Processor Architecture

Springer-Verlag Berlin Heidelberg GmbH

Jurij Šilc • Borut Robič • Theo Ungerer

Processor Architecture

From Dataflow to Superscalar and Beyond

With 132 Figures and 34 Tables

Springer

Dr. Jurij Šilc
Computer Systems Department
Jožef Stefan Institute
Jamova 39
SI-1001 Ljubljana, Slovenia

Assistant Professor
Dr. Borut Robič
Faculty of Computer and Information Science
University of Ljubljana
Tržaška cesta 25
SI-1001 Ljubljana, Slovenia

Professor Dr. Theo Ungerer
Department of Computer Design and Fault Tolerance
University of Karlsruhe
P.O. Box 6980
D-76128 Karlsruhe, Germany

Library of Congress Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek – CIP-Einheitsaufnahme

Šilc, Jurij:
Processor architecture: from dataflow to superscalar and beyond/
Jurij Šilc; Borut Robič; Theo Ungerer. - Berlin; Heidelberg; New
York; Barcelona; Hong Kong; London; Milan; Paris; Singapore;
Tokyo: Springer, 1999
ISBN 978-3-540-64798-0 ISBN 978-3-642-58589-0 (eBook)
DOI 10.1007/978-3-642-58589-0

Originally published by Springer-Verlag Berlin Heidelberg New York in 1999

Typesetting: Camera ready pages by the authors
Cover Design: Künkel + Lopka, Werbeagentur, Heidelberg
Printed on acid-free paper SPIN 10665103 33/3142 – 5 4 3 2 1 0

To my father, Alfonz,
my wife, Marjetka, and my sons, Tomaž & Jaka.

Jurij

To Marinka, Gregor, and Rebeka
for their patience, support, and love

Borut

To my wife Gertraud

Theo

Preface

Today's microprocessors are the powerful descendants of the von Neumann computer dating back to a memo[1] of Burks, Goldstine, and von Neumann of 1946. The so-called von Neumann architecture is characterized by a sequential control flow resulting in a sequential instruction stream. A program counter addresses the next instruction if the preceding instruction is not a control instruction such as, e.g., jump, branch, subprogram call or return. An instruction is coded in an instruction format of fixed or variable length, where the opcode is followed by one or more operands that can be data, addresses of data, or the address of an instruction in the case of a control instruction. The opcode defines the types of operands. Code and data are stored in a common storage that is linear, addressed in units of memory words (bytes, words, etc.).

The overwhelming design criterion of the von Neumann computer was the minimization of hardware and especially of storage. The most simple implementation of a von Neumann computer is characterized by a microarchitecture that defines a closely coupled control and arithmetic logic unit (ALU), a storage unit, and an I/O unit, all connected by a single connection unit. The instruction fetch by the control unit alternates with operand fetches and result stores for the ALU. Both fetches access the same storage and are performed over the same connection unit – this turned out to be a bottleneck, sometimes coined by latter authors as the von Neumann bottleneck.

The sequential operating principle of the von Neumann architecture is still the basis for today's most widely used high-level programming languages, and even more astounding, of the instruction sets of all modern microprocessors. While the characteristics of the von Neumann architecture still determine those of a contemporary microprocessor, its internal structure has considerably changed. The main goal of the von Neumann design – minimal hardware structure – is today far outweighed by the goal of maximum performance. However, the architectural characteristics of the von Neumann design are still valid due to the sequential high-level programming languages that are used today and that originate in the von Neumann architecture paradigm.

[1] A.P. Burks, H.H. Goldstine, J. von Neumann, Preliminary Discussion of the Logical Design of an Electronic Computing Instrument. Report to the U.S. Army Ordnance Department, 1946. Reprint in: W. Aspray, A.P. Burks (eds.) *Papers of John von Neumann.* MIT Press, Cambridge, MA, 1987, pages 97–146.

A slightly more efficient implementation than the alternating of instruction fetch and operand fetch is the overlapping of the following two steps: next PC computation and instruction fetch and decode, with operand fetch, instruction execution, and result storage. This overlapping already defines two-stage instruction pipelining.

A more consistent use of overlapping results in an instruction pipeline with the following basic steps that are characteristic of so-called reduced instruction set computer (RISC) processors: instruction fetch, instruction decode and operand fetch, instruction execution, memory access in the case of a load/store instruction, and result write-back. Ideally each step takes about the same amount of time.

However, a storage access today needs much more time than a single pipeline step. The introduction of registers on the processor chip and restricting the operands of ALU instructions to register accesses allows the pipeline to be balanced again. However, the problem of the memory accesses – the von Neumann bottleneck – is still one of the main hindrances to high performance even today. A whole memory hierarchy of cache storages now exists to widen that bottleneck.

Current superscalar microprocessors are a long way from the original von Neumann computer. However, despite the inherent use of out-of-order parallelism within superscalar microprocessors today, the order of the instruction flow as seen from outside by the compiler or assembly language programmer still retains the sequential program order as defined by the von Neumann architecture.

Radically different operating principles, such as the dataflow principle and the reduction machine principle, were surveyed very early on. The dataflow principle states that an instruction can be executed when all operands are available (data-driven) while the reduction principle triggers instruction execution when the result is needed (demand-driven). We find a modified variant of the dataflow principle, called local dataflow, in today's superscalar microprocessor cores to decide when instructions are issued to the functional units.

Since present-day microprocessors are still an evolutionary progress from the von Neumann computer, at least four classes of future possible developments can be distinguished:

- Microarchitectures that retain the von Neumann architecture principle (the result sequentiality), although instruction execution is internally performed in a highly parallel fashion. However, only instruction-level parallelism can be exploited by contemporary microprocessors. Because instruction-level parallelism is limited for sequential threads, the exploited parallelism is enhanced by speculative parallelism. Besides the superscalar principle applied in commodity microprocessors, the superspeculative, multiscalar, and trace processor principles are hot research topics. All these approaches belong to the same class of implementation techniques because result sequentiality

must be preserved. A reordering of results is performed in a retirement phase in order to conform this requirement.
- Processors that modestly deviate from the von Neumann architecture but allow the use of the sequential von Neumann languages. Programs are compiled to the new instruction set principles. Such architectural deviations include very long instruction word (VLIW), SIMD in the case of multimedia instructions, and vector operations.
- Processors that optimize the throughput of a multiprogramming workload by executing multiple threads of control simultaneously. Each thread of control is a sequential thread executable on a von Neumann computer. The new processor principles are the single-chip multiprocessor and the simultaneous multithreaded processor.
- Architectures that break totally with the von Neumann principle and that need to use new languages, such as, e.g., dataflow with dataflow single-assignment languages, or hardware-software codesign with hardware description languages. The processor-in-memory, reconfigurable computing, and the asynchronous processor approaches also point in that direction.

In particular processor architecture covers the following two aspects of computer design:
- the instruction set architecture which defines the boundary between hardware and software (often also referred to as the "architecture" of a processor), and
- the "microarchitecture", i.e., the internal organization of the processor concerning features like pipelining, superscalar techniques, primary cache organization, etc.

Moreover, processor architecture must take into account the technological aspects of the hardware, such as logic design and packaging technology.

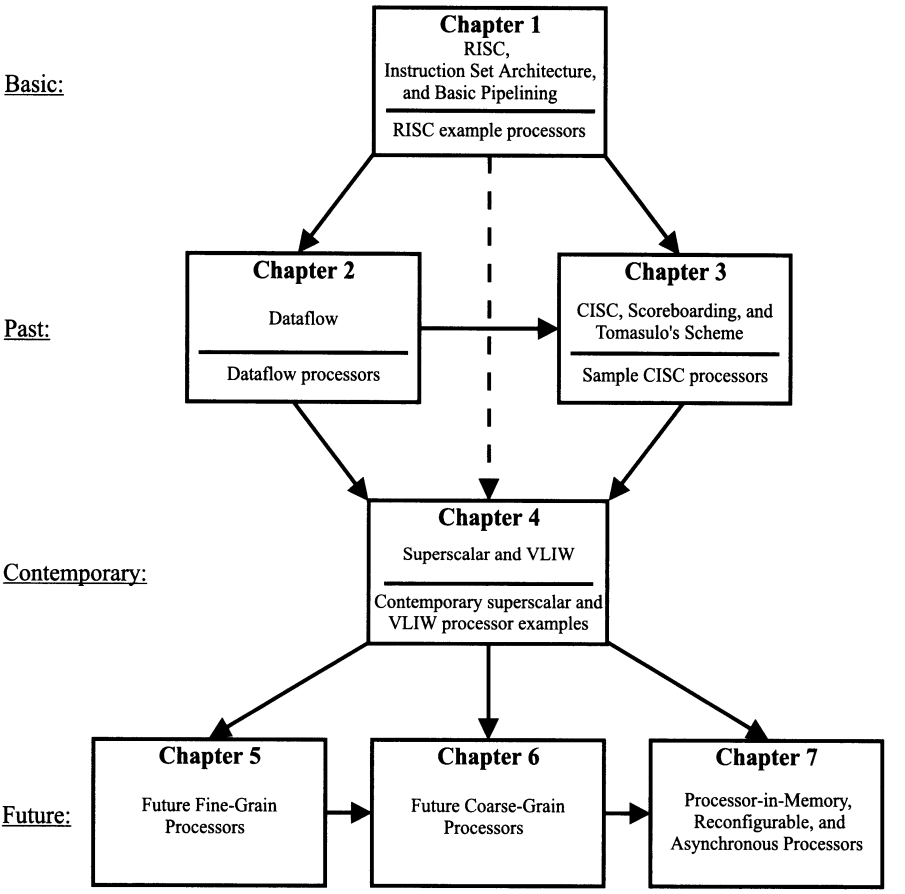
Intended Audience

The primary intended audience of this book are computer and/or electrical engineers and researchers in the fields of computer science. It can also be used as a textbook for processor architecture or advanced microprocessor courses at the graduate student level of computer science or electrical engineering. As such it is not intended for beginners.

The book surveys architectural mechanisms and implementation techniques for exploiting fine-grained and coarse-grained parallelism within microprocessors. It starts with a review of the basic instruction set architecture and pipelining techniques, continues with a comprehensive account of state-of-the-art superscalar and VLIW techniques used in microprocessors. It covers both the concepts involved and implementations in modern microprocessors. The book ends with a thorough review of the research techniques that will lead to future microprocessors.

Using the book

Each book chapter comprises a tutorial on the specific techniques and comprehensive sections on sample processors. The reader may quickly browse the sample processor sections, if interested mainly in learning the techniques. The conversant reader may even start with Chap. 4 – the main chapter of the book, while the student is advised to read at least Chaps. 1 and 3.



Overview of the book

Chapter 1. Basic Pipelining and Simple RISC Processors. After a period of programming in assembly language, the permanent desire for reduced software costs ultimately resulted in the appearance of high-level programming languages. However, at that time – about two decades after the von Neumann architecture had been proposed – processor design did not provide

hardware-based support for most of the high-level language features. Thus, the programmer's view of the machine was removed from the architect's view – the so-called semantic gap appeared. In the 1970s, microelectronic technology made it possible to replace software with hardware and, in particular, to incorporate high-level language features in the processor instruction set architecture. This resulted in complex instruction set computer (CISC) processors characterized by a large number of instructions, addressing modes, and instruction formats. As an alternative, the RISC approach was born in the mid-1970s, advocating the hardware support of only the most frequent instructions while implementing the others as instruction sequences. After the pioneering architecture of the IBM 801, the main initial research in RISC was carried out by teams at Berkeley and Stanford University. While the former relied on a large number of registers to minimize the memory latency, the later pared hardware down to a minimum and relied on a smart compiler. The two research studies initiated a number of other projects that resulted in modern RISC microprocessors.

The goal of RISC architecture during the 1980s was to develop processor designs that can come close to issuing one instruction each clock cycle. This was made possible by using hardwired, instead of microcoded, control, by supporting a small set of equal-length instructions, most of which are of the register-register type, by relying on a high-performance memory hierarchy as well as instruction pipelining and optimizing compilers. Moreover, superpipelined processors allowed for higher clock rates by a longer pipeline, although they still issue one instruction after the other. Since all these RISC processors issue only one instruction at a time, they are said to be scalar. Instruction set architecture and basic pipelining techniques are explained in this chapter in the context of scalar RISC processors. When more than one instruction can be issued at once, the resulting overlap between instructions is called instruction-level parallelism. The processors capable of utilizing instruction-level parallelism and issuing more than one instruction each clock cycle are superscalar and VLIW processors, as well as dataflow processors.

Chapter 2. Dataflow Processors. Dataflow computers have the potential for exploiting all the parallelism available in a program. Since execution is driven only by the availability of operands at the inputs to the functional units, there is no need for a program counter in this architecture, and its parallelism is limited only by the actual data dependences in the application program. Dataflow architectures represent a radical alternative to the von Neumann architecture because they use dataflow graphs as their machine language. Dataflow graphs, as opposed to conventional machine languages, specify only a partial order for the execution of instructions and thus provide opportunities for parallel and pipelined execution at the level of individual instructions.

While the dataflow concept offers the potential of high performance, the performance of an actual dataflow implementation can be restricted by a limited number of functional units, limited memory bandwidth, and the need to match pending operations associatively with available functional units. Since the early 1970s, there have been significant developments in both fundamental research and practical realizations of dataflow models of computation. In particular, there has been active research and development in the multithreaded architectures that have evolved from the dataflow model. These developments have also had a certain impact on the conception of high-performance processor architectures in the "post-RISC" era.

Chapter 3. CISC Processors. Even stronger impact on the high-performance "post-RISC" architecture was made by CISC processors. These processors date back to the mainframe computers of the 1960s, exemplified by the CDC 6600, IBM System/360, DEC PDP-11, etc. which were rack-based machines implemented with discrete logic. Their processors used complex instruction sets with hundreds of instructions, dozens of addressing modes, and more than ten different instruction lengths. In the 1970s, several breakthroughs in technology made it possible to produce microprocessors. Several CISC-type microprocessor families were developed, including the Intel 80x86 and Motorola MC 680xx, whose descendants such as the Pentium II and MC 68060 represent a strong alternative to the RISC-type processors.

The competition between CISC and RISC continues, with each of the two taking ideas of the other and using them to increase its performance. Such an idea, originating from CISC machines, is out-of-order execution where instructions are allowed to complete out of the original program order. In the CDC 6600 the control of out-of-order execution was centralized (with scoreboarding), while in the IBM System/360 Model 91 it was distributed (with Tomasulo's scheme). Scoreboarding and Tomasulo's scheme faded from use for nearly 25 years before being broadly employed in modern microprocessors in the 1990s. Other old ideas are being revived: out-of-order execution implemented with scoreboarding or Tomasulo's scheme is quite similar to dataflow computing with simplified matching and handling of data structures.

Chapter 4. Multiple-Issue Processors. Superscalar processors started to conquer the microprocessor market at the beginning of the 1990s with dual-issue processors. The principal motivation was to overcome the single-issue of scalar RISC processors by providing the facility to fetch, decode, issue, execute, retire, and write back results of more than one instruction per cycle. One technique crucial for the high performance of today's and future microprocessors is an excellent branch handling technique. Many instructions are in different stages in the pipeline of a wide-issue superscalar processor. However, approximately every seventh instruction in an instruction

stream is a branch instruction which potentially interrupts the instruction flow through the pipeline.

VLIW processors use a long instruction word that contains a (normally) fixed number of operations that are fetched, decoded, issued, and executed synchronously. VLIW relies on a sequential stream of long instruction words, i.e., instruction tuples, in contrast to superscalar processors, that issue from a sequential stream of "normal" instructions. The instructions are scheduled statically by the compiler, in contrast to superscalar processors which rely on dynamic scheduling by the hardware. VLIW is not as flexible as superscalar and therefore has been confined to signal processors during the last decade. Recently the VLIW technique has come into focus again in the explicitly parallel instruction computing (EPIC) design style proposed by Intel for its IA-64 processor Merced.

The chapter presents all components of superscalar and VLIW-based multiple-issue processors in detail and provides descriptions of nearly all major superscalar microprocessors.

Chapter 5. Future Processors to Use Fine-Grain Parallelism. Current microprocessors utilize instruction-level parallelism by a deep processor pipeline and by the superscalar instruction issue technique. VLSI technology will allow future generations of microprocessors to exploit aggressively instruction-level parallelism up to 16 or even 32 instructions per cycle. Technological advances will replace the gate-delay by an on-chip wire-delay as the main obstacle to increase chip complexity and cycle rate. The implication for the microarchitecture is a functionally partitioned design with strict nearest neighbor connections.

One proposed solution is a uniprocessor chip featuring a very aggressive superscalar design combined with a trace cache and superspeculative techniques. Superspeculative techniques exceed the classical dataflow limit which says: Even with unlimited machine resources a program cannot execute any faster than the execution of the longest dependence chain introduced by the program's data dependences. Superspeculative processors also speculate about data dependences.

The trace cache stores dynamic instruction traces contiguously and fetches instructions from the trace cache rather than from the instruction cache. Since a dynamic trace of instructions may contain multiple taken branches, there is no need to fetch from multiple targets, as would be necessary when predicting multiple branches and fetching 16 or 32 instructions from the instruction cache.

Multiscalar and trace processors define several processing cores that speculatively execute different parts of a sequential program in parallel. Multiscalar uses a compiler to partition the program segments, whereas a trace processor uses a trace cache to generate dynamically trace segments for the processing cores.

A DataScalar processor runs the same sequential program redundantly on several processing elements with different data sets.

Chapter 6. Future Processors to Use Coarse-Grain Parallelism. The instruction-level parallelism found in a conventional instruction stream is limited. Recent studies have shown the limits of processor utilization even of today's superscalar microprocessors. The solution is the additional utilization of more coarse-grained parallelism. The main approaches are the multiprocessor chip and the multithreaded processor which optimize the throughput of multiprogramming workloads rather than single-thread performance. The multiprocessor chip integrates two or more complete processors on a single chip. Every unit of a processor is duplicated and used independently of its copies on the chip.

In contrast, the multithreaded processor stores multiple contexts in different register sets on the chip. The functional units are multiplexed between the threads in the register sets. Because of the multiple register sets, context switching is very fast. The multiprocessor chip is easier to implement, but does not have the capability of multithreaded processors to tolerate memory latencies, by overlapping the long-latency operations of one thread with the execution of other threads.

The performance of a superscalar processor suffers when instruction-level parallelism is low. The underutilization due to missing instruction-level parallelism can be overcome by simultaneous multithreading, where a processor can issue multiple instructions from multiple threads each cycle. Simultaneous multithreaded processors combine the multithreading technique with a wide-issue superscalar processor such that the full issue bandwidth is utilized by potentially issuing instructions from different threads simultaneously. Depending on the specific simultaneous multithreaded processor design, only a single instruction pipeline is used, or a single issue unit issues instructions from different instruction buffers simultaneously.

Chapter 7. Processor-in-Memory, Reconfigurable, and Asynchronous Processors. Architectural techniques that partly give up the result serialization that is characteristic of von Neumann architectures arise from an on-chip processor–memory integration and from reconfigurable architectures. Such innovations have the potential to define highly parallel chip architectures.

The processor-in-memory or intelligent RAM approach integrates processor and memory on the same chip to increase memory bandwidth. The starting points for processor and memory integration can be either a scalar or superscalar microprocessor chip that is enhanced by RAM memory rather than cache memory, or it can be a RAM chip combined with some computing capacity. Researchers at Sun Microsystems propose a processor-in-memory design that couples a RISC processor with multi-banked DRAM memory.

The Mitsubishi M32R/D is a similar processor-in-memory approach designed for embedded systems applications. Vector intelligent RAM processors couple vector processor execution with large, high-bandwidth, on-chip DRAM banks. The Active Page approach is at the other end of the design spectrum which may be characterized as smart memory approaches. Active pages provide data access and manipulation functions for data arrays integrated in a RAM chip, the processor staying off-chip.

Reconfigurable computing devices replace fixed hardware structures with reconfigurable structures, in order to allow the hardware to adapt to the needs of the application dynamically at run-time. The MorphoSys reconfigurable architecture combines a reconfigurable array of processing elements with a RISC processor core. The Raw architecture approach is a set of replicated tiles, wherein each tile contains a simple RISC-like processor, a small amount of bit-level reconfigurable logic and some memory for instructions and data. Each Raw tile has an associated programmable switch which connects the tiles in a wide-channel point-to-point interconnect. The Xputer defines a non-von-Neumann paradigm implemented on a reconfigurable Datapath Architecture.

Conventional synchronous processors are based on global clocking whereby global synchronization signals control the rate at which different elements operate. For example, all functional units operate in lockstep under the control of a central clock. As the clocks get faster, the chips get bigger and the wires get finer. As a result, it becomes increasingly difficult to ensure that all parts of the processor are ticking along in step with each other.

The asynchronous processors attack clock-related timing problems by asynchronous (or self-timed) design techniques. Asynchronous processors remove the internal clock. Instead of a single central clock that keeps the chip's functional units in step, all parts of an asynchronous processor (e.g., the arithmetic units, the branch units, etc.) work at their own pace, negotiating with each other whenever data needs to be passed between them. Several projects are presented, one of these – the Superscalar Asynchronous Low-Power Processor (SCALP) – is presented in more detail.

Additional Information

The book's home page provides various supplementary information on the book, its topics, and the processor architecture field in general. Lecture slides are available in PowerPoint, PDF, and Postscript, covering the whole book. Over time, enhancements, links to processor architecture-related web sites, corrigenda, and reader's comments will be provided. The book's home page is located at `goethe.ira.uka.de/~ungerer/proc-arch/` and can also be accessed via the Springer-Verlag home page `www.springer.de/cgi-bin/search_book.pl?isbn=3-540-64798-8`.

The home pages of the authors are `www-csd.ijs.si/silc` for Jurij Šilc, `www-csd.ijs.si/robic/robic.html` for Borut Robič, and `goethe.ira.uka.de/people/ungerer` for Theo Ungerer.

Additional information can be drawn from the "WWW Computer Architecture Home Page" of the University of Wisconsin at `www.cs.wisc.edu/~arch/www/` which provides comprehensive information on computer architecture research. Links are provided to architecture research projects, the home pages and email addresses of people in computer architecture, calls for papers, calls for conference participation, technical organizations, etc.

The National Technology Roadmap for Semiconductors `www.sematech.org/public/home.htm` is a description of the semiconductor technology requirements for ensuring advancements in the performance of integrated circuits. Sponsored by the Semiconductor Industry Association (SIA) and published by SEMATECH, this report is the result of a collaborative effort between industry manufacturers and suppliers, government organizations, consortia, and universities.

The CPU Info Center of the University of California, Berkeley, at `infopad.eecs.berkeley.edu/CIC/` collects information on commercial microprocessors such as, e.g., CPU announcements, on-line technical documentations, a die photo gallery, and much more.

An excellent guide to resources on high-performance microprocessors is the "VLSI microprocessor" home page `www.microprocessor.sscc.ru` at the Supercomputer Software Department RAS.

The newest commercial microprocessors are presented at the Microprocessor Forum, the Hot Chips Conference, and the International Solid State Circuit Conference (ISSCC). The most important conferences on research in processor architecture are the Annual International Symposium on Computer Architecture (ISCA), the biennial International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), the International Symposium on High-Performance Computer Architecture (HPCA), the Annual International Symposium on Microarchitecture (MICRO), and the Parallel Architectures and Compilation Techniques (PACT) conferences.

Acknowledgments

Several people in academia have provided invaluable assistance in reviewing drafts of the book. We would especially like to thank *Bruce Shriver* who greatly influenced the structure and terminology of this book by sharing his experiences from his own recent book "The Anatomy of a High-Performance Microprocessor".

Jochen Kreuzinger of the University of Karlsruhe and *Andreas Unger* of the University of Jena provided thorough and detailed critics of several chap-

ters of the book leading to considerable improvements in the text. *Silvia Müller* of the University of Saarbrücken, *Ulrich Sigmund* of Viona GmbH, Karlsruhe, and *Ulrich Nageldinger* of the University of Kaiserslautern provided valuable improvements to the sections on scoreboarding and Tomasulo's scheme, superscalar processors, and reconfigurable computing. *Yuetsu Kodama* of the Electrotechnical Laboratory at Tsukuba provided invaluable information on threaded dataflow and, especially, on the EM-X and RWC-1 parallel computers. Several contributions to the chapter on RISC processors are due to *Veljko M. Milutinović* of the University of Belgrade. Numerous suggestions and improvements in the chapter on future processors using fine-grain parallelism have been made by *Lalit Mohar Patnaik* of the Indian Institute of Science at Bangalore. We have also received significant help from *Nader Bagherzadeh* of the University of California at Irvine, *Reiner W. Hartenstein* of the University of Kaiserslautern, and *Krishna M. Kavi* of the University of Alabama at Huntsville. Many thanks to all of them.

We also thank *Daniela Tautz* of the University of Karlsruhe who has drawn several figures, and *Gregor Papa* of the Jožef Stefan Institute, Ljubljana, who has drawn the tables.

Numerous students of the University of Karlsruhe have improved this book by their questions and ideas. In particular we have to thank *Alexander Schulz*, *Jan Albiez*, and *Patrick Ohly*.

Finally we would like to express our thanks to our editors, *Hermann Engesser* and *Brygida Georgiadis*, and the anonymous copy-editor of Springer-Verlag.

Ljubljana and Karlsruhe, March 1999

Jurij Šilc
Borut Robič
Theo Ungerer

Contents