Computer Architecture for Scientists

The dramatic increase in computer performance has been extraordinary, but not for all computations: it has key limits and structure. Software architects, developers, and even data scientists need to understand how to exploit the fundamental structure of computer performance to harness it for future applications.

Using a principled approach, *Computer Architecture for Scientists* covers the four key pillars of computer performance and imparts a high-level basis for reasoning with and understanding these concepts. These principles and models provide approachable high-level insights and quantitative modeling without distracting low-level detail. The pillars include:

- Small is fast: how size scaling drives performance.
- Hidden parallelism: how a sequential program can be executed faster with parallelism.
- Dynamic locality: skirting physical limits, by arranging data in a smaller space.
- Explicit parallelism: increasing performance with teams of workers.

Finally, the text covers the GPU and machine-learning accelerators that have become important for more and more mainstream applications. Ideal for upper-level undergraduates.

Andrew A. Chien is the William Eckhardt Professor at the University of Chicago, Director of the CERES Center for Unstoppable Computing, and a Senior Scientist at Argonne National Laboratory. Since 2017, he has served as Editor-in-Chief of the Communications of the ACM. Chien is a global research leader in parallel computing, computer architecture, clusters, and cloud computing, and has received numerous awards for his research. In 1994 he was named an National Science Foundation Young Investigator. Dr. Chien served as Vice President of Research at Intel Corporation from 2005 to 2010, and on advisory boards for the National Science Foundation, Department of Energy, Japan RWCP, and distinguished universities such as Stanford, UC Berkeley, EPFL, and the University of Washington. From 1998 to 2005, he was the SAIC Chair Professor at UCSD, and prior to that a professor at the University of Illinois. Dr. Chien is a Fellow of the ACM, Fellow of the IEEE, and Fellow of the AAAS, and earned his PhD, MS, and BS from the Massachusetts Institute of Technology.

Cambridge University Press 978-1-316-51853-3 — Computer Architecture for Scientists Andrew A. Chien Frontmatter <u>More Information</u>

Computer Architecture for Scientists

Principles and Performance

ANDREW A. CHIEN University of Chicago



Cambridge University Press 978-1-316-51853-3 — Computer Architecture for Scientists Andrew A. Chien Frontmatter <u>More Information</u>

CAMBRIDGE UNIVERSITY PRESS

University Printing House, Cambridge CB2 8BS, United Kingdom

One Liberty Plaza, 20th Floor, New York, NY 10006, USA

477 Williamstown Road, Port Melbourne, VIC 3207, Australia

314-321, 3rd Floor, Plot 3, Splendor Forum, Jasola District Centre, New Delhi - 110025, India

103 Penang Road, #05-06/07, Visioncrest Commercial, Singapore 238467

Cambridge University Press is part of the University of Cambridge.

It furthers the University's mission by disseminating knowledge in the pursuit of education, learning, and research at the highest international levels of excellence.

www.cambridge.org Information on this title: www.cambridge.org/highereducation/isbn/9781316518533 DOI: 10.1017/9781009000598

© Andrew A. Chien 2022

This publication is in copyright. Subject to statutory exception and to the provisions of relevant collective licensing agreements, no reproduction of any part may take place without the written permission of Cambridge University Press.

First published 2022

Printed in the United Kingdom by TJ Books Limited, Padstow, Cornwall, 2022

A catalogue record for this publication is available from the British Library.

ISBN 978-1-316-51853-3 Hardback

Additional resources for this publication at www.cambridge.org/chien

Cambridge University Press has no responsibility for the persistence or accuracy of URLs for external or third-party internet websites referred to in this publication and does not guarantee that any content on such websites is, or will remain, accurate or appropriate.

For my father, who inspired a deep love of science, technology, and teaching, and for my mother, who inspired a deep joy of life

Contents

Preface

1	Computing and the Transformation of Society		
	1.1	Computing Transforms Society and Economy	1
		1.1.1 Home	2
		1.1.2 Automobiles and Transportation	2
		1.1.3 Commerce	3
	1.2	Computing Transforms Science and Discovery	5
	1.3	Extraordinary Characteristics of Computing	6
	1.4	What is Computer Architecture?	8
	1.5	Four Pillars of Computer Performance: Miniaturization, Hidden	
		Parallelism, Dynamic Locality, and Explicit Parallelism	9
	1.6	Expected Background	10
	1.7	Organization of the Book	11
	1.8	Summary	13
	1.9	Problems	13
2	Instr	ructions Sets, Software, and Instruction Execution	16
	2.1	Computer Instruction Sets	16
	2.2	Computer Systems Architecture	17
	2.3	Instruction Set Architecture: RISC-V Example	18
		2.3.1 Computation Instructions	19
		2.3.2 Conditional Control and Procedure Linkage Instructions	20
		2.3.3 Memory Instructions	22
	2.4	Machine Instructions and Basic Software Structures	23
		2.4.1 Implementing Basic Expressions	23
		2.4.2 Implementing Data Structures: Structs and Objects	23
		2.4.3 Implementing One- and Multi-dimensional Arrays	26
		2.4.4 Implementing Conditional Iterative Constructs: Loops	27
		2.4.5 Implementing Procedure Call and Return and the Stack	30
	2.5	Basic Instruction Execution and Implementation	33
		2.5.1 Sequential State and Instruction Execution	35
		2.5.2 Hardware Implementation of Instruction Execution	35
	2.6	Speeding Up Instruction Execution and Program Performance	36

Cambridge University Press 978-1-316-51853-3 — Computer Architecture for Scientists Andrew A. Chien Frontmatter <u>More Information</u>

viii	Cont	tents		
	2.7	Summary	37	
	2.8	Digging Deeper	38	
	2.9	Problems	39	
3	Processors and Scaling: Small is Fast!			
	3.1	Miniaturization and Information Processing	48	
	3.2	What Is the Natural Size of a Computer?	50	
		3.2.1 Example: Bit Size and Speed	52	
		3.2.2 Shrinking Computers	53	
	3.3	Computer Size and Speed	53	
		3.3.1 Smaller Computers Are Faster	54	
		3.3.2 Example: Applying the Size and Clock Period Model	56	
		3.3.3 Size Scaling Computers from Room-Sized to a Single Chip	57	
		3.3.4 Size Scaling Single-Chip Computers: The Power	C (1)	
		Problem and Dennard's Solution	59	
	2.4	5.5.5 The End of Dennard Scaling	02 62	
	5.4 3.5	Size in Other Technologies	02 64	
	3.5	Tiny Computers Enable an Explosion of Applications	65	
	3.0	Summary	67	
	3.8	Digging Deeper	68	
	3.9	Problems	70	
4	Seq	uential Abstraction, But Parallel Implementation	76	
	4.1	Sequential Computation Abstraction	76	
		4.1.1 Sequential Programs	76	
		4.1.2 Instruction-Level Parallelism: Pipelining and More	79	
		4.1.3 Data Dependence and the Illusion of Sequence	80	
	4.2	The Illusion of Sequence: Renaming and Out-of-Order Execution	83	
		4.2.1 Variable and Register Renaming	85	
		4.2.2 Implementing Register Renaming: The Reorder Buffer	88	
		4.2.3 Limits of Out-of-Order Execution	89	
	4.3	Illusion of Causality: Speculative Execution	90	
		4.3.1 Branch Prediction	91	
		4.3.2 Speculative Execution	93	
		4.3.3 Accurate Branch Predictors	97	
		4.3.4 Security Risks of Speculation: Spectre and Meltdown	99	
	4.4	Summary	101	
	4.5 4.6	Digging Deeper Problems	102	
-	NA	nevice. Evaluating Dynamic Leoglity	110	
Ð	ivien	nories: Exploiting Dynamic Locality	113	
	5.1	Miniaturization, and Growing Capacity	113	
	5.2	Software and Applications Demand Memory Capacity	118	

Cambridge University Press 978-1-316-51853-3 — Computer Architecture for Scientists Andrew A. Chien Frontmatter <u>More Information</u>

		Contents	ix
	5.3	Memory System Challenges: The Memory Wall	122
	5.4	Memory Latency	122
		5.4.1 Warping Space–Time (Caches)	123
		5.4.2 Dynamic Locality in Programs	126
		5.4.3 Address Filters (Caches)	128
		5.4.4 The Effectiveness of Filters (Caches)	131
		5.4.5 Implementing Caches (Warping and Filtering)	132
		5.4.6 Recursive Filtering (Multi-level Caches)	133
		5.4.7 Modeling Average Memory Hierarchy Performance	135
	5.5	Why Caches Work so Well and Programming for Locality	137
	5.6	Measuring Application Dynamic Locality and Modeling Performance	142
		5.6.1 Measuring Dynamic Locality: Reuse Distance	142
		5.6.2 Reuse Distance and Dynamic Locality	143
		5.6.3 Modeling an Application's Memory Performance	
		Using Reuse Distance	145
		5.6.4 Tuning a Program for Dynamic Locality	146
	5.7	Access Rate and Parallel Memory Systems	148
	5.8	Summary	150
	5.9	Digging Deeper	151
	5.10	Problems	152
6	The	General Purpose Computer	161
	6.1	A Commercial Processor: Intel Skylake	161
	6.2	A Commercial Memory Hierarchy: Intel Skylake	164
		6.2.1 Caches and Power	165
	6.3	CPUs Are General Purpose Computers	168
	6.4	Perspective: Mathematical Universality and Complexity	170
	6.5	Summary	171
	6.6	Digging Deeper	172
	6.7	Problems	173
7	Bey	ond Sequential: Parallelism in MultiCore and the Cloud	176
	7.1	The End of Dennard Scaling and the Shift to Parallelism	176
	7.2	Parallel Single-Chip Computers: Multicore CPUs	179
		7.2.1 Example: AMD Ryzen Multicore Chip and System	181
	7.3	Programming Multicore Computers: OpenMP and pthreads	182
		7.3.1 OpenMP: Pragma-Based Parallelism	183
		7.3.2 pthreads: Explicit Thread-Parallelism	184
		7.3.3 Challenging Parallelism in a Single Multicore CPU	185
		7.3.4 Simpler Use of Multicore: Libraries and Servers	186
	7.4	Million-Way Parallelism: Supercomputers and the Cloud	187
	7.5	Efficient Parallelism: Computation Grain Size	189
	7.6	Programming Cloud Computers: Coarse-Grained Parallelism	192
		7.6.1 Three-Tier Web: Scalable Web Services	192

Cambridge University Press 978-1-316-51853-3 — Computer Architecture for Scientists Andrew A. Chien Frontmatter <u>More Information</u>

х	Cont	ents		
		7.6.2 Scale-Out Map–Reduce (Hadoop and Spark)	193	
		7.6.3 Microservices: Modular Reliability and Evolution	194	
		7.6.4 Serverless (Function-as-a-Service)	195	
	7.7	Summary	196	
	7.8	Digging Deeper	198	
	7.9	Problems	198	
8	Accelerators: Customized Architectures for Performance			
	8.1	The Emergence of Accelerators	205	
		8.1.1 Accelerator Hardware Opportunities	206	
		8.1.2 Programming and Software Challenges	206	
	8.2	Parallelism Accelerators	208	
		8.2.1 The Architecture of GPUs	208	
		8.2.2 Diverse GPUs and Performance	210	
	8.3	Machine Learning Accelerators	212	
		8.3.1 Google's Tensor Processing Unit	213	
		8.3.2 Cerebras CS-2: A Wafer-Scale Machine Learning Accelerator	215	
		8.3.3 Small Machine Learning Accelerators (Edge)	216	
	8.4	Other Opportunities for Acceleration	217	
	8.5	Limitations and Drawbacks of Accelerated Computing	217	
	8.6	Summary	219	
	8.7	Digging Deeper	219	
	8.8	Problems	220	
9	Com	puting Performance: Past, Present, and Future	226	
	9.1	Historical Computer Performance	226	
	9.2	Future Computer Performance: Opportunities for Performance Increase	228	
		9.2.1 Hardware Scaling and Opportunities	228	
		9.2.2 Resulting Programming and Software Challenges	230	
	9.3	New Computing Models	231	
		9.3.1 Higher-Level Architecture	232	
		9.3.2 Quantum Computing	233	
		9.3.3 Neuromorphic Computing	233	
	9.4	Summary	234	
	9.5	Digging Deeper	235	
	9.6	Problems	235	
Appendix	RISC-V Instruction Set Reference Card			
	Refe	erences	241	
	Inde	2X	247	

Preface

Because of computing's history, the pedagogy of computer architecture has been aimed at future computer designers and engineers. However, a growing number of data scientists, engineers who are not computer engineers, and even the majority of computer scientists (i.e. artificial intelligence and more) see computing as an intellectual tool. These scientists and engineers need an understanding of computer architecture for insights into how hardware enables, shapes, and limits performance. They need an understanding of computer architecture that allows them to reason about performance today and scaling into the future. It is for them that I have undertaken this book.

In fall 2011 I joined the University of Chicago faculty and taught undergraduate computer architecture. As the quarter progressed, it became clear that my students had different backgrounds, motivations, and interests – a marked contrast to the engineering students I taught at the University of Illinois and University of California, San Diego. Over the course of several quarters, I realized that the traditional bottomup approach to teaching computer architecture (from gates to sequential circuits, from instruction sets to pipelines and caches – with an emphasis on how it works and how to optimize it) was not reaching these students. This traditional pedagogy is designed for computer engineering students, and increasingly even many computer science students do not find it compelling.

The University of Chicago students were interested in the "scientific principles" of computer architecture, principles that would enable them to reason about hardware performance for higher-level ends. Their view of computation was as an intellectual multiplier, and they were interested in a top-down view of capabilities, scaling, and limits – not mechanisms. Their point of view reflects computing's broad expansion, most recently manifest in artificial intelligence and data science, that has shifted computer science's center of mass upward and its boundary outward – into a wide variety of sciences (physical, biological, and social) as well as nearly every aspect of society, commerce, and even government.

For these data science, artificial intelligence, and applied machine learning students, a foundation in computer architecture should provide scientific understanding of how the remarkable power of computing is possible, and what limits that power – and, as we look forward to the future, how the continued progress in technology will

xii **Preface**

increase computing capabilities. This book provides such a principles-based grounding in the current and future capabilities of computing hardware. It is designed for several different curricula and settings:

- A quarter-long course in a Computer Science program in a liberal arts college or Data Science program: Chapters 1–3, 5, and 7. Depending on the curricular fit, perhaps add parts of microarchitecture (Chapter 4), framing CPUs as practically universal (Chapter 6), and accelerators (Chapter 8).
- A semester-long course in a Computer Science program in a liberal arts or Data Science program: Chapters 1–5, 7, and 9. If there is time, perhaps add back the hot topic of accelerators (Chapter 8).
- A semester-long course in a Math–Computer Science or Data Science program at an institution with a deep engineering culture: Chapters 1–9. This includes the full text.

I would like to thank the students in my "Computer Architecture for Scientists" course at the University of Chicago for their contributions – their thoughtful questions and responses were critical to increasing the clarity of the text, lectures, and exercises.

Finally, a heartfelt thanks to my family, including my wife, Ellen, and my two daughters, Jennifer and Athena. For their steadfast support for my writing this book and my absence on many late nights, I am grateful. Without your patience, tolerance, and encouragement, it would have been impossible. Thank you!

I am indebted to the University of Chicago, whose spirit of innovative scholarship and pedagogy has been both an inspiration and support for this book!