RAMAMOORTHY, C. V. and GONZALEZ, M. J. (1969). A survey of techniques for recognizing parallel processable streams in computer programs, *Proc. AFIPS 1969 FJCC.*
SCHNECK, P. B. (1972). Automatic recognition of parallel vector operation in a higher level language, *Proc. ACM 1972 National Conference.*
SCHNECK, P. B. (1975). Movement of implicit parallel and vector expressions out of program loops, *ACM SIGPLAN NOTICES*, Vol. 10, pp. 103-106.
STONE, H. S. (1967). One-pass compilation of arithmetic expressions for a parallel processor, *CACM*, Vol. 10, pp. 220-223.

# Book reviews

*Microcomputer Problem Solving Using Pascal* by K. L. Bowles, 1977; 563 pages. (*Springer-Verlag*, $9·80)

This book is yet another contender for the 'introduction to problem solving and computer programming' market and, like other recent books, is based on PASCAL. The book is derived from courses presented at the University of Calfornia at San Diego (UCSD). UCSD have implemented a stand-alone single user PASCAL system which is interpreter based and runs on a variety of microcomputers, for example LSI-11, Z80 and Intel 8080 based systems. The book is based on this PASCAL system, and is intended for students with both mathematical and non-mathematical backgrounds. The problem examples are of a non-numeric nature, concentrating on graphical and string manipulation examples. UCSD have extended PASCAL with built-in functions and procedures to suit these applications.

For those students who have access to a UCSD system with graphics hardware, the book could be useful. However, I do not feel that I can recommend the book for students using other PASCAL systems. There are other books on the same topic which seem to be much more system independent and therefore more useful to a wider audience.

P. A. LEE (Newcastle upon Tyne)

*The Design of Well Structured and Correct Programs* by S. Alagic and M. A. Arbib, 1978; 292 pages. (*Springer-Verlag*, $14·00)

I am impressed–this book is a welcome change from the usual programming text. The authors' aim is to teach top-down program development using Hoare invariance methods; they succeed. The reader will not only pick up many good programming habits, but also learn some PASCAL along the way. For programmers solving the class of problems for which this methodology is suitable (i.e. most) this book is a must. The authors assume the reader has done some computer programming, but everything else is introduced carefully, logically and well.

The book is well illustrated throughout by numerous examples and exercises (minor criticism—no solutions) and the motivation for the next step in the argument is always given. The examples used cover a wide range of applications and are all taken from the literature but the proofs are usually new. This means that the reader will often be familar with the problem (gcd, file merging, 8 queens, etc.) and can concentrate on the method of solution. I found this advantageous as the text is quite concise and can best be digested in small bites.

The first three chapters cover the basic ground work and introduce top-down design, PASCAL structured statements and their proof rules and PASCAL data structures. In Chapter 4 these are brought together in a number of case studies where particular solutions are developed. This chapter is called 'Programs and proofs' but due to the size of the examples used it would probably have been better titled 'Routines and proofs'. Having seen how to develop reliable components the authors turn, in Chapter 5, to the very important task of their interconnection. Procedures, functions and block structure are dealt with in this chapter whilst Chapter 6 deals with recursive algorithms *and* data structures. Had the book stopped at this point, it would have been highly recommended, but the best is yet to come; Chapter 7 covers the final structuring statement **goto**! Here much new material is introduced and I find it the most sane discussion on the goto problem which I have ever read. The book closes with appendices on PASCAL syntax, summaries of proof rules, a glossary and a good bibliography.

Like many good ideas Hoare's preconditions and postconditions are, once they have been pointed out, brillantly simple. Invariance I would claim, is one of the research topics which seems to offer most help to the practising programmer. This text should be read by anyone who is, or who wishes to be, a professional programmer. Good simple ideas like these should be brought to the attention of a much wider public and this book is a good first step in this direction. The book meets its aims and one cannot criticise it for things it does not try to do but perhaps we can now look forward to similar texts based on FORTRAN and COBOL. If we continue in this way, work which starts from a theoretical standpoint can impact upon and improve our professional practice.

D. SIMPSON (Sheffield)

*Artificial Intelligence*, edited by A. Bundy, 1978; 253 pages. (Edinburgh University Press, £5·00)

This is a collection of lecture notes from the course Artificial Intelligence 2 given at the University of Edinburgh by the staff of the Department of Artificial Intelligence. The course surveys all those areas of artificial intelligence which must now be considered classical and are under the headings Problem solving, Natural languages, Question answering and inference, Visual perception, and Learning. It was aimed at second and third year undergraduates from disciplines other than computing, but according to the teaching notes students were expected to spend three hours a week at an interactive computer terminal. There must be some doubt whether it is reasonable to mix the initial teaching of computer usage with consideration of the ultimate power of machines, and in an afternote the authors seem to accept this criticism.

There are four different authors, all well known in the field, and different sections inevitably show a different style of presentation. However, these are *notes*, terse and pithy, but not lengthy explanations. They are not suitable for self-study by a novice, and would be best appreciated by an intending teacher of a similar kind of course. The general approach is to stimulate the students to find out for themselves and constant use of the computer together with extensive reading in a variety of topics is indicated. It would seem to make heavy demands on an undergraduate's time and mental agility, in a way which is currently not fashionable, and the authors say that later courses had less practical work and that students were divided into groups according to programming ability. One thing has been achieved; the case for this kind of course within a general curriculum has been made firmly.

J. J. FLORENTIN (London)