ON THE TIME REQUIRED TO PARSE AN ARITHMETIC EXPRESSION FOR PARALLEL PROCESSING

Ross A. Towle
Federal and Special Systems Group
Burroughs Corporation
Paoli, Pa. 19301

Richard P. Brent
Computer Centre
The Australian National University
Canberra, Australia

Several algorithms that attempt to reduce the tree height of a parse tree of an arithmetic expression by using associativity and commutivity have been proposed [1] - [5]. Baer and Bovet [1] conjectured that their algorithm obtained a minimal-height tree. Later Beatty [2] proved this conjecture. Without distributivity, the upper bound on the tree height is: $1 + 2d + \lceil \log_2 n \rceil$ where n is the number of operands and d is the depth of parenthesis nesting [6]. The selective use of distribution reduces the upper bound on the tree height to $\lceil 4 \log (n-1) \rceil$ [7]. Several algorithms which use distribution have been proposed [7] - [9].

Using the algorithms of Beatty [2] and Brent [7] we can show that the use of associativity and commutivity adds O(N) steps to the parsing process, while the use of associativity, commutivity, and distributivity adds $O(N \log_2 N)$ steps. Both algorithms work on parse trees produced by ordinary compilers and the times quoted are in addition to the ordinary parsing time.

We shall assume that an arithmetic expression contains N tokens (identifiers, constants, and operators). In calculating the number of operations required to parse an expression we shall count each instance of an arithmetic operation, logical operation, push on stack, pop stack, and store as one operation.

The results are summarized in the following theorem:

<u>Theorem</u> [10] Let E be any arithmetic expression with N tokens. The additional time to parse E is at most:
1. $13N + 8$ by Beatty's algorithm
2. $31N \log_2 N$ by Brent's algorithm

References

(1) J.L. Baer and D.P. Bovet, "Compilation of Arithmetic Expressions for Parallel Computations", Proc. IFIP Congress 1968, pp. 340-346.

(2) J.C. Beatty, "An Axiomatic Approach to Code Optimization for Expressions", Journal of the ACM (October, 1974), pp.613-640.

(3) J. Hellerman, "Parallel Processing of Algebraic Expressions", IEEE Trans. on Electronic Computers (January, 1966), pp. 82-91.

(4) J.S. Squire, "A Translation Algorithm for Multiple Processor Computers", Proc ACM 18th Nat. Conf., 1963.

(5) H.S. Stone, "One-pass Compilation of Arithmetic Expressions for a Parallel Processor", Comm. ACM (April, 1967), pp. 220-223.

(6) D. Kuck and Y. Muraoka, "Bounds on the Parallel Evaluation of Arithmetic Expressions Using Associativity and Commutivity", Acta Informatica (August, 1974), pp. 203-216.

(7) R.P. Brent, "The Parallel Evaluation of General Arithmetic Expressions", Journal of the ACM, (April, 1974), pp. 201-206.

(8) Y. Muraoka, Parallelism Exposure and Exploitation in Programs, Dept. of Computer Science, University of Illinois at Urbana-Champaign, Report 424, (February, 1971), 236 pp.

(9) R.P. Brent, D.J. Kuck, and K.M. Maruyama, "The Parallel Evaluation of Arithmetic Expressions without Division", IEEE Trans. on Computers, (May, 1973), pp. 532-534.

(10) R.A. Towle, Control and Data Dependence for Program Transformations, Dept. of Computer Science, University of Illinois at Urbana-Champaign, UIUCDCS-R-76-788, (March, 1976), 113 pp.

This work was sponsored by the National Science Foundation through grant US NSF DCR73-07980 A02 while the author was at the University of Illinois.