J. Moore, and R. Mendez [1985]. "A benchmark comparison of three supercomputers: Fujitsu VP-200, Hitachi S810/20, and Cray X-MP/2," *Computer* 18:12 (December), 10–24.

McMahon, F. M. [1986]. "The Livermore FORTRAN kernels: A computer test of numerical performance range," Tech. Rep. UCRL-55745, Lawrence Livermore National Laboratory, Univ. of California, Livermore (December).

Metropolis, N., J. Howlett, and G. C. Rota, eds. [1980]. *A History of Computing in the Twentieth Century,* Academic Press, New York.

Mukherjee S. S., C. Weaver, J. S. Emer, S. K. Reinhardt, and T. M. Austin [2003]. "Measuring architectural vulnerability factors," *IEEE Micro* 23:6, 70–75.

Oliker, L., A. Canning, J. Carter, J. Shalf, and S. Ethier [2004]. "Scientific computations on modern parallel vector systems," *Proc. ACM/IEEE Conference on Supercomputing,* 10.

Patterson, D. [2004]. "Latency lags bandwidth," *CACM* 47:10 (October), 71–75.

Redmond, K. C., and T. M. Smith [1980]. *Project Whirlwind—The History of a Pioneer Computer,* Digital Press, Boston.

Shurkin, J. [1984]. *Engines of the Mind: A History of the Computer*, W. W. Norton, New York.

Slater, R. [1987]. *Portraits in Silicon,* MIT Press, Cambridge, Mass.

Smith, J. E. [1988]. "Characterizing computer performance with a single number," *Comm. ACM* 31:10 (October), 1202–1206.

SPEC [1989]. *SPEC Benchmark Suite Release 1.0* (October 2).

SPEC [1994]. *SPEC Newsletter* (June).

Stern, N. [1980]. "Who invented the first electronic digital computer?" *Annals of the History of Computing* 2:4 (October), 375–376.

Touma, W. R. [1993]. *The Dynamics of the Computer Industry: Modeling the Supply of Workstations and Their Components,* Kluwer Academic, Boston.

Weicker, R. P. [1984]. "Dhrystone: A synthetic systems programming benchmark," *Comm. ACM* 27:10 (October), 1013–1030.

Wilkes, M. V. [1985]. *Memoirs of a Computer Pioneer,* MIT Press, Cambridge, Mass.

Wilkes, M. V. [1995]. *Computing Perspectives,* Morgan Kaufmann, San Francisco.
Wilkes, M. V., D. J. Wheeler, and S. Gill [1951]. *The Preparation of Programs for an Electronic Digital Computer*, Addison-Wesley, Cambridge, Mass.

K.3 The Evolution of Instruction Sets (Appendices B, I, and J)

One's eyebrows should rise whenever a future architecture is developed with a stack- or register-oriented instruction set.

Meyers [1978, p. 20]

The earliest computers, including the UNIVAC I, the EDSAC, and the IAS computers, were accumulator-based computers. The simplicity of this type of computer made it the natural choice when hardware resources were very constrained. The first general-purpose register computer was the Pegasus, built by Ferranti, Ltd., in 1956. The Pegasus had eight general-purpose registers, with R0 always being zero. Block transfers loaded the eight registers from the drum memory.

Stack Architectures

In 1963, Burroughs delivered the B5000. The B5000 was perhaps the first computer to seriously consider software and hardware-software trade-offs. Barton and the designers at Burroughs made the B5000 a stack architecture (as described in Barton [1961]). Designed to support high-level languages such as ALGOL, this stack architecture used an operating system (MCP) written in a high-level language. The B5000 was also the first computer from a U.S. manufacturer to support virtual memory. The B6500, introduced in 1968 (and discussed in Hauck and Dent [1968]), added hardware-managed activation records. In both the B5000 and B6500, the top two elements of the stack were kept in the processor and the rest of the stack was kept in memory. The stack architecture yielded good code density, but only provided two high-speed storage locations. The authors of both the original IBM 360 paper [Amdahl, Blaauw, and Brooks 1964] and the original PDP-11 paper [Bell et al. 1970] argue against the stack organization. They cite three major points in their arguments against stacks:

1. Performance is derived from fast registers, not the way they are used.
2. The stack organization is too limiting and requires many swap and copy operations.
3. The stack has a bottom, and when placed in slower memory there is a performance loss.

Stack-based hardware fell out of favor in the late 1970s and, except for the Intel 80x86 floating-point architecture, essentially disappeared. For example, except for the 80x86, none of the computers listed in the SPEC report use a stack.

In the 1990s, however, stack architectures received a shot in the arm with the success of the Java Virtual Machine (JVM). The JVM is a software interpreter for an intermediate language produced by Java compilers, called *Java bytecodes* [Lindholm and Yellin 1999]. The purpose of the interpreter is to provide software compatibility across many platforms, with the hope of "write once, run everywhere." Although the slowdown is about a factor of 10 due to interpretation, there are times when compatibility is more important than performance, such as when downloading a Java "applet" into an Internet browser.

Although a few have proposed hardware to directly execute the JVM instructions (see McGhan and O'Connor [1998]), thus far none of these proposals have been significant commercially. The hope instead is that *just in time* (JIT) Java compilers—which compile during run time to the native instruction set of the computer running the Java program—will overcome the performance penalty of interpretation. The popularity of Java has also led to compilers that compile directly into the native hardware instruction sets, bypassing the illusion of the Java bytecodes.

Computer Architecture Defined

IBM coined the term *computer architecture* in the early 1960s. Amdahl, Blaauw, and Brooks [1964] used the term to refer to the programmer-visible portion of the IBM 360 instruction set. They believed that a *family* of computers of the same architecture should be able to run the same software. Although this idea may seem obvious to us today, it was quite novel at that time. IBM, although it was the leading company in the industry, had five different architectures before the 360. Thus, the notion of a company standardizing on a single architecture was a radical one. The 360 designers hoped that defining a common architecture would bring six different divisions of IBM together. Their definition of architecture was

> . . . the structure of a computer that a machine language programmer must understand to write a correct (timing independent) program for that machine.

The term "machine language programmer" meant that compatibility would hold, even in machine language, while "timing independent" allowed different implementations. This architecture blazed the path for binary compatibility, which others have followed.

The IBM 360 was the first computer to sell in large quantities with both byte addressing using 8-bit bytes and general-purpose registers. The 360 also had register-memory and limited memory-memory instructions. Appendix J summarizes this instruction set.

In 1964, Control Data delivered the first supercomputer, the CDC 6600. As Thornton [1964] discusses, he, Cray, and the other 6600 designers were among the first to explore pipelining in depth. The 6600 was the first general-purpose, load-store computer. In the 1960s, the designers of the 6600 realized the need to simplify architecture for the sake of efficient pipelining. Microprocessor and minicomputer designers largely neglected this interaction between architectural simplicity and implementation during the 1970s, but it returned in the 1980s.

High-Level Language Computer Architecture

In the late 1960s and early 1970s, people realized that software costs were growing faster than hardware costs. McKeeman [1967] argued that compilers and operating systems were getting too big and too complex and taking too long to develop. Because of inferior compilers and the memory limitations of computers, most systems programs at the time were still written in assembly language. Many researchers proposed alleviating the software crisis by creating more powerful, software-oriented architectures. Tanenbaum [1978] studied the properties of high-level languages. Like other researchers, he found that most programs are simple. He then argued that architectures should be designed with this in mind and that they should optimize for program size and ease of compilation. Tanenbaum proposed a stack computer with frequency-encoded instruction formats to accomplish these goals. However, as we have observed, program size does not translate directly to cost-performance, and stack computers faded out shortly after this work.

Strecker's article [1978] discusses how he and the other architects at DEC responded to this by designing the VAX architecture. The VAX was designed to simplify compilation of high-level languages. Compiler writers had complained about the lack of complete orthogonality in the PDP-11. The VAX architecture was designed to be highly orthogonal and to allow the mapping of a high-level language statement into a single VAX instruction. Additionally, the VAX designers tried to optimize code size because compiled programs were often too large for available memories. Appendix J summarizes this instruction set.

The VAX-11/780 was the first computer announced in the VAX series. It is one of the most successful—and most heavily studied—computers ever built. The cornerstone of DEC's strategy was a single architecture, VAX, running a single operating system, VMS. This strategy worked well for over 10 years. The large number of papers reporting instruction mixes, implementation measurements, and analysis of the VAX makes it an ideal case study [Wiecek 1982; Clark and Levy 1982]. Bhandarkar and Clark [1991] give a quantitative analysis of the disadvantages of the VAX versus a RISC computer, essentially a technical explanation for the demise of the VAX.

While the VAX was being designed, a more radical approach, called *high-level language computer architecture* (HLLCA), was being advocated in the research community. This movement aimed to eliminate the gap between high-level languages and computer hardware—what Gagliardi [1973] called the "semantic gap"—by bringing the hardware "up to" the level of the programming language. Meyers [1982] provides a good summary of the arguments and a history of high-level language computer architecture projects.

HLLCA never had a significant commercial impact. The increase in memory size on computers eliminated the code size problems arising from high-level languages and enabled operating systems to be written in high-level languages. The combination of simpler architectures together with software offered greater performance and more flexibility at lower cost and lower complexity.

Reduced Instruction Set Computers

In the early 1980s, the direction of computer architecture began to swing away from providing high-level hardware support for languages. Ditzel and Patterson [1980] analyzed the difficulties encountered by the high-level language architectures and argued that the answer lay in simpler architectures. In another paper [Patterson and Ditzel 1980], these authors first discussed the idea of Reduced Instruction Set Computers (RISC) and presented the argument for simpler architectures. Clark and Strecker [1980], who were VAX architects, rebutted their proposal.

The simple load-store computers such as MIPS are commonly called RISC architectures. The roots of RISC architectures go back to computers like the 6600, where Thornton, Cray, and others recognized the importance of instruction set simplicity in building a fast computer. Cray continued his tradition of keeping computers simple in the CRAY-1. Commercial RISCs are built primarily on the work of three research projects: the Berkeley RISC processor, the IBM 801, and the Stanford MIPS processor. These architectures have attracted enormous industrial interest because of claims of a performance advantage of anywhere from two to five times over other computers using the same technology.

Begun in 1975, the IBM project was the first to start but was the last to become public. The IBM computer was designed as a 24-bit ECL minicomputer, while the university projects were both MOS-based, 32-bit microprocessors. John Cocke is considered the father of the 801 design. He received both the Eckert-Mauchly and Turing awards in recognition of his contribution. Radin [1982] describes the highlights of the 801 architecture. The 801 was an experimental project that was never designed to be a product. In fact, to keep down cost and complexity, the computer was built with only 24-bit registers.

In 1980, Patterson and his colleagues at Berkeley began the project that was to give this architectural approach its name (see Patterson and Ditzel [1980]). They built two computers called RISC-I and RISC-II. Because the IBM project was not widely known or discussed, the role played by the Berkeley group in promoting the RISC approach was critical to the acceptance of the technology. They also built one of the first instruction caches to support hybrid format RISCs (see Patterson et al. [1983]). It supported 16-bit and 32-bit instructions in memory but 32 bits in the cache. The Berkeley group went on to build RISC computers targeted toward Smalltalk, described by Ungar et al. [1984], and LISP, described by Taylor et al. [1986].

In 1981, Hennessy and his colleagues at Stanford published a description of the Stanford MIPS computer. Efficient pipelining and compiler-assisted scheduling of the pipeline were both important aspects of the original MIPS design. MIPS stood for Microprocessor without Interlocked Pipeline Stages, reflecting the lack of hardware to stall the pipeline, as the compiler would handle dependencies.

These early RISC computers—the 801, RISC-II, and MIPS—had much in common. Both university projects were interested in designing a simple com-

puter that could be built in VLSI within the university environment. All three computers used a simple load-store architecture, fixed-format 32-bit instructions, and emphasized efficient pipelining. Patterson [1985] describes the three computers and the basic design principles that have come to characterize what a RISC computer is. Hennessy [1984] provides another view of the same ideas, as well as other issues in VLSI processor design.

In 1985, Hennessy published an explanation of the RISC performance advantage and traced its roots to a substantially lower CPI—under 2 for a RISC processor and over 10 for a VAX-11/780 (though not with identical workloads). A paper by Emer and Clark [1984] characterizing VAX-11/780 performance was instrumental in helping the RISC researchers understand the source of the performance advantage seen by their computers.

Since the university projects finished up, in the 1983–84 time frame, the technology has been widely embraced by industry. Many manufacturers of the early computers (those made before 1986) claimed that their products were RISC computers. These claims, however, were often born more of marketing ambition than of engineering reality.

In 1986, the computer industry began to announce processors based on the technology explored by the three RISC research projects. Moussouris et al. [1986] describe the MIPS R2000 integer processor, while Kane's book [1986] is a complete description of the architecture. Hewlett-Packard converted their existing minicomputer line to RISC architectures; Lee [1989] describes the HP Precision Architecture. IBM never directly turned the 801 into a product. Instead, the ideas were adopted for a new, low-end architecture that was incorporated in the IBM RT-PC and described in a collection of papers [Waters 1986]. In 1990, IBM announced a new RISC architecture (the RS 6000), which is the first superscalar RISC processor. In 1987, Sun Microsystems began delivering computers based on the SPARC architecture, a derivative of the Berkeley RISC-II processor; SPARC is described in Garner et al. [1988]. The PowerPC joined the forces of Apple, IBM, and Motorola. Appendix J summarizes several RISC architectures.

To help resolve the RISC versus traditional design debate, designers of VAX processors later performed a quantitative comparison of VAX and a RISC processor for implementations with comparable organizations. Their choices were the VAX 8700 and the MIPS M2000. The differing goals for VAX and MIPS have led to very different architectures. The VAX goals, simple compilers and code density, led to powerful addressing modes, powerful instructions, efficient instruction encoding, and few registers. The MIPS goals were high performance via pipelining, ease of hardware implementation, and compatibility with highly optimizing compilers. These goals led to simple instructions, simple addressing modes, fixed-length instruction formats, and a large number of registers.

Figure K.1 shows the ratio of the number of instructions executed, the ratio of CPIs, and the ratio of performance measured in clock cycles. Since the organizations were similar, clock cycle times were assumed to be the same. MIPS executes about twice as many instructions as the VAX, while the CPI for the VAX is about six times larger than that for the MIPS. Hence, the MIPS M2000 has almost three times the

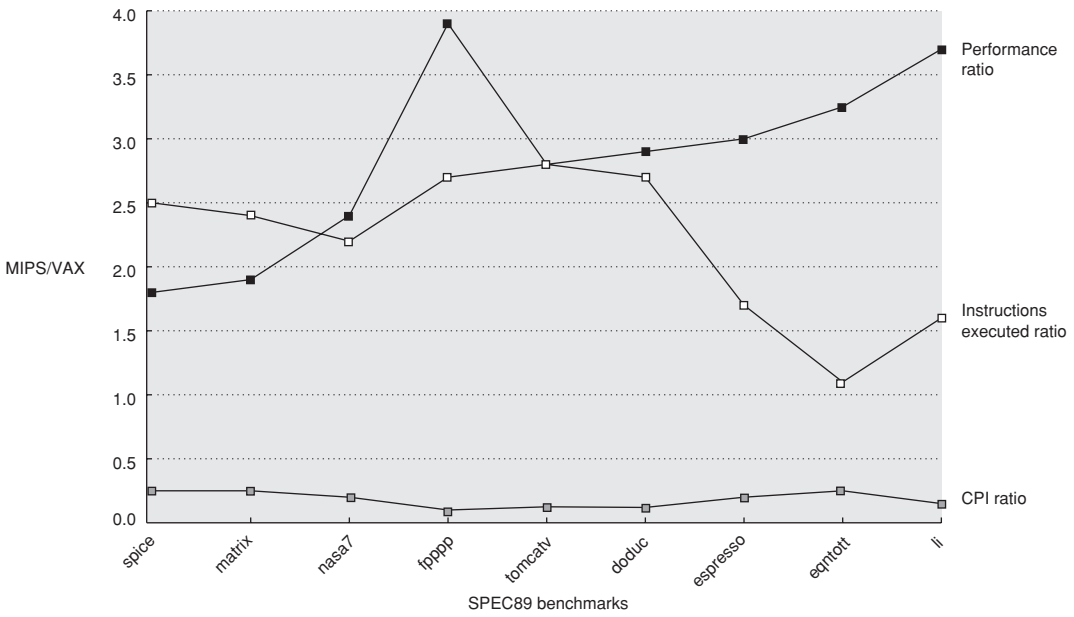

Figure K.1 Ratio of MIPS M2000 to VAX 8700 in instructions executed and performance in clock cycles using SPEC89 programs. On average, MIPS executes a little over twice as many instructions as the VAX, but the CPI for the VAX is almost six times the MIPS CPI, yielding almost a threefold performance advantage. (Based on data from Bhandarkar and Clark [1991].)

performance of the VAX 8700. Furthermore, much less hardware is needed to build the MIPS processor than the VAX processor. This cost-performance gap is the reason the company that used to make the VAX introduced a MIPS-based product, and then has dropped the VAX completely and switched to Alpha, which is quite similar to MIPS. Bell and Strecker [1998] summarize the debate inside the company. Today, DEC, once the second largest computer company and the major success of the minicomputer industry, exists only in remnants within HP and Intel.

Looking back, only one Complex Instruction Set Computer (CISC) instruction set survived the RISC/CISC debate, and that one had binary compatibility with PC software. The volume of chips is so high in the PC industry that there is a sufficient revenue stream to pay the extra design costs—and sufficient resources due to Moore's Law—to build microprocessors that translate from CISC to RISC internally. Whatever loss in efficiency, due to longer pipeline stages and bigger die size to accommodate translation on the chip, was overcome by the enormous volume and the ability to dedicate IC processing lines specifically to this product.

Interestingly, Intel also concluded that the future of the 80x86 line was doubtful. They created the IA-64 architecture to support 64-bit addressing and to move to a RISC-style instruction set. The embodiment of the IA-64 (see Huck et al. [2000]) architecture in the Itanium-1 and Itanium-2 has been a mixed success.

Although high performance has been achieved for floating-point applications, the integer performance was never impressive. In addition, the Itanium implementations have been large in transistor count and die size and power hungry. The complexity of the IA-64 instruction set, standing at least in partial conflict with the RISC philosophy, no doubt contributed to this area and power inefficiency.

AMD decided instead to just stretch the architecture from a 32-bit address to a 64-bit address, much as Intel had done when the 80386 stetched it from a 16-bit address to a 32-bit address. Intel later followed AMD's example. In the end, the tremendous marketplace advantage of the 80x86 presence was too much even for Intel, the owner of this legacy, to overcome!

References

Alexander, W. G., and D. B. Wortman [1975]. "Static and dynamic characteristics of XPL programs," *IEEE Computer* 8:11 (November), 41–46.

Amdahl, G. M., G. A. Blaauw, and F. P. Brooks, Jr. [1964]. "Architecture of the IBM System 360," *IBM J. Research and Development* 8:2 (April), 87–101.

Barton, R. S. [1961]. "A new approach to the functional design of a computer," *Proc. Western Joint Computer Conf.,* 393–396.

Bell, G., R. Cady, H. McFarland, B. DeLagi, J. O'Laughlin, R. Noonan, and W. Wulf [1970]. "A new architecture for mini-computers: The DEC PDP-11," *Proc. AFIPS SJCC*, 657–675.

Bell, G., and W. D. Strecker [1998]. "Computer structures: What have we learned from the PDP-11?" *25 Years of the International Symposia on Computer Architecture (Selected Papers)*, ACM, 138–151.

Bhandarkar, D. P. [1995]. *Alpha Architecture and Implementations,* Digital Press, Newton, Mass.

Bhandarkar, D., and D. W. Clark [1991]. "Performance from architecture: Comparing a RISC and a CISC with similar hardware organizations," *Proc. Fourth Conf. on Architectural Support for Programming Languages and Operating Systems,* IEEE/ACM (April), Palo Alto, Calif., 310–319.

Bier, J. [1997]. "The evolution of DSP processors," presentation at U.C. Berkeley, November 14.

Boddie, J. R. [2000]. "History of DSPs," *www.lucent.com/micro/dsp/dsphist.html.*

Case, R. P., and A. Padegs [1978]. "The architecture of the IBM System/370," *Communications of the ACM* 21:1, 73–96.

Chow, F. C. [1983]. *A Portable Machine-Independent Global Optimizer—Design and Measurements,* Ph.D. thesis, Stanford Univ. (December).

Clark, D., and H. Levy [1982]. "Measurement and analysis of instruction set use in the VAX-11/780," *Proc. Ninth Symposium on Computer Architecture* (April), Austin, Tex., 9–17.

Clark, D., and W. D. Strecker [1980]. "Comments on 'the case for the reduced instruction set computer'," *Computer Architecture News* 8:6 (October), 34–38.

Crawford, J., and P. Gelsinger [1988]. *Programming the 80386,* Sybex Books, Alameda, Calif.

Darcy, J. D., and D. Gay [1996]. "FLECKmarks: Measuring floating point performance using a full IEEE compliant arithmetic benchmark," CS 252 class project, U.C. Berkeley (see *HTTP.CS.Berkeley.EDU/~darcy/Projects/cs252/*).

Digital Semiconductor [1996]. *Alpha Architecture Handbook, Version 3,* Digital Press, Maynard, Mass.

Ditzel, D. R., and D. A. Patterson [1980]. "Retrospective on high-level language computer architecture," *Proc. Seventh Annual Symposium on Computer Architecture,* La Baule, France (June), 97–104.

Emer, J. S., and D. W. Clark [1984]. "A characterization of processor performance in the VAX-11/780," *Proc. 11th Symposium on Computer Architecture* (June), Ann Arbor, Mich., 301–310.

Furber, S. B. [1996]. *ARM System Architecture*, Addison-Wesley, Harlow, England (see *www.cs.man.ac.uk/amulet/publications/books/ARMsysArch*).

Gagliardi, U. O. [1973]. "Report of workshop 4—software-related advances in computer hardware," *Proc. Symposium on the High Cost of Software*, Menlo Park, Calif., 99–120.

Game, M., and A. Booker [1999]. "CodePack code compression for PowerPC processors," *MicroNews,* 5:1, *www.chips.ibm.com/micronews/vol5_no1/codepack.html.*

Garner, R., A. Agarwal, F. Briggs, E. Brown, D. Hough, B. Joy, S. Kleiman, S. Muchnick, M. Namjoo, D. Patterson, J. Pendleton, and R. Tuck [1988]. "Scalable processor architecture (SPARC)," *COMPCON, IEEE* (March), San Francisco, 278–283.

Hauck, E. A., and B. A. Dent [1968]. "Burroughs' B6500/B7500 stack mechanism," *Proc. AFIPS SJCC*, 245–251.

Hennessy, J. [1984]. "VLSI processor architecture," *IEEE Trans. on Computers* C-33:11 (December), 1221–1246.

Hennessy, J. [1985]. "VLSI RISC processors," *VLSI Systems Design* 6:10 (October), 22–32.

Hennessy, J., N. Jouppi, F. Baskett, and J. Gill [1981]. "MIPS: A VLSI processor architecture," *Proc. CMU Conf. on VLSI Systems and Computations* (October), Computer Science Press, Rockville, Md.

Hewlett-Packard [1994]. *PA-RISC 2.0 Architecture Reference Manual,* 3rd ed.

Hitachi [1997]. *SuperH RISC Engine SH7700 Series Programming Manual* (see *www.halsp.hitachi.com/tech_prod/* and search for title).

IBM [1994]. *The PowerPC Architecture,* Morgan Kaufmann, San Francisco.

Intel [2001]. "Using MMX instructions to convert RGB to YUV color conversion," *cedar.intel.com/cgi-bin/ids.dll/content/content.jsp?cntKey=Legacy::irtm_AP548_9996&cntType=IDS_ EDITORIAL.*

Kahan, J. [1990]. "On the advantage of the 8087's stack," unpublished course notes, Computer Science Division, University of California at Berkeley.

Kane, G. [1986]. *MIPS R2000 RISC Architecture,* Prentice Hall, Englewood Cliffs, N.J.

Kane, G. [1996]. *PA-RISC 2.0 Architecture*, Prentice Hall PTR, Upper Saddle River, N.J.

Kane, G., and J. Heinrich [1992]. *MIPS RISC Architecture,* Prentice Hall, Englewood Cliffs, N.J.

Kissell, K. D. [1997]. *MIPS16: High-Density for the Embedded Market* (see *www.sgi.com/MIPS/arch/MIPS16/MIPS16.whitepaper.pdf*).

Kozyrakis, C. [2000]. "Vector IRAM: A media-oriented vector processor with embedded DRAM," presentation at Hot Chips 12 Conf., Palo Alto, Calif, 13–15.

Lee, R. [1989]. "Precision architecture," *Computer* 22:1 (January), 78–91.

Levy, H., and R. Eckhouse [1989]. *Computer Programming and Architecture: The VAX,* Digital Press, Boston.

Lindholm, T., and F. Yellin [1999]. *The Java Virtual Machine Specification,* second edition, Addison-Wesley, Reading, Mass. Also available online at *java.sun.com/docs/books/vmspec/.*

Lunde, A. [1977]. "Empirical evaluation of some features of instruction set processor architecture," *Comm. ACM* 20:3 (March), 143–152.

Magenheimer, D. J., L. Peters, K. W. Pettis, and D. Zuras [1988]. "Integer multiplication and division on the HP precision architecture," *IEEE Trans. on Computers* 37:8, 980–990.

McGhan, H., and M. O'Connor [1998]. "PicoJava: A direct execution engine for Java bytecode," *Computer* 31:10 (October), 22–30.

McKeeman, W. M. [1967]. "Language directed computer design," *Proc. 1967 Fall Joint Computer Conf.,* Washington, D.C., 413–417.

Meyers, G. J. [1978]. "The evaluation of expressions in a storage-to-storage architecture," *Computer Architecture News* 7:3 (October), 20–23.

Meyers, G. J. [1982]. *Advances in Computer Architecture*, second edition, Wiley, New York.

MIPS [1997]. *MIPS16 Application Specific Extension Product Description* (see *www.sgi.com/MIPS/arch/MIPS16/mips16.pdf*).

Mitsubishi [1996]. *Mitsubishi 32-Bit Single Chip Microcomputer M32R Family Software Manual* (September).

Morse, S., B. Ravenal, S. Mazor, and W. Pohlman [1980]. "Intel microprocessors—8080 to 8086," *Computer* 13:10 (October).

Moussouris, J., L. Crudele, D. Freitas, C. Hansen, E. Hudson, S. Przybylski, T. Riordan, and C. Rowen [1986]. "A CMOS RISC processor with integrated system functions," *Proc. COMPCON, IEEE* (March), San Francisco, 191.

Muchnick, S. S. [1988]. "Optimizing compilers for SPARC," *Sun Technology* 1:3 (Summer), 64–77.

Palmer, J., and S. Morse [1984]. *The 8087 Primer,* J. Wiley, New York, 93.

Patterson, D. [1985]. "Reduced instruction set computers," *Comm. ACM* 28:1 (January), 8–21.

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

Patterson, D. A., P. Garrison, M. Hill, D. Lioupis, C. Nyberg, T. Sippel, and K. Van Dyke [1983]. "Architecture of a VLSI instruction cache for a RISC," *10th Annual Int'l Conf. on Computer Architecture Conf. Proc.,* Stockholm, Sweden, June 13–16, 108–116.

Radin, G. [1982]. "The 801 minicomputer," *Proc. Symposium Architectural Support for Programming Languages and Operating Systems* (March), Palo Alto, Calif., 39–47.

Riemens, A., K. A. Vissers, R. J. Schutten, F. W. Sijstermans, G. J. Hekstra, and G. D. La Hei [1999]."Trimedia CPU64 application domain and benchmark suite," *Proc. 1999 IEEE Int'l Conf. on Computer Design: VLSI in Computers and Processors, ICCD'99,* Austin, Tex., October 10–13, 580–585.

Ropers, A., H. W. Lollman, and J. Wellhausen [1999]. "DSPstone: Texas Instruments TMS320C54x," Technical Report Nr. IB 315 1999/9-ISS-Version 0.9, Aachen University of Technology, *www.ert.rwth-aachen.de/Projekte/Tools/coal/dspstone_c54x/index.html.*

Shustek, L. J. [1978]. *Analysis and Performance of Computer Instruction Sets,* Ph.D. dissertation, Stanford University (January).

Silicon Graphics [1996]. *MIPS V Instruction Set* (see *http://www.sgi.com/MIPS/arch/ISA5/#MIPSV_indx*).

Sites, R. L., and R. Witek, eds. [1995]. *Alpha Architecture Reference Manual,* second edition, Digital Press, Newton, Mass.

Strauss, W. [1998]. "DSP strategies 2002," *Forward Concepts, www.usadata.com/market_research/spr_05/spr_r127-005.htm.*

Strecker, W. D. [1978]. "VAX-11/780: A virtual address extension of the PDP-11 family," *Proc. AFIPS National Computer Conf.* 47, 967–980.

Sun Microsystems [1989]. *The SPARC Architectural Manual,* Version 8, Part No. 800-1399-09, August 25.

Tanenbaum, A. S. [1978]. "Implications of structured programming for machine architecture," *Comm. ACM* 21:3 (March), 237–246.

Taylor, G., P. Hilfinger, J. Larus, D. Patterson, and B. Zorn [1986]. "Evaluation of the SPUR LISP architecture," *Proc. 13th Symposium on Computer Architecture* (June), Tokyo.

Texas Instruments [2000]. "History of innovation: 1980s," *www.ti.com/corp/docs/company /history/1980s.shtml.*

Thornton, J. E. [1964]. "Parallel operation in Control Data 6600," *Proc. AFIPS Fall Joint Computer Conf., Part II,* 26, 33–40.

Ungar, D., R. Blau, P. Foley, D. Samples, and D. Patterson [1984]. "Architecture of SOAR: Smalltalk on a RISC," *Proc. 11th Symposium on Computer Architecture* (June), Ann Arbor, Mich., 188–197.

van Eijndhoven, J. T. J., F. W. Sijstermans, K. A. Vissers, E. J. D. Pol, M. I. A. Tromp, P. Struik, R. H. J. Bloks, P. van der Wolf, A. D. Pimentel, and H. P. E. Vranken [1999]. "Trimedia CPU64 architecture," *Proc. 1999 IEEE Int'l Conf. on Computer Design: VLSI in Computers and Processors, ICCD'99,* Austin, Tex., October 10–13, 586–592.

Wakerly, J. [1989]. *Microcomputer Architecture and Programming,* Wiley, New York.

Waters, F., ed. [1986]. *IBM RT Personal Computer Technology,* IBM, Austin, Tex., SA 23-1057.

Weaver, D. L., and T. Germond [1994]. *The SPARC Architectural Manual,* Version 9, Prentice Hall, Englewood Cliffs, N.J.

Weiss, S., and J. E. Smith [1994]. *Power and PowerPC,* Morgan Kaufmann, San Francisco.

Wiecek, C. [1982]. "A case study of the VAX 11 instruction set usage for compiler execution," *Proc. Symposium on Architectural Support for Programming Languages and Operating Systems* (March), IEEE/ACM, Palo Alto, Calif., 177–184.

Wulf, W. [1981]. "Compilers and computer architecture," *Computer* 14:7 (July), 41–47.

K.4 The Development of Pipelining and Instruction-Level Parallelism (Chapters 2 and 3; Appendices A and G)

Early Pipelined CPUs

The first general-purpose pipelined processor is considered to be Stretch, the IBM 7030. Stretch followed the IBM 704 and had a goal of being 100 times faster than the 704. The goal was a stretch from the state of the art at that time—hence the nickname. The plan was to obtain a factor of 1.6 from overlapping fetch, decode, and execute, using a four-stage pipeline. Bloch [1959] and Bucholtz [1962] describe the design and engineering trade-offs, including the use of ALU bypasses.

A series of general pipelining descriptions that appeared in the late 1970s and early 1980s provided most of the terminology and described most of the basic techniques used in simple pipelines. These surveys include Keller [1975], Ramamoorthy and Li [1977], Chen [1980], and Kogge's book [1981], devoted entirely to pipelining. Davidson and his colleagues [1971, 1975] developed the concept of pipeline reservation tables as a design methodology for multicycle pipelines with feedback (also described in Kogge [1981]). Many designers use a variation of these concepts, in either designing pipelines or in creating software to schedule them.

The RISC processors were originally designed with ease of implementation and pipelining in mind. Several of the early RISC papers, published in the early 1980s, attempt to quantify the performance advantages of the simplification in instruction set. The best analysis, however, is a comparison of a VAX and a MIPS implementation published by Bhandarkar and Clark in 1991, 10 years after the first published RISC papers (see Figure K.1). After 10 years of arguments about the implementation benefits of RISC, this paper convinced even the most skeptical designers of the advantages of a RISC instruction set architecture.

J. E. Smith and his colleagues have written a number of papers examining instruction issue, exception handling, and pipeline depth for high-speed scalar CPUs. Kunkel and Smith [1986] evaluate the impact of pipeline overhead and dependences on the choice of optimal pipeline depth; they also have an excellent discussion of latch design and its impact on pipelining. Smith and Pleszkun [1988] evaluate a variety of techniques for preserving precise exceptions. Weiss and Smith [1984] evaluate a variety of hardware pipeline scheduling and instruction issue techniques.

The MIPS R4000 was one of the first deeply pipelined microprocessors and is described by Killian [1991] and by Heinrich [1993]. The initial Alpha implementation (the 21064) has a similar instruction set and similar integer pipeline structure, with more pipelining in the floating-point unit.

The Introduction of Dynamic Scheduling

In 1964 CDC delivered the first CDC 6600. The CDC 6600 was unique in many ways. In addition to introducing scoreboarding, the CDC 6600 was the first processor to make extensive use of multiple functional units. It also had peripheral processors that used multithreading. The interaction between pipelining and instruction set design was understood, and a simple, load-store instruction set was used to promote pipelining. The CDC 6600 also used an advanced packaging technology. Thornton [1964] describes the pipeline and I/O processor architecture, including the concept of out-of-order instruction execution. Thornton's book [1970] provides an excellent description of the entire processor, from technology to architecture, and includes a foreword by Cray. (Unfortunately, this book is currently out of print.) The CDC 6600 also has an instruction scheduler for the FORTRAN compilers, described by Thorlin [1967].

The IBM 360 Model 91: A Landmark Computer

The IBM 360/91 introduced many new concepts, including tagging of data, register renaming, dynamic detection of memory hazards, and generalized forwarding. Tomasulo's algorithm is described in his 1967 paper. Anderson, Sparacio, and Tomasulo [1967] describe other aspects of the processor, including the use of branch prediction. Many of the ideas in the 360/91 faded from use for nearly 25 years before being broadly resurrected in the 1990s. Unfortunately, the 360/91 was not successful, and only a handful were sold. The complexity of the design made it late to the market and allowed the Model 85, which was the first IBM processor with a cache, to outperform the 91.

Branch-Prediction Schemes

The 2-bit dynamic hardware branch-prediction scheme was described by J. E. Smith [1981]. Ditzel and McLellan [1987] describe a novel branch-target buffer for CRISP, which implements branch folding. The correlating predictor we examine was described by Pan, So, and Rameh [1992]. Yeh and Patt [1992, 1993] generalized the correlation idea and described multilevel predictors that use branch histories for each branch, similar to the local history predictor used in the 21264. McFarling's tournament prediction scheme, which he refers to as a combined predictor, is described in his 1993 technical report. There are a variety of more recent papers on branch prediction based on variations in the multilevel and correlating predictor ideas. Kaeli and Emma [1991] describe return address prediction. Evers et al. [1998] is an in-depth analysis of multilevel predictors. The data shown in Chapter 2 is from Skadron et al. [1999]. There are several schemes for prediction that may offer some additional benefit beyond tournament predictors. Eden and Mudge [1998] and Jimenez and Lin [2002] describe such approaches.

The Development of Multiple-Issue Processors

IBM did pioneering work on multiple issue. In the 1960s, a project called ACS was under way in California. It included multiple-issue concepts, a proposal for dynamic scheduling (although with a simpler mechanism than Tomasulo's scheme, which used backup registers), and fetching down both branch paths. The project originally started as a new architecture to follow Stretch and surpass the CDC 6600/6800. ACS started in New York but was moved to California, later changed to be S/360 compatible, and eventually canceled. John Cocke was one of the intellectual forces behind the team that included a number of IBM veterans and younger contributors, many of whom went on to other important roles in IBM and elsewhere: Jack Bertram, Ed Sussenguth, Gene Amdahl, Herb Schorr, Fran Allen, Lynn Conway, and Phil Dauber, among others. While the compiler team published many of their ideas and had great influence outside IBM, the architecture ideas were not widely disseminated at that time. The most complete

accessible documentation of this important project is at *www.cs.clemson.edu/~mark/acs.html,* which includes interviews with the ACS veterans and pointers to other sources. Sussenguth [1999] is a good overview of ACS.

Most of the early multiple-issue processors that actually reached the market followed an LIW or VLIW design approach. Charlesworth [1981] reports on the Floating Point Systems AP-120B, one of the first wide-instruction processors containing multiple operations per instruction. Floating Point Systems applied the concept of software pipelining both in a compiler and by handwriting assembly language libraries to use the processor efficiently. Since the processor was an attached processor, many of the difficulties of implementing multiple issue in general-purpose processors, for example, virtual memory and exception handling, could be ignored.

One of the interesting approaches used in early VLIW processors, such as the AP-120B and i860, was the idea of a pipeline organization that requires operations to be "pushed through" a functional unit and the results to be caught at the end of the pipeline. In such processors, operations advance only when another operation pushes them from behind (in sequence). Furthermore, an instruction specifies the destination for an instruction issued earlier that will be pushed out of the pipeline when this new operation is pushed in. Such an approach has the advantage that it does not specify a result destination when an operation first issues but only when the result register is actually written. This separation eliminates the need to detect WAW and WAR hazards in the hardware. The disadvantage is that it increases code size since no-ops may be needed to push results out when there is a dependence on an operation that is still in the pipeline and no other operations of that type are immediately needed. Instead of the "push-and-catch" approach used in these two processors, almost all designers have chosen to use *self-draining pipelines* that specify the destination in the issuing instruction and in which an issued instruction will complete without further action. The advantages in code density and simplifications in code generation seem to outweigh the advantages of the more unusual structure.

Several research projects introduced some form of multiple issue in the mid-1980s. For example, the Stanford MIPS processor had the ability to place two operations in a single instruction, although this capability was dropped in commercial variants of the architecture, primarily for performance reasons. Along with his colleagues at Yale, Fisher [1983] proposed creating a processor with a very wide instruction (512 bits) and named this type of processor a VLIW. Code was generated for the processor using trace scheduling, which Fisher [1981] had developed originally for generating horizontal microcode. The implementation of trace scheduling for the Yale processor is described by Fisher et al. [1984] and by Ellis [1986].

Although IBM canceled ACS, active research in the area continued in the 1980s. More than 10 years after ACS was canceled, John Cocke made a new proposal for a superscalar processor that dynamically made issue decisions; he and Tilak Agerwala described the key ideas in several talks in the mid-1980s and coined the term *superscalar.* He called the design America; it is described by

Agerwala and Cocke [1987]. The IBM Power1 architecture (the RS/6000 line) is based on these ideas (see Bakoglu et al. [1989]).

J. E. Smith [1984] and his colleagues at Wisconsin proposed the decoupled approach that included multiple issue with limited dynamic pipeline scheduling. A key feature of this processor is the use of queues to maintain order among a class of instructions (such as memory references) while allowing it to slip behind or ahead of another class of instructions. The Astronautics ZS-1 described by Smith et al. [1987] embodies this approach with queues to connect the load-store unit and the operation units. The Power2 design uses queues in a similar fashion. J. E. Smith [1989] also describes the advantages of dynamic scheduling and compares that approach to static scheduling.

The concept of speculation has its roots in the original 360/91, which performed a very limited form of speculation. The approach used in recent processors combines the dynamic scheduling techniques of the 360/91 with a buffer to allow in-order commit. Smith and Pleszkun [1988] explored the use of buffering to maintain precise interrupts and described the concept of a reorder buffer. Sohi [1990] describes adding renaming and dynamic scheduling, making it possible to use the mechanism for speculation. Patt and his colleagues were early proponents of aggressive reordering and speculation. They focused on checkpoint and restart mechanisms and pioneered an approach called HPSm, which is also an extension of Tomasulo's algorithm [Hwu and Patt 1986].

The use of speculation as a technique in multiple-issue processors was evaluated by Smith, Johnson, and Horowitz [1989] using the reorder buffer technique; their goal was to study available ILP in nonscientific code using speculation and multiple issue. In a subsequent book, Johnson [1990] describes the design of a speculative superscalar processor. Johnson later led the AMD K-5 design, one of the first speculative superscalars.

In parallel with the superscalar developments, commercial interest in VLIW approaches also increased. The Multiflow processor (see Colwell et al. [1987]) was based on the concepts developed at Yale, although many important refinements were made to increase the practicality of the approach. Among these was a controllable store buffer that provided support for a form of speculation. Although more than 100 Multiflow processors were sold, a variety of problems, including the difficulties of introducing a new instruction set from a small company and the competition provided from commercial RISC microprocessors that changed the economics in the minicomputer market, led to the failure of Multiflow as a company.

Around the same time as Multiflow, Cydrome was founded to build a VLIW-style processor (see Rau et al. [1989]), which was also unsuccessful commercially. Dehnert, Hsu, and Bratt [1989] explain the architecture and performance of the Cydrome Cydra 5, a processor with a wide-instruction word that provides dynamic register renaming and additional support for software pipelining. The Cydra 5 is a unique blend of hardware and software, including conditional instructions and register rotation, aimed at extracting ILP. Cydrome relied on more hardware than the Multiflow processor and achieved competitive perfor-

mance primarily on vector-style codes. In the end, Cydrome suffered from problems similar to those of Multiflow and was not a commercial success. Both Multiflow and Cydrome, although unsuccessful as commercial entities, produced a number of people with extensive experience in exploiting ILP as well as advanced compiler technology; many of those people have gone on to incorporate their experience and the pieces of the technology in newer processors. Fisher and Rau [1993] edited a comprehensive collection of papers covering the hardware and software of these two important processors.

Rau had also developed a scheduling technique called *polycyclic scheduling,* which is a basis for most software-pipelining schemes (see Rau, Glaeser, and Picard [1982]). Rau's work built on earlier work by Davidson and his colleagues on the design of optimal hardware schedulers for pipelined processors. Other historical LIW processors have included the Apollo DN 10000 and the Intel i860, both of which could dual-issue FP and integer operations.

Compiler Technology and Hardware Support for Scheduling

Loop-level parallelism and dependence analysis was developed primarily by D. Kuck and his colleagues at the University of Illinois in the 1970s. They also coined the commonly used terminology of *antidependence* and *output dependence* and developed several standard dependence tests, including the GCD and Banerjee tests. The latter test was named after Uptal Banerjee and comes in a variety of flavors. Recent work on dependence analysis has focused on using a variety of exact tests ending with a linear programming algorithm called Fourier-Motzkin. D. Maydan and W. Pugh both showed that the sequences of exact tests were a practical solution.

In the area of uncovering and scheduling ILP, much of the early work was connected to the development of VLIW processors, described earlier. Lam [1988] developed algorithms for software pipelining and evaluated their use on Warp, a wide-instruction-word processor designed for special-purpose applications. Weiss and Smith [1987] compare software pipelining versus loop unrolling as techniques for scheduling code on a pipelined processor. Rau [1994] developed modulo scheduling to deal with the issues of software-pipelining loops and simultaneously handling register allocation.

Support for speculative code scheduling was explored in a variety of contexts, including several processors that provided a mode in which exceptions were ignored, allowing more aggressive scheduling of loads (e.g., the MIPS TFP processor [Hsu 1994]). Several groups explored ideas for more aggressive hardware support for speculative code scheduling. For example, Smith, Horowitz, and Lam [1992] created a concept called boosting that contains a hardware facility for supporting speculation but provides a checking and recovery mechanism, similar to those in IA-64 and Crusoe. The sentinel scheduling idea, which is also similar to the speculate-and-check approach used in both Crusoe and the IA-64 architectures, was developed jointly by researchers at the University of Illinois and HP Laboratories (see Mahlke et al. [1992]).

In the early 1990s, Wen-Mei Hwu and his colleagues at the University of Illinois developed a compiler framework, called IMPACT (see Chang et al. [1991]), for exploring the interaction between multiple-issue architectures and compiler technology. This project led to several important ideas, including superblock scheduling (see Hwu et al. [1993]); extensive use of profiling for guiding a variety of optimizations (e.g., procedure inlining); and the use of a special buffer (similar to the ALAT or program-controlled store buffer) for compile-aided memory conflict detection (see Gallagher et al. [1994]). They also explored the performance trade-offs between partial and full support for predication in Mahlke et al. [1995].

The early RISC processors all had delayed branches, a scheme inspired from microprogramming, and several studies on compile time branch prediction were inspired by delayed branch mechanisms. McFarling and Hennessy [1986] did a quantitative comparison of a variety of compile time and run time branch-prediction schemes. Fisher and Freudenberger [1992] evaluated a range of compile time branch-prediction schemes using the metric of distance between mispredictions. Ball and Larus [1993] and Calder et al. [1997] describe static prediction schemes using collected program behavior.

EPIC and the IA-64 Development

The roots of the EPIC approach lie in earlier attempts to build LIW and VLIW machines—especially those at Cydrome and Multiflow—and in a long history of compiler work that continued after these companies failed at HP, the University of Illinois, and elsewhere. Insights gained from that work led designers at HP to propose a VLIW-style, 64-bit architecture to follow the HP PA RISC architecture. Intel was looking for a new architecture to replace the x86 (now called IA-32) architecture and to provide 64-bit capability. In 1995, they formed a partnership to design a new architecture, IA-64 (see Huck et al. [2000]), and build processors based on it. Itanium (see Sharangpani and Arora [2000]) is the first such processor. In 2002, Intel introduced the second-generation IA-64 design, the Itanium 2 (see McNairy and Soltis [2003] and McCormick and Knies [2002]).

Studies of ILP and Ideas to Increase ILP

A series of early papers, including Tjaden and Flynn [1970] and Riseman and Foster [1972], concluded that only small amounts of parallelism could be available at the instruction level without investing an enormous amount of hardware. These papers dampened the appeal of multiple instruction issue for more than 10 years. Nicolau and Fisher [1984] published a paper based on their work with trace scheduling and asserted the presence of large amounts of potential ILP in scientific programs.

Since then there have been many studies of the available ILP. Such studies have been criticized since they presume some level of both hardware support and compiler technology. Nonetheless, the studies are useful to set expectations as

well as to understand the sources of the limitations. Wall has participated in several such studies, including Jouppi and Wall [1989], Wall [1991], and Wall [1993]. Although the early studies were criticized as being conservative (e.g., they didn't include speculation), the last study is by far the most ambitious study of ILP to date and the basis for the data in Section 3.2. Sohi and Vajapeyam [1989] give measurements of available parallelism for wide-instruction-word processors. Smith, Johnson, and Horowitz [1989] also used a speculative superscalar processor to study ILP limits. At the time of their study, they anticipated that the processor they specified was an upper bound on reasonable designs. Recent and upcoming processors, however, are likely to be at least as ambitious as their processor. Skadron et al. [1999] examine the performance trade-offs and limitations in a processor comparable to the most aggressive processors in 2005, concluding the larger window sizes will not make sense without significant improvements on branch prediction for integer programs.

Lam and Wilson [1992] looked at the limitations imposed by speculation and showed that additional gains are possible by allowing processors to speculate in multiple directions, which requires more than one PC. (Such schemes cannot exceed what perfect speculation accomplishes, but they help close the gap between realistic prediction schemes and perfect prediction.) Wall's 1993 study includes a limited evaluation of this approach (up to eight branches are explored).

Going Beyond the Data Flow Limit

One other approach that has been explored in the literature is the use of value prediction. Value prediction can allow speculation based on data values. There have been a number of studies of the use of value prediction. Lipasti and Shen published two papers in 1996 evaluating the concept of value prediction and its potential impact on ILP exploitation. Calder, Reinman, and Tullsen [1999] explored the idea of selective value prediction. Sodani and Sohi [1997] approach the same problem from the viewpoint of reusing the values produced by instructions. Moshovos et al. [1997] show that deciding when to speculate on values, by tracking whether such speculation has been accurate in the past, is important to achieving performance gains with value speculation. Moshovos and Sohi [1997] and Chrysos and Emer [1998] focus on predicting memory dependences and using this information to eliminate the dependence through memory. González and González [1998], Babbay and Mendelson [1998], and Calder, Reinman, and Tullsen [1999] are more recent studies of the use of value prediction. This area is currently highly active, with new results being published in every conference.

Recent Advanced Microprocessors

The years 1994–95 saw the announcement of wide superscalar processors (three or more issues per clock) by every major processor vendor: Intel Pentium Pro and Pentium II (these processors share the same core pipeline architecture, described by Colwell and Steck [1995]); AMD K-5, K-6, and Athlon; Sun UltraSPARC

(see Lauterbach and Horel [1999]); Alpha 21164 (see Edmondson et al. [1995]) and 21264 (see Kessler [1999]); MIPS R10000 and R12000 (see Yeager et al. [1996]); PowerPC 603, 604, 620 (see Diep, Nelson, and Shen [1995]); and HP 8000 (Kumar [1997]). The latter part of the decade (1996–2000) saw second generations of much of these processors (Pentium III, AMD Athlon, Alpha 21264, among others). The second generation, although similar in issue rate, could sustain a lower CPI and provided much higher clock rates. All included dynamic scheduling, and almost universally supported speculation. In practice, many factors, including the implementation technology, the memory hierarchy, the skill of the designers, and the type of applications benchmarked, all play a role in determining which approach is best.

The period from 2000 to 2005 was dominated by three trends among superscalar processors: the introduction of higher clock rates achieved through deeper pipelining (e.g., in the Pentium 4 (Hinton et al. [2001]); the introduction of multithreading by IBM in the Power 4 and by Intel in the Pentium 4 Extreme; and the beginning of the movement to multicore by IBM in the Power 4, AMD in Opteron (see Keltcher et al. [2003]), and most recently by Intel (see Douglas 2005).

Multithreading and Simultaneous Multithreading

The concept of multithreading dates back to one of the earliest transistorized computers, the TX-2. TX-2 is also famous for being the computer on which Ivan Sutherland created Sketchpad, the first computer graphics system. TX-2 was built at MIT's Lincoln Laboratory and became operational in 1959. It used multiple threads to support fast context switching to handle I/O functions. Clark [1957] describes the basic architecture, and Forgie [1957] describes the I/O architecture. Multithreading was also used in the CDC 6600, where a fine-grained multithreading scheme with interleaved scheduling among threads was used as the architecture of the I/O processors. The HEP processor, a pipelined multiprocessor designed by Denelcor and shipped in 1982, used fine-grained multithreading to hide the pipeline latency as well as to hide the latency to a large memory shared among all the processors. Because the HEP had no cache, this hiding of memory latency was critical. Burton Smith, one of the primary architects, describes the HEP architecture in a 1978 paper, and Jordan [1983] published a performance evaluation. The TERA processor extends the multithreading ideas and is described by Alverson et al. in a 1992 paper. The Niagara multithreading approach is similar to those of the HEP and TERA systems, although Niagara employs caches reducing the need for thread-based latency hiding.

In the late 1980s and early 1990s, researchers explored the concept of coarse-grained multithreading (also called block multithreading) as a way to tolerate latency, especially in multiprocessor environments. The SPARCLE processor in the Alewife system used such a scheme, switching threads whenever a high-latency exceptional event, such as a long cache miss, occurred. Agarwal et al. describe SPARCLE in a 1993 paper. The IBM Pulsar processor uses similar ideas.

By the early 1990s, several research groups had arrived at two key insights. First, they realized that fine-grained multithreading was needed to get the maximum performance benefit, since in a coarse-grained approach, the overhead of thread switching and thread start-up (e.g., filling the pipeline from the new thread) negated much of the performance advantage (see Laudon, Gupta, and Horowitz [1994]). Second, several groups realized that to effectively use large numbers of functional units would require both ILP and thread-level parallelism (TLP). These insights led to several architectures that used combinations of multithreading and multiple issue. Wolfe and Shen [1991] describe an architecture called XIMD that statically interleaves threads scheduled for a VLIW processor. Hirata et al. [1992] describe a proposed processor for media use that combines a static superscalar pipeline with support for multithreading; they report speedups from combining both forms of parallelism. Keckler and Dally [1992] combine static scheduling of ILP and dynamic scheduling of threads for a processor with multiple functional units. The question of how to balance the allocation of functional units between ILP and TLP and how to schedule the two forms of parallelism remained open.

When it became clear in the mid-1990s that dynamically scheduled superscalars would be delivered shortly, several research groups proposed using the dynamic scheduling capability to mix instructions from several threads on the fly. Yamamoto et al. [1994] appears to be the first such proposal, though the simulation results for their multithreaded superscalar architecture use simplistic assumptions. This work was quickly followed by Tullsen, Eggers, and Levy [1995], which was the first realistic simulation assessment and coined the term *simultaneous multithreading*. Subsequent work by the same group together with industrial coauthors addressed many of the open questions about SMT. For example, Tullsen et al. [1996] addressed questions about the challenges of scheduling ILP versus TLP. Lo et al. [1997] is an extensive discussion of the SMT concept and an evaluation of its performance potential, and Lo et. al. [1998] evaluates database performance on an SMT processor. Tuck and Tullsen [2003] review the performance of SMT on the Pentium 4.

The IBM Power4 introduced multithreading (see Tendler et al. [2002]), while the Power5 used simultaneous multithreading. Mathis et al. [2005] explores the performance of SMT in the Power5, while Sinharoy et al. [2005] describes the system architecture.

References

Agarwal, A., J. Kubiatowicz, D. Kranz, B.-H. Lim, D. Yeung, G. D'Souza, and M. Parkin [1993]. "Sparcle: An evolutionary processor design for large-scale multiprocessors," *IEEE Micro* 13 (June), 48–61.

Agerwala, T., and J. Cocke [1987]. "High performance reduced instruction set processors," IBM Tech. Rep. (March).

Alverson, G., R. Alverson, D. Callahan, B. Koblenz, A. Porterfield, and B. Smith [1992]. "Exploiting heterogeneous parallelism on a multithreaded multiprocessor," *Proc. 1992 Int'l Conf. on Supercomputing* (November), 188–197.

Anderson, D. W., F. J. Sparacio, and R. M. Tomasulo [1967]. "The IBM 360 Model 91: Processor philosophy and instruction handling," *IBM J. Research and Development* 11:1 (January), 8–24.

Austin, T. M., and G. Sohi [1992]. "Dynamic dependency analysis of ordinary programs," *Proc. 19th Symposium on Computer Architecture* (May), Gold Coast, Australia, 342–351.

Babbay, F., and A. Mendelson [1998]. "Using value prediction to increase the power of speculative execution hardware," *ACM Trans. on Computer Systems* 16:3 (August), 234–270.

Bakoglu, H. B., G. F. Grohoski, L. E. Thatcher, J. A. Kaeli, C. R. Moore, D. P. Tattle, W. E. Male, W. R. Hardell, D. A. Hicks, M. Nguyen Phu, R. K. Montoye, W. T. Glover, and S. Dhawan [1989]. "IBM second-generation RISC processor organization," *Proc. Int'l Conf. on Computer Design,* IEEE (October), Rye, N.Y., 138–142.

Ball, T., and J. Larus [1993]. "Branch prediction for free," *Proc. SIGPLAN'93 Conference Programming Language Design and Implementation,* 300–313.

Bhandarkar, D., and D. W. Clark [1991]. "Performance from architecture: Comparing a RISC and a CISC with similar hardware organizations," *Proc. Fourth Conf. on Architectural Support for Programming Languages and Operating Systems*, IEEE/ACM (April), Palo Alto, Calif., 310–319.

Bhandarkar, D., and J. Ding [1997]. "Performance characterization of the Pentium Pro processor," *Proc. Third Int'l Symposium on High Performance Computer Architecture,* IEEE (February), San Antonio, 288–297.

Bloch, E. [1959]. "The engineering design of the Stretch computer," *1959 Proceedings of the Eastern Joint Computer Conf.,* 48–59.

Bucholtz, W. [1962]. *Planning a Computer System: Project Stretch,* McGraw-Hill, New York.

Calder, B., D. Grunwald, M. Jones, D. Lindsay, J. Martin, M. Mozer, and B. Zorn [1997]. "Evidence-based static branch prediction using machine learning," *ACM Trans. Program. Lang. Syst.* 19:1, 188–222.

Calder, B., G. Reinman, and D. M. Tullsen [1999]. "Selective value prediction," *Proc. 26th Int'l Symposium on Computer Architecture (ISCA),* Atlanta, June.

Chang, P. P., S. A. Mahlke, W. Y. Chen, N. J. Warter, and W. W. Hwu [1991]. "IMPACT: An architectural framework for multiple-instruction-issue processors," *Proc. 18th Int'l Symposium on Computer Architecture* (May), 266–275.

Charlesworth, A. E. [1981]. "An approach to scientific array processing: The architecture design of the AP-120B/FPS-164 family," *Computer* 14:9 (September), 18–27.

Chen, T. C. [1980]. "Overlap and parallel processing," in *Introduction to Computer Architecture,* H. Stone, ed., Science Research Associates, Chicago, 427–486.

Chrysos, G. Z., and J. S. Emer [1998]. "Memory dependence prediction using store sets," *Proc. 25th Int'l Symposium on Computer Architecture (ISCA),* June, Barcelona, 142–153.

Clark, D. W. [1987]. "Pipelining and performance in the VAX 8800 processor," *Proc. Second Conf. on Architectural Support for Programming Languages and Operating Systems,* IEEE/ACM (March), Palo Alto, Calif., 173–177.

Clark, W. A. [1957]. "The Lincoln TX-2 computer development," *Proc. Western Joint Computer Conference* (February), Institute of Radio Engineers, Los Angeles, 143–145.

Colwell, R. P., R. P. Nix, J. J. O'Donnell, D. B. Papworth, and P. K. Rodman [1987]. "A VLIW architecture for a trace scheduling compiler," *Proc. Second Conf. on Architectural Support for Programming Languages and Operating Systems,* IEEE/ACM (March), Palo Alto, Calif., 180–192.

Colwell, R. P., and R. Steck [1995]. "A 0.6um BiCMOS processor with dynamic execution." *Proc. of Int'l Symposium on Solid State Circuits,* 176–177.

Cvetanovic, Z., and R. E. Kessler [2000]. "Performance analysis of the Alpha 21264-based Compaq ES40 system," *Proc. 27th Symposium on Computer Architecture* (June), Vancouver, Canada, 192–202.

Davidson, E. S. [1971]. "The design and control of pipelined function generators," *Proc. Conf. on Systems, Networks, and Computers,* IEEE (January), Oaxtepec, Mexico, 19–21.

Davidson, E. S., A. T. Thomas, L. E. Shar, and J. H. Patel [1975]. "Effective control for pipelined processors," *COMPCON, IEEE* (March), San Francisco, 181–184.

Dehnert, J. C., P. Y.-T. Hsu, and J. P. Bratt [1989]. "Overlapped loop support on the Cydra 5," *Proc. Third Conf. on Architectural Support for Programming Languages and Operating Systems* (April), IEEE/ACM, Boston, 26–39.

Diep, T. A., C. Nelson, and J. P. Shen [1995]. "Performance evaluation of the PowerPC 620 microarchitecture," *Proc. 22nd Symposium on Computer Architecture* (June), Santa Margherita, Italy.

Ditzel, D. R., and H. R. McLellan [1987]. "Branch folding in the CRISP microprocessor: Reducing the branch delay to zero," *Proc. 14th Symposium on Computer Architecture* (June), Pittsburgh, 2–7.

Douglas, J. [2005]. "Intel 8xx series and Paxville Xeon-MP Microprocessors," *Proc. Hot Chips,* Stanford University, August.

Eden, A., and T. Mudge [1998]. "The YAGS branch prediction scheme," *Proc. of the 31st Annual ACM/IEEE International Symposium on Microarchitecture,* 69–80.

Edmondson, J. H., P. I. Rubinfield, R. Preston, and V. Rajagopalan [1995]. "Superscalar instruction execution in the 21164 Alpha microprocessor," *IEEE Micro* 15:2, 33–43.

Ellis, J. R. [1986]. *Bulldog: A Compiler for VLIW Architectures*, MIT Press, Cambridge, Mass.

Emer, J. S., and D. W. Clark [1984]. "A characterization of processor performance in the VAX-11/780," *Proc. 11th Symposium on Computer Architecture* (June), Ann Arbor, Mich., 301–310.

Evers, M., S. J. Patel, R. S. Chappell, and Y. N. Patt [1998]. "An analysis of correlation and predictability: What makes two-level branch predictors work," *Proc. 25th Annual Int. Sym. Comp. Arch. (ISCA)*, 52–61.

Fisher, J. A. [1981]. "Trace scheduling: A technique for global microcode compaction," *IEEE Trans. on Computers* 30:7 (July), 478–490.

Fisher, J. A. [1983]. "Very long instruction word architectures and ELI-512," *Proc. Tenth Symposium on Computer Architecture* (June), Stockholm, 140–150.

Fisher, J. A., J. R. Ellis, J. C. Ruttenberg, and A. Nicolau [1984]. "Parallel processing: A smart compiler and a dumb processor," *Proc. SIGPLAN Conf. on Compiler Construction* (June), Palo Alto, Calif., 11–16.

Fisher, J. A., and S. M. Freudenberger [1992]. "Predicting conditional branches from previous runs of a program," *Proc. Fifth Conf. on Architectural Support for Programming Languages and Operating Systems,* IEEE/ACM (October), Boston, 85–95.

Fisher, J. A., and B. R. Rau [1993]. *Journal of Supercomputing (*January*),* Kluwer.

Forgie, J. W. [1957]. "The Lincoln TX-2 input-output system," *Proc. Western Joint Computer Conference* (February), Institute of Radio Engineers, Los Angeles, 156–160.

Foster, C. C., and E. M. Riseman [1972]. "Percolation of code to enhance parallel dispatching and execution," *IEEE Trans. on Computers* C-21:12 (December), 1411–1415.

Gallagher, D. M., W. Y. Chen, S. A. Mahlke, J. C. Gyllenhaal, and W.W. Hwu [1994]. "Dynamic memory disambiguation using the memory conflict buffer," *Proc. Sixth Int'l Conf. on Architectural Support for Programming Languages and Operating Systems* (October), Santa Clara, Calif., 183–193.

González, J., and A. González [1998]. "Limits of instruction level parallelism with data speculation," *Proc. of the VECPAR Conf.,* 585–598.

Heinrich, J. [1993]. *MIPS R4000 User's Manual,* Prentice Hall, Englewood Cliffs, N.J.

Hinton, G., D. Sager, M. Upton, D. Boggs, D. Carmean, A. Kyker, and P. Roussel [2001]. "The microarchitecture of the Pentium 4 processor," *Intel Technology Journal*, February.

Hirata, H., K. Kimura, S. Nagamine, Y. Mochizuki, A. Nishimura, Y. Nakase, and T. Nishizawa [1992]. "An elementary processor architecture with simultaneous instruction issuing from multiple threads," *Proc. 19th Annual Int'l Symposium on Computer Architecture* (May), 136–145.

Hopkins, M. [2000]. "A critical look at IA-64: Massive resources, massive ILP, but can it deliver?" *Microprocessor Report* (February).

Hsu, P. [1994]. "Designing the TFP microprocessor," *IEEE Micro* 18:2 (April), 2333.

Huck, J., et al. [2000]. "Introducing the IA-64 Architecture" *IEEE Micro*, 20:5, (September–October), 12–23.

Hwu, W. W., S. A. Mahlke, W. Y. Chen, P. P. Chang, N. J. Warter, R. A. Bringmann, R. O. Ouellette, R. E. Hank, T. Kiyohara, G. E. Haab, J. G. Holm, and D. M. Lavery [1993]. "The superblock: An effective technique for VLIW and superscalar compilation," *J. Supercomputing* 7:1, 2 (March), 229–248.

Hwu, W.-M., and Y. Patt [1986]. "HPSm, a high performance restricted data flow architecture having minimum functionality," *Proc. 13th Symposium on Computer Architecture* (June), Tokyo, 297–307.

IBM [1990]. "The IBM RISC System/6000 processor" (collection of papers), *IBM J. Research and Development* 34:1 (January).

Jimenez, D. A., and C. Lin [2002]. "Neural methods for dynamic branch prediction," *ACM Trans. Computer Sys* 20:4, (November), 369–397.

Johnson, M. [1990]. *Superscalar Microprocessor Design*, Prentice Hall, Englewood Cliffs, N.J.

Jordan, H. F. [1983]. "Performance measurements on HEP—a pipelined MIMD computer," *Proc. 10th Int'l Symposium on Computer Architecture* (June), Stockholm, 207– 212.

Jouppi, N. P., and D. W. Wall [1989]. "Available instruction-level parallelism for superscalar and superpipelined processors," *Proc. Third Conf. on Architectural Support for Programming Languages and Operating Systems,* IEEE/ACM (April), Boston, 272– 282.

Kaeli, D. R., and P. G. Emma [1991]. "Branch history table prediction of moving target branches due to subroutine returns," *Proc. 18th Int'l Symposium on Computer Architecture (ISCA),* Toronto, May, 34–42.

Keckler, S. W., and W. J. Dally [1992]. "Processor coupling: Integrating compile time and runtime scheduling for parallelism," *Proc. 19th Annual Int'l Symposium on Computer Architecture* (May), 202–213.

Keller, R. M. [1975]. "Look-ahead processors," *ACM Computing Surveys* 7:4 (December), 177–195.

Keltcher, C. N., K. J. McGrath, A. Ahmed, and P. Conway [2003]. "The AMD Opteron processor for multiprocessor servers," *IEEE Micro* 23:2 (March–April), 66–76.

Kessler, R. [1999]. "The Alpha 21264 microprocessor," *IEEE Micro* 19:2 (March/April) 24–36.

Killian, E. [1991]. "MIPS R4000 technical overview–64 bits/100 MHz or bust," *Hot Chips III Symposium Record* (August), Stanford University, 1.6–1.19.

Kogge, P. M. [1981]. *The Architecture of Pipelined Computers,* McGraw-Hill, New York.

Kumar, A. [1997]. "The HP PA-8000 RISC CPU," *IEEE Micro* 17:2 (March/April).

Kunkel, S. R., and J. E. Smith [1986]. "Optimal pipelining in supercomputers," *Proc. 13th Symposium on Computer Architecture* (June), Tokyo, 404–414.

Lam, M. [1988]. "Software pipelining: An effective scheduling technique for VLIW processors," *SIGPLAN Conf. on Programming Language Design and Implementation,* ACM (June), Atlanta, Ga., 318–328.

Lam, M. S., and R. P. Wilson [1992]. "Limits of control flow on parallelism," *Proc. 19th Symposium on Computer Architecture* (May), Gold Coast, Australia, 46–57.

Laudon, J., A. Gupta, and M. Horowitz [1994]. "Interleaving: A multithreading technique targeting multiprocessors and workstations," *Proc. Sixth Int'l Conf. on Architectural Support for Programming Languages and Operating Systems* (October), Boston, 308–318.

Lauterbach, G., and T. Horel [1999]. "UltraSPARC-III: Designing third generation 64-bit performance," *IEEE Micro* 19:3 (May/June).

Lipasti, M. H., and J. P. Shen [1996]. "Exceeding the dataflow limit via value prediction," *Proc. 29th Annual ACM/IEEE Int'l Symposium on Microarchitecture* (December).

Lipasti, M. H., C. B. Wilkerson, and J. P. Shen [1996]. "Value locality and load value prediction," *Proc. Seventh Symposium on Architectural Support for Programming Languages and Operating Systems* (October), 138–147.

Lo, J., L. Barroso, S. Eggers, K. Gharachorloo, H. Levy, and S. Parekh [1998]. "An analysis of database workload performance on simultaneous multithreaded processors," *Proc. 25th Int'l Symposium on Computer Architecture* (June), 39–50.

Lo, J., S. Eggers, J. Emer, H. Levy, R. Stamm, and D. Tullsen [1997]. "Converting thread-level parallelism into instruction-level parallelism via simultaneous multithreading," *ACM Transactions on Computer Systems* 15:2 (August), 322–354.

Mahlke, S. A., W. Y. Chen, W.-M. Hwu, B. R. Rau, and M. S. Schlansker [1992]. "Sentinel scheduling for VLIW and superscalar processors," *Proc. Fifth Conf. on Architectural Support for Programming Languages and Operating Systems* (October), Boston, IEEE/ACM, 238–247.

Mahlke, S. A., R. E. Hank, J. E. McCormick, D. I. August, and W. W. Hwu [1995]. "A comparison of full and partial predicated execution support for ILP processors," *Proc. 22nd Annual Int'l Symposium on Computer Architecture* (June), Santa Margherita Ligure, Italy, 138–149.

Mathis, H. M., A. E. Mercias, J. D. McCalpin, R. J. Eickemeyer, and S. R. Kunkel [2005]. "Characterization of the multithreading (SMT) efficiency in Power5," *IBM J. Res. & Dev.,* 49:4/5 (July/September), 555–564.

McCormick, J,. and A. Knies [2002]. "A brief analysis of the SPEC CPU2000 benchmarks on the Intel Itanium 2 processor," *Proc. Hot Chips Conference*, Stanford University, August.

McFarling, S. [1993]. "Combining branch predictors," WRL Technical Note TN-36 (June), Digital Western Research Laboratory, Palo Alto, Calif.

McFarling, S., and J. Hennessy [1986]. "Reducing the cost of branches," *Proc. 13th Symposium on Computer Architecture* (June), Tokyo, 396–403.

McNairy, C., and D. Soltis [2003]. "Itanium 2 processor microarchitecture," *IEEE Micro* 23:2 (March–April), 44–55.

Moshovos, A., S. Breach, T. N. Vijaykumar, and G. S. Sohi [1997]. "Dynamic speculation and synchronization of data dependences," *Proc. 24th Int'l Symposium on Computer Architecture (ISCA),* June, Boulder, Colo.

Moshovos, A., and G. S. Sohi [1997]. "Streamlining inter-operation memory communication via data dependence prediction," *Proc. 30th Annual Int'l Symposium on Microarchitecture (MICRO-30),* December, 235–245.

Nicolau, A., and J. A. Fisher [1984]. "Measuring the parallelism available for very long instruction word architectures," *IEEE Trans. on Computers* C-33:11 (November), 968–976.

Pan, S.-T., K. So, and J. T. Rameh [1992]. "Improving the accuracy of dynamic branch prediction using branch correlation," *Proc. Fifth Conf. on Architectural Support for Programming Languages and Operating Systems,* IEEE/ACM (October), Boston, 76– 84.

Postiff, M.A., D. A. Greene, G. S. Tyson, and T. N. Mudge [1999]. "The limits of instruction level parallelism in SPEC95 applications," *Computer Architecture News* 27:1 (March), 31–40.

Ramamoorthy, C. V., and H. F. Li [1977]. "Pipeline architecture," *ACM Computing Surveys* 9:1 (March), 61–102.

Rau, B. R. [1994]. "Iterative modulo scheduling: An algorithm for software pipelining loops," *Proc. 27th Annual Int'l Symposium on Microarchitecture* (November), San Jose, Calif., 63–74.

Rau, B. R., C. D. Glaeser, and R. L. Picard [1982]. "Efficient code generation for horizontal architectures: Compiler techniques and architectural support," *Proc. Ninth Symposium on Computer Architecture* (April), 131–139.

Rau, B. R., D. W. L. Yen, W. Yen, and R. A. Towle [1989]. "The Cydra 5 departmental supercomputer: Design philosophies, decisions, and trade-offs," *IEEE Computers* 22:1 (January), 12–34.

Riseman, E. M., and C. C. Foster [1972]. "Percolation of code to enhance paralled dispatching and execution," *IEEE Trans. on Computers* C-21:12 (December), 1411–1415.

Rymarczyk, J. [1982]. "Coding guidelines for pipelined processors," *Proc. Symposium on Architectural Support for Programming Languages and Operating Systems,* IEEE/ACM (March), Palo Alto, Calif., 12–19.

Sharangpani, H., and K. Arora [2000]. "Itanium Processor Microarchitecture," *IEEE Micro*, 20:5 (September–October), 24–43.

Sinharoy, B., R. N. Koala, J. M. Tendler, R. J. Eickemeyer, and J. B. Joyner [2005]. "POWER5 system microarchitecture," *IBM J. Res. & Dev*, 49:4-5, 505–521.

Sites, R. [1979]. *Instruction Ordering for the CRAY-1 Computer,* Tech. Rep. 78-CS-023 (July), Dept. of Computer Science, Univ. of Calif., San Diego.

Skadron, K., P. S. Ahuja, M. Martonosi, and D. W. Clark [1999]. "Branch prediction, instruction-window size, and cache size: Performance tradeoffs and simulation techniques," *IEEE Trans. on Computers*, 48:11. (November).

Smith, A., and J. Lee [1984]. "Branch prediction strategies and branch-target buffer design," *Computer* 17:1 (January), 6–22.

Smith, B. J. [1978]. "A pipelined, shared resource MIMD computer," *Proc. 1978 ICPP* (August), 6–8.

Smith, J. E. [1981]. "A study of branch prediction strategies," *Proc. Eighth Symposium on Computer Architecture* (May), Minneapolis, 135–148.

Smith, J. E. [1984]. "Decoupled access/execute computer architectures," *ACM Trans. on Computer Systems* 2:4 (November), 289–308.

Smith, J. E. [1989]. "Dynamic instruction scheduling and the Astronautics ZS-1," *Computer* 22:7 (July), 21–35.

Smith, J. E., G. E. Dermer, B. D. Vanderwarn, S. D. Klinger, C. M. Rozewski, D. L. Fowler, K. R. Scidmore, and J. P. Laudon [1987]. "The ZS-1 central processor," *Proc. Second Conf. on Architectural Support for Programming Languages and Operating Systems,* IEEE/ACM (March), Palo Alto, Calif., 199–204.

Smith, J. E., and A. R. Pleszkun [1988]. "Implementing precise interrupts in pipelined processors," *IEEE Trans. on Computers* 37:5 (May), 562–573. (This paper is based on an earlier paper that appeared in *Proc. 12th Symposium on Computer Architecture,* June 1988.)

Smith, M. D., M. Horowitz, and M. S. Lam [1992]. "Efficient superscalar performance through boosting," *Proc. Fifth Conf. on Architectural Support for Programming Languages and Operating Systems* (October), Boston, IEEE/ACM, 248–259.

Smith, M. D., M. Johnson, and M. A. Horowitz [1989]. "Limits on multiple instruction issue," *Proc. Third Conf. on Architectural Support for Programming Languages and Operating Systems,* IEEE/ACM (April), Boston, 290–302.

Sodani, A., and G. Sohi [1997]. "Dynamic instruction reuse," *Proc. 24th Int'l Symposium on Computer Architecture* (June).

Sohi, G. S. [1990]. "Instruction issue logic for high-performance, interruptible, multiple functional unit, pipelined computers," *IEEE Trans. on Computers* 39:3 (March), 349– 359.

Sohi, G. S., and S. Vajapeyam [1989]. "Tradeoffs in instruction format design for horizontal architectures," *Proc. Third Conf. on Architectural Support for Programming Languages and Operating Systems,* IEEE/ACM (April), Boston, 15–25.

Sussenguth, E. [1999]. "IBM's ACS-1 Machine," *IEEE Computer* 22:11 (November).

Tendler, J. M., J. S. Dodson, J. S. Fields, Jr., H. Le, and B. Sinharoy [2002]. "Power4 system microarchitecture," *IBM J. Res & Dev*, 46:1, 5–26.

Thorlin, J. F. [1967]. "Code generation for PIE (parallel instruction execution) computers," *Proc. Spring Joint Computer Conf.,* 27.

Thornton, J. E. [1964]. "Parallel operation in the Control Data 6600," *Proc. AFIPS Fall Joint Computer Conf., Part II,* 26, 33–40.

Thornton, J. E. [1970]. *Design of a Computer, the Control Data 6600,* Scott, Foresman, Glenview, Ill.

Tjaden, G. S., and M. J. Flynn [1970]. "Detection and parallel execution of independent instructions," *IEEE Trans. on Computers* C-19:10 (October), 889–895.

Tomasulo, R. M. [1967]. "An efficient algorithm for exploiting multiple arithmetic units," *IBM J. Research and Development* 11:1 (January), 25–33.

Tuck, N., and D. Tullsen [2003]. "Initial observations of the simultaneous multithreading Pentium 4 processor," *Proc. 12th Int. Conf. on Parallel Architectures and Compilation Techniques (PACT'03*), 26–34.

Tullsen, D. M., S. J. Eggers, J. S. Emer, H. M. Levy, J. L. Lo, and R. L. Stamm [1996]. "Exploiting choice: Instruction fetch and issue on an implementable simultaneous multithreading processor," *Proc. 23rd Annual Int'l Symposium on Computer Architecture* (May), 191–202.

Tullsen, D. M., S. J. Eggers, and H. M. Levy [1995]. "Simultaneous multithreading: Maximizing on-chip parallelism," *Proc. 22nd Int'l Symposium on Computer Architecture* (June), 392–403.

Wall, D. W. [1991]. "Limits of instruction-level parallelism," *Proc. Fourth Conf. on Architectural Support for Programming Languages and Operating Systems* (April), Santa Clara, Calif., IEEE/ACM, 248–259.

Wall, D. W. [1993]. *Limits of Instruction-Level Parallelism,* Research Rep. 93/6, Western Research Laboratory, Digital Equipment Corp. (November).

Weiss, S., and J. E. Smith [1984]. "Instruction issue logic for pipelined supercomputers," *Proc. 11th Symposium on Computer Architecture* (June), Ann Arbor, Mich., 110–118.

Weiss, S., and J. E. Smith [1987]. "A study of scalar compilation techniques for pipelined supercomputers," *Proc. Second Conf. on Architectural Support for Programming Languages and Operating Systems* (March), IEEE/ACM, Palo Alto, Calif., 105–109.

Wilson, R. P., and M. S. Lam [1995]. "Efficient context-sensitive pointer analysis for C programs," *Proc. ACM SIGPLAN'95 Conf. on Programming Language Design and Implementation,* La Jolla, Calif., June, 1–12.

Wolfe, A., and J. P. Shen [1991]. "A variable instruction stream extension to the VLIW architecture," *Proc. Fourth Conference on Architectural Support for Programming Languages and Operating Systems* (April), Santa Clara, Calif., 2–14.

Yamamoto, W., M. J. Serrano, A. R. Talcott, R. C. Wood, and M. Nemirosky [1994]. "Performance estimation of multistreamed, superscalar processors," *Proc. 27th Hawaii Int'l Conf. on System Sciences* (January), I:195–204.

Yeager, K. [1996]. "The MIPS R10000 superscalar microprocessor," *IEEE Micro* 16:2 (April), 28–40.

Yeh, T., and Y. N. Patt [1993]. "Alternative implementations of two-level adaptive branch prediction," *Proc. 19th Int'l Symposium on Computer Architecture* (May), Gold Coast, Australia, 124–134.

Yeh, T., and Y. N. Patt [1993]. "A comparison of dynamic branch predictors that use two levels of branch history," *Proc. 20th Symposium on Computer Architecture* (May), San Diego, 257–266.

K.5 The History of Multiprocessors and Parallel Processing (Chapter 4; Appendices E, F, and H)

There is a tremendous amount of history in multiprocessors; in this section we divide our discussion by both time period and architecture. We start with the SIMD approach and the Illiac IV. We then turn to a short discussion of some other early experimental multiprocessors and progress to a discussion of some of the great debates in parallel processing. Next we discuss the historical roots of the present multiprocessors and conclude by discussing recent advances.

SIMD Computers: Attractive Idea, Many Attempts, No Lasting Successes

The cost of a general multiprocessor is, however, very high and further design options were considered which would decrease the cost without seriously degrad-

ing the power or efficiency of the system. The options consist of recentralizing one of the three major components. . . . Centralizing the [control unit] gives rise to the basic organization of [an] . . . array processor such as the Illiac IV.

Bouknight et al. [1972]

The SIMD model was one of the earliest models of parallel computing, dating back to the first large-scale multiprocessor, the Illiac IV. The key idea in that multiprocessor, as in more recent SIMD multiprocessors, is to have a single instruction that operates on many data items at once, using many functional units.

The earliest ideas on SIMD-style computers are from Unger [1958] and Slotnick, Borck, and McReynolds [1962]. Slotnick's Solomon design formed the basis of the Illiac IV, perhaps the most infamous of the supercomputer projects. Although successful in pushing several technologies that proved useful in later projects, it failed as a computer. Costs escalated from the $8 million estimate in 1966 to $31 million by 1972, despite construction of only a quarter of the planned multiprocessor. Actual performance was at best 15 MFLOPS, versus initial predictions of 1000 MFLOPS for the full system [Hord 1982]. Delivered to NASA Ames Research in 1972, the computer took three more years of engineering before it was usable. These events slowed investigation of SIMD, with Danny Hillis [1985] resuscitating this style in the Connection Machine, which had 65,636 1-bit processors.

Real SIMD computers need to have a mixture of SISD and SIMD instructions. There is an SISD host computer to perform operations such as branches and address calculations that do not need parallel operation. The SIMD instructions are broadcast to all the execution units, each of which has its own set of registers. For flexibility, individual execution units can be disabled during an SIMD instruction. In addition, massively parallel SIMD multiprocessors rely on interconnection or communication networks to exchange data between processing elements.

SIMD works best in dealing with arrays in for loops. Hence, to have the opportunity for massive parallelism in SIMD there must be massive amounts of data, or *data parallelism*. SIMD is at its weakest in case statements, where each execution unit must perform a different operation on its data, depending on what data it has. The execution units with the wrong data are disabled so that the proper units can continue. Such situations essentially run at 1/*n*th performance, where *n* is the number of cases.

The basic trade-off in SIMD multiprocessors is performance of a processor versus number of processors. Recent multiprocessors emphasize a large degree of parallelism over performance of the individual processors. The Connection Multiprocessor 2, for example, offered 65,536 single-bit-wide processors, while the Illiac IV had 64 64-bit processors.

After being resurrected in the 1980s, first by Thinking Machines and then by MasPar, the SIMD model has once again been put to bed as a general-purpose multiprocessor architecture, for two main reasons. First, it is too inflexible. A number of important problems cannot use such a style of multiprocessor, and the

architecture does not scale down in a competitive fashion; that is, small-scale SIMD multiprocessors often have worse cost-performance compared with that of the alternatives. Second, SIMD cannot take advantage of the tremendous performance and cost advantages of microprocessor technology. Instead of leveraging this low-cost technology, designers of SIMD multiprocessors must build custom processors for their multiprocessors.

Although SIMD computers have departed from the scene as general-purpose alternatives, this style of architecture will continue to have a role in special-purpose designs. Many special-purpose tasks are highly data parallel and require a limited set of functional units. Thus designers can build in support for certain operations, as well as hardwired interconnection paths among functional units. Such organizations are often called *array processors,* and they are useful for tasks like image and signal processing.

Other Early Experiments

It is difficult to distinguish the first MIMD multiprocessor. Surprisingly, the first computer from the Eckert-Mauchly Corporation, for example, had duplicate units to improve availability. Holland [1959] gave early arguments for multiple processors.

Two of the best-documented multiprocessor projects were undertaken in the 1970s at Carnegie Mellon University. The first of these was C.mmp [Wulf and Bell 1972; Wulf and Harbison 1978], which consisted of 16 PDP-11s connected by a crossbar switch to 16 memory units. It was among the first multiprocessors with more than a few processors, and it had a shared-memory programming model. Much of the focus of the research in the C.mmp project was on software, especially in the OS area. A later multiprocessor, Cm* [Swan et al. 1977], was a cluster-based multiprocessor with a distributed memory and a nonuniform access time. The absence of caches and a long remote access latency made data placement critical. This multiprocessor and a number of application experiments are well described by Gehringer, Siewiorek, and Segall [1987]. Many of the ideas in these multiprocessors would be reused in the 1980s when the microprocessor made it much cheaper to build multiprocessors.

Great Debates in Parallel Processing

The turning away from the conventional organization came in the middle 1960s, when the law of diminishing returns began to take effect in the effort to increase the operational speed of a computer.… Electronic circuits are ultimately limited in their speed of operation by the speed of light . . . and many of the circuits were already operating in the nanosecond range.

W. Jack Bouknight et al.
The Illiac IV System (1972)

. . . sequential computers are approaching a fundamental physical limit on their potential computational power. Such a limit is the speed of light . . .

Angel L. DeCegama
The Technology of Parallel Processing, Volume I (1989)

. . . today's multiprocessors . . . are nearing an impasse as technologies approach the speed of light. Even if the components of a sequential processor could be made to work this fast, the best that could be expected is no more than a few million instructions per second.

David Mitchell
The Transputer: The Time Is Now (1989)

The quotes above give the classic arguments for abandoning the current form of computing, and Amdahl [1967] gave the classic reply in support of continued focus on the IBM 360 architecture. Arguments for the advantages of parallel execution can be traced back to the 19th century [Menabrea 1842]! Yet the effectiveness of the multiprocessor for reducing latency of individual important programs is still being explored. Aside from these debates about the advantages and limitations of parallelism, several hot debates have focused on how to build multiprocessors.

It's hard to predict the future, yet in 1989 Gordon Bell made two predictions for 1995. We included these predictions in the first edition of the book, when the outcome was completely unclear. We discuss them in this section, together with an assessment of the accuracy of the prediction.

The first was that a computer capable of sustaining a teraFLOPS—one million MFLOPS—would be constructed by 1995, either using a multicomputer with 4K to 32K nodes or a Connection Multiprocessor with several million processing elements [Bell 1989]. To put this prediction in perspective, each year the Gordon Bell Prize acknowledges advances in parallelism, including the fastest real program (highest MFLOPS). In 1989 the winner used an eight-processor Cray Y-MP to run at 1680 MFLOPS. On the basis of these numbers, multiprocessors and programs would have to have improved by a factor of 3.6 each year for the fastest program to achieve 1 TFLOPS in 1995. In 1999, the first Gordon Bell prize winner crossed the 1 TFLOPS bar. Using a 5832-processor IBM RS/6000 SST system designed specially for Livermore Laboratories, they achieved 1.18 TFLOPS on a shock wave simulation. This ratio represents a year-to-year improvement of 1.93, which is still quite impressive.

What has become recognized since the 1990s is that although we may have the technology to build a TFLOPS multiprocessor, it is not clear that the machine is cost-effective, except perhaps for a few very specialized and critically important applications related to national security. We estimated in 1990 that to achieve 1 TFLOPS would require a machine with about 5000 processors and would cost about $100 million. The 5832-processor IBM system at Livermore cost $110 million. As might be expected, improvements in the performance of individual

microprocessors both in cost and performance directly affect the cost and performance of large-scale multiprocessors, but a 5000-processor system will cost more than 5000 times the price of a desktop system using the same processor. Since that time, much faster multiprocessors have been built, but the major improvements have increasingly come from the processors in the past five years, rather than fundamental breakthroughs in parallel architecture.

The second Bell prediction concerned the number of data streams in supercomputers shipped in 1995. Danny Hillis believed that although supercomputers with a small number of data streams may be the best sellers, the biggest multiprocessors would be multiprocessors with many data streams, and these would perform the bulk of the computations. Bell bet Hillis that in the last quarter of calendar year 1995 more sustained MFLOPS would be shipped in multiprocessors using few data streams (≤ 100) rather than many data streams (≥ 1000). This bet concerned only supercomputers, defined as multiprocessors costing more than $1 million and used for scientific applications. Sustained MFLOPS was defined for this bet as the number of floating-point operations per *month,* so availability of multiprocessors affects their rating.

In 1989, when this bet was made, it was totally unclear who would win. In 1995, a survey of the current publicly known supercomputers showed only six multiprocessors in existence in the world with more than 1000 data streams, so Bell's prediction was a clear winner. In fact, in 1995, much smaller microprocessor-based multiprocessors (≤ 20 processors) were becoming dominant. In 1995, a survey of the 500 highest-performance multiprocessors in use (based on Linpack ratings), called the Top 500, showed that the largest number of multiprocessors were bus-based shared-memory multiprocessors! By 2005, various clusters or multicomputers played a large role. For example, in the top 25 systems, 11 were custom clusters, such as the IBM Blue Gene system or the Cray XT3, 10 were clusters of shared-memory multiprocessors (both using distributed and centralized memory), and the remaining 4 were clusters built using PCs with an off-the-shelf interconnect.

More Recent Advances and Developments

With the primary exception of the parallel vector multiprocessors (see Appendix F) and more recently of the IBM Blue Gene design, all other recent MIMD computers have been built from off-the-shelf microprocessors using a bus and logically central memory or an interconnection network and a distributed memory. A number of experimental multiprocessors built in the 1980s further refined and enhanced the concepts that form the basis for many of today's multiprocessors.

The Development of Bus-Based Coherent Multiprocessors

Although very large mainframes were built with multiple processors in the 1960s and 1970s, multiprocessors did not become highly successful until the 1980s. Bell [1985] suggests the key was that the smaller size of the microprocessor

allowed the memory bus to replace the interconnection network hardware, and that portable operating systems meant that multiprocessor projects no longer required the invention of a new operating system. In this paper, Bell defines the terms *multiprocessor* and *multicomputer* and sets the stage for two different approaches to building larger-scale multiprocessors.

The first bus-based multiprocessor with snooping caches was the Synapse N + 1 described by Frank [1984]. Goodman [1983] wrote one of the first papers to describe snooping caches. The late 1980s saw the introduction of many commercial bus-based, snooping cache architectures, including the Silicon Graphics 4D/240 [Baskett, Jermoluk, and Solomon 1988], the Encore Multimax [Wilson 1987], and the Sequent Symmetry [Lovett and Thakkar 1988]. The mid-1980s saw an explosion in the development of alternative coherence protocols, and Archibald and Baer [1986] provide a good survey and analysis, as well as references to the original papers. Figure K.2 summarizes several snooping cache coherence protocols and shows some multiprocessors that have used or are using that protocol.

The early 1990s saw the beginning of an expansion of such systems with the use of very wide, high-speed buses (the SGI Challenge system used a 256-bit, packet-oriented bus supporting up to 8 processor boards and 32 processors) and later the use of multiple buses and crossbar interconnects, for example, in the Sun SPARCCenter and Enterprise systems (Charlesworth [1998] discusses the interconnect architecture of these multiprocessors). In 2001, the Sun Enterprise servers represent the primary example of large-scale (> 16 processors), symmetric

Name	Protocol type	Memory write policy	Unique feature	Multiprocessors using
Write Once	Write invalidate	Write back after first write	First snooping protocol described in literature	
Synapse N + 1	Write invalidate	Write back	Explicit state where memory is the owner	Synapse multiprocessors; first cache-coherent multiprocessors available
Berkeley (MOESI)	Write invalidate	Write back	Owned shared state	Berkeley SPUR multiprocessor; Sun Enterprise servers
Illinois (MESI)	Write invalidate	Write back	Clean private state; can supply data from any cache with a clean copy	SGI Power and Challenge series
"Firefly"	Write broadcast	Write back when private, write through when shared	Memory updated on broadcast	No current multiprocessors; SPARCCenter 2000 closest

Figure K.2 Five snooping protocols summarized. Archibald and Baer [1986] use these names to describe the five protocols, and Eggers [1989] summarizes the similarities and differences as shown in this figure. The Firefly protocol was named for the experimental DEC Firefly multiprocessor, in which it appeared. The alternative names for protocols are based on the states they support: M = Modified, E = Exclusive (private clean), S = Shared, I = Invalid, O = Owner (shared dirty).

multiprocessors in active use. Today, most bus-based machines offer only four or so processors and switches, or alternative designs are used for eight or more.

Toward Large-Scale Multiprocessors

In the effort to build large-scale multiprocessors, two different directions were explored: message-passing multicomputers and scalable shared-memory multiprocessors. Although there had been many attempts to build mesh and hypercube-connected multiprocessors, one of the first multiprocessors to successfully bring together all the pieces was the Cosmic Cube built at Caltech [Seitz 1985]. It introduced important advances in routing and interconnect technology and substantially reduced the cost of the interconnect, which helped make the multicomputer viable. The Intel iPSC 860, a hypercube-connected collection of i860s, was based on these ideas. More recent multiprocessors, such as the Intel Paragon, have used networks with lower dimensionality and higher individual links. The Paragon also employed a separate i860 as a communications controller in each node, although a number of users have found it better to use both i860 processors for computation as well as communication. The Thinking Multiprocessors CM-5 made use of off-the-shelf microprocessors and a fat tree interconnect (see Appendix E). It provided user-level access to the communication channel, thus significantly improving communication latency. In 1995, these two multiprocessors represented the state of the art in message-passing multicomputers.

Early attempts at building a scalable shared-memory multiprocessor include the IBM RP3 [Pfister et al. 1985], the NYU Ultracomputer [Schwartz 1980; Elder et al. 1985], the University of Illinois Cedar project [Gajksi et al. 1983], and the BBN Butterfly and Monarch [BBN Laboratories 1986; Rettberg et al. 1990]. These multiprocessors all provided variations on a nonuniform distributed-memory model (and hence are distributed shared-memory, or DSM, multiprocessors), but did not support cache coherence, which substantially complicated programming. The RP3 and Ultracomputer projects both explored new ideas in synchronization (fetch-and-operate) as well as the idea of combining references in the network. In all four multiprocessors, the interconnect networks turned out to be more costly than the processing nodes, raising problems for smaller versions of the multiprocessor. The Cray T3D/E (see Arpaci et al. [1995] for an evaluation of the T3D and Scott [1996] for a description of the T3E enhancements) builds on these ideas, using a noncoherent shared address space but building on the advances in interconnect technology developed in the multicomputer domain (see Scott and Thorson [1996]).

Extending the shared-memory model with scalable cache coherence was done by combining a number of ideas. Directory-based techniques for cache coherence were actually known before snooping cache techniques. In fact, the first cache coherence protocols actually used directories, as described by Tang [1976] and implemented in the IBM 3081. Censier and Feautrier [1978] described a directory coherence scheme with tags in memory. The idea of distrib-

uting directories with the memories to obtain a scalable implementation of cache coherence was first described by Agarwal et al. [1988] and served as the basis for the Stanford DASH multiprocessor (see Lenoski et al. [1990, 1992]), which was the first operational cache-coherent DSM multiprocessor. DASH was a "plump" node cc-NUMA machine that used four-processor SMPs as its nodes, interconnecting them in a style similar to that of Wildfire but using a more scalable two-dimensional grid rather than a crossbar for the interconnect.

The Kendall Square Research KSR-1 [Burkhardt et al. 1992] was the first commercial implementation of scalable coherent shared memory. It extended the basic DSM approach to implement a concept called *cache-only memory architecture* (COMA), which makes the main memory a cache. In the KSR-1 memory blocks could be replicated in the main memories of each node with hardware support to handle the additional coherence requirements for these replicated blocks. (The KSR-1 was not strictly a pure COMA because it did not migrate the home location of a data item, but always kept a copy at home. Essentially, it implemented only replication.) Many other research proposals [Hagersten, Landin, and Haridi 1992; Stenström, Joe, and Gupta 1992; Saulsbury et al. 1995; Falsafi and Wood 1997] for COMA-style architectures and similar approaches that reduce the burden of nonuniform memory access through migration [Chandra et al. 1994; Soundararajan et al. 1998] were developed, but there have been no further commercial implementations.

The Convex Exemplar implemented scalable coherent shared memory using a two-level architecture: At the lowest level eight-processor modules are built using a crossbar. A ring can then connect up to 32 of these modules, for a total of 256 processors (see Thekkath et al. [1997] for an evaluation). Laudon and Lenoski [1997] describe the SGI Origin, which was first delivered in 1996 and is closely based on the original Stanford DASH machine, although including a number of innovations for scalability and ease of programming. Origin uses a bit vector for the directory structure, which is either 16 or 32 bits long. Each bit represents a node, which consists of two processors; a coarse bit vector representation allows each bit to represent up to 8 nodes for a total of 1024 processors. As Galles [1996] describes, a high-performance fat hypercube is used for the global interconnect. Hristea, Lenoski, and Keen [1997] is a thorough evaluation of the performance of the Origin memory system.

Several research prototypes were undertaken to explore scalable coherence with and without multithreading. These include the MIT Alewife machine [Agarwal et al. 1995] and the Stanford FLASH multiprocessor [Kuskin et al. 1994; Gibson et al. 2000].

Clusters

Clusters were probably "invented" in the 1960s by customers who could not fit all their work on one computer, or who needed a backup machine in case of failure of the primary machine [Pfister 1998]. Tandem introduced a 16-node cluster in 1975. Digital followed with VAX clusters, introduced in 1984. They were orig-

inally independent computers that shared I/O devices, requiring a distributed operating system to coordinate activity. Soon they had communication links between computers, in part so that the computers could be geographically distributed to increase availability in case of a disaster at a single site. Users log onto the cluster and are unaware of which machine they are running on. DEC (now HP) sold more than 25,000 clusters by 1993. Other early companies were Tandem (now HP) and IBM (still IBM). Today, virtually every company has cluster products. Most of these products are aimed at availability, with performance scaling as a secondary benefit.

Scientific computing on clusters emerged as a competitor to MPPs. In 1993, the Beowulf project started with the goal of fulfilling NASA's desire for a 1 GFLOPS computer for under $50,000. In 1994, a 16-node cluster built from off-the-shelf PCs using 80486s achieved that goal [Bell and Gray 2001]. This emphasis led to a variety of software interfaces to make it easier to submit, coordinate, and debug large programs or a large number of independent programs.

Efforts were made to reduce latency of communication in clusters as well as to increase bandwidth, and several research projects worked on that problem. (One commercial result of the low-latency research was the VI interface standard, which has been embraced by Infiniband, discussed below.) Low latency then proved useful in other applications. For example, in 1997 a cluster of 100 UltraSPARC desktop computers at UC Berkeley, connected by 160 MB/sec per link Myrinet switches, was used to set world records in database sort—sorting 8.6 GB of data originally on disk in 1 minute—and in cracking an encrypted message—taking just 3.5 hours to decipher a 40-bit DES key.

This research project, called Network of Workstations [Anderson, Culler, and Patterson 1995], also developed the Inktomi search engine, which led to a start-up company with the same name. Google followed the example of Inktomi to build search engines from clusters of desktop computers rather large-scale SMPs, which was the strategy of the leading search engine Alta Vista that Google overtook [Brin and Page 1998]. In 2006, nearly all Internet services rely on clusters to serve their millions of customers.

Clusters are also very popular with scientists. One reason is their low cost, so individual scientists or small groups can own a cluster dedicated to their programs. Such clusters can get results faster than waiting in the long job queues of the shared MPPs at supercomputer centers, which can stretch to weeks.

For those interested in learning more, Pfister [1998] has written an entertaining book on clusters.

Recent Trends in Large-Scale Multiprocessors

In the mid-to-late 1990s, it became clear that the hoped for growth in the market for ultralarge-scale parallel computing was unlikely to occur. Without this market growth, it became increasingly clear that the high-end parallel computing market could not support the costs of highly customized hardware and software designed for a small market. Perhaps the most important trend to come out of this observa-

tion was that clustering would be used to reach the highest levels of performance. There are now four general classes of large-scale multiprocessors:

1. Clusters that integrate standard desktop motherboards using interconnection technology such as Myrinet or Infiniband.
2. Multicomputers built from standard microprocessors configured into processing elements and connected with a custom interconnect. These include the Cray XT3 (which used an earlier version of Cray interconnect with a simple cluster architecture) and IBM Blue Gene (more on this unique machine momentarily).
3. Clusters of small-scale shared-memory computers, possibly with vector support, which includes the Earth Simulator (which has its own journal available online).
4. Large-scale shared-memory multiprocessors, such as the Cray X1 [Dunigan et al. 2005] and SGI Origin and Altix systems. The SGI systems have also been configured into clusters to provide more than 512 processors, although only message passing is supported across the clusters.

The IBM Blue Gene is the most interesting of these designs since its rationale parallels the underlying causes of the recent trend towards multicore in uniprocessor architectures. Blue Gene started as a research project within IBM aimed at the protein sequencing and folding problem. The Blue Gene designers observed that power was becoming an increasing concern in large-scale multiprocessors and that the performance/watt of processors from the embedded space was much better that those in the high-end uniprocessor space. If parallelism was the route to high performance, why not start with the most efficient building block and simply have more of them?

Thus, Blue Gene is constructed using a custom chip that includes an embedded PowerPC microprocessor offering half the performance of a high-end PowerPC, but at a much smaller fraction of the area of power. This allows more system functions, including the global interconnect, to be integrated onto the same die. The result is a highly replicable and efficient building block, allowing Blue Gene to reach much larger processor counts more efficiently. Instead of using stand-alone microprocessors or standard desktop boards as building blocks, Blue Gene uses processor cores. There is no doubt that such an approach provides much greater efficiency. Whether the market can support the cost of a customized design and special software remains an open question.

In 2006, a Blue Gene processor at Lawrence Livermore with 32K processors (and scheduled to go to 65K in late 2005) holds a factor of 2.6 lead in Linpack performance over the third-place system consisting of 20 SGI Altix 512-processor systems interconnected with Infiniband as a cluster.

Blue Gene's predecessor was an experimental machine, QCDOD, which pioneered the concept of a machine using a lower-power embedded microprocessor and tightly integrated interconnect to drive down the cost and power consumption of a node.

Developments in Synchronization and Consistency Models

A wide variety of synchronization primitives have been proposed for shared-memory multiprocessors. Mellor-Crummey and Scott [1991] provide an overview of the issues as well as efficient implementations of important primitives, such as locks and barriers. An extensive bibliography supplies references to other important contributions, including developments in spin locks, queuing locks, and barriers.

Lamport [1979] introduced the concept of sequential consistency and what correct execution of parallel programs means. Dubois, Scheurich, and Briggs [1988] introduced the idea of weak ordering (originally in 1986). In 1990, Adve and Hill provided a better definition of weak ordering and also defined the concept of data-race-free; at the same conference, Gharachorloo and his colleagues [1990] introduced release consistency and provided the first data on the performance of relaxed consistency models. More relaxed consistency models have been widely adopted in microprocessor architectures, including the Sun SPARC, Alpha, and IA-64. Adve and Gharachorloo [1996] is an excellent tutorial on memory consistency and the differences among these models.

Other References

The concept of using virtual memory to implement a shared address space among distinct machines was pioneered in Kai Li's Ivy system in 1988. There have been subsequent papers exploring hardware support issues, software mechanisms, and programming issues. Amza et al. [1996] describe a system built on workstations using a new consistency model, Kontothanassis et al. [1997] describe a software shared-memory scheme using remote writes, and Erlichson et al. [1996] describe the use of shared virtual memory to build large-scale multiprocessors using SMPs as nodes.

There is an almost unbounded amount of information on multiprocessors and multicomputers: Conferences, journal papers, and even books seem to appear faster than any single person can absorb the ideas. No doubt many of these papers will go unnoticed—not unlike the past. Most of the major architecture conferences contain papers on multiprocessors. An annual conference, Supercomputing *XY* (where *X* and *Y* are the last two digits of the year), brings together users, architects, software developers, and vendors and publishes the proceedings in book, CD-ROM, and online (see *www.scXY.org*) form. Two major journals, *Journal of Parallel and Distributed Computing* and the *IEEE Transactions on Parallel and Distributed Systems,* contain papers on all aspects of parallel processing. Several books focusing on parallel processing are included in the following references, with Culler, Singh, and Gupta [1999] being the most recent, large-scale effort. For years, Eugene Miya of NASA Ames has collected an online bibliography of parallel-processing papers. The bibliography, which now contains more than 35,000 entries, is available online at *liinwww.ira.uka.de/bibliography/Parallel/Eugene/index.html.*

In addition to documenting the discovery of concepts now used in practice, these references also provide descriptions of many ideas that have been explored and found wanting, as well as ideas whose time has just not yet come. Given the move toward multicore and multiprocessors as the future of high-performance computer architecture, we expect many new approaches will be explored in the years ahead. A few of them will manage to solve the hardware and software problems that have been the key to using multiprocessing for the past 40 years!

References

Adve, S. V., and K. Gharachorloo [1996]. "Shared memory consistency models: A tutorial," *IEEE Computer* 29:12 (December), 66–76.

Adve, S. V., and M. D. Hill [1990]. "Weak ordering—a new definition," *Proc. 17th Int'l Symposium on Computer Architecture* (June), Seattle, Wash., 2–14.

Agarwal, A., R. Bianchini, D. Chaiken, K. Johnson, and D. Kranz [1995]. "The MIT Alewife machine: Architecture and performance," *Int'l Symposium on Computer Architecture* (Denver, Colo.), June, 2–13.

Agarwal, A., J. L. Hennessy, R. Simoni, and M. A. Horowitz [1988]. "An evaluation of directory schemes for cache coherence," *Proc. 15th Int'l Symposium on Computer Architecture* (June), 280–289.

Agarwal, A., J. Kubiatowicz, D. Kranz, B.-H. Lim, D. Yeung, G. D'Souza, and M. Parkin [1993]. "Sparcle: An evolutionary processor design for large-scale multiprocessors," *IEEE Micro* 13 (June), 48–61.

Alles, A. [1995]. "ATM internetworking" (May), *www.cisco.com/warp/public/614/12.html.*

Almasi, G. S., and A. Gottlieb [1989]. *Highly Parallel Computing,* Benjamin/Cummings, Redwood City, Calif.

Alverson, G., R. Alverson, D. Callahan, B. Koblenz, A. Porterfield, and B. Smith [1992]. "Exploiting heterogeneous parallelism on a multithreaded multiprocessor," *Proc. 1992 Int'l Conf. on Supercomputing* (November), 188–197.

Amdahl, G. M. [1967]. "Validity of the single processor approach to achieving large scale computing capabilities," *Proc. AFIPS Spring Joint Computer Conf.* 30*,* Atlantic City, N.J. (April), 483–485.

Amza C., A. L. Cox, S. Dwarkadas, P. Keleher, H. Lu, R. Rajamony, W. Yu, and W. Zwaenepoel [1996]. "Treadmarks: Shared memory computing on networks of workstations," *IEEE Computer* 29(2) (February), 18–28.

Anderson, T. E., D. E. Culler, and D. Patterson [1995]. "A case for NOW (networks of workstations)," IEEE Micro 15:1 (February), 54–64.

Ang, B., D. Chiou, D. Rosenband, M. Ehrlich, L. Rudolph, and Arvind [1998]. "StarT-Voyager: A flexible platform for exploring scalable SMP issues," *Proc. of SC'98,* Orlando, Fla., November.

Archibald, J., and J.-L. Baer [1986]. "Cache coherence protocols: Evaluation using a multiprocessor simulation model," *ACM Trans. on Computer Systems* 4:4 (November), 273–298.

Arpaci, R. H., D. E. Culler, A. Krishnamurthy, S. G. Steinberg, and K. Yelick [1995]. "Empirical evaluation of the CRAY-T3D: A compiler perspective," *Proc. 23rd Int'l Symposium on Computer Architecture* (June), Italy.

Baer J-L., and W-H. Wang [1988]. "On the inclusion properties for multi-level cache hierarchies," *Proc. 15th Annual Int'l Symposium on Computer Architecture* (May–June), Honolulu, 73–80.

Balakrishnan, H. V., N. Padmanabhan, S. Seshan, and R. H. Katz [1997]. "A comparison of mechanisms for improving TCP performance over wireless links," *IEEE/ACM Trans. on Networking* 5:6 (December), 756–769.

Barroso, L. A., K. Gharachorloo, and E. Bugnion [1998]. "Memory system characterization of commercial workloads," *Proc. 25th Int'l Symposium on Computer Architecture,* Barcelona (July), 3–14.

Baskett, F., T. Jermoluk, and D. Solomon [1988]. "The 4D-MP graphics superworkstation: Computing + graphics = 40 MIPS + 40 MFLOPS and 10,000 lighted polygons per second," *Proc. COMPCON Spring,* San Francisco, 468–471.

BBN Laboratories [1986]. "Butterfly parallel processor overview," Tech. Rep. 6148, BBN Laboratories, Cambridge, Mass.

Bell, C. G. [1985]. "Multis: A new class of multiprocessor computers," *Science* 228 (April 26), 462–467.

Bell, C. G. [1989]. "The future of high performance computers in science and engineering," *Comm.* ACM 32:9 (September), 1091–1101.

Bell, C. G., and J. Gray [2002]. "What's Next in High Performance Computing," *CACM,* 45:2 (February), 91–95.

Bell, G., and J. Gray [2001]. "Crays, clusters and centers," Microsoft Research Technical Report, MSR-TR-2001-76.

Blue Gene [2005]. *IBM J. Res. & Dev.,* 49:2/3.

Bouknight, W. J., S. A. Deneberg, D. E. McIntyre, J. M. Randall, A. H. Sameh, and D. L. Slotnick [1972]. "The Illiac IV system," *Proc. IEEE* 60:4, 369–379. Also appears in D. P. Siewiorek, C. G. Bell, and A. Newell, *Computer Structures: Principles and Examples,* McGraw-Hill, New York (1982), 306–316.

Brain, M. [2000]. "Inside a digital cell phone," *www.howstuffworks.com/inside-cell-phone.htm.*

Brewer, E. A., and B. C. Kuszmaul [1994]. "How to get good performance from the CM-5 data network," *Proc. Eighth Int'l Parallel Processing Symposium* (April), Cancun, Mexico.

Brin, S., and L. Page [1998]. "The anatomy of a large-scale hypertextual Web search engine," *Proc. 7th Int'l World Wide Web Conf.,* Brisbane, Qld., Australia (April 14–18), 107–117.

Burkhardt, H., III, S. Frank, B. Knobe, and J. Rothnie [1992]. "Overview of the KSR1 computer system," Tech. Rep. KSR-TR-9202001, Kendall Square Research, Boston (February).

Censier, L., and P. Feautrier [1978]. "A new solution to coherence problems in multicache systems," *IEEE Trans. on Computers* C-27:12 (December), 1112–1118.

Chandra, R., S. Devine, B. Verghese, A. Gupta, and M. Rosenblum [1994]. "Scheduling and page migration for multiprocessor compute servers," *Proc. Sixth Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS-VI), ACM,* Santa Clara, Calif., October, 12–24.

Charlesworth, A. [1998]. "Starfire: Extending the SMP envelope," *IEEE Micro* 18:1 (January/February), 39–49.

Clark, W. A. [1957]. "The Lincoln TX-2 computer development," *Proc. Western Joint Computer Conference* (February), Institute of Radio Engineers, Los Angeles, 143–145.

Comer, D. [1993]. *Internetworking with TCP/IP,* second edition, Prentice Hall, Englewood Cliffs, N.J.

Culler, D. E., J. P. Singh, and A. Gupta [1999]. *Parallel Computer Architecture: A Hardware/Software Approach,* Morgan Kaufmann, San Francisco.

Dally, W. J., and C. I. Seitz [1986]. "The torus routing chip," *Distributed Computing* 1:4, 187–196.

Davie, B. S., L. L. Peterson, and D. Clark [1999]. *Computer Networks: A Systems Approach,* second edition, Morgan Kaufmann, San Francisco.

Desurvire, E. [1992]. "Lightwave communications: The fifth generation," *Scientific American* (International Edition) 266:1 (January), 96–103.

Dongarra, J., T. Sterling, H. Simon, and E. Strohmaier [2005]. "High-performance computing: Clusters, constellations, MPPs, and future directions," *Computing in Science and Engineering,* IEEE, March/April, 51–59.

Dubois, M., C. Scheurich, and F. Briggs [1988]. "Synchronization, coherence, and event ordering," *IEEE Computer* 9-21 (February).

Dunigan, W., K. Vetter, K. White, and P. Worley [2005]. "Performance evaluation of the Cray X1 distributed shared memory architecture," *IEEE Micro,* Jan/Feb.

Eggers, S. [1989]. *Simulation Analysis of Data Sharing in Shared Memory Multiprocessors,* Ph.D. thesis, Univ. of California, Berkeley. Computer Science Division Tech. Rep. UCB/CSD 89/501 (April).

Elder, J., A. Gottlieb, C. K. Kruskal, K. P. McAuliffe, L. Randolph, M. Snir, P. Teller, and J. Wilson [1985]. "Issues related to MIMD shared-memory computers: The NYU Ultracomputer approach," *Proc. 12th Int'l Symposium on Computer Architecture* (June), Boston, 126–135.

Erlichson, A., N. Nuckolls, G. Chesson, and J. L. Hennessy [1996]. "SoftFLASH: Analyzing the performance of clustered distributed virtual shared memory," *Proc. 7th Symposium on Architectural Support for Programming Languages and Operating Systems (ASPLOS-VII),* October, 210–220.

Falsafi, B., and D. A. Wood [1997]. "Reactive NUMA: A design for unifying S-COMA and CC-NUMA," *Proc. 24th Int'l Symposium on Computer Architecture,* June, Denver, Colo., 229–240.

Flynn, M. J. [1966]. "Very high-speed computing systems," *Proc. IEEE* 54:12 (December), 1901–1909.

Forgie, J. W. [1957]. "The Lincoln TX-2 input-output system," *Proc. Western Joint Computer Conference* (February), Institute of Radio Engineers, Los Angeles, 156–160.

Frank, S. J. [1984]. "Tightly coupled multiprocessor systems speed memory access time," *Electronics* 57:1 (January), 164–169.

Gajski, D., D. Kuck, D. Lawrie, and A. Sameh [1983]. "CEDAR—a large scale multiprocessor," *Proc. Int'l Conf. on Parallel Processing* (August), 524–529.

Galles, M. [1996]. "Scalable pipelined interconnect for distributed endpoint routing: The SGI SPIDER chip," *Proc. Hot Interconnects '96,* Stanford University, August.

Gehringer, E. F., D. P. Siewiorek, and Z. Segall [1987]. *Parallel Processing: The Cm* Experience*, Digital Press, Bedford, Mass.

Gharachorloo, K., A. Gupta, and J. L. Hennessy [1992]. "Hiding memory latency using dynamic scheduling in shared-memory multiprocessors," *Proc. 19th Annual Int'l Symposium on Computer Architecture,* Gold Coast, Australia, June.

Gharachorloo, K., D. Lenoski, J. Laudon, P. Gibbons, A. Gupta, and J. L. Hennessy [1990]. "Memory consistency and event ordering in scalable shared-memory multiprocessors," *Proc. 17th Int'l Symposium on Computer Architecture* (June), Seattle, Wash., 15–26.

Gibson, J., R. Kunz, D. Ofelt, M. Horowitz, J. Hennessy, and M. Heinrich [2000]. "FLASH vs. (simulated) FLASH: Closing the simulation loop," *Proc. 9th Conf. on Architectural Support for Programming Languages and Operating Systems* (November), San Jose, Calif., 49–58.

Goodman, J. R. [1983]. "Using cache memory to reduce processor memory traffic," *Proc. 10th Int'l Symposium on Computer Architecture* (June), Stockholm, Sweden, 124– 131.

Goralski, W. [1997]. *SONET: A Guide to Synchronous Optical Network,* McGraw-Hill, New York.

Grice, C., and M. Kanellos [2000]. "Cell phone industry at crossroads: Go high or low?," *CNET News (*August 31), *technews.netscape.com/news/0-1004-201-2518386-0.html?tag=st.ne.1002.tgif.sf.*

Groe, J. B., and L. E. Larson [2000]. *CDMA Mobile Radio Design,* Artech House, Boston.

Hagersten E., and M. Koster [1998]. "WildFire: A scalable path for SMPs," *Proc. Fifth Int'l Symposium on High Performance Computer Architecture.*

Hagersten, E., A. Landin, and S. Haridi [1992]. "DDM—a cache-only memory architecture," *IEEE Computer* 25:9 (September), 44–54.

Hill, M. D. [1998]. "Multiprocessors should support simple memory consistency models," *IEEE Computer* 31:8 (August), 28–34.

Hillis, W. D. [1985]. *The Connection Multiprocessor,* MIT Press, Cambridge, Mass.

Hirata, H., K. Kimura, S. Nagamine, Y. Mochizuki, A. Nishimura, Y. Nakase, and T. Nishizawa [1992]. "An elementary processor architecture with simultaneous instruction issuing from multiple threads," *Proc. 19th Annual Int'l Symposium on Computer Architecture* (May), 136–145.

Hockney, R. W., and C. R. Jesshope [1988]. *Parallel Computers-2, Architectures, Programming and Algorithms,* Adam Hilger Ltd., Bristol, England.

Holland, J. H. [1959]. "A universal computer capable of executing an arbitrary number of subprograms simultaneously," *Proc. East Joint Computer Conf.* 16, 108–113.

Hord, R. M. [1982]. *The Illiac-IV, The First Supercomputer,* Computer Science Press, Rockville, Md.

Hristea, C., D. Lenoski, and J. Keen [1997]. "Measuring memory hierarchy performance of cache-coherent multiprocessors using micro benchmarks," *Proc. Supercomputing 97,* San Jose, Calif., November.

Hwang, K. [1993]. *Advanced Computer Architecture and Parallel Programming*, McGraw-Hill, New York.

Infiniband Trade Association [2001]. *InfiniBand Architecture Specifications Release 1.0.a, www.infinibandta.org.*

Jordan, H. F. [1983]. "Performance measurements on HEP—a pipelined MIMD computer," *Proc. 10th Int'l Symposium on Computer Architecture* (June), Stockholm, 207–212.

Kahn, R. E. [1972]. "Resource-sharing computer communication networks," *Proc. IEEE* 60:11 (November), 1397–1407.

Keckler, S. W., and W. J. Dally [1992]. "Processor coupling: Integrating compile time and runtime scheduling for parallelism," *Proc. 19th Annual Int'l Symposium on Computer Architecture* (May), 202–213.

Kontothanassis, L., G. Hunt, R. Stets, N. Hardavellas, M. Cierniak, S. Parthasarathy, W. Meira, S. Dwarkadas, and M. Scott [1997]. "VM-based shared memory on low-latency, remote-memory-access networks," *Proc. 24th Annual Int'l. Symposium on Computer Architecture (*June), Denver, Colo.

Kurose, J. F., and K. W. Ross [2001]. *Computer Networking: A Top-Down Approach Featuring the Internet,* Addison-Wesley, Boston.

Kuskin, J., D. Ofelt, M. Heinrich, J. Heinlein, R. Simoni, K. Gharachorloo, J. Chapin, D. Nakahira, J. Baxter, M. Horowitz, A. Gupta, M. Rosenblum, and J. L. Hennessy [1994]. "The Stanford FLASH multiprocessor," *Proc. 21st Int'l Symposium on Computer Architecture,* Chicago, April.

Lamport, L. [1979]. "How to make a multiprocessor computer that correctly executes multiprocess programs," *IEEE Trans. on Computers* C-28:9 (September), 241–248.

Laudon, J., A. Gupta, and M. Horowitz [1994]. "Interleaving: A multithreading technique targeting multiprocessors and workstations," *Proc. Sixth Int'l Conf. on Architectural Support for Programming Languages and Operating Systems* (October), Boston, 308–318.

Laudon, J., and D. Lenoski [1997]. "The SGI Origin: A ccNUMA highly scalable server," *Proc. 24th Int'l Symposium on Computer Architecture* (June), Denver, Colo., 241–251.

Lenoski, D., J. Laudon, K. Gharachorloo, A. Gupta, and J. L. Hennessy [1990]. "The Stanford DASH multiprocessor," *Proc. 17th Int'l Symposium on Computer Architecture* (June), Seattle, Wash., 148–159.

Lenoski, D., J. Laudon, K. Gharachorloo, W.-D. Weber, A. Gupta, J. L. Hennessy, M. A. Horowitz, and M. Lam [1992]. "The Stanford DASH multiprocessor," *IEEE Computer* 25:3 (March).

Li, K. [1988]. "IVY: A shared virtual memory system for parallel computing," *Proc. 1988 Int'l Conf. on Parallel Processing,* Pennsylvania State University Press.

Lo, J., L. Barroso, S. Eggers, K. Gharachorloo, H. Levy, and S. Parekh [1998]. "An analysis of database workload performance on simultaneous multithreaded processors," *Proc. 25th Int'l Symposium on Computer Architecture* (June), 39–50.

Lo, J., S. Eggers, J. Emer, H. Levy, R. Stamm, and D. Tullsen [1997]. "Converting thread-level parallelism into instruction-level parallelism via simultaneous multithreading," *ACM Transactions on Computer Systems* 15:2 (August), 322–354.

Lovett, T., and S. Thakkar [1988]. "The Symmetry multiprocessor system," *Proc. 1988 Int'l Conf. of Parallel Processing*, University Park, Penn., 303–310.

Mellor-Crummey, J. M., and M. L. Scott [1991]. "Algorithms for scalable synchronization on shared-memory multiprocessors," *ACM Trans. on Computer Systems* 9:1 (February), 21–65.

Menabrea, L. F. [1842]. "Sketch of the analytical engine invented by Charles Babbage," *Bibiothèque Universelle de Genève* (October).

Metcalfe, R. M. [1993]. "Computer/network interface design: Lessons from Arpanet and Ethernet." *IEEE J. on Selected Areas in Communications* 11:2 (February), 173–180.

Metcalfe, R. M., and D. R. Boggs [1976]. "Ethernet: Distributed packet switching for local computer networks," *Comm. ACM* 19:7 (July), 395–404.

Mitchell, D. [1989]. "The Transputer: The time is now," *Computer Design* (RISC supplement), 40–41.

Miya, E. N. [1985]. "Multiprocessor/distributed processing bibliography," *Computer Architecture News* (ACM SIGARCH) 13:1, 27–29.

National Research Council [1997]. *The Evolution of Untethered Communications,* Computer Science and Telecommunications Board, National Academy Press, Washington, D.C.

Nikhil, R. S., G. M. Papadopoulos, and Arvind [1992]. "*T: A multithreaded massively parallel architecture," *Proc. 19th Int'l Symposium on Computer Architecture,* Gold Coast, Australia, May, 156–167.

Noordergraaf, L., and R. van der Pas [1999]. "Performance experiences on Sun's WildFire prototype," *Proc. Supercomputing 99,* Portland, Ore., November.

Partridge, C. [1994]. *Gigabit Networking*, Addison-Wesley, Reading, Mass.

Pfister, G. F. [1998]. *In Search of Clusters,* second edition, Prentice Hall, Upper Saddle River, N.J.

Pfister, G. F., W. C. Brantley, D. A. George, S. L. Harvey, W. J. Kleinfekder, K. P. McAuliffe, E. A. Melton, V. A. Norton, and J. Weiss [1985]. "The IBM research parallel processor prototype (RP3): Introduction and architecture," *Proc. 12th Int'l Symposium on Computer Architecture* (June), Boston, 764–771.

Reinhardt, S. K., J. R. Larus, and D. A. Wood [1994]. "Tempest and Typhoon: User-level shared memory," *Proc. 21st Annual Int'l Symposium on Computer Architecture,* Chicago, April, 325–336.

Rettberg, R. D., W. R. Crowther, P. P. Carvey, and R. S. Towlinson [1990]. "The Monarch parallel processor hardware design," *IEEE Computer* 23:4 (April).

Rosenblum, M., S. A. Herrod, E. Witchel, and A. Gupta [1995]. "Complete computer simulation: The SimOS approach," in *IEEE Parallel and Distributed Technology* (now called *Concurrency*) 4:3, 34–43.

Saltzer, J. H., D. P. Reed, and D. D. Clark [1984]. "End-to-end arguments in system design," *ACM Trans. on Computer Systems* 2:4 (November), 277–288.

Satran, J., D. Smith, K. Meth, C. Sapuntzakis, M. Wakeley, P. Von Stamwitz, R. Haagens, E. Zeidner, L. Dalle Ore, and Y. Klein [2001]. "iSCSI," IPS working group of IETF, Internet draft *www.ietf.org/internet-drafts/draft-ietf-ips-iscsi-07.txt.*

Saulsbury, A., T. Wilkinson, J. Carter, and A. Landin [1995]. "An argument for Simple COMA," *Proc. First Conf. on High Performance Computer Architectures* (January), Raleigh, N. C., 276–285.

Schwartz, J. T. [1980]. "Ultracomputers," *ACM Trans. on Programming Languages and Systems* 4:2, 484–521.

Scott, S. L. [1996]. "Synchronization and communication in the T3E multiprocessor," *Proc. Architectural Support for Programming Languages and Operating Systems (ASPLOS-VII),* Cambridge, Mass., October, 26–36.

Scott, S. L., and G. M. Thorson [1996]. "The Cray T3E network: Adaptive routing in a high performance 3D torus," *Proc. Symposium on High Performance Interconnects (Hot Interconnects 4),* Stanford University, August, 14–156.

Seitz, C. L. [1985]. "The Cosmic Cube (concurrent computing)," *Communications of the ACM* 28:1 (January), 22–33.

Singh, J. P., J. L. Hennessy, and A. Gupta [1993]. "Scaling parallel programs for multiprocessors: Methodology and examples," *Computer* 26:7 (July), 22–33.

Slotnick, D. L., W. C. Borck, and R. C. McReynolds [1962]. "The Solomon computer," *Proc. Fall Joint Computer Conf.* (December), Philadelphia, 97–107.

Smith, B. J. [1978]. "A pipelined, shared resource MIMD computer," *Proc. 1978 ICPP* (August), 6–8.

Soundararajan, V., M. Heinrich, B. Verghese, K. Gharachorloo, A. Gupta, and J. L. Hennessy [1998]. "Flexible use of memory for replication/migration in cache-coherent DSM multiprocessors," *Proc. 25th Int'l Symposium on Computer Architecture* (June), Barcelona, 342–355.

Spurgeon, C. [2001]. "Charles Spurgeon's Ethernet Web site," *wwwhost.ots.utexas.edu/ethernet/ethernet-home.html.*

Stenström, P., T. Joe, and A. Gupta [1992]. "Comparative performance evaluation of cache-coherent NUMA and COMA architectures," *Proc. 19th Annual Int'l Symposium on Computer Architecture,* May, Queensland, Australia, 80–91.

Sterling, T. [2001]. *Beowulf PC Cluster Computing with Windows and Beowulf PC Cluster Computing with Linux,* MIT Press, Cambridge, Mass.

Stevens, W. R. [1994–1996]. *TCP/IP Illustrated* (three volumes), Addison-Wesley, Reading, Mass.

Stone, H. [1991]. *High Performance Computers,* Addison-Wesley, New York.

Swan, R. J., A. Bechtolsheim, K. W. Lai, and J. K. Ousterhout [1977]. "The implementation of the Cm* multi-microprocessor," *Proc. AFIPS National Computing Conf.*, 645– 654.

Swan, R. J., S. H. Fuller, and D. P. Siewiorek [1977]. "Cm*—a modular, multi-microprocessor," *Proc. AFIPS National Computer Conf.* 46, 637–644.

Tanenbaum, A. S. [1988]. *Computer Networks,* second edition, Prentice Hall, Englewood Cliffs, N.J.

Tang, C. K. [1976]. "Cache design in the tightly coupled multiprocessor system," *Proc. AFIPS National Computer Conf.,* New York (June), 749–753.

Thacker, C. P., E. M. McCreight, B. W. Lampson, R. F. Sproull, and D. R. Boggs [1982]. Alto: A personal computer," in *Computer Structures: Principles and Examples*, D. P. Siewiorek, C. G. Bell, and A. Newell, eds., McGraw-Hill, New York, 549–572.

Thekkath, R., A. P. Singh, J. P. Singh, S. John, and J. L. Hennessy [1997]. "An evaluation of a commercial CC-NUMA architecture—the CONVEX Exemplar SPP1200," *Proc. 11th Int'l Parallel Processing Symposium (IPPS '97),* Geneva, Switzerland, April.

Tullsen, D. M., S. J. Eggers, J. S. Emer, H. M. Levy, J. L. Lo, and R. L. Stamm [1996]. "Exploiting choice: Instruction fetch and issue on an implementable simultaneous multithreading processor," *Proc. 23rd Annual Int'l Symposium on Computer Architecture* (May), 191–202.

Tullsen, D. M., S. J. Eggers, and H. M. Levy [1995]. "Simultaneous multithreading: Maximizing on-chip parallelism," *Proc. 22nd Int'l Symposium on Computer Architecture* (June), 392–403.

Unger, S. H. [1958]. "A computer oriented towards spatial problems," *Proc. Institute of Radio Engineers* 46:10 (October), 1744–1750.

Walrand, J. [1991]. *Communication Networks: A First Course*, Aksen Associates: Irwin, Homewood, Ill.

Wilson, A. W., Jr. [1987]. "Hierarchical cache/bus architecture for shared-memory multiprocessors," *Proc. 14th Int'l Symposium on Computer Architecture* (June), Pittsburgh, 244–252.

Wolfe, A., and J. P. Shen [1991]. "A variable instruction stream extension to the VLIW architecture." *Proc. Fourth Conference on Architectural Support for Programming Languages and Operating Systems* (April), Santa Clara, Calif., 2–14.

Wood, D. A., and M. D. Hill [1995]. "Cost-effective parallel computing," *IEEE Computer* 28:2 (February).

Wulf, W., and C. G. Bell [1972]. "C.mmp—A multi-mini-processor," *Proc. AFIPS Fall Joint Computing Conf.* 41, part 2, 765–777.

Wulf, W., and S. P. Harbison [1978]. "Reflections in a pool of processors—an experience report on C.mmp/Hydra," *Proc. AFIPS 1978 National Computing Conf.* 48 (June), Anaheim, Calif., 939–951.

Yamamoto, W., M. J. Serrano, A. R. Talcott, R. C. Wood, and M. Nemirosky [1994]. "Performance estimation of multistreamed, superscalar processors," *Proc. 27th Hawaii Int'l Conf. on System Sciences* (January), I:195–204.

K.6 The Development of Memory Hierarchy and Protection (Chapter 5; Appendix C)

Although the pioneers of computing knew of the need for a memory hierarchy and coined the term, the automatic management of two levels was first proposed by Kilburn et al. [1962]. It was demonstrated with the Atlas computer at the University of Manchester. This computer appeared the year *before* the IBM 360 was announced. Although IBM planned for its introduction with the next generation (System/370), the operating system TSS wasn't up to the challenge in 1970. Virtual memory was announced for the 370 family in 1972, and it was for this computer that the term "translation lookaside buffer" was coined [Case and Padegs 1978]. The only computers today without virtual memory are a few supercomputers, embedded processors, and older personal computers.

Both the Atlas and the IBM 360 provided protection on pages, and the GE 645 was the first system to provide paged segmentation. The earlier Burroughs computers provided virtual memory using segmentation, similar to the segmented address scheme of the Intel 8086. The 80286, the first 80x86 to have the protection mechanisms described in Appendix C, was inspired by the Multics protection software that ran on the GE 645. Over time, computers evolved more elaborate mechanisms. The most elaborate mechanism was *capabilities,* which reached its highest interest in the late 1970s and early 1980s [Fabry 1974; Wulf, Levin, and Harbison 1981]. Wilkes [1982], one of the early workers on capabilities, had this to say:

> Anyone who has been concerned with an implementation of the type just described [capability system], or has tried to explain one to others, is likely to feel that complexity has got out of hand. It is particularly disappointing that the attractive idea of capabilities being tickets that can be freely handed around has become lost. . . .
>
> Compared with a conventional computer system, there will inevitably be a cost to be met in providing a system in which the domains of protection are small and frequently changed. This cost will manifest itself in terms of additional hardware, decreased runtime speed, and increased memory occupancy. It is at present an open question whether, by adoption of the capability approach, the cost can be reduced to reasonable proportions. [p. 112]

Today there is little interest in capabilities either from the operating systems or the computer architecture communities, despite growing interest in protection and security.

Bell and Strecker [1976] reflected on the PDP-11 and identified a small address space as the only architectural mistake that is difficult to recover from. At the time of the creation of PDP-11, core memories were increasing at a very slow rate. In addition, competition from 100 other minicomputer companies meant that DEC might not have a cost-competitive product if every address had to go through the 16-bit data path twice; hence, the architect's decision to add only 4 more address bits than found in the predecessor of the PDP-11.

The architects of the IBM 360 were aware of the importance of address size and planned for the architecture to extend to 32 bits of address. Only 24 bits were used in the IBM 360, however, because the low-end 360 models would have been even slower with the larger addresses in 1964. Unfortunately, the architects didn't reveal their plans to the software people, and programmers who stored extra information in the upper 8 "unused" address bits foiled the expansion effort. (Apple made a similar mistake 20 years later with the 24-bit address in the Motorola 68000, which required a procedure to later determine "32-bit clean" programs for the Macintosh when later 68000s used the full 32-bit virtual address.) Virtually every computer since then will check to make sure the unused bits stay unused, and trap if the bits have the wrong value.

As mentioned in the text, system virtual machines were pioneered at IBM as part of its investigation into virtual memory. IBM's first computer with virtual memory was the IBM 360/67, introduced in 1967. IBM researchers wrote the program CP-67 that created the illusion of several independent 360 computers. They then wrote an interactive, single-user operating system called CMS that ran on these virtual machines. CP-67 led to the product VM/370, and today IBM sells z/VM for its mainframe computers [Meyer and Seawright 1970; Van Vleck 2005].

A few years after the Atlas paper, Wilkes published the first paper describing the concept of a cache [1965]:

> The use is discussed of a fast core memory of, say, 32,000 words as slave to a slower core memory of, say, one million words in such a way that in practical cases the effective access time is nearer that of the fast memory than that of the slow memory. [p. 270]

This two-page paper describes a direct-mapped cache. Although this is the first publication on caches, the first implementation was probably a direct-mapped instruction cache built at the University of Cambridge. It was based on tunnel diode memory, the fastest form of memory available at the time. Wilkes states that G. Scarott suggested the idea of a cache memory.

Subsequent to that publication, IBM started a project that led to the first commercial computer with a cache, the IBM 360/85 [Liptay 1968]. Gibson [1967] describes how to measure program behavior as memory traffic as well as miss rate and shows how the miss rate varies between programs. Using a sample of 20 programs (each with 3 million references!), Gibson also relied on average memory access time to compare systems with and without caches. This precedent is more than 30 years old, and yet many used miss rates until the early 1990s.

Conti, Gibson, and Pitkowsky [1968] describe the resulting performance of the 360/85. The 360/91 outperforms the 360/85 on only 3 of the 11 programs in the paper, even though the 360/85 has a slower clock cycle time (80 ns versus 60 ns), less memory interleaving (4 versus 16), and a slower main memory (1.04 microsecond versus 0.75 microsecond). This paper was also the first to use the term "cache."

Others soon expanded the cache literature. Strecker [1976] published the first comparative cache design paper examining caches for the PDP-11. Smith

[1982] later published a thorough survey paper, using the terms "spatial locality" and "temporal locality"; this paper has served as a reference for many computer designers.

Although most studies relied on simulations, Clark [1983] used a hardware monitor to record cache misses of the VAX-11/780 over several days. Clark and Emer [1985] later compared simulations and hardware measurements for translations.

Hill [1987] proposed the three C's used in Section 5.1 and Appendix C to explain cache misses. Jouppi [1998] retrospectively says that Hill's three C's model led directly to his invention of the victim cache to take advantage of faster direct-mapped caches and yet avoid most of the cost of conflict misses. Sugumar and Abraham [1993] argue that the baseline cache for the three C's model should use optimal replacement; this eliminates the anomalies of LRU-based miss classification and allows conflict misses to be broken down into those caused by mapping and those caused by a nonoptimal replacement algorithm.

One of the first papers on nonblocking caches is by Kroft [1981]. Kroft [1998] later explained that he was the first to design a computer with a cache at Control Data Corporation, and when using old concepts for new mechanisms, he hit upon the idea of allowing his two-ported cache to continue to service other accesses on a miss.

Baer and Wang [1988] did one of the first examinations of the multilevel inclusion property. Wang, Baer, and Levy [1989] then produced an early paper on performance evaluation of multilevel caches. Later, Jouppi and Wilton [1994] proposed multilevel exclusion for multilevel caches on chip.

In addition to victim caches, Jouppi [1990] also examined prefetching via streaming buffers. His work was extended by Farkas, Jouppi, and Chow [1995] to streaming buffers that work well with nonblocking loads and speculative execution for in-order processors, and later Farkas et al. [1997] showed that while out-of-order processors can tolerate unpredictable latency better, they still benefit. They also refined memory bandwidth demands of stream buffers.

Proceedings of the Symposium on Architectural Support for Compilers and Operating Systems (ASPLOS) and the International Computer Architecture Symposium (ISCA) from the 1990s are filled with papers on caches. (In fact, some wags claimed ISCA really stood for the International *Cache* Architecture Symposium.)

Chapter 5 relies on the measurements of SPEC2000 benchmarks collected by Cantin and Hill [2001]. There are several other papers used in Chapter 5 that are cited in the captions of the figures that use the data: Agarwal and Pudar [1993]; Barroso, Gharachorloo, and Bugnion [1998]; Farkas and Jouppi [1994]; Jouppi [1990]; Lam, Rothberg, and Wolf [1991]; Lebeck and Wood [1994]; McCalpin [2005]; Mowry, Lam, and Gupta [1992]; and Torrellas, Gupta, and Hennessy [1992].

References

Agarwal, A. [1987]. *Analysis of Cache Performance for Operating Systems and Multiprogramming*, Ph.D. thesis, Stanford Univ., Tech. Rep. No. CSL-TR-87-332 (May).

Agarwal, A., and S. D. Pudar [1993]. "Column-associative caches: A technique for reducing the miss rate of direct-mapped caches," 20th Annual Int'l Symposium on Computer Architecture ISCA '20, San Diego, Calif., May 16–19. *Computer Architecture News* 21:2 (May), 179–190.

Baer, J.-L., and W.-H. Wang [1988]. "On the inclusion property for multi-level cache hierarchies," *Proc. 15th Annual Symposium on Computer Architecture* (May–June), Honolulu, 73–80.

Barham, P., B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, and R. Neugebauer [2003]. "Xen and the art of virtualization," *Proc. of the ACM Symposium on Operating Systems Principles.*

Barroso, L. A., K. Gharachorloo, and E. Bugnion [1998]. "Memory system characterization of commercial workloads," *Proc. 25th Int'l Symposium on Computer Architecture,* Barcelona (July), 3–14.

Bell, C. G,. and W. D. Strecker [1976]. "Computer structures: What have we learned from the PDP-11?," *Proc. Third Annual Symposium on Computer Architecture* (January), Pittsburgh, 1–14.

Bhandarkar, D. P. [1995]. *Alpha Architecture Implementations,* Digital Press, Newton, Mass.

Borg, A., R. E. Kessler, and D. W. Wall [1990]. "Generation and analysis of very long address traces," *Proc. 17th Annual Int'l Symposium on Computer Architecture*, Seattle, Wash., May 28–31, 270–279.

Cantin, J. F., and M. D. Hill [2001]. *Cache Performance for Selected SPEC CPU2000 Benchmarks, www.jfred.org/cache-data.html* (June).

Cantin, J., and M. Hill [2003]. "Cache performance for SPEC CPU2000 benchmarks, version 3.0," *http://www.cs.wisc.edu/multifacet/misc/spec2000cache-data/index.html.*

Case, R. P., and A. Padegs [1978]. "The architecture of the IBM System/370," *Communications of the ACM* 21:1, 73–96. Also appears in D. P. Siewiorek, C. G. Bell, and A. Newell, *Computer Structures: Principles and Examples,* McGraw-Hill, New York (1982), 830–855.

Clark, B., T. Deshane, E. Dow, S. Evanchik, M. Finlayson, J. Herne,and J. Neefe Matthews [2004]. "Xen and the art of repeated research," *Proc. USENIX Annual Technical Conf.,* 135–144.

Clark, D. W. [1983]. "Cache performance of the VAX-11/780," *ACM Trans. on Computer Systems* 1:1, 24–37.

Clark, D. W., and J. S. Emer [1985]. "Performance of the VAX-11/780 translation buffer: Simulation and measurement," *ACM Trans. on Computer Systems* 3:1 (February), 31–62.

Compaq Computer Corporation [1999]. *Compiler Writer's Guide for the Alpha 21264,* Order Number EC-RJ66A-TE, June, *www1.support.compaq.com/alpha-tools/documentation/current/21264_EV67/ec-rj66a-te_comp_writ_gde_for_alpha21264.pdf.*

Conti, C., D. H. Gibson, and S. H. Pitkowsky [1968]. "Structural aspects of the System/360 Model 85, Part I: General organization," *IBM Systems J.* 7:1, 2–14.

Crawford, J., and P. Gelsinger [1988]. *Programming the 80386*, Sybex, Alameda, Calif.

Cvetanovic, Z., and R. E. Kessler [2000]. "Performance analysis of the Alpha 21264-based Compaq ES40 system," *Proc. 27th Annual Int'l Symposium on Computer Architecture*, Vancouver, Canada, June 10–14, IEEE Computer Society Press, 192–202.

Fabry, R. S. [1974]. "Capability based addressing," *Comm. ACM* 17:7 (July), 403–412.

Farkas, K. I., P. Chow, N. P. Jouppi, and Z. Vranesic [1997]. "Memory-system design considerations for dynamically-scheduled processors," *Proc. 24th Annual Int'l Symposium on Computer Architecture,* Denver, Col., June 2–4, 133–143.

Farkas, K. I., and N. P. Jouppi [1994]. "Complexity/performance trade-offs with non-blocking loads," *Proc. 21st Annual Int'l Symposium on Computer Architecture*, Chicago (April).

Farkas, K. I., N. P. Jouppi, and P. Chow [1995]. "How useful are non-blocking loads, stream buffers and speculative execution in multiple issue processors?," *Proc. First IEEE Symposium on High-Performance Computer Architecture,* Raleigh, N.C., January 22–25, 78–89.

Gao, Q. S. [1993]. "The Chinese remainder theorem and the prime memory system," 20th Annual Int'l Symposium on Computer Architecture ISCA '20, San Diego, Calif., May 16–19, *Computer Architecture News* 21:2 (May), 337–340.

Gee, J. D., M. D. Hill, D. N. Pnevmatikatos, and A. J. Smith [1993]. "Cache performance of the SPEC92 benchmark suite," *IEEE Micro* 13:4 (August), 17–27.

Gibson, D. H. [1967]. "Considerations in block-oriented systems design," *AFIPS Conf. Proc.* 30, SJCC, 75–80.

Handy, J. [1993]. *The Cache Memory Book*, Academic Press, Boston.

Heald, R., K. Aingaran, C. Amir, M. Ang, M. Boland, A. Das, P. Dixit, G. Gouldsberry, J. Hart, T. Horel, W.-J. Hsu, J. Kaku, C. Kim, S. Kim, F. Klass, H. Kwan, R. Lo, H. McIntyre, A. Mehta, D. Murata, S. Nguyen, Y.-P. Pai, S. Patel, K. Shin, K. Tam, S. Vishwanthaiah, J. Wu, G. Yee, and H. You [2000]. "Implementation of third-generation SPARC V9 64-b microprocessor," *ISSCC Digest of Technical Papers,* 412–413 and slide supplement.

Hill, M. D. [1987]. *Aspects of Cache Memory and Instruction Buffer Performance*, Ph.D. thesis, University of Calif. at Berkeley, Computer Science Division, Tech. Rep. UCB/CSD 87/381 (November).

Hill, M. D. [1988]. "A case for direct mapped caches," *Computer* 21:12 (December), 25– 40.

Horel, T., and G. Lauterbach [1999]. "UltraSPARC-III: Designing third-generation 64-bit performance," *IEEE Micro* 19:3 (May–June), 73–85.

Hughes, C. J., P. Kaul, S. V. Adve, R. Jain, C. Park, and J. Srinivasan [2001]. "Variability in the execution of multimedia applications and implications for architecture," *Proc. 28th Annual Int'l Symposium on Computer Architecture,* Goteborg, Sweden, June 30–July 4, 254–265.

IEEE [2005]. "Intel virtualization technology, computer," *IEEE Computer Society* 38:5 (May), 48–56.

Jouppi, N. P. [1990]. "Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers," *Proc. 17th Annual Int'l Symposium on Computer Architecture*, 364–73.

Jouppi, N. P. [1998]. "Retrospective: Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers," *25 Years of the Int'l Symposia on Computer Architecture (Selected Papers)*, ACM, 71–73.

Jouppi, N. P., and S. J. E. Wilton [1994]. "Trade-offs in two-level on-chip caching," *Proc. 21st Annual Int'l Symposium on Computer Architecture,* Chicago, April 18–21, 34–45.

Kessler, R. E. [1999]. "The Alpha 21264 microprocessor," *IEEE Micro* 19:2 (March/April), 24–36.

Kilburn, T., D. B. G. Edwards, M. J. Lanigan, and F. H. Sumner [1962]. "One-level storage system," *IRE Trans. on Electronic Computers* EC-11 (April) 223–235. Also appears in D. P. Siewiorek, C. G. Bell, and A. Newell, *Computer Structures: Principles and Examples* (1982), McGraw-Hill, New York, 135–148.

Kroft, D. [1981]. "Lockup-free instruction fetch/prefetch cache organization," *Proc. Eighth Annual Symposium on Computer Architecture* (May 12–14), Minneapolis, 81–87.

Kroft, D. [1998]. "Retrospective: Lockup-free instruction fetch/prefetch cache organization," *25 Years of the Int'l Symposia on Computer Architecture (Selected Papers),* ACM, 20–21.

Kunimatsu, A., N. Ide, T. Sato, Y. Endo, H. Murakami, T. Kamei, M. Hirano, F. Ishihara, H. Tago, M. Oka, A. Ohba, T. Yutaka, T. Okada, and M. Suzuoki [2000]. "Vector unit architecture for emotion synthesis," *IEEE Micro* 20:2 (March–April), 40–47.

Lam, M. S., E. E. Rothberg, and M. E. Wolf [1991]. "The cache performance and optimizations of blocked algorithms," Fourth Int'l Conf. on Architectural Support for Programming Languages and Operating Systems, Santa Clara, Calif., April 8–11. *SIGPLAN Notices* 26:4 (April), 63–74.

Lebeck, A. R., and D. A. Wood [1994]. "Cache profiling and the SPEC benchmarks: A case study," *Computer* 27:10 (October), 15–26.

Liptay, J. S. [1968]. "Structural aspects of the System/360 Model 85, Part II: The cache," *IBM Systems J.* 7:1, 15–21.

Luk, C.-K., and T. C Mowry [1999]. "Automatic compiler-inserted prefetching for pointer-based applications," *IEEE Trans. on Computers,* 48:2 (February), 134–141.

McCalpin, J. D. [2005]. *STREAM: Sustainable Memory Bandwidth in High Performance Computers, www.cs.virginia.edu/stream/.*

McFarling, S. [1989]. "Program optimization for instruction caches," *Proc. Third Int'l Conf. on Architectural Support for Programming Languages and Operating Systems* (April 3–6), Boston, 183–191.

Menon, A., J. Renato Santos, Y. Turner, G. Janakiraman, W. Zwaenepoel [2005]. "Diagnosing performance overheads in the xen virtual machine environment," *Proc. 1st ACM/USENIX Int'l. Conf. on Virtual Execution Environments,* Chicago, 13–23.

Meyer, R. A., and L. H. Seawright [1970]. "A virtual machine time sharing system," *IBM Systems J.* 9:3, 199–218.

Mowry, T. C., S. Lam, and A. Gupta [1992]. "Design and evaluation of a compiler algorithm for prefetching," Fifth Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS-V), Boston, October 12–15, *SIGPLAN Notices* 27:9 (September), 62–73.

Oka, M., and M. Suzuoki [1999]. "Designing and programming the emotion engine," *IEEE Micro* 19:6 (November–December), 20–28.

Pabst, T. [2000]. "Performance showdown at 133 MHz FSB—the best platform for coppermine," *www6.tomshardware.com/mainboard/00q1/000302/.*

Palacharla, S., and R. E. Kessler [1994]. "Evaluating stream buffers as a secondary cache replacement," *Proc. 21st Annual Int'l Symposium on Computer Architecture*, Chicago, April 18–21, 24–33.

Przybylski, S. A. [1990]. *Cache Design: A Performance-Directed Approach,* Morgan Kaufmann, San Francisco.

Przybylski, S. A., M. Horowitz, and J. L. Hennessy [1988]. "Performance trade-offs in cache design," *Proc. 15th Annual Symposium on Computer Architecture* (May–June), Honolulu, 290–298.

Reinman, G., and N. P. Jouppi. [1999]. "Extensions to CACTI," *research.compaq.com/wrl/people/jouppi/CACTI.html.*

Robin, J., and C. Irvine [2000]. "Analysis of the Intel Pentium's ability to support a secure virtual machine monitor," *Proc. USENIX Security Symposium,* Denver, Colo., August 14–17.

Saavedra-Barrera, R. H. [1992]. *CPU Performance Evaluation and Execution Time Prediction Using Narrow Spectrum Benchmarking*, Ph.D. dissertation, University of Calif., Berkeley (May).

Samples, A. D., and P. N. Hilfinger [1988]. "Code reorganization for instruction caches," Tech. Rep. UCB/CSD 88/447 (October), University of Calif., Berkeley.

Sites, R. L. (ed.) [1992]. *Alpha Architecture Reference Manual,* Digital Press, Burlington, Mass.

Skadron, K., and D. W. Clark [1997]. "Design issues and tradeoffs for write buffers," *Proc. Third Int'l Symposium on High-Performance Computer Architecture,* 144–155.

Smith, A. J. [1982]. "Cache memories," *Computing Surveys* 14:3 (September), 473–530.

Smith, J. E., and J. R. Goodman [1983]. "A study of instruction cache organizations and replacement policies," *Proc. 10th Annual Symposium on Computer Architecture* (June 5–7), Stockholm, 132–137.

Stokes, J. [2000]. "Sound and vision: A technical overview of the emotion engine," *arstechnica.com/reviews/1q00/playstation2/ee-1.html.*

Strecker, W. D. [1976]. "Cache memories for the PDP-11?," *Proc. Third Annual Symposium on Computer Architecture* (January), Pittsburgh, 155–158.

Sugumar, R. A., and S. G. Abraham [1993]. "Efficient simulation of caches under optimal replacement with applications to miss characterization," *1993 ACM Sigmetrics Conf. on Measurement and Modeling of Computer Systems,* Santa Clara, Calif., May 17–21, 24–35.

Tarjan, D., and N. Jouppi [2005]. HPL Technical Report on CACTI 4.0.

Torrellas, J., A. Gupta, and J. Hennessy [1992]. "Characterizing the caching and synchronization performance of a multiprocessor operating system," Fifth Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS-V), Boston, October 12–15, *SIGPLAN Notices* 27:9 (September), 162–174.

Van Vleck, T. [2005]. "The IBM 360/67 and CP/CMS," *http://www.multicians.org/thvv/360-67.html.*

Wang, W.-H., J.-L. Baer, and H. M. Levy [1989]. "Organization and performance of a two-level virtual-real cache hierarchy," *Proc. 16th Annual Symposium on Computer Architecture* (May 28–June 1), Jerusalem, 140–148.

Wilkes, M. [1965]. "Slave memories and dynamic storage allocation," *IEEE Trans. Electronic Computers* EC-14:2 (April), 270–271.

Wilkes, M. V. [1982]. "Hardware support for memory protection: Capability implementations," *Proc. Symposium on Architectural Support for Programming Languages and Operating Systems* (March 1–3), Palo Alto, Calif., 107–116.

Wulf, W. A., R. Levin, and S. P. Harbison [1981]. *Hydra/C.mmp: An Experimental Computer System,* McGraw-Hill, New York.

K.7 The History of Magnetic Storage, RAID, and I/O Buses (Chapter 6)

Mass storage is a term used there to imply a unit capacity in excess of one million alphanumeric characters…

Hoagland [1963]

The variety of storage I/O and issues leads to a varied history for the rest of the story. (Smotherman [1989] explores the history of I/O in more depth.) This section discusses magnetic storage, RAID, and I/O buses and controllers. Jain [1991] and Lazowska et al. [1984] offer books for those interested in learning more about queuing theory.

Magnetic Storage

Magnetic recording was invented to record sound, and by 1941, magnetic tape was able to compete with other storage devices. It was the success of the ENIAC in 1947 that led to the push to use tapes to record digital information. Reels of magnetic tapes dominated removable storage through the 1970s. In the 1980s, the IBM 3480 cartridge became the de facto standard, at least for mainframes. It can transfer at 3 MB/sec by reading 18 tracks in parallel. The capacity is just 200 MB for this 1/2-inch tape. The 9840 cartridge, used by StorageTek in the PowderHorn, transfers at 10 MB/sec and stores 20,000 MB. This device records the tracks in a zigzag fashion rather than just longitudinally, so that the head reverses direction to follow the track. This technique is called *serpentine recording*. Another 1/2-inch tape is Digital Linear Tape, with DLT7000 storing 35,000 MB and transferring at 5 MB/sec. Its competitor is helical scan, which rotates the head to get the increased recording density. In 2001, the 8 mm helical-scan tapes contain 20,000 MB and transfer at about 3 MB/sec. Whatever their density and cost, the serial nature of tapes creates an appetite for storage devices with random access.

In 1953, Reynold B. Johnson of IBM picked a staff of 15 scientists with the goal of building a radically faster random access storage system than tape. The goal was to have the storage equivalent of 50,000 standard IBM punch cards and to fetch the data in a single second. Johnson's disk drive design was simple but untried: The magnetic read/write sensors would have to float a few thousandths of an inch above the continuously rotating disk. Twenty-four months later the team emerged with the functional prototype. It weighed one ton and occupied about 300 cubic feet of space. The RAMAC-350 (Random Access Method of Accounting Control) used 50 platters that were 24 inches in diameter, rotated at 1200 RPM, with a total capacity of 5 MB and an access time of 1 second.

Starting with the RAMAC, IBM maintained its leadership in the disk industry, with its storage headquarters in San Jose, California, where Johnson's team did its work. Many of the future leaders of competing disk manufacturers started their careers at IBM, and many disk companies are located near San Jose.

Although RAMAC contained the first disk, a major breakthrough in magnetic recording was found in later disks with air-bearing read/write heads, where the head would ride on a cushion of air created by the fast-moving disk surface. This cushion meant the head could both follow imperfections in the surface and yet be very close to the surface. Subsequent advances have come largely from improved quality of components and higher precision. In 2001, heads fly 2–3 microinches above the surface, whereas in the RAMAC drive they were 1000 microinches away.

Moving-head disks quickly became the dominant high-speed magnetic storage, although their high cost meant that magnetic tape continued to be used extensively until the 1970s. The next important development for hard disks was the removable hard disk drive developed by IBM in 1962; this made it possible to share the expensive drive electronics and helped disks overtake tapes as the preferred storage medium. The IBM 1311 disk in 1962 had an areal density of 50,000 bits per square inch and a cost of about $800 per megabyte.

IBM also invented the floppy disk drive in 1970, originally to hold microcode for the IBM 370 series. Floppy disks became popular with the PC about 10 years later.

The second major disk breakthrough was the so-called Winchester disk design in about 1973. Winchester disks benefited from two related properties. First, integrated circuits lowered the costs of not only CPUs, but also of disk controllers and the electronics to control disk arms. Reductions in the cost of the disk electronics made it unnecessary to share the electronics, and thus made nonremovable disks economical. Since the disk was fixed and could be in a sealed enclosure, both the environmental and control problems were greatly reduced. Sealing the system allowed the heads to fly closer to the surface, which in turn enabled increases in areal density. The first sealed disk that IBM shipped had two spindles, each with a 30 MB disk; the moniker “30-30” for the disk led to the name Winchester. (America’s most popular sporting rifle, the Winchester 94, was nicknamed the “30-30” after the caliber of its cartridge.) Winchester disks grew rapidly in popularity in the 1980s, completely replacing removable disks by the middle of that decade. Before this time, the cost of the electronics to control the disk meant that the media had to be removable.

As mentioned in Chapter 6, as DRAMs started to close the areal density gap and appeared to be catching up with disk storage, internal meetings at IBM called into question the future of disk drives. Disk designers concluded that disks must improve at 60% per year to forestall the DRAM threat, in contrast to the historical 29% per year. The essential enabler was magnetoresistive heads, with giant magnetoresistive heads enabling the current densities.

Because of this competition, the gap in time between when a density record is achieved in the lab and when a disk is shipped with that density has closed considerably.

The personal computer created a market for small form factor disk drives, since the 14-inch disk drives used in mainframes were bigger than the PC. In 2006, the 3.5-inch drive was the market leader, although the smaller 2.5-inch

drive needed for laptop computers was significant in sales volume. It remains to be seen whether handheld devices like iPODs or video cameras, requiring even smaller disks, will remain significant in sales volume . For example, 1.8-inch drives were developed in the early 1990s for palmtop computers, but that market chose Flash instead, and hence 1.8-inch drives disappeared.

RAID

The small form factor hard disks for PCs in the 1980s led a group at Berkeley to propose redundant arrays of inexpensive disks (RAID). This group had worked on the reduced instruction set computers effort, and so expected much faster CPUs to become available. Their questions were, What could be done with the small disks that accompanied their PCs? and What could be done in the area of I/O to keep up with much faster processors? They argued to replace one mainframe drive with 50 small drives, as you could get much greater performance with that many independent arms. The many small drives even offered savings in power consumption and floor space.

The downside of many disks was much lower MTTF. Hence, on their own they reasoned out the advantages of redundant disks and rotating parity to address how to get greater performance with many small drives yet have reliability as high as that of a single mainframe disk.

The problem they experienced when explaining their ideas was that some researchers had heard of disk arrays with some form of redundancy, and they didn't understand the Berkeley proposal. Hence, the first RAID paper [Patterson, Gibson, and Katz 1987] is not only a case for arrays of small form factor disk drives, but something of a tutorial and classification of existing work on disk arrays. Mirroring (RAID 1) had long been used in fault-tolerant computers such as those sold by Tandem; Thinking Machines had arrays with 32 data disks and 7 check disks using ECC for correction (RAID 2) in 1987, and Honeywell Bull had a RAID 2 product even earlier; and disk arrays with a single parity disk had been used in scientific computers in the same time frame (RAID 3). Their paper then described a single parity disk with support for sector accesses (RAID 4) and rotated parity (RAID 5). Chen et al. [1994] survey the original RAID ideas, commercial products, and more recent developments.

Unknown to the Berkeley group, engineers at IBM working on the AS/400 computer also came up with rotated parity to give greater reliability for a collection of large disks. IBM filed a patent on RAID 5 before the Berkeley group wrote their paper. Patents for RAID 1, RAID 2, and RAID 3 from several companies predate the IBM RAID 5 patent, which has led to plenty of courtroom action.

The Berkeley paper was written before the World Wide Web, but it captured the imagination of many engineers, as copies were faxed around the world. One engineer at what is now Seagate received seven copies of the paper from friends and customers.

EMC had been a supplier of DRAM boards for IBM computers, but around 1988 new policies from IBM made it nearly impossible for EMC to continue to sell IBM memory boards. Apparently, the Berkeley paper also crossed the desks of EMC executives, and so they decided to go after the market dominated by IBM disk storage products instead. As the paper advocated, their model was to use many small drives to compete with mainframe drives, and EMC announced a RAID product in 1990. It relied on mirroring (RAID 1) for reliability; RAID 5 products came much later for EMC. Over the next year, Micropolis offered a RAID 3 product; Compaq offered a RAID 4 product; and Data General, IBM, and NCR offered RAID 5 products.

The RAID ideas soon spread to the rest of the workstation and server industry. An article explaining RAID in *Byte* magazine (see Anderson [1990]) led to RAID products being offered on desktop PCs, which was something of a surprise to the Berkeley group. They had focused on performance with good availability, but higher availability was attractive to the PC market.

Another surprise was the cost of the disk arrays. With redundant power supplies and fans, the ability to "hot swap" a disk drive, the RAID hardware controller itself, the redundant disks, and so on, the first disk arrays cost many times the cost of the disks. Perhaps as a result, the "inexpensive" in RAID morphed into "independent." Many marketing departments and technical writers today know of RAID only as "redundant arrays of independent disks."

The EMC transformation was successful; in 2006 EMC was the leading supplier of storage systems, and NetApp was the leading supplier of Network-Attached Storage systems. RAID was a $30 billion industry in 2006, and more than 80% of the non-PC drive sales were found in RAIDs.

In recognition of their role, in 1999 Garth Gibson, Randy Katz, and David Patterson received the IEEE Reynold B. Johnson Information Storage Award "for the development of Redundant Arrays of Inexpensive Disks (RAID)."

I/O Buses and Controllers

The ubiquitous microprocessor has inspired not only the personal computers of the 1970s, but also the trend in the late 1980s and 1990s of moving controller functions into I/O devices. I/O devices continued this trend by moving controllers into the devices themselves. These devices are called *intelligent devices,* and some bus standards (e.g., SCSI) have been created specifically for them. Intelligent devices can relax the timing constraints by handling many low-level tasks themselves and queuing the results. For example, many SCSI-compatible disk drives include a track buffer on the disk itself, supporting read ahead and connect/disconnect. Thus, on a SCSI string some disks can be seeking and others loading their track buffer while one is transferring data from its buffer over the SCSI bus. The controller in the original RAMAC, built from vacuum tubes, only needed to move the head over the desired track, wait for the data to pass under the head, and transfer data with calculated parity.

SCSI, which stands for *small computer systems interface,* is an example of one company inventing a bus and generously encouraging other companies to build devices that would plug into it. Shugart created this bus, originally called SASI. It was later standardized by the IEEE.

There have been several candidates to be the successor to SCSI, with the current leading contender being Fibre Channel Arbitrated Loop (FC-AL). The SCSI committee continues to increase the clock rate of the bus, giving this standard a new life, and SCSI is lasting much longer than some of its proposed successors. With the creation of serial interfaces for SCSI ("Serial Attach SCSI") and ATA ("Serial ATA"), they may have very long lives.

Perhaps the first multivendor bus was the PDP-11 Unibus in 1970 from DEC. Alas, this open-door policy on buses is in contrast to companies with proprietary buses using patented interfaces, thereby preventing competition from plug-compatible vendors. Making a bus proprietary also raises costs and lowers the number of available I/O devices that plug into it, since such devices must have an interface designed just for that bus. The PCI bus pushed by Intel represented a return to open, standard I/O buses inside computers. Its immediate successor is PCI-X, with Infiniband under development in 2000. Both were standardized by multicompany trade associations.

The machines of the RAMAC era gave us I/O interrupts as well as storage devices. The first machine to extend interrupts from detecting arithmetic abnormalities to detecting asynchronous I/O events is credited as the NBS DYSEAC in 1954 [Leiner and Alexander 1954]. The following year, the first machine with DMA was operational, the IBM SAGE. Just as today's DMA has, the SAGE had address counters that performed block transfers in parallel with CPU operations.

The early IBM 360s pioneered many of the ideas that we use in I/O systems today. The 360 was the first commercial machine to make heavy use of DMA, and it introduced the notion of I/O programs that could be interpreted by the device. Chaining of I/O programs was an important feature. The concept of channels introduced in the 360 corresponds to the I/O bus of today.

Myer and Sutherland [1968] wrote a classic paper on the trade-off of complexity and performance in I/O controllers. Borrowing the religious concept of the "wheel of reincarnation," they eventually noticed they were caught in a loop of continuously increasing the power of an I/O processor until it needed its own simpler coprocessor. Their quote in Chapter 6 captures their cautionary tale.

The IBM mainframe I/O channels, with their I/O processors, can be thought of as an inspiration for Infiniband, with their processors on their Host Channel Adaptor cards.

References

Anderson, D. [2003]. "You don't know jack about disks," *Queue* 1:4 (June), 20–30.

Anderson, D., J. Dykes, and E. Riedel [2003]. "SCSI vs. ATA—More than an interface," *Conf. on File and Storage Technology (FAST),* San Francisco, April.

Anderson, M. H. [1990]. "Strength (and safety) in numbers (RAID, disk storage technology)," *Byte* 15:13 (December), 337–339.

Anon. et al. [1985]. "A measure of transaction processing power," Tandem Tech. Rep. TR 85.2. Also appeared in *Datamation,* 31:7 (April), 112–118.

Bashe, C. J., W. Buchholz, G. V. Hawkins, J. L. Ingram, and N. Rochester [1981]. "The architecture of IBM's early computers," *IBM J. Research and Development* 25:5 (September), 363–375.

Bashe, C. J., L. R. Johnson, J. H. Palmer, and E. W. Pugh [1986]. *IBM's Early Computers*, MIT Press, Cambridge, Mass.

Blaum, M., J. Brady, J. Bruck, and J. Menon [1994]. "EVENODD: An optimal scheme for tolerating double disk failures in RAID architectures," *Proc. 21st Annual Symposium on Computer Architecture* (April), Chicago, Ill., 245–254.

Blaum, M., J. Brady, J. Bruck, and J. Menon [1995]."EVENODD: An optimal scheme for tolerating double disk failures in RAID architectures," *IEEE Trans. on Computers* 44:2, (February), 192–202.

Blaum, M., J. Brady, J., Bruck, J. Menon, and A. Vardy [2001]. "The EVENODD code and its generalization," *High Performance Mass Storage and Parallel I/O: Technologies and Applications*, edited by H. Jin, T. Cortes, and R. Buyya, IEEE & Wiley Press, New York, Chapter 14, 187–208.

Blaum, M., J. Bruck, and A. Vardy [1996]. "MDS array codes with independent parity symbols," *IEEE Trans. on Information Theory*, IT-42 (March), 529–542.

Brady, J. T. [1986]. "A theory of productivity in the creative process," *IEEE CG&A* (May), 25–34.

Brown, A., and D. A. Patterson [2000]. "Towards maintainability, availability, and growth benchmarks: A case study of software RAID systems." *Proc. 2000 USENIX Annual Technical Conf.* (June), San Diego, Calif.

Bucher, I. V., and A. H. Hayes [1980]. "I/O performance measurement on Cray-1 and CDC 7000 computers," *Proc. Computer Performance Evaluation Users Group, 16th Meeting*, NBS 500-65, 245–254.

Chen, P. M., G. A. Gibson, R. H. Katz, and D. A. Patterson [1990]. "An evaluation of redundant arrays of inexpensive disks using an Amdahl 5890," *Proc. 1990 ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems* (May), Boulder, Colo.

Chen, P. M., and E. K. Lee [1995]. "Striping in a RAID level 5 disk array," *Proc. 1995 ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems* (May), 136–145.

Chen, P. M., E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson [1994]. "RAID: High-performance, reliable secondary storage," *ACM Computing Surveys* 26:2 (June), 145–188.

Corbett, P., B. English, A. Goel, T. Grcanac, S. Kleiman, J. Leong, and S. Sankar [2004]. "Row-diagonal parity for double disk failure correction," *Proc. FAST '04.*

Denehy, T. E., J. Bent, F. I. Popovici, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau [2004]. "Deconstructing storage arrays," *Proc. 11th Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS XI),* 59–71, Boston, Mass., October.

Doherty, W. J., and R. P. Kelisky [1979]. "Managing VM/CMS systems for user effectiveness," *IBM Systems J.* 18:1, 143–166.

Douceur, J. R., and W. J. Bolosky [1999]. "A large scale study of file-system contents," *Proc. 1999 ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems (SIGMETRICS '99),* 59–69, Atlanta, Ga. May.

Enriquez, P. [2001]. "What happened to my dial tone? A study of FCC service disruption reports," poster, *Richard Tapia Symposium on the Celebration of Diversity in Computing,* October 18–20, Houston, Tex.

Friesenborg, S. E., and R. J. Wicks [1985]. "DASD expectations: The 3380, 3380-23, and MVS/XA," Tech. Bulletin GG22-9363-02 (July 10), Washington Systems Center.

Gibson, G. A. [1992]. *Redundant Disk Arrays: Reliable, Parallel Secondary Storage,* ACM Distinguished Dissertation Series, MIT Press, Cambridge, Mass.

Goldstein, S. [1987]. "Storage performance—an eight year outlook," Tech. Rep. TR 03.308-1 (October), Santa Teresa Laboratory, IBM, San Jose, Calif.

Gray, J. [1990]. "A census of Tandem system availability between 1985 and 1990," *IEEE Transactions on Reliability*, 39:4 (October), 409–418.

Gray, J. (ed.) [1993]. *The Benchmark Handbook for Database and Transaction Processing Systems,* second edition, Morgan Kaufmann, San Francisco.

Gray, J., and A. Reuter [1993]. *Transaction Processing: Concepts and Techniques,* Morgan Kaufmann, San Francisco.

Gray, J., and D. P. Siewiorek [1991]. "High-availability computer systems." *Computer* 24:9 (September), 39–48.

Gray, J., and C. van Ingen [2005]. "Empirical measurements of disk failure rates and error rates," Microsoft Research, MSR-TR-2005-166, December.

Henly, M., and B. McNutt [1989]. "DASD I/O characteristics: A comparison of MVS to VM," Tech. Rep. TR 02.1550 (May), IBM, General Products Division, San Jose, Calif.

Hewlett-Packard [1998]. "HP's '5NINES:5MINUTES' vision extends leadership and redefines high availability in mission-critical environments" (February 10), *www.future.enterprisecomputing.hp.com/ia64/news/5nines_vision_pr.html.*

Hoagland, A. S. [1963]. *Digital Magnetic Recording,* Wiley, New York.

Hospodor, A. D., and A. S. Hoagland [1993]. "The changing nature of disk controllers." *Proc. IEEE* 81:4 (April), 586–594.

IBM [1982]. *The Economic Value of Rapid Response Time,* GE20-0752-0, White Plains, N.Y., 11–82.

Imprimis [1989]. *Imprimis Product Specification,* 97209 Sabre Disk Drive IPI-2 Interface 1.2 GB, Document No. 64402302 (May).

Jain, R. [1991]. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*, Wiley, New York.

Katz, R. H., D. A. Patterson, and G. A. Gibson [1989]. "Disk system architectures for high performance computing," *Proc. IEEE* 77:12 (December), 1842–1858.

Kim, M. Y. [1986]. "Synchronized disk interleaving," *IEEE Trans. on Computers* C-35:11 (November), 978–988.

Kuhn, D. R. [1997]. "Sources of failure in the public switched telephone network," *IEEE Computer* 30:4 (April), 31–36.

Lambright, D. [2000]. "Experiences in measuring the reliability of a cache-based storage system," *Proc. of First Workshop on Industrial Experiences with Systems Software* (WIESS 2000), collocated with the 4th Symposium on Operating Systems Design and Implementation (OSDI), San Diego, Calif. (October 22).

Laprie, J.-C. [1985]. "Dependable computing and fault tolerance: Concepts and terminology," *Fifteenth Annual Int'l Symposium on Fault-Tolerant Computing FTCS 15*. Digest of Papers. Ann Arbor, Mich. (June 19–21), 2–11.

Lazowska, E. D., J. Zahorjan, G. S. Graham, and K. C. Sevcik [1984]. *Quantitative System Performance: Computer System Analysis Using Queueing Network Models,* Prentice Hall, Englewood Cliffs, N.J. (Although out of print, it is available online at *www. cs.washington.edu/homes/lazowska/qsp/.)*

Leiner, A. L. [1954]. "System specifications for the DYSEAC," *J. ACM* 1:2 (April), 57– 81.

Leiner, A. L., and S. N. Alexander [1954]. "System organization of the DYSEAC," *IRE Trans. of Electronic Computers* EC-3:1 (March), 1–10.

Maberly, N. C. [1966]. *Mastering Speed Reading,* New American Library, New York.

Major, J. B. [1989]. "Are queuing models within the grasp of the unwashed?," *Proc. Int'l Conf. on Management and Performance Evaluation of Computer Systems,* Reno, Nev. (December 11–15), 831–839.

Mueller, M., L. C. Alves, W. Fischer, M. L. Fair, I. Modi [1999]. "RAS strategy for IBM S/390 G5 and G6," *IBM J. Research and Development,* 43:5–6 (September–November), 875–888.

Murphy, B., and T. Gent [1995]. "Measuring system and software reliability using an automated data collection process," *Quality and Reliability Engineering International,* 11:5 (September–October), 341–353.

Myer, T. H., and I. E. Sutherland [1968]. "On the design of display processors," *Communications of the ACM,* 11:6 (June), 410–414.

National Storage Industry Consortium [1998]. *Tape Roadmap* (June), *www.nsic.org.*

Nelson, V. P. [1990]. "Fault-tolerant computing: Fundamental concepts," *Computer* 23:7 (July*)*, 19–25.

Nyberg, C. R., T. Barclay, Z. Cvetanovic, J. Gray, and D. Lomet [1994]. "AlphaSort: A RISC machine sort," *Proc. 1994 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD '04),* Minneapolis, Minn., May.

Okada, S., S. Okada, Y. Matsuda, T. Yamada, and A. Kobayashi [1999]. "System on a chip for digital still camera," *IEEE Trans. on Consumer Electronics* 45:3 (August), 584–590.

Patterson, D. A., G. A. Gibson, and R. H. Katz [1987]. "A case for redundant arrays of inexpensive disks (RAID)," Tech. Rep. UCB/CSD 87/391, Univ. of Calif. Also appeared in *ACM SIGMOD Conf. Proc.*, Chicago, June 1–3, 1988, 109–116.

Pavan, P., R. Bez, P. Olivo, and E. Zanoni [1997]. "Flash memory cells—an overview," *Proc. IEEE* 85:8 (August), 1248–1271.

Robinson, B., and L. Blount [1986]. "The VM/HPO 3880-23 performance results," IBM Tech. Bulletin GG66-0247-00 (April), Washington Systems Center, Gaithersburg, Md.

Salem, K., and H. Garcia-Molina [1986]. "Disk striping," *IEEE 1986 Int'l Conf. on Data Engineering,* February 5–7, Washington, D.C., 249–259.

Scranton, R. A., D. A. Thompson, and D. W. Hunter [1983]. "The access time myth," Tech. Rep. RC 10197 (45223) (September 21), IBM, Yorktown Heights, N.Y.

Seagate [2000]. *Seagate Cheetah 73 Family: ST173404LW/LWV/LC/LCV Product Manual,* Volume 1, *www.seagate.com/support/disc/manuals/scsi/29478b.pdf.*

Smotherman, M. [1989]. "A sequencing-based taxonomy of I/O systems and review of historical machines," *Computer Architecture News* 17:5 (September), 5–15. Reprinted in *Computer Architecture Readings,* Morgan Kaufmann, San Francisco, 1999, 451–461.

Talagala, N. [2000]. *"Characterizing large storage systems: Error behavior and performance benchmarks,"* Ph.D. dissertation CSD-99-1066, June 13, 2000.

Talagala, N., R. Arpaci-Dusseau, and D. Patterson [2000]. "Micro-benchmark based extraction of local and global disk characteristics," CSD-99-1063, June 13, 2000.

Talagala, N., S. Asami, D. Patterson, R. Futernick, and D. Hart [2000]. "The art of massive storage: A case study of a Web image archive," *Computer* (November).

Talagala, N., and D. Patterson [1999]. "An analysis of error behavior in a large storage system," Tech. Report UCB//CSD-99-1042, Computer Science Division, University of California at Berkeley (February).

Thadhani, A. J. [1981]. "Interactive user productivity," *IBM Systems J.* 20:4, 407–423.