

COMPUTER
ARCHITECTURE
A Quantitative Approach

John L. Hennessy ■ David A. Patterson

ヘネシー ■ パターソン

コンピュータアーキテクチャ
定量的アプローチ [第6版]

ジョン・L・ヘネシー、デイビッド・A・パターソン
中條拓伯、天野英晴、鈴木 貢 訳

SiB
access

The edition of *Computer Architecture: A Quantitative Approach, 6e* by John L. Hennessy, David A. Patterson (ISBN: 978-0128119051) are published by arrangement with Elsevier Inc.

Copyright © 2019 Elsevier Inc. All rights reserved.

[邦題：ヘネシー ■ パターソン コンピュータアーキテクチャ：定量的アプローチ [第6版]]

No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or any information storage and retrieval system, without permission in writing from the publisher. Details on how to seek permission, further information about the Publisher's permissions policies and our arrangements with organizations such as the Copyright Clearance Center and the Copyright Licensing Agency, can be found at our website: www.elsevier.com/permissions.

This book and the individual contributions contained in it are protected under copyright by the Publisher (other than as may be noted herein).

Notices

Knowledge and best practice in this field are constantly changing. As new research and experience broaden our understanding, changes in research methods, professional practices, or medical treatment may become necessary.

Practitioners and researchers must always rely on their own experience and knowledge in evaluating and using any information, methods, compounds, or experiments described herein. In using such information or methods they should be mindful of their own safety and the safety of others, including parties for whom they have a professional responsibility.

To the fullest extent of the law, neither the Publisher nor the authors, contributors, or editors, assume any liability for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions, or ideas contained in the material herein.

Andrea、Linda、および 4 人の我々の息子に捧ぐ

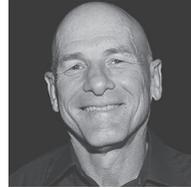
チューリング賞受賞の著者



ジョン・L・ヘネシー (John L. Hennessy) は、1977年からスタンフォード大学の Department of Electrical Engineering and Computer Science に所属し、2000年から2016年の間、同大学の学長を務め、現在は Knight-Hennessy Fellowship の理事である。さらに IEEE と ACM のフェローであり、全米技術アカデミーとアメリカ芸術科学アカデミーの会員である。2001年に RISC 技術への貢献により Eckert-Mauchly Award を受賞、2001年に Seymour Cray Computer Engineering Award を受賞、2000年には John von Neumann Award をデイビッド・パターソンと共同受賞、他数多くの賞を受賞している。また、10 の名誉博士を授与されている。

1981年、彼はスタンフォード大学で大学院生とともに MIPS プロジェクトを開始した。1984年、プロジェクトを終わらせた後、1年の研究休暇の間に、MIPS Computer Systems を共同起業し、最初の商用 RISC マイクロプロセッサを開発した。2017年の時点で50億個をこえる MIPS マイクロプロセッサが、ビデオゲームやパームトップコンピュータからレーザープリンタやネットワークスイッチに至る機器に組み込まれている。彼は続いて最初の拡張可能なキャッシュ透過型マルチプロセッサのプロトタイプである DASH (Director Architecture for Shared Memory) プロジェクトを率いた。このプロジェクトは、現代的なマルチプロセッサで採用されている多くの重要なアイデアを生み出した。技術的な功績や大学での教育に加えて、多くの企業の立ち上げで初期段階のアドバイスや投資を継続して行っている。

[編集補足：ヘネシーは現在 Google 社持ち株会社 Alphabet 社の会長で、本書の共著者デイビッド・パターソンと共に、「2017年度チューリング賞」を2018年に受賞している。]



デイビッド・A・パターソン (David A. Patterson) は、カリフォルニア大学 (UC) バークレーでの40年にわたる教員活動の後に、2016年に Google に移り、Distinguished Engineer に就任した。彼は UCLA 卒業後、直ちに UC バークレーの一員となった。彼は現在も1週間のうち1日は、バークレーで計算機科学の名誉教授として生活を送っている。カリフォルニア大学から教育活動に対して Distinguished Teaching Award を、ACM から Karlstrom Award を、IEEE から Mulligan Education Medal と Undergraduate Teaching Award を受賞している。パターソンは RISC への貢献により IEEE Technical Achievement Award と ACM Eckert-mauchly Award を、RAID (Redundant Arrays of Inexpensive Disks) に対する貢献により IEEE Johnson Information Storage Award を共同受賞している。また、IEEE John von Neumann Medal と C & C 賞をヘネシーと共同受賞している。ヘネシーと同じくパターソンは、アメリカ芸術科学アカデミー、コンピュータ歴史博物館、ACM、および IEEE のフェローであり、全米技術アカデミー、米国科学アカデミー、および Silicon Valley Engineering Hall of Fame に選出されている。彼はアメリカ大統領の情報技術についてのアドバイザリコミッティ、US バークレーの EECS 学科の CS 学科の学科長、Computing Research Association の委員長、ACM 会長を務めた。この功績によって ACM、CRA、そして SIGARCH より、功労賞を受賞した。彼は現在、RISC-V Foundation の取締役副会長である。

US バークレーで、パターソンは最初の VLSI RISC である RISC I の設計と実装を指導した。この研究は商用 SPARC アーキテクチャの基礎となった (これは現在でも SUN Microsystems、富士通他で用いられている)。彼は RAID プロジェクトの指導者であり、多くの企業による高信頼性ディスクの開発に結び付いた。彼は、Network of Workstations (NOW) プロジェクトにも参画し、このプロジェクトは、インターネット企業によるクラスタ技術や、のちにクラウドコンピューティングに結び付いた。彼の現在の興味は、機械学習のための領域特化アーキテクチャの設計や、Open RISC-V 命令セットアーキテクチャの普及活動、そして UC Berkeley RISELab (Real-time Intelligent Secure Execution) の支援にある。

[編集補足：2018年、本書共著者ジョン・ヘネシーと共に「2017年度チューリング賞」を受賞している。]

序 文

Norman P. Jouppi, Google

この40年間の計算機の性能向上の多くは、Mooreの法則と Dennard スケーリングが牽引した大規模かつ並列度が高いシステムによるコンピュータアーキテクチャの進歩に依っている。Mooreの法則は、集積可能なトランジスタの数が2年で倍になるということである。Dennard スケーリングは、MOSの電源電圧が下がるのと同時に形状も小さくなれば、電力密度はほぼ一定であるということである。10年前に Dennard スケーリングが終焉を迎え、Mooreの法則が物理的、あるいは経済的な要因による最近の鈍化を考慮すると、我々の業界における卓越した教科書の第6版ほど、時宜を得たものはないだろう。その理由は以下の通り。

第一の理由は、領域特化アーキテクチャは、過去の Moore の法則と Dennard スケーリングにおける3世代分に匹敵する電力と性能の恩恵をもたらし、汎用アーキテクチャの未来の拡張よりも優れた実装を供することにある。そして、計算機応用の領域が多様化している今日、領域特化アーキテクチャによるアーキテクチャ革命の対象となり得る潜在的な領域がたくさん転がっている。

第二の理由は、オープンソースアーキテクチャの上質な実装は、Mooreの法則が鈍化したためにより長い寿命を有するようになったこと。これにより、最適化と改良を継続する機会が増え、そうする価値が生じてきた。

第三の理由は、Mooreの法則が鈍化したために、異なる技術要素がヘテロジニアスに規模拡大してきたこと。さらに、2.5次元積層や、新型の不揮発性メモリ、光学的内部接続のような新しい技術が開発され、Mooreの法則が単体で頑張るのを補うようになったこと。これらの新技術を使い、均質でない規模拡大を効果的に用いるには、最初の原理から設計上の決定を再検討する必要がある。したがって学生、教員、産業界の実務家を問わず、古きから最新に至るアーキテクチャの技術を身に着けることは重要である。結局のところ領域特化アーキテクチャは、コンピュータアーキテクチャ界における、25年前の命令レベル並列性でのゴールドラッシュ以来の、ワクワドキドキであると信じる。

この版における大きな変更は、領域特化アーキテクチャの章を加えたことである。特別な領域特化アーキテクチャは、汎用プロセッサの実装に比べて、高い性能と、低い消費電力、少ないシリコン面積を達成できることは既知であった。しかし、汎用プロセッサが単一スレッドでの性能を年に40%向上（図1.11を参照）してきた間は、専用アーキテクチャを開発するのに要する余計な時間と、最新の標準マイクロプロセッサの利用が対決すると、専用アーキテクチャの利点を以てしても多くが負けていた。一方で単一コアの性能

の伸びがとて鈍くなった今日では、以前とは異なり、最新の汎用プロセッサ技術を以てしても、専用アーキテクチャが陳腐化しない状況が長く続くことを意味する。第7章ではいくつかの領域特化アーキテクチャを取り扱っている。ディープニューラルネットからは最も大量だが低いデータ精度で済む計算の要求がある。この組み合わせは、専用アーキテクチャから大きな恩恵を受けられる。ディープニューラルネットのための2つのアーキテクチャの実装例が示されている。1つ目は推論に最適化されており、2つ目は学習に最適化されている。画像処理はもう1つの領域の例題であり、大量の計算という要求の一方で、低い精度のデータ型という恩恵がある。さらにこの例題はモバイルデバイスで必要とされることが多いので、専用アーキテクチャによる低消費電力化は非常に有益である。最後に、本質的に再プログラミング可能であるので、FPGAに基づくアクセラレータは1つのデバイスで多様な領域特化アーキテクチャを実現するのに使うことができる。これらは、ネット検索のように頻繁に更新される非定型のアプリケーションで利がある。

アーキテクチャという重要な概念は永遠であるので、この版は最も新しく開発された技術やコスト、例題、そして参考文献によって全体的に更新されている。オープンソースのアーキテクチャにおける最近の開発のペースに合わせて、本書ではRISC-V命令セットアーキテクチャを使うように刷新した。

個人的なコメントとして、ヘネシー先生の博士課程の学生と一緒に研究する機会の後に、現在はGoogleでパターンソン氏の下で働くという機会を得ている。まあ、なんと素晴らしい上司の組み合わせ！

コンピュータアーキテクチャ：定量的アプローチ [第6版] 推薦の言葉

アーキテクチャという重要な概念は永遠であるので、この版は最も新しく開発された技術やコスト、例題、そして参考文献によって全体的に更新されている。オープンソースのアーキテクチャにおける最近の開発のペースに合わせて、本書では RISC-V 命令セットアーキテクチャを使うように刷新した。

—序文から、Norman P. Jouppi、Google

『コンピュータアーキテクチャ：定量的アプローチ』は良質なワインのようにますます美味しくなってきた。最初買ったのは学部卒業の時に、今でも最も頻繁に見る教科書の1つである。

—James Hamilton、Amazon Web Service

ヘネシーとパターソンが本書の最初の版を著したのは、修士課程の学生が5万トランジスタの計算機を構成するような時であった。今日では、ウェアハウス規模の計算機群は、各々が数ダースの独立したプロセッサと数十億のトランジスタから成る沢山のサーバで構成されている。コンピュータアーキテクチャの進化は急速で半端でない。『コンピュータアーキテクチャ：定量的アプローチ』は出版のペースを保ちつつその各版で、新規に登場した重要なアイデアであってこの業界を盛り上げているものを取り上げ、正確な説明と解析を行っている。

—James Larus、Microsoft Research

旧版に対するもう1つの時宜を得た関連性のある追加内容は、半端でなく興奮させるコンピュータアーキテクチャの進化への1つの窓である！ Moore の法則の鈍化に対するこの版の新しい議論と未来のシステムへの示唆は、コンピュータアーキテクトやシステムの実務家の広範囲に渡って必読である。

—Parthasarathy (Partha) Ranganathan、Google

私は『定量的アプローチ』を取る本を愛する、というのはそれらがエンジニアによるエンジニアのためのものであるからだ。ジョン・ヘネシーとデーブ・パターソンは数学が課す限界と物性物理学がもたらす可能性を示す。彼らは現実の例を使って、アーキテクトが実働システムを構成するのに、どのように解析し、測定し、妥協するのかを教示している。この第6版は、Moore の法則が終焉を迎え、深層学習が前例のないほど計算サイクルを要求しているという、絶妙なタイミングで出版された。領域特化アーキテクチャの章では、いくつかの有望なアプローチを説明し、コンピュータアーキテ

クチャにおける1つの復興を予想している。ヨーロッパのルネサンスのさなかの学者と同様に、コンピュータアーキテクトは我々の歴史を理解しなければならない。そして、歴史の教訓と新技術を結合し、世界を再構成するのだ。

—Cliff Young、Google

まえがき

我々がこの本を書いた理由

本書の6つの版を通じて、我々のゴールは、明日への技術的發展となろう基本的原則を繙くこと^{ひもと}であった。コンピュータアーキテクチャにおいて、こういった機会をいただいているという興奮は今でも冷めることはなく、初版でこの分野について書いたことをもう一度繰り返そう。

「これは決して動くことのないペーパーマシンについて語るようなつまらない科学の話ではない。断じて違う！ 切れ味鋭い知的興味に端を発する学問であり、市場からのコスト-性能-消費電力に関する要求圧力に対しバランスをうまく取りながら、結果として、栄誉ある失敗とともに、重要な成功に導いてくれるのである。」

最初の版を書いた時に、我々が主に掲げた目的は、人々がコンピュータアーキテクチャを学び、それについて考える方法を変えることであった。このゴールは依然として変わっておらず、今もなお重要であると感じている。この分野は日進月歩であり、全く現実感のない定義や設計を集めたものではなく、実際の例と現実のコンピュータを用いた測定結果により学ばなければならない。我々は今までに我々と考えを共にする人々は誰でも熱烈に歓迎し、今またこれに加わろうとする人たちを同じように迎え入れている。いずれにせよ、現実のシステムに対して、同じ定量的アプローチと分析ができることを約束しよう。

以前の版と同じように、我々は、新しい版が、先進的コンピュータアーキテクチャと設計の授業で使ってもらえるように、プロの技術者とアーキテクトにとって役に立つようにと努力してきた。最初の版のように、この版は新たなプラットフォーム——パーソナルモバイルデバイスやウェアハウススケールコンピュータ——、および新しいアーキテクチャ——特に領域特化アーキテクチャ——に鮮明な焦点を当てている。以前の版と同じように、この版はコスト-性能-電力のトレードオフと優れた技術的デザインに重点を置いて、コンピュータアーキテクチャを解き明かしている。我々は、この分野が成熟し続け、厳格な定量的基礎に裏打ちされた長く確立された科学技術の学問へと向かうと信じている。

この版について

Moorの法則やDennard則が終焉したことは、コンピュータアーキテクチャに対してマルチコアへの切り替えという大きなインパクトになった。本書では引き続き、携帯電話やタブレット等のパー

ソナルモバイルデバイス (PMD) のクライアントから、クラウドコンピューティングのサーバであるウェアハウススケールコンピュータに至るまでの、コンピューティングのサイズを深掘りする。さらにそれらの中にある別の並列性のテーマであるデータレベル並列性 (DLP) については第1章と第4章で、命令レベル並列性 (ILP) については第3章で、スレッドレベル並列性については第5章で、そして要求レベル並列性 (RLP) については第6章で詳述する。

この版における最も広範囲な変更は、命令セットをMIPSからRISC-Vに切り替えたことである。この現代的でモジュール化されたオープンな命令セットが、情報技術産業に大きな影響を与えるのかは未知数である。このアーキテクチャはオペレーティングシステム界におけるLinuxと同様の重要性を持つようになるかもしれない。

この版で新規に追加したのは第7章で、産業界の実際の例をいくつか示しながら、領域特化アーキテクチャを紹介している。

以前と同様に本書の最初の3つの付録は、*Computer Organization and Design*[†]のような本を読んだことがない読者を対象にして、RISC-V命令セット、メモリ階層、そしてパイプラインについて基礎的な解説を行っている。本書のコストを下げながら興味がある読者を満足させるために、付録を含む補助教材を次のURLからオンラインで取得可能である：<https://www.elsevier.com/books-and-journals/book-companion/9780128119051>。なんと付録のページ数の方が本書のページ数よりも多い！

この版でも引き続き、アイデアの例示を現実の例を使って行うという伝統を守っており、「総合的な実例」の節は一新されている。この版の「総合的な実例」には、ARM Coretex-A53プロセッサやIntel Core i7プロセッサのパイプライン構成やメモリ階層、Google ウェアハウスコンピュータの1つが含まれている。

話題の選択と構成

従来通り、話題の選択については保守的なアプローチを取っている。というのも、この分野では、基礎的な原則を取り扱うことでそこそこカバーできるアイデアよりも、もっと興味深いアイデアがたくさんあるからである。我々は、読者が出会うかもしれないアーキテクチャをすべて広範囲に渡って調査することは避けるようにした。その代わりに、どのような新たなマシンにでも見受けられる

[†] 訳注：デイビッド・A・パターソン、ジョン・L・ヘネシー著、成田光彰訳『コンピュータの構成と設計：ハードウェアとソフトウェアのインタフェース』、日経BP社刊

ような、核心となる概念を説明するように心がけた。アイデアを選ぶのに鍵となる基準は、そのアイデアが定量的な点における議論に耐えうるほどに十分に検証され、広く利用されているかどうかである。

我々の意図は、他の情報源から同じ形では利用できないものに極力焦点を当てることである。このため、可能な限り先進的な内容であることを強く訴えて続けている。実際、ここに取り上げているシステムの中には、文献では記述が見当たらないものもある（コンピュータアーキテクチャについてもっと基礎的なことに限って興味がある方々には、前掲の『コンピュータの構成と設計』を読んでもらいたい）。

内容の概観

第1章では、エネルギー、静的な消費電力、動的な消費電力、半導体のコスト、信頼性、可用性に関する公式を示している（これらの公式は、フロントカバーの内側にも記載してある）。これらを他の章においても活用していただきたい。コンピュータ設計や性能計測の定量的原則の古典的なものに加えて、汎用プロセッサの性能向上の鈍化を示し、これが領域特化アーキテクチャの示唆になっている。

我々の見解では、1990年代に比べて、命令セットアーキテクチャの果たす役割は低下しており、このため、この内容については付録Aに移動した。ここではRISC-Vアーキテクチャを用いている（すぐに参照できるように、RISC-Vの命令セット（ISA）を巻末にまとめてある）。ISAフリークのために、付録Kでは10種類のRISCアーキテクチャ、80x86、DEC VAX、IBM360/370について収録している。

それから、第2章ではメモリ階層の話題に移る。というのも、価格性能比（コストパフォーマンス）——消費電力の原則をこれに適用するのが容易であり、メモリは他の章にとってもクリティカルなリソースであるからである。以前の版にあるように、付録Bにはキャッシュの原理の導入的なおさらいを含めており、必要な場合に見てもらおうためのものである。

第2章では、10種類のキャッシュの先進的な改良技術について論じている。この章では、保護、ソフトウェア管理、ハードウェア管理において有効に働き、クラウドコンピューティングにおいて重要な役割を果たす仮想マシンについて触れている。この章にはSRAMやDRAM技術を対象とするだけでなく、フラッシュメモリに関する新たな技術資料も含まれている。総合的な実例としては、PMDで用いられているARM Cortex A8やサーバに利用されているIntelのCore i7を取り上げている。

第3章は、高性能プロセッサにおける命令レベル並列性の利用について述べており、スーパースカラ実行、分岐予測、投機実行、動的スケジューリング、同時マルチスレッドが含まれている。以前に述べたように、必要となる場合に備えて、付録Cはパイプラインのおさらいである。第3章においても命令レベル並列性（ILP）の限界についてまとめている。第2章と同様に、総合的な実例は、再度ARM Cortex A53とIntel Core i7を取り上げている。第3版では、ItaniumとVLIWに多くのページを割いたが、この内容について

は、付録Hに移動し、このアーキテクチャが、以前主張したことに応えなかったといった我々の考えを示した。

ゲームや動画処理といったマルチメディアアプリケーションの重要性が増すことによって、データレベル並列性を利用できるアーキテクチャの重要性も増えることとなった。特に、グラフィック処理ユニット（GPU）を用いる計算に対する興味が増していく中、GPUが実際にどのように動作しているのか理解しているアーキテクトがほとんどいない。我々はコンピュータアーキテクチャのこの新しい方式を明らかにするために重きを置き、新たな章を追加することに決めた。第4章は、ベクタアーキテクチャの導入から始まるが、これはマルチメディア系のSIMD命令セット拡張やGPUを説明する上での基本となる（付録Gでは、ベクタアーキテクチャについて、さらに深く掘り下げている）。この章ではルーフライン性能モデルを導入し、それを用いてIntel Core i7とNVIDIA GTX280 GPU、それにXeon Platinum 8180とNVIDIA P100 GPUとを比較している。この章はPMD用のTegra P1 GPUについても述べている。

第5章は、マルチコアプロセッサについて述べている。構成原則と性能の両方について精査し、対称メモリおよび分散メモリアーキテクチャについて探究している。本章への第一の追加には、マルチコア-マルチレベルのキャッシュ、マルチコアのコヒーレンス方式、そしてオンチップでのマルチコア相互接続が含まれている。同期やメモリコンシステンシモデルについての話は次に示す。例としては、Intel Core i7を取り上げた。オンチップの相互結合ネットワークに興味がある読者は付録Fを読むべきであり、さらに大規模な並列計算機や科学技術アプリケーションに興味がある人には、付録Iを用意してあるので、ぜひ読んでいただきたい。

第6章はウェアハウススケールコンピュータ（WSC）について述べている。この章は、Google社とAmazon Web Servicesのエンジニアの助けで大きく改訂された。この章はほとんどアーキテクトが気にすることのなかったWSCの設計、コスト、性能に関する詳細についてまとめたものである。コストを含めて、WSCのアーキテクチャや物理的な実装について述べる前に、現在注目を集めているMapReduceプログラミングモデルから始めている。そのコストからクラウドコンピューティングが出現したことが説明でき、それによって所属するところにあるデータセンターよりも、クラウドコンピューティングでWSCを用いて計算する方がもっと安くつくことになる。総合的な実例では、本書で初めて公表することとなったGoogle社のWSCについて述べている。

新規追加の第7章は、領域特化アーキテクチャ（DSA）の必要性が動機になっている。この章では4つのDSAの具体例に基づいてDSAの原理を導き出している。DSAは各々商用機器に配備されたチップに対応している。そしてさらに、単一スレッドの汎用プロセッサの性能がどん詰まりになると、なぜDSAと呼ばれるコンピュータアーキテクチャのルネサンスが期待されるのかを説明している。

これにより付録A～Mを用意することとなった。付録Aは、RISC-V64を含めた命令セットアーキテクチャ（ISA）の原理についてまとめており、付録Kでは、RISC-Vの64ビット版、ARM、MIPS、Power、SPARC、および、これらのマルチメディア拡張について示

してある (80x86、VAX、IBM360/370 といった) 古典的なアーキテクチャや、(Thumb-2、microMIPS、RISC-V C といった) 広く使われている組み込み用命令セットについても示してある。関連する付録 H では、VLIW 命令セットのアーキテクチャや、そのコンパイラについて述べている。

前に述べたように、付録 B と付録 C はキャッシュやパイプラインの基本的なチュートリアルである。キャッシュについて比較的初心者、第 2 章の前に付録 B を読むようにし、パイプラインについて初めて知る読者は、第 3 章の前に付録 C を読んでいただきたい。

付録 D の「ストレージシステム」では、信頼性と可用性について突っ込んだ議論を展開しており、RAID6 の方式について述べつつ RAID について解説し、実際のシステムの統計的な障害について扱っている。引き続き、待ち行列理論と入出力性能のベンチマークを紹介している。我々は、Internet Archive という実際のクラスタでコスト、性能、信頼性について評価している。「総合的な実例」では、NetApp FAS6000 ファイルサーバを取り上げている。

付録 E は、Thomas M. Conte によるもので、組み込み関係についてまとめている。

付録 F は、相互結合ネットワークに関するもので、Timothy M. Pinkston と José Duato によってまとめられた。付録 G は、元々は Krste Asanović によって書かれ、ベクタプロセッサについて詳説している。これらの 2 つ付録は、知りうる中で最も素晴らしい解説であると思う。

付録 H は VLIW と Itanium のアーキテクチャである EPIC について述べている。

付録 I では、並列処理アプリケーションと大規模共有メモリ型の並列処理のコヒーレンスプロトコルについて述べている。付録 J は、David Goldberg によるもので、コンピュータの演算法を詳説している。

Abhishek Bhattacharjee による付録 L は新規に追加されたもので、仮想マシンや、非常に広大なアドレス空間のアドレス変換の設計に焦点を当てている。クラウドプロセッサが増えるにつれて、これらのアーキテクチャ的な増強はさらに重要性を増す。

付録 M は、それぞれの章の「歴史展望と参考文献」をまとめて 1 つの付録にしたものである。これは、各章におけるいろんなアイデアがまっとうなものであることを示すとともに、こういった発明を取りまく歴史的なセンスを示すという狙いがある。ここで記したことは、コンピュータ設計における人間ドラマをとして楽しむことができる。この付録は、アーキテクチャの学生にとって必要となるような参考文献も掲載している。もし時間があるならば、ここで紹介することをお勧めする。それは、なかなか楽しいものであり、直接、発案者からアイデアを聞くことができる点で教育面での効果もある。「歴史展望」は、今までの版で最も人気が高かった部分である。

この教科書の読み方

読者のみなさんには第 1 章から読み始めていただきたいところであるが、そうでない場合は、章と付録を読む順番に関して最善の道が 1 つあるわけではない。すべてを読みたくはないのであれば、

次のような順番はどうだろう。

- メモリ階層：付録 B、第 2 章、付録 D、付録 M
- 命令レベル並列性：付録 C、第 3 章、付録 H
- データレベル並列性：第 4 章、第 6 章、第 7 章、付録 G
- スレッドレベル並列性：第 5 章、付録 F、付録 I
- 要求レベル並列性：第 6 章
- ISA：付録 A、付録 K

付録 E はいつ読んでも良いが、ISA とキャッシュを読んだ後に読むのが最も学習効果がある。付録 J は算術演算に関する知識が必要になったらいつでも読んで欲しい。各章を読み終えた後は付録 M の関連するところを読んでみよう。

章構成

それぞれの章に、同じように発展的な記述の枠組みが設けてある。まずそれぞれの章本体でアイデアを説明する。これを「他の章との関連」で、1 つの章で取り上げたアイデアがどのように他の章に影響を与えるかを示している。その次には「総合的な実例」の節がある。この節は、こういったアイデアがどのように組み合わせる実際のマシンで用いられているのかを示している。

その次に続くのは「誤った考えと落とし穴」の節では、他の人が陥りそうな間違いについて学んでもらうためのものである。すなわち、よくある誤解とか、あなたを待ち受けている、分かっているも陥ってしまうようなアーキテクチャ上の罠について、実例を挙げている。「誤った考えと落とし穴」の節は、この本で最も人気のある節である。そして、それぞれの章は「おわりに」の節でしめくくっている。

ケーススタディと演習問題

各章の終わりには、ケーススタディとそれに伴う演習問題を付けている。このケーススタディは、産業界や大学の専門家によるもので、鍵となる概念を探り、段階的に演習に取り組んでもらうことで理解度を確かめることができる。このケーススタディは、先生らがこれをベースにして、自分の演習問題を作るのに役立つよう、十分に詳細が示されていることが分かるだろう。

各演習問題にある章、節 $\langle m.n \rangle$ は、その問題を解くのにまず必要な本書の関連する節に対応している。これは、読者がまだ読んでいない節に関する問題に手を出すのを避けるとともに、元となる情報の参照を容易にすることを狙っている。演習問題には、読者がその問題を解き終えるのに必要な時間について把握してもらうために、難易度が付けてある。

[10] 5 分以内 (読めばすぐ理解できる)

[15] すべて解くのに 5~15 分

[20] すべて解くのに 15~20 分

[25] 完全に解答を書き上げるのに 1 時間

[30] 短いプログラム作成の課題：プログラミングには、まる 1 日を要しない程度

[40] 大掛かりなプログラム作成課題：2 週間くらい必要

[議論] 他の人と議論を要する話題

ケーススタディと演習問題の解法については、<http://textbooks>。

elsevier.comの Web ページに登録すれば参照できるようになっている。

補助的な記述部分

いろいろな教材については<https://www.elsevier.com/books/computer-architecture/hennessy/978-0-12-811905-1>において、オンラインで利用可能であり、以下のような内容が含まれている。

- 参考となる付録：いくつかは、そのテーマの専門家によるゲスト著者によるもの、一定の先進的なトピックを扱っている。
- 歴史展望：本書の各章で解説する鍵となるアイデアがどのように発展していったか探ったもの。
- 授業用のパワーポイントスライド
- 本書の図の PDF、JPEG、PPT フォーマット
- 関連する文献の Web へのリンク
- 正誤表

新たな資料や他の有用なものが掲載されている Web へのリンクは、一定の期間毎に付け加える予定である。

本書改善の手助けのお願い

最後に、この本を読んでお小遣いを稼げることができる（コスト性能比について話そう!）。この後の謝辞を読んでいただければ、ミス訂正に対して延々と連なるリストがあるのがお分かりいただけるだろう。この本は増刷を重ねているため、訂正の機会も多くなる。もし、まだ残っているバグを発見したら、電子メール (ca6bugs@mkp.com) にて出版社まで連絡していただきたい。

我々は本書についてのさまざまなご意見についても受け入れ、もう1つのアドレスであるca6comments@mkp.com宛にメールをお送りいただきたい。

おわりに

繰り返すが、本書は真の意味での共著であり、我々二人が半分の章と半分の付録を書いたものであることを伝えておく。かくも長きにわたり、この仕事の半分を分かち合い、作業がうまくいかないような時にはインスピレーションを交換し合い、難しい概念を説明するために鍵となるような見識を持ち合い、週末を費やして本文を査読し合い、他にいろいろある職務の重みがのしかかって、なかなかペンを持つことができないような時には、心から同情し合った（我々の履歴書が証明するように、この職務は版を重ねるにつれ、指数的に増大していった）。

それゆえ、繰り返すが、これからお読みいただくものに対しては、その非難をも等しく分け合うつもりであることをもう一度申し上げておこう。

ジョン・ヘネシー ■ デイビッド・パターソン

謝 辞

本書は単に第6版となっているが、実際は、11種類の版を作成した。最初の版については3種類のバージョン（アルファ、ベータ、最終版）があり、第2版、第3版、第4版に対しては、それぞれ2つのバージョン（ベータ、最終版）があるのである。その長い道のりの過程で、何百人もの査読者とユーザの助けをお借りした。皆、本書を良くしたいという思いで助けてくれた。それゆえ、この本のいずれかのバージョンに貢献してくれたすべての方々のリストを付けることにした。

第6版の貢献者

旧版同様、本書は大勢のボランティアの方々を含むコミュニティの努力の成果である。彼らの助けなしには、本版がかくも磨き上げられることはなかった。

査読者

Jason D. Bakos, University of South Carolina ; Rajeev Balasubramonian, University of Utah ; Jose Delgado-Frias, Washington State University ; Diana Franklin, The University of Chicago ; Norman P. Jouppi, Google ; Hugh C. Lauer, Worcester Polytechnic Institute ; Gregory Peterson, University of Tennessee ; Bill Pierce, Hood College ; Parthasarathy Ranganathan, Google ; William H. Robinson, Vanderbilt University ; Pat Stakem, Johns Hopkins University ; Cliff Young, Google ; Amr Zaky, University of Santa Clara ; Gerald Zarnett, Ryerson University ; Huiyang Zhou, North Carolina State University。

California 大学 Berkeley 校 Par 研究室と RAD 研究室メンバー（第1章、第4章、第6章を何度も査読いただき、GPUとWSCに関する解説を形作ってくれた）：Krste Asanović、Michael Armbrust、Scott Beamer、Sarah Bird、Bryan Catanzaro、Jike Chong、Henry Cook、Derrick Coetzee、Randy Katz、Yunsup Lee、Leo Meyervich、Mark Murphy、Zhangxi Tan、Vasily Volkov、and Andrew Waterman。

付 録

Krste Asanović, University of California, Berkeley (付録 G) ; Abhishek Bhattacharjee, Rutgers University (付録 L) ; Thomas M. Conte, North Carolina State University (付録 E) ; José Duato, Universitat Politècnica de València and Simula (付録 F) ; David Goldberg, Xerox PARC (付録 J) ; Timothy M. Pinkston,

University of Southern California (付録 F)。

Universidad Politècnica de Valencia の José Flich は、付録 F を改訂する上で、重要な貢献をいただいた。

ケーススタディと演習問題

Jason D. Bakos, University of South Carolina (第3章と第4章) ; Rajeev Balasubramonian, University of Utah (第2章) ; Diana Franklin, University of Chicago (第1章と付録 C) ; Norman P. Jouppi, Google (第2章) ; Naveen Muralimanohar, HP Labs (第2章) ; Gregory Peterson, University of Tennessee (付録 A) ; Parthasarathy Ranganathan, Google (第6章) ; Cliff Young, Google (第7章) ; Amr Zaky, University of Santa Clara (第5章と付録 B)。

Jichuan Chang, Junwhan Ahn, Rama Govindaraju、および Milad Hashemi は、第6章におけるケーススタディと演習問題を作成したり、チェックするのにお手伝いいただいた。

追加資料

John Nickolls, Steve Keckler と NVIDIA の Michael Toksvig (第4章の NVIDIA GPU) ; Intel の Victor Lee (第4章の Core i7 と GPU との比較) ; LBNL の John Shalf (第4章の最近のベクタアーキテクチャ) ; LBNL の Sam Williams (第4章のコンピュータのルーフラインモデル) ; Australian National University の Steve Blackburn とオースチンにある University of Texas (オースチン校) の Kathryn McKinley (第5章の Intel 製プロセッサの性能と消費電力の計測) ; Google の Luiz Barroso, Urs Hölzle, Jimmy Clidaris, Bob Felderman と Chris Johnson (第6章の Google 社の WSC) ; Amazon Web Services の James Hamilton (第6章の配電とコストモデル)。

University of South Carolina の Jason D. Bakos は、本版の授業用スライドを更新してくれた。

本書は、もちろん出版社なしでは出版することができなかった。すべての Morgan Kaufmann/Elsevier のスタッフに、その努力とサポートについて感謝したい。この第5版については、特に編集者の Nate McFadden と Steve Merken には、サーベイを調整し、専門委員会をまとめ、ケーススタディと演習問題を発展させ、グループ、査読結果、付録の更新をまとめてくれたことに対し、特別な感謝を捧げたい。

我々の大学のスタッフである Margaret Rowland と Roxana Infante には、数知れない速達の発送とともに、本書のために我々が

作業していた間の、Stanford 大学と California 大学 Berkeley 校の留守を守ってくれたことに感謝したい。

最後となるが、早朝から読んだり、考えたり、書いたりする事態がどんどん増えていく状況に耐え忍んでくれた我々の妻たちに、感謝の思いを捧げたい。

旧版の貢献者

査読者

George Adams, Purdue University ; Sarita Adve, University of Illinois at Urbana-Champaign ; Jim Archibald, Brigham Young University ; Krste Asanović, Massachusetts Institute of Technology ; Jean-Loup Baer, University of Washington ; Paul Barr, Northeastern University ; Rajendra V. Boppana, University of Texas, San Antonio ; Mark Brehob, University of Michigan ; Doug Burger, University of Texas, Austin ; John Burger, SGI ; Michael Butler ; Thomas Casavant ; Rohit Chandra ; Peter Chen, University of Michigan ; the classes at SUNY Stony Brook, Carnegie Mellon, Stanford, Clemson, and Wisconsin ; Tim Coe, Vitesse Semiconductor ; Robert P. Colwell ; David Cummings ; Bill Dally ; David Douglas ; José Duato, Universitat Politècnica de València and Simula ; Anthony Duben, Southeast Missouri State University ; Susan Eggers, University of Washington ; Joel Emer ; Barry Fagin, Dartmouth ; Joel Ferguson, University of California, Santa Cruz ; Carl Feynman ; David Filo ; Josh Fisher, Hewlett-Packard Laboratories ; Rob Fowler, DIKU ; Mark Franklin, Washington University (セントルイス) ; Kourosh Gharachorloo ; Nikolas Gloy, Harvard University ; David Goldberg, Xerox Palo Alto Research Center ; Antonio González, Intel and Universitat Politècnica de Catalunya ; James Goodman, University of Wisconsin-Madison ; Sudhanva Gurumurthi, University of Virginia ; David Harris, Harvey Mudd College ; John Heinlein ; Mark Heinrich, Stanford ; Daniel Helman, University of California, Santa Cruz ; Mark D. Hill, University of Wisconsin-Madison ; Martin Hopkins, IBM ; Jerry Huck, Hewlett-Packard Laboratories ; Wen-mei Hwu, University of Illinois at Urbana-Champaign ; Mary Jane Irwin, Pennsylvania State University ; Truman Joe ; Norm Jouppi ; David Kaeli, Northeastern University ; Roger Kieckhafer, University of Nebraska ; Lev G. Kirischian, Ryerson University ; Earl Killian ; Allan Knies, Purdue University ; Don Knuth ; Jeff Kuskin, Stanford ; James R. Larus, Microsoft Research ; Corinna Lee, University of Toronto ; Hank Levy ; Kai Li, Princeton University ; Lori Liebrock, University of Alaska, Fairbanks ; Mikko Lipasti, University of Wisconsin-Madison ; Gyula A. Mago, University of North Carolina, Chapel Hill ; Bryan Martin ; Norman Matloff ; David Meyer ; William Michalson, Worcester Polytechnic Institute ; James Mooney ; Trevor Mudge, University of Michigan ; Ramadass Nagarajan, University of Texas at Austin ; David Nagle, Carnegie Mellon University ; Todd Narter ; Victor

Nelson ; Vojin Oklobdzija, University of California, Berkeley ; Kunle Olukotun, Stanford University ; Bob Owens, Pennsylvania State University ; Greg Papadapoulous, Sun Microsystems ; Joseph Pfeiffer ; Keshav Pingali, Cornell University ; Timothy M. Pinkston, University of Southern California ; Bruno Preiss, University of Waterloo ; Steven Przybylski ; Jim Quinlan ; Andras Radics ; Kishore Ramachandran, Georgia Institute of Technology ; Joseph Rameh, University of Texas, Austin ; Anthony Reeves, Cornell University ; Richard Reid, Michigan State University ; Steve Reinhardt, University of Michigan ; David Rennels, University of California, Los Angeles ; Arnold L. Rosenberg, University of Massachusetts, Amherst ; Kaushik Roy, Purdue Acknowledgments University ; Emilio Salgueiro, Unysis ; Karthikeyan Sankaralingam, University of Texas at Austin ; Peter Schnorf ; Margo Seltzer ; Behrooz Shirazi, Southern Methodist University ; Daniel Siewiorek, Carnegie Mellon University ; J. P. Singh, Princeton ; Ashok Singhal ; Jim Smith, University of Wisconsin-Madison ; Mike Smith, Harvard University ; Mark Smotherman, Clemson University ; Gurindar Sohi, University of Wisconsin-Madison ; Arun Somani, University of Washington ; Gene Tagliarin, Clemson University ; Shyamkumar Thoziyoor, University of Notre Dame ; Evan Tick, University of Oregon ; Akhilesh Tyagi, University of North Carolina, Chapel Hill ; Dan Upton, University of Virginia ; Mateo Valero, Universidad Politecnica de Cataluña, Barcelona ; Anujan Varma, University of California, Santa Cruz ; Thorsten von Eicken, Cornell University ; Hank Walker, Texas A&M ; Roy Want, Xerox Palo Alto Research Center ; David Weaver, Sun Microsystems ; Shlomo Weiss, Tel Aviv University ; David Wells ; Mike Westall, Clemson University ; Maurice Wilkes ; Eric Williams ; Thomas Willis, Purdue University ; Malcolm Wing ; Larry Wittie, SUNY Stony Brook ; Ellen Witte Zegura, Georgia Institute of Technology ; Sotirios G. Zivavras, New Jersey Institute of Technology.

付 録

ベクタプロセッサの付録は、マサチューセッツ工科大学の Krste Asanović によって修正していただいた。浮動小数点に関する付録は元々は Xerox 社 PARC の David Goldberg 氏が書かれたものである。

演習問題

George Adams, Purdue University ; Todd M. Bezenek, University of Wisconsin, Madison (彼の祖母 Ethel Eshom さんの記念に) ; Susan Eggers ; Anoop Gupta ; David Hayes ; Mark Hill ; Allan Knies ; Ethan L. Miller, University of California, Santa Cruz ; Parthasarathy Ranganathan, Compaq Western Research Laboratory ; Brandon Schwartz, University of Wisconsin, Madison ; Michael Scott ; Dan Siewiorek ; Mike Smith ; Mark Smotherman ; Evan Tick ; Thomas Willis.

ケーススタディと演習問題

Andrea C. Arpaci-Dusseau, University of Wisconsin-Madison ; Remzi H. Arpaci-Dusseau, University of Wisconsin-Madison ; Robert P. Colwell, R&E Colwell & Assoc., Inc. ; Diana Franklin, California Polytechnic State University, San Luis Obispo ; Wen-mei W. Hwu, University of Illinois at Urbana-Champaign ; Norman P. Jouppi, HP Labs ; John W. Sias, University of Illinois at Urbana-Champaign ; David A. Wood, University of Wisconsin-Madison.

特別な謝意

Duane Adams, Defense Advanced Research Projects Agency ; Tom Adams ; Sarita Adve, University of Illinois at Urbana-Champaign ; Anant Agarwal ; Dave Albonesi, University of Rochester ; Mitch Alsup ; Howard Alt ; Dave Anderson ; Peter Ashenden ; David Bailey ; Bill Bandy, Defense Advanced Research Projects Agency ; Luiz Barroso, Compaq's Western Research Lab ; Andy Bechtolsheim ; C. Gordon Bell ; Fred Berkowitz ; John Best, IBM ; Dileep Bhandarkar ; Jeff Bier, BDTI ; Mark Birman ; David Black ; David Boggs ; Jim Brady ; Forrest Brewer ; Aaron Brown, University of California, Berkeley ; E. Bugnion, Compaq's Western Research Lab ; Alper Buyuktosunoglu, University of Rochester ; Mark Callaghan ; Jason F. Cantin ; Paul Carrick ; Chen-Chung Chang ; Lei Chen, University of Rochester ; Pete Chen ; Nhan Chu ; Doug Clark, Princeton University ; Bob Cmelik ; John Crawford ; Zarka Cvetanovic ; Mike Dahlin, University of Texas, Austin ; Merrick Darley ; DEC Western Research Laboratory のスタッフ ; John DeRosa ; Lloyd Dickman ; J. Ding ; Susan Eggers, University of Washington ; Wael El-Essawy, University of Rochester ; Patty Enriquez, Mills ; Milos Ercegovac ; Robert Garner ; K. Gharachorloo, Compaq's Western Research Lab ; Garth Gibson ; Ronald Greenberg ; Ben Hao ; John Henning, Compaq ; Mark Hill, University of Wisconsin-Madison ; Danny Hillis ; David Hodges ; Urs Hölzle, Google ; David Hough ; Ed Hudson ; Chris Hughes, University of Illinois at Urbana-Champaign ; Mark Johnson ; Lewis Jordan ; Norm Jouppi ; William Kahan ; Randy Katz ; Ed Kelly ; Richard Kessler ; Les Kohn ; John Kowaleski, Compaq Computer Corp ; Dan Lambright ; Gary Lauterbach, Sun Microsystems ; Corinna Lee ; Ruby Lee ; Don Lewine ; Chao-Huang Lin ; Paul Losleben, Defense Advanced Research Projects Agency ; Yung-Hsiang Lu ; Bob Lucas, Defense Advanced Research Projects Agency ; Ken Lutz ; Alan Mainwaring, Intel Berkeley Research Labs ; Al Marston ; Rich Martin, Rutgers ; John Mashey ; Luke McDowell ; Sebastian Mirola, Trimedia Corporation ; Ravi Murthy ; Biswadeep Nag ; Lisa Noordergraaf, Sun Microsystems ; Bob Parker, Defense Advanced Research Projects Agency ; Vern Paxson, Center for Internet Research ; Lawrence Prince ; Steven Przybylski ; Mark Pullen, Defense Advanced Research Projects Agency ; Chris

Rowen ; Margaret Rowland ; Greg Semeraro, University of Rochester ; Bill Shannon ; Behrooz Shirazi ; Robert Shomler ; Jim Slager ; Mark Smotherman, Clemson University ; the SMT research group at the University of Washington ; Steve Squires, Defense Advanced Research Projects Agency ; Ajay Sreekanth ; Darren Staples ; Charles Stapper ; Jorge Stolfi ; Peter Stoll ; 最初の本書制作時に我慢してくれた Stanford と Berkeley の学生 ; Bob Supnik ; Steve Swanson ; Paul Taysom ; Shreekant Thakkar ; Alexander Thomasian, New Jersey Institute of Technology ; John Toole, Defense Advanced Research Projects Agency ; Kees A. Vissers, Trimedia Corporation ; Willa Walker ; David Weaver ; Ric Wheeler, EMC ; Maurice Wilkes ; Richard Zimmerman.

John Hennessy ■ David Patterson



目次

チューリング賞受賞の著者	iv		
序文 [Norman P. Jouppi, Google]	v		
推薦の言葉	vi		
まえがき	vii		
我々がこの本を書いた理由	vii		
この版について	vii		
話題の選択と構成	vii		
内容の概観	viii		
この教科書の読み方	ix		
章構成	ix		
ケーススタディと演習問題	ix		
補助的な記述部分	x		
本書改善の手助けのお願い	x		
おわりに	x		
謝辞	xi		
第6版の貢献者	xi		
旧版の貢献者	xii		
1 定量的な設計と解析の基礎	1		
1.1 はじめに	1		
1.2 コンピュータのクラス	3		
1.2.1 Internet of Things/組み込みコンピュータ	3		
1.2.2 パーソナルモバイルデバイス	4		
1.2.3 デスクトップコンピューティング	4		
1.2.4 サーバ	5		
1.2.5 クラスタ/ウェアハウススケールコンピュータ	5		
1.2.6 並列性のクラスと並列アーキテクチャ	5		
1.3 コンピュータアーキテクチャを設計する	6		
1.3.1 命令セットアーキテクチャ：コンピュータアーキテクチャの表層的な見方	6		
1.3.2 正統的なコンピュータアーキテクチャ：構成とハードウェアを、目標と機能的要求を満足するように設計する	9		
1.4 テクノロジーのトレンド	10		
1.4.1 性能のトレンド：レイテンシを上回るバンド幅	11		
1.4.2 トランジスタ性能と配線のスケールリング	12		
1.5 半導体の電力とエネルギーのトレンド	13		
1.5.1 電力とエネルギー：システム面	13		
1.5.2 マイクロプロセッサのエネルギーと電力	14		
1.5.3 エネルギーの限界によるコンピュータアーキテクチャの移り変わり	15		
1.6 コストのトレンド	16		
1.6.1 時間、量、標準部品化のインパクト	16		
1.6.2 集積回路のコスト	17		
1.6.3 コストと価格	19		
1.6.4 製造コスト対運用コスト	19		
1.7 確実性	19		
1.8 性能の測定、報告、整理の方法	20		
1.8.1 ベンチマーク	21		
1.8.2 性能評価の結果の報告	24		
1.8.3 性能評価のまとめ方	24		
1.9 コンピュータ設計の定量的な原則	25		
1.9.1 並列性を利用せよ	25		
1.9.2 局所性の原則	26		
1.9.3 共通の場合に集中せよ	26		
1.9.4 Amdahlの法則	26		
1.9.5 プロセッサの性能式	27		
1.10 総合的な実例：性能、価格、電力	29		
1.11 誤った考えと落とし穴	31		
1.12 おわりに	33		
1.13 歴史展望と参考文献	35		
1.14 ケーススタディと演習問題	35		
ケーススタディ1：製造コスト	35		
ケーススタディ2：コンピュータシステムの消費電力	35		
演習問題	37		
2 メモリ階層の設計	41		
2.1 はじめに	41		
2.1.1 メモリ階層の基本：簡単なおさらい	43		
2.2 メモリ技術と最適化	45		
2.2.1 SRAMテクノロジー	45		
2.2.2 DRAMテクノロジー	45		
2.2.3 DRAMチップ内での性能の改善：SDRAM	46		
2.2.4 グラフィックデータ用DRAM	48		
2.2.5 パッケージの技術革新： 積層DRAMと組み込みDRAM	48		
2.2.6 フラッシュメモリ	48		
2.2.7 相変化メモリ技術	49		

2.2.8	メモリのディペンダビリティの向上	49	3.4.1	動的スケジューリング：その発想	99
2.3	キャッシュの性能を向上させる10の高度な改良法	50	3.4.2	Tomasuloのアプローチを用いる動的スケジューリング	100
2.4	保護：仮想メモリと仮想マシン	61	3.5	動的スケジューリング：例題とアルゴリズム	103
2.4.1	仮想メモリを通じた保護	62	3.5.1	Tomasuloアルゴリズムの詳細	105
2.4.2	仮想マシンを用いた保護	62	3.5.2	Tomasuloアルゴリズム：ループベースの例	106
2.4.3	仮想マシンモニタの要求	63	3.6	ハードウェアベースの投機処理	107
2.4.4	仮想マシンを支援する命令セットアーキテクチャ	63	3.7	複数命令発行と静的スケジューリングを用いた命令レベル並列性の抽出	113
2.4.5	仮想メモリとI/O上の仮想マシンの影響	64	3.7.1	基本的なVLIWプロセッサのアプローチ	113
2.4.6	効率的な仮想化とセキュリティの向上のための命令セット拡張	64	3.8	動的スケジューリング、複数命令発行および投機処理を用いた命令レベル並列性の抽出	115
2.4.7	VMMの例：Xen仮想マシン	65	3.9	命令供給と投機処理のための高度な技術	119
2.5	他の章との関連：メモリ階層の設計	66	3.9.1	命令フェッチバンド幅の改良	119
2.5.1	保護、仮想化と命令セットアーキテクチャ	66	3.9.2	特殊な分岐の予測器：予測手続きリターン、間接ジャンプ、およびループ分岐	121
2.5.2	自律的命令フェッチユニット	66	3.9.3	投機処理：実装に関する検討項目および拡張	121
2.5.3	投機的実行とメモリアクセス	66	3.10	他の章との関連：ILPのアプローチとメモリシステム	124
2.5.4	特殊な命令キャッシュ	66	3.10.1	「ハードウェアによる投機」対「ソフトウェアによる投機」	124
2.5.5	キャッシュされたデータの一貫性	66	3.10.2	投機実行とメモリシステム	125
2.6	総合的な実例：ARM Cortex-A53とIntel Core i7 6700のメモリ階層	67	3.11	マルチスレッディング：単一プロセッサスループット改善のためのスレッドレベル並列性抽出	125
2.6.1	ARM Cortex-A53	67	3.11.1	スーパースカラプロセッサ上での同時マルチスレッディングの効果	127
2.6.2	Cortex-A53メモリ階層の性能	68	3.12	総合的な実例：ARM Cortex-A53とCore i7 6700	128
2.6.3	Intel Core i7	69	3.12.1	ARM Cortex-A53	128
2.7	誤った考えと落とし穴	74	3.12.2	Intel Core i7	130
2.8	おわりに：将来予測	76	3.13	誤った考えと落とし穴	133
2.9	歴史展望と参考文献	77	3.14	おわりに：次は何か	136
2.10	ケーススタディと演習問題	77	3.15	歴史展望と参考文献	137
	ケーススタディ1：		3.16	ケーススタディと演習問題	137
	最新技術によるキャッシュの性能改善	77		ケーススタディ：マイクロアーキテクチャ技法の影響	137
	ケーススタディ2：			演習問題	141
	総合的な実例：高並列メモリシステム	78	4	ベクタ、SIMD、GPUにおけるデータレベル並列性	145
	ケーススタディ3：		4.1	はじめに	145
	さまざまなメモリシステムの構成の影響を調べる	80	4.2	ベクタアーキテクチャ	146
	演習問題	81	4.2.1	RV64V拡張	146
3	命令レベル並列性とその活用	87	4.2.2	ベクタプロセッサはどのように動くのか：その一例	148
3.1	命令レベル並列性：概念とチャレンジ	87	4.2.3	ベクタ実行時間	149
3.1.1	命令レベル並列性とは何か	88	4.2.4	複数のレーン：	
3.1.2	データ依存とハザード	88		1クロックサイクル当たり1要素を超えて	150
3.1.3	制御依存	90	4.2.5	ベクタ長レジスタ：32以外のループを取り扱う	152
3.2	命令レベル並列性技術のためのコンパイラの基本	91	4.2.6	プレディケートレジスタ：	
3.2.1	基本的なパイプラインスケジューリングとループアンローリング	91		ベクタループのIF文の制御	152
3.2.2	ループアンローリングとスケジューリングのまとめ	93	4.2.7	メモリバンク：	
3.3	進んだ分岐予測による分岐コストの削減	94		ベクタロード-ストアユニットへの帯域の確保	153
3.3.1	相関を利用する分岐予測	94			
3.3.2	トーナメント予測：ローカル予測とグローバル予測を適切に組み合わせる方式	96			
3.3.3	TAGE（タグ付きハイブリッド予測器）	96			
3.3.4	Intel Core i7分岐予測の進歩	98			
3.4	動的スケジューリングによるデータハザードの克服	98			

4.2.8 スライド： ベクタアーキテクチャにおける多次元配列処理	154	5.2.6 基本コヒーレンスプロトコルの拡張	200
4.2.9 ギャザーとスカッター： ベクタアーキテクチャでの疎行列の扱い	154	5.2.7 対称型共有メモリ型マルチプロセッサとスヌーププロ トコルの限界	201
4.2.10 ベクタアーキテクチャのプログラミング	155	5.2.8 スヌープキャッシュコヒーレンス制御の実現	202
4.3 マルチメディア向けSIMD拡張命令セット	156	5.3 対称型共有メモリ型マルチプロセッサの性能	203
4.3.1 マルチメディアSIMDアーキテクチャのプログラミング	157	5.3.1 実アプリケーション処理	204
4.3.2 性能可視化のルーフラインモデル	158	5.3.2 マルチプログラミングとOSのワークロード	206
4.4 グラフィック処理ユニット	159	5.3.3 マルチプログラムとOSワークロードの性能	207
4.4.1 GPUのプログラミング	159	5.4 分散共有メモリとディレトリベースコヒーレンス制御	209
4.4.2 NVIDIA GPUの計算機械としての構造	160	5.4.1 ディレトリベースのキャッシュコヒーレンスプロト コル：基本概念	210
4.4.3 NVIDIA GPUの命令セットアーキテクチャ	164	5.4.2 ディレトリプロトコル実際例	211
4.4.4 GPUにおける条件分岐	166	5.5 同期：その基本	213
4.4.5 NVIDIA GPUのメモリ構成	168	5.5.1 基本ハードウェアプリミティブ	213
4.4.6 Pascal GPUアーキテクチャにおける革新	168	5.5.2 コヒーレンス制御を用いたロック機構の実現	215
4.4.7 ベクタアーキテクチャとGPUの類似点と相違点	170	5.6 メモリコンシステンシモデル：導入	216
4.4.8 マルチメディアSIMD計算機とGPUの類似点と相違点	172	5.6.1 プログラマからの見え方	217
4.4.9 まとめ	172	5.6.2 リラックスコンシステンシモデル： その基本とリリースコンシステンシ	218
4.5 ループレベル並列性の検出と増強	173	5.7 他の章との関連	219
4.5.1 依存性の発見	175	5.7.1 コンパイラによる最適化とコンシステンシモデル	219
4.5.2 依存する計算の除去	177	5.7.2 厳密なコンシステンシモデルにおける遅延隠蔽のため の投機実行	219
4.6 他の章との関連	177	5.7.3 包含 (Inclusion) とその実現	220
4.6.1 エネルギーとDLP：遅くて幅広と速くて幅狭	177	5.7.4 マルチプロセッシングとマルチスレッドを用いた性能 向上	221
4.6.2 バンクメモリとグラフィックメモリ	177	5.8 総合的な事例：マルチコアプロセッサとその性能	221
4.6.3 スライドアクセスとTLBミス	178	5.8.1 マルチプログラムワークロードにおけるマルチコア ベースのマルチプロセッサの性能	221
4.7 総合的な事例：組み込み対サーバGPU、およびTesla 対Core i7	178	5.8.2 さまざまなワークロードによるXeon MPのスケラビ リティ	226
4.7.1 GPUとマルチメディアSIMD付きMIMDの比較	178	5.8.3 Intel Core i7 920マルチコアの性能とエネルギー効率	227
4.7.2 比較結果の更新	181	5.9 誤った考えと落とし穴	228
4.8 誤った考えと落とし穴	182	5.10 マルチコアの性能向上の将来	230
4.9 おわりに	183	5.11 おわりに	232
4.10 歴史展望と参考文献	183	5.12 歴史展望と参考文献	232
4.11 ケーススタディと演習問題	183	5.13 ケーススタディと演習問題	233
ケーススタディ： ベクタカーネルのベクタプロセッサやGPUでの実装	183	ケーススタディ1： シングルチップマルチコアマルチプロセッサ	233
演習問題	185	ケーススタディ2： 単純なディレトリベースのコヒーレンス制御	235
5 スレッドレベル並列性	189	ケーススタディ3：メモリコンシステンシ 演習問題	237 238
5.1 はじめに	189	6 要求レベル並列性/データレベル並列性を利用した ウェアハウススケールコンピュータ	243
5.1.1 マルチプロセッサアーキテクチャ：問題と解決策	190	6.1 はじめに	243
5.1.2 並列処理の目指すもの	192	6.2 ウェアハウススケールコンピュータのプログラミング	
5.2 集中共有メモリ型アーキテクチャ	194		
5.2.1 マルチプロセッサのキャッシュコヒーレンス制御とは	194		
5.2.2 コヒーレンスを維持するための基本方式	195		
5.2.3 スヌープコヒーレンスプロトコル	196		
5.2.4 基本的な実装方法	197		
5.2.5 プロトコル例	198		

モデルとワークロード	246	7.4.3 TPUの命令セットアーキテクチャ	290
6.3 ウェアハウススケールコンピュータのコンピュータアーキテクチャ	249	7.4.4 TPUのマイクロアーキテクチャ	290
6.3.1 ストレージ	250	7.4.5 TPUの実装	291
6.3.2 WSCメモリ階層	250	7.4.6 TPUのソフトウェア	292
6.4 ウェアハウススケールコンピュータの効率とコスト	252	7.4.7 TPUの改良	292
6.4.1 WSCの効率測定	253	7.4.8 まとめ：TPUはガイドラインにどう沿っているか	293
6.4.2 WSCのコスト	254	7.5 MicrosoftのCatapult：柔軟なデータセンターのアクセラレータ	294
6.5 クラウドコンピューティング：ユーティリティコンピューティングの復活	256	7.5.1 Catapultの実装とアーキテクチャ	294
6.5.1 Amazon Webサービス	257	7.5.2 Catapultのソフトウェア	295
6.5.2 AWSクラウドの規模	260	7.5.3 CatapultでのCNN	295
6.6 他の章との関連	261	7.5.4 Catapultでの検索高速化	296
6.6.1 WSCネットワーク上のボトルネックの回避	261	7.5.5 Catapult V1の配備	297
6.6.2 サーバ内における効率的なエネルギー利用方法	262	7.5.6 Catapult V2	298
6.7 総合的な事例：Google社のウェアハウススケールコンピュータ	263	7.5.7 まとめ： Catapultはガイドラインにどう沿っているか	299
6.7.1 Google WSCでの配電	263	7.6 IntelのCrest：学習向けデータセンターのアクセラレータ	299
6.7.2 Google WSCの冷却	263	7.7 Pixel Visual Core：パーソナルモバイルデバイスのための画像処理ユニット	300
6.7.3 Google WSCのラック	265	7.7.1 ISP：IPUの祖先のハードウェア実装版	301
6.7.4 Google WSCにおけるネットワーク関連	265	7.7.2 Pixel Visual Coreのソフトウェア	301
6.7.5 Google WSCにおけるサーバ	266	7.7.3 Pixel Visual Coreの哲学	301
6.7.6 結 論	267	7.7.4 Pixel Visual Coreのhalo	302
6.8 誤った考えと落とし穴	267	7.7.5 Pixel Visual Coreのプロセッサ	303
6.9 おわりに	269	7.7.6 Pixel Visual Coreの命令セットアーキテクチャ	303
6.10 歴史展望と参考文献	270	7.7.7 Pixel Visual Coreの例	303
6.11 ケーススタディと演習問題	270	7.7.8 Pixel Visual Coreのプロセッシングエレメント	304
ケーススタディ1： ウェアハウススケールコンピュータの設計方針に影響を 与える総所有コスト	270	7.7.9 2次元のラインバッファとそのコントローラ	304
ケーススタディ2： WSCにおけるリソース割り当てとTCO	271	7.7.10 Pixel Visual Coreの実装	305
演習問題	272	7.7.11 まとめ：Pixel Visual Coreはガイドラインにどのように 沿っているか	305
7 領域特化アーキテクチャ	281	7.8 他の章との関連	306
7.1 はじめに	281	7.8.1 不均質性とSystem on a Chip (SoC)	306
7.2 DSAのガイドライン	282	7.8.2 オープンな命令セット	306
7.3 領域の例：深層ニューラルネットワーク	284	7.9 総合的な事例：CPU、GPU、DNNアクセラレータの比較	307
7.3.1 DNNのニューロン	284	7.9.1 性能：ルーフライン、応答時間、そしてスループット	308
7.3.2 学習と推論	285	7.9.2 価格性能比、TCO、ワット当たりの性能	310
7.3.3 多層パーセプトロン (MLP)	285	7.9.3 CatapultとPixel Visual Coreの評価	310
7.3.4 畳み込みニューラルネットワーク (CNN)	286	7.10 誤った考えと落とし穴	311
7.3.5 リカレントニューラルネットワーク (RNN)	287	7.11 おわりに	312
7.3.6 バッチ	288	7.12 歴史展望と参考文献	313
7.3.7 量子化	288	7.13 ケーススタディと演習問題	313
7.3.8 DNNのまとめ	288	ケーススタディ： GoogleのTensorプロセッシングユニットと深層学習ネット ワークのアクセラレーション	313
7.4 GoogleのTensorプロセッシングユニット：推論データセンターの加速器	289	演習問題	316
7.4.1 TPUの起源	289		
7.4.2 TPUアーキテクチャ	289		

付録A 命令セットの原理	319
A.1 はじめに	319
A.2 命令セットアーキテクチャの分類	320
A.2.1 まとめ：命令セットアーキテクチャの分類	322
A.3 メモリアドレッシング	322
A.3.1 メモリアドレスの解釈	322
A.3.2 アドレッシングモード	323
A.3.3 ディスプレースメント（ベース相対）アドレッシングモード	324
A.3.4 即値（リテラル）アドレッシングモード	325
A.3.5 まとめ：メモリアドレッシング	325
A.4 オペランドタイプとオペランドサイズ	325
A.5 命令セットにおける命令操作	326
A.6 制御のための命令	327
A.6.1 制御命令のためのアドレッシングモード	327
A.6.2 条件分岐における選択肢	328
A.6.3 手続き呼び出しにおける選択肢	328
A.6.4 まとめ：制御のための命令	329
A.7 命令セットのエンコード	329
A.7.1 RISCにおけるコードサイズの削減	330
A.7.2 まとめ：命令セットのエンコード	330
A.8 他の章との関連：コンパイラの役割	331
A.8.1 最近のコンパイラの構造	331
A.8.2 レジスタ割り付け	332
A.8.3 最適化が性能に及ぼす影響	332
A.8.4 コンパイラ技術がアーキテクトの意思決定に及ぼす影響	333
A.8.5 コンパイラ作成者へのアーキテクトの支援	333
A.8.6 マルチメディア命令へのコンパイラの支援（というより支援不足）	334
A.8.7 まとめ：コンパイラの役割	335
A.9 総合的な事例：RISC-Vアーキテクチャ	335
A.9.1 RISC-V命令セットの構成	335
A.9.2 RISC-Vのレジスタ	336
A.9.3 RISC-Vのデータタイプ	336
A.9.4 RISC-Vのデータ転送におけるアドレッシングモード	336
A.9.5 RISC-Vの命令フォーマット	336
A.9.6 RISC-Vのオペレーション	337
A.9.7 RISC-Vの制御命令	338
A.9.8 RISC-Vの浮動小数点演算	339
A.9.9 RISC-V命令セットの使用	340
A.10 誤った考えと落とし穴	340
A.11 おわりに	342
A.12 歴史展望と参考文献	342
A.13 演習問題	342
付録B メモリ階層の復習	347
B.1 はじめに	347
B.1.1 キャッシュ性能の復習	348

B.1.2 メモリ階層における4つの疑問	349
B.1.3 事例：Opteronのデータキャッシュ	352
B.2 キャッシュの性能	354
B.2.1 平均メモリアクセス時間とプロセッサ性能	355
B.2.2 アウトオブオーダー実行プロセッサのミスペナルティ	356
B.3 6つの基本的なキャッシュ改良法	358
B.3.1 基本的なキャッシュ改良法のまとめ	366
B.4 仮想メモリ	367
B.4.1 メモリ階層に対する4つの問いへの再訪	368
B.4.2 高速アドレス変換技術	369
B.4.3 ページサイズを選択	370
B.4.4 仮想メモリとキャッシュのまとめ	370
B.5 仮想メモリの保護とその例	371
B.5.1 プロセス保護	372
B.5.2 セグメント化仮想メモリの例：Intel Pentiumの保護	372
B.5.3 ページ化仮想メモリの例：64ビットOpteronメモリ管理	374
B.5.4 まとめ：32ビットIntel Pentiumと64ビットAMD Opteronの保護機構比較	376
B.6 誤った考えと落とし穴	376
B.7 おわりに	377
B.8 歴史展望と参考文献	377
B.9 演習問題	377

付録C パイプライン処理： 基本および中間的な概念	381
C.1 はじめに	381
C.1.1 パイプライン処理とは何か	381
C.1.2 RISC命令セットの基礎	382
C.1.3 RISC命令セットのシンプルな実装例	382
C.1.4 RISCプロセッサの古典的な5段パイプライン	383
C.1.5 パイプラインの基本性能	385
C.2 パイプライン処理の主要な障害： パイプラインハザード	385
C.2.1 ストール時のパイプライン性能	386
C.2.2 データハザード	386
C.2.3 分岐ハザード	389
C.2.4 予測による分岐コストの削減	391
C.2.5 静的分岐予測	391
C.2.6 動的分岐予測と分岐予測バッファ	392
C.3 パイプラインの実装法	393
C.3.1 RISC-Vの簡単な実装例	393
C.3.2 RISC-Vの基本パイプライン	395
C.3.3 RISC-Vパイプラインの制御信号の実装	397
C.3.4 パイプラインでの分岐の扱い	399
C.4 何がパイプラインの実装を困難にするのか	399
C.4.1 例外への対処	399
C.4.2 命令セットの複雑さ	402
C.5 複数サイクル演算を扱うためのRISC-V整数パイプライン	

ン拡張	403
C.5.1 レイテンシの長いパイプラインにおけるハザードと フォワーディング	405
C.5.2 正確な例外の維持	407
C.5.3 RISC-V FPパイプラインの性能	408
C.6 総合的な実例：MIPS R4000パイプライン	409
C.6.1 浮動小数点パイプライン	411
C.6.2 R4000パイプラインの性能	413
C.7 他の章との関連	413
C.7.1 RISC命令セットとパイプラインの効率	413
C.7.2 動的スケジューリングパイプライン	413
C.8 誤った考えと落とし穴	416
C.9 おわりに	416
C.10 歴史展望と参考文献	416
C.11 演習問題	417
参考文献	421
索引	441
その他の付録	447
コンピュータアーキテクチャの公式	447
経験則	447
本書で使われているCPUの用語とNVIDIAおよび OpenCL用語との対比	448
RV64G命令サブセット	449
訳者あとがき	450

訳者あとがき

2014年3月の『コンピュータアーキテクチャ 定量的アプローチ 第5版』の発刊から3年9か月後、2017年12月に、本書の原著である *Computer Architecture: A Quantitative Approach, Sixth Edition* が世に出た。これまで MIPS アーキテクチャをベースにしていたものから、方針を大転換し、ほとんどの章において **RISC-V** をベースプロセッサにして書き換わり、さらに「Domain-Specific Architectures (領域特化アーキテクチャ)」という新たな章が加わった。そしてなんと、その翌年、原著者の2人ヘネシーとパターソンが、コンピュータ科学における最高峰「チューリング賞、2017年度」を受賞してしまったのだ。さらにはヘネシーは Alphabet 社 (Google の親会社) の会長、パターソンは Google の Distinguished Engineer にも就いていた。これは心してかからねばならぬということで、第5版の翻訳メンバーの中から、3人で翻訳チームを組み、この第6版の翻訳に取り組むこととなった。この3人は以下の部分について分担して翻訳作業に当たった。

- 天野英晴：第1章、第2章、第3章、付録B、付録C
- 鈴木 貢：第4章、第7章
- 中條拓伯：第5章、第6章、付録A

チームを3人に絞ったのは、そのメンバーで御茶ノ水にあるコーヒーショップでざっと原著を眺めて、章・節構成などを前版と比較した結果、前述の RISC-V への転換と第7章のみの追加ということで、差分を取ればそれほど大した作業とはならないだろうと高を括ってしまった。しかしながら、その考えは甘かったことを後に痛感することとなった。

実は、第5版の翻訳を終え、編集の段階で手が加わり、校正作業の時に、その編集操作 (敢えてこう言わせていただく) において、元の翻訳とは微妙に異なる技術的に誤った記述が散見された。したがって、第5版の最終稿の中で、原著と変更のない部分をそのまま転用することは危険であり、技術的・学術的に誤りがないかをチェックしながら進めなければならなかった。さらに加えて、原著においてもデータを新たなものに刷新したこともあったからか、致命的な誤りが信じられないくらい多数見られたのだ。つまり、この翻訳作業は英語から日本語に変換するという作業とともに、疑心暗鬼になりながら、常に疑いの目を光らせて進めていくという作業に終始し、これは極めて精神的に堪える作業となったのである。

現在コンピュータ設計に従事する技術者、この分野を将来背負っていく若い世代に、早く本書を届けたいと思いつつ、学内や学会などにおいて、責任のかかる任務を遂行しつつ、翻訳・校正に取り

組む時間を確保するのが困難であったため、翻訳完了が予定より大幅に遅れてしまった。お叱りは覚悟しているが、上記の状況であったことをご理解いただきたい。

第5版と同様、技術的な点について原著の誤りを修正しつつ、翻訳ミスのできる限り排除する方向で翻訳、推敲、校正を重ね、ようやく完成に至ったが、まだ細かな誤りはあるかと思う。しかしながら、旬のうちに世に出すべく発刊を急ぐこととした。

本書を手取る技術者、大学生、大学院生、研究者は、この分野に深い造詣を有している方々だと思われ、些細な誤りについては前後の脈略から判断できる方々であると信じている。

原著は、2人のチューリング賞受賞者のみによるものではなく、世界中のコンピュータアーキテクチャ研究者、設計者らとともに、最新の大規模高性能計算機を設計・実装・活用している IT 系大企業からの情報をもとにまとめられたものであり、研究論文、技術報告書にも引用されることも少なくないと思われる。したがって、間違った記述には慎重になる必要があり、本書での学術的・技術的な誤りについては、誤植とともに、読者となる方々と今後も情報を共有していくべく、以下の翻訳版情報サイトを準備していただいた。

<http://www.am.ics.keio.ac.jp/wp/caqa6th>

本書に関する誤植情報、最新情報、時に議論などを掲載していく予定なので、ご覧いただくとともに、情報を提供いただけると幸いです。また、このサイト等を通じて、積極的に情報を発信することで、世の中の誤解を正すことができればと願っている。

前版の「訳者あとがき」にも述べられていたが、本書は2人の著者の名前から日本では「ヘネバタ」と呼ばれ親しまれてきた。そのヘネバタの初期の頃は、「コンピュータをやさしく解説してくれる名著である」とか、「原文は分かりやすいので、直接当たった方がよい」と囁かれていた。さらに、「第3版以降は、コンピュータのカタログに墮してしまっている」といった誤解が世の中にはびこっていた。これらは、この第6版においもて、やはり全く当てはまらない。ヘネバタは他の工学系の専門書と同様、高度で難解な専門書であり、読んですら理解できるような代物ではないことは第6版でも同じである。

ヘネバタが名著である所以は、この版においても、膨大で徹底的な定量的評価をまとめた情報量によるとともに、アーキテクチャに関連する分野・技術に関して、その核となる技術とともに、それを支える周辺技術についても、しっかりと網羅されている点にあ

る。また、本書の最初にある謝辞に記載されている査読者、委員会にある名前をご覧いただきたい。ACM、IEEEなどの難関な国際会議で招待講演、基調講演を担うような著名な方々の名前にお気づきいただけると思う。それとともに、旧版を含めての査読者の数は驚愕に値する。それでもなお、原著には見過ごされている誤りが多数残されていた。

本書の翻訳を批評したいのであれば、最低限原著と照らし合わせた上で行っていただきたく、世のサイトの匿名の口コミを読んだだけでそれを鵜呑みすることは避けていただきたい。

本書において理解しがたい部分があれば、翻訳のせいであると即断する前に、ぜひ上記の翻訳版情報サイトを訪れていただきたい。そして貴重なご意見をいただき、正誤表に記載するような貢献をいただければ、ご了解のうえ、翻訳版情報サイトとともに、本書の電子版に修正を反映させ、協力者の名前を記載させていただく予定である。

我々は本書の真価を世に問うべく、これからも世の中の誤解を正す使命をもとに時間の許す限り情報を発信し、電子版を進化させていき、恐らくこの第6版が最後となるであろう「ヘネパタ」を、誤りの無い、完璧なものになるよう進化させ、完成版を残したい。

中條拓伯

謝 辞

本書の完成にあたり、第5版における翻訳部分を流用させていただいた前版以前の翻訳メンバーであった福岡大学佐藤寿倫教授、東京工業大学吉瀬謙二准教授に感謝いたします。また、以下のようにさまざまな方々から助けをいただいた（敬称略）。

天野英晴：

各章を精査していただいた慶應義塾大学理工学部の以下の学生諸君に心から感謝します。

河野隆太、弘中和衛、高野茂幸、小島拓也、飯塚健介、池添赳治、丹羽直也、戸村遼平、伊藤光平、大和田彩夏、四釜快弥、清水智貴、寺岡朋弘。

中條拓伯：

本書校正に貢献いただいた以下の方々に感謝いたします。

山形大学大学院理工学研究科多田十兵衛助教、同大学工学部東良輔。

東京農工大学工学部中條研究室：識名朝彬、西川 凜、山下遼太、中野道彦、新村研人。

●訳者紹介

中條拓伯（なかじょう ひろのり）

1987年、神戸大学工学研究科電子工学専攻修了、博士（工学）

1989年より神戸大学工学部システム工学科に勤務

1998年よりIllinois大学Center for Supercomputing Research and Development（CSRD）にて、Visiting Research Assistant Professor

1999年、東京農工大学工学部に赴任

現在、東京農工大学大学院共生科学技術研究院准教授

プロセッサアーキテクチャ、FPGAを用いた高性能計算機システムの研究に従事。

鈴木 貢（すずき みつぐ）

1995年、電気通信大学電気通信学研究科情報工学専攻博士後期課程単位取得満期退学

1995年より電気通信大学電気通信学部に勤務

現在、島根大学総合理工学部知能情報デザイン学科准教授、博士（工学）

特殊命令セット向けコンパイラ最適化と、オープンソース活用向け工学教育に興味を持つ。

天野英晴（あまの ひではる）

1986年、慶應義塾大学工学研究科電気工学専攻修了、工学博士

1985年より慶應義塾大学工学部に勤務

1889年より1990年までStanford大学CSLのVisiting Assistant Professor

現在、慶應義塾大学理工学部情報工学科教授

並列計算機アーキテクチャ、リコンフィギャラブルシステムの研究に従事。

●カバー：原著カバー（Christian J. Bilbowデザイン）をElsevier社より許可を得て、再利用させていただきました。

ヘネシー ■ パターソン

コンピュータアーキテクチャ

定量的アプローチ [第6版]

2019年9月25日 初版第1刷発行

著者 ジョン・L・ヘネシー、デイビッド・A・パターソン

訳者 中條拓伯、天野英晴、鈴木 貢

発行人 富澤 昇

発行所 株式会社エスアイビー・アクセス (<http://www.sibaccess.co.jp>)
〒183-0015 東京都府中市清水が丘3-7-15
TEL: 042-334-6780 / FAX: 042-352-7191 / e-メール: sib-tom@hh.ij4u.or.jp発売所 株式会社星雲社
〒112-0005 東京都文京区水道1-3-30
TEL: 03-3868-3275 / FAX: 03-3868-6588

印刷製本 デジタル・オンデマンド出版センター

This edition of *Computer Architecture: A Quantitative Approach, 6e* by John L. Hennessy, David A. Patterson (978-0128119051) is published by arrangement with Elsevier Inc.

Copyright © 2019 Elsevier Inc. All rights reserved.

Disclaimer: The translation has been undertaken by SIB Access Co. Ltd. at its sole responsibility. Practitioners and researchers must always rely on their own experience and knowledge in evaluating and using any information, methods, compounds or experiments described herein. Because of rapid advances in the medical sciences, in particular, independent verification of diagnoses and drug dosages should be made. To the fullest extent of the law, no responsibility is assumed by Elsevier, authors, editors or contributors in relation to the translation or for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions, or ideas contained in the material herein.

JAPANESE language edition published by SIBaccess Co. Ltd., Copyright © 2019.

ISBN 978-4-434-26400-9

Printed in Japan

●落丁・乱丁本はお取り替えいたします。

●本書の内容に関するご質問は（株）エスアイビー・アクセスまでe-メール、ファックスまたは封書にてお寄せください（電話によるお問い合わせはご容赦ください）。また、本書の範囲を越えるご質問等につきましてはお答えできかねる場合もあります。あらかじめご承知おきください。