HZ BOOKS
华章教育

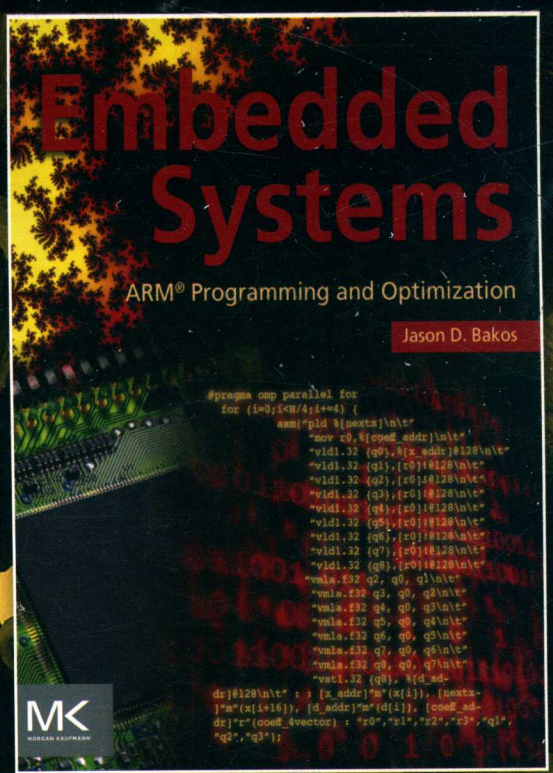
计 算 机 科 学 丛 书

ELSEVIER
爱思唯尔

ARM嵌入式系统编程与优化

[美] 詹森 D. 巴克斯（Jason D. Bakos）著
梁元宇 译

Embedded Systems
ARM Programming and Optimization



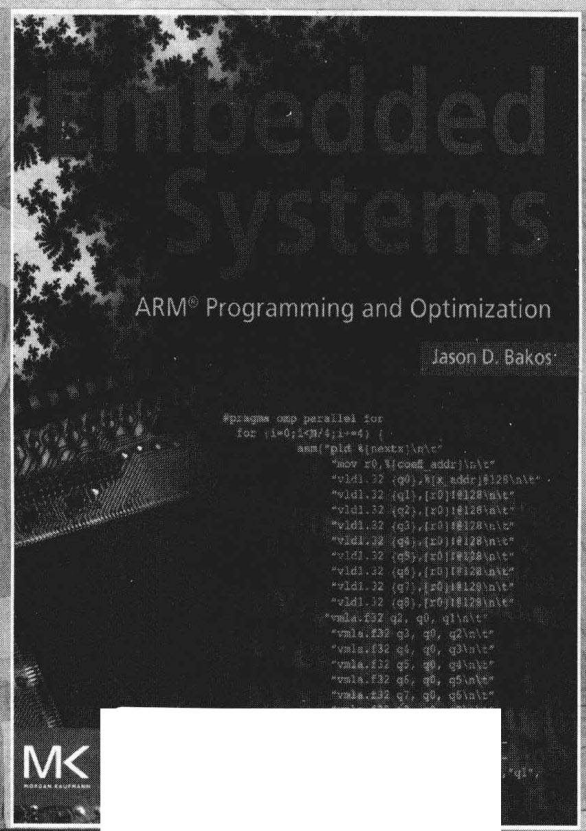
机械工业出版社
China Machine Press

计 算 机 科 学 丛 书

ARM嵌入式系统编程与优化

[美] 詹森 D. 巴克斯（Jason D. Bakos）著
梁元宇 译

Embedded Systems
ARM Programming and Optimization



机械工业出版社
China Machine Press

图书在版编目（CIP）数据

ARM 嵌入式系统编程与优化 /（美）詹森 D. 巴克斯（Jason D. Bakos）著；梁元宇译. —北京：机械工业出版社，2017.8
（计算机科学丛书）
书名原文：Embedded Systems: ARM Programming and Optimization

ISBN 978-7-111-57803-1

I. A… II. ①詹… ②梁… III. 微处理器－系统设计 IV. TP332

中国版本图书馆 CIP 数据核字（2017）第 208411 号

本书版权登记号：图字：01-2016-4746

Embedded Systems: ARM Programming and Optimization
Jason D. Bakos
ISBN: 978-0-12-800342-8

本书结合 ARM 架构和 Linux 工具，关注以性能为导向的嵌入式编程，深入讲解如何通过对数据、算法和存储等层面的优化，最终实现性能的显著提升。本书先讲解 ARM 架构和嵌入式系统的基础知识，然后结合图像变换、分形生成和计算机视觉等应用案例，详细说明不同的优化方法。读者可在 Raspberry Pi 等平台上动手运行并比较不同算法，掌握实践技巧。

本书适合作为本科或研究生嵌入式系统课程的教材，也适合从事相关开发工作的程序员参考。

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码：100037）
责任编辑：吴晋瑜　　责任校对：李秋荣
印　　刷：北京瑞德印刷有限公司　　版　　次：2017 年 9 月第 1 版第 1 次印刷
开　　本：185mm×260mm　1/16　　印　　张：13.5
书　　号：ISBN 978-7-111-57803-1　　定　　价：59.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991　88361066　　投稿热线：（010）88379604
购书热线：（010）68326294　88379649　68995259　　读者信箱：hzjsj@hzbook.com

文艺复兴以来，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的优势，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭示了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短的现状下，美国等发达国家在其计算机科学发展的几十年间积淀和发展的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起到积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章公司较早意识到“出版要为教育服务”。自 1998 年开始，我们就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与 Pearson，McGraw-Hill，Elsevier，MIT，John Wiley & Sons，Cengage 等世界著名出版公司建立了良好的合作关系，从他们现有的数百种教材中甄选出 Andrew S. Tanenbaum，Bjarne Stroustrup，Brian W. Kernighan，Dennis Ritchie，Jim Gray，Afred V. Aho，John E. Hopcroft，Jeffrey D. Ullman，Abraham Silberschatz，William Stallings，Donald E. Knuth，John L. Hennessy，Larry L. Peterson 等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及珍藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力相助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专门为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近两百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍。其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证。随着计算机科学与技术专业学科建设的不断完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都将步入一个新的阶段，我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

华章网站：www.hzbook.com
电子邮件：hzjsj@hzbook.com
联系电话：（010）88379604
联系地址：北京市西城区百万庄南街 1 号
邮政编码：100037

华章科技图书出版中心

译者序

Embedded Systems: ARM Programming and Optimization

这是我独立翻译的第一本书。虽然我对 Linux 和 ARM 的了解并不是那么精深，但想来这样一本 300 多页的介绍主流 ARM 嵌入式系统的书，翻译起来应该没多大难度。很遗憾，随着翻译工作的进行，我发现自己的技术储备并不充足。

虽然本书篇幅不长，但介绍了非常多的内容。本书的大部分章节都在介绍 ARM 技术及主流的优化方法，作者对各种知识点的介绍都很简洁，有些地方只是点到为止，甚至只是给出一些基本思路。读者不必因为不明白其中的某些细节而感到沮丧，把本书看作一个学习 ARM 嵌入式系统的快速通道就好。在此基础上，对于感兴趣的内容，自己再进一步深入学习。

虽然翻译过程没有想象的那样顺利，但我本人还是收获颇丰。收获之一便是对 ARM 嵌入式系统有了更多的了解。另一个收获是，我发现本书还涉及异构并行等前沿概念，满足了我学习新事物的好奇心。

目前该领域相关的图书资料还相当缺乏，本书作者显然想在第一时间和大家分享他的知识，因此本书难免存在不足之处，有些地方尚有一定的改进空间。在翻译过程中，我已经修正了多处比较明显的小错误，但因为自身水平和精力有限，所以可能会在翻译过程中引入新的错误，恳请广大读者不吝赐教。

最后要感谢华章公司的编辑在审校等环节的辛勤付出。

梁元宇

2017 年 7 月于南京

多年来，我一直工作在可重构计算领域。可重构计算领域的目标是开发有效的工具和方法，以促进现场可编程门阵列（FPGA）作为协处理器在高性能计算机系统中的使用。

这个学科的主要挑战之一是“程序设计问题”，即 FPGA 的实际应用从根本上受到烦琐和容易出错的程序模型的限制。这个问题值得我们特别关注，因为它是技术优势所导致的结果：FPGA 实现了细粒度并发操作，这样程序员可以控制芯片中每个电路的同步行为。然而，这种控制还要求程序员管理细粒度的控制，例如片上存储使用和路由拥塞。另一方面，CPU 程序只需要考虑每一行代码的可能 CPU 状态，片上资源在硬件运行时将自动管理。

最近我意识到，现代嵌入式系统可能很快就会面临类似的程序设计问题。电池技术仍然相对滞后，并且在用近 6 年时间实现了从 65nm 到 28nm 的制造工艺后，摩尔定律的发展速度开始明显减缓。与此同时，消费者已经开始期待嵌入式系统功能的不断进步，例如能够在一副眼镜上的处理器中运行实时增强现实（AR）软件。

鉴于这些能源效率和性能的要求，许多嵌入式处理器厂商正在为微体系结构寻求更节能的方法，并经常涉及对并行类型的选择，而这一类型是不能从软件中自动提取的。这就需要程序员协助编写并行代码。这带来了很多问题：程序员要在资源和能量均有限的平台上兼顾功能和性能，要知道，在这个平台上可能包括从多核到 GPU 着色器单元等各种并行资源。

许多大学已经开展了“统一”的并行编程课程，这些课程涵盖了从分布式系统到多核处理器的并行编程系列。然而，教授这类主题的角度通常是高性能计算而非嵌入式计算。

随着最近 Raspberry Pi 等先进嵌入式平台的爆发，我意识到需要开发针对嵌入式系统性能的编程课程，这些课程应涵盖从计算机体系结构到并行编程的相关主题。我也想纳入一些有趣的相关项目和课程的案例研究，这样可以避开枯燥的传统嵌入式系统课程项目（例如闪烁的 LED）和并行编程课程（例如编写和优化快速傅里叶变换）。

在自己的嵌入式系统课程中使用这些想法时，我经常发现学生们会争相实现最快的图像旋转或最快的曼德布罗特集合生成器。这种竞争也激发了学生的学习热情。

如何使用本书

本书面向初级或高级本科计算机科学或计算机工程课程。虽然嵌入式系统课程可能关注控制理论、机器人技术、低功耗设计、实时系统或其他相关的主题，但本书旨在介绍轻量级片上系统嵌入式处理器上的以性能为导向的编程。

本书应该结合 Raspberry Pi 等嵌入式设计平台一起使用，这样学生可以评估书中所述的实践和方法。

在使用本书时，学生应该预先学习 C 编程语言和 Linux 操作系统的基本知识，并了解诸如任务同步等基本的并发。

教辅支持[⊖]

可访问网站 booksite.elsevier.com/9780128003428 查看本书的幻灯片、习题答案和勘误表。

⊖ 关于本书教辅资源，只有使用本书作为教材的教师才可以申请，需要的教师访问爱思唯尔的教材网站 https://textbooks.elsevier.com/ 进行申请。——编辑注

致　谢

Embedded Systems: ARM Programming and Optimization

感谢帮助我完成本书的几位学生。

2013 年春季和夏季，本科生 Benjamin Morgan、Jonathan Kilby、Shawn Weaver、Justin Robinson 以及 Amadeo Bellotti 评估了 Raspberry Pi Broadcom BCM2835 和 Xilinx Zynq 7020 上的 DMA 控制器和性能监控单元。

2014 年夏季，本科生 Daniel Clements 帮助我开发了在 ARM11、ARM Cortex A9 和 ARM Cortex A15 上使用 Linux perf_event 的统一方法。Daniel 还评估了图像技术的 OpenCL 运行时，以及描述了在 ODROID XU Exynos 5 平台上的 PowerVR 544 GPU 的性能特点。

2015 年夏季，本科生 Friel“Scottie”Scott 帮助我评估了 ODROID XU3 平台上的 Mali T628 GPU，并且校对了第 5 章的内容。

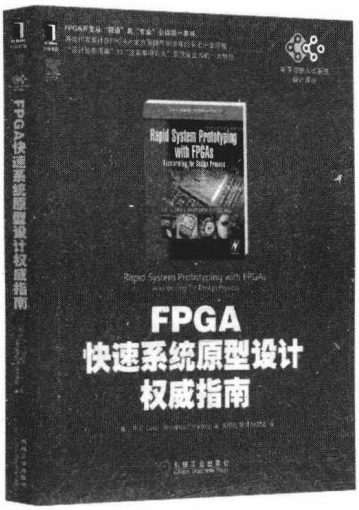
许多关于计算机视觉算法存储优化的见解来自我的研究生 Fan Zhang 的关于德州仪器关键数字信号处理器架构的自动优化模板循环的论文。

感谢以下评论者，他们在本书的编写过程中提供了反馈、见解以及有用的建议：

- Miriam Leeser，美国东北大学
- Larry D. Pyeatt，美国南达科他矿业理工学院
- Andrew N. Sloss，美国华盛顿大学，同时在 ARM 公司做顾问工程师
- Amr Zaky，美国圣塔克拉拉大学

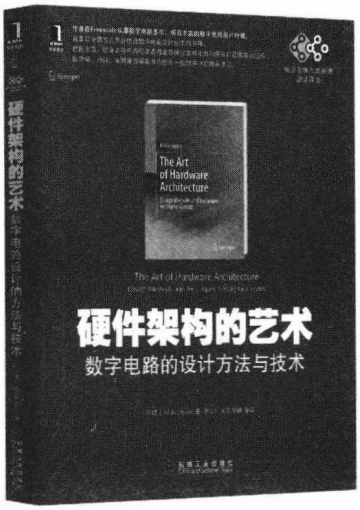
感谢 Morgan Kaufmann 出版公司，感谢 Nate McFadden 在整个写作过程中给予我的不断鼓励和无限耐心。特别感谢 Nate 对于本书内容所持的开放和灵活的态度，这使我在写作时能够不断跟进新发布的基于 ARM 的嵌入式开发平台。也要感谢 Sujatha Thirugnana Sambandam 的细心编辑，还要感谢 Mark Rogers 为本书设计封面。

推荐阅读



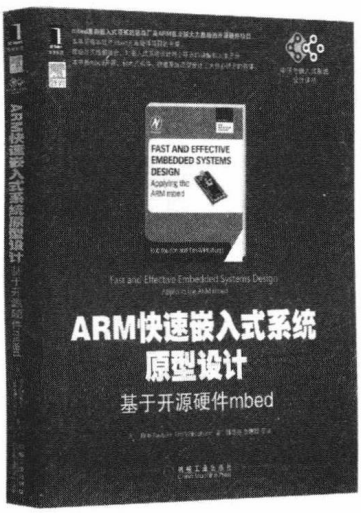
FPGA快速系统原型设计权威指南

作者：R.C. Cofer 等 ISBN：978-7-111-44851-8 定价：69.00元



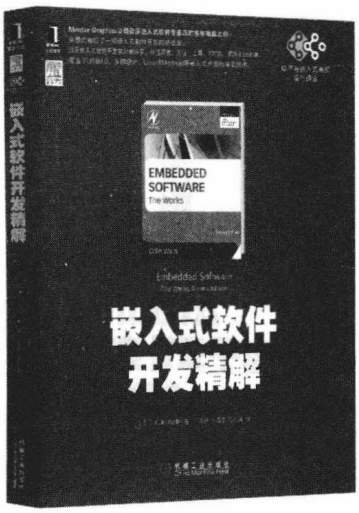
硬件架构的艺术：数字电路的设计方法与技术

作者：Mohit Arora ISBN：978-7-111-44939-3 定价：59.00元



ARM快速嵌入式系统原型设计：基于开源硬件mbed

作者：Rob Toulson 等 ISBN：978-7-111-46019-0 定价：69.00元



嵌入式软件开发精解

作者：Colin Walls ISBN：978-7-111-44952-2 定价：79.00元

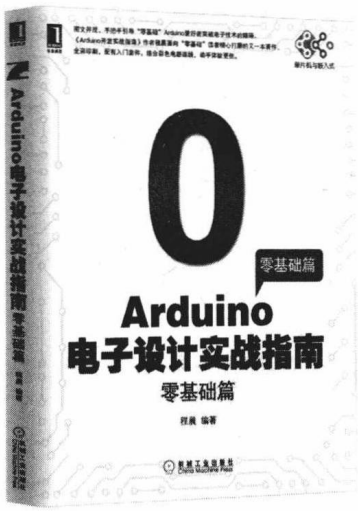
推荐阅读

嵌入式系统软硬件协同设计实战指南：基于Xilinx Zynq

作者：陆佳华 等 ISBN：978-7-111-41107-9 定价：69.00元

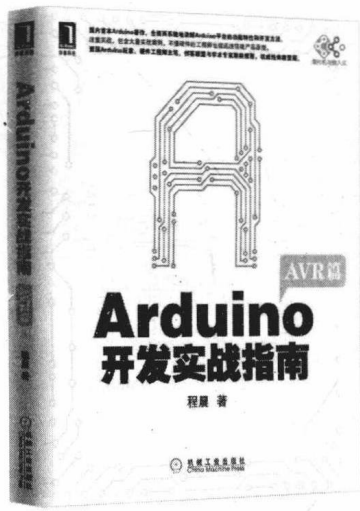
兼容ARM9的软核处理器设计：基于FPGA

作者：李新兵 ISBN：978-7-111-37572-2 定价：69.00元



Arduino电子设计实战指南：零基础篇

作者：程晨 ISBN：978-7-111-41717-0 定价：59.00元



Arduino开发实战指南：AVR篇

作者：程晨 ISBN：978-7-111-37005-5 定价：59.00元

第 1 章

Embedded Systems: ARM Programming and Optimization

Linux/ARM 嵌入式平台

可以传输大量多媒体、具有高速无线宽带连通性和高分辨遥感和信号处理的移动设备和可穿戴设备对我们的文化的影响日益深远。这些设备的出现归功于两项关键技术。

第一项技术是在无形中影响着所有设备的 ARM 处理器技术。ARM 处理器在 20 世纪 80 年代的消费性电子产品中开始使用，之后逐渐成为实际的嵌入式处理器技术。

与台式机和服务器处理器不同，ARM 处理器不是独立芯片产品，而是具有多样性和异质性嵌入式片上系统（*SOC*）的集成部件，是可以针对各种特定产品实现定制化的部件。

嵌入式片上系统是一组在单片上相互连接的硬件模块。典型的片上系统包括一个或者多个 ARM 处理器内核。这些处理器充当整个片上系统的“主人”或者中央控制器，它们对用户交互、外围设备的持续控制以及专用协处理器负责。

除了处理器内核外，典型的片上系统还包含一系列不同类型的存储接口（例如同步动态随机存储器、闪存等）、通信接口（USB、蓝牙、WiFi 等），以及图形、视频等专用处理器（例如图形处理单元）。

图 1-1 所示的是苹果 A8 芯片，它是苹果 iPhone 6 的处理器。其中有一块称为“CPU”的区域，它包含一个双核 ARM 处理器；还有一块称为“GPU”的区域，它包含一组用于视频和图形的专用处理器。图中大块未标记区域包含其他模块，它们常用于各种支持 iPhone 功能的外围设备。

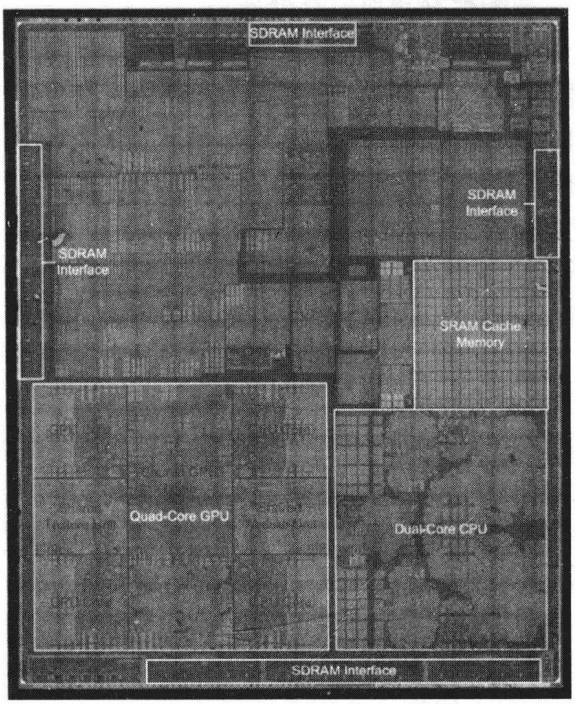


图 1-1　苹果 A8 芯片包内的处理器的集成电路。它有两个 ARM 处理器内核、4 个 PowerVR G6450 内核、4 个 DRAM 接口（连接片外存储器）、一个 LCD 接口、一个摄像头接口和一个 USB 接口（图像来自 *ChipWorks* 网站）

从执行用任何编程语言编写的任意程序的能力这种意义上来说，ARM 处理器包含了片上系统的“计算”部分。然而，根据现代台式机和服务器的制定标准，ARM 处理器是低效率处理器，因为相对于台式机和服务器处理器，它们的设计更注重高能效性。

催生现代消费性电子产品的第二项技术是 Linux 操作系统（OS）——几乎所有的 ARM 处理器都运行它。由于免费可用，Linux 被成功且广泛用于台式机和服务器，并且成为一个通用标准的嵌入式操作系统。作为一个嵌入式操作系统，Linux 大大促进了不同嵌入式平台间的代码开发和代码重用。

1.1 以性能为导向的编程

一般情况下，嵌入式处理器与台式机和服务器处理器有着不同的设计目标。*移动设备嵌入式处理器特别强调能效目标高于其他所有目标，并且强调高效执行代码，这是因为和为台式机和服务器处理器编写代码相比，嵌入式系统一般在编程方面需要更多的工作。*

因为，台式机和服务器处理器的设计是从两个方面获得最大性能：其一是*遗留代码*，即很久以前处理器早期版本中编写和编译的代码；其二是*与处理器无关的代码*，即不是为任何特定目标处理器编写的代码。对于大量的实际遗留代码和能耗问题，可通过应用以下两点获取性能：

- **指令级并行提取**。处理器试图：在每个时钟周期执行尽可能多的指令；改变指令执行顺序来减少相关指令之间的等待时间；在知道它们是否被需要以及必要时抛弃不必要的结果之前预测执行行为和执行指令。这允许处理器实现最大化指令执行速率，即便在任何特定方式下程序代码中的指令处于无序状态。
- **大而复杂的缓存**。处理器试图最大化存储系统性能，即使程序访问附加内存这些非理想存储。这涉及智能数据预取、存储器访问调度、高关联性，以及为避免重复访问相同存储而允许缓存等技术。

但是，嵌入式处理器通常放弃大多数这些特性，因为它们会使得嵌入式处理器在程序有无“性能优化”之间具有更大的性能差距。

牢记一点，虽然这种差异仅在*受到计算量限制*的程序中明显，这些程序的速度或响应时间是由处理器或存储速度决定的，而不是由输入和输出决定。例如，嵌入式处理器压缩一段视频明显比服务器处理器需要更多的时间，因为等待时间是由处理器和存储速度决定的。

另一方面，*受到 I/O 限制*的程序的性能是由通信信道或其他外设的速度决定的。例如，一个程序强迫用户等待下载的数据，无论处理器技术如何，它也不会更快。在这种情况下，响应时间仅由设备完成下载有多快决定。

那么，有多少移动设备嵌入式程序受到计算量限制，而不是 I/O 限制？大多数图像和视频编码受到计算量限制，但因为这些通常是基于变化很少的标准。大多数的片上系统厂商是“骗子”——通过把这些任务交给专用硬件而不是在 ARM 处理器内核运行的软件来处理。

然而，诸如计算机视觉等下一代嵌入式应用将会用到大量进化算法，这些算法大部分是受到计算量限制的。这种例子包括一些将全景图像中的独立图像连接到一起、人脸检测和识别、增强现实和目标识别等程序。

本书针对一些影响处理器性能的程序设计方法给出了一个总体概览，即程序员可以仅对代码做改变而不改变程序语法来改进代码性能。

这些技术通常需要程序员以暴露底层微体系结构中的特性这种方式编写代码。换言之，

程序员必须编写低抽象水平的代码，只有这样，程序才能知道是底层处理器技术。这通常被称为性能优化或代码优化。

这些思想并不新颖，许多技术在高性能计算领域中是很常见的。不过，本书将介绍如何在嵌入式处理器特别是 ARM 处理器中使用这些想法，也将介绍深入了解计算机体系结构、应用程序设计和嵌入式系统，以及在为现代嵌入式系统设计的嵌入式软件领域中得以实践的知识。

在讲述这些方法论时，本书将用到几个应用程序例子，其中包括图像变换、分形生成、图像卷积等，以及一些计算机视觉任务。这些例子将在下面三个 ARM 内核上使用 ARMv6 和 ARMv7-A 指令集架构：

- ARM11，在 Raspberry Pi 1 中使用。
- Cortex-A9，在 Xilinx Zynq 中使用（在 Avnet 公司的 Zedboard、Adapteva 公司的 Parallela 和 National Instruments 公司的 myRIO 设备中用到）。
- Cortex-A15，在 NVIDIA 公司的 Tegra K1 中使用。

本书还介绍了使用 OpenCL 编程模型编写移动通用图形处理单元（GPGPU）的方法。

本书将充分利用 Linux 操作系统提供的系统工具，包括 Linux GCC 编译工具链和调试工具、性能监控支持、OpenMP 多核运行环境、视频帧缓冲及视频捕获功能。

本书是配合当前市场上许多低成本的 Linux/ARM 嵌入式开发板而设计的。具体包括以下几类例子：

- 价值 35 美元的带有 700MHz ARM11 处理器的 Raspberry Pi，以及有双核 Cortex-A9 处理器的新一代的 Raspberry Pi 2。
- 价值 65 美元的带有 1.7GHz 四核 ARM Cortex A9 处理器的 ODROID-U3。
- 价值 90 美元的带有 1GHz ARM Cortex A8 处理器的 BeagleBone Black。
- 价值 99 美元的带有 1GHz 双核 ARM Cortex A9 处理器的 Parallella 平台。
- 价值 169 美元的带有 1.6GHz 四核 ARM Cortex A15 处理器的 ODROID-XU3。
- 价值 182 美元的带有 1.2GHz 双核 ARM Cortex A9 处理器的 PandaBoard。
- 价值 192 美元的带有 2.23GHz 四核 ARM Cortex A15 处理器的 NVIDIA Jetson TK1。
- 价值 199 美元的带有 1.8GHz 四核 ARM Cortex A15 处理器的 Arndale Octa。
- 价值 199 美元的带有 666MHz 双核 ARM Cortex A9 处理器的 Avnet MicroZed。

这些平台允许程序员使用一个交互式登录会话来编译、调试和描述代码，省去了对复杂的交叉编译器、远程调试器和体系结构模拟器的需求。

1.2 ARM 技术

ARM 处理器技术是由 ARM Holdings 公司掌控的。ARM 代表“高级精简指令集机器”，RISC 代表“精简指令集计算机”。RISC 是一门设计哲学，它将处理器原生语言或者说指令集设计成极简指令表，这就要求处理器通过执行大量简单指令来执行一个程序。这种方法的优势是：相对于处理器具有独立的指令来执行更多工作，即使一个程序需要执行 N 次更简单指令，简单指令的执行速度也比复杂指令平均快 N 倍，并提供了更好的整体性能。

RISC 指令一般严格分为三种主要类型：*算术运算指令*、*存储指令*和*控制指令*。算术运算指令是执行任何实际数学计算的唯一指令类型，而存储指令和控制指令需要必要开销来实现与外部存储器交换数据以及实现有数据依赖的行为。存储器指令和控制指令平均会比算术

运算指令花更多的时间。虽然取决于程序的存储访问模式和处理器存储层次结构的性能，但是，特别的是存储指令，算术运算指令的执行速度一般会比它快 10 ~ 20 倍。

1.3 ARM 简史

ARM 指令集架构最初是20世纪80年代英国公司 Acorn Computers 为其 ARM1、ARM2 和 ARM3 中央处理器开发的。这些中央处理器被用作台式个人计算机的中央处理器，但在当时败给了竞争对手英特尔的 x86 和摩托罗拉的 68 000 处理器后，ARM 改变了它的商业模式，从销售中央处理器转为销售其处理器设计和指令集架构的使用权。他们的第一个大客户是苹果公司——苹果公司在其牛顿掌上电脑（Newton PDA）上使用 ARM 处理器。

目前 ARM 处理器作为一个可重用的宏单元出售，这是作为片上系统的模块使用的一种预制设计。因此，ARM 宏单元可以插入其他宏单元的现有设计中，形成一个定制的异构片上系统。另外，ARM 指令集架构的特点之一就是可以授权，允许从 0 开始设计实现技术。而且，一致指令集架构的广泛使用允许程序员利用成熟的前端开发工具，例如编译器、调试器以及代码库。ARM 指令集架构和宏单元都有几个不同的版本，但它们都不能独立提供客户所期望的丰富的多媒体功能，所以它们几乎总是与专用协处理器结合起来代表中央处理器完成大部分的多媒体算法。

ARM 指令集架构是在不断发展的。在 ARM 指令集架构第 6 版（ARMv6）发布后，ARM 指令集架构分成 3 个不同的版本，各自针对特定的应用进行优化。目前，用于优化微控制器（ARMv6-M、ARMv7-M 和 ARMv7EM）的指令集架构有 3 种：一个为实时应用（ARMv7-R）的优化，两个为通用应用软件（ARMv7-A 和 ARMv8-A）的优化。

本书着重介绍 ARMv 6 和 ARMv 7 架构。ARMv 6 常应用于 Raspberry Pi 的 ARM 11 处理器，而 ARMv 7 目前常应用于智能手机和平板电脑等大多数现代嵌入式设备中。ARMv 6 和 ARMv 7 非常相似，二者最显著的差异也许是在 ARMv 7 中添加了 NEON 指令。关于 NEON 指令的内容会在本章后面进行介绍。

ARMv 8 架构于 2013 年引入，其中包括与 v6 和 v7 版本的几个基本差异。这些差异包括寄存器文件结构的变化、重要指令的增删以及条件执行字段的废弃。

本书并不是要读者彻底学习 ARM 汇编语言或者 ARM 微架构。但是，为了有助于理解并提高代码性能，对由编译器生成的汇编代码进行解释通常是很有必要的。在许多情况下，还必须用汇编语言编写代码片段，以描述一个比编译器更高效的特定操作。注意：这种手写的汇编语言可以嵌入一个高级语言编写的程序中。

1.4 ARM 编程

可以用各种高级编程语言编写 ARM 处理器。一些 ARM 处理器本身还可以执行 Java 字节码。话虽如此，但因为本书主要关注的是代码性能，所以采用 C 语言编程。

GCC 和 Clang 是两个最流行的开源 C/C++ 的编译工具链，它们包括 ARM 处理器的后端，支持完整的 C/C++ 开发、库支持以及调试。ARM Holdings、Keil 公司以及德州仪器为 ARM 提供商业编译工具链。商业编译器可以产生比开源编译器更快的目标码，但是为了忠于 Linux，本书使用 Linux 的官方编译器 GCC 来描述高级代码的特征。

像 ARM 这样的 RISC 架构起初是用一个小而简单的指令集设计的。这使得编译器能够有效地利用可用指令。然而，大多数现代指令集架构包括那些来自 ARM 和英特尔的 ISA 都

增加了诸如多媒体、数字信号处理等复杂指令。许多这些指令允许单一指令处理多个输入（这就是所谓的单指令多数据流即 SIMD 指令）。现在的编译器面临一个困难，即如何在程序员不参与的情况下有效地使用这些指令。

一般来说，利用这些指令（并获得由此产生的性能提升）要求程序员使用内联汇编或者内联函数。内联函数是用于解析编译时具体指令的函数。内联函数比汇编语言更易使用，但一些优化技术还是需要用到汇编语言。

1.5 ARM 体系集架构

ARM 是一种“加载 / 存储（load-store）”架构。这意味着程序员必须在数据被处理前显式地把输入数据从存储器加载（读）到寄存器。同样，程序员必须在数据已被处理后显式地把输出数据存储到存储器。所有算术运算指令以寄存器内容作为输入和结果。寄存器也可以用来存储临时或中间结果，例如循环计数器或子表达式值。程序员（或使用高级语言的编译器）有对寄存器状态的完全控制权。例如当两个值相加时，程序必须决定临时分配哪些寄存器给每个值和计算之和。在之前的内容不再需要时，寄存器可以任意重用。

1.5.1 ARM 通用寄存器

ARM 是一个三地址架构，意思是一个单指令可以涉及三个寄存器地址。例如，算术运算指令中的“ add”可以指定两个寄存器，其中一个用于读取输入值，另一个用于存储所计算的和。在使用 GCC 汇编时，目标寄存器会列在第一位。

ARM 中有 16 个用户可访问的整数寄存器，它们被命名为 r0 ~ r15。除了以下两种特殊情况外，完全用汇编语言编写的程序可以自由地使用任一寄存器，而不从硬件推导：

- 寄存器 r14 称为链接寄存器（也称为 lr 或 LR），它的值是在执行一个“分支并链接”指令时由硬件更新。
- 寄存器 r15 称为程序计数器（也称为 pc 或 PC），它的值是由硬件维护，常用于控制程序流程。

通常在编写一个汇编语言程序——尤其是在嵌入或调用 C 代码时，程序员在使用某些有特殊意义且被 ARM 过程调用标准（APCS）规定的寄存器时要格外小心。小心是必要的，因为这些寄存器可以由编译器生成的代码或其他程序员编写的代码随意改变。

这些寄存器的组成如下：

- 寄存器 r0 和 r1 用于函数返回值。与 MIPS 指令集不同，寄存器 r0 不是“固定”包含零值。
- 寄存器 r0 ~ r3（相对于参数寄存器，也可以称为 a1 ~ a4）常用于向函数传递参数。程序员可以自由地使用这些寄存器作为“暂存寄存器”，但应该知道它们的状态在不同函数调用中可能不会被保存。
- 寄存器 r4 ~ r11（相对于可变寄存器，也可以称为 v1 ~ v8）通常可以安全使用，除了一些不起眼的编译器用 r9 作为静态库寄存器（也称为 sb、SB 或 v6），以及 r10 作为栈限制寄存器（也称为 sl、SL 或 v7）。
- GCC 这类编译器用 r11 作为帧指针（也称为 fp、FP 或 v8），指向当前激活帧的基址。激活帧包含诸如局部变量的本地子例程数据。
- 寄存器 r13 称为栈指针（也称为 sp 或 SP），常用来指向激活栈的栈顶。