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, 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 cars look like under the hood in order to drive them. 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 trade-offs. 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 the ways that 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 that 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

This book is the outgrowth of two computer science organization and architecture classes taught at Penn State Harrisburg. 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 was 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 about 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 integrated 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 CS2013

In October 2013, the ACM/IEEE Joint Task Force unveiled Computer Science Curricula 2013 (CS2013). Although we are primarily concerned with the Computer Architecture knowledge area, these new guidelines suggest integrating the core knowledge throughout the curriculum. Therefore, we also call attention to additional knowledge areas beyond architecture that are addressed in this book.

CS2013 is a comprehensive revision of CS2008, mostly the result of focusing on the *essential concepts* in the Computer Science curriculum while still being flexible enough to meet individual institutional needs. These guidelines introduce the notion of Core Tier-1 and Core Tier-2 topics, in addition to elective topics. Core Tier-1 topics are those that should be part of every Computer Science curriculum. Core Tier-2 topics are those that are considered essential enough that a Computer Science curriculum should contain 90–100% of these topics. Elective topics are those that allow curricula to provide breadth and depth. The suggested coverage for each topic is listed in lecture hours.

The main change in the Architecture and Organization (AR) knowledge area from CS2008 to CS2013 is a reduction of lecture hours from 36 to 16; however, a new area, System Fundamentals (SF), has been introduced and includes some concepts previously found in the AR module (including hardware building blocks and architectural organization). The interested reader is referred to the CS2013 guidelines (http://www.acm.org/education/curricula-recommendations) for more information on what the individual knowledge areas include.

We are pleased that the fourth edition of *The Essentials of Computer Organization and Architecture* is in direct correlation with the ACM/IEEE CS2013 guidelines for computer organization and architecture, in addition to integrating material from additional knowledge units. Table P.1 indicates which chapters of this textbook satisfy the eight topics listed in the AR knowledge area. For the other knowledge areas, only the topics that are covered in this textbook are listed.

AR – Architecture	**Core Tier 1 Hours**	**Core Tier 2 Hours**	**Includes Electives**	**Chapters**
Digital Logic and Digital Systems		3	N	1, 3, 4
Machine-Level Representation of Data		3	N	1, 2
Assembly-Level Machine Organization		6	N	1, 4, 5, 7, 8, 9
Memory System Organization and Arch		3	N	2, 6, 7, 13
Interfacing and Communication		1	N	4, 7, 12
Functional Organization			Y	4, 5
Multiprocessing and Alternative Archs			Y	9
Performance Enhancements			Y	9, 11
NC – Networking and Communication	**Core Tier 1 Hours**	**Core Tier 2 Hours**	**Includes Electives**	**Chapters**
Introduction	1.5		N	12
Networked Applications	1.5		N	12
Reliable Data Delivery		2	N	12
Routing and Forwarding		1.5	N	12
OS – Operating Systems	**Core Tier 1 Hours**	**Core Tier 2 Hours**	**Includes Electives**	**Chapters**
Overview of Operating Systems	2		N	8
Memory Management		3	N	6
Virtual Machines			Y	8
File Systems			Y	7
Real-Time and Embedded Systems			Y	10
System Performance Evaluations			Y	6, 11
PD – Parallel and Distributed Computing	**Core Tier 1 Hours**	**Core Tier 2 Hours**	**Includes Electives**	**Chapters**
Parallel Architecture	1	1	N	9
Distributed Systems			Y	9
Cloud Computing			Y	1, 9, 13
SF – Systems Fundamentals	**Core Tier 1 Hours**	**Core Tier 2 Hours**	**Includes Electives**	**Chapters**
Computational Paradigms	3		N	3, 4, 9
State and State Machines	6		N	3
Parallelism	3		N	9
Evaluation	3		N	11
Proximity		3	N	6
SP – Social Issues and Professional Practice	**Core Tier 1 Hours**	**Core Tier 2 Hours**	**Includes Electives**	**Chapters**
History			Y	1

TABLE P.1 ACM/IEEE CS2013 Topics Covered in This Book

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 35-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 effect this has on computer science students. Students, however, must have a strong understanding of the basic concepts before they can understand and appreciate the intangible 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, the similarities, and the roles various platforms play in 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:

- *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 of 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 that 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 data compression, 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 35-plus years of combined teaching experience, but also 30-plus 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 holds a PhD in computer science from Iowa State University, an MS in computer science from Iowa State University, an MS in computer science education from Northwest Missouri State University, an MS in mathematics education from Northwest Missouri State University, and a BS in mathematics and English from Northwest Missouri State University. She has been teaching mathematics and computer science for more than 35 years and is currently the computer science graduate program coordinator and associate program chair at the Pennsylvania State University Harrisburg campus, where she has been a member of the faculty since 1995. She has received numerous teaching awards including the Penn State Teaching Fellow Award and the Teaching Excellence Award. Her areas of interest include computer organization and architecture, operating systems, computer science education, and computer security.

Julia Lobur has been a practitioner in the computer industry for more than 30 years. She has held positions as systems consultant, staff programmer/analyst, systems and network designer, software development manager, and project manager, in addition to part-time teaching duties. Julia holds an MS in computer science and is an IEEE Certified Software Development Professional.

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 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 13 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. Codes for data transmission and recording are described in a special "Focus On" section.
- **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 classic 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® Technologies, ARM, 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 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. Various data compression formats are introduced in a special "Focus On" section.
- **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, quantum computing, 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** covers concepts and topics of interest in embedded systems that have not been covered in previous chapters. Specifically, this chapter focuses on embedded hardware and components, embedded system design topics, the basics of embedded software construction, and embedded operating systems features.
- **Chapter 11** 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 12** 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.

- **Chapter 13** introduces some popular I/O architectures suitable for large and small systems, including SCSI, ATA, IDE, SATA, PCI, USB, and IEEE 1394. This chapter also provides a brief overview of storage area networks and cloud computing.
- **Appendix A** is a short appendix on data structures that is provided for those situations in which students may need a brief introduction or review of such topics as stacks, queues, and linked lists.

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.

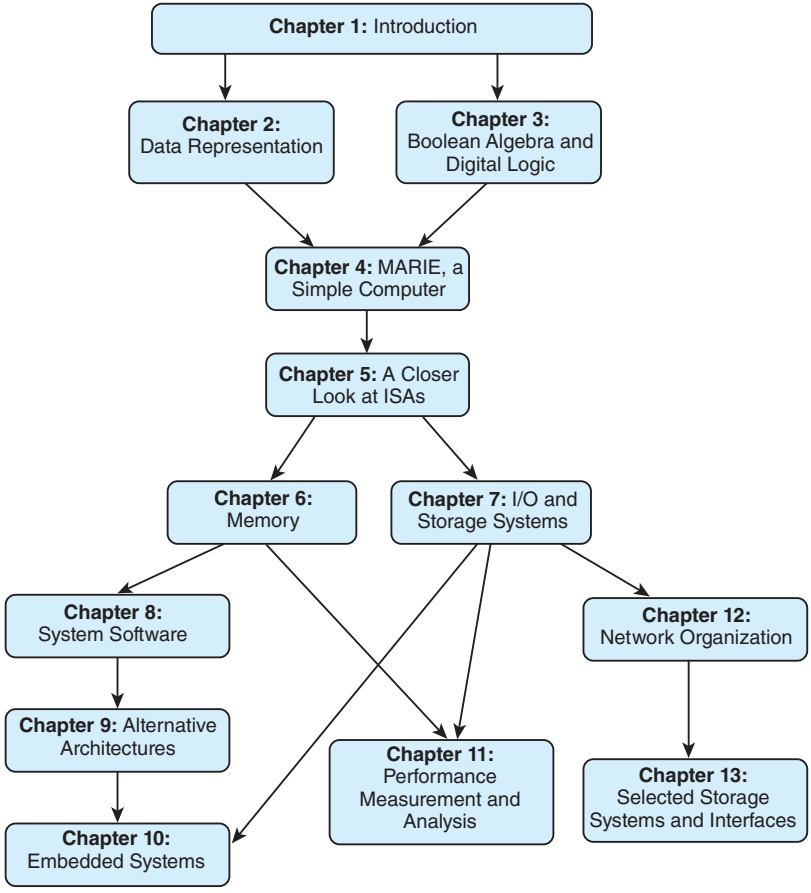

FIGURE P.1 Prerequisite Relationship Between Chapters

What's New in the Fourth Edition

In the years since the third edition of this book was created, the field of computer architecture has continued to grow. In this fourth edition, we have incorporated many of these new changes in addition to expanding topics already introduced in the first three editions. Our goal in the fourth edition was to update content and references, add new material, expand current discussions based on reader comments, and expand the number of exercises in all of the core chapters. Although we cannot itemize all the changes in this edition, the list that follows highlights those major changes that may be of interest to the reader:

- **Chapter 1** has been updated to include new examples and illustrations, tablet computers, computing as a service (Cloud computing), and cognitive computing. The hardware overview has been expanded and updated (notably, the discussion on CRTs has been removed and a discussion of graphics cards has been added), and additional motivational sidebars have been added. The non-von Neumann section has been updated, and a new section on parallelism has been included. The number of exercises at the end of the chapter has been increased by 26%.
- **Chapter 2** contains a new section on excess-M notation. The simple model has been modified to use a standard format, and more examples have been added. This chapter has a 44% increase in the number of exercises.
- **Chapter 3** has been modified to use a prime (′) instead of an overbar to indicate the NOT operator. Timing diagrams have been added to help explain the operation of sequential circuits. The section on FSMs has been expanded, and additional exercises have been included.
- **Chapter 4** contains an expanded discussion of memory organization (including memory interleaving) as well as additional examples and exercises. We are now using the "0x" notation to indicate hexadecimal numbers. More detail has been added to the discussions on hardwired and microprogrammed control, and the logic diagrams for MARIE's hardwired control unit and the timing diagrams for MARIE's microoperations have all been updated.
- **Chapter 5** contains expanded coverage of big and little endian and additional examples and exercises, as well as a new section on ARM processors.
- **Chapter 6** has updated figures, an expanded discussion of associative memory, and additional examples and discussion to clarify cache memory. The examples have all been updated to reflect hexadecimal addresses instead of decimal addresses. This chapter now contains 20% more exercises than the third edition.
- **Chapter 7** has expanded coverage of solid state drives and emerging data storage devices (such as carbon nanotubes and memristors), as well as additional coverage of RAID. There is a new section on MP3 compression and in addition to a 20% increase in the number of exercises at the end of this chapter.
- **Chapter 8** has been updated to reflect advances in the field of system software.

- **Chapter 9** has an expanded discussion of both RISC vs. CISC (integrating this debate into the mobile arena) and quantum computing, including a discussion of the technological singularity.
- **Chapter 10** contains updated material for embedded operating systems.
- **Chapter 12** has been updated to remove obsolete material and integrate new material.
- **Chapter 13** has expanded and updated coverage of USB, expanded coverage of Cloud storage, and removal of obsolete material.

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 weeks, 42 lecture hours) course; however, all the material in the book cannot be mastered by the average student in a one-semester class. If the instructor plans to cover all topics in detail, a two-semester sequence would be optimal. The organization is such that an instructor can cover the major topic areas at different levels of depth, depending on the experience and needs of the students. Table P.2 gives the instructor an idea of the amount of time required to cover the topics, and also lists the corresponding levels of accomplishment for each chapter.

It is our intention that this book serve as a useful reference long after the formal course is complete.

	One Semester (42 Hours)		Two Semesters (84 Hours)	
Chapter	Lecture Hours	Expected Level	Lecture Hours	Expected Level
1	3	Mastery	3	Mastery
2	6	Mastery	6	Mastery
3	6	Mastery	6	Mastery
4	8	Mastery	8	Mastery
5	4	Familiarity	6	Mastery
6	3	Familiarity	8	Mastery
7	2	Familiarity	6	Mastery
8	2	Exposure	7	Mastery
9	2	Familiarity	7	Mastery
10	1	Exposure	5	Familiarity
11	2	Exposure	9	Mastery
12	2	Exposure	7	Mastery
13	1	Exposure	6	Mastery

TABLE P.2 Suggested Lecture Hours

Support Materials

A textbook is a fundamental tool in learning, but its effectiveness is greatly enhanced by supplemental materials and exercises, which emphasize the major concepts, provide immediate feedback to the reader, and motivate understanding through repetition. We have, therefore, created the following ancillary materials for the fourth edition of *The Essentials of Computer Organization and Architecture*:

- *Test bank*.
- *Instructor's Manual*. This manual contains answers to exercises. In addition, it provides hints on teaching various concepts and trouble areas often encountered by students.
- *PowerPoint Presentations*. These slides contain lecture material appropriate for a one-semester course in computer organization and architecture.
- *Figures and Tables*. For those who wish to prepare their own lecture materials, we provide the figures and tables in downloadable form.
- *Memory Tutorial and Simulator*. This package allows students to apply the concepts on cache and virtual memory.
- *MARIE Simulator*. This package allows students to assemble and run MARIE programs.
- *Datapath Simulator*. This package allows students to trace the MARIE datapath.
- *Tutorial Software*. Other tutorial software is provided for various concepts in the book.
- *Companion Website*. All software, slides, and related materials can be downloaded from the book's website:

go.jblearning.com/ecoa4e

The exercises, sample exam problems, and solutions have been tested in numerous classes. The *Instructor's Manual*, which includes suggestions for teaching the various chapters in addition to answers for the book's exercises, suggested programming assignments, and sample example questions, is available to instructors who adopt the book. (Please contact your Jones & Bartlett Learning representative at 1-800-832-0034 for access to this area of the website.)

The Instructional Model: MARIE

In a computer organization and architecture book, the choice of architectural model affects the instructor as well as the students. If the model is too complicated, both the instructor and the students tend to get bogged down in details that really have no bearing on the concepts being presented in class. Real architectures, although interesting, often have far too many peculiarities to make them usable in an introductory class. To make things even more complicated, real architectures change from day to day. In addition, it is difficult to find a book incorporating a model that matches the local computing platform in a given department, noting that the platform, too, may change from year to year.

To alleviate these problems, we have designed our own simple architecture, MARIE, specifically for pedagogical use. MARIE (Machine Architecture that is Really Intuitive and Easy) allows students to learn the essential concepts of computer organization and architecture, including assembly language, without getting caught up in the unnecessary and confusing details that exist in real architectures. Despite its simplicity, it simulates a functional system. The MARIE machine simulator, MarieSim, has a user-friendly GUI that allows students to (1) create and edit source code, (2) assemble source code into machine object code, (3) run machine code, and (4) debug programs.

Specifically, MarieSim has the following features:

- Support for the MARIE assembly language introduced in Chapter 4
- An integrated text editor for program creation and modification
- Hexadecimal machine language object code
- An integrated debugger with single step mode, break points, pause, resume, and register and memory tracing
- A graphical memory monitor displaying the 4096 addresses in MARIE's memory
- A graphical display of MARIE's registers
- Highlighted instructions during program execution
- User-controlled execution speed
- Status messages
- User-viewable symbol tables
- An interactive assembler that lets the user correct any errors and reassemble automatically, without changing environments
- Online help
- Optional core dumps, allowing the user to specify the memory range
- Frames with sizes that can be modified by the user
- A small learning curve, allowing students to learn the system quickly

MarieSim was written in the Java language so that the system would be portable to any platform for which a Java Virtual Machine (JVM) is available. Students of Java may wish to look at the simulator's source code, and perhaps even offer improvements or enhancements to its simple functions.

Figure P.2, the MarieSim Graphical Environment, shows the graphical environment of the MARIE machine simulator. The screen consists of four parts: the menu bar, the central monitor area, the memory monitor, and the message area.

Menu options allow the user to control the actions and behavior of the MARIE simulator system. These options include loading, starting, stopping, setting breakpoints, and pausing programs that have been written in MARIE assembly language.

The MARIE simulator illustrates the process of assembly, loading, and execution, all in one simple environment. Users can see assembly language statements

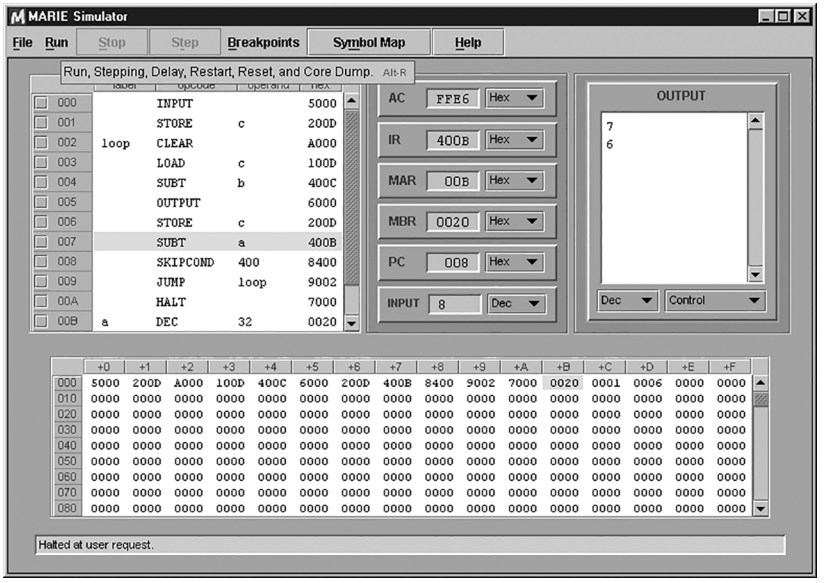

FIGURE P.2 The MarieSim Graphical Environment

directly from their programs, along with the corresponding machine code (hexadecimal) equivalents. The addresses of these instructions are indicated as well, and users can view any portion of memory at any time. Highlighting is used to indicate the initial loading address of a program in addition to the currently executing instruction while a program runs. The graphical display of the registers and memory allows the student to see how the instructions cause the values in the registers and memory to change.

If You Find an Error

We have attempted to make this book as technically accurate as possible, but even though the manuscript has been through numerous proofreadings, errors have a way of escaping detection. We would greatly appreciate hearing from readers who find any errors that need correcting. Your comments and suggestions are always welcome; please send on email to ECOA@jblearning.com.

Credits and Acknowledgments

Few books are entirely the result of one or two people's unaided efforts, and this one is no exception. We realize that writing a textbook is a formidable task and only possible with a combined effort, and we find it impossible to adequately thank those who have made this book possible. If, in the following acknowledgments, we inadvertently omit anyone, we humbly apologize.

A number of people have contributed to the fourth edition of this book. We would first like to thank all of the reviewers for their careful evaluations

of previous editions and their thoughtful written comments. In addition, we are grateful for the many readers who have emailed useful ideas and helpful suggestions. Although we cannot mention all of these people here, we especially thank John MacCormick (Dickinson College) and Jacqueline Jones (Brooklyn College) for their meticulous reviews and their numerous comments and suggestions. We extend a special thanks to Karishma Rao and Sean Willeford for their time and effort in producing a quality memory software module.

We would also like to thank the individuals at Jones & Bartlett Learning who worked closely with us to make this fourth edition possible. We are very grateful to Tiffany Silter, Laura Pagluica, and Amy Rose for their professionalism, commitment, and hard work on the fourth edition.

I, Linda Null, would personally like to thank my husband, Tim Wahls, for his continued patience while living life as a "book widower" for a fourth time, for listening and commenting with frankness about the book's contents and modifications, for doing such an extraordinary job with all of the cooking, and for putting up with the almost daily compromises necessitated by my writing this book—including missing our annual fly-fishing vacation and forcing our horses into prolonged pasture ornament status. I consider myself amazingly lucky to be married to such a wonderful man. I extend my heartfelt thanks to my mentor, Merry McDonald, who taught me the value and joys of learning and teaching, and doing both with integrity. Lastly, I would like to express my deepest gratitude to Julia Lobur, as without her, this book and its accompanying software would not be a reality. It has been both a joy and an honor working with her.

I, Julia Lobur, am deeply indebted to my lawful spouse, Marla Cattermole, who married me despite the demands that this book has placed on both of us. She has made this work possible through her forbearance and fidelity. She has nurtured my body through her culinary delights and my spirit through her wisdom. She has taken up my slack in many ways while working hard at her own career. I would also like to convey my profound gratitude to Linda Null: first, for her unsurpassed devotion to the field of computer science education and dedication to her students and, second, for giving me the opportunity to share with her the ineffable experience of textbook authorship.