A Simple Experimental Computer with Negative Basis

By A. Łazarkiewicz and W. Balasiński

Summary. This paper presents the technical data, logic, and control organization of a simple experimental computer operating in the minus-two system. The principles for composing the instructions from elementary operations, the characteristics of minus-two computer arithmetic, and the logic of the arithmetic unit are briefly explained. The paper is divided into the following sections: 1, Introduction; 2, Fundamental Logical Circuits; 3, The Memory; 4, Block Diagram; 5, The Control Unit; 6, Coordination with Teleprinter; 7, Instructions; 8, Minus-Two System and Range of Numbers in the Computer; 9, The Adding Unit; 10, The Sign and Overflow Register; 11, Multiplication; 12, Acknowledgments.

1. Introduction. An experimental negative-base digital computer is being tested at the Warsaw Technical University. It is a small general-purpose machine for scientific and technical calculations which works at a speed of about 100 oper/sec. The computer makes use of the serial mode of operation; it is a single-address, fixed-point machine with a magnetic drum as the only memory. Punched paper tape has been used for input of data and the results are printed by means of a standard teleprinter.

It is intended to supply some university computation centers in Poland with an improved version of this computer for training purposes.

2. Fundamental Logical Circuits. Fundamental logical circuits have been worked out in dynamic tube technique with the use of germanium diodes for obtaining "and" and "or" logic. The arithmetic part of the machine and its control have been built up from a few normalized types of these circuits. The circuits are shown in Figure 1.

The delay unit is based on the Havens circuit used with different modifications in computers like the Pegasus, NORC, in the IBM 701 to some extent, etc. After an input pulse has been applied, an impulse, regenerated both in time and amplitude and delayed by a clock unit of time, appears at the output of the delay unit. A dynamic trigger may be obtained by feeding back the output to the input, as shown in Figure 2.

At the output of every fundamental circuit there is a cathode follower. A pulse representing the digit "one" has about 30 volts amplitude and lasts approximately 7.5 μs. The fundamental circuits are linked into blocks according to fixed rules which take into account delays and pulse deformations in amplitude caused by gates and negations.

3. The Memory. A magnetic drum serves as the memory. It is made of aluminium covered with iron oxide about 20 μ thick. The drum is fitted with heads set

Received June 1, 1960; revised December 15, 1960.

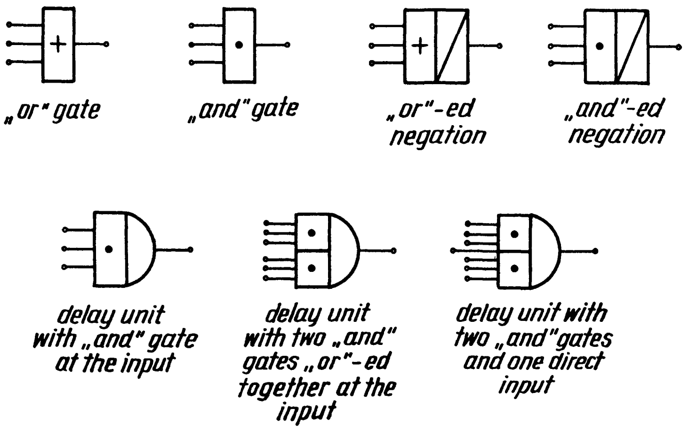
„or" gate
„and" gate
„or"-ed negation
„and"-ed negation
delay unit with „and" gate at the input
delay unit with two „and" gates „or"-ed together at the input
delay unit with two „and" gates and one direct input

FIG. 1.—Fundamental Logical Circuits.

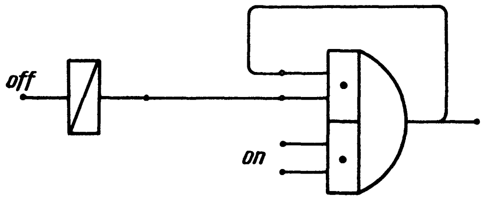

FIG. 2.—A Dynamic Trigger.

about 30 μ from its surface. They are made of permalloy strips 1 mm wide on which 160 turns of wire are wound. Over 2700 rpm give a maximum access time of 22 ms. A read-record technique of phase modulation has been adopted. A "one" is recorded by a current pulse flowing through the coil of the head, immediately followed by an equal but oppositely directed pulse. The direction of both current pulses is reversed in "zero" record. Each zero-one sequence contains an even number of positive and negative pulses so it has no direct component. The read-frequency varies from 50 kc for 010101 ··· to 100 kc for "zeros" or "ones" alone.

Switching the tracks for reading and writing is accomplished by means of an electronic switch. The organization of the computer allows neglect of the transition states in read control circuits.

In addition to the memory itself there are 3 dynamic circulating registers on the drum, each about 1 word long. They are the accumulator, the current order address register (the so-called counter register), and read-out instruction register.

Drum registers are equipped with a writing head and a reading head which are placed close to each other on the same track, as well as amplifiers for writing and reading which contain pulse-forming circuits. The two heads can be positioned to secure correct performance. To eliminate unwanted signals which may be picked up, the heads of the registers are screened with permalloy strips placed inside the cylindrical case holding the head (the case diameter is 7 mm).

Six-bit addresses of separate words are written on the address track and successive addresses are placed every 13 words, as below

·····59, 0, 5, 10, ················50, 55, 60, 1, 6······

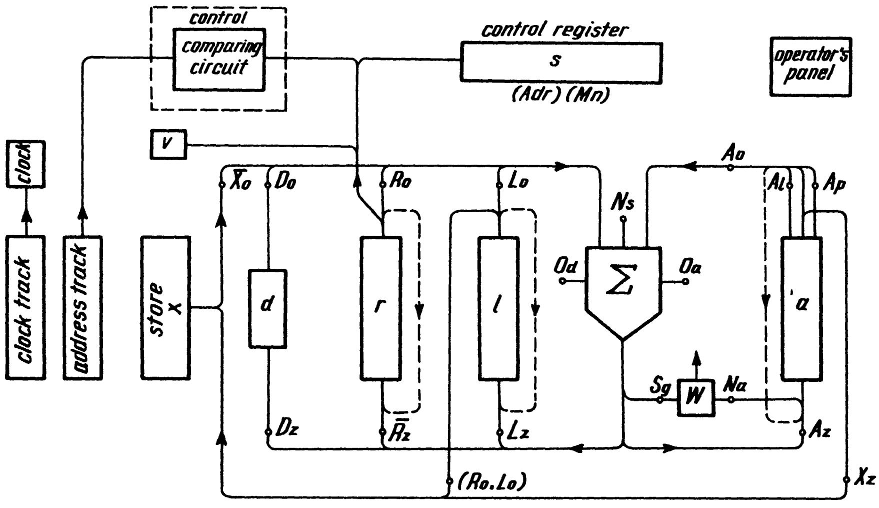

FIG. 3.—Block Diagram of the Computer.

Symbols:

x—drum memory	a—accumulator register
r—instruction register	d—input-output register
l—counter register	$\sum$—adding unit
v—track register	w—sign-overflow register
s—control register	R_o , L_o , X_z , A_z , et cetera—gates

This system of addressing has been chosen in order to facilitate optimum programming work in respect to the access time.

The capacity of the memory is 4096 words.

4. Block Diagram. The block scheme of the computer has been so designed that it is possible to combine transfer of numbers and instructions with performance of operations.

The adding unit is the central element of the transfer (Fig. 3). It is linked with the counter register "l," the instruction register "r," the memory and the input-output register "d" on one side, and with the accumulator "a" on the other side.

Writing on the drum is done directly from the accumulator or from the counter register. Individual parts of the computer are linked through control gates. To execute an instruction, the appropriate gates on the transfer line should be open, e.g., the line of transfer through gates Xo and Az must be free to bring a number from the memory into the accumulator. To add "one" to the counter register "l," gates Lo, Lz, Ns must be open. Gate Ns makes it possible to introduce plus one into the adder as the third argument in any operation.

The accumulator register is a drum register of one word length with the possibility of right and left shift. Figure 4 gives a simplified scheme of this register.

The basic loop is closed through two delay-lines, and the drum delay $T - 2$ equals 34 clock periods. The introduction of a new word from the adder takes place at $Az = 1$, and it automatically erases the previous value by breaking the loop by means of the negation n_2 . Shifting to the right occurs at $Ap = 1$; it shortens the

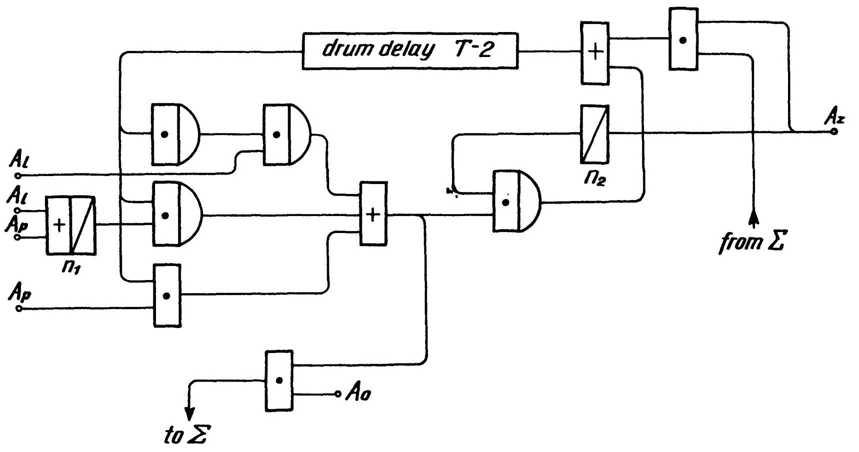

Fig. 4.—Logical Circuit of the Accumulator Register.

loop of the accumulator for a word period by one bit; the normal path is then blocked by negation n_1. When shifting to the left ($Al = 1$) the loop is lengthened by one bit; its normal path is blocked too.

The "*d*" register is a 5-bit trigger register serving as a buffer between the teleprinter and the computer. Its five positions correspond to five holes in the punched paper tape. Loading and unloading the "*d*" register by a converting device coordinated with the standard teleprinter is accomplished by means of a matrix controlled by a separate counter.

The current address of an actually executed instruction is kept by the counter register; the contents of the counter corresponding to the operational part of the order equal zero. During multiplication the counter register is used as the multiplier register.

The "*r*" register is the instruction register. It serves to keep order during the waiting time required for access to the right memory address from which the number is to be read out, and it acts as a buffer for the control register. During multiplication it stores the multiplicand.

The "*v*" register is a six-bit trigger register which holds the number of the track; it controls the matrix which defines the proper track on the drum memory.

The control register "*s*" holds the operational part of an instruction. It consists of twenty triggers, the states of which are determined by a serially introduced command from the "*r*" register or by parallel setting from the simple matrix of the control unit.

The "one" state of each position of the "*s*" register directly opens a gate for each appropriate elementary operation. Positions $\overline{Xo}$ and $\overline{Rz}$ are an exception, and their active states break their corresponding paths while "zero" states open them.

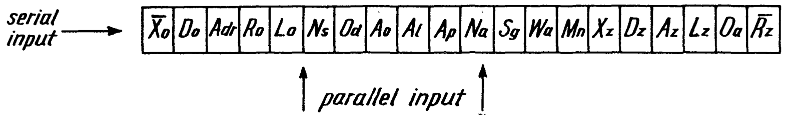

Fig. 5.—The Control Register.

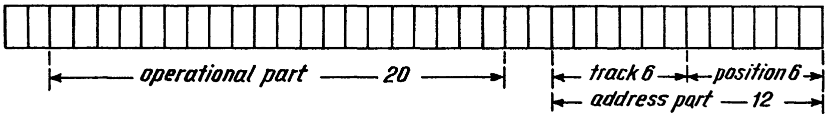

FIG. 6.—The Structure of an Instruction.

5. The Control Unit. An instruction is defined by 32 bits placed as follows (Fig. 6).

Twelve digits are reserved for the address part, 6 bits point out the proper track, and 6 bits—the time (position on the track). The operational part has 20 digits.

The machine works in three steps; namely, 1) forming the next instruction address, 2) extracting the instruction from the memory, 3) the execution of the instruction.

1. Instructions are placed in the memory in successive addresses. Defining the new instruction address requires the addition of "one" to the counter register "l." This is done by the adding unit. The new instruction address is forwarded at the same time to the "r" register. That occupies a period of time of one word, i.e., 36 clock units of time. During the next word period its operational part is shifted to the "s" register and the address part, defining the track on the drum store, to the "v" register. Since there is no "1" in the contents of the counter (in its operational part) we shall find only zeros in the "s" register, which means the operation of the second step. The time part (position part) of an instruction circulating among the others in the "r" register is compared to the output of the address track. As soon as coincidence is reached the second step begins.

2. In the second step the instruction is read out from the memory and brought through the adding unit to the instruction register "r." Then the operational part is shifted to the control register "s," and track address to the track register "v" in the next word period. Similarly, the time part of the address circulates in the "r" register and is compared with the address track. Coincidence starts the third step.

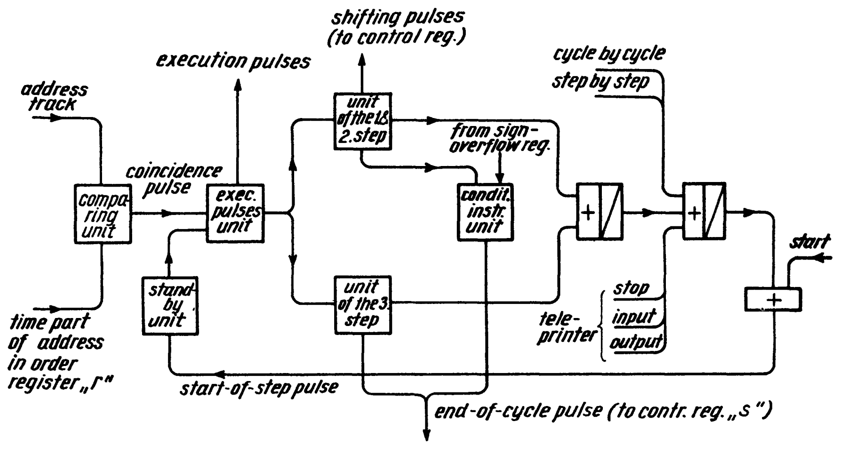

FIG. 7.—Block Diagram of the Control Unit.

3. During the third step the operation defined by the contents of the "*s*" register is executed.

It is evident that the minimum execution time for one instruction is equal to five word periods, i.e., 1.8 ms.

If the instruction to be obeyed is conditional and the sign-overflow register is in the "one" state, the "end of cycle" pulse is produced just before the execution time (the third step), which clears the control register and places the elementary operation of the first step in it. Thus, each (conditional) instruction can be ignored, depending upon fulfillment of a given condition.

The machine may be stopped by a stop-order or during reading and printing. The computer is also stopped in the cycle-by-cycle mode of operation before the execution of the instruction placed in the control register "*s*." Step-by-step operation causes a stop after every step.

To start the program from the address n, the number n must be placed in the counter register "*l*," e.g., from the "*d*" register by means of manually operated switches.

6. Input-Output. The "*d*" register and its converting unit link the computer with the input-output units (the teleprinter and the reader). Five positions of the "*d*" register correspond to the holes in the punched paper tape. Writing or reading is performed by the converting unit in a parallel manner with a matrix controlled by a separate simple counter. The converting unit produces the end-of-reading and the end-of-printing pulses according to the following rules, and receives the start of the printing pulse.

1. The instruction "read from *d*" is stopped until the end-of-reading pulse comes.
2. The instruction "put in *d*" sends the print pulse to the converting unit. During printing the next "put in *d*" instruction is withheld.

7. Instructions. The instructions consist of 20 elementary operations defined by the positions of the control register "*s*." The function of these positions is directly visible from the block diagram. Several remarks should be added:

1. *Adr*—the operation to which this position is adjoined is executed only on the address part of the word.
2. *Mn*—position "multiply" initiates the cycle of operations forming the product.
3. In order to write the contents of register "*l*" in the memory the positions *Ro* and *Lo* must be activated.
4. The "stop" order is realized by the positions $Do = Dz = 1$.
5. The execution of an instruction to which position *Wa* is adjoined depends upon the state of the sign-overflow register. If the register is in the "1" state, this order is omitted and the computer performs the next instruction.

The number of instructions composed of elementary operations is very great. Several operations may be executed simultaneously, and this increases the effective speed of computing.

Examples of instructions:

$(x + a)a$	n—add the number from address "n" to the contents of the accumulator

aL		—shift the contents of the accumulator one place left
Wa Adr (*r*)*l*	*n*-1	—if the sign-overflow register is in "0" state put the address part of this order into counter register "*l*" (jump to the instruction in "*n*"); if the sign-overflow register is in the "1" state carry out the next instruction
Adr Sg (*Nsr*)*a*	*n*	—add 1 to the address part of the instruction in the "*r*" register, transfer the result into the accumulator and examine the sign
Adr (*l*)*x*, (*r*)*l*	*n*	—put the current instruction address in memory cell "*n*"; set "*n*" in the counter register (jump to a subroutine)
aL, (*Nsr*)*r*	(10924-*n*)	—shift the contents of accumulator "*n*" places left ($n \geqq 2$); instruction *aL* is repeatedly executed until the carry produced by the address part of the instruction blocks the position $\overline{Rz}$ in the register "*s*."

A simple method for placing the input routine in the memory as well as a system for coding instructions has been worked out. A special routine located on a few protected drum tracks makes it possible to prepare programs even without knowledge of the internal organization and the number system used in the computer.

8. Minus-Two System and the Range of Numbers in the Computer.

Each real number may be represented in the minus-two expansion

$$\alpha = (a_n \cdots a_1 a_0 \cdot a_{-1} a_{-2} \cdots)_{-2} = \sum_{-\infty}^{N} a_i \cdot (-2)^i$$

where a takes the values of 0 and -1 written as 0 and 1 (the so-called negative notation) and N is the greatest i for which "a_i" still has a nonzero value.

In the minus-two system the weights of expansion orders are as follows

expansion order (i)	⋯ 3	2	1	0	−1	−2	−3 ⋯
value of "1"	8	−4	2	−1	$\frac{1}{2}$	$-\frac{1}{4}$	$\frac{1}{8}$

For expansions of positive numbers N is odd, for negative numbers N is even.

For numbers limited to $i_{\max}$ ($i_{\max} \geqq N$), the position of zero is nonsymmetrical, e.g., when $i_{\max} = 1$ the numbers are comprised within the range

$$-\tfrac{1}{3} \leqq \alpha \leqq \tfrac{2}{3}.$$

Shifting a number one place left or right is equivalent to multiplication or division by 2 with inversion of sign.

Examples of numbers represented in the (-2) system:

−3	111	$-\frac{1}{3}$	0.010101 ⋯	or	1.101010 ⋯
−2	110	$-\frac{1}{4}$	0.010000 ⋯		
−1	1	$-\frac{1}{6}$	0.011111 ⋯		
0	0	0	0.000000 ⋯		
1	11	$\frac{1}{6}$	0.110101 ⋯	or	0.001010 ⋯
2	10	$\frac{1}{4}$	0.110000 ⋯		
3	1101	$\frac{1}{3}$	0.111111 ⋯		
4	1100	$\frac{1}{2}$	0.100000 ⋯		
5	1111	$\frac{2}{3}$	0.101010 ⋯	or	11.010101 ⋯
6	1110	$\frac{3}{4}$	11.010000 ⋯		
7	1011	1	11.000000 ⋯		

The algorithms of operations $\alpha + \beta$, $\alpha - \beta$ and $-\alpha + \beta$ include two carries $p_1 = +1$ and $p_2 = -1$, and the algorithm of operation $-\alpha - \beta$ only one. The adder for carrying out this algorithm has a structure equivalent to the adder of the kind $\alpha + \beta$ for the well known binary system $(+2)$ (see Table 9b).

Multiplication can be performed by successive subtractions of the multiplicand from the shifted partial product according to the values of digits of the multiplier.

Algorithms for other operations such as division, rounding off, symmetrization of the range and even finding square-roots have structures similar to the algorithms in the $(+2)$ system commonly used and require an almost equal quantity of hardware.

Due to the experimental character of this machine, only multiplication was accomplished; other operations are left to programming.

The 36-bit word adopted in the computer is interpreted as follows

$$a_1a_0 \cdot a_{-1}a_{-2} \cdots a_{-33}a_{-34} .$$

In the memory, however, only digits behind the point are held, i.e., the numbers in the range

$$-\tfrac{1}{3} < \alpha < \tfrac{2}{3}.$$

The error caused by rejecting further bits lies in the range

$$-0.19 \cdot 10^{-10} < \delta < 0.39 \cdot 10^{-10}.$$

The accumulator contains the whole word, e.g., four times the greatest value of a number read out from memory.

In multiplication both arguments can be represented as

$$0a_0 \cdot a_{-1}a_{-2} \cdots a_{-34} .$$

This enables execution of many operations without overflow.

9. The Adding Unit. The adding unit is composed of two sign-inversion circuits, P_1 and P_2, as well as the adder proper.

Units of sign inversion are controlled by positions *0a* and *0d* of the "*s*" register. When $0a = 0d = 0$ (i.e., $\overline{0d} = \overline{0a} = 1$) both circuits P invert the signs of arguments α and β and the sum $\gamma = \alpha + \beta$ appears in the output of the adding unit. If $0a = 0$ and $0d = 1$ the sign of the number β is not changed and $\gamma = \alpha - \beta$.

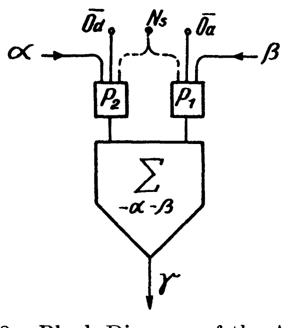

FIG. 8.—Block Diagram of the Adder.

a_i	p_i	$(-a)_i$	p_{i+1}
0	0	0	0
1	0	1	1
0	1	1	0
1	1	0	0

FIG. 9a.—Table of Sign Inversion of the Number α in $-\alpha$.

a_i	b_i	p_i	c_i	p_{i+1}
0	0	0	0	0
1	0	0	1	1
0	1	0	1	1
1	1	0	0	1
0	0	1	1	0
1	0	1	0	0
0	1	1	0	0
1	1	1	1	1

FIG. 9b.—Table of Operation $\gamma = -\alpha - \beta$.

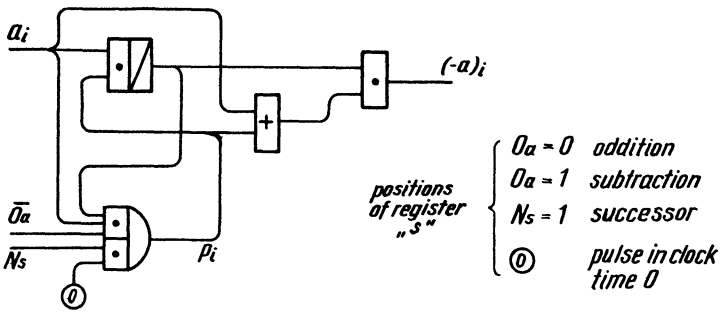

FIG. 10.—The Circuit for Inverting the Sign of a Number.

Position Ns of the control register adds one to the result of normal addition, e.g., $\gamma = \alpha + \beta + (-2)^{-34}$.

The algorithms of sign inversion and of operation $\gamma = -\alpha - \beta$ are presented in Tables 9a and 9b. In these tables a_i, b_i denote digits of arguments α, β; $(-a)_i$, c_i — digits of the result $-\alpha$, γ; p_i — carries.

Inversion of the sign of a number is accomplished by the logical circuit shown in Figure 10.

The suppression of carries by the position $Oa = 1$ has the effect of transporting the incoming number without changing the sign, which causes subtraction. The existence of an initial carry means adding of $(-2)^{-34}$ to both arguments but, of course, only when $Oa = 1$ or $Od = 1$.

It is evident from the truth-table for the adding unit that if the carries are suppressed and $p_i = a_i + b_i$ is assumed (in Boolean algebra), the adding unit performs the operation of the logical product $\gamma = \alpha \;\&\; \beta$. In this case the circuits P do not invert the signs of the arguments, i.e., $Oa = Od = 1$.

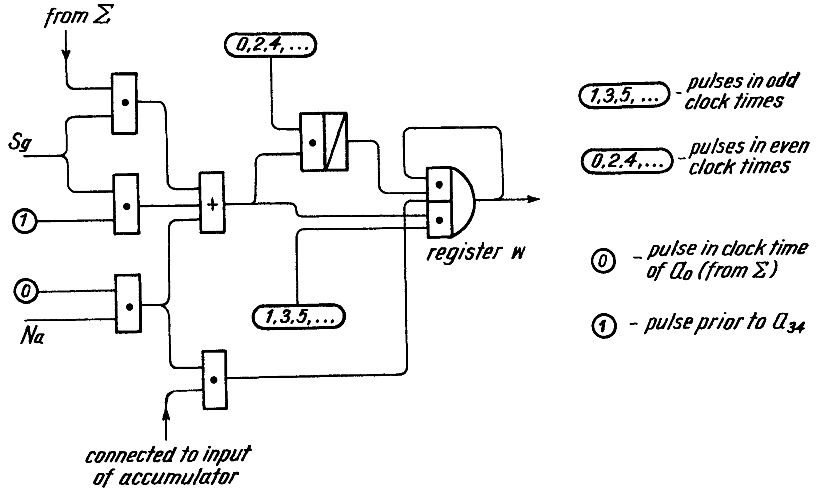
from Σ
Sg
1
0
Na
0,2,4,...
1,3,5,...
register w
connected to input of accumulator
1,3,5,... - pulses in odd clock times
0,2,4,... - pulses in even clock times
0 - pulse in clock time of a_0 (from Σ)
1 - pulse prior to a_{34}

FIG. 11.—The Sign and Overflow Register.

10. The Sign and Overflow Register. When position *Sg* in the "*s*" register is "1," each number coming out from the adding unit is advanced to the sign defining unit. If the examined number is positive or equal to zero the register is left in state "1"; if not, the register is cleared. For this purpose the clock pulse (marked ① in Fig. 11) turns on the trigger. Bits with value "1" in even expansion orders of the number switch off the trigger (which was previously turned on by the clock pulse ①), and in odd orders switch it on. Thus the state of the register is determined by the "one" bit of the highest expansion order (including a_0).

The overflow of a number in the accumulator or of a number forwarded there is examined in a similar way. The trigger is turned on if bit a_0 (the ⓪ pulse in Fig. 11) is equal to "1," and is switched off in the opposite case.

The contents of the sign and overflow register are not changed until a new instruction arrives.

11. Multiplication. Multiplication consists of the preparatory steps, proper multiplication, and final steps. In the last clock time of every step new positions defining the next step are set in the "*s*" register with a simple matrix. This matrix is controlled by the special positions of the "*s*" register and by the coincidence of suitable addresses taken from the drum track.

One of the arguments of multiplication (the multiplier) is placed in the accumulator, the other (the multiplicand) in the "*r*" register; when squaring the argument is placed simultaneously in "*r*" and "*a*."

Counting the multiplication steps begins with address zero of the address track, and the contents of the counter register "*l*" are transferred to this location (track zero, position zero).

Simultaneously the multiplier is placed in register "*l*" and the accumulator is cleared. Then proper multiplication follows: the contents of "*l*" are shifted right, the outgoing "1" bit of the multiplier opens the path for the multiplicand (with inverted sign) to the adding unit, the contents of "*a*" are shifted right, the outgoing bit is placed in the emptied position of the register "*l*." Coincidence of the constant address with the address track ends this procedure.

After completing multiplication the contents of register "*l*" (the least significant part of the product) are written in the storage cell on track zero and in actual position under the writing head. Finally, after the waiting time is completed the contents of the zero address are brought again to register "*l*." Consequently, the most significant part of the product is to be found in the accumulator, and the least significant part in memory cell -7.

The speed of multiplication is 16–36 mult./sec., depending on the access time.

This scheme makes it possible to perform many operations with a single instruction, e.g., $x \cdot x$, $x \cdot a$, $a \cdot a$, $-a \cdot a$, $4a^2$, $\frac{a^2}{4}$, $(x + a) \cdot a$, $(x - a) \cdot a$, $(x + a)^2$ etc.

12. Acknowledgments. The computer described here was built in the Department of Construction for Telecommunications and Broadcasting of the Warsaw Technical University directed by Prof. A. Kiliński.

The machine was built under the direction of A. Łazarkiewicz, the organization was designed by Dr. Z. Pawlak, the arithmetic and the logical circuits were prepared by W. Balasiński.

Politechnika Warszawska, Z.K.T.R.
Warsaw, Al. Jerozolimskie 69
Poland

1. W. S. Elliott, C. E. Owen, C. H. Devonald & B. G. Maudsley, "The design philosophy of Pegasus, a quantity-production computer," *Proc. Inst. Elec. Engrs. B*, Vol. 103, Supplement No. 2, October 1956, p. 188.

2. I. W. Merry & B. G. Maudsley, "The magnetic-drum store of the computer Pegasus," *Proc. Inst. Elec. Engrs. B*, Vol. 103, Supplement No. 2, Octoper, 1956, p. 197.

3. Z. Pawlak & A. Wakulicz, "Use of expansions with a negative basis in the arithmetic element of a digital computer," *Bull. Acad. Polon. Sci., Cl. III*, Vol. V, 1957.

4. W. L. Van der Poel, "The logical principles of some simple computers," University of Amsterdam Thesis, The Hague, Netherlands, 1956.

5. Z. Pawlak, "An electronic digital computer based on the "-2" system," *Bull. Acad. Polon Sci., Cl. III*, Vol. VII, No. 12, 1959.

6. W. Balasiński, "A mode of performing four basic arithmetic operations of the first grade in electronic and other digital devices," Polish Patent Nr. 42954, March 1960.