setting named bits, at virtually no extra cost.

These same subroutines will also be employed to implement the automatic scheduling philosophy described in Section 4.4, in a trial reimplementation of the Hardware Kernel's process scheduler (using priority scheduling). We chose not to implement this in the first version, out of respect for the problems of debugging the microcode, the synchronising design and the code of the Kernel processes simultaneously!

References

BURROUGHS. (1972). The Burroughs B6700 Information Processing Systems Reference Manual, Burroughs Corporation, Detroit, Michigan, 1972.

DIJKSTRA, E. W. (1968a). Cooperating Sequential Processes in *Programming Languages*, ed. F. Genuys, Academic Press, London and New York, 1968.

DIJKSTRA, E. W. (1968b). The Structure of the 'THE'-Multiprogramming System, *CACM*, Vol. 11, No. 5, p. 341.

DIJKSTRA, E. W. (1972). Hierarchical Ordering of Sequential Processes in *Operating System Techniques*, ed. C. A. R. Hoare and R. H. Perrott, Academic Press, London and New York, 1972.

KEEDY, J. L. (1977). An Outline of the ICL2900 Series System Architecture, *Australian Computer Journal*, Vol. 9, No. 2, July, 1977.

PARNAS, D. L. (1975). On a Solution to the Cigarette Smokers' Problem (without conditional statements), *CACM*, Vol. 18, No. 3, p. 181.

Book reviews

Compiler Design Theory, by P. M. Lewis II, D. J. Rosenkrantz and R. E. Stearns, 1976; 647 pages. (*Addison Wesley*, £18·50)

This book is intended as an undergraduate course on compiler design theory. Some teachers may prefer not to base their course on such a formal, and largely automata theory, footing but all should find the book a rich source of 'academically respectable' material which could supplement a more practical treatment of the subject. It is meticulously prepared with many worked and even more unworked examples. The authors' claim that it should be understandable to a wide range of readers is justified, but it is by no means light reading.

Although the approach is consistently and deliberately formal the application of the concepts is demonstrated on the design of an actual compiler. Unfortunately, in the reviewer's opinion, the language chosen for this demonstration, a subset of BASIC, presents little challenge to the compiler writer. A reader might be excused for feeling that the techniques described represent 'sledge hammer to crack a nut'.

The first four chapters give a good and adequate introduction to finite state machines leading up to the design of the lexical analysis stage for the BASIC subset compiler. At this point push down machines are introduced and their application as recognisers and translators of abstract input sequences is considered.

After a diversion in Chapters 6 and 7 to introduce the basic theory of context free grammars, translation grammars and attributed translation grammars, the application of push down machines to top down parsing and translation is developed. This leads in Chapter 10 to the detailed design of the syntax analysis phase of the BASIC subset compiler.

The alternative of bottom up analysis is described in Chapters 11, 12 and 13, again in terms of a push down machine implementation. An alternative (compatible) design for the syntax analysis phase of the BASIC subset compiler is given.

Code generation and optimisation are the subject of the final two chapters of the book. Many compiler writers might feel that this is the area where the main problems lie, but this book devotes only 31 of its 600-odd pages to them. Perhaps the problems are symptomatic of the dearth of formal treatment of the subject.

In conclusion it must be said that this book presents a great deal of compiler theory in a new way and is worthy of a place on any computer scientist's bookshelf. However, if funds are a limiting factor, the book *Principles of Compiler Design* by Aho and Ullmann might be a preferred alternative since it is more broadly based.

D. MORRIS (Manchester)

An Introduction to Programming and Problem Solving with Pascal, by G. Michael Schneider, Steven W. Weingart and David M. Perlman, 1978; 394 pages. (*John Wiley*, £9·20)

This book, like a growing number in recent years, is derived from programming courses which have been taught at university, and is based on the PASCAL programming language. The main aim of the book is to introduce all of the aspects concerned with programming and problem solving, from problem specification to design, implementation, debugging and documentation and maintenance. Secondary aims are to teach what constitutes a good programming style, and to teach the syntax of PASCAL.

Following the introduction, Chapter 2 informally introduces the concepts of algorithms and the basic forms of flow of control. The chapter also discusses the efficiency of algorithms and recursion. Chapters 3 to 8 introduce various aspects of PASCAL, with Chapter 6 devoted to the important topic of debugging and testing programs. Chapter 9, entitled 'Building quality programs', is rightly regarded by the authors as being one of the most important chapters in the book and discusses techniques for developing and managing large 'real world' programs.

The book is well presented and has many illustrative examples. Each chapter is followed by exercises and some solutions are provided. Some people will no doubt find minor faults with the order of presentation or with the emphasis given (or not given) to some topics. However, the book does appear to have been well thought out (probably at the expense of many undergraduates!) and to satisfy its aims. It should certainly be considered as a contender for a recommended textbook for introductory programming courses, particularly those based on PASCAL.

P. A. LEE (Newcastle)

Principles of Compiler Design, by A. V. Aho and J. D. Ullmann, 1977; 604 pages. (*Addison-Wesley*, £15·20)

This book will be valuable as an encyclopaedia of techniques used in the construction of compilers. It deals in depth with the construction of lexical scanners, basic parsing techniques, the mechanical construction of LR, SLR, and LALR parsers (including a useful exposition of the treatment of ambiguous grammars by parser generators for LR grammars), syntax-directed translation, symbol tables, run time storage administration, and error detection and recovery during parsing.

The book also contains a comprehensive three-chapter treatment of code optimisation. The usefulness of the extensive bibliography is enhanced by the notes and references at the end of each chapter. These give valuable references to further works of relevance to the topics covered in the chapter.

Unfortunately the book's suitability as a text for an undergraduate course in compiling is somewhat reduced by the lack of a detailed case study of a working compiler. A companion volume containing the authors' own implementation of the project which they suggest in Appendix B—the building of a compiler for a subset of PASCAL —would remedy this. Another important omission is any treatment of the relationship between compilers and debugging aids. Finally, the book is marred by several typographical errors which could be misleading to the uninformed reader.

Despite these shortcomings the book will, as the authors claim, be useful as a source of ideas and techniques for software designers working both within and outside the field of compiler design.

BERNARD SUFRIN (Oxford)