HZ BOOKS
华章教育

PEARSON

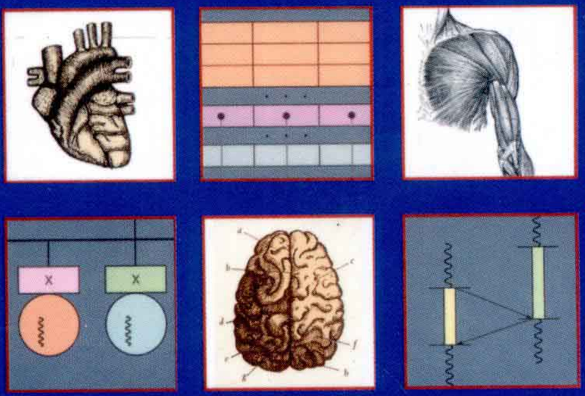
经 典 原 版 书 库

计算机系统
集成方法

（英文版）

COMPUTER SYSTEMS

An Integrated Approach to Architecture and Operating Systems



Umakishore Ramachandran
（美） William D. Leahy, Jr. 著
佐治亚理工学院

机械工业出版社
China Machine Press

经 典 原 版 书 库

计算机系统

集成方法

（英文版）

Computer Systems

An Integrated Approach to Architecture and Operating Systems

常州大学图书馆
藏 书 章

Umakishore Ramachandran
（美） William D. Leahy, Jr. 著
佐治亚理工学院

机 械 工 业 出 版 社
China Machine Press

English reprint edition copyright © 2010 by Pearson Education Asia Limited and China Machine Press.

Original English language title: *Computer Systems: An Integrated Approach to Architecture and Operating Systems* (ISBN 978-0-321-48613-4) by Umakishore Ramachandran and William D. Leahy, Jr., Copyright © 2011.

All rights reserved.

Published by arrangement with the original publisher, Pearson Education, Inc., publishing as Addison-Wesley.

For sale and distribution in the People's Republic of China exclusively (except Taiwan, Hong Kong SAR and Macau SAR).

本书英文影印版由 Pearson Education Asia Ltd. 授权机械工业出版社独家出版。未经出版者书面许可，不得以任何方式复制或抄袭本书内容。

仅限于中华人民共和国境内（不包括中国香港、澳门特别行政区和中国台湾地区）销售发行。

本书封面贴有 Pearson Education（培生教育出版集团）激光防伪标签，无标签者不得销售。

封底无防伪标均为盗版
版权所有，侵权必究
本书法律顾问　北京市展达律师事务所

本书版权登记号：图字：01-2010-5383

图书在版编目（CIP）数据

计算机系统：集成方法（英文版）/（美）拉姆阿堪德兰（Ramachandran, U.），利海（Leahy, W. D.）著. —北京：机械工业出版社，2010. 9
（经典原版书库）
书名原文：Computer Systems: An Integrated Approach to Architecture and Operating Systems

ISBN 978-7-111-31955-9

Ⅰ. 计…　Ⅱ. ①拉…　②利…　Ⅲ. 计算机体系结构－英文　Ⅳ. TP303

中国版本图书馆 CIP 数据核字（2010）第 183857 号

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：李俊竹
北京京师印务有限公司印刷
2011年1月第1版第1次印刷
150mm × 214mm · 24. 125 印张
标准书号：ISBN 978-7-111-31955-9
定价：69. 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
购书热线：（010）68326294；88379649；68995259
投稿热线：（010）88379604
读者信箱：hzjsj@ hzbook. com

华章经典原版书库丛书

书名	书号	定价	出版时间	作者
计算机组成与设计：硬件/软件接口（英文版 第4版 ARM版）	30288	95.00元	2010-05-11	（美）David A.Patterson
逻辑与计算机设计基础　英文版 第4版	30310	58.00元	2010-05-05	（美）M.Morris Mano
嵌入式微控制器与处理器设计（英文版）	29250	49.00元	2010-01-12	（美） GregOsborn
C++程序设计原理与实践 （英文版）	28248	89.00元	2009-09-27	（美）Bjarne Stroustrup
搜索引擎信息检索实践（英文版）	28247	45.00元	2009-09-27	（美）W.Bruce Croft
组合数学　（英文版 第5版）	26525	49.00元	2009-04-03	（美）Richard A.Brualdi
嵌入式计算系统设计原理　（英文版.第2版）	25360	75.00元	2008-12-25	（美）Wayne Wolf
数学建模 方法与分析　（英文版.第3版）	25364	49.00元	2008-12-23	（美）MarkM.Meerschaert
自适应信号处理（英文版）	23918	56.00元	2008-05-09	（美）威德罗，斯特恩斯
计算机科学概论（英文版.第3版）	23516	66.00元	2008-03-31	（美）Nell Dale
计算机文化（英文版.第10版）	23250	69.00元	2008-03-31	（美）June Jamrich Parsons
自动网络管理系统（英文版）	23515	48.00元	2008-03-31	（美）Douglas E.Comer
Java语言程序设计 基础篇（英文版.第6版）	23367	66.00元	2008-02-14	（美）Y.Daniel Liang
微机电系统基础（英文版）	23167	59.00元	2008-01-28	（美）Chang Liu
数字设计和计算机体系结构（英文版）	22393	65.00元	2007-09-29	（美）David Money Harris
计算机网络－系统方法（英文版.第4版）	21401	85.00元	2007-05-24	（美）Larry L.Peterson
线性代数（英文版.第7版）	21198	58.00元	2007-04-27	（美）Steven J.Leon
MIPS体系结构透视（英文版.第2版）	20681	65.00元	2007-01-09	（英）Dominic Sweetman
数字逻辑基础与Verlog设计（英文版）	20356	56.00元	2006-12-12	（加）Stephen Brown Zvonko Design
Internet技术基础（英文版.第4版）	20419	45.00元	2006-12-07	（美）Douglas E.Comer
数据结构与算法分析：Java语言描述（英文版.第2版）	19876	55.00元	2006-10-17	（美）Mark Allen Weiss
计算机系统概论（英文版.第2版）	19766	66.00元	2006-09-13	（美）Yale N.Pstt Sanjay J.Patel
Intel微处理器（英文版.第7版）	19609	88.00元	2006-08-07	（美）Barry B.Brey
C 程序设计语言 （英文版.第2版）	19626	35.00元	2006-08-03	（美）Brian W.Kernighan

教师服务登记表

尊敬的老师：

您好！感谢您购买我们出版的 ________________________ 教材。

机械工业出版社华章公司本着为服务高等教育的出版原则，为进一步加强与高校教师的联系与沟通，更好地为高校教师服务，特制此表，请您填妥后发回给我们，我们将定期向您寄送华章公司最新的图书出版信息。为您的教材、论著或译著的出版提供可能的帮助。欢迎您对我们的教材和服务提出宝贵的意见，感谢您的大力支持与帮助！

个人资料（请用正楷完整填写）

教师姓名		□先生 □女士	出生年月		职务		职称：□教授 □副教授 □讲师 □助教 □其他	
学校			学院			系别		
联系电话	办公： 宅电： 移动：			联系地址及邮编				
				E-mail				
学历		毕业院校		国外进修及讲学经历				
研究领域								
主讲课程			现用教材名		作者及出版社	共同授课教师	教材满意度	
课程： □专 □本 □研 □MBA 人数： 学期：□春□秋							□满意 □一般 □不满意 □希望更换	
课程： □专 □本 □研 □MBA 人数： 学期：□春□秋							□满意 □一般 □不满意 □希望更换	
样书申请								
已出版著作				已出版译作				
是否愿意从事翻译/著作工作 □是 □否				方向				
意见和建议								

填妥后请选择以下任何一种方式将此表返回：（如方便请赐名片）

地　址：北京市西城区百万庄南街1号　华章公司营销中心　　邮编：100037

电　话：(010) 68353079 88378995　传真：(010)68995260

E-mail:hzedu@hzbook.com　markerting@hzbook.com　　图书详情可登录http://www.hzbook.com网站查询

此为试读，需要完整PDF请访问：www.ertongbook.com

出版者的话

文艺复兴以降，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；也正是这样的传统，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，Brain 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襄助，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，有的还专程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

HZ BOOKS
华章教育

华章科技图书出版中心

Preface

Why a New Book on Computer Systems?

There is excitement when you talk to high school students about computers. There is a sense of mystery as to what is "inside the box" that makes the computer do such things as play video games with cool graphics, play music—be it rap or symphony—send instant messages to friends, and so on. The purpose behind this textbook is to take the journey together to discover the mystery of what is inside the box. As a glimpse of what is to come, let us say at the outset that what makes the box interesting is not just the hardware, but also how the hardware and the system software work in tandem to make it all happen. Therefore, the path we take in this book is to look at hardware and software together to see how one helps the other and how together they make the box interesting and useful. We call this approach "unraveling the box"—that is, resolving the mystery of what is inside the box: We look inside the box and understand how to design the key hardware elements (processor, memory, and peripheral controllers) and the OS abstractions needed to manage all the hardware resources inside a computer, including processor, memory, I/O and disk, multiple processors, and network. Hence, this is a textbook for a first course in **computer systems** embodying a novel **integrated approach** to these topics.

The book is intended to give the breadth of knowledge in these topics at an early stage in a student's undergraduate career (in computer science or computer engineering). This text serves the need for teaching a course in an integrated fashion so that students can see the connection between the architecture and the system software. The material may be taught as a four-credit semester course, as a five-credit quarter course, or as a two-quarter three-credit course sequence. A course based on this textbook would serve well to prepare a student for more in-depth senior-level or graduate-level courses that go deeper into computer architecture, operating systems, and networking, to cater to specialization in those areas. Further, such a course kindles interest in systems early that can be capitalized on to involve students in undergraduate research.

Key features of the book (in addition to processor and memory systems) include the following:

1. a detailed treatment of storage systems;
2. an elaborate chapter devoted to networking issues; and
3. an elaborate chapter devoted to multiprocessing and multithreaded programming.

Pedagogical Style

The pedagogical style taken in the book is one of "discovery" as opposed to "instruction" or "indoctrination." Further, the presentation of a topic is "top down" in the sense that the reader is first exposed to the problem we are trying to solve and then initiated into the solution approach. Take, for example, memory management (Chapter 8). We first start with the question, "What is memory management?" Once the need for memory management is understood, we start identifying software techniques for memory management and the corresponding hardware support needed. Thus, the textbook almost takes a storytelling approach to presenting concepts, which students seem to love. Where appropriate, we have included worked-out examples of problems in the different chapters in order to elucidate a point.

Our focus in writing the textbook has always been on the students. One can see this commitment to students in the number of worked-out examples in the textbook that help solidify the concepts that were just discussed. In our experience as educators, we have seen that students really appreciate getting a historical context (names of famous computer scientists and organizations that have been instrumental in the evolution of computing) to where we are today and how we got here. We sprinkle such historical nuggets throughout the textbook. Additionally, we include a section on historical perspective in several chapters where it makes sense. Another thing that we learned and incorporated from listening to students is giving references to external work to amplify a point in context, rather than as an afterthought. One can see this in the number of footnotes throughout the textbook. Additionally, we also give bibliographic notes and pointers to further reading at the end of each chapter in a section devoted to external references (textbooks and seminal work) that may or may not be directly cited in the text, but should be useful to increasing the students' knowledge base. Today, with the abundance of information via the Internet, it is often tempting to point to URLs for additional information. However, we have desisted from this temptation (except for the inclusion of reliable links published by authoritative sources). Having said that, we know that students of this generation will go to the web first, before they go to the library, and of course, they should. In this context, we would like to share a note of caution to the students: Be judicious in your use of the Internet as a resource for information. Often, a Google search may be the quickest way to get information that you are seeking. However, you have to sift the information to ensure its veracity. As a rule of thumb, use the information from the web to answer curiosity questions or gossip. (How did DEC go out of business? Why did Linux succeed while Unix BSD did not? What is the history of the Burroughs corporation? Who are the real pioneers in computer systems?) For technical references (What is the pipeline structure of the Pentium 4? What is the instruction-set architecture of the VAX 11/780?), seek out published books and refereed conference and journal papers (many of which are available online, of course).

Incidentally, this textbook grew out of teaching such an integrated course, every semester from the fall of 1999, in the College of Computing at Georgia Institute of Technology. In the beginning, the authors developed a comprehensive set of notes and slides for the course and used two standard textbooks (one for architecture and one for operating systems) as background reference material for the students to supplement the course material. In the spring of 2005, we turned our courseware into a manuscript for a textbook, because the students continually communicated to us a need for a textbook that matched the style and contents of our course. An online version of this textbook has been in use at Georgia Tech since the spring of 2005 for this course that presents an integrated introduction to systems. The course is offered three times every year (including summers), with over 80 students taking the course every semester. Thus, the manuscript has received continuous feedback and improvement from students taking the course, for over 15 consecutive semesters, before going to print.

In designing the course from which this book was born, as well as in writing the book, we have learned a lot from the way a first course in systems is taught at other institutions, and from a number of excellent textbooks. For example, the first course in systems taught at MIT[1] has a long history and tradition, and is truly one of a kind. The book [Saltzer, 2009] that grew out of that course is a great resource for students aspiring to specialize in computer systems. In writing our book, we will freely admit that we have been inspired by the pedagogical styles of [Ward, 1989] and [Kurose, 2006].

The Structure of the Book and Possible Pathways Through the Book

The intellectual content of the book is broken up into five modules. The roadmap that follows suggests a possible pathway through the material. The pathway assumes a roughly equal coverage of the architecture and operating systems topics.

1. **Processor:** The first module deals with the processor, and software issues associated with the processor. We start by discovering how to design the brain inside the box[2], the processor. What are the software issues? Since computers are programmed, for the most part, in high-level language, we consider the influence of high-level language (HLL) constructs on the instruction set of the processor (Chapter 2). Once we understand the design of the instruction set, we focus on the hardware issue of implementing the processor. We start with a simple implementation of the processor (Chapter 3), and then go on to consider a performance-conscious implementation with the use of pipelining techniques (Chapter 5). The processor is a precious resource that has to be multiplexed among several competing programs that may need to run on it, as illustrated by the video-game example in Chapter 1 (see Section 1.3). It is the OS's responsibility to use this resource well. The module concludes with OS algorithms for processor scheduling (Chapter 6).

   We expect each of Chapters 2, 3, 5, and 6 to require three hours of classroom instruction, with an hour of recitation help for each chapter.
2. **Memory System:** The second module deals with memory systems and memory hierarchies. A computer program comprises code and data, and therefore needs space in which to reside. The memory system of a computer is perhaps the most crucial factor in determining its performance. The processor speed (measured in gigahertz these days) may mean nothing if the memory system does not match that speed by providing, in a timely manner, the code and data necessary for executing a program. Whereas the size of memory systems is growing by leaps and bounds, thanks to advances in technology, applications' appetite for using memory is growing equally fast, if not faster. Thus, memory also is a precious resource, and it is the responsibility of the OS to manage this resource well. The first part of this module concerns the OS

1. http://mit.edu/6.033/www/.
2. The anatomical allusion in the cover design is meant to illustrate the analogy of computing to the networked distributed processing that happens so naturally in the human body.

algorithms for efficient management of memory and the architectural assists for supporting it (Chapters 7 and 8); the second part deals with the memory hierarchies that help to reduce the latency seen by the processor when accessing code and data (Chapter 9).

We expect each of Chapters 7, 8, and 9 to require three hours of classroom instruction, with an hour of recitation help for each chapter.

3. **Storage System:** The third module deals with the I/O (particularly, stable storage) and the file system. What makes the computer useful and interesting is being able to interact with it. First, we deal with hardware mechanisms for grabbing the attention of the processor away from the currently executing program (Chapter 4). These mechanisms deal with both external events and internal exceptions encountered by the processor during program execution. Associated with the hardware mechanisms are software issues that address "discontinuities" in the normal program execution, which include remembering our location in the original program and the current state of the program execution. Next, we delve into the mechanisms for interfacing the processor to I/O devices and the corresponding low-level software issues such as device drivers (Chapter 10), with a special emphasis on the disk subsystem. This is followed by a comprehensive treatment of the file system (Chapters 11) built on stable storage such as the disk.

   We expect each of Chapters 4 and 10 to require three hours of classroom instruction and an hour each of recitation help; Chapter 11 should require six hours of classroom instruction and two hours of recitation help.

4. **Parallel System:** Computer architecture is a fast-changing field. Chip density, processor speed, memory capacity, etc., have all been showing exponential growth over the last two decades and are expected to continue that trend for the foreseeable future. Parallel processing is no longer an esoteric concept reserved for supercomputers. With the advent of multicore technology that houses multiple CPUs inside a single chip, parallelism is becoming a commodity. Therefore, understanding the hardware and software issues surrounding parallelism is necessary to answer the question "What is inside a box?" This module deals with operating systems issues and the corresponding architectural features in multiprocessors for supporting parallel programming (Chapter 12).

   We expect Chapter 12 to require six hours of classroom instruction, with two hours of recitation help.

5. **Networking:** In the world we live in, a box is almost useless unless it is connected to the outside world. The multiplayer video game (introduced in Chapter 1), with your friends as fellow players on the network, is a nice motivating example, but even in our everyday mundane activities, we need the network for e-mail, web browsing, etc. What distinguishes the network from other input/output devices is the fact that your box is now exposed to the world! You need a language to talk to the outside world from your box and deal with the vagaries of the network, such as temporary or permanent disconnections. This module deals with the evolution of networking hardware, and the features of the network protocol stack (which is part of the operating system) for dealing with the vagaries of the network (Chapter 13).

   We expect Chapter 13 to require six hours of classroom instruction, with two hours of recitation help.

In a nutshell, Chapters 2 through 10 will require one week of instruction each; Chapters 11, 12 and 13 will require two weeks of instruction each, rounding up a 15-week

此为试读，需要完整PDF请访问：www.ertongbook.com

semester. The hardware and software issues for each of the five modules are treated together in this textbook. The suggested pathway above through the material follows this treatment.

It is possible to tilt the coverage unevenly between architecture and operating systems topics if one so chooses, without loss of continuity. Let us consider the processor module. Chapters 3 and 5 deal with the hardware implementation issues of the processor. For a course that is more OS-oriented, Chapter 5, which deals with pipelined processor implementation (starting from Section 5.7), may be lightly covered or skipped altogether (depending on time constraints), without loss of continuity. Similarly, in a course that is more architecture oriented, Chapter 6, dealing with processor scheduling issues, may be skipped altogether, without loss of continuity.

In the memory module, Chapter 8 deals with details of page-based memory management, from an OS perspective. An architecture-oriented course could skip this chapter if it so chooses, without loss of continuity. Similarly, an OS-oriented course may choose to tone down the detailed treatment of cache memories in Chapter 9.

In the storage module, an architecture-oriented course may choose to tone down the treatment of file systems coverage in Chapter 11, without concern over loss of continuity.

In the parallel module (Chapter 12), an architecture-oriented course may skip topics such as OS support for multithreading and advanced topics such as multiprocessor scheduling, deadlocks, and classic problems and solutions in concurrency; similarly, an OS-oriented course may choose to skip advanced topics in architecture, such as multiprocessor cache coherence, taxonomy of parallel machines, and interconnection networks. Given the importance of parallelism, it would be prudent to cover the chapter as completely as possible, subject to time constraints, in any course offering.

In the networking module (Chapter 13), an architecture-oriented course offering may skip the detailed treatment of the transport and network layers (Sections 13.6 and 13.7, respectively). An OS-oriented course may choose to cover less of the link layer of the protocol stack (Section 13.8) and the networking hardware (Section 13.9).

Where Does This Textbook Fit into the Continuum of CS Curriculum?

Figure P.1 shows the levels of abstraction in a computer system. We can try to relate the levels of abstraction in Figure P.1 to courses in a typical CS curriculum. Courses such as basic programming, object-oriented design and programming, graphics, and HCI

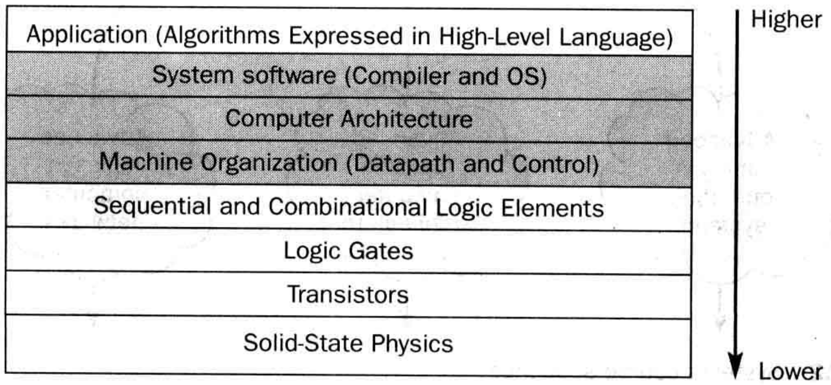

Application (Algorithms Expressed in High-Level Language)
System software (Compiler and OS)
Computer Architecture
Machine Organization (Datapath and Control)
Sequential and Combinational Logic Elements
Logic Gates
Transistors
Solid-State Physics
Higher
Lower

Figure P.1 Levels of abstraction in a computer system.

generally deal with higher layers of abstraction. Typically, computer science and computer engineering curricula offer courses dealing with the fundamentals of digital electronics and logic design, followed by a course on computer organization that deals purely with the hardware design of a computer. Beyond the computer organization course (moving up the levels of abstraction shown in Figure P.1), most curricula take a stovepipe approach: distinct courses dealing with advanced concepts in computer architecture, operating systems, and computer networks, respectively.

Design of computer systems is such an integrated process today that one has to seriously question this stovepipe approach, especially in the early stages of the development of a student in an undergraduate curriculum in computer science.

A course structured around the topics covered in this book is a unique attempt to present concepts in the middle (covering topics in the shaded area of Figure P.1—systems software and their relationship to computer architecture) in a unified manner in an introductory systems course. Such a course would serve as a solid preparation for students aspiring to learn advanced topics in computer architecture, operating systems, and networking (Figure P.2).

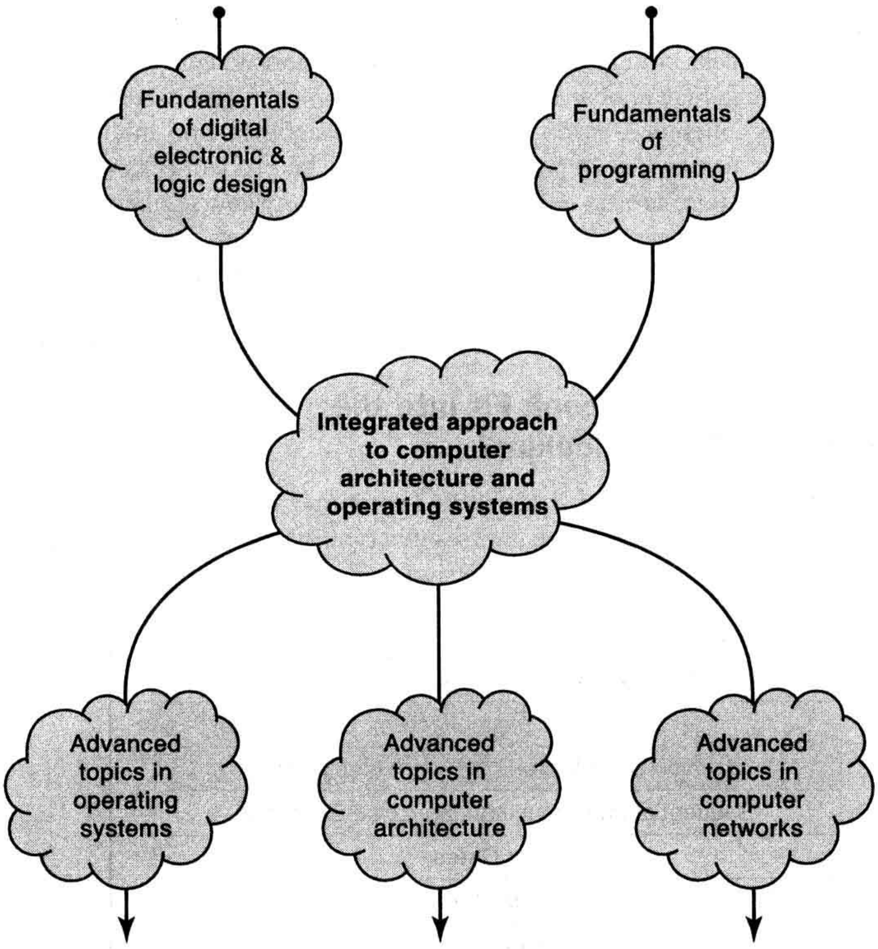

Fundamentals of digital electronic & logic design
Fundamentals of programming
Integrated approach to computer architecture and operating systems
Advanced topics in operating systems
Advanced topics in computer architecture
Advanced topics in computer networks

Figure P.2 Systems course sequence.

The prerequisites for a course structured around the topics covered in this textbook are quite straightforward: basic logic design and programming, using a high-level language (preferably, C). In other words, a fundamental understanding of the levels of abstractions above and below the topics covered in this book (see Figure P.1) is required.

There are excellent textbooks that cater to the fundamentals of digital electronics and logic design, as well as fundamentals of programming. Similarly, there are excellent textbooks that deal with advanced topics in computer architecture, operating systems, and computer networking. ***What is missing is a simplified and integrated introduction to computer systems that serves as the bridge between the fundamentals and the advanced topics. The aim of this textbook is to serve as this bridge.***

The boundary of computer science as a discipline has expanded. Correspondingly, students coming into this discipline have varied interests. There is a need for CS curricula to offer choices for students to pursue in their undergraduate careers. At the same time, there is a responsibility to ensure that students acquire "core" knowledge in systems (broadly defined) regardless of these choices. We believe that a course structured around this textbook would fulfill such a core systems requirement. If taught right, it should give ample opportunity for students to pursue deeper knowledge in systems, through further coursework. For example, our recommendation would be to have a course based on this textbook in the sophomore year. In the junior year, the students may be ready to take courses that are more design oriented—specializing in architecture, operating systems, and/or networking—building on the basic concepts they learned in their sophomore year via this textbook. Finally, in their senior year the students may be able to take more conceptual courses on advanced topics in these areas.

The textbook balances the treatment of both architecture and operating systems topics. It is the belief of the authors that students majoring in computer science should get an equal treatment of the two topics early in their undergraduate preparation, regardless of their career objectives. Certainly, students aspiring to becoming system architects, must understand the interplay between hardware and software, as laid out in this textbook. Even for students aspiring to specialize in software development, such an understanding is essential to becoming better programmers. However, it is up to individual instructors how much emphasis to place on the two topics. The good news is that the textbook allows instructors to go into as much depth as they deem necessary, commensurate with the curricular structure existing in their institutions. For example, if an instructor chooses to scale back on the architecture side, it would be quite easy to tread lightly on the implementation chapters addressing the processor (Chapters 3 and 5), without losing continuity in the discourse. In discussing the structure of this textbook, we have already given similar suggestions for each of the five modules that this textbook comprises.

Supplementary Material for Teaching an Integrated Course in Systems

The authors fully understand the challenge an instructor faces in teaching an integrated course in computer systems that touches on architecture, operating systems, and networking.

To this end, we make available a set of online resources. Since we have been teaching this course—three offerings in each calendar year for the last 11 years—as a requirement for all computer science majors, we have amassed a significant collection of online resources:

1. We have PowerPoint slides for all the topics covered in the course, making preparation and transition (from the stovepipe model) easy.
2. A significant project component dovetails each of the five modules. We have detailed project descriptions of several iterations of these projects, along with software modules (such as simulators) for specific aspects of the projects.
3. In addition to the problems at the end of each chapter, we have additional problem sets for the different modules of the course, as well as homework problems and midterm and final exams used thus far in the course.

Example Project Ideas Included in the Supplementary Material

Processor Design

Students are supplied with a data path design that is 90% complete. Students complete the data path to help them become familiar with the design. Then they design the microcode-based control logic (using a logic design software such as LogicWorks) for implementing a simple instruction set using the data path. This allows the students to get a good understanding of how a data path functions and to appreciate some of the design tradeoffs. The students get actual circuit design experience and functionally test their design, using the built-in functional simulator of the logic design software.

Interrupts and Input/Output

Students take the design from the first project and add circuitry to implement an interrupt system. Then they write (in assembly language) an interrupt handler. The circuit design part of the project is once again implemented and functionally simulated using LogicWorks software system. In addition, the students are supplied with a processor simulator that they enhance with the interrupt support, and use it in concert with the interrupt handler, which they write in assembly language. This project not only makes operation of the interrupt system clear, but also illustrates fundamental concepts of low-level device input/output.

Virtual Memory Subsystem

Students implement a virtual memory subsystem that operates with a supplied processor simulator. The students get the feel for developing the memory-management part of an operating system through this project by implementing and experimenting with different page replacement policies. The project is implemented in the C programming language.

Multi-Threaded Operating System

Students implement the basic modules of a multithreaded operating system, including CPU and I/O scheduling queues, on top of a simulator that we supply. They experiment with different processor scheduling policies. The modules are implemented in C, using pthreads. The students get experience with parallel programming, as well as exposure to different CPU scheduling algorithms.

Reliable Transport Layer

Students implement a simple reliable transport layer on top of a simulated network layer provided to them. Issues that must be dealt with in the transport layer include corrupt packets, missing packets, and out-of-order delivery. This project is also implemented in C, using pthreads.

A Note of Caution

We offer one word of caution as we launch on our journey to explore the insides of a computer system: In a textbook that presents computer system design, it is customary to back up concepts with numerical examples to illustrate the concepts. The past is indicative of the future. If there is a constant in the technology landscape, it is *change*. When you buy a new car, the minute it rolls out of the showroom, it becomes a used car. In the same manner, any numbers we may use in the numerical examples as to the speed of the processor, or the capacity of memory, or the transfer rate of peripherals become outdated instantly. What endure are the *principles*, which is the focus of this textbook. A comfort factor is that whereas the absolute numbers may change with time—from megaHertz to gigaHertz, and megabytes to gigabytes—the *relative* numbers stay roughly the same as technology advances, thus making the numerical examples in the textbook endure with time.

Acknowledgments

We are deeply indebted to several colleagues, nationally and internationally, who have been either directly or indirectly responsible for the creation of this textbook. First and foremost, we would like to thank Yale Patt, who, back in the summer of 2004 when we described the course that we teach at Georgia Tech, told us in his inimitable forceful style that we should write a textbook because there is a crying need for a book that presents the systems concepts in an integrated manner. We can honestly say that, but for his encouragement, we may not have embarked on this path. Our colleagues at other institutions who deserve special mention for encouraging us to write this textbook include Jim Goodman (University of Wisconsin-Madison and University of Auckland, New Zealand), Liviu Iftode (Rutgers University), Phil McKinley (Michigan State University), and Anand Sivasubramaniam (Pennsylvania State University and TCS). We are particularly thankful to Jim Goodman for his careful reading of an early draft of the manuscript and providing detailed feedback that helped improve the discourse tremendously. Besides these folks, we received much positive reinforcement for our project from several colleagues in other institutions who helped to get us started.

The first step was creating a manuscript for internal consumption by students at Georgia Tech. We cannot thank the students of CS 2200 at Georgia Tech enough. The feedback from several generations of students, who have used the online version of this textbook since the spring of 2005, has been immensely useful in improving the presentation clarity, refining specific worked-out examples in the text, providing historical pointers that would interest the reader, and other contributions. In addition, three undergraduate students helped with some of the artwork in this textbook: Kristin Champion, John Madden, and Vu Ha.

Several colleagues in the College of Computing, including Nate Clark, Tom Conte, Constantine Dovrolis, Gabriel Loh, Ken Mackenzie, and Milos Prvulovic, have given suggestions and insightful comments that have helped clarify the discourse in this textbook. We owe a lot to Constantine Dovrolis for suggestions and feedback on early drafts of the networking chapter that have helped improve both the content and the ordering of its presentation in that chapter. Ken Mackenzie's suggestions helped us to come up with a simple control regime for processor design in Chapter 3. Tom Conte gave detailed comments on the pipelining chapter that helped in improving the clarity and the content. Eric Rotenberg of North Carolina State University provided very useful feedback on early drafts of the pipelining chapter. Junsuk Shin wrote the simple client–server socket code that appears in the appendix. Our special thanks go to all of them.

We wish to thank Georgia Tech, and the vision of the College of Computing that encourages such creative thinking on the teaching side. Indeed, it is the revision of the entire undergraduate curriculum back in 1996 which started us on the path of looking critically at how we teach our undergraduates and understanding what we are missing in the curriculum that ultimately led us to develop a first course in systems as an integrated offering spanning architecture, operating systems, and networking.

Being novices at book publishing, we turned to successful textbook authors to learn from their experiences. Yale Patt (University of Texas), Jim Kurose (University of Massachusetts), Jim Foley (Georgia Tech), Andy van Dam (Brown University), Sham Navathe (Georgia Tech), Rich LeBlanc (Georgia Tech), and Larry Snyder (University of Washington) deserve special mention. We cannot thank them enough for their generosity in sharing their experiences as authors and guiding us through the various aspects of book publishing, including choosing a publisher, working with an editor, framing questions for potential reviewers, and effectively using the reviews in revising the manuscript.

The manuscript went through several rounds of external review. Most of the anonymous reviewers were thoughtful and skillfully surgical in pointing out ways to improve the manuscript. We are extremely grateful for their time and help in shaping the final product.

Our special thanks to Addison-Wesley for publishing our textbook. As our manuscript editor in charge of overseeing the review process and giving us feedback on how to improve the manuscript, Matt Goldstein has been superb. His style of looking over our shoulders without being overbearing is unique. He has been patient with us when we let schedules slip and has been unwaveringly supportive of the vision behind this book project. Our thanks to Marilyn Lloyd, senior production manager at Pearson, who was in charge of our textbook production. Our thanks go also to Jeff Holcomb, Chelsea Bell, and Dan Parker of Pearson. As the project manager overseeing the day-to-day details of the production process, Dennis Free of Aptara and the Aptara staff, including Jawwad Ali Khan and Rajshri Walia, and Brian Baker of Write With, Inc., deserve special mention for bringing the production of this book to fruition in a timely manner.

Finally, we would like to thank our families for their love, understanding, and support, which sustained us throughout the writing of this book. As an aside, Umakishore's father was a famous novelist (with the pen-name "Umachandran") with several fictional books in Tamil to his credit; memories of him served as an inspiration for undertaking this book-writing project.

Umakishore Ramachandran
William D. Leahy, Jr.

此为试读，需要完整PDF请访问：www.ertongbook.com