A. P. ERSHOV

1958

Abstract

Full Text

A. P. ERSHOV

ON THE PROGRAMMING OF ARITHMETIC OPERATORS

(Presented by Academician S. L. Sobolev on 2 VII 1957)

The concepts used without explanation are taken from ([1]).

1°. Algorithms for programming arithmetic operators (arith. op.) consist of three parts.

The first part A1 successively generates the commands of the program of the arith. op.

The second part A2, for each constructed command, generates a conditional number (cond. n.) denoting the result of the programmed operation, and replaces the programmed expression in the formula by it. In the operation of A2, occurrences of identical expressions in the formula of the arith. op. are identified, so as not to program identical expressions repeatedly (economy of commands).

The third part A3 replaces, in the constructed program, the cond. n. denoting intermediate results by the codes of working cells (w. c.). The article proposes new principles for constructing the algorithms A2 and A3.

2°. **Assumptions and definitions.** The programming of an arith. op. is performed on a three-address computing machine for itself. The left-hand sides of the formulas of an arith. op. are superpositions of two-place and one-place operations, each of which is realized by one command. Each command has one binary digit σ, which is not included either in the operation code or in the address part of the command. Algorithm A1 generates the commands of the arith. op. in the form

a	b	c	σ	θ

where θ is the operation code, a and b are cond. n. denoting the components, and c is a cond. n. denoting the result (if θ is a one-place operation, then $b = 0$; $c \neq 0$ only for a **result command**, which is the final command for computing a formula of the arith. op.; the contents of the digit σ are initially equal to 0). The **array of a finished operator** (a. f. op.) is a group of n cells with addresses $L + 1, \ldots, L + n$, in which are placed the commands of the arith. op. generated by algorithm A1. The **array of result commands** (a. r. c.) is the group of cells in which all the result commands of the arith. op. are placed consecutively. A **conditional number of the 1st kind** denotes a

Fig. 1

Figure 1: Fig. 1

variable or a constant entering the formula. A **conditional number of the 2nd kind** denotes an intermediate result in the computation of the formula. It is generated by algorithm A2 and, for each non-result command, is equal to the address of this command in the a. f. op. The **scale** of the a. f. op. is a group of consecutively located memory cells with continuous numbering of digits, under which the s-th digit of the scale of the a. f. op. corresponds to the cell $L+s$ of the a. f. op. The scale of cond. n. of the 1st kind has an analogous structure. The **array of working cells** (a. w. c.) is a group of cells with addresses $r+1, \ldots, r+m$, where $r+1, \ldots, r+m$ are the codes of the w. c.

The **array of the completed program** is the group of cells in which the completed program of ar. op. will be placed. The symbol (T) denotes the contents of cell T.

3°. In the existing methods of command saving, the total operating time of algorithm A2 is proportional to the square of the number of commands in the program of ar. op.

Fig. 1 gives a scheme of algorithm A2, which makes it possible to carry out command saving in time proportional to the number of commands in the program of ar. op. The basis of the proposed algorithm is the assumption that there exists some integer-valued function $F = F(\theta, a, b)$ $(L+1 \leq F \leq L+n)$, defined for any command of ar. op.

a	b	c	σ	θ

Fig. 1

The operation of algorithm A2 begins after algorithm A1 has constructed the next command K of ar. op. (for simplicity, algorithm A2 is described which does not perform saving of resultant commands).

Operator 1 examines whether command K is resultant (no –operator 2).

Operator 2 computes $F(\theta, a, b)$ for command K and sends the result to cell S. Obviously, $L+1 \leq (S) \leq L+n$. Let $(S) = L+p$.

Operator 3 checks whether $(L+p)$ is equal to zero (no –operator 4).

Operator 4 checks whether, in commands K and $(L+p)$, the operation codes, the first two addresses, and the digits σ coincide (yes –exit I).

Operator 5 increases p by one if $p < n$, and places $L+1$ in cell S if $p = n$.

Operators 6–8 operate if command K has not been saved.

Operator 6 examines the conditional parts of the address part of command K. If among them there are conditional parts of the 1st kind, ones are placed in the corresponding digits of the scale of conditional parts of the 1st kind and in the p-th digit of the scale of the m. g. op. (before the beginning of programming of ar. op., zeros stand in all digits of both scales).

Operator 7 computes for command K certain quantities needed for the operation of algorithm A3 (see 5°).

Operator 8 sends command K to cell $L + p$.

Operator 9 operates if K is a resultant command ($c \neq 0$). If in the digit of the scale of conditional parts of the 1st kind corresponding to conditional part c there is a one, then, with the aid of the scale of the m. g. op., the commands of the m. g. op. containing conditional parts of the 1st kind are scanned. Commands containing conditional part c are marked by a one in digit σ. Consequently, during the operation of operator 4, none of the commands containing conditional part c in the address part and constructed after K will coincide with any of the commands constructed before K. Then K is transferred to the next free cell of the m. r. c.

Scheme A2 has two exits, I and II. At exit I, cell S contains a conditional part of the 2nd kind, denoting the result of the constructed nonresultant command. Exit II corresponds to a resultant command.

4°. The duration of operation of A2 is determined by the number of repetitions of operators 3-5. This number depends on the distribution of the values of $F(\theta, a, b)$ in the interval $[L + 1, L + n]$. (We note that the usual algorithms for command saving correspond to $F(\theta, a, b) = L+1$.) Obviously, the most favorable case is the uniform distribution of the values of $F(\theta, a, b)$ on $[L + 1, L + n]$ for a random composition of the formulas of ar. op. In this case it is possible to compute the mathematical expectation φ of the number of repetitions of operators 3-5 as a function of the number k of commands of the program being constructed and the number of cells n.

in m.a. op. ($\varphi = \varphi_n(k)$). The derivation of analytic estimates proved difficult, and the values $\varphi_n(k)$ were computed by the Monte Carlo method. Figure 2 shows the curves obtained for $\lg_{10} \varphi_n(k)$ for $n = 150$ (50) 450. For comparison, the curves $\lg_{10} k$ and $\lg_{10}(k^2/2)$ are given. From an analysis of the results obtained it follows that, practically for all n, if m.a. op. exceeds the number of instructions in ar. op. by at least one and a half times, then each instruction of ar. op. will on average have no more than one execution of operators 3–5.

Simplicity of computation and a sufficiently good uniformity of distribution of the values are the only criteria limiting the choice of $F(\theta, a, b)$. For the actual construction of $F(\theta, a, b)$ it is advisable to use methods for obtaining uniformly distributed pseudorandom numbers. Of great importance for a successful choice of $F(\theta, a, b)$ is a study of the statistical structure of the formulas of programmable ar. op.

Fig. 2

Figure 2: Fig. 2

5°. Between the operations entering into the formula of an ar. op. there are definite relations concerning the order of their execution. These relations are specified by the rule that, for any operation of the formula, its components are first computed, and only then the operation itself. Therefore a formula may be regarded as a partially ordered set of the operations contained in it. In constructing a program for computing a formula, the operations are ordered, this being caused by the sequential arrangement of instructions in the program; hence the problem of programming a formula may be posed as the problem of ordering the operations of the formula while preserving the given partial order. It is obvious that the number of working cells needed for its computation depends on the way in which the operations of the formula are ordered. For example, to compute the formula

$$ab + (cd - ef(gh + ij(kl - mn))) \Rightarrow y$$

when the operations are performed from left to right, 7 working cells are required, whereas if the computation is begun with the innermost parentheses, only 2 working cells are required. In this connection the following problem arises: to find such an admissible ordering of the operations of a formula for which its computation requires the minimal number of working cells.

Fig. 2

The problem posed is partially solved by means of algorithm A3 for ordering the operations of a formula, whose scheme is shown in Fig. 3. Preparation for the operation of algorithm A3 consists in computing, by operator 7 of algorithm A2, for each nonresult instruction K of two-place functions whose values are placed in the third address of the instruction before

passing it to the m. g. op. The first function—the **order function** $P(K)$—is specified by an inductive definition:

A) If the instruction K does not contain a conditional part of the 2nd kind, then $P(K) = 1$.

B_1) If in one address of the instruction K there is a conditional part of the 2nd kind denoting the result of the instruction K_1, then $P(K) = P(K_1)$.

B_2) If in the first and second addresses of the instruction K there are conditional parts of the 2nd kind denoting the results of the instructions K_1 and K_2, then

$$P(K) = \begin{cases} \max\{P(K_1), P(K_2)\}, & \text{if } P(K_1) \neq P(K_2); \\ P(K_1) + 1, & \text{if } P(K_1) = P(K_2). \end{cases}$$

Fig. 3

Figure 3: Fig. 3

The second function—the **occurrence counter**—is computed as follows. When the instruction K is transferred to the m. g. op., its occurrence counter is equal to 0. If an instruction K' is then transferred to the m. g. op. and contains a conditional part denoting the result of the instruction K, 1 is added to the occurrence counter of the instruction K.

Algorithm A3 begins operating after the completion of A1 and A2.

Operator 1 transfers to cell R the next resultant instruction of the ar. op., beginning with the last cell of the m. r. c. Let R contain the instruction K.

Operator 2 replaces in K conditional parts of the 2nd kind by codes of r. cells. If K contains a conditional part of the 2nd kind $L+s$, the contents of the cell $L+s$ are examined. The instruction K' from $L+s$ is transferred to the first free cell of the m. r. cells. In $L+s$ the instruction K' is replaced by the address $r+i$, indicating where K' was transferred. If, however, K', during the processing of one of the preceding instructions of the ar. op., has already been replaced by the address $r+i$, 1 is subtracted from the occurrence counter of the instruction K' located in $r+i$. In K the conditional part $L+s$ is replaced by the code $r+i$ of an r. cell. If K contains two conditional parts of the 2nd kind, $L+s_1$ and $L+s_2$, and the cells $L+s_1$ and $L+s_2$ contain the instructions K_1 and K_2, then into the m. r. cells there is first transferred that one of the instructions K_1, K_2 for which the value of the order function is larger.

Fig. 3

Operator 3 transfers K to the next cell of the array of the finished program, beginning with the last cell.

Operator 4, scanning the m. r. cells from the end, finds the first instruction with occurrence counter equal to 1. If no such instruction is found in the m. r. cells, or if there is not a single instruction in the m. r. cells, control is transferred to operator 6.

Operator 5 transfers the found instruction from cell $r+j$ to R, puts $r+j$ in the third address of this instruction, and then clears cell $r+j$.

Operator 6 transfers control to operator 1 if not all instructions have yet been transferred from the m. r. c.

For ar. op. in which the occurrence counter of each instruction is equal to 1, the algorithm described completely solves the problem of the most advantageous ordering. This follows from the following two assertions, valid under the restrictions indicated above:

1. For the purpose of minimizing the expenditure of working cells, for any

two-place operation it is necessary first to compute that one of its components for which the minimum number of r. cells required for its computation is larger.

2. For each instruction its order function is equal to the minimum number of r. cells required to compute the expression in which the last operation is realized by the given instruction.

Moscow State University
named after M. V. Lomonosov

Received
27 VI 1957

REFERENCES

1. A. P. Ershov, *A programming program for BESM*, Moscow, 1958.

Note: Figure translations are in progress. See original paper for figures.