Vorwort zur englischsprachigen Originalausgabe

Die schönste Sache, die wir erleben können, ist das Geheimnisvolle. Es ist die Quelle aller wahren Kunst und Wissenschaft.
Albert Einstein, *What I Believe*, 1930

Zu diesem Buch

Wir sind der Meinung, dass das Studium der Informatik und der Ingenieurwissenschaften den aktuellen Stand des Forschungsgebietes widerspiegeln sollte, und dazu gehört auch die Einführung in die Prinzipien des modernen Computings. Wir meinen außerdem, dass die Leser die Organisationsprinzipien der Hardware und der Software kennen sollten, die über Kapazität, Leistung und letztendlich den Erfolg von Computersystemen entscheiden.

In der modernen Computertechnologie benötigt man in jedem Teilgebiet der Informatik professionelles Wissen, um sowohl Hardware als auch Software verstehen zu können. Das Zusammenspiel zwischen Hardware und Software auf den unterschiedlichsten Ebenen bildet das Grundgerüst, auf dem die Informatik aufgebaut ist. Egal, ob Sie hauptsächlich an Hardware oder an Software, am Computing oder an Elektrotechnik interessiert sind, die zentralen Konzepte beim Aufbau und Entwurf von Computern sind überall dieselben. Wir versuchen deshalb in diesem Buch, die Beziehung zwischen Hardware und Software aufzuzeigen und uns auf die Konzepte zu konzentrieren, die die Grundlage für moderne Computer bilden.

Der Wechsel von Einzelprozessoren zu Multicore-Mikroprozessoren und das aktuelle Interesse an domänenspezifischen Architekturen bestätigen die Stichhaltigkeit dieser Aussage, an der sich seit der ersten Auflage nichts geändert hat. Die Programmierer sind möglicherweise versucht, diesen Hinweis zu ignorieren und sich darauf zu verlassen, dass ihre Programme dank der Entwicklungen in der Computerarchitektur, beim Compilerbau und in der Siliziumtechnologie immer schneller werden, ohne dass sie etwas dafür tun müssen. Doch diese Zeiten sind vorbei. Damit Programme schneller laufen, müssen sie parallel arbeiten. Viele Forscher verfolgen immer noch das Ziel, dass die Programmierer die zugrundeliegende parallele Natur der von ihnen pro-

https://doi.org/10.1515/9783110756180-202

grammierten Hardware nicht kennen müssen, aber es wird noch viele Jahre dauern, bis diese Vision Wirklichkeit werden kann. Unserer Meinung nach müssen zumindest in den nächsten 10 Jahren die meisten Programmierer die Hardware/Software-Schnittstelle verstehen, wenn sie wollen, dass ihre Programme effizient auf parallelen Computern laufen.

Dieses Buch richtet sich sowohl an Leser, die den grundlegenden Aufbau von Computern kennenlernen wollen und wenig Erfahrungen mit Assemblersprachen oder Logikdesign besitzen, als auch an Leser, die lernen wollen, wie man einen Computer entwirft, oder verstehen wollen, wie ein System funktioniert und warum es sich so verhält, wie es sich verhält.

Das andere Buch

Einige Leser kennen möglicherweise schon das Buch *Computer Architecture: A Quantitative Approach*, auch als „Hennessy und Patterson“ bezeichnet. (Das vorliegende Buch dagegen heißt „Patterson und Hennessy“.) Unsere Motivation für das ältere Buch war, die Prinzipien der Rechnerarchitektur zu beschreiben und dazu fundierte Grundlagen aus den Ingenieurwissenschaften zu verwenden und quantitative Kosten-Leistungs-Abwägungen zu zeigen. Wir haben einen Ansatz verfolgt, der Beispiele und Messungen von kommerziellen Systemen verwendet, um realistische Entwurfserfahrungen aufzuzeigen. Unser Ziel war es zu zeigen, dass die Rechnerarchitektur besser unter Verwendung quantitativer Methodologie anstatt nur über einen deskriptiven Ansatz vermittelt wird. Diese Herangehensweise richtet sich an den ernsthaften IT-Profi, der sich detailliertes Wissen über Computern aneignen will.

Ein Großteil der Leser dieses Buches hat nicht vor, Rechnerarchitekt zu werden. Die Performanz und die Energieeffizienz zukünftiger Softwaresysteme wird jedoch ganz wesentlich davon beeinflusst, wie gut Softwaredesigner die grundlegende, in einem System eingesetzte Hardware verstehen. Compilerentwickler, Betriebssystemdesigner, Datenbankprogrammierer und die meisten anderen Software-Ingenieure benötigen fundierte Kenntnisse der in diesem Buch vorgestellten Konzepte. Analog müssen Hardwaredesigner in der Lage sein, die Auswirkung ihrer Arbeit auf Softwareanwendungen genau zu verstehen.

Wir wussten, dass dieses Buch sehr viel mehr als eine Teilmenge der Informationen aus *Computer Architecture* sein musste, und es wurde umfassend überarbeitet, um ein weiter gefasstes Publikum anzusprechen. Wir waren mit dem Ergebnis so zufrieden, dass die nachfolgenden Auflagen von *Computer Architecture* überarbeitet wurden, um einen Großteil der einführenden Informationen zu entfernen. Damit gibt es heute viel weniger Überlappungen als bei den ersten beiden Auflagen beider Bücher.

Änderungen für die 6. Auflage

Seit der 5. Auflage hat es vermutlich mehr technologische und kommerzielle Veränderungen im Bereich der Computerarchitektur gegeben als während der ersten fünf Auflagen:

- *Verlangsamung des Moore'schen Gesetzes.* Nachdem 50 Jahre lang eine Verdopplung der Anzahl der Transistoren pro Chip zu beobachten war, ist Gordon Moores Vorhersage mittlerweile nicht mehr gültig. Es wird auch weiterhin Verbesserungen in der Halbleitertechnologie geben, doch sie werden langsamer vonstatten gehen und nicht mehr so gut vorhersagbar sein.
- *Domänenspezifische Architekturen (DSA).* Die Verlangsamung des Moore'schen Gesetzes und das Ende der Dennard-Skalierung haben dazu geführt, dass sich Allzweckprozessoren heute nur noch um wenige Prozent pro Jahr verbessern. Zudem limitiert das Amdahl'sche Gesetz den praktischen Nutzen, der sich aus der wachsenden Zahl der Prozessoren pro Chip ziehen lässt. Im Jahr 2020 wird allgemein angenommen, dass die Entwicklung von domänenspezifischen Architekturen für die Zukunft der erfolgversprechendste Weg ist. Bei diesem Ansatz wird im Unterschied zu Allzweckprozessoren nicht angestrebt, dass auf der Architektur *alles* gut läuft. Vielmehr fokussiert sich jede solche Architektur auf einen bestimmten Anwendungsbereich, in dem sie viel besser als konventionelle Architekturen ist.
- *Mikroarchitektur als Angriffsfläche.* Das Angriffsszenario Spectre zeigte auf, dass spekulative Out-of-Order-Ausführung und Hardware-Multithreading zeitbasierte Seitenkanalattacken möglich machen. Dabei handelt es sich nicht um Bugs, die sich beheben lassen, sondern um eine fundamentale Herausforderung für diese Art des Prozessordesigns.
- *Offene Befehlssätze und Open-Source-Implementierungen.* Die Möglichkeiten und der Auswirkungen von Open-Source-Software schlagen sich mittlerweile auch in der Computerarchitektur nieder. Offene Standards für Befehlssätze wie RISC-V versetzen Organisationen in die Lage, ihre eigenen Prozessoren zu entwickeln, ohne zuerst Lizenzverhandlungen zu führen. Dadurch sind sowohl Open-Source-Implementierungen möglich geworden, die frei heruntergeladen und genutzt werden können, als auch proprietäre Implementierungen von RISC-V. Open-Source-Software und -Hardware sind ein Segen für die akademische Forschung und die Lehre, da sie es den Studenten erlauben, leistungsstarke Technolgie direkt zu betrachten und zu verbessern.
- *Re-Vertikalisierung der IT-Industrie.* Das Cloud-Computing wird von einer kleinen Zahl von Unternehmen – nicht mehr als ein halbes Dutzend – dominiert, die Computer-Infrastruktur und Rechenleistung als Dienstleistung anbieten. Ähnlich wie IBM in den 1960er- und 1970er-Jahren legen diese Unternehmen sowohl die Auswahl an Software als auch die verwendete Hardware fest. Die zuvor angesprochenen Veränderungen haben dazu geführt, dass einige dieser „Hyperscaler" ihre eigenen DSA- und RISC-V-Chips entwickelt haben, um sie in ihren Clouds einzusetzen.

Die 6. Auflage spiegelt diese neuen Entwicklungen wider. Außerdem wurden viele Beispiele und Abbildungen aktualisiert, kleine Änderungen in Reaktion auf Anfragen von Lehrenden vorgenommen und eine didaktische Verbesserung eingeführt, zu der mich die Lehrbücher inspiriert haben, die ich verwende, um meinen Enkeln bei ihren Matheaufgaben zu helfen.

- Abschnitte zur Beschleunigung gibt es nun in jedem Kapitel. Den Anfang bildet eine Python-Version der Matrixmultiplikation, die in Kapitel 1 vorgestellt wird und deren schlechte Performanz Grund genug ist, C zu lernen. Die in C umgeschriebene Matrixmultiplikation wird dann in Kapitel 2 betrachtet. In den verbleibenden Kapiteln wird die Matrixmultiplikation beschleunigt, indem die Parallelität auf Datenebene, die Parallelität auf Befehlsebene und die Parallelität auf Thread-Ebene ausgenutzt und der Speicherzugriff an die Speicherhierarchie eines modernen Servers angepasst wird. Dieser Computer hat 521-Bit SIMD-Operationen, spekulative Out-of-Order-Ausführung, drei Cache-Levels und 48 Cores. Insgesamt umfassen diese vier Optimierungen nur 21 Zeilen C-Code, doch sie beschleunigen die Matrixmultiplikation fast auf das 50 000-Fache. Aus beinahe sechs Stunden in der Python-Version kommen wir so auf weniger als eine Sekunde für die optimierte C-Version. Wäre ich noch mal Student, dann würde mich dieses Beispiel dazu bewegen, C zu verwenden und mir die in diesem Buch besprochenen Hardwarekonzepte anzueignen, die den Optimierungen zugrunde liegen.
- In dieser Auflage enthält jedes Kapitel einen Abschnitt mit Fragestellungen, die das Nachdenken anregen sollen. Im Anschluss an die Fragestellungen werden Antworten vorgestellt, so dass Sie Ihre eigenen Antworten evaluieren können.
- Zusätzlich zu der Erklärung, dass das Moore'sche Gesetz und die Dennard-Skalierung nicht mehr gelten, haben wir unsere Hervorhebung des Moore'schen Gesetzes als treibende Kraft zurückgenommen, die in der 5. Auflage sehr betont wurde.
- In Kapitel 2 wird ausführlicher erläutert, dass binäre Daten keine inhärente Bedeutung haben, sondern dass es das Programm ist, das den Datentyp bestimmt. Erfahrungsgemäß ist dieses Konzept für Anfänger nicht ganz einfach zu verstehen.
- Kapitel 2 enthält außerdem eine kurze Beschreibung von RISC-V als einem Befehlssatz neben und in Konkurrenz zu ARMv7, ARMv8 und x86. (Es gibt auch eine Begleitversion zu diesem Buch, die auf RISC-V anstatt auf MIPS basiert und die wir parallel aktualisieren.)
- Das Benchmark-Beispiel in Kapitel 2 wurde auf SPEC2017 aktualisiert.
- Auf Anfrage von Lehrenden haben wir die Mehrtaktimplementierung von MIPS in Kapitel 4 wieder aufgenommen, und zwar in Form eines Online-Abschnitts zwischen der Eintaktimplementierung und der Pipeline-Implementierung. Einige Lehrende finden den Zugang über diese drei Schritte didaktisch hilfreich, um die Idee des Pipelinings zu vermitteln.

- Die Fallbeispiele in den Kapiteln 4 und 5 wurden auf die modernen Mikroarchitekturen des ARM A53 und des Intel i7 6700 Skylake aktualisiert.
- In den Abschnitten zu Fallstricken und Trugschlüssen in den Kapiteln 5 und 6 wurden neuere Sicherheitsprobleme aufgenommen, die unter den Namen *Rowhammer* und *Spectre* bekannt geworden sind.
- Kapitel 6 enthält einen neuen Abschnitt, in dem Googles Tensorprozessor TPUv1 als Beispiel einer domänenspezifischen Architektur eingeführt wird. Das Fallbeispiel in diesem Kapitel wurde dahingehend aktualisiert, dass es Googles TPUv3-Supercomputer mit einem Cluster aus NVIDIA Volta GPUs vergleicht.
- Schließlich haben haben wir in allen Kapiteln die Aufgaben aktualisiert.

Obwohl sich einige Elemente geändert haben, ist doch ein großer, weiterhin nützlicher Teil erhalten geblieben. Damit das Buch gut als Referenz verwendet werden kann, stehen in der Randspalte auch weiterhin Kurzdefinitionen neu eingeführter Begriffe. Nach wie vor gibt es eine Reihe von wiederkehrenden Elementen. Das Element „Zur Programmperformanz“ soll den Lesern dabei helfen, die Performanz ihrer eigenen Programme zu verstehen und zu verbessern. Das Element „Hardware-Software-Schnittstelle“ ist den Beziehungen an dieser Schnittstelle gewidmet. In grau hinterlegten Boxen ist „Das Wesentliche“ kurz zusammengefasst, damit es nicht passiert, dass der Leser den Wald vor lauter Bäumen nicht mehr sieht. Am Ende der meisten Abschnitte gibt es einen „Selbsttest“, der dem Leser Gelegenheit gibt, sein Wissen über den behandelten Stoff zu überprüfen. Zur Kontrolle stehen die Antworten jeweils am Ende des Kapitels. Auch in dieser Auflage finden Sie die MIPS-Zusammenfassung, die durch die „Green Card“ des IBM System/360 inspiriert wurde. Die Karte soll eine praktische Referenz darstellen, wenn Sie Programme in MIPS-Assemblersprache schreiben.

Unterstützung für Lehrende

Wir haben eine große Menge an Material gesammelt, um Lehrende zu unterstützen, die dieses Buch in ihren Kursen einsetzen. Lösungen zu den Aufgaben, Abbildungen aus dem Buch, Vorlesungsfolien und anderes Material kann zu diesem Zweck vom Verlag der US-Ausgabe übernommen werden. Hier finden Sie weitere Informationen:

`www.degruyter.com`

Abschließende Bemerkungen

Wenn Sie den nachfolgenden Abschnitt mit den Danksagungen lesen, werden Sie feststellen, dass wir uns größte Mühe gegeben haben, Fehler zu korrigieren. Weil dieses Buch häufig nachgedruckt wird, haben wir die Gelegenheit, immer mehr Korrekturen auszuführen. Falls Sie weitere, bisher nicht entdeckte Fehler finden, wenden Sie sich bitte an den Verlag.

Kapitel oder Anhang	Abschnitte	Fokus auf Software	Fokus auf Hardware
1. Abstraktionen und Technologien	1.1 bis 1.12	*****	*****
	1.13 (Geschichte)	**	**
2. Befehle: Die Sprache der Computer	2.1 bis 2.14	*****	****
	2.15 (Compiler und Java)	***	
	2.16 bis 2.22	*****	****
	2.23 (Geschichte)	**	**
E. RISC-Architektur	E.1 bis E.6	***	
3. Rechnerarithmetik	3.1 bis 3.5	****	****
	3.6 bis 3.8 (Subwort-Parallelität)	*****	*****
	3.9 bis 3.10 (Fallstricke)	****	****
	3.11 (Geschichte)	**	**
B. Basics of Logic Design	B.1 bis B.13		****
4. Der Prozessor	4.1 (Überblick)	*****	*****
	4.2 (Logik-Konventionen)		*****
	4.3., 4.4 (Einfache Implementierung)	****	*****
	4.5 (Mehrtaktimplementierung)		***
	4.6 (Pipelining Überblick)		*****
	4.7 (Pipeline Datenpfad)	****	*****
	4.8 bis 4.10 (Konflikte, Ausnahmen)		*****
	4.11 bis 4.13 (Parallität, Fallstudie)	*****	*****
	4.14 (Verilog Pipeline Control)		***
	4.15 bis 4.16 (Fallstricke)	*****	*****
	4.17 (Geschichte)	**	**
D. Mapping Control to Hardware	D.1 bis D.6	*****	*****
5. Schnell und groß: Ausnutzung der Speicherhierarchie	5.1 bis 5.10	**	**
	5.11(Parallism & Memory Hierarchy)	***	***
	5.12 (Verilog Cache Controller)		***
	5.13 bis 5.16	*****	*****
	5.17 (Geschichte)	**	**
6. Parallele Prozessoren: vom Client zur Cloud	6.1 bis 6.9	*****	*****
	6.10 (Cluster)	***	***
	6.11 bis 6.15	*****	*****
	6.16 (Geschichte)	**	**
A. Assemblers, Linkers, and the SPIM Simulator	A.1 bis A.11	***	***
C. Graphics Processor Units	C.1 bis C.11	*	*

Hinweise: Mit dem Symbol gekennzeichnete Themen sind im Buch nicht enthalten, sondern als Online-Ressourcen verfügbar. Die Anhänge A und B wurden ohne Übersetzung aus der amerikanischen Originalfassung übernommen. Die verwendeten Symbole haben folgende Bedeutung: ***** Abschnitte, die intensiv studiert werden sollten; **** Abschnitte, die zumindest quergelesen werden sollten; *** kann zunächst ausgelassen und bei Gelegenheit nachgeholt werden; ** historisch interessant; * Referenz

Diese Auflage markiert die dritte Unterbrechung der langjährigen Zusammenarbeit zwischen Hennessy und Patterson, die 1989 begann. Als Leiter einer der bedeutendsten Universitäten der Welt war es Präsident Hennessy nicht mehr möglich, substanziell zu einer Neuauflage beizutragen. Der verbliebene Autor fühlte sich einmal mehr wie ein Drahtseilkünstler ohne Sicherheitsnetz. Aus diesem Grund haben die in der Danksagung erwähnten Personen und die Kollegen in Berkeley diesmal eine noch größere Rolle bei der Gestaltung der Inhalte in diesem Buch gespielt. Nichtsdestotrotz hat diesmal nur ein Autor allein den neuen Stoff zu verantworten, den Sie in diesem Buch lesen werden.

Danksagungen für die 6. Auflage

Bei jeder Auflage dieses Buches haben wir das große Glück, Unterstützung von vielen Lesern, Rezensenten und Mitarbeitern zu bekommen. Jeder dieser Menschen hat dazu beigetragen, das Buch besser zu machen.

Ein besonderes Dankeschön geht an Dr. Rimas Avizenis, der verschiedene Versionen der Matrixmultiplikation entwickelt hat und von dem ich außerdem Leistungskennzahlen erhalten habe. Da ich als graduierter Student an der University of California, Los Angeles, mit seinem Vater zusammengearbeitet habe, hat sich durch die Zusammenarbeit mit Rimas eine schöne Symmetrie ergeben.

Ich möchte auch meinem langjährigen Mitarbeiter **Randy Katz** an der University Berkeley danken, der mir dabei geholfen hat, das System der acht wichtigen Konzepte der Computerarchitektur zu entwickeln. Diese Arbeit war Teil einer intensiven Überarbeitung eines Grundkurses, den wir gemeinsam gehalten haben.

Ich danke **David Kirk, John Nickolls** und ihren Kollegen bei NVIDIA (Michael Garland, John Montrym, Doug Voorhies, Lars Nyland, Erik Lindholm, Paulius Micikevicius, Massimiliano Fatica, Stuart Oberman und Vasily Volkov) für den ersten detaillierten Anhang über GPUs. Und ich möchte **Jim Larus,** inzwischen Dekan der School of Computer and Communications Science an der EPFL, für seine Bereitschaft danken, seine Erfahrung in der Assemblerprogrammierung beizutragen, und den Lesern dieses Buches den von ihm entwickelten und unterhaltenen Simulator bereitzustellen.

Ebenso dankbar bin ich **Jason Bakos** von der University of South Carolina, der das Update der Übungsaufgaben übernommen hat und darüber hinaus neue Aufgaben für die vorliegende Auflage beigesteuert hat. Ausgangsbasis waren die Aufgaben der 4. Auflage, aufbereitet von **Perry Alexander** (The University of Kansas), **Javier Bruguera** (Universidade de Santiago de Compostela), **Matthew Farrens** (University of California, Davis), **David Kaeli** (Northeastern University), **Nicole Kaiyan** (University of Adelaide), **John Oliver** (Cal Poly, San Luis Obispo), **Milos Prvulovic** (Georgia Tech) sowie **Jichuan Chang, Jacob Leverich, Kevin Lim** und **Partha Ranganathan** (alle von Hewlett-Packard). Außerdem danke ich **Peter J. Ashden** (Ashden Design Pty Ltd) für seine Mitwirkung an vorherigen Auflagen.

Ein zusätzliches Dankeschön geht an **Jason Bakos** für die Ausarbeitung der neuen Vorlesungsfolien.

Sehr dankbar bin ich den vielen Dozenten, die Anfragen des Verlags beantwortet, unsere Vorschläge gesichtet und an Testgruppen teilgenommen haben, um unsere Pläne für die vorliegende und vorherige Auflagen zu analysieren. Zu ihnen gehören die folgenden Personen: Testgruppe in 2012: Bruce Barton (Suffolk County Community College), Jeff Braun (Montana Tech), Ed Gehringer (North Carolina State), Michael Goldweber (Xavier University), Ed Harcourt (St. Lawrence University), Mark Hill (University of Wisconsin, Madison), Patrick Homer (University of Arizona), Norm Jouppi (HP Labs), Dave Kaeli (Northeastern University), Christos Kozyrakis (Stanford University), Zachary Kurmas (Grand Valley State University), Jae C. Oh (Syracuse University), Lu Peng (LSU), Milos Prvulovic (Georgia Tech), Partha Ranganathan (HP Labs), David Wood (University of Wisconsin), Craig Zilles (University of Illinois at Urbana-Champaign). Surveys and Reviews: Mahmoud Abou-Nasr (Wayne State University), Perry Alexander (The University of Kansas), Hakan Aydin (George Mason University), Hussein Badr (State University of New York at Stony Brook), Mac Baker (Virginia Military Institute), Ron Barnes (George Mason University), Douglas Blough (Georgia Institute of Technology), Kevin Bolding (Seattle Pacific University), Miodrag Bolic (University of Ottawa), John Bonomo (Westminster College), Jeff Braun (Montana Tech), Tom Briggs (Shippensburg University), Scott Burgess (Humboldt State University), Fazli Can (Bilkent University), Warren R. Carithers (Rochester Institute of Technology), Bruce Carlton (Mesa Community College), Nicholas Carter (University of Illinois at Urbana-Champaign), Anthony Cocchi (The City University of New York), Don Cooley (Utah State University), Robert D. Cupper (Allegheny College), Edward W. Davis (North Carolina State University), Nathaniel J. Davis (Air Force Institute of Technology), Molisa Derk (Oklahoma City University), Nathan B. Dodge (The University of Texas at Dallas), Derek Eager (University of Saskatchewan), Ernest Ferguson (Northwest Missouri State University), Rhonda Kay Gaede (The University of Alabama), Etienne M. Gagnon (UQAM), Costa Gerousis (Christopher Newport University), Paul Gillard (Memorial University of Newfoundland), Michael Goldweber (Xavier University), Georgia Grant (College of San Mateo), Merrill Hall (The Master's College), Tyson Hall (Southern Adventist University), Ed Harcourt (St. Lawrence University), Justin E. Harlow (University of South Florida), Paul F. Hemler (Hampden-Sydney College), Martin Herbordt (Boston University), Steve J. Hodges (Cabrillo College), Kenneth Hopkinson (Cornell University), Dalton Hunkins (St. Bonaventure University), Baback Izadi (State University of New York – New Paltz), Reza Jafari, Robert W. Johnson (Colorado Technical University), Bharat Joshi (University of North Carolina, Charlotte), Nagarajan Kandasamy (Drexel University), Rajiv Kapadia, Ryan Kastner (University of California, Santa Barbara), E.J. Kim (Texas A&M University), Jihong Kim (Seoul National University), Jim Kirk (Union University), Geoffrey S. Knauth (Lycoming College), Manish M. Kochhal (Wayne State), Suzan Koknar-Tezel

(Saint Joseph's University), Angkul Kongmunvattana (Columbus State University), April Kontostathis (Ursinus College), Christos Kozyrakis (Stanford University), Danny Krizanc (Wesleyan University), Ashok Kumar, S. Kumar (The University of Texas), Zachary Kurmas (Grand Valley State University), Robert N. Lea (University of Houston), Baoxin Li (Arizona State University), Li Liao (University of Delaware), Gary Livingston (University of Massachusetts), Michael Lyle, Douglas W. Lynn (Oregon Institute of Technology), Yashwant K Malaiya (Colorado State University), Bill Mark (University of Texas at Austin), Ananda Mondal (Claflin University), Euripides Montagne (University of Central Florida), Tali Moreshet (Boston University), Alvin Moser (Seattle University), Walid Najjar (University of California, Riverside), Danial J. Neebel (Loras College), John Nestor (Lafayette College), Jae C. Oh (Syracuse University), Joe Oldham (Centre College), Timour Paltashev, James Parkerson (University of Arkansas), Shaunak Pawagi (SUNY at Stony Brook), Steve Pearce, Ted Pedersen (University of Minnesota), Lu Peng (Louisiana State University), Gregory D Peterson (The University of Tennessee), Milos Prvulovic (Georgia Tech), Partha Ranganathan (HP Labs), Dejan Raskovic (University of Alaska, Fairbanks) Brad Richards (University of Puget Sound), Roman Rozanov, Louis Rubinfield (Villanova University), Md Abdus Salam (Southern University), Augustine Samba (Kent State University), Robert Schaefer (Daniel Webster College), Carolyn J. C. Schauble (Colorado State University), Keith Schubert (CSU San Bernardino), William L. Schultz, Kelly Shaw (University of Richmond), Shahram Shirani (McMaster University), Scott Sigman (Drury University), Bruce Smith, David Smith, Jeff W. Smith (University of Georgia, Athens), Mark Smotherman (Clemson University), Philip Snyder (Johns Hopkins University), Alex Sprintson (Texas A&M), Timothy D. Stanley (Brigham Young University), Dean Stevens (Morningside College), Nozar Tabrizi (Kettering University), Yuval Tamir (UCLA), Alexander Taubin (Boston University), Will Thacker (Winthrop University), Mithuna Thottethodi (Purdue University), Manghui Tu (Southern Utah University), Dean Tullsen (UC San Diego), Rama Viswanathan (Beloit College), Ken Vollmar (Missouri State University), Guoping Wang (Indiana-Purdue University), Patricia Wenner (Bucknell University), Kent Wilken (University of California, Davis), David Wolfe (Gustavus Adolphus College), David Wood (University of Wisconsin, Madison), Ki Hwan Yum (University of Texas, San Antonio), Mohamed Zahran (City College of New York), Amr Zakr (Santa Clara University), Gerald D. Zarnett (Ryerson University), Nian Zhang (South Dakota School of Mines & Technology), Jiling Zhong (Troy University), Huiyang Zhou (The University of Central Florida), Weiyu Zhu (Illinois Wesleyan University).

Ein besonderes Dankeschön geht außerdem an Mark Smotherman, der das Manuskript mehrmals gelesen hat, um typografische Fehler und Tippfehler zu finden. Durch seinen Einsatz konnte die Qualität der vorliegenden Auflage signifikant verbessert werden.

Wir danken der großen Familie bei Morgan Kaufmann für die Bereitschaft, dieses Buch wieder unter der fähigen Leitung von **Steve Merken** und **Beth**

LoGiudice zu veröffentlichen – ohne sie hätte ich es nicht geschafft, das Buch fertigzustellen. Unser Dank geht auch an **Beula Christopher,** die den Herstellungsprozess koordiniert hat, sowie an **Patrick Ferguson** für die Gestaltung des Covers. Sein Cover verbindet auf geschickte Weise die Inhalte der PostPC-Ära, die für diese Auflage maßgeblich sind, mit dem Cover der ersten Auflage.

Die Mitwirkung der fast 150 Menschen, die wir hier erwähnt haben, hat dazu beigetragen, diese 6. Auflage zu unserem hoffentlich bisher besten Buch zu machen. Viel Spaß beim Lesen!

David A. Patterson