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 fulfill 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 Organick'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 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). Organick'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 Organick'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 8, 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
Dept. 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 devices are presented using photocell masks and current-sensing devices. Binaryizing 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-