15 [12].—A. P. YERSHOV, Editor, *The ALPHA Automatic Programming System*, J. McWilliam, translator, Academic Press, London and New York, 1971, xi + 247 pp., 23 cm. Price $15.00 (A.P.I.C. Studies in Data Processing No. 7).

This anthology is a detailed description of a 24-pass compiler for ALPHA, the Russian dialect of Algol 60. It was written about ten years ago and translated only recently. A previous review in another journal [1] describes the limitations and operating characteristics of the machine, the compiler, and the book. The author of that review admits that he did not check all sections. Over an embarrassingly long period, I have read the entire book. Some sections are extremely difficult reading and some are just plain hard. The authors sometimes gave too much detail (like bit maps and octal constants), but the translator was more often too literal in his interpretation. Several sections refer to a "scale of ···" and it took a long time for me to guess that a scale is a vector. The most mysterious sentence in the whole book is, "It is perfectly obvious that if the PP [compiler] is always started according to a single plan, dealing with many problems will be like breaking a butterfly on the wheel." That sentence must have been in the kernel of the book (that which vanishes on translation). A smaller problem for the western reader is that he is assumed to be familiar with the ALPHA language. I believe only one other book about ALPHA has been translated into English [2]. I have not read it.

I disagree with the opening statement of the previous review. This book does not describe the development of a compiler. I wish it had. How does one successfully manage a team of eleven programmers (35 man-years) developing a large system (45,000 instructions) on a small machine (4096 words)? This aspect is mentioned briefly in the planning documents which were included as appendices—Yershov even laments the scarcity of such papers. This subject is not discussed in the post-project reports which form the main part of the book. These reports are detailed descriptions of various parts of the final ALPHA compiler.

The a priori and ad hoc limitations of the 24-pass ALPHA compiler are in several ways different from those of a one-pass compiler for a superset of Algol 60 [3] from my own experience. We felt obliged to implement recursive procedures because our additional data types were recursively defined. Ad hoc limitations arose from trying to squeeze everything into one pass. We ignored dynamic own arrays. The limitations of ALPHA, on the other hand, seem to arise from the decision to optimize as much as possible. Recursive procedures are not permitted, side effects are ignored, the value of the expression $E - E$ is not necessarily zero (even in the absence of side effects), and so on. Dynamic own arrays were implemented but never used in the first year of experimental operation of the translator. Some very fancy optimization methods were developed, like application of a heuristic solution of the generalized map coloring problem to memory optimization. (Variables are colored by the memory locations allocated for them.) No comparison of fancy methods with more ordinary methods was reported. For each method the interesting measurement would be the number of production runs required to "pay" for the incremental cost in compilation using the fancy method versus the ordinary.

Several efforts in this country are related to concepts developed in the ALPHA translator. The relationship might have been ancestral if we had paid more attention to the Russian efforts. Let us travel back in time to Novosibirsk of ten years ago

and imagine what advice we would give as consultants to the ALPHA group.

Several improvements can be made on the hash addressing scheme used in the ALPHA translator. Perhaps the most surprising is that discovered by Brent [5]. A survey is given by Knuth in [6].

Abrams shows how many of the APL operations on arrays can be carried out on descriptions of the arrays [7]. This relates to the treatment of subscript expressions in ALPHA. Wagner takes the notion of loop fusion for componentwise array options somewhat further and describes several other algorithms for matrix multiplication, optimizing on the choice of algorithm for each product [8].

Cocke and Schwartz also develop a method for factoring common subexpressions by using hash addressing [9]. Lipovski finds largest combinable classes by representing the constraints by a boolean formula and converting it to disjunctive normal form [10]. This method might apply to memory economy.

Wegbreit's treatment of procedures allows for progressive refinement of the code as additional information is supplied by the programmer [11]. ALPHA looks at all the calls on a procedure and extracts all the information it can. I have examined the notion of call by result in terms of operator definition [12]. I give some examples which have a nonprocedural, declarative flavor.

In conclusion, the Russians of ten years ago had some interesting ideas about compilation of programs on and for small machines. The worth of the book is approximately equal to the patience it takes to read it.

RUDOLPH A. KRUTAR

Computer Science Department
College of Engineering
The University of Utah
Salt Lake City, Utah 84112

1. J. G. SANDERSON, (Book Review), *Austral. Comput. J.*, Vol. 4, No. 3, August 1972, p. 144.
2. A. P. YERSHOV, G. I. KOZHAKHIN & U. M. VOLOSHIN, *Input Language for Automatic Programming Systems*, Academic Press, New York and London, 1963.
3. R. ITURRIAGA, T. STANDISH, R. KRUTAR & J. EARLEY, *The Implementation of Formula Algol*, Carnegie Institute of Technology, October 1966; available from DDC (AD 803 577); also, a briefer description is found in [4].
4. R. ITURRIAGA, T. STANDISH, R. KRUTAR & J. EARLEY, "Techniques and advantages of using the formal compiler writing system FSL to implement a formula Algol compiler," *Proc. SJCC*, v. 28, 1966, pp. 241–252.
5. R. P. BRENT, "Reducing the retrieval time of scatter storage techniques," *Comm. ACM*, v. 16, no. 2, February 1973, pp. 105–109.
6. D. E. KNUTH, *The Art of Computer Programming: Volume 3/Sorting and Searching*, Addison-Wesley Publishing Co., Reading, Mass., 1973, pp. 506–549.
7. P. ABRAMS, *An APL Machine*, Stanford Linear Accelerator Center, SLAC Report No. 114.
8. R. A. WAGNER, *Some Techniques for Algorithm Optimization with Application to Matrix Arithmetic Expressions*, Doctoral Dissertation, Carnegie-Mellon University, June 1968; available from DDC (AD 678 629).
9. J. COCKE & J. T. SCHWARTZ, *Programming Languages and Their Compilers*, second revised version, preliminary notes, New York University, April 1970, pp. 306–523.
10. G. J. LIPOVSKI, *An Improved Method of Finding all Largest Combinable Classes*, University of Illinois, Urbana, Illinois, August 1967; available from DDC (AD 656 608).
11. B. WEGBREIT, "An overview of the ECL programming system," *SIGPLAN Notices*, Vol. 6, No. 12, December 1971, pp. 26–28.
12. R. A. KRUTAR, "An algebra of assignment," Courant Institute of Mathematical Sciences, *SETL Newsletter* No. 59, October 1971. See also [13].
13. J. T. SCHWARTZ, "Sinister calls," Courant Institute of Mathematical Sciences, *SETL Newsletter* No. 30, May 1971.