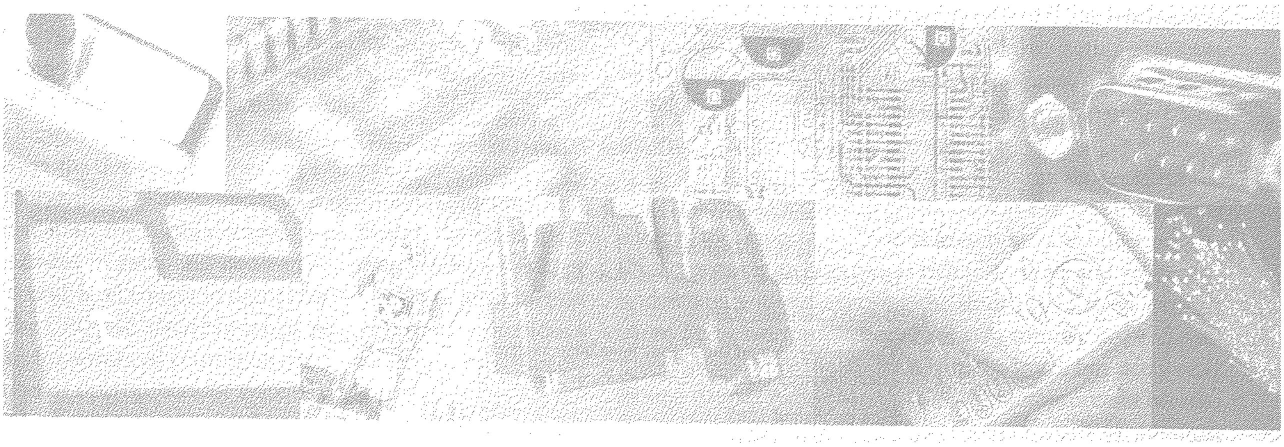
PRINCIPLES OF COMPUTER HARDWARE



Alan Clements
School of Computing
University of Teesside

Fourth Edition

OXFORD
UNIVERSITY PRESS

PREFACE

Principle of Computer Hardware is aimed at students taking an introductory course in electronics, computer science, or information technology. The approach is one of *breadth before depth* and we cover a wide range of topics under the general umbrella of *computer hardware.*

I have written *Principles of Computer Hardware* to achieve two goals. The first is to teach students the basic concepts on which the stored-program digital computer is founded. These include the representation and manipulation of information in binary form, the structure or *architecture* of a computer, the flow of information within a computer, and the exchange of information between its various peripherals. We answer the questions, 'How does a computer work', and 'How is it organized?' The second goal is to provide students with a foundation for further study. In particular, the elementary treatment of gates and Boolean algebra provides a basis for a second-level course in digital design, and the introduction to the CPU and assembly-language programming provides a basis for advanced courses on computer architecture/organization or microprocessor systems design.

This book is written for those with no previous knowledge of computer architecture. The only background information needed by the reader is an understanding of elementary algebra. Because students following a course in computer science or computer technology will also be studying a high-level language, we assume that the reader is familiar with the concepts underlying a high-level language.

When writing this book, I set myself three objectives. By adopting an informal style, I hope to increase the enthusiasm of students who may be put off by the formal approach of more traditional books. I have also tried to give students an insight into computer hardware by explaining why things are as they are, instead of presenting them with information to be learned and accepted without question. I have included subjects that would seem out of place in an elementary first-level course. Topics like advanced computer arithmetic, timing diagrams, and reliability have been included to show how the computer hardware of the real world often differs from that of the first-level course in which only the basics are taught. I've also broadened the range of topics normally found in first-level courses in computer hardware and provided sections introducing operating systems and local area networks, as these two topics are so intimately related to the hardware of the computer. Finally, I have discovered that stating a formula or a theory is not enough—many students like to see an actual application of the formula. Wherever possible I have provided examples.

Like most introductory books on computer architecture, I have chosen a specific microprocessor as a vehicle to illustrate some of the important concepts in computer architecture. The ideal computer architecture is rich in features and yet easy to understand without exposing the student to a steep learning curve. Some microprocessors have very complicated architectures that confront the students with too much fine detail early in their course. We use Motorola's 68K microprocessor because it is easy to understand and incorporates many of the most important features of a high-performance architecture. This book isn't designed to provide a practical assembly language programming course. It is intended only to illustrate the operation of a central processing unit by means of a typical assembly language. We also take a brief look at other microprocessors to show the range of computer architectures available.

You will see the words *computer*, CPU, *processor, microprocessor*, and *microcomputer* in this and other texts. The part of a computer that actually executes a program is called a CPU (central processing unit) or more simply a *processor.* A *microprocessor* is a CPU fabricated on a single chip of silicon. A computer that is constructed around a *microprocessor* can be called a *microcomputer.* To a certain extent, these terms are frequently used interchangeably.

READING GUIDE

We've already said that this book provides a traditional introductory course in computer architecture plus additional material to broaden its scope and fill in some of the gaps left in such courses. To help students distinguish between foreground and background material, the following guide will help to indicate the more fundamental components of the course.

Chapter 2 introduces the logic of computers and deals with essential topics such as gates, Boolean algebra, and Karnaugh maps. Therefore this chapter is essential reading.

Chapter 3 introduces *sequential circuits* such as the counter that steps through the instructions of a program and demonstrates how sequential circuits are designed. We first introduce the *bistable* (flip-flop) used to construct *sequential circuits* such as registers and counters. We don't provide a comprehensive introduction to the design of sequential circuits; we show how gates and flip-flops can be used to create a computer.

Chapter 4 deals with the representation of numbers and shows how arithmetic operations are implemented. Apart from some of the coding theory and details of multiplication and division, almost all this chapter is essential reading. Multiplication and division can be omitted if the student is not interested in how these operations are implemented.

Chapter 5 is the heart of the book and is concerned with the structure and operation of the computer itself. We examine the instruction set of a processor with a sophisticated architecture.

Chapter 6 provides an overview of assembly language programming and the design of simple 68K assembly language programs. This chapter relies heavily on the 68K cross-assembler and simulator provided with the book. You can use this software to investigate the behavior of the 68K on a PC.

Chapter 7 begins with a description of the functional units that make up a computer and the flow of data during the execution of an instruction. We then describe the operation of the computer's control unit, which decodes and executes instructions. The control unit may be omitted on a first reading. Although the control unit is normally encountered in a second- or third-level course, we've included it here for the purpose of completeness and to show how the computer turns a binary-coded instruction into the sequence of events that carry out the instruction.

Chapter 8 is concerned with the quest for performance. We look at how performance is measured and describe three techniques used to accelerate processors. All students should read about the first two acceleration techniques, pipelining and cache memory, but may omit parallel processing.

Chapter 9 describes two contrasting computer architectures. Introductory texts on computer architecture are forced to concentrate on one processor because students do not have the time to plow through several different instruction sets. However, if we don't cover other architectures, students can end the course with a rather unbalanced view of processors. In this chapter we provide a very brief overview of several contrasting processors. We do not expect students to learn the fine details of these processors. The purpose of this chapter is to expose students to the range of processors that are available to the designer.

Chapter 10 deals with input/output techniques. We are interested in the way in which information is transferred between a computer and peripherals. We also examine the buses, or data highways, along which data flows. This chapter is essential reading.

Chapter 11 introduces some of the basic peripherals you'd find in a typical PC such as the keyboard, display, printer, and mouse, as well as some of the more unusual peripherals that, for example, can measure how fast a body is rotating. Although these topics are often omitted from courses in computer hardware, students should scan this chapter to get some insight into how computers control the outside world.

Chapter 12 looks at the memory devices used to store data in a computer. Information isn't stored in a computer in just one type of storage device. It's stored in DRAM and on disk, CD-ROM, DVD, and tape. This chapter examines the operating principles and characteristics of the storage devices found in a computer. There's a lot of detail in this chapter. Some readers may wish to omit the design of memory systems (for example, address decoding and interfacing) and just concentrate on the reasons why computers have so many different types of memory.

Chapter 13 deals with hardware topics that are closely related to the computer's operating system. The two most important elements of a computer's hardware that concern the operating system are *multiprogramming* and *memory management*. These topics are intimately connected with interrupt handling and data storage techniques and serve as practical examples of the use of the hardware described elsewhere. Those who require a basic introduction to computer hardware may omit this chapter, although it best illustrates how hardware and software come together in the operating system.

Chapter 14 describes how computers can communicate with each other. The techniques used to link computers to create computer networks are not always covered by first-level texts on computer architecture. However, the growth of both local area networks and the Internet have propelled computer communications to the forefront of computing. For this reason we would expect students to read this chapter even if some of it falls outside the scope of their syllabus.

THE HISTORY OF THIS BOOK

Like people, books are born. *Principles of Computer Hardware* was conceived in December 1980. At the end of their first semester our freshmen were given tests to monitor their progress. The results of the test in my 'Principles of computer hardware' course were not as good as I'd hoped, so I decided to do something about it. I thought that detailed lecture notes written in a style accessible to the students would be the most effective solution.

Having volunteered to give a course on computer communications to the staff of the Computer Center during the Christmas vacation, I didn't have enough free time to produce the notes. By accident I found that the week before Christmas was the cheapest time of the year for vacations. So I went to one of the Canary Islands for a week, sat down by the pool, surrounded by folders full of reference material, with a bottle of Southern Comfort, and wrote the core of this book—number bases, gates, Boolean algebra, and binary arithmetic. Shortly afterwards I added the section on the structure of the CPU.

These notes produced the desired improvement in the end-of-semester exam results and were well received by the students. In the next academic year my notes were transferred from paper to a mainframe computer and edited to include new material and to clean up the existing text.

I decided to convert the notes into a book. The conversion process involved adding topics, not covered by our syllabus, to produce a more rounded text. While editing my notes, I discovered what might best be called the *inkblot effect.* Text stored in a computer tends to expand in all directions because it's so easy to add new material at any point; for example, you might write a section on disk drives. When you next edit the section on disks, you can add more depth or breadth.

The final form of this book took a *breadth before depth* approach. That is, I covered a large number of topics rather than treating fewer topics in greater depth. It was my intention to give students taking our introductory hardware/architecture course a reasonably complete picture of the computer *system.*

The first edition of *Principles of Computer Hardware* proved successful and I was asked to write a second edition, which was published in 1990. The major change between the first and second editions was the adoption of the 68K microprocessors as a vehicle to teach computer architecture. I have retained this processor in the current edition. Although members of the Intel family have become the standard processors in the PC world, Motorola's 68K family of microprocessors is much better suited to teaching computer architecture. In short, it supports most of the features that computer scientists wish to teach students, and just as importantly, it's much easier to understand. The 68K family and its derivatives are widely used in embedded systems.

By the mid-1990s the second edition was showing its age. The basic computer science and the underlying principles were still fine, but the actual hardware had changed dramatically over a very short time. The most spectacular progress was in the capacity of hard disks—by the late 1990s disk capacity was increasing by 60% per year.

This third edition included a 68K cross-assembler and simulator allowing students to create and run 68K programs on any PC. It also added details of interesting microprocessor architecture, the ARM, which provides an interesting contrast to the 68K.

When I used the second edition to teach logic design to my students, they built simple circuits using *logic trainers*—boxes with power supplies and connectors that allow you to wire a handful of simple chips together. Dave Barker, one of my former students, has constructed a logic simulator program as part of his senior year project called Digital Works, which runs under Windows on a PC. Digital Works allows you to place logic elements anywhere within a window and to wire the gates together. Inputs to the gates can be provided manually (via the mouse) or from clocks and sequence generators. You can observe the outputs of the gates on synthesized LEDs or as a waveform or table. Moreover, Digital Works permits you to encapsulate a circuit in a macro and then use this macro in other circuits. In other words, you can take gates and build simple circuits, and take the simple circuits and build complex circuits, and so on.

I began writing a fourth edition of this text in late 2003. The fundamental principles have changed little since the third edition, but processors had become faster by a factor of 10 and the capacity of hard disks has grown enormously. This new edition is necessary to incorporate some of the advances. After consultation with those who adopt this book, we have decided to continue to use the 68K family to introduce the computer instruction set because this processor still has one of the most sophisticated of all instruction set architectures.

ACKNOWLEDGEMENTS

Few books are entirely the result of one person's unaided efforts and this is no exception. I would like to thank all those who wrote the books about computers on which my own understanding is founded. Some of these writers conveyed the sheer fascination of computer architecture that was to change the direction of my own academic career. It really is amazing how a large number of gates (a circuit element whose operation is so simple as to be trivial) can be arranged in such a way as to perform all the feats we associate computers with today.

I am grateful for all the comments and feedback I've received from my wife, colleagues, students, and reviewers over the years. Their feedback has helped me to improve the text and eliminate some of the errors I'd missed in editing. More importantly, their help and enthusiasm has made the whole project worthwhile.

Although I owe a debt of gratitude to a lot of people, I would like to mention four people who have had a considerable impact. Alan Knowles of Manchester University read drafts of both the second and third editions with a precision well beyond that of the average reviewer. Paul Lambert, one of my colleagues at The University of Teesside, wrote the 68K cross-assembler and simulator that I use in my teaching. In this edition we have used a Windows-based graphical 68K simulator kindly provided by Charles Kelly.

Dave Barker, one of my former students and an excellent programmer, wrote the logic simulator called Digital Works that accompanies this book. I would particularly like to thank Dave for providing a tool that enables students to construct circuits and test them without having to connect wires together.

One of the major changes to the third edition was the chapter on the ARM processor. I would like to thank Steve Furber of Manchester University (one of the ARM's designers) for encouraging me to use this very interesting device.

CONTENTS