ON A NEW TYPE OF ELECTRONIC COMPUTING MACHINES

CYBERNETICS AND CONTROL THEORY

1966

SovietRxiv

Abstract

Full Text

UDC 519.95

CYBERNETICS AND CONTROL THEORY

V. L. ARLAZAROV, A. S. KRONROD, V. A. KRONROD

ON A NEW TYPE OF ELECTRONIC COMPUTING MACHINES

(Presented by Academician M. V. Keldysh on 16 VIII 1966)

1°. The discrepancy between memory access time and the time required to perform operations in an electronic computer. In existing computer designs no distinction is made between cells that store numbers and cells that contain instructions. John von Neumann' s important idea consisted precisely in having the machine operate on instructions in exactly the same way as on numbers. At present, the slow reading of words from memory (not less than 10^{-6} sec. for the best ferrite-memory models) limits the operating speed of the machine, since modern transistors make it possible to perform operations several times faster. The use of N. I. Bessonov' s cascade principle can accelerate the execution of arithmetic operations by a factor of 5–10.

The use of ultrafast memory on active elements for storing only recently used operands (numbers, logical words) improves matters, but now the technical speed of the machine begins to be determined by the program reading time.

2°. With a large number of index registers (IR), instruction readdressing is not needed. Experience with the BESM-6 machine, which has 15_{10} IRs, has shown that operations on instructions are practically unnecessary in such a machine. V. A. Kronrod and V. L. Arlazarov compiled for the BESM-6 a library B-65 containing not a single readdressing and not a single instruction formation, although the authors of B-65 by no means set themselves the goal of avoiding readdressing, but were concerned only with the speed of program execution and memory economy.

3°. Separation of memory. One of the possible ways of substantially increasing the speed of a computer is now clear: for storing the program it is necessary to create a separate memory permitting rapid reading, while comparatively slow writing can be allowed. In addition, of course, the usual active memory for storing numbers and variable words will remain.

4°. Word length of memory. The separation of memory into "numeric" (active) and "program" (semi-active) makes it possible to have different word lengths of cells in these blocks. For computational problems, a "numeric" cell

may contain, for example, 45 bits (1 bit for the sign, 8 bits for the exponent, 36 for the mantissa), or even 42 bits (33 bits in the mantissa).

For noncomputational problems it is desirable to have more bits, for example 66 or even 72. However, here too one can reduce the length, say to 33 bits, if this is justified economically. With a long active-memory cell for floating-point numbers, only part of the bits can safely be used; this will reduce the volume of equipment and speed up floating-point operations. An active-memory capacity of $2^{15} = 32\,768$ cells is desirable, but smaller capacities are also admissible. For program memory it is likewise good to have 2^{15} cells. The desirable number of index registers is $63_{10} = 77_8$.

In terms of addressing, 5 basic variants are possible:

A. Single-address machines. 15–18 bits for the address, 6 for the IR, 7–8 for the operation code, for a total of 28–32 bits. If reading from memory takes practically no time, writing two instructions into a cell does not help.

B. One-and-a-half-address machines with a short additional address. If the number of cells on active elements is 63_{10}, then the short address should naturally be made 6-bit. One bit will go to indicating whether the I or II address in the given instruction is short, and another 3–6 bits to the IR for the short address. In all, this gives 38–45 bits.

C. Two-address machines. Each address will take 15–18 bits, and another 6 to indicate the IR. Altogether this gives 49–56 bits.

D. $2\frac{1}{2}$-address machines. A 6-bit short address is added, and 3–6 bits to indicate its IR. Another 2 bits are needed to indicate which address is the short one. Altogether we have 60–70 bits.

E. Three-address machines. For the same memory capacities, 45–54 bits are needed for addresses, 18 bits for the IRs, and, for example, 9 bits for the code (in a 3-address machine it is useful to have a richer set of operations). The total comes to 72–81 bits. Moreover, since the instructions are quasi-stationary, no operations with them will have to be performed, and the equipment of the arithmetic unit and the control unit will not increase.

5°. Choice of the addressing of the machine. The experience of mathematicians' work both here and abroad has shown that more than half the time is spent not on computation, but on debugging programs. Moreover, if, in order to save debugging time, one begins simplifying a program, such simplification often costs an increase in computation time by 10, 100, or more times. Therefore, when choosing the addressing of a machine, one must first of all listen to the opinion of the user-mathematician.

The authors have worked on one-address (BESM-6), two-address (Minsk), and three-address (M-2, M-20, BESM-3M) machines. It turned out that three-address machines noticeably save the mathematician' s effort and time. The overall productivity of the human–machine system rises sharply. Therefore the

authors strongly advocate the transition (return) to three-address machines. Incidentally, in this case A. L. Brudno' s system of programming in meaningful designations removes a whole series of problems connected with translators, and this appreciably speeds up the commissioning of new machine models. For 1-$2\frac{1}{2}$-address machines the system mentioned is also suitable, but the usual, almost "old-mathematical," writing of the meaningful part is transformed in an unfavorable direction, programming becomes more burdensome, and as a result time and money are lost.

6°. Fast cells. In machines of the type under consideration it is necessary to allocate at least 7, and much better $77_8 = 63_{10}$, "numeric" cells on active elements; otherwise the capabilities of the machine will not be realized.

Let us note that in the presence of fast cells the advantage in speed of single-address machines over three-address machines, associated with a reduction in the number of intermediate writes, is completely lost.

7°. Various remarks.

1. Cells for instructions may have the same numbers as cells for numbers. However, if the word length of the instruction permits, it is better to have these numbers different—then logical constants, as well as permanent numbers, can be kept among the instructions.

2. All registers of the machine, including the IRs, must receive addresses. The usefulness of this uncomplicated measure has been convincingly confirmed by two years' experience in operating an M-20 machine converted in this way at the Institute of Theoretical and Experimental Physics.

3. The instruction system of our serial machines* is obsolete. It should be discussed in a printed debate, in which the authors would willingly take part.

* At least 3-address.

4. The instruction set must necessarily include **all elementary machine operations** of the type: transfer from register to register, logical addition (multiplication) without output to ferrite memory, and so on. This will make it possible to **program new machine operations directly**, without rebuilding the machine. N. I. Bessonov' s commutator should be in every large machine—a small addition of equipment is repaid by rapid shifts, by acceleration of addition, and by the simplification of complex logical operations.

5. Cells on active elements should be assigned **small** numbers. Then several elementary machine operations can be written in a single instruction cell. For a machine with fast program memory this, to be sure, will save almost no time, but it will save space. For a machine with ordinary memory, time will also be saved.

6. In all machines the instruction **FA** (⌢1) should be provided. Experience has shown that programming is greatly simplified and accelerated by this.
7. The number "minus zero" must: a) not be produced as the result of floating operations; b) serve as a stop instruction; c) produce an Avost when an attempt is made to perform any **arithmetic** operation with this number.
8. The implementation of what was said in item 4, in the presence of fast memory for instructions, replaces microprogramming. If, for some reason, the creation of semi-active memory encounters serious difficulties (soldered memory is suitable only for a library), then microprogramming will have to be taken up seriously. The fact is that noncomputational tasks (game, logical, recognition tasks, etc.) are rapidly becoming the most important of all that is done on electronic computers.

And precisely in these tasks, as preliminary studies have shown, microprogramming, given the presence of at least a dozen fast registers, will yield a gain in speed by a factor of 8–10 or more.

Institute of Theoretical
and Experimental Physics

Received
18 VI 1966

REFERENCES CITED

1. G. M. Adelson-Velskii, A. L. Brudno et al., DAN, **154**, No. 3 (1964).

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