Part II (96 pp.), *Designing the System*, tries to present a basic design methodology for OLS, by identifying and discussing the basic design decisions to be made at the start of each project, such as goals, kind of applications, hardware and operating system choice and functional specifications. However, the core of this part is a case study of the development effort of a medical information system, the MEDINET system. We learn about some first hand experiences and reasons for the failure of this ambitious and large project. This most interesting chapter shows the critical need for careful planning and good management, and the morals at the end of the story should be heeded by everybody engaged in any sizable system development effort. The chapter on design calculations discusses the roles of mathematical modelling, simulation, and rules of thumb calculations, with obvious preference for the latter. For the designer, the main purpose of these tools is proving the feasibility of and exposing bottlenecks within the developed system. A fairly cursory treatment of statistics and performance measurement concludes this part, which does not yet reflect the ready availability of commercial hardware and software monitors and the explosion of activity in this area. In this part, the Widget Corporation of America and their sales order entry OLS is introduced to illustrate the use of rules of thumb for arrival and processing rates. This last text feasibility of a design concept. This fictitious corporation resurfaces in several later chapters to give practical significance to the concepts presented.

Part III (87 pp.), *Application Programs for On-Line Systems*, is intended for the designer of such programs. It introduces the facilities offered and the techniques recommended in the more sophisticated environment of an on-line system: multitasking, overlays, such as Wait and Post, modular programming, overlap of I/O and processing, memory management, and how to take advantage of virtual storage systems, terminal control, file sharing, and last but not least, job control language. The next chapter deals with the structure of application programs (the terminal-program-file graph) and their suitability to the three different types of OLS: scientific time sharing, business OLS with and without multitasking. The pros and cons of the most common languages for application programs are also discussed. The remaining third of this part returns to the Widget Corporation's OLS. The structure of their concurrently executing application programs is developed: a teleprocessing module handles 16 types of transactions by multiprogramming, four processing modules which work on an inventory and an account file.

Part IV (122 pp.), *Files and Data Bases for OLS*, presents in its first half a rather elementary treatment of buffering, blocking and various search and access techniques, such as randomizing, linked lists, and inverted files. The second half adds to this the need for a lock/unlock capability to handle concurrent accesses and shows how deadlock situations can arise. The treatment does not measure up to the level available in the tutorial literature (see, e.g., Coffman, et al., in acm computing surveys, June 1971), and no references are given in this chapter. A brief but lucid and comprehensive discussion of file security follows dealing with user identification, access keys and protection problems. Similarly, the discussion on file system recovery presents a quite exhaustive set of practical situations. In the last 25 pages, the choice of file access technique and file organizations for the Widget Corporation's OLS illustrates the application of the ideas developed in this part.

Part V (160 pp.), *Operating Systems for OLS*, intends to develop a working knowledge of these by explaining the functions and general structure of a typical operating system. Firstly, the concept of priority interrupt structures is developed, followed by a treatment of scheduler programs (dispatching), covering round robin and multilevel priority algorithms. Then, I/O Routines are discussed, separating the trap (logical) from the interrupt (physical) part. Optimization of rotating devices for throughput and responsiveness gets its share, as well as I/O error recovery and recovery. The ensuing chapter on core allocation gives a fairly complete tour d'horizon of various approaches ranging from fixed allocation through elementary paging schemes to working set and reclaiming techniques. The GE-445 is used as paging hardware example. Motivation for and problems with various schemes are presented, and there are even some quite educated technological predictions. A final chapter is devoted to system failures and error recovery for OLS, deploring the naivete of many system designers in this area. Reliability is treated in terms of mean time between failure, mean time to repair and data base protection. The following types of errors, their possible consequences and remedial action are discussed: CPU, memory, peripheral power or environmental failure: operator error, data base erosion, saturation and, last but not least, unexplained failures. The basic philosophies for recovery are presented as orderly system or user shutdown, fall back to degraded mode, switch to stand-by subsystem or system. Part VI (51 pp.), *Testing and Debugging*, distinguishes testing, i.e., "documenting the presence or absence of errors" from debugging, i.e., "finding and removing bugs." The point is made that OLS need increased testing efforts, with possibly half of the project time devoted to stable and debugging, and that a separate parallel testing task force should be established. Concepts explained include simulated input packages, test data base generators, tracing facilities capable of selecting system calls, and terminal I/O or file accesses. A number of test transactions are proposed, which are dynamically introduced into the live system to test its behavior under load. The final chapter deals with a dynamic debugging technique ("DDT"), distinguishing stand-alone DDT, mainly for minicomputers, from user DDT and supervisor DDT, the latter differing in that the scope of monitoring and manipulation capabilities extents over the operating system or over the application program, letting the other one operate without interference, respectively. The concept of a sample implementation of a design concept in which the DDT communicates the benefits.

Taken as a whole, this is a very readable book, written in a fresh style and loaded with examples, which betray the author to be a hardened practitioner as well as a skilled teacher. The book is not slanted towards a single vendor's line. It is not and does not pretend to be a theoretical text. (Viz the almost normal distribution in Fig. SI2, p. 483, which the author calls an "exponential curve"). Rather, the frequently humorous narrative is tying together in a sensible perspective the basic ideas, problems encountered and pitfalls to avoid, such that substantial portions of the book become worthwhile reading for anyone involved in a software engineering project. Academicians, on the other hand, might get a better appreciation of the real world from it, or at least some idea on how to relate their ideas to the "guys in the field."

Some minor criticism: The larger part of the book is not peculiar to on-line systems, it applies to batch systems as well. This reviewer had expected more emphasis on the data communication aspects of these systems, a great area of real difference from batch systems—although one is advised to turn to J. Martin's books for this area available from the same publisher. The illustrations which are plentiful seem to be often taken from lecture slides, their information content is low, they tend to inflate the book (lots of white space) and are not infallible. The second half is frequently out of phase with the text. The index has not kept up with the editing of the book: this reviewer found one rupture around p. 200 where one has to subtract 12 from page numbers given in the index. The references vary in quality and quantity. There are no references on resource sharing and deadlocks, whereas design calculations are adequately covered by a representative set of 64 references.


The two volumes in this set belong in every library on compiler design. Together they achieve a level of rigor and completeness not attained by any other book on the subject.

The two volumes form an integrated work with consecutively numbered pages and chapters. The subtitles of the volumes would indicate that Volume I covers parsing theory while Volume II covers the remaining aspects of compiler design. Actually the first half of Volume II, Chapters 7 and 8, contain material on parsing that could well have been included in Volume I if space had been available. Chapter 7 presents techniques for reducing the size of the parsing
tables needed for bottom up parsing. This material is not particularly basic or mathematically elegant, but unfortunately is needed to design bottom up parsers.

Chapter 8 is a theoretical study of deterministic parsing that explores in more depth some of the concepts introduced in Volume I. Chapters 9, 10, and 11 cover the material on “compiling” indicated by the subtitle. The topics discussed are translations, bookkeeping and optimization. In general this material is not as well understood theoretically as is parsing theory, but the presentation is at a higher level of rigor than in most other compiler books. For example, the section on optimizing the code generated for arithmetic expressions without common subexpressions proves the optimality of the generated code under a variety of cost criteria including program length and number of accumulators used.

In the chapters on “compiling” the authors tended to be somewhat more selective in their choice of material than in the chapters on parsing which are almost encyclopedic in their completeness.

One possible source of confusion for some readers occurs occasionally throughout the book and is more prevalent in Chapter 9. The authors do not distinguish carefully enough in some places between what is “true” in a mathematical sense about compilers and what is current practice and only “true” about some compilers in existence today.

The book contains many illustrative examples and a wide variety of exercises graded for difficulty.

In summary, this two volume work sets a new standard in books on compiler design.

PHILIP M. LEWIS
Corporate Research and Development
General Electric Company
Schenectady, N.Y. 12305


Historians tell us that in ancient times the usual mode of operation of a Computer System Manufacturer was for an electronic engineer to emerge from his laboratory one day and tell the waiting programmer “Look! Here’s our next model!” The engineer then disappeared to work on the next-but-one model, leaving the programmer to work around any deficiencies or awkwardness in the hardware design which had been handed to him. In modern times, the idea that a machine should be designed to fulfil a function, that the intended use of the machine should supply constraints on its design (rather than the converse), has received at least lip service in most quarters. Some manufacturers have even adopted it as a way of life.

Professor Organic’s book reflects this modern approach. In response to the question “What do we want the system to do?” he answers “We want it to execute a (time-invariant) algorithm, expressed in a block-structured language, calling upon data and other items from (a time-varying) environment.” Thus, in the simplest view, the hardware processor functions by maintaining a pair of pointers, one to the algorithm (the instruction pointer) and one to the environment. As a block or procedure in the algorithm is entered, or exited, the environment changes with the scope of the variable names. Chapters 2 and 3 introduce and elaborate these concepts, by asking at each step “What information must be available to enable the algorithm to be executed?”

However, the idea of an algorithm as a sequence of procedural steps is regarded as inadequate for the 1970’s. Rather, it is proposed that a task be defined in this manner, and that the definition of an algorithm be generalized, so that it consists of a structure (often nested) of interdependent, normally asynchronous tasks. Chapters 4 and 5 develop the requirements for implementing this generalization, that is, requirements for multitasking. The treatment is still on the level of asking what information must (somehow) be made known, to what task, and when.

In traditional treatments, the introduction of interrupts represents a dramatic new concept to be grasped by the reader. However, Professor Organick regards interrupts as “merely unexpected procedure calls,” so the treatment of interrupts in Chapter 6 can be quite brief. The emphasis here is on software interrupts, and the author covers some of the philosophical problems encountered, such as interrupting a “sleeping task.”

Chapter 7 discusses storage control strategies and Chapter 8 is a discussion of the pros and cons of the computer organization which has emerged from the considerations of earlier chapters. It will be no surprise to readers of this review that a computer organized in this manner is commercially available, and is known as the Burroughs B6700. Chapter 9, which is written by J. S. Cleary and has the nature of an appendix, reveals some of the hardware details of the B6700 implementation. As a further example of the flavor of the book, we note that it is only in this last chapter that we are told that some of the information necessary for the execution of the algorithm is kept in primary storage (main memory), and some in temporary storage (the processor registers).

The publication of this book is a highly significant event, reflecting the relocation of the hardware/software interface which has occurred in recent years, and the trend toward “higher level language processors.” Readers who expect an exposition on the level of the machine’s instruction set will be disappointed. Such readers should consult Bell and Newell’s excellent book (see review B72-3). Organic’s book might well be regarded as complementary to that of Bell and Newell. For example, a student baffled by Bell and Newell’s comment that the language ALGOL is the antecedent of the B5000, will understand that comment fully after reading Organic’s book.

Some minor negative comments should be made. The emphasis on multitasking may be too great, in view of the author’s admission that “as of 1972 there has hardly been amassed any abundance of applications programming experience with tasking . . . .”. The concept of interrupts as “merely unexpected procedure calls” is too superficial for critical real-time problems. The comment in Chapter 2 that languages like FORTRAN “can be regarded as degenerate examples of block structured languages” implies a promise which is broken in Chapter 5, which discusses why FORTRAN programs do not execute rapidly on the machine described.

The existence of this book means that ideas long buried in fragmentary form in the manuals of the manufacturer, or in one or two conference papers, are now readily available in coherent form, to teachers and students. It is to be hoped that both groups take advantage of this new opportunity.

A. C. L. BARNARD
Dep. of Inform. Sci.
University of Alabama in Birmingham
Birmingham, Ala. 35294


Pattern Recognition Techniques does just what the doctor ordered; it serves as a broad informal introduction for engineers, computer scientists, and biologists to the field of pattern recognition. All necessary background is developed within the text, which is illustrated in terms of character recognition. It would be difficult to find a more extensive bibliography. This greatly contributes to the usefulness of this book.

CHAPTER I—MASK MATCHING

Chapter I begins with an introduction to character recognition via optical mask matching. Treatment is given to several machines which recognize characters printed in a single font. The concepts of best match, reject threshold, substitution error, and success rate are introduced. In the interest of speed, electronic mask matching machines are presented using photocell mask and current measuring devices. Binairizing circuits are added to couple the photocell signal to the output. Both analog and digital devices are discussed. The concepts of maximization and minimization for selection are both introduced. Peephole masks and negative weights end the chapter. The author is very successful in this chapter in establishing a back-