BULLETIN DE L'ACADÉMIE
POLONAISE DES SCIENCES
Série des sciences techniques
Volume IX, No. 2 — 1961

E16205

Organization of Address-free Computer with Separate Memory of Partial Results

by

Z. PAWLAK

Presented by P. SZULKIN on November 10, 1960

In the papers [1] and [2] a certain method of organization of an address-free computer was given; the present paper describes the organization of an address-free computer, which is particularly useful for realization of a computer with ferrite memory.

This problem was set by Professor A. H. Taub (personal communication).

In the computer described in [1], each partial result was located in the memory under the address having been computed from addresses of the arguments. In [2] the author demonstrates the organization of a computer, in which partial result was placed "at the nearest blank space".

In the first case (see [1]) a special unit for calculating the addresses of partial results is necessary. The second idea (cf. [2]) is convenient rather for realization of a computer with a serial memory, e.g. a drum memory, because selection of the nearest blank space in the serial memory is very simple and requires little hardware.

The selection of the nearest blank space in a parallel memory, for instance the ferrite memory is more tedious and time consuming.

Therefore, the address-free computer under consideration requires a somewhat different organization than the computers previously discussed.

Language of the computer

We will define the language of the computer as follows. As primitive symbols we assume:

i. $a, b, c, \ldots, x, y$, which are independent variables,

ii. $+, -, \cdot, /$ denote the symbols of dyadic operations: adding, subtracting, multiplying and dividing,

iii. $0, 1$ are constants, the meaning of which will be explained in the further part of the paper.

The expression 11Δ, where Δ is one of the symbols of operations, is called a well-formed expression.

If α and β are well-formed expressions, then $\alpha\beta^*$ is also a well-formed expression, where β^* denotes an expression which is obtained from β by replacing an arbitrary "1" by "0".

If α is a well-formed expression containing n of "1"-s ($n \geqslant 2$) and β — a sequence of n arbitrary independent variables, then $\alpha\beta$ will be called a formula.

For the sake of clarity the formula may be written in the form $[\alpha]\,(\beta)$, however, the use of parenthesis is needless with reference to the computer.

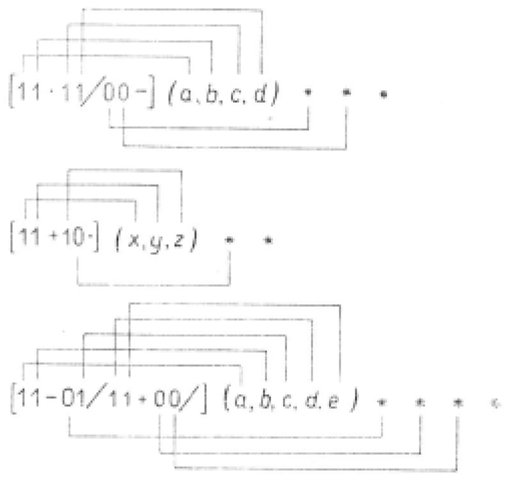
Let us give some examples of formulae in the form of the aforesaid language

$$[11\cdot 11/00\,-]\,(a, b, c, d)$$
$$[11+10\cdot]\,(x, y, z)$$
$$[11-01/11+00/]\,(a, b, c, d, e)$$

The expression in the square brackets can be interpreted as a function symbol in conventional notation, e.g. $G\,(a, b, c, d)$, $H\,(x, y, z)$, $K\,(a, b, c, d, e)$.

To facilitate the reading of the formulae we add to the latter $n+1$ of "asterisks", where n is the number of zeros in the function symbol.

We further set a correspondence between "1"-s and independent variables and the same correspondence between zeros and asterisks

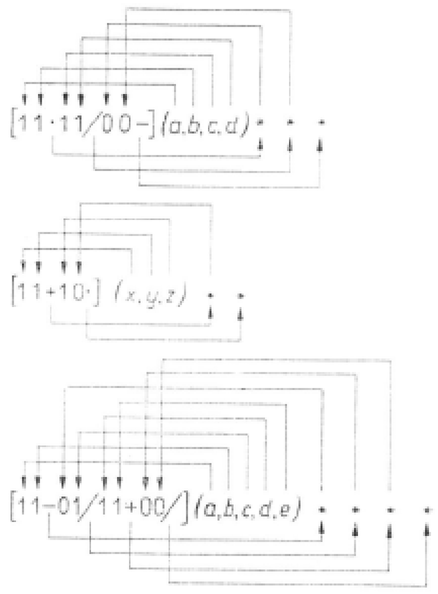

[11 · 11/00 −] (a, b, c, d) * * *
[11 + 10 ·] (x, y, z) * *
[11 − 01/11 + 00/] (a, b, c, d, e) * * * *

Scheme 1

in the following way: successive independent variables correspond to successive "1"-s, and successive asterisks correspond to successive zeros. This correspondence, for the examples given above, is illustrated below (Scheme 1).

The last asterisk does not correspond to any zero. As arguments of each operation are independent variables or asterisks corresponding to zeros and "1"-s, respectively, located at the left-hand of the given symbol of operation. The successive partial results are recorded in place of successive asterisks, starting with the first asterisk.

The course of computation, for the examples given above, is illustrated by the following schemes (Scheme 2):


[11·11/00−] (a,b,c,d) * * *
[11+10·] (x,y,z) * *
[11−01/11+00/] (a,b,c,d,e) * * * *

Scheme 2

It is easy to show that the formalism just presented is equivalent to parenthetical language. The examples written in the conventional symbolism have the following form:

$$((a\cdot b) - (c/d))$$
$$(z\cdot(x+y))$$
$$(((a-b)/c)/(d+e))$$

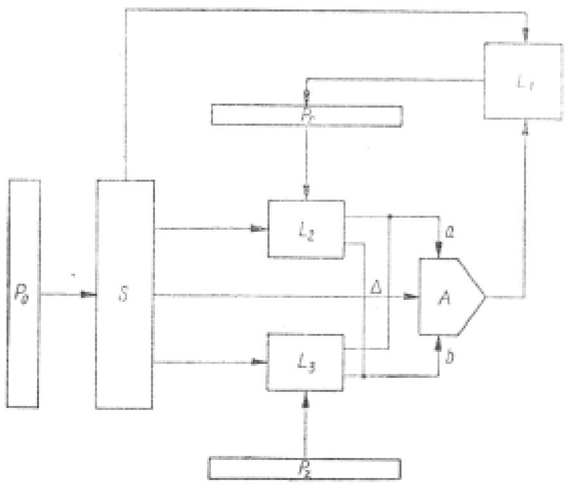
It may also be shown that the method of representing the functions demonstrated here is equivalent to the λ-system of Church [3].

Organization of the computer

The language discussed in the preceding section of the paper has a very simple technical realization. The organization of the computer, working according to the above presented formalized language, is illustrated by the Figure.

The computer consists of three memories: partial results memory — *Pr*, data memory — *Pz*, memory of operation — *Po*. Moreover, it consists of arithmometer — *A*, control unit — *S* and three counters *L*1, *L*2 and *L*3.

In the memory — *Po* are recorded consecutive symbols of operations of the formula and zeros and "1"-s corresponding to them. In the memory — *Pz* the consecutive values of independent variables are located.



The control unit — *S* takes up the consecutive symbols from the memory — *Po*, sets the arithmometer — *A* to a suitable operation, selects, from the *Pz*- and *Pr* memories, both arguments of operation and places in the memory — *Pr* the result of each computation.

After each operation the content of the counter *L*1 is increased by 1. The counters *L*2 and *L*3 increase their contents, according to the rules given below:

i. If we take up the expression 11 Δ from *Po*, then both arguments are consecutively received from memory — *Pz*, the counter *L*3 increases its content by 2, whereas the counter *L*2 does not.

ii. If we take up the expression 00 Δ from *Po*, then both arguments are received consecutively from memory — *Pr*; the counter *L*2 increases its value by 2; but the counter *L*3 does not.

iii. If we take up the expression 01 Δ from *Po*, then the left argument is received from memory — *Pr*, the right argument — from memory — *Pz*, and both counters (*L*2 and *L*3) increase their contents by 1.

iv. If we take up successively the expression 10 Δ, from *Po*, then the left argument is received from memory — *Pz*, and the right — from memory — *Pr*; the contents of both counters — *L*2 and *L*3 — are increased by 1.

The organization of an address-free computer considered enables the location of partial results in the memory without necessity of selection of successive locations of the memory. This ability in a case of application of the ferrite memory allows to obtain a high speed in computations.

Sometimes it may be more convenient to dispose of a common memory of data and operations. Then the language of the computer undergoes certain modification.

The primitive symbols remain the same, but the formulae will be defined somewhat differently:

The expression $xy11\ \Delta$ is a formula.

If α and β are formulae, then $\alpha\beta^*$ is a formula also, where β^* is the expression obtained from β by replacing the arbitrary "1" by zero in β and by omitting in β the independent variable corresponding to that "1".

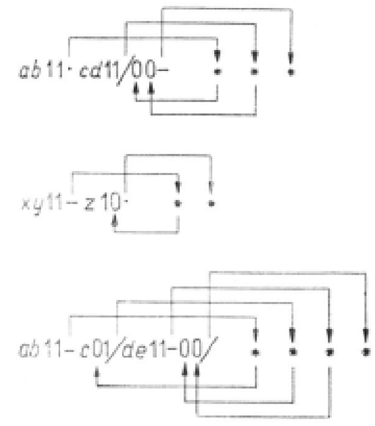
In this new "language" some examples are given below:

$$ab\,11\cdot cd\,11/00-$$

$$xy\,11-z\,10\cdot$$

$$ab\,11-c\,01/de\,11-00/$$

The "course" of computation in the new formalism is as follows (Scheme 3).



Scheme 3

This new scheme of the computer differs but slightly from the first one, represented in the Figure (p. 126).

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

REFERENCES

[1] Z. P a w l a k, *The organization of digital computers and computable functions*, Bull. Acad. Polon. Sci., Sér. sci. techn., **8** (1960), 41.

[2] — , *Organization of the address-free digital computer for calculating simple arithmetical expressions*, ibid., **8** (1960), 193.

[3] A. C h u r c h, *A set of postulates for the foundation of logics*, Ann. of Math., 2nd. Ser., 33, p.346.