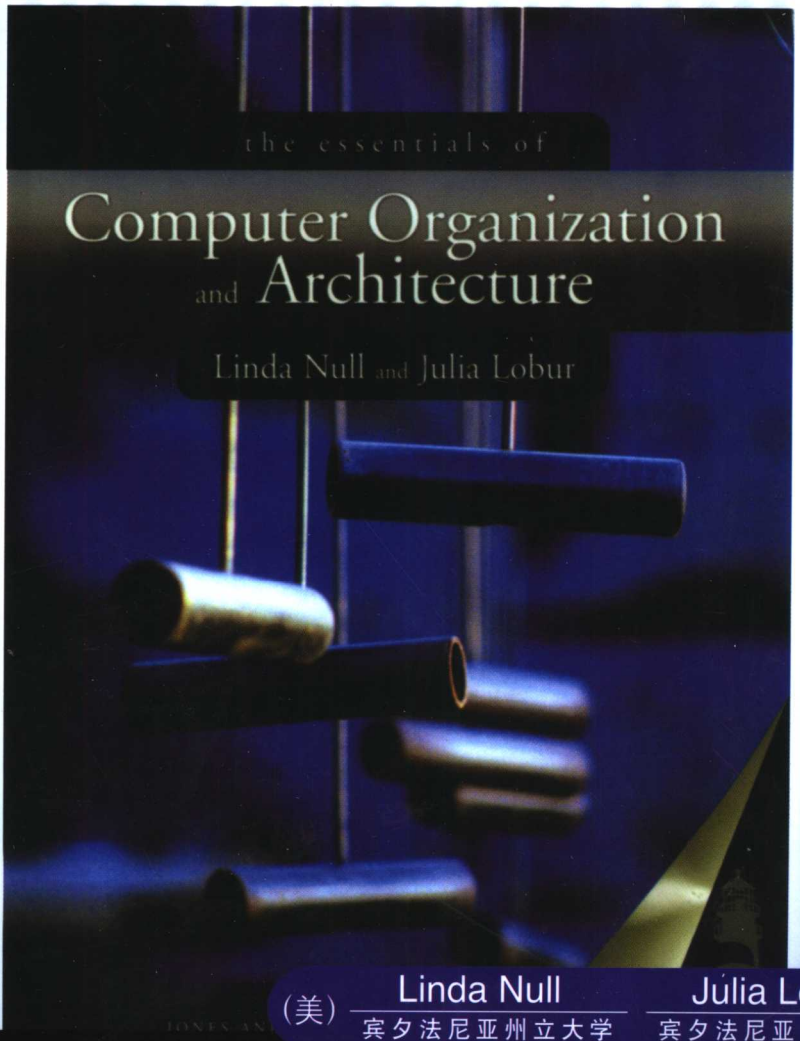
HZ BOOKS

经 典 原 版 书 库

计算机组成与体系结构

（英文版）



（美） Linda Null 宾夕法尼亚州立大学 Julia Lobur 宾夕法尼亚州立大学 著

机械工业出版社
China Machine Press

经 典 原 版 书 库

计算机组成与体系结构

（英文版）

The Essentials of Computer Organization and Architecture

（美）Linda Null 宾夕法尼亚州立大学
Julia Lobur 宾夕法尼亚州立大学 著

机械工业出版社
China Machine Press

Linda Null and Julia Lobur: The Essentials of Computer Organization and Architecture (ISBN 0-7637-0444-X).

Original English language edition published by Jones and Bartlett Publishers, Inc., 40 Tall Pine Drive, Sudbury, MA 01776.

本书版权登记号：图字：01-2004-5737

图书在版编目（CIP）数据

计算机组成与体系结构（英文版）/（美）纳尔（Null, L.）等著；—北京：机械工业出版社，2004.11

（经典原版书库）

书名原文：The Essentials of Computer Organization and Architecture

ISBN 7-111-15311-1

Ⅰ.计… Ⅱ.纳… Ⅲ.计算机体系结构—英文 Ⅳ.TP303

中国版本图书馆CIP数据核字（2004）第098878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：迟振春

北京中兴印刷有限公司印刷 · 新华书店北京发行所发行

2004年11月第1版第1次印刷

787mm × 1092mm 1/16 · 38.25 印张

印数：0 001-3 000 册

定价：55.00 元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：（010）68326294

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，还揭橥了学术的源变，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、从业人员较少的现状下，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图文信息有限公司较早意识到“出版要为教育服务”。自1998年开始，华章公司就将工作重点放在了遴选、移译国外优秀教材上。经过几年的不懈努力，我们与Prentice Hall，Addison-Wesley，McGraw-Hill，Morgan Kaufmann等世界著名出版公司建立了良好的合作关系，从它们现有的数百种教材中甄选出Tanenbaum，Stroustrup，Kernighan，Jim Gray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、研究及庋藏。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，有的还专诚为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近百个品种，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，为进一步推广与发展打下了坚实的基础。

随着学科建设的初步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步入一个新的阶段。为此，华章公司将加大引进教材的力度，在“华章教育”的总规划之下出版三个系列的计算机教材：除“计算机科学丛书”之外，对影印版的教材，则单独开辟出“经典原版书库”；同时，引进全美通行的教学辅导书“Schaum's Outlines”系列组成“全美经典学习指导系列”。为了保证这三套丛书的权威性，同时也为了更好地为学校和老师们服务，华章公司聘请了中国科学院、北京大学、清华大学、国防科技大学、复旦大学、上海交通大学、南京大学、浙江大学、中国科技大学、哈尔滨工业大学、西安交通大学、中国人民大学、北京航空航天大学、北京邮电大学、中山大学、解放军理工大学、郑州大学、湖北工学院、中国国

家信息安全测评认证中心等国内重点大学和科研机构在计算机的各个领域的著名学者组成“专家指导委员会”，为我们提供选题意见和出版监督。

这三套丛书是响应教育部提出的使用外版教材的号召，为国内高校的计算机及相关专业的教学度身订造的。其中许多教材均已为M. I. T.，Stanford，U.C. Berkeley，C. M. U. 等世界名牌大学所采用。不仅涵盖了程序设计、数据结构、操作系统、计算机体系结构、数据库、编译原理、软件工程、图形学、通信与网络、离散数学等国内大学计算机专业普遍开设的核心课程，而且各具特色——有的出自语言设计者之手、有的历经三十年而不衰、有的已被全世界的几百所高校采用。在这些圆熟通博的名师大作的指引之下，读者必将在计算机科学的宫殿中由登堂而入室。

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，但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。教材的出版只是我们的后续服务的起点。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：

电子邮件：hzedu@hzbook.com
联系电话：(010) 68995264
联系地址：北京市西城区百万庄南街1号
邮政编码：100037

专家指导委员会

（按姓氏笔画顺序）

尤晋元	王　珊	冯博琴	史忠植	史美林
石教英	吕　建	孙玉芳	吴世忠	吴时霖
张立昂	李伟琴	李师贤	李建中	杨冬青
邵维忠	陆丽娜	陆鑫达	陈向群	周伯生
周立柱	周克定	周傲英	孟小峰	岳丽华
范　明	郑国梁	施伯乐	钟玉琢	唐世渭
袁崇义	高传善	梅　宏	程　旭	程时端
谢希仁	裘宗燕	戴　葵		

秘　书　组

武卫东　　温莉芳　　刘　江　　杨海玲

In memory of my father, Merrill Cornell, a pilot and man of endless talent and courage, who taught me that when we step into the unknown, we either find solid ground, or we learn to fly.

—L. M. N.

To the loving memory of my mother, Anna J. Surowski, who made all things possible for her girls.

—J. M. L.

PREFACE

TO THE STUDENT

This is a book about computer organization and architecture. It focuses on the function and design of the various components necessary to process information digitally. We present computing systems as a series of layers, starting with low-level hardware and progressing to higher-level software, including assemblers and operating systems. These levels constitute a hierarchy of virtual machines. The study of computer organization focuses on this hierarchy and the issues involved with how we partition the levels and how each level is implemented. The study of computer architecture focuses on the interface between hardware and software, and emphasizes the structure and behavior of the system. The majority of information contained in this textbook is devoted to computer hardware, and computer organization and architecture, and their relationship to software performance.

Students invariably ask, "Why, if I am a computer science major, must I learn about computer hardware? Isn't that for computer engineers? Why do I care what the inside of a computer looks like?" As computer users, we probably do not have to worry about this any more than we need to know what our car looks like under the hood in order to drive it. We can certainly write high-level language programs without understanding how these programs execute; we can use various application packages without understanding how they really work. But what happens when the program we have written needs to be faster and more

efficient, or the application we are using doesn't do precisely what we want? As computer scientists, we need a basic understanding of the computer system itself in order to rectify these problems.

There is a fundamental relationship between the computer hardware and the many aspects of programming and software components in computer systems. In order to write good software, it is very important to understand the computer system as a whole. Understanding hardware can help you explain the mysterious errors that sometimes creep into your programs, such as the infamous segmentation fault or bus error. The level of knowledge about computer organization and computer architecture that a high-level programmer must have depends on the task the high-level programmer is attempting to complete.

For example, to write compilers, you must understand the particular hardware to which you are compiling. Some of the ideas used in hardware (such as pipelining) can be adapted to compilation techniques, thus making the compiler faster and more efficient. To model large, complex, real-world systems, you must understand how floating-point arithmetic should, and does, work (which are not necessarily the same thing). To write device drivers for video, disks, or other I/O devices, you need a good understanding of I/O interfacing and computer architecture in general. If you want to work on embedded systems, which are usually very resource-constrained, you must understand all of the time, space, and price tradeoffs. To do research on, and make recommendations for, hardware systems, networks, or specific algorithms, you must acquire an understanding of benchmarking and then learn how to present performance results adequately. Before buying hardware, you need to understand benchmarking and all of the ways in which others can *manipulate* the performance results to "prove" that one system is better than another. Regardless of our particular area of expertise, as computer scientists, it is imperative that we understand how hardware interacts with software.

You may also be wondering why a book with the word *essentials* in its title is so large. The reason is twofold. First, the subject of computer organization is expansive and it grows by the day. Second, there is little agreement as to which topics from within this burgeoning sea of information are truly essential and which are just helpful to know. In writing this book, one goal was to provide a concise text compliant with the computer architecture curriculum guidelines jointly published by the Association for Computing Machinery (ACM) and the Institute of Electrical and Electronic Engineers (IEEE). These guidelines encompass the subject matter that experts agree constitutes the "essential" core body of knowledge relevant to the subject of computer organization and architecture.

We have augmented the ACM/IEEE recommendations with subject matter that we feel is useful—if not essential—to your continuing computer science studies and to your professional advancement. The topics we feel will help you in your continuing computer science studies include operating systems, compilers, database management, and data communications. Other subjects are included because they will help you understand how actual systems work in real life.

We hope that you find reading this book an enjoyable experience, and that you take time to delve deeper into some of the material that we have presented. It is our intention that this book will serve as a useful reference long after your formal course is complete. Although we give you a substantial amount of information, it is only a foundation upon which you can build throughout the remainder of your studies and your career. Successful computer professionals continually add to their knowledge about how computers work. Welcome to the start of your journey.

TO THE INSTRUCTOR

About the Book

This book is the outgrowth of two computer science organization and architecture classes taught at The Pennsylvania State University Harrisburg campus. As the computer science curriculum evolved, we found it necessary not only to modify the material taught in the courses but also to condense the courses from a two-semester sequence into a three credit, one-semester course. Many other schools have also recognized the need to compress material in order to make room for emerging topics. This new course, as well as this textbook, is primarily for computer science majors, and is intended to address the topics in computer organization and architecture with which computer science majors must be familiar. This book not only integrates the underlying principles in these areas, but it also introduces and motivates the topics, providing the breadth necessary for majors, while providing the depth necessary for continuing studies in computer science.

Our primary objective in writing this book is to change the way computer organization and architecture are typically taught. A computer science major should leave a computer organization and architecture class with not only an understanding of the important general concepts on which the digital computer is founded, but also with a comprehension of how those concepts apply to the real world. These concepts should transcend vendor-specific terminology and design; in fact, students should be able to take concepts given in the specific and translate to the generic and vice versa. In addition, students must develop a firm foundation for further study in the major.

The title of our book, *The Essentials of Computer Organization and Architecture,* is intended to convey that the topics presented in the text are those for which every computer science major should have exposure, familiarity, or mastery. We do not expect students using our textbook to have complete mastery of all topics presented. It is our firm belief, however, that there are certain topics that must be mastered; there are those topics for which students must have a definite familiarity; and there are certain topics for which a brief introduction and exposure are adequate.

We do not feel that concepts presented in sufficient depth can be learned by studying general principles in isolation. We therefore present the topics as an inte-

grated set of solutions, not simply a collection of individual pieces of information. We feel our explanations, examples, exercises, tutorials, and simulators all combine to provide the student with a total learning experience that exposes the inner workings of a modern digital computer at the appropriate level.

We have written this textbook in an informal style, omitting unnecessary jargon, writing clearly and concisely, and avoiding unnecessary abstraction, in hopes of increasing student enthusiasm. We have also broadened the range of topics typically found in a first-level architecture book to include system software, a brief tour of operating systems, performance issues, alternative architectures, and a concise introduction to networking, as these topics are intimately related to computer hardware. Like most books, we have chosen an architectural model, but it is one that we have designed with simplicity in mind.

Relationship to Computing Curricula 2001

In December of 2001, the ACM/IEEE Joint Task Force unveiled the 2001 Computing Curricula (CC-2001). These new guidelines represent the first major revision since the very popular Computing Curricula 1991. CC-2001 represents several major changes from CC-1991, but we are mainly concerned with those that address computer organization and computer architecture. CC-1991 suggested approximately 59 lecture hours for architecture (defined as both organization and architecture and labeled AR), including the following topics: digital logic, digital systems, machine-level representation of data, assembly-level machine organization, memory system organization and architecture, interfacing and communication, and alternative architectures. The latest release of CC-2001 (available at www.computer.org/education/cc2001/) reduces architecture coverage to 36 core hours, including digital logic and digital systems (3 hours), machine-level representation of data (3 hours), assembly-level machine organization (9 hours), memory system organization and architecture (5 hours), interfacing and communication (3 hours), functional organization (7 hours), and multiprocessing and alternative architectures (3 hours). In addition, CC-2001 suggests including performance enhancements and architectures for networks and distributed systems as part of the architecture and organization module for CC-2001. We are pleased, after completely revising our course and writing this textbook, that our new material is in direct correlation with the ACM/IEEE 2001 Curriculum guidelines for computer organization and architecture as follows:

AR1. Digital logic and digital systems (core): Chapters 1 and 3
AR2. Machine-level representation of data (core): Chapter 2
AR3. Assembly-level machine organization (core): Chapters 4, 5 and 6
AR4. Memory system organization and architecture (core): Chapter 6
AR5. Interfacing and communication (core): Chapter 7
AR6. Functional organization (core): Chapters 4 and 5
AR7. Multiprocessing and alternative architectures (core): Chapter 9

AR8. Performance enhancements (elective): Chapters 9 and 10

AR9. Architecture for networks and distributed systems (elective): Chapter 11

Why another text?

No one can deny there is a plethora of textbooks for teaching computer organization and architecture already on the market. In our 25-plus years of teaching these courses, we have used many very good textbooks. However, each time we have taught the course, the content has evolved, and, eventually, we discovered we were writing significantly more course notes to bridge the gap between the material in the textbook and the material we deemed necessary to present in our classes. We found that our course material was migrating from a computer engineering approach to organization and architecture toward a computer science approach to these topics. When the decision was made to fold the organization class and the architecture class into one course, we simply could not find a textbook that covered the material we felt was necessary for our majors, written from a computer science point of view, written without machine-specific terminology, and designed to motivate the topics before covering them.

In this textbook, we hope to convey the spirit of design used in the development of modern computing systems and what impact this has on computer science students. Students, however, must have a strong understanding of the basic concepts before they can understand and appreciate the non-tangible aspects of design. Most organization and architecture textbooks present a similar subset of technical information regarding these basics. We, however, pay particular attention to the level at which the information should be covered, and to presenting that information in the context that has relevance for computer science students. For example, throughout this book, when concrete examples are necessary, we offer examples for personal computers, enterprise systems, and mainframes, as these are the types of systems most likely to be encountered. We avoid the "PC bias" prevalent in similar books in the hope that students will gain an appreciation for the differences, similarities, and the roles various platforms play within today's automated infrastructures. Too often, textbooks forget that motivation is, perhaps, the single most important key in learning. To that end, we include many real-world examples, while attempting to maintain a balance between theory and application.

Features

We have included many features in this textbook to emphasize the various concepts in computer organization and architecture, and to make the material more accessible to students. Some of the features are listed below:

- *Sidebars.* These sidebars include interesting tidbits of information that go a step beyond the main focus of the chapter, thus allowing readers to delve further into the material.

- *Real-World Examples.* We have integrated the textbook with examples from real life to give students a better understanding of how technology and techniques are combined for practical purposes.
- *Chapter Summaries.* These sections provide brief yet concise summaries of the main points in each chapter.
- *Further Reading.* These sections list additional sources for those readers who wish to investigate any of the topics in more detail, and contain references to definitive papers and books related to the chapter topics.
- *Review Questions.* Each chapter contains a set of review questions designed to ensure that the reader has a firm grasp on the material.
- *Chapter Exercises.* Each chapter has a broad selection of exercises to reinforce the ideas presented. More challenging exercises are marked with an asterisk.
- *Answers to Selected Exercises.* To ensure students are on the right track, we provide answers to representative questions from each chapter. Questions with answers in the back of the text are marked with a blue diamond.
- *Special "Focus On" Sections.* These sections provide additional information for instructors who may wish to cover certain concepts, such as Kmaps and input/output, in more detail. Additional exercises are provided for these sections as well.
- *Appendix.* The appendix provides a brief introduction or review of data structures, including topics such as stacks, linked lists, and trees.
- *Glossary.* An extensive glossary includes brief definitions of all key terms from the chapters.
- *Index.* An exhaustive index is provided with this book, with multiple cross-references, to make finding terms and concepts easier for the reader.

About the Authors

We bring to this textbook not only 25-plus years of combined teaching experience, but also 20 years of industry experience. Our combined efforts therefore stress the underlying principles of computer organization and architecture, and how these topics relate in practice. We include real-life examples to help students appreciate how these fundamental concepts are applied in the world of computing.

Linda Null received a Ph.D. in Computer Science from Iowa State University in 1991, an M.S. in Computer Science from Iowa State University in 1989, an M.S. in Computer Science Education from Northwest Missouri State University in 1983, an M.S. in Mathematics Education from Northwest Missouri State University in 1980, and a B.S. in Mathematics and English from Northwest Missouri State University in 1977. She has been teaching mathematics and computer science for over 25 years and is currently the Computer Science graduate program coordinator at The Pennsylvania State University Harrisburg campus, where she has been a member of the faculty since 1995. Her areas of interest include computer organization and architecture, operating systems, and computer security.

Julia Lobur has been a practitioner in the computer industry for over 20 years. She has held positions as a systems consultant, a staff programmer/analyst, a systems and network designer, and a software development manager, in addition to part-time teaching duties.

Prerequisites

The typical background necessary for a student using this textbook includes a year of programming experience using a high-level procedural language. Students are also expected to have taken a year of college-level mathematics (calculus or discrete mathematics), as this textbook assumes and incorporates these mathematical concepts. This book assumes no prior knowledge of computer hardware.

A computer organization and architecture class is customarily a prerequisite for an undergraduate operating systems class (students must know about the memory hierarchy, concurrency, exceptions, and interrupts), compilers (students must know about instruction sets, memory addressing, and linking), networking (students must understand the hardware of a system before attempting to understand the network that ties these components together), and of course, any advanced architecture class. This text covers the topics necessary for these courses.

General Organization and Coverage

Our presentation of concepts in this textbook is an attempt at a concise, yet thorough, coverage of the topics we feel are essential for the computer science major. We do not feel the best way to do this is by "compartmentalizing" the various topics; therefore, we have chosen a structured, yet integrated approach where each topic is covered in the context of the entire computer system.

As with many popular texts, we have taken a bottom-up approach, starting with the digital logic level and building to the application level that students should be familiar with before starting the class. The text is carefully structured so that the reader understands one level before moving on to the next. By the time the reader reaches the application level, all of the necessary concepts in computer organization and architecture have been presented. Our goal is to allow the students to tie the hardware knowledge covered in this book to the concepts learned in their introductory programming classes, resulting in a complete and thorough picture of how hardware and software fit together. Ultimately, the extent of hardware understanding has a significant influence on software design and performance. If students can build a firm foundation in hardware fundamentals, this will go a long way toward helping them to become better computer scientists.

The concepts in computer organization and architecture are integral to many of the everyday tasks that computer professionals perform. To address the numerous areas in which a computer professional should be educated, we have taken a high-level look at computer architecture, providing low-level coverage only when deemed necessary for an understanding of a specific concept. For example, when discussing ISAs, many hardware-dependent issues are introduced in the context

of different case studies to both differentiate and reinforce the issues associated with ISA design.

The text is divided into eleven chapters and an appendix as follows:

- **Chapter 1** provides a historical overview of computing in general, pointing out the many milestones in the development of computing systems, and allowing the reader to visualize how we arrived at the current state of computing. This chapter introduces the necessary terminology, the basic components in a computer system, the various logical levels of a computer system, and the von Neumann computer model. It provides a high-level view of the computer system, as well as the motivation and necessary concepts for further study.
- **Chapter 2** provides thorough coverage of the various means computers use to represent both numerical and character information. Addition, subtraction, multiplication and division are covered once the reader has been exposed to number bases and the typical numeric representation techniques, including one's complement, two's complement, and BCD. In addition, EBCDIC, ASCII, and Unicode character representations are addressed. Fixed- and floating-point representation are also introduced. Codes for data recording and error detection and correction are covered briefly.
- **Chapter 3** is a classic presentation of digital logic and how it relates to Boolean algebra. This chapter covers both combinational and sequential logic in sufficient detail to allow the reader to understand the logical makeup of more complicated MSI (medium scale integration) circuits (such as decoders). More complex circuits, such as buses and memory, are also included. We have included optimization and Kmaps in a special "Focus On" section.
- **Chapter 4** illustrates basic computer organization and introduces many fundamental concepts, including the fetch-decode-execute cycle, the data path, clocks and buses, register transfer notation, and of course, the CPU. A very simple architecture, MARIE, and its ISA are presented to allow the reader to gain a full understanding of the basic architectural organization involved in program execution. MARIE exhibits the classical von Neumann design, and includes a program counter, an accumulator, an instruction register, 4096 bytes of memory, and two addressing modes. Assembly language programming is introduced to reinforce the concepts of instruction format, instruction mode, data format, and control that are presented earlier. This is not an assembly language textbook and was not designed to provide a practical course in assembly language programming. The primary objective in introducing assembly is to further the understanding of computer architecture in general. However, a simulator for MARIE is provided so assembly language programs can be written, assembled, and run on the MARIE architecture. The two methods of control, hardwiring and microprogramming, are introduced and compared in this chapter. Finally, Intel and MIPS architectures are compared to reinforce the concepts in the chapter.
- **Chapter 5** provides a closer look at instruction set architectures, including instruction formats, instruction types, and addressing modes. Instruction-level

pipelining is introduced as well. Real-world ISAs (including Intel, MIPS, and Java) are presented to reinforce the concepts presented in the chapter.

- **Chapter 6** covers basic memory concepts, such as RAM and the various memory devices, and also addresses the more advanced concepts of the memory hierarchy, including cache memory and virtual memory. This chapter gives a thorough presentation of direct mapping, associative mapping, and set-associative mapping techniques for cache. It also provides a detailed look at overlays, paging and segmentation, TLBs, and the various algorithms and devices associated with each. A tutorial and simulator for this chapter is available on the book's website.
- **Chapter 7** provides a detailed overview of I/O fundamentals, bus communication and protocols, and typical external storage devices, such as magnetic and optical disks, as well as the various formats available for each. DMA, programmed I/O, and interrupts are covered as well. In addition, various techniques for exchanging information between devices are introduced. RAID architectures are covered in detail, and various data compression formats are introduced.
- **Chapter 8** discusses the various programming tools available (such as compilers and assemblers) and their relationship to the architecture of the machine on which they are run. The goal of this chapter is to tie the programmer's view of a computer system with the actual hardware and architecture of the underlying machine. In addition, operating systems are introduced, but only covered in as much detail as applies to the architecture and organization of a system (such as resource use and protection, traps and interrupts, and various other services).
- **Chapter 9** provides an overview of alternative architectures that have emerged in recent years. RISC, Flynn's Taxonomy, parallel processors, instruction-level parallelism, multiprocessors, interconnection networks, shared memory systems, cache coherence, memory models, superscalar machines, neural networks, systolic architectures, dataflow computers, and distributed architectures are covered. Our main objective in this chapter is to help the reader realize we are not limited to the von Neumann architecture, and to force the reader to consider performance issues, setting the stage for the next chapter.
- **Chapter 10** addresses various performance analysis and management issues. The necessary mathematical preliminaries are introduced, followed by a discussion of MIPS, FLOPS, benchmarking, and various optimization issues with which a computer scientist should be familiar, including branch prediction, speculative execution, and loop optimization.
- **Chapter 11** focuses on network organization and architecture, including network components and protocols. The OSI model and TCP/IP suite are introduced in the context of the Internet. This chapter is by no means intended to be comprehensive. The main objective is to put computer architecture in the correct context relative to network architecture.

An appendix on data structures is provided for those situations where students may need a brief introduction or review of such topics as stacks, queues, and linked lists.

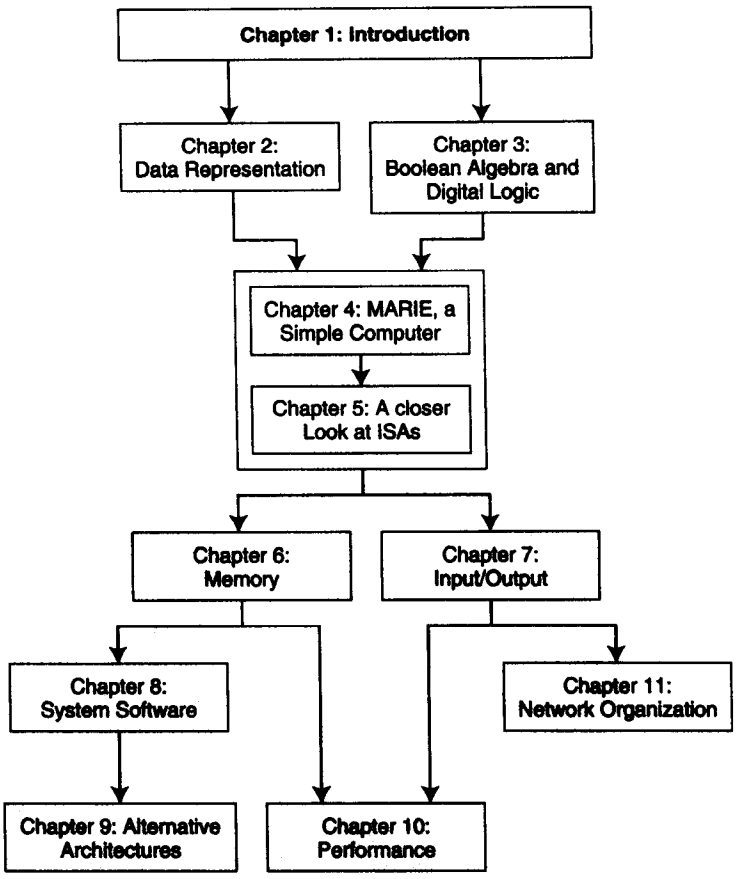


FIGURE P.1 Prerequisite Relationship Among Chapters

The sequencing of the chapters is such that they can be taught in the given numerical order. However, an instructor can modify the order to better fit a given curriculum if necessary. Figure P.1 shows the prerequisite relationships that exist between various chapters.

Intended Audience

This book was originally written for an undergraduate class in computer organization and architecture for computer science majors. Although specifically directed toward computer science majors, the book does not preclude its use by IS and IT majors.

This book contains more than sufficient material for a typical one-semester (14 week, 42 lecture hours) course; however, all of the material in the book cannot be mastered by the average student in a one-semester class. If the instructor