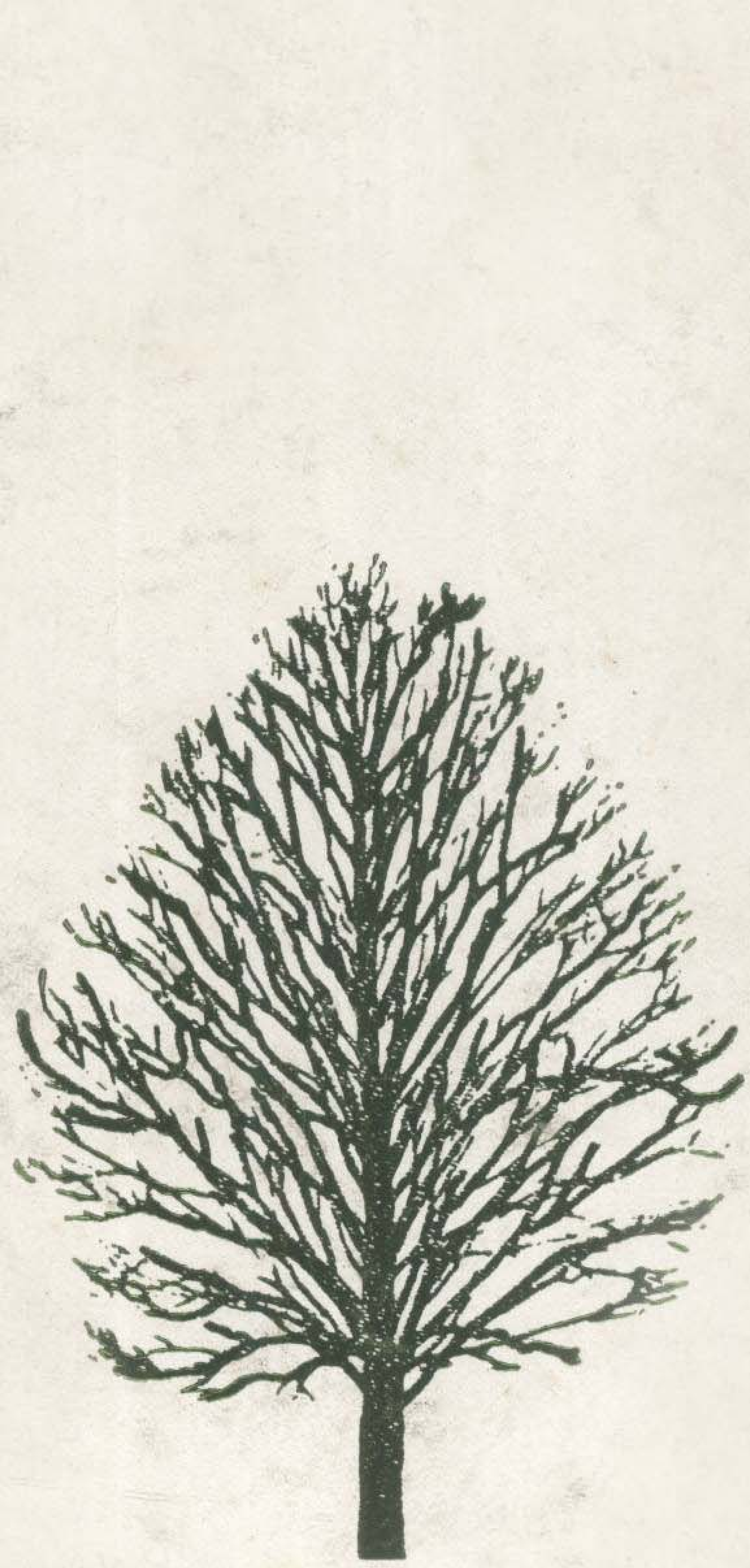
DAVID J. KUCK

THE STRUCTURE OF Computers and Computations

VOLUME ONE



THE STRUCTURE OF COMPUTERS AND COMPUTATIONS

VOLUME 1

DAVID J. KUCK
Professor of Computer Science
University of Illinois at Urbana-Champaign

JOHN WILEY & SONS
New York • Santa Barbara
Chichester • Brisbane • Toronto

Library of Congress Cataloging in Publication Data:

Kuck, David J
The structure of computers and computations.
Includes bibliographical references and index.
1. Computer architecture. I. Title.
QA76.9.A73K83   621.3819'52   78-5412
ISBN 0-471-02716-2

THE STRUCTURE OF COMPUTERS AND COMPUTATIONS

NOTATION

The following notational conventions are used throughout the book:

$\log x = \log_2 x$

$f(x) = O(g(x))$ means there is a constant $r > 0$ such that $\lim_{x\to\infty} (f(x)/g(x)) = r$.

$\lceil x \rceil$, the ceiling of x is the smallest integer greater than or equal to x.

$\lfloor x \rfloor$, the floor of x, is the greatest integer less than or equal to x.

$\log^2 x = (\log x) \cdot (\log x)$

$\mathrm{K} = 1024$

P denotes the fewest processors needed to obtain the best known speedup of some computation, while p, $1 \leq p < P$, denotes any number of processors used for some computation.

PREFACE

Art is I
Science is We
Engineering is They

Much ado has been made about whether the design of computer hardware and software is art, science or engineering. The above expansion of Claude Bernard's "Art is I, Science is We" seems to capture the essence of the matter. Computer system design is all three.

A good computer system designer must first heed the problems of the intended system users, the technology from which the system will be built, and the people who will carry out the construction and programming of his design. All of these considerations form the "They" of the above quotation and constitute the engineering aspects of computer system design.

Various physical and mathematical principles must be followed in carrying out the design of a system. The "We" of the above quotation represents the collection of people who, as computer system design has evolved, have developed various guiding principles. Both theoretical work and experimental work have been done in a scientific way, to give us a useful set of computer system design principles.

Finally, with "They" and "We" in mind, the individual designer of a piece of hardware or software must employ his own creative talents to produce a new design. The "I" of the above quotation often creates a distinctive, individual design. The "artistry" of designers is obvious when one considers the variety of different types of machines and software sold to the same market. Indeed, the artistry extends to whole companies, which develop "corporate styles" of hardware and software design.

As we discuss throughout this book, there is no "one best way" to design a computer system. Many tradeoffs within and between the artistic, scientific, and engineering levels must be made. The goal of this book is to clarify these tradeoffs by pointing out what they are, how they can be analyzed, and how they affect one another. Abstractions as well as practical computer design topics are considered.

The title of this book describes its contents and its purpose. The structures of both computers and their computations are examined, and the importance of understanding the close ties between them is emphasized. A computer's structure obviously consists of a collection of interconnected hardware devices.

But, despite the fact that most computers are called "general-purpose machines," in recent years computers have become more and more specialized. This specialization is reflected in their physical and logical structure, but it is motivated by the uses to which various machines are put. Indeed, the computations to be performed by a machine *should* lead designers to the structure (or architecture or organization) of the machine. Examples of different structures are minicomputers, supercomputers, and time-shared computers.

This leads us to the second half of the title: "The Structure of Computations." Although this phrase may be less exact than "The Structure of Computers," the different uses of the kinds of computers mentioned above do indeed lead to computations of widely different structures. Examples are the dominance of short, integer arithmetic versus long, floating point arithmetic; the heavy use of scalars, one-, two-, or more-dimensional arrays, or trees as data items; the sizes of these data structures and the amount of input and output required; the relative frequency of branching instructions, arithmetic, stores and fetches; and the users' view of the machine as a quick response, small-problem solver or as an overnight, large-problem solver. Each of these features, and many of the other features discussed in the book are closely related to the structure of the computer itself. Obviously, a machine should be tailored to match the characteristics of the computations that it is to perform.

Most of the qualities mentioned above are useful for characterizing the structure of an individual user's program or computation. Also, certain characterizations of the structure of the system software (compilers and operating systems) are relevant here. For example, hardware features that can aid in compilation include various types of word formats and arithmetic (for number conversion), push-down stacks (for parsing), and hardware register management (to avoid scheduling complexities). Operating systems can be greatly aided by hardware for handling interrupts conveniently and by virtual memory and other input-output hardware, and the like. Again, the type of software anticipated for the system may be an important consideration in designing the computer's architecture.

The theme emphasized throughout the book is that unless computer system designers appreciate and understand both the structure of the computer and the structure of its computations, serious design errors are likely to result. They may show up in the cost, speed, usefulness, or reliability of the resulting hardware and software system. I do not claim that such errors can be totally avoided, but the book's goal is to spell out the issues and to clarify the tradeoffs and solutions that are known. These points are important for software designers as well as hardware designers. The book discusses a number of structural features of computers that must be taken into account by compiler writers and emphasizes the program transformations that form the basis of compiler algorithms for various computer structures.

GOALS AND ORIENTATION

This is a book about how computer systems work and why they are organized as they are. It includes the history of real machines to explain how system organization arrived at its present position, and also projects what may lie ahead, based on current knowledge. The purpose is to provide the reader with facts about *how* things are as well as good intuition about *why* they are that way.

This is not a book about logic design, at least not in the traditional sense. Nor is it an attempt to explain a number of different, real computer systems. However, I believe that readers will come away from the book with a good idea of the important issues of logic design as well as a clear understanding of the operating principles of most current computer systems.

Traditional logic design books are concerned with abstractions about Boolean algebra and switching theory, and with concrete examples of many different kinds of useful circuits. However, most of these abstractions are of little use in modern logic design, and the study of half a dozen different adder circuits, for example, overkills such a problem. In this book I instead develop some theory that can be used to estimate the speed and component counts in a wide variety of useful circuits. With this background, the relative costs and speeds of different designs for a given circuit or for various parts of a processor can be easily estimated. This is just one example of our approach to the material.

Books that explain real computer systems in detail are valuable references, but it is often difficult to extract any general principles from them. In this book, individual chapters on processors, control units, memories, interconnection switches, and memory hierarchies provide background material, with examples drawn from real systems. Some principles are also given to explain why these devices take the various forms they do. Thus readers will be able to compare various machine features in a detailed way, and they will also be able to compare these features with certain absolute measures of speed and cost. The quality and usefulness of each of these parts of computer systems are also discussed in each chapter.

Frequent references are given throughout to manufacturer's model numbers, even though there is a risk of being outdated by new announcements. Actually, most manufacturers make few fundamental changes in a 5- to 10-year period. Thus, the earliest model numbers usually refer to a sequence of machine families. For example, the IBM System/360 and System/370 had few fundamental differences, and I sometimes use "IBM 360" to designate all of these. Similarly, I refer to the CDC 6600 and 7600 frequently, but without mentioning the more recent but structurally similar CYBER series machines. Also, I use the PDP-11 to reflect the DEC minicomputers.

The machines I discuss as examples are typical of those sold by major manufacturers. Of course, much of the computer market involves specialpurpose digital devices or a dedicated use of general-purpose machines. Frequent examples are given of unusual machines that are effective in particular areas. For a survey of recent high speed computers, including discussions of the structure of their computations and performance, see [KuLS 77].

It is expected that users of this book will have a good background in programming computers. The intuition about how computers work, which is gained in this way, makes it feasible to avoid various detailed explanations. No background in logic design is assumed.

In Chapter 1, I first present criteria by which computer systems can be judged. Next, we attempt to coordinate the backgrounds of all readres with a brief discussion of some elementary hardware and software topics. As a preview of the later chapters, I then give an overview of computer system organization. A rather detailed history of computers is given, which provides some interesting and useful facts about real systems. Moreover, knowledge about how much had been built before 1950—and, indeed, how much was known before 1850—should instill a good deal of humility in all computer people. I conclude the chapter with an overview of the rest of the book.

All readers should scan Chapter 1 to determine the kind of background material that is assumed. Selective reading should be used to fill in any gaps in background at the beginning or later during the study of particular chapters. This leads to an important question: For whom is this book intended?

For university courses concerned with computer organization, computer design, or computer architecture, this book can serve as a text. These are the courses that are usually designed for advanced undergraduates and beginning graduate students in computer science or electrical engineering. There is more material in this book than can be included in a normal one-semester course. In fact, depending on local circumstances, there is enough for two quarters or perhaps two semesters; if enhanced by readings from research papers, parts of the book can be the basis for an advanced graduate course on computer system theory and design.

The students who take an introductory "computer organization" course may have one of several motivations. First, they may be interested in a career as a computer system designer. For these individuals the book provides a broad, solid background for further work. A second type of student may be interested in system software design, including compilers and operating systems. For such students it is very important to have a good working knowledge of the details of computer systems. The hardware background provided here should carry over to aid system programming on any real computer. Finally, there are students who are interested in computers from the point of view of machine users or owners. That is, they want to be aware of tradeoffs that exist in comparing machines, and they want to have a general background about the architecture of computers. By selectively reading this book, they should be able to meet these goals.

The statements above, although explicitly concerned with university students, are equally valid for professional people who are not formally students. Sufficient background material is provided throughout so that the book can be used in a self-study mode.

It is important to realize that there is a great deal of interdependence between various parts of this book. For example, the control unit chapter assumes, from time to time, that the processor chapter has been read, the memory hierarchy chapter assumes that a number of previous chapters have been read, and so on. A number of explicit cross-references are provided in the chapter discussion, and I suggest that if a reader is confused, he or she consult the book's index or table of contents for earlier references to background material. Since a goal of the book is to present an integrated and coherent view of the structure and operation of whole computer systems, such interdependence seems a necessary and, indeed, important fact of the book's organization.

Chapters 3 to 7 discuss various parts of computers: processors, control units, main memories, interconnection networks, and memory hierarchies, respectively. Chapter 2 presents some theoretical background that reappears at several places throughout the book. Most students at the level for which this book is intended may find Chapter 2 "hard" or "unusual," and to shorten the time spent on it this chapter may be treated lightly. However, it should not be avoided entirely. The introduction is fairly long and gives the idea of the chapter. To reflect the type of course being taught, this and selections from the four following sections may be chosen.

Much of the material in Chapters 3 to 7, with Chapter 1 used as needed and parts of Chapter 2 used to set as much of a theoretical background as is desired, can be profitably used in courses given in computer science and electrical engineering. More comments about the book's contents appear in Section 1.4. of Chapter 1.

The homework problems provided with each chapter are given in two groups—"easy" and "medium and hard." Within these groups they are arranged roughly to follow the sequence of the chapter discussion. Some of the problems are theoretical and others are design-type exercises. Any finer distinction than "easy" and "medium and hard" would be difficult because of the diversity of the subject matter; the backgrounds of individuals have a wide range. A number of the problems are included to fill in gaps in the text, where it seems appropriate to expect students to be able to fill in these gaps. For example, some problems relate the general ideas of the chapter discussion to one specific computer or another. In this case, the problems are intended to be self-contained, but for more background about one machine or another, manufacturers' literature may be studied.

Many of the problems have been written in the spirit of their accompanying chapter, with certain built-in assumptions. This is sometimes necessary to avoid even longer problem statements. Students should make and state any assumptions needed to make the problem clear for solution. The problem statements contain numbers that are more or less realistic; however, simplifications and idealizations are often used. Detailed solutions to all of the problems may be found in the Instructor's Manual.

Only a few problems are included in Chapter 1, since this chapter is intended as background material. However, for a good understanding of the material in this book, a number of problems should be solved in each of the succeeding chapters.

David J. Kuck

Many complain that the words of the wise are always merely parables
    and of no use in daily life,
    which is the only life we have.
All these parables really set out to say
    merely that the incomprehensible is incomprehensible,
    and we know that already,
But the cares we have to struggle with every day:
    that is a different matter.

Concerning this a man once said: Why such reluctance?
If you only followed the parables you yourselves would become parables
    and with that rid of all your daily cares.

Another said: I bet that is also a parable

The first said: You have won.

The second said: But unfortunately only in parable.

The first said: No, in reality: in parable you have lost.

On Parables
Franz Kafka

ACKNOWLEDGMENTS

I am indebted to many people for help with ideas that led to parts of this book as well as for help with the material as it appears in the book. First, I thank the many anonymous students and teaching assistants in CS 333 (and its predecessors) at the University of Illinois who asked questions and made suggestions that improved my notes over the past ten years. I am also indebted for ideas or help with the presentation of the book's material to Walid Abu-Sufah, Dan Atkins, Utpal Banerjee, Richard Brent, Paul Budnik, Roy Campbell, Donald Chang, Milos Ercegovac, Dan Gajski, Lee Hollaar, Sharon Kuck, B. Kumar, Bruce Leasure, Kiyoshi Maruyama, Yoichi Muraoka, David Padua, Stott Parker, John Pasta, Michael Schlansker, William Stellhorn, Richard Stokes, Ross Towle, and Michael Wolfe.

The entire manuscript was read by Richard Brown, Won Kim, and Duncan Lawrie, and they made many very valuable suggestions about the presentation. Robert Kuhn made many original contributions to the homework problems and helped put all of the problems in their final form. I am especially indebted to S. C. Chen, Duncan Lawrie, and Ahmed Sameh, for background ideas, comments about the book, and endless discussions about the material.

Finally, my sincerest thanks are reserved for Vivian Alsip, who typed the manuscript several times and suffered without complaint a good deal of abuse. She always produced the highest quality drawings and typing, quickly and pleasantly.

D. K.

CH 1

However, if I had waited long enough I probably would never have written anything at all since there is a tendency when you really begin to learn something about a thing
not to want to write about it but rather to keep on learning about it always and at no time, unless you are very egotistical, which, of course, accounts for many books,
will you be able to say: now I know all about this and will write about it.

Certainly I do not say that now; every year I know there is more to learn, but I know some things which may be interesting now,
and I may be away from the bullfights for a long time and I might as well write what I know about them now.

Death in the Afternoon

Ernest Hemingway

Anyone who says he knows how computers should be built should have his head examined!
The man who says it is either inexperienced or really mad.

Computer Architecture

James E. Thornton

CONTENTS