BULLETIN DE L'ACADÉMIE
POLONAISE DES SCIENCES
Série des sciences techniques
Volume IX, No. 5 — 1961

E 16201

New Conception of Two-address Computer

by

Z. PAWLAK

Presented by P. SZULKIN on February 3, 1961

In a three-address computer two addresses point out whence both operational arguments should be taken up (assuming the dyadic operation is to be carried out); the third address points to the place where the result of operation ought to be located.

This paper contains the description of two ideas of digital computers, in which giving of the address of the result is superfluous.

For simplicity, only arithmetical commands have been taken under consideration, input-output commands and others being disregarded, because they may be identical with those in the three-address computer.

1. Language of the machine M_1

As primitive symbols we assume the symbols of four arithmetical operations: addition "+", subtraction "—", multiplication "·" and division "/" and, moreover, lower case italics with low subscripts, which denote variables or otherwise — addresses.

The language of the machine under consideration we will define as follows:

i. $+\alpha\beta$, $-\alpha\beta$, $\cdot\alpha\beta$, $/\alpha\beta$ are commands, in which symbols α and β denote addresses,

ii. If $\alpha_1, \ldots, \alpha_n$ are commands, then we call $[\alpha_1, \ldots, \alpha_n]$ a programme or a well-formed expression.

If the programme comprises n commands, we say that this programme is of n length, command α_i being the i-th command.

If α_i is the i-th command, then x_{i_1} and x_{i_2} denote respectively addresses of left and right arguments of the command.

If α is a n-length programme and β being an ordered sequence of all $2n+1$ addresses $x_1, x_2, \ldots, x_{2n+1}$ occurring in α, then $\alpha(\beta)$ will be called a formula; α we name "head", whereas (β) a—"tail" of the formula. The head of the formula we will denote by the letters F, G, H.

If in the formula $F(x_1, \ldots, x_{2n+1})$ the number k is substituted for variable x_i, then we write $F(x_1, x_2, \ldots, x_i(k), \ldots, x_{2n+1})$. The commands are performed in the order of their rising subscripts.

BIBLIOTEKA
Instytutu Matematycznego U. W.
Nr Inw. E 16201

We substitute the result of order α_i for the variable x_{i+n+1}. One may easily observe that the arbitrary function, determined by elementary output functions, addition, subtraction, multiplication and division and the rule of substitution, may be written in the language given above. Namely, $(x_1 \cdot x_2 + x_3)/x_4 \cdot x_5$ we write in the form $[\cdot x_1\ x_2, + x_6\ x_3, \cdot x_4\ x_5, /x_6\ x_7]\ (x_1, x_2, x_3, x_4, x_5, x_6, x_7)$.

2. Organization of two-address machine M_1

A machine realizing the language, given in the preceding section of the paper, consists of the following elements: memory P, arithmometer A, register of orders R, order counter L_1 and, moreover, of the counter of partial results L_2. Consecutive locations of the memory are denoted by $x_1, x_2, \ldots, x_m$, respectively ,where $m > 2n+1$. The head of the formula is recorded in the memory comprising addresses greater than $2n+1$; in each location of the memory one command is written. The order counter brings about the "taking" of consecutive orders from the memory into the control register, whereas the counter of partial results induces locating of partial results on the proper places in the memory.

As a rule, the counter of partial results is needless because its "duty" may be fulfilled by the counter of orders, since in the given conception the address of results is closely connected with the address of order, thus it may be computed by the control unit from the address of the command under execution. In practical realization it would be more convenient to locate all partial results in places of the memory established "for ever", e.g. either on the last or the first one hundred. In this way, partial results would always be located on the same places — independently of the site of the formula in the memory — which would somewhat facilitate programming.

We assume, then, that the partial results are necessary merely in the course of computation of the given formula, and thus they may be erased from the memory at the time when the value of the formula is computed.

3. Language of machine M_2

In the machine just discussed, it is necessary to supply both arguments of each arithmetical operation. Now we will discuss a new conception of machine, differring from the former, in which, since the suitable sequence of arithmetical operations is taken into account, the addresses of the arguments being partial results are not required.

Language for such a machine we will define as follows:

As primitive symbols we assume, analogically to the previous conception of the machine, in the former part of the paper, symbols of four arithmetical operations, lower case italics with subscripts and a special symbol * (asterisk) denoting partial results.

i. $+\alpha\beta$, $-\alpha\beta$, $\cdot\alpha\beta$, $/\alpha\beta$, are commands, in which α, β are addresses or asterisks.

ii. If $\alpha_1, \ldots, \alpha_n$ are commands, then $[\alpha_1, \ldots, \alpha_n]$ is a programme or a well-formed expression.

If α is a programme of n-length and β a sequence of $n+1$ variables (of all variables occurring in the programme), then we call $\alpha(\beta)$ a formula.

If the "data" are arguments of an operation having been just computed, we perform the operation on the "basis" of the respective numbers written down within the tail of the formula.

Assume that an additional sequence of variables $y_1, y_2, \ldots, y_n$ is given. The result of operation of command α_i is written on the position of variable y_i.

If k-th asterisk is the argument (counting from left), then as its value we take the number recorded on the position of variable y_k. By this way we disregard the addresses of partial results.

Thus the example discussed above takes the following form:

$\cdot x_1 x_2, + * x_3, \cdot x_4 x_5, / ** (x_1, x_2, x_3, x_4 x_5)$.

Variables, on which partial results are to be written, are not given.

4. Organization of machine M_2

The machine M_2, similarly to machine M_1, consists of: memory P, arithmometer A, order register R, counter of order L_1 and counter of partial results L_2 and, moreover, of counter of argument. Counter L_1 causes the "taking" of consecutive commands into the order register. Counter L_2 brings about appropriate location of partial results into the memory whereas counter L_3 counts successive asterisks in the programme, and correspondingly to the latter "action" causes taking up of partial results, which have to take part as arguments of operation.

It seems that the organizations presented in this paper may find practical application in certain cases in the construction of automatic computers.

INSTITUTE OF MATHEMATICS, POLISH ACADEMY OF SCIENCES
(INSTYTUT MATEMATYCZNY, PAN)

BIBLIOTEKA INSTYTUT MATEMATYCZNY U.W.