Preface

Morgan Kaufmann is pleased to present material from a preliminary draft of Readings in Computer Architecture; the material is (c) Copyright 1999 Morgan Kaufmann Publishers. This material may not be used or distributed for any commercial purpose without the express written consent of Morgan Kaufmann Publishers. Please note that this material is a draft of forthcoming publication, and as such neither Morgan Kaufmann nor the authors can be held liable for changes or alterations in the final edition.

In theory, theory and practice are the same.
In practice, they're different.

--Attribution unknown

We are pleased to present you with this collection of readings in computer architecture. Computer architecture can be considered the science and art of selecting and interconnecting hardware components to create a computer that meets functional, performance, and cost goals. Computer architecture has a rich history of practice whose mastery aids those interested in moving the field forward.

Computer architecture continues to flourish in a dynamic environment. One can think of it as an interface with hardware implementation technologies below and application and system software above. From below exponential semiconductor advances continue to provide a greater number of faster transistors, while at the same time altering trade-offs both on and between chips (e.g., on-chip wires can exceed gate delays). Architects use the additional transistors in new ways to make the compound growth of a computer system's capacity exceed the rate at which transistors are getting faster.

From above the applications that use architectures change. Classic applications have evolved, while new application domains emerge as advances make computing cost-effective to new areas. Application areas include scientific computing (e.g., quantum dynamics to vehicle crash simulation), business data processing (e.g., accounting to data mining), personal productivity tools (e.g., word processing and spreadsheets), global connectivity (e.g., electronic mail to telecommuting), and emerging applications (e.g., personal digital assistants to virtual reality).

It is to the students, researchers, and practitioners of this dynamic field of computer architecture that we offer this new collection of readings. The rest of this preface discusses why are we introducing this reader now, how we select and introduce papers, and how one might use this reader.

Why have a computer architecture reader now?

Why a reader now when the last widely successful readers are decades old, there are good textbooks in the field, and the web is expanding as a source for technical information? In our view, the last widely-successful collection of readings in computer architecture was from Bell and Newell in 1971 [2] updated by Siewiorek in 1982 [6]. These readers were excellent, but much has happened since they were published and they are currently out of print.

Another reader than complements this one is the recently-published collection of selected papers from the first 25 years of the *International Symposium on Computer Architecture* (ISCA) edited by Sohi in 1998 [8]. Our reader differs from the ISCA reader in two important ways. First, we have drawn papers from many sources, not just ISCA. Second, we have contributed introductory material that puts multiple papers in context. In contrast, the ISCA reader has an introduction by each of the original author sets that describes how their paper came about, and, sometimes, their view on its future impact. We find these insights valuable and recommend the ISCA reader to you as well.

Why have a reader when there are good textbooks in the field? The answer to this question has deep roots in the history of science. A *primary source* is a work written by the people who did the work (discovery, invention, etc.) when they did it. A *secondary source* is a work written after the primary source and usually by different authors. Textbooks are the vehicle usually used to introduce people to scientific or engineering disciplines. Textbooks are secondary sources designed for teaching. They summarize, synthesize, and interpret work in the intellectual model (*paradigm*) that prevails when they are written. They are indispensable for giving people a reasonably-recent snapshot of the state of the art. Noteworthy textbooks in computer architecture include Almasi and Gottlieb [1], Culler and Singh with Gupta [3], Hennessy and Patterson [4], and Stone [9].

Students and professionals who wish to contribute to advancing a field, however, must eventually move beyond textbooks to primary sources. New primary sources provide the most recent thinking that will not appear in textbooks for one or more years. In computer architecture, the pressure on "bleeding edge" primary sources is so great that conference papers have more impact on intellectual progress than do journal papers (which have a longer delay between submission and appearance).

This reader contains many old primary sources. We see three reasons people should read both old and new primary sources. First, experience reading old primary sources trains people to read new ones. Second, it makes them aware of the process of discovering. Primary sources over time give a "motion picture" rather than a "snap shot," enabling people to see how the movie moves forward, and how they might continue moving the movie forward. Third, it allows people to see ideas in the context of multiple paradigms, instead of just within the current paradigm. Students in the early 1980's, for example, could take whole courses on computer architecture with no mention of out-of-order execution, which was previously invented but out of vogue. We encourage readers interested in how science advances to read the landmark primary source by Thomas Kuhn [5].

Why have a reader when people can just use the web? The primary value of this reader is our editorial judgement for selecting papers (discussed in the following section) and background we provide in introducing them and placing them in context. This value is not diminished by the existence of the World Wide Web.

A secondary value of a reader is the coalescing into one hardcopy the hardcopies of many papers that are time-consuming to obtain. This value remains present for the many pre-web papers we have selected (and is enhanced as post-web researchers are more reluctant to do the leg work to obtain hardcopies the traditional way). This value is significantly diminished, however, for post-web papers.

Our approach is to turn the web into an advantage. This hardcopy reader is supplemented with a web component at URL `http://www.mkp.com/architecture-readings`. This web page is modeled after this hardcopy reader. Its ten sections follow the ten chapters of this reader. Each contain pointers to recent papers and some text to introduce them and put them in context. In particular, the papers represent the best of the state of the art (e.g., a new microprocessor paper). Furthermore, our web component will be updated, at least, annually to make this reader a living document that responds more rapidly than is possible with hardcopy editions. Thus, the web allows us to dynamically extend the value our editorial judgement in a way not practical before the web.

What's in this reader and why?

This reader focuses on what we believe are and will continue to be the critical issues facing computer architects. These include issues in *technology, evaluation methods, instruction set design, instruction level parallelism, dataflow/multithreading, memory systems, input/output systems, single-instruction multiple data parallelism,* and *multiple-instruction multiple data parallelism*. The downside of our focus, however, is that many important areas of inquiry are not covered (e.g., computer aided design and computer arithmetic).

After limiting our focus, we were still left with the challenging task of selecting 50-odd papers to fill a single bound volume from thousands of computer architecture papers that have been published. (For this reason, we hope that you do not judge us too harshly if we left our a few of your favorite papers). We have used several guidelines for selecting papers. We looked for papers that:

- were primary sources,
- had lasting impact,
- are readable,

- are an illustrative case study, and
- contribute insight to the modern reader.

We did not apply these guidelines rigidly, in part, because they can contradict each other (e.g., some primary sources are inscrutable to the modern reader). One consequence of our emphasis on primary sources is that we exclude most survey papers, including influential ones (e.g., Smith's cache survey [7]).

Furthermore, the papers selected (and our original material) underwent two rounds of thoughtful review and discussion with a dozen researchers from academia and industry. They—see the acknowledgements at the end of this preface—helped to refine the selected papers to even better reflect what the community considers important as well as what students should know. We, the editors, resolved conflicts in the advice we got and accept all responsibility for the final version.

We have organized this reader into ten chapters. Each chapter is introduced by some original material that places the papers in context, introducing each, and, in many cases, providing additional content pertinent to the chapter topic but beyond the scope of the included papers.

Chapter 1, *Technology, Implementation, and Economics: Classic Machines*, examines the foundation of computer architecture. This chapter begins a discussion of classic machines from the ENIAC to Cray's machines with an emphasis on how implementation technologies influenced architecture. It then discusses Gordon Moore's amazing technology predictions from the 1960s (e.g., Moore's Law) and how they lead to microprocessor-based computers. Included papers discuss IBM 360, CDC 6600, Cray 1, Moore's predictions, and early Intel microprocessors.

Chapter 2, *Methods*, is not about computer architecture, per se, but about some of the methods used to evaluate and refine architectures. Without these methods, the progress we have come to expect would not be possible. This chapter discusses the scientific method and the three major classes of methods that computer architects use: analytic modeling, simulation, and system monitoring. Techniques are illustrated with case studies from memory hierarchy evaluation. Included papers present Amdahl's Law, a system monitoring study of the VAX-11/780, and trace-driven simulation algorithms of evaluating alternative caches.

Chapter 3, *Instruction Sets*, discusses the interface between software and hardware that is so critical in that it enables software to run on multiple generations of hardware. This chapter examines how changes in implementation technology and compiler technology have changed instruction set trade-offs, including putting the Complex Instruction Set Computer (CISC) versus Reduced Instruction Set Computer (RISC) debate of the 1980s into perspective. Included papers discuss instruction sets as compiler targets, RISC, CISC, the most widely-used instruction set (Intel 80386), and ideas on predication pertinent to emerging instruction sets such as IA-64.

Chapter 4, *Instruction Level Parallelism (ILP)*, examines issues critical to accelerating processor execution rates beyond the increases provided by faster transistors and bit-level parallelism. This chapter examines issues for executing multiple instructions in parallel that include hazards, precise exceptions, speculative execution, branch prediction, and explicitly parallel architectures such as Very Long Instruction Word (VLIW). Included papers discuss the IBM 360/91, IBM RS/6000, MIPS R10000, branch prediction, precise exceptions, out-of-order execution, and a survey of instruction level parallelism issues.

Chapter 5, *Dataflow and Multithreading*, discusses these two approaches to increasing parallelism and tolerating latency. Dataflow has had considerable intellectual impact, while multithreading appears poised to significantly impact practice. This chapter explains how classic data flow pushes limits by eliminating the program counter, while multithreading more conventionally (and more practically) uses multiple program counters. Included papers discuss foundational dataflow work, the tagged-token dataflow approach, an early multithreaded computer (HEP), and recent simultaneous multithreading ideas.

Chapter 6, *Memory Systems*, examines caches and virtual memory, two critical aspects of the memory hierarchy that can largely determine sustained computer performance. Since textbooks cover both concepts in detail, both discussions begin with early concepts and then examine selected more-recent concepts that are illustrated in the papers. Included cache papers discuss the first data cache proposals (Wilkes), the first commercial cache (IBM 380/85), non-blocking caches, snooping cache coherence, and victim caches/stream buffers. Included virtual memory papers present the first virtual memory system (Atlas), a 1980s virtual memory system from VAX-11/780, and a case study of some of the interactions between caches and virtual memory.

Chapter 7, I/O: *Storage Systems, Networks, and Graphics*, examines systems needed to make computers interact with the outside world. In particular, this chapter discusses Input/Output (I/O) economics, presents a case study of Personal Computer (PC) I/O systems options, and introduces included papers. Included papers discuss historical I/O systems, disk modeling, Redundant Arrays of Inexpensive Disks (RAID), Ethernet local area network, routing in interconnection networks, and a case study of graphics support.

Chapter 8, *Single-Instruction Multiple Data (SIMD) Parallelism*, discusses an approach to parallel computing where a single thread of control directs the manipulation of multiple data operations. SIMD's perceived utility has waxed and waned several times over the past decades. Even though interest in SIMD has currently waned readers should be familiar with SIMD because it is likely to wax again in the future. This chapter introduces basic SIMD concepts and included papers. Included papers discuss the original SIMD/MIMD taxonomy, a SIMD machine that issued dependent operations (BSP), and massive processing in memory.

Chapter 9, *Multiprocessors and Multicomputers,* discusses computing systems in which multiple processors can operate independently. This is called *multiple-instruction multiple data (MIMD) parallelism*. Such systems have long been the "future" of computing, but now they have finally earned substantial commercial success. This chapter discusses parallel software, shared memory multiprocessors (processors joined via the memory system), multicomputers (processors joined via the I/O system), and selected future trends. Included shared memory multiprocessor papers discuss an early prototype machine (CMU C.mmp), memory consistency models, cache coherence, a recent scalable example (Stanford DASH), and a cache-only memory architecture (COMA). Multicomputer papers discuss an early multicomputer (Caltech Cosmic Cube) and shared memory on multicomputers.

Finally, Chapter 10, *Recent Implementations and Future Prospects*, revisits for modern machines some of the issues Chapter 1 raised for classic machines. Included papers discuss Intel Pentium, Intel Pentium Pro, and future microprocessor trends.

How to use this book

We have prepared this book with the hope that it will be valuable to both professionals and students. Both groups can read these primary sources and our original commentary to learn more about the rich history of computer architecture and to better see how to move the field forward.

For instructors, we see three beneficial ways to use our reader. First, it can be used as a supplement to a primary textbook. At Wisconsin, for example, we have a primary graduate architecture course that focuses on uniprocessors. Most recent instructors have used Hennessy and Patterson [4] and a custom reader of papers. We anticipate some instructors will use this reader and some very recent papers from the web (at this reader's web site and elsewhere) to replace their current custom reader. Wisconsin also has a secondary graduate architecture course that focuses on parallel processing. It has recently used Culler and Singh with Gupta [3] and a custom reader. As above, one could replace the custom reader with this reader and supplementary web papers.

A second approach fits schools that have a first course that uses a textbook and a second course whose orientation toward projects or case studies favors a custom reader. Once again this reader and web papers could replace the custom reader.

A third model is a graduate course or courses that eschew textbooks and just use readers (as occurs for some offerings of Wisconsin's courses). This approach requires considerable skill and effort from faculty to tie disjoint ideas together. Using this reader and selected other papers, possibly web only, will ease the instructor's burden since we make considerable effort to tie our included papers together.

Acknowledgments

We wish to thank the many people that have contributed ideas, comments and criticism to this reader, especially the anonymous referees recruited by Morgan Kaufmann, *Arvind, Matt Farrens, Jim Goodman, Rishiyur Nikhil, Daniel Sorin, Jim Smith,* and *David Wood*. Of course, all responsibility for what is said or included in this reader falls on us.

We also want thank the wonderful staff at Morgan Kaufmann for their talents, efforts, and encouragement, especially *Denise Penrose, Meghan Keeffe,* and *Marilyn Alan*.

References

[1] G. S. Almasi and A. Gottlieb. *Highly Parallel Computing*. Benjamin / Cummings Publishing Company, Inc., 1994.

[2] C. G. Bell and A. Newell. *Computer Structures: Readings and Examples*. McGraw Hill, 1971.

[3] David Culler, J. P. Singh, and Anoop Gupta. *Parallel Computer Architecture: A Hardware/Software Approach*. Morgan Kaufmann, 1998.

[4] John L. Hennessy and David A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, second edition, 1996.

[5] T. S. Kuhn. *The Structure of Scientific Revolutions*. Univ. of Chicago Press, second edition, 1970.

[6] D. P. Siewiorek, C. G. Bell, and A. Newell. *Computer Structures: Principles and Examples*. McGraw Hill, 1982.

[7] Alan J. Smith. Cache Memories. *ACM Computing Surveys*, 14(3):473–530, 1982.

[8] Gurindar Sohi. *25 Years of the International Symposia on Computer Architecture: Selected Pa-*

pers. ACM Press, 1998.

[9] Harold S. Stone. *High-Performance Computer Architecture*. Addison Wesley, third edition, 1993.