

July 31, 1962

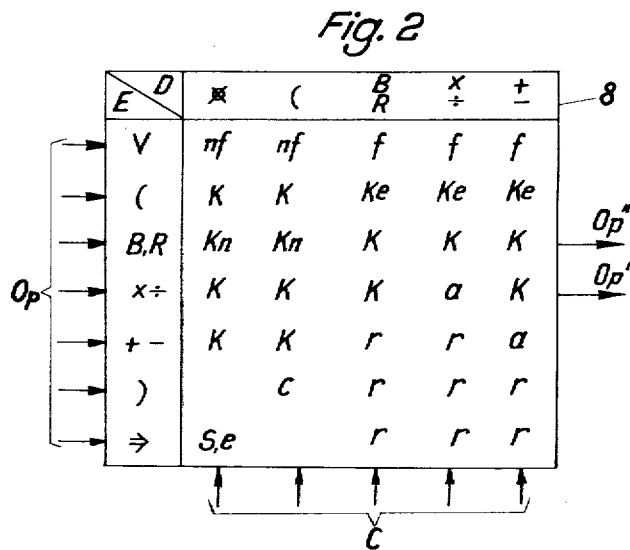
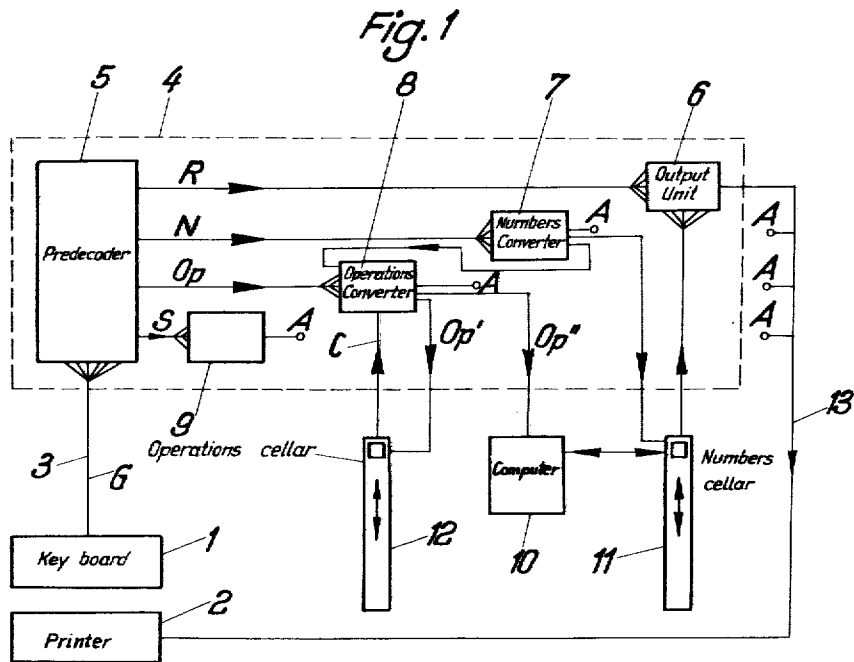
F. L. BAUER ET AL

3,047,228

AUTOMATIC COMPUTING MACHINES AND METHOD OF OPERATION

Filed March 28, 1958

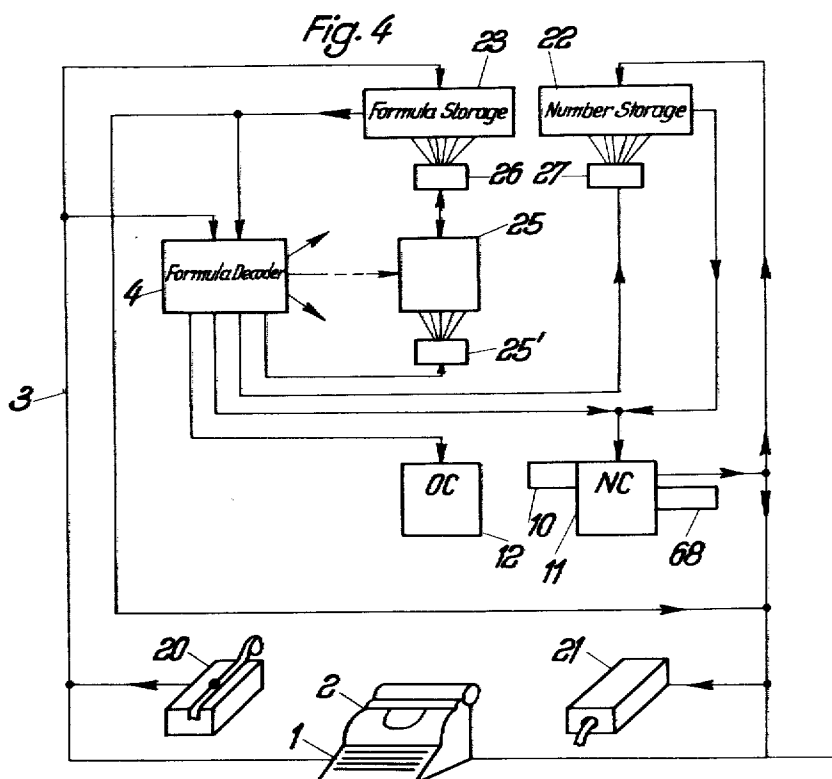
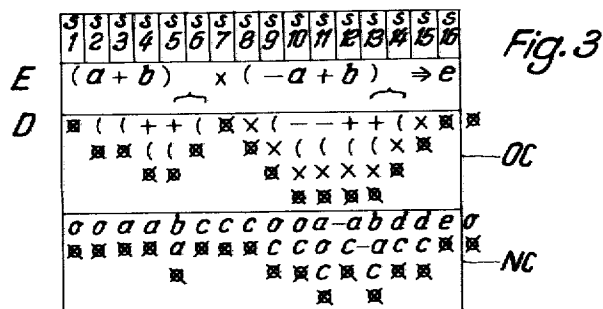
14 Sheets-Sheet 1



Inventors
Friedrich Ludwig Bauer
Klaus Sornheim
Symmetrisch-Technik
Attorneys

3,047,228

14 Sheets-Sheet 2



Inventors
Friedrich Ludwig Bauer
Klaus Samelson
Symantec's Technics
Attorneys

July 31, 1962

F. L. BAUER ET AL

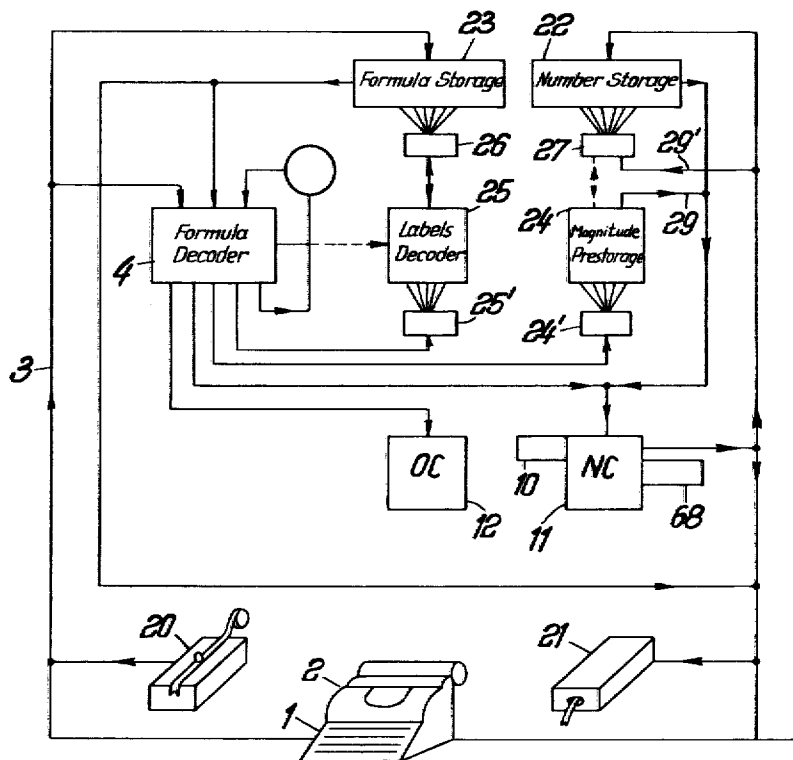
3,047,228

AUTOMATIC COMPUTING MACHINES AND METHOD OF OPERATION

Filed March 28, 1958

14 Sheets-Sheet 3

Fig. 4a



Inventors
Friedrich Ludwig Bauer
Klaus Samelson
Symmetrisch & Seckner
Attorneys

July 31, 1962

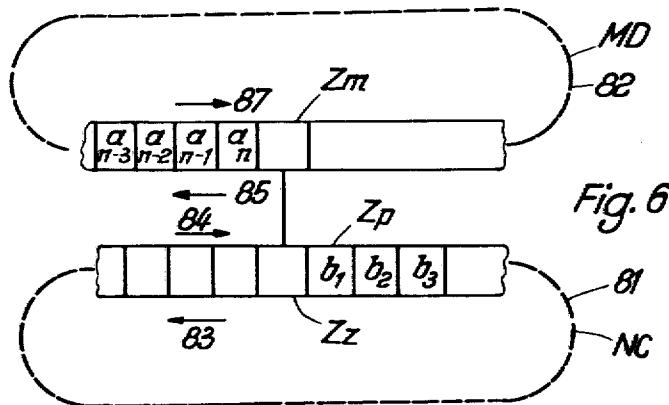
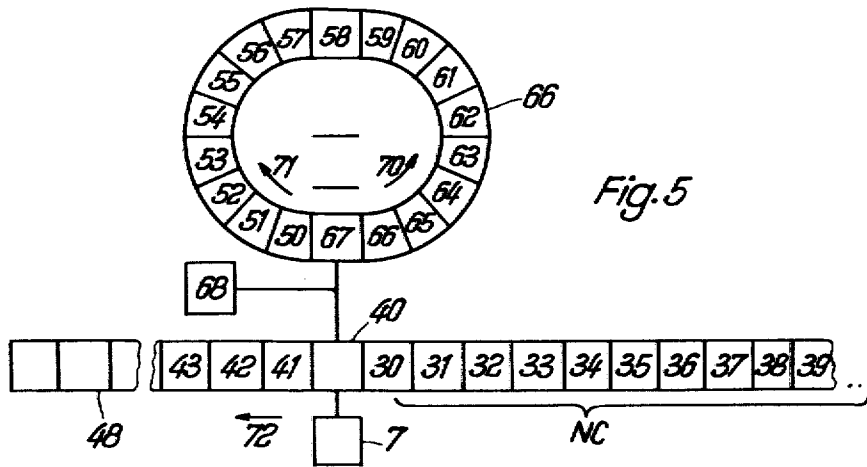
F. L. BAUER ET AL

3,047,228

AUTOMATIC COMPUTING MACHINES AND METHOD OF OPERATION

Filed March 28, 1958

14 Sheets-Sheet 4



Inventors
Friedrich Ludwig Bauer
Klaus Samelson
Symmetrisch & Partner
Attorneys

July 31, 1962

F. L. BAUER ET AL

3,047,228

AUTOMATIC COMPUTING MACHINES AND METHOD OF OPERATION

Filed March 28, 1958

14 Sheets-Sheet 5

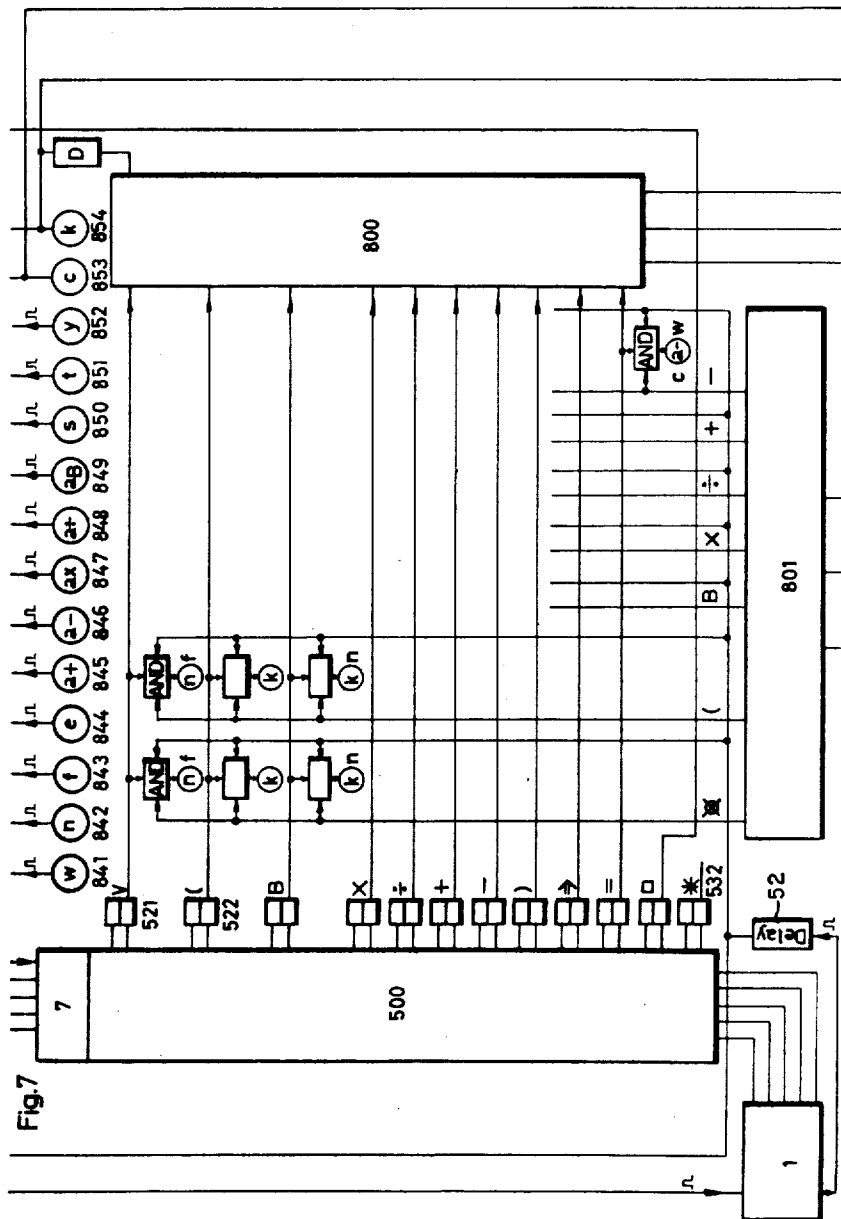


Fig. 7

Inventors
 Friedrich Ludwig Bauer
 Olaus Samelson
 Symmetrisch & Sechner
 Attorneys

July 31, 1962

F. L. BAUER ET AL

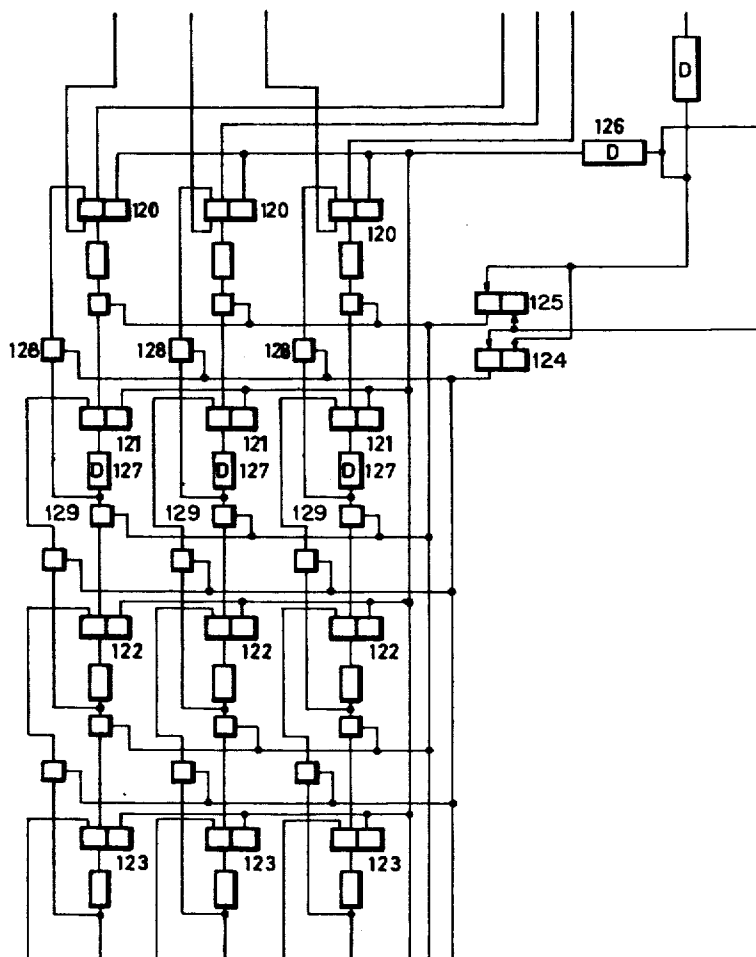
3,047,228

AUTOMATIC COMPUTING MACHINES AND METHOD OF OPERATION

Filed March 28, 1958

14 Sheets-Sheet 6

Fig.8



Inventors
Friedrich Ludwig Bauer
Klaus Samel
Symmetrisch & Lechner
Attorneys

July 31, 1962

F. L. BAUER ET AL

3,047,228

AUTOMATIC COMPUTING MACHINES AND METHOD OF OPERATION

Filed March 28, 1958

14 Sheets-Sheet 7

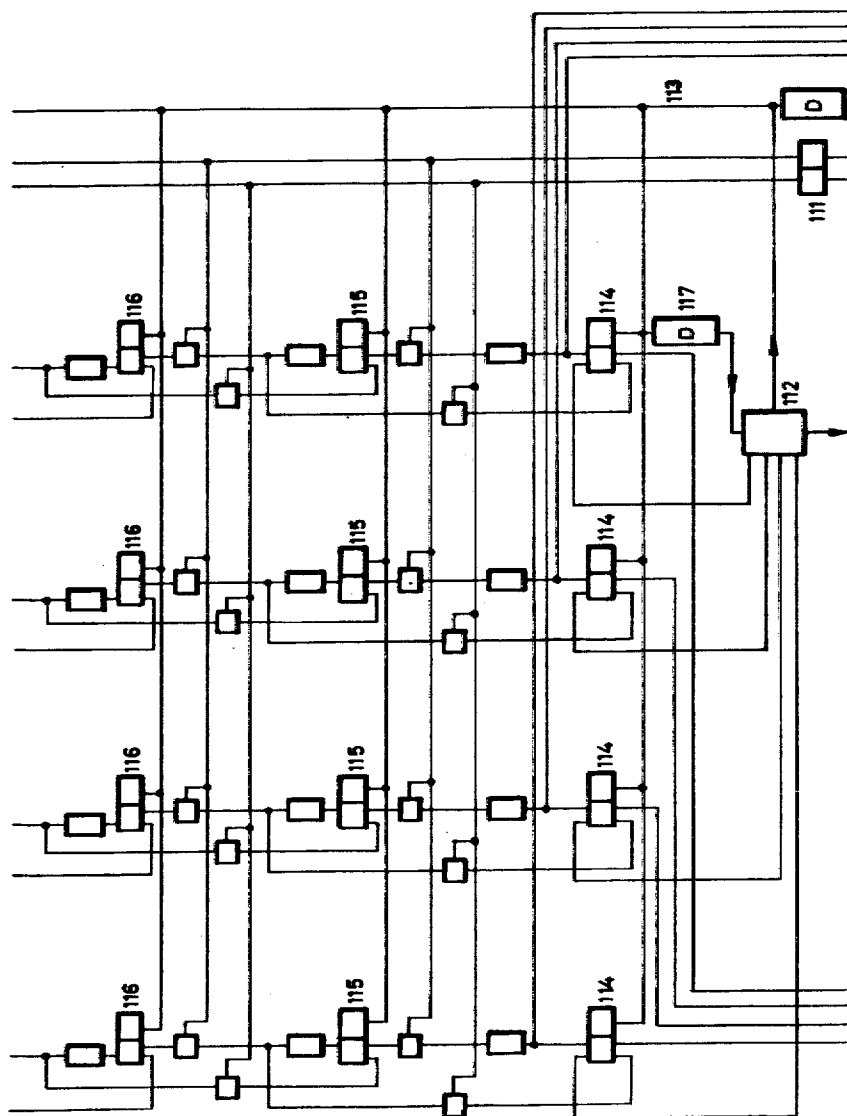


Fig. 9

Inventors
Frederick Ludwig Bauer
Alfred S. Sauer
Synthesist & Technics
Attorneys

July 31, 1962

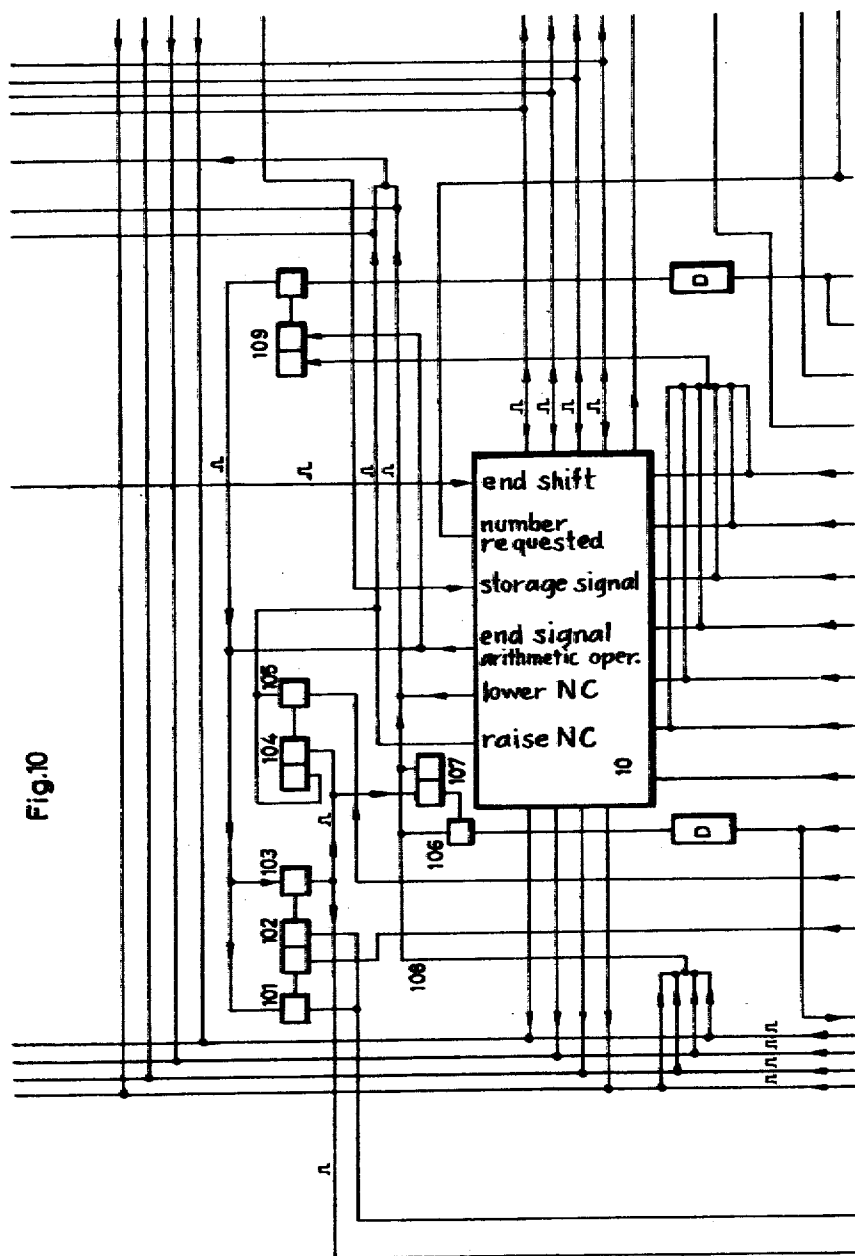
F. L. BAUER ET AL

3,047,228

AUTOMATIC COMPUTING MACHINES AND METHOD OF OPERATION

Filed March 28, 1958

14 Sheets-Sheet 8



Inventors
Frederick Leising Bauer
Hans Samuelson
Symmesett & Schuler
Attorneys

July 31, 1962

F. L. BAUER ET AL

3,047,228

AUTOMATIC COMPUTING MACHINES AND METHOD OF OPERATION

Filed March 28, 1958

14 Sheets-Sheet 9

Fig.11

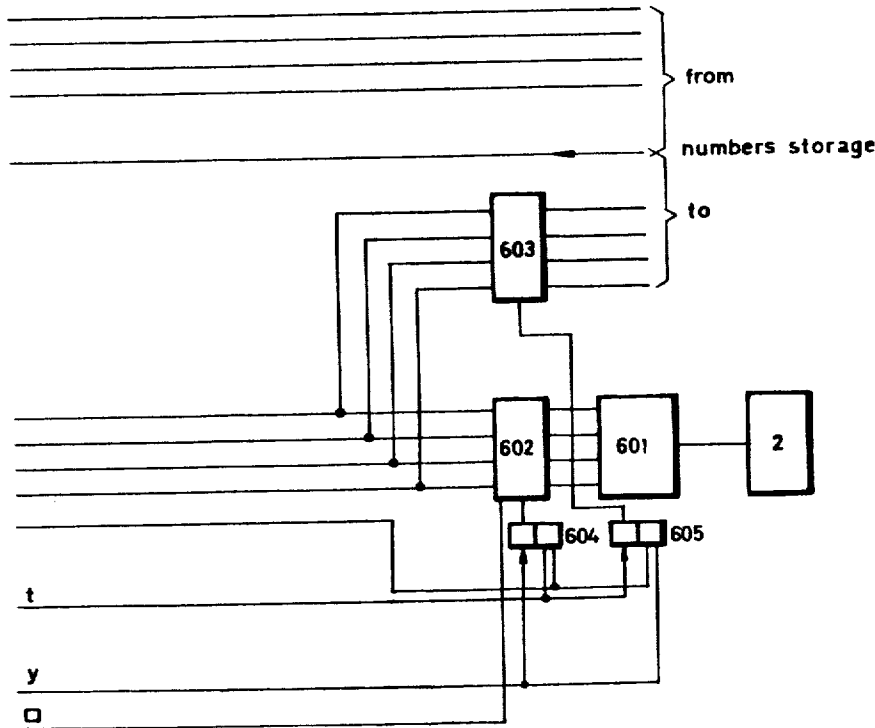


Fig.12

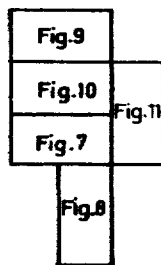
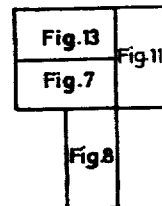


Fig.12a



Inventors
Friedrich Ludwig Bauer
Klaus Samelson
Symmetrecht & Lehner
 Attorneys

July 31, 1962

F. L. BAUER ET AL

3,047,228

AUTOMATIC COMPUTING MACHINES AND METHOD OF OPERATION

Filed March 28, 1958

14 Sheets-Sheet 10

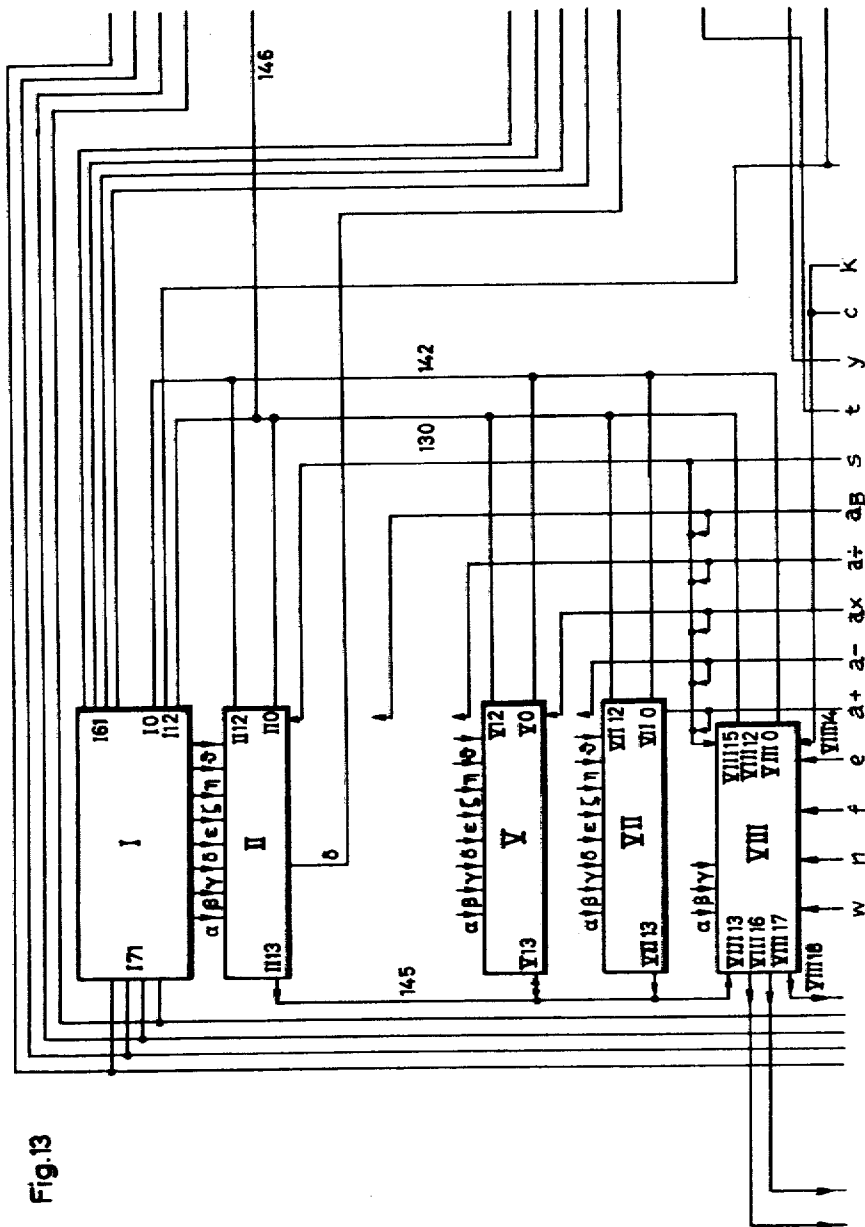
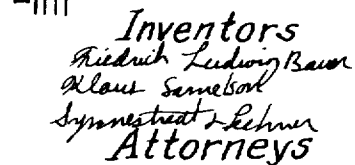


Fig. 13

Inventors
Frederick Ludwig Bauer
Alene Samelson
 Symmes & Sechner
 Attorneys

3,047,228

14 Sheets-Sheet 11



July 31, 1962

F. L. BAUER ET AL

3,047,228

AUTOMATIC COMPUTING MACHINES AND METHOD OF OPERATION

Filed March 28, 1958

14 Sheets-Sheet 12

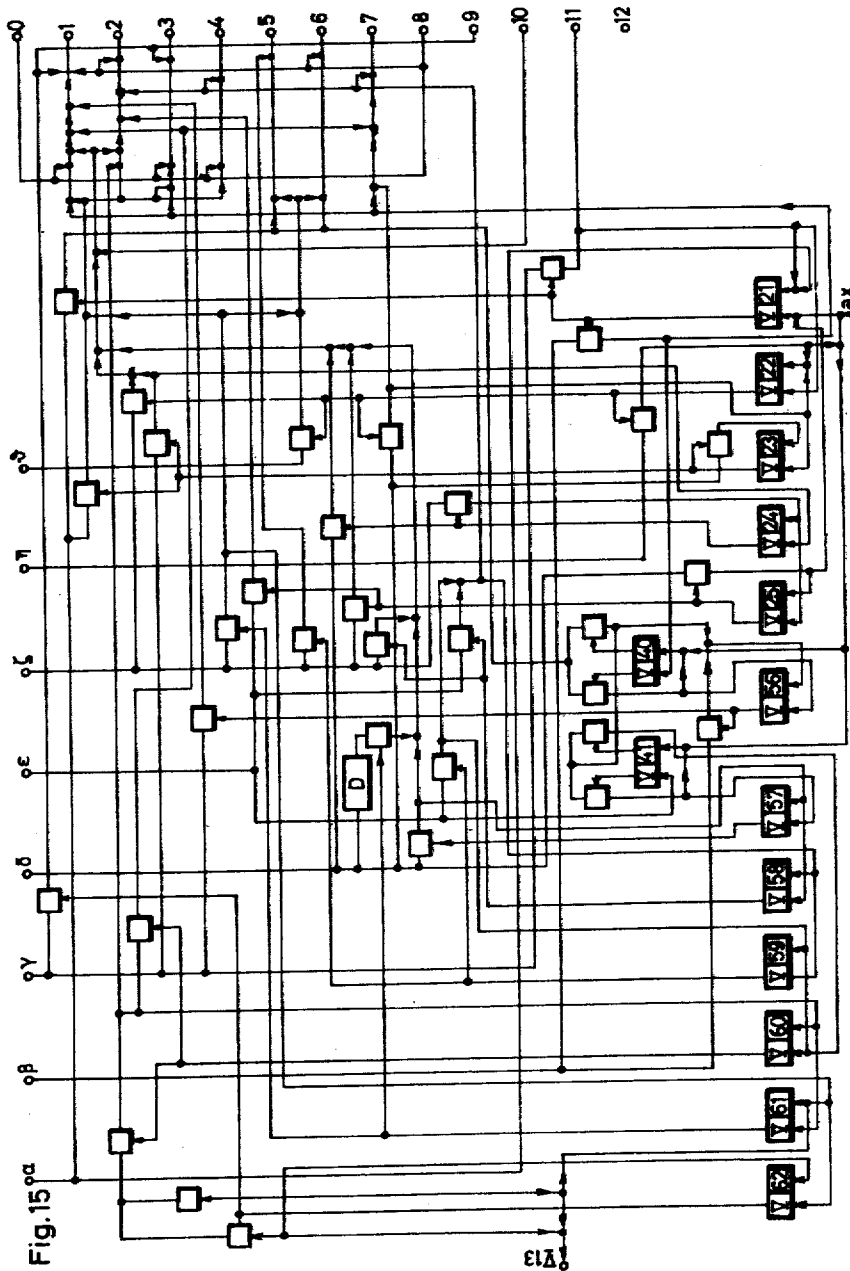


Fig. 15

Inventors
Friedrich Ludwig Bauer
Klaus Samelson
Sylvester A. Lechner
Attorneys

July 31, 1962

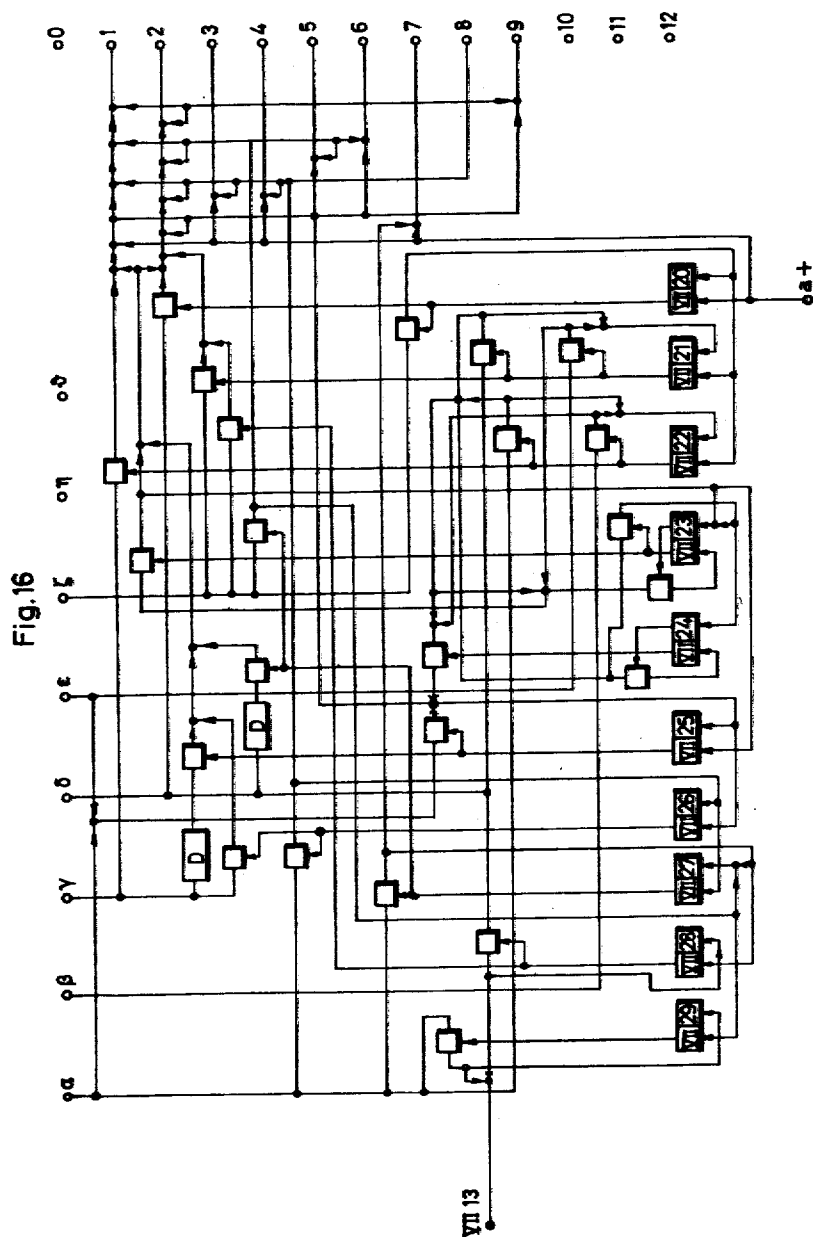
F. L. BAUER ET AL

3,047,228

AUTOMATIC COMPUTING MACHINES AND METHOD OF OPERATION

Filed March 28, 1958

14 Sheets-Sheet 13



Inventors
Friedrich Ludwig Bauer
Klaus Sampl
Symmetrisch & Kechner
Attorneys

July 31, 1962

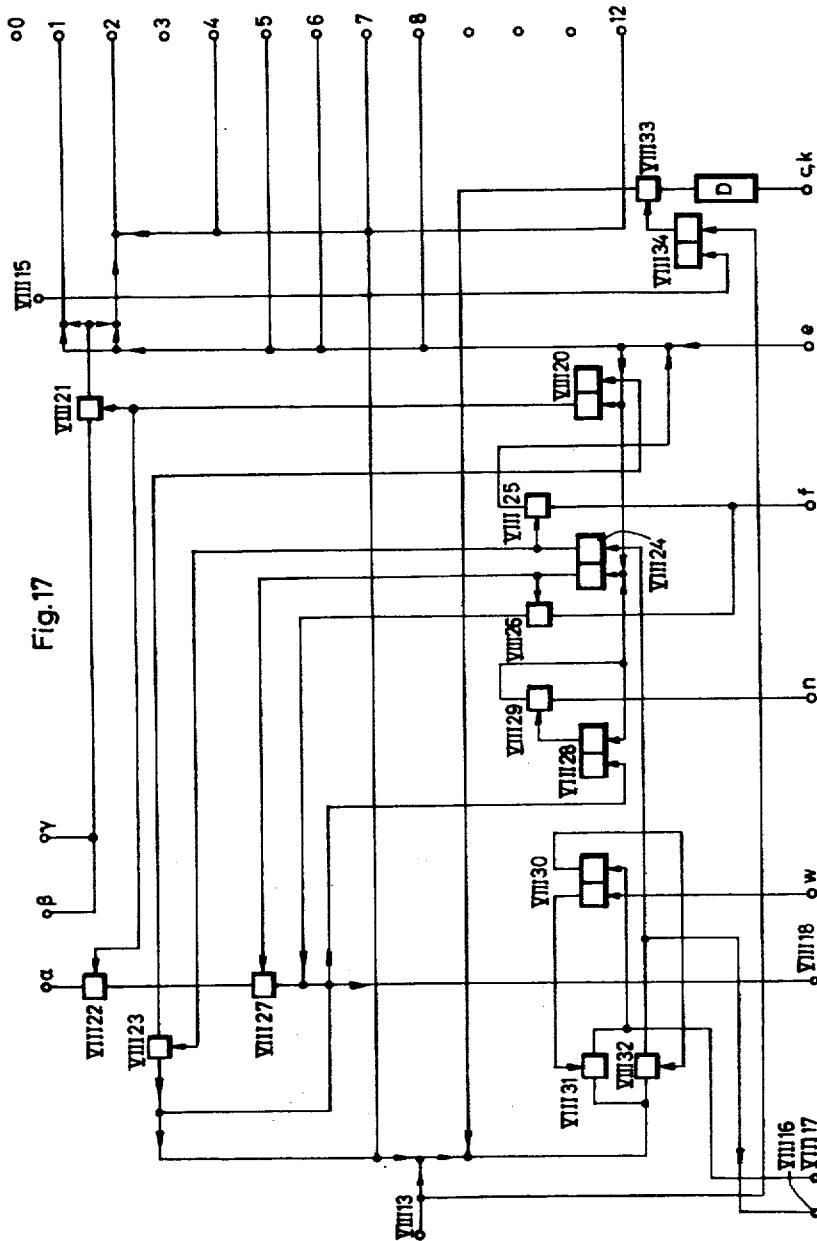
F. L. BAUER ET AL

3,047,228

AUTOMATIC COMPUTING MACHINES AND METHOD OF OPERATION

Filed March 28, 1958

14 Sheets-Sheet 14



Inventors
Friedrich Ludwig Bauer
Klaus Sammler
Symonstett & Hechner
Attorneys

1

3,047,228

AUTOMATIC COMPUTING MACHINES AND METHOD OF OPERATION

Friedrich Ludwig Bauer, 40 Portschacherstrasse, and
Klaus Samelson, 19 Hiltenspergerstrasse, both of
Munich, Germany

Filed Mar. 28, 1958, Ser. No. 724,770
14 Claims. (Cl. 235—157)

This invention relates to automatic, mechanical, electronic or electric computing machines and is particularly concerned with the construction and logical design of computing machines as well as the input and output devices associated therewith.

Known automatic computing devices and data processing systems require instructions as to the manner and order in which numerical or other information processing steps are to be carried out. The pattern of writing out such instructions for each such system, is chosen early in the development of the system in such a way as to describe elementary technological functions of the system. The instructions thus written are usually called a "program." The program for a process of calculation, and the mathematical formulas usually used by the mathematicians for describing this process, each characterize one and the same operation, although in two basically different languages.

The translation from the language of mathematical formulas into the program is usually called programming. This translation is in practice a time consuming and generally undesirable procedure, subject to errors. The language of programming is an unusual one for the mathematician and, furthermore, is different for different machines. This differentiation of programming technique from machine to machine indicates the extent to which the programming systems of hitherto used machines depend upon the constructional features of the particular machine and how little the mathematical expressions used uniformly throughout the world have been considered by the designers of computing machines.

The drawbacks of the usual programming procedure have been clearly recognized in technical literature for several years. The easiest way to overcome these drawbacks has been to utilize existing computing machines of universal design for certain routine processes of programming, a task which itself represents a data processing problem. There exist today programs which, with certain limitations, effect the whole translation work from the mathematical formula to the program of the computing machine.

These translating programs are very complicated and consequently very voluminous. Small computing devices are not capable of solving such problems. The translation of voluminous formulas requires excessively long times even in medium size systems.

The present invention is concerned with an automatic computing machine which is directly controlled by mathematical formulas written out in conventional style. It represents important advances in the art, both in its design and in its capabilities, as compared with machines employing the usual programming methods. According to the invention, instructions are offered to the control device of the machine in a form of writing essentially identical with the notation of mathematics. Each character of the formula is examined for its meaning and is either immediately operated on or postponed for later recall, preferably in the sequence of its arrival, only until all characters of the formula required for its evaluation have been offered to the control device. A further postponement may, of course, be effected, but is not essential and can only be justified by other advantages lying outside the invention.

2

The characters of the formula representing numerical symbols, i.e. numbers, are separated from the characters representing operational symbols, and, if they must be postponed, both are fed to storage devices, preferably to two different "cellars," namely the numbers cellar and the operations cellar, and are accessible to the control device from these storage devices.

It is preferable to place the characters arriving in the numbers cellar or the operations cellar respectively at the top of the corresponding sequence and to withdraw the characters automatically from the top of the corresponding sequence.

The operational symbols of arithmetic $+$ $-$ $()$ are used, preferably in form of code symbols, for initiating control operations, i.e. machine functions. The time of the final start of the control operation by a code sign is determined, among other things, by the fact that one or more succeeding code signs necessary to the operation have arrived. For this reason the code signs are postponed or stored in the operations cellar, and are withdrawn from this operations cellar only when the time of carrying out the operation is reached, the order of presenting said characters being the correct one automatically in case of the above-mentioned sequential ordering in the cellar. In a similar manner, the numbers cellar guarantees that the operations are automatically carried out with the respective numbers as soon as all numerical values required for carrying out the operation are present. The numbers which are to be used in a certain calculation may be represented by digital symbols, for instance in the decimal system, and corresponding code signs.

Those characters of the formula requiring a result, particularly the equality sign, are fed into a special device, namely into the output control.

The method described above in its basic terms is described later on in an embodiment designated as type I, allowing the processing of the most simple formulas.

Another embodiment for carrying out the method of the invention is described in the following paragraphs and designated as type II.

In many cases it is desirable to compute only once certain partial results which are to be used repeatedly, and to designate such results by special symbols, for instance by letters, in writing down the formula. In accordance with the further invention, special characters are used as algebraic symbols of quantities, and code signs associated therewith are used for expressing numbers or groups of numbers, for instance results and partial results in formulas, in such a manner that each character entering the control device for the first time produces a reservation of certain places for a number or a group of numbers within a storage device. A coordination between this place or these places and the respective symbol of the quantity is maintained until further notice. The quantity symbol is used by the control device in formulas as a substitute for the number or group of numbers stored in the corresponding place or places.

The entering of a number into a place designated by the symbol of a quantity in the numerical storage is produced by the result symbol $=>$, followed by the quantity symbol. It may be of advantage to carry out the above-mentioned reservation of places only with the just mentioned entering of numbers. The operation is postponed if a quantity symbol within a formula arrives in the control device which has not yet been covered by a number in the numbers storage. New information is asked for from outside the machine until the place to be reserved has been occupied by a number.

By using letters as characters it becomes possible to designate the initial values of a calculation from the beginning.

ning by letters, so that the mathematical formulas can be written entirely or partly with algebraic symbols.

Further improvements of the computing device are possible by means described in the following paragraphs as type III. While the above-mentioned embodiments provide a computing device using a direct control by formulas and writing out the entire operation, i.e. while the input and output has the form of mathematically written formulas, it is often desirable to utilize the possibility of repeating formulas. For this purpose the entire incoming information, serving as before as direct formula control, is also stored at the same time in a formula storage. Special characters are used as characterizing symbols for numbering groups of formulas in order to open up further possibilities. Each characterizing symbol arriving for the first time at the control device establishes the coordination between the place occupied by the beginning of the group of formulas in a formula storage and the characterizing symbol, which coordination is maintained until further notice.

In particular it is possible to label the beginning of groups of formulas; the labeling symbol may consist of numbers with the addition of a special sign which identifies the symbol as an address in the formula storage. An asterisk (*) is used for this purpose. Consecutive numbering is not required. It is only necessary to arrange a prestorage before the formula storage. The prestorage shows under the number of each labeling symbol as input the address in the main storage identified by the respective labeling symbol.

The method may further be developed in such a way that the input of a labeling symbol in combination with a special sign, for instance an arrow ($\rightarrow$), as jump symbol, suffices to effect a repetition of the calculation or, more generally, that the calculation is continued with the beginning of the group of formulas noted under the respective label in the formula storage. The transfer may depend upon conditions in a known manner. The operation is postponed if a jumping symbol leads to a group of formulas not yet noted in the formula storage, so that new information is asked for from the outside. In this case as well as in the above-mentioned case in which a quantity symbol not yet covered by a number in the number storage appears in a formula, the required information is entered only into formula storage and the symbols designating the beginning of individual groups of formulas are evaluated in the manner mentioned above. This procedure is automatically interrupted when the labeling symbol of the called-for group of formulas is received, or when the called-for quantity has been occupied by a number. The calculation is resumed at the point of interruption, and in the last-mentioned case, the jump is made. When this method is carried out, the calculation is interrupted also when the character last noted in the formula storage has been consumed in the process without effecting a jump towards an existing label. The information that the formula storage is empty results in the control device asking for new information from the outside, which request is noted in the formula storage and executed at the same time.

Instead of the hitherto described principle of carrying out all operational steps of formula symbols as soon as possible, it is also possible to postpone the operation either entirely or partly until a suitable later time if this should be required.

A special character may be interpreted as symbol meaning "no storage," noting down so that following this sign the storing of incoming information is suppressed until further notice, for instance by a dissolving symbol or by the next incoming marking symbol for groups of formulas.

The machine embodying the invention comprises, in its most simple form, a predecoder receiving all characters of the formula in the usual sequence of writing the formula. The predecoder has a number of outputs leading to a number converter, to an operations converter, and to

an output control device, as well as to control means for insignificant characters, i.e. characters symbolizing "carriage return" and the like. The converters of the control device may be connected with a printer. The numbers cellar and the numbers converter are connected in such a manner that the numbers cellar may receive numbers in the sequence in which they come in from the converter and that, furthermore, the first number of the sequence is fed as result to the printer when a corresponding instruction is received.

The operations converter is connected with the operations cellar so that it can supply the operation symbols in the sequence of their arrival, whereby either the last-arrived symbol of the previously supplied and upwardly moving symbol or the newly arriving symbol may be supplied to the computing means in order to carry out those operations for which the corresponding operands are present at the top of the sequence stored in the numerical cellar.

It is preferable that the characters closing a formula, particularly the equality sign or the result sign, cause an examination for "sense of formula."

The operations cellar and the numbers cellar may comprise means storing the supplied characters sequentially by moving the stored characters downwardly in the order of their arrival and allowing a withdrawal of the last stored character or of the uppermost of the upwardly moving characters only.

In another embodiment, the operations cellar and/or the numbers cellar may comprise means for storing the supplied characters sequentially, by putting each incoming character on the place before the last arrived character, by marking this place, and by feeding out the character from the place marked last. The computing unit operates with the numbers contained in the uppermost or in the two uppermost "stories" or levels of the numbers cellar, in conformity with the instructions received from the operation control, and feeds the result again into the uppermost story of the numbers cellar.

The output control means is preferably connected with the numbers cellar in such a manner that the connection between the numbers cellar and the printer is completed upon the arrival of an equality sign and an eventually succeeding sign meaning "digit required," and that the number contained in the upperstory of the numbers cellar is supplied entirely or partly to the writing means.

The machine, according to the invention, can also operate with subscript quantities. The calculation with subscripts is carried out in a manner analogous to the calculation with other quantities. For designating subscripted quantities, special characters may be used as subscripting symbols which mark the beginning and the end of subscripts and the separation between individual subscripts; these signs are fed to special means effecting immediately an interruption of the current calculation, the evaluation of the expressions in the subscript positions in accordance with the above-mentioned methods, and the selection of the single components of the respective subscripted quantity designated by the subscript values obtained. An example for carrying out such calculations is described below and designated as type IV.

Further features and advantages of the invention will be pointed out in the following description of embodiments, represented in the drawings, in which:

FIG. 1 is a block diagram of a computing machine according to the invention in its most simple embodiment;

FIG. 2 is a diagram illustrating the operations converter 8 of FIG. 1;

FIG. 3 is a tabular representation of the processing of an illustrative calculation demonstrating the roles of the numbers cellar 11 and operations cellar 12 of FIG. 1;

FIG. 4 is a block diagram of a computing machine, including further components not illustrated in FIG. 1;

FIG. 4a is a block diagram similar to FIG. 4, and in-

cluding components for processing subscripted characters;

FIG. 5 is a schematic representation of the numbers cellar and computer device of the invention;

FIG. 6 is a schematic representation similar to FIG. 5 and illustrating a computing device and numbers cellar having two ring circuits;

FIG. 7 illustrates the predecoder and operations converter of a complete machine in accordance with FIG. 1;

FIG. 8 illustrates the arrangement of the operations cellar of such a complete machine;

FIG. 9 illustrates the arrangement of the numbers cellar of such a machine;

FIG. 10 illustrates the computer device of such a machine;

FIG. 11 illustrates the output elements of such a machine;

FIG. 12 is a key to the assembly of FIGS. 7 through 11;

FIG. 13 is a block diagram of a combined computer device and numbers cellar of the type illustrated in FIGS. 9 and 10, the interconnections of FIG. 13 with FIGS. 7, 8 and 11 being shown in FIG. 12a;

FIG. 14 shows the numbers cellar adding network of FIG. 13;

FIG. 15 shows the multiplication control of FIG. 13;

FIG. 16 shows the addition control circuit of FIG. 13; and

FIG. 17 shows yet another of the control circuits of FIG. 13.

Throughout the machine 13 line cables have been used and in the drawings the individual lines of such cables are identified with the numerals 0 through 12. In some cases, for convenience, only the first and last lines have been drawn to indicate the entire cable. Furthermore, 13 line cables have been illustrated in some instances in which one or more of the 13 lines is not used. In FIGURE 13, for example, the output of arithmetic unit V is indicated as consisting of V-0 and V-12. These collectively indicate a 13 line cable. It happens that, in this particular cable, line 12 is unused and hence no connection is shown to it in FIGURE 15.

FIG. 1 shows schematically the circuit arrangement of a computing machine of simple design according to the invention, representing type I. In this machine expressions of most simple nature, for instance:

$$0.23 \times 4.433 - (2 + 5.28764) =$$

may be computed. The computation is carried out by direct control from the input and the machine answers with the result -6.26575 so that the printer writes down the complete equation

$$0.23 \times 4.443 - (2 + 5.28764) = -6.26575 \quad (1)$$

The machine comprises an electrical typewriter with the keyboard 1 and printer 2. The keyboard 1 may include keys for the following symbols represented by appropriately combined mechanical or electrical signals:

Information N: the ten digits 0, 1, . . . 9; the decimal point.

Information Op: 7 operational symbols + - × ÷ () B; the letter B means "absolute value."

In the same manner functions of a variable may be added, e.g. reciprocal (R), square root (√), sine (sin) etc.

Information R: the equality symbol = and character "digit requested" (will not be printed).

Further writing and operational characters without significance to the computing machine may be provided, for instance letters, line feed, carriage return, space etc. which form the "information S." In the present example at least twenty characters are of significance to the computing machine, requiring at least five elements or bits for coding.

The coded characters are supplied over a cable 3 to a device 4 which may be designated as formula decoder or

central control and which includes the components arranged within the dotted lines. Cable 3 is directly connected to a predecoder 5 having four output terminals. One terminal is connected to the output control 6, another to the numbers converter 7, the third output is connected to an operational converter 8 and the fourth output to a control device 9 for insignificant characters.

The predecoder 5 separates the information G arriving over the cable 3 into the four groups, and supplies them to the respective devices, the information N to the numbers converter 7, the information Op to the operations converter 8, the information R to the output control 6, and the information S to the control device 9.

The machine includes furthermore a computing unit 10 which may be designed in a manner known per se and is able to carry out additions, subtractions, multiplications and divisions. The machine includes furthermore a numbers cellar 11 and an operations cellar 12. The input of the numbers cellar is connected to the numbers converter 7 and to the computing unit 10. The output of the numbers cellar may be selectively connected to the computing unit and the output control 6. The input of the operations cellar as well as its output are connected with the operations converter while the operations converter controls the computer in a manner described below.

The devices 6, 7, 8 and 9 are all connected over outputs A and a line 13 to the typing elements 2.

FIG. 2 shows the operations converter 8 more in detail in form of a matrix. The cables Op coming from the predecoder 5 are connected to the horizontal lines of the matrix. The cables C coming from the operations cellar 12 are connected to the individual vertical columns of the matrix. The output cables Op' and Op'' on the right-hand side of the matrix are connected to the operations cellar or the computing unit respectively.

The first vertical column of FIG. 2 indicates the symbols arriving on the cable from the predecoder. The uppermost line shows the symbols arriving from the uppermost story of the operations cellar. If now, for example, a plus sign + arrives from the predecoder, at the same time when a sign "(" is present in the operations cellar, the operation k is initiated, i.e. the symbol coming from the predecoder is introduced into the operations cellar and stored therein. The letters indicated in the matrix are abbreviations for the following components or operations of the machine:

Table 1

D:	discriminating register (uppermost story of the operations cellar);
E:	decoding register (output of the predecoder);
N.C.:	numbers cellar;
O.C.:	operations cellar;
V:	signal "beginning of number" from numbers converter;
B:	absolute value;
R:	reciprocal;
⊗:	empty condition of the operations cellar;
f:	number into N.C.;
e:	O into N.C.;
k:	E into O.C.;
a:	execute D, E into D;
n:	raise N.C.;
r:	execute D, raise O.C. and repeat;
c:	raise O.C.;
s:	"write" (or later "store").

"Raise" means that the element at the top of O.C. or N.C. is discarded.

"Execute" means that the computer carries out the operation indicated by the content of D with the two uppermost or with the uppermost number respectively present in the numbers cellar. "Repeat" means that the operation of FIG. 2 is repeated with the new content of D.

$E \rightarrow D$ can also be carried out as lower O.C., $E \rightarrow O.C.$; $\rightarrow O.C. \rightarrow N.C.$ means "to place at the top of O.C. or N.C. respectively." The matrix arrangement may be replaced by a triangular array or another logical circuit arrangement of equivalent design as is well known in the art.

In the operation of the device the coded characters, for instance in form of groups of pulses, are supplied from the keyboard 1 to the predecoder 5 and are examined therein for their meaning. If they are numbers, the information contained therein is supplied to the numbers converter 7. The point may be changed into an exponent. The signal V may be directed to the operations converter. If the signal is an operational symbol, it is supplied to the operations converter.

In order to explain the operation more in detail, reference is made to FIG. 3 of the drawing, representing in a diagram the operation of the numbers cellar and the operations cellar.

The uppermost line shows a formula

$$(a+b) \times (-a+b) = \quad (2)$$

the letters a and b representing numbers. The block O.C. below the line represents an operational cellar; the vertical columns S1 to S16 contain the marking symbols which are received during the individual steps of operation. The block N.C. below the block O.C. contains the numbers stored in the numbers cellar. The sign O means that no digits are contained in the numbers cellar and the sign $\otimes$ means that the operations cellar or the numbers cellar respectively are empty in the downward direction. The first column of the operations cellar is empty, and the numbers cellar contains the number 0. If now the left parenthesis (is written on the keyboard, this character is fed into the predecoder. The predecoder recognizes that the character is an operational symbol and directs it to the operations converter 8. The operations converter compares the information Op with an information Op' which may be present in the operations cellar and feeds Op to the operations cellar as represented in the column S2 of FIG. 3.

The number a is now written on the keyboard 1 and arrives in the predecoder. The predecoder recognizes that this is a number and produces a signal V, whereupon the number is fed through the numbers converter 7 to the numbers cellar 11. This step is represented in the third column S3 of FIG. 3. As no further numbers are present, the computer 10 is not yet actuated.

In the next step of operation the keyboard writes the sign +. This character is supplied by way of the predecoder and the cable Op , as well as the operations converter, into the operations cellar 12, as shown in column S4 of FIG. 3.

In the next step of operation the number b is supplied by the keyboard to the predecoder. As this is a number, it is directed to the numbers cellar 11, in which the number a is advanced downwardly. If now the next character consists of a right parenthesis), this character is directed to the operations converter which makes a comparison with the information Op' from the operations cellar. In accordance with the result of this comparison the information Op'' (that is the character "+") is supplied to the computer; the computer withdraws the two numbers b and a from the numbers cellar, adds them and supplies the intermediate result c to the numbers cellar. The characters + and (in the operations cellar are erased as may be seen from columns S6 and S7 in FIG. 3.

The next step consists in delivering the multiplication sign $\times$ from the keyboard to the operations converter which enters it in the operations cellar. Now follows the character (. This character is fed into the operations cellar but produces a number 0 in the numbers cellar as shown in column S9 of FIG. 3, so that the quantity of the parenthetical expression may be computed. If, now, the minus sign - follows, this is directed into the

operations cellar containing now three characters. The next step supplies the number a again, which is directed into the numbers cellar (column S11 in FIG. 3). The following characters are processed in a manner similar to the one described in connection with the first parenthetical expression. When the right parenthesis arrives, the sum within the second parenthesis is formed. The multiplication of the intermediate results c and d is carried out when the equality sign arrives. This character as well as the character "number requested" connects the numbers cellar with the typing elements so that the result may be typed.

FIG. 4 shows the complete layout of a computing machine for carrying out operations of type II and type III.

The computing machine comprises again an electrical typewriter with the keyboard 1 and the printer 2, as well as the formula decoder 4 receiving the code signals by way of a cable 3 from the keyboard. A punched tape scanner or magnetic tape scanner 20 is provided for feeding the machine, whereas a punch or magnetic recorder 21 serves for recording the results. The operations cellar 12 and the numbers cellar 11 as well as the computing unit 10 correspond to the parts represented in FIG. 1. The machine of FIG. 4 contains furthermore a numbers storage 22 and a formula storage 23. The numbers storage serves for holding the numbers, whereas the formula storage stores groups of formulas. Both storages may have the form of a magnetic drum storage known per se, or of an array of magnetic cores, or of a storage tube with scanning cathode ray. The formula decoder is connected in a manner similar to FIG. 1 with the operations cellar and the numbers cellar. A further connection exists in the machine of FIG. 4 from the formula decoder 4 to the scanning unit 27 of the numbers storage 22, the places of which are occupied by special empty symbols in the initial condition. The inputs for the scanning unit 27 are preferably letters, e.g. the 26 letters of the alphabet (information B). A special reservation of places is not necessary in type II operations. Further connections exist between the formula decoder 4, the address decoder 25, and the formula storage 23. A scanning unit 25' is provided for the address decoder 25, serving as address storage for the places in formula storage 23 corresponding to the address as input. When a labeling symbol arrives the address corresponding to the label in the formula storage 23 is taken out of the scanning unit 26 and reintroduced into the respective section of the labels decoder 25. Direct connections are provided from the keyboard 1 or the scanner 20 to the formula storage 23 as well as from the formula storage 23 to the formula decoder 4. A further connection from the formula storage 23 to the typing set 2 is explained below.

The system represented in FIG. 4 may be used to evaluate an expression

$$(2.27 + 3.328) \times 64.45 - 2 \div (2.27 + 3.328) = \quad (3)$$

in simplified form by computing the expression $2.27 + 3.328$ only once and by designating it by an algebraic quantity symbol, e.g. the letter z . Furthermore, the above-mentioned "result sign" $=>$ is used as a symbol and code sign initiating the use of the quantity symbol and the holding of the partial results under an address selected as indicated below. This is written in the following manner:

$$\begin{aligned} 2.27 + 3.328 &=> z \\ z \times 64.45 - 2 &\div z = \end{aligned}$$

The machine answers this input by placing the result behind the equality sign. If the partial result is to be written down the expression $z =$ is inserted after the sign $=> z$ whereupon the machine answers with the partial result 5.598. In writing down this result the second line

from the character x up to the equation sign follows. The complete written out statement reads then:

$$\begin{aligned} 2.27+3.328 &=> z \\ z &= 5.598 \\ x \times 64.45 - 2 \div z &= 360.43383 \end{aligned}$$

The results or answers are underlined. They may be printed in colors. The storage of the partial results is effected in the numbers storage 22. This numbers storage contains a multi-place storage position for each letter, the partial result corresponding to the respective letter being stored in these storage positions. By searching for the respective letter the stored partial result can be taken out of the numbers storage and can be fed to the numbers cellar or to the computer or to the printer.

The computing operations of type III, particularly the repetitions of instructions, become possible by using the formula storage 23. The asterisk sign * following a number, e.g. 33*, indicates that the label decoder is to respond to the number 33 by giving the address in the formula storage where the formula following the label 33* begins. The repetition of a formula is effected by the jump symbol $\rightarrow$ and by adding the desired labeling symbol. The repetition may be dependent upon conditions in a manner known per se. This requires also the introduction of code signs with symbols $<$ or $>$ with the usual meaning.

Assume that it is intended, for instance, to compute the square root $\sqrt{a}$ from the positive number a in accordance with the method of Newton by approximation. As a first approximation, the number 1 is used.

$$\begin{aligned} 1 &=> x \\ 33^* \quad x &=> t \\ (t+a \div t) \times 0.5 &=> x \\ B((t-x) \div x) - s &> o \rightarrow 33^* \end{aligned}$$

The formulas following 33* are to be repeated. The arrow sign $\rightarrow$ initiates the operation which is usually called a "jump" because the computation is repeated from the place which has been noted under the input 33 of the prestorage. The s is the limit for the admissible difference; the operation ceases when this limit is reached. The machine operates then with empty formula storage and requires new information.

Special measures are required when, in the evaluation of alternatives, a branch of the fork can be reached only by a forward jump.

Example:

$$\begin{aligned} & \cdot \\ & \cdot \\ & \cdot \\ & \rightarrow 17^* \\ & \cdot \\ & \cdot \\ & 17^* \end{aligned}$$

The labels decoder signalizes in this case "17*" not occupied." The writing down of the incoming code signs is continued undisturbed. The immediate execution is, however, interrupted for the time being with the exception of noting the beginnings marked by the symbol *, which must, of course, be carried out. The operation is resumed at the moment in which the asked-for address 17 in the labels decoder is occupied by the characters 17*; at this point the operation is continued as before.

The equality sign = of type II, which is now noted in the formula storage, effects as above the printing of the intermediate result of the operation. The number of desired digits inclusive of the position of the point may be prescribed by inserting "digit requested" symbols and corresponding code signs. If a "digit requested" symbol is not present the whole number of digits is asked for.

The following modifications may be of advantage: If an equality sign is followed by a $=>$ besides "digit requested" symbols and a point, this character and the

subsequent quantity symbol shall also be printed. The storage connected with the character $=>$ is not influenced thereby.

Example: The formula storage contains

$$\begin{aligned} 5 \quad z \times 64.45 - 2 \div z &= \square \square \square \square => y \\ (\square \text{ means "number requested"}) \end{aligned}$$

The printer writes $=> y$.

The operation "O into N.C." indicated in FIG. 2 under the inputs = for E and $\otimes$ for D must be suppressed if, as in the example, the equality "=" is followed immediately by the character $=>$.

An embodiment of the numbers cellar is represented in FIG. 5. This embodiment has the special feature that the numbers cellar is combined with the computing unit in order to obtain a most simple arrangement.

The numbers cellar N.C. includes a chain of storage elements; FIG. 5 shows the elements 30 through 39. These storage elements are arranged adjacent to an accumulator element 40 receiving the numbers from the outside from the number converter 7. A chain of storage elements is arranged on the other side of the accumulator element 40. This chain is designated as "appendix" and includes the elements 41 through 48. The arrangement includes furthermore a ring register 86 consisting of storage elements, in the present case 17 elements 50 through 66. This number may be larger or smaller and is chosen only for the purpose of representation. The element 67 of the ring is connected to the accumulator element 40 and is used as addend register. A counter 68 is connected to the connecting line between the accumulator element 40 and the element 67.

When the numbers a and b are to be added with this arrangement, the number a is introduced from the number converter 7 by way of the accumulator element 40 into the numbers cellar. This number is then introduced into the ring so that the ones digit occupies the storage element 50, the tens digit occupies the element 51, the hundreds digit occupies the element 52 etc. The number b is then introduced into the numbers cellar in such a manner that the most significant digit occupies the element 30 and the succeeding less significant digits are introduced into the following elements (the numbers cellar may contain further previously introduced numbers in further storage positions; these numbers are shifted towards the right in FIG. 5 when a further digit of a number is introduced.) The next operation is the shifting of the operand b from the numbers cellar towards the left into the appendix until the last digit, i.e. if it is a number without point, the ones digit arrives in the storage element 41.

In order to take care of the point existing in the number a counter 68 is used. This counter is connected to the adding element 67. The counter is counted up when the number a is shifted from the numbers cellar into the ring and counts therefore how many places behind the point are present. When the number b is shifted from the numbers cellar into the appendix the counter is counted down from the place of the point on. Two cases should be distinguished:

- (1) When the counter is set to zero before the final character arrives, i.e. in case the number b has more digits behind the point than the number a , the ring is shifted synchronously up to the final character, i.e. zeros are introduced into the ring and the number a moves further into the ring in the direction of the arrow 71.
- (2) When, however, in the other case the final sign arrives before the counter has been set to zero, i.e. when the number b has less marks or digits behind the point than number a , zeros are introduced into the appendix until the counter reaches zero.

The numbers a and b are now positioned in such a manner that in the following addition the points of the

two numbers meet simultaneously in the elements 40 and 67 respectively. In carrying out the addition, the numbers present in the element 67 or in the accumulator 40 respectively are added and the result is introduced into the numbers cellar. The zeros which have been introduced either into the ring or into the appendix are added with the digits of the number which are taken out of the appendix or of the ring. The addition is carried out successively for each mark or digit of the number and takes into account the carryover in a known manner until the first end character arrives. If the end character arrives first in the element 67, the addition continues until the second end character arrives in the accumulator. If the end character arrives first in the accumulator, zeros are added until the end character arrives also in the element 67. Any carry-over which may be present is then noted and the number is shifted one more place into the numbers cellar.

In carrying out a multiplication the arrangement operates as follows: for simplifying the description the multiplier is abbreviated as "mer" and the multiplicand as "mand." It may be assumed that the mer has been introduced into the numbers cellar before the mand. Then the mand is shifted into the ring as in carrying out an addition and the counter is counted up from the point. Zeros are then introduced into the appendix. The next step is to examine the most significant place of the mer and to store it if it differs from zero. Then the mand and the appendix are added as in carrying out an addition. The mand is shifted undestroyed into the ring in the direction of the arrow 70. When the last-place of the mand is reached the final carry-over is effected also in case it is zero. The number present in the ring is then shifted back in the direction of the arrow 71 and the first partial sum in the numbers cellar is shifted synchronously in the direction of the arrow 72 into the appendix until the initial position of the mand and at the same time of the partial sum in the appendix is reached. The next addition of the mand follows and a new partial sum is formed in the numbers cellar, while the mand is shifted back in the ring with simultaneous shifting of the new partial sum into the appendix. This operation is repeated as many times as the first digit of the mer indicates.

The partial sum is then shifted for one place in order to carry out the multiplication with the next decimal mark or digit of the mer until all digits of the mer have been exhausted. The marks or digits of the mer already used up can be discarded step by step. When the end character in the mer arrives, the result is shifted into the numbers cellar.

The counter is counted up during the multiplication process for each digit of the mer until the point of the mer arrives. The counter 68 is then stopped. When the results is shifted into the numbers cellar the counter is counted down for each shift, and the point is set as soon as the counter indicates zero.

Another example of a numbers cellar combined with the computer is represented in FIG. 6. In this case, the numbers cellar N.C. has the form of a ring 81 with n digital elements. The individual elements of the ring consist again of storage cells which are able to shift their contents into the next succeeding storage element in the direction of arrow 84 when a new digit is introduced at the element zz . A smaller ring 82 having m elements with the same shifting properties and a fixed element Zm as point of introduction is provided for carrying out the function of a multiplicand-divisor-register (abbreviated MD register). An adding device as well as a transfer device are provided between the input elements Zz of the numerical cellar and Zm of the MD register. The MD register is at the same time the uppermost story of the numbers cellar and therefore is always occupied by the uppermost operand (addend, multiplicand, divisor).

The operations of addition, subtraction, multiplication and division are explained below, each in an example.

For carrying out an addition or subtraction the following problem is to be solved:

$$a + b = > c$$

In this case a is a number with the marks or digits $a_1, a_2, \dots, a_n$ and b a number with the marks or digits $b_1, b_2, \dots, b_m$. The number a is introduced into the MD register in such a manner that a_n is located at the left of Zm and the remaining digits of the number follow on the left side.

The number b is introduced into the numerical cellar 81 in such a manner that the digit b_1 is located at the right of the input element Zz in element Zp and the remaining digits follow to the right-hand side. At the beginning of the operation a shifting towards the left side in the numbers cellar in the direction of the arrow 83 is initiated simultaneously with a shifting towards the right side in the MD register in the direction of the arrow 87 until the two decimal points are present in the input elements Zz .

The addition $b_1 + a_1 + \text{carry-over} = > c_1$ is now carried out digit by digit into Zz and a simultaneous shifting in the direction of arrows 84 and 87 of NC and MD up to the highest digit a_1 or b_1 . The digits c_1 of the result are introduced beginning with c_0 (highest carry-over) from the numbers cellar 81 into the MD register while simultaneously shifting the two rings towards the left up to the digit c_m . Zeros or empty characters may be introduced simultaneously into the numbers cellar at the left of Zz . A further shifting of the MD register in the direction of the arrow 85 follows in the second of the above-mentioned cases, until the digit $c_n = a_n$ arrives in Zm or has just passed this input point.

The subtraction is carried out in an equivalent manner. For a multiplication, e.g.

$$a \times b = > c$$

the original position is the same as for the addition. The digit present in the cell Zp may be the multiplier. At the left of b_1 the numerical cell is occupied by zeros. The multiplication is carried out by m repetitions of the following cycle: The content a of the MD register is multiplied digit by digit with the mer digit in Zp and the MD register as well as the numbers cellar are shifted simultaneously towards the right. The multiplication is effected by adding MD into the places to the left of Zp as many times as the number present in Zp indicates. Then the MD register and the numerical cellar are shifted towards the left until the MD register is again in its initial position in which a_n is located at the left of the input element Zm . Then the old mer which is now again in the element Zp is erased and the numbers cellar is shifted towards the left for one mark or digit whereby the next mark or digit of the mer is moved into element Zp . This cycle is then repeated until number b has been completely used up. The position of the decimal point is determined by counting the places behind the point in both factors.

The product c is found after carrying out this operation at first without point in the numbers cellar at the left of the input element Zz . The product c is transferred into the MD register by shifting the numbers cellar and the MD register towards the right and introducing the point. Subsequently, the two rings are shifted towards the left until the last place of the result c is located at the left of the input element Zm and the first place of the number following b at the right of the element Zp .

A division is carried out by repeated subtraction of the divisor from the dividend and by subsequent single back addition of the divisor for setting back the remainder. It is also possible to effect subtraction and addition alternatively in the manner known as non-restor-

ing division. Subsequently the current remainder of the division and the partial quotient are shifted into the numbers cellar towards the left until these numbers stand at the left of Zz ; the current digit of the quotient, is then introduced at the input element Zz .

The introduction of the point into the quotient is effected when the points in the MD register and the numbers cellar coincide.

If the divisor has j significant digits more to the left of the point than the dividend, it is necessary to shift the dividend for j places towards the left before beginning the division, to eliminate the point and to begin the quotient with 0.0 . . . 0 so that the number of zeros behind the point is equal to $j-1$.

Instead of shifting the numbers cellar and MD relatively to the fixed elements Zz or Zm , it is also possible to move the input elements along the register. This means that the numbers cellar (and eventually the MD register) is designed as a storage which can be tapped at each element but which is not shiftable. The tapping is effected by a scanning device, the input of which may be counted up and downwardly. The shifting to the left has then to be replaced by counting up and shifting to the right by counting down the scanning device. The separation between the numbers cellar and the MD register may then be dispensed with and it is possible to treat the MD register directly as the first story of N.C. or to begin with N.C. and MD at both ends of one and the same continuously numbered storage.

After this description of several embodiments of the numbers cellar and the computing unit and of a combination of these two devices, reference is now made to the method of operation identified in the introduction as type IV, which method opens up new possibilities. Reference is made now to FIG. 4A.

For many purposes the formula storage described up to now is fully sufficient. There are, however, problems which require equivalent operations for different numbers so that they must be written down repeatedly, for instance

$$a+b+c+d+e=>z \quad (4)$$

The equivalence of the adding process becomes apparent when the above-mentioned formula is written by introducing subscripts and the intermediate sums, i.e., when a_1 is written instead of a , a_2 instead of b , etc.

$$\begin{aligned} o &=> h_1 \\ h_1 + a_1 &=> h_2 \\ h_2 + a_2 &=> h_3 \\ h_3 + a_3 &=> h_4 \\ h_4 + a_4 &=> h_5 \\ h_5 + a_5 &=> h_6 \\ h_6 &=> z \end{aligned} \quad (5)$$

The superfluous informations in this lengthened formula can be eliminated when the subscripts are not used as distinguishing marks but rather as counting marks, and if, quantity symbols are introduced for the subscripts, as is usual in mathematics. The Equations 4 and 5 are then written:

$$\begin{aligned} o &=> h_1 \\ 1 &=> i \\ 27^* \quad h_1 + a_1 &=> h_1 + 1 \\ i + 1 &=> i \\ i - 5 \leq o &\rightarrow 27^* \\ h_6 &=> z \end{aligned} \quad (6)$$

If the knowledge of the intermediate sum h_1 is not important, all intermediate sums may be designated with the sum symbol z . A change of meaning, however, takes place at the moment in which the result sign appears in the third line

$$\begin{aligned} o &=> z \\ 1 &=> i \end{aligned} \quad (7)$$

$$\begin{aligned} 27^* \quad z + a_1 &=> z \\ i + 1 &=> i \\ i - 5 \leq o &\rightarrow 27^* \end{aligned}$$

After carrying out this loop, z has assumed the meaning h_6 , so that the last line can be eliminated.

That above-mentioned formulas show that calculations are carried upon the subscript characters in the same manner as upon other characters. It must, however, be taken into consideration that the subscripts are often functions of subscript quantities, as for instance above h_{i+1} (the subscript is $i+1$, i.e. a function of the subscript quantity i) or $P_{i+1;1-1}$.

The introduction of subscripts requires the introduction of two symbols $\downarrow \uparrow$ or code signs having the meaning of lowering or raising the line of typing and which correspond with functions of the typewriter. Furthermore a separation symbol ";" for the individual subscript positions (and a corresponding code sign) becomes necessary.

Furthermore, in the system of FIG. 4A, the signs $\downarrow$; $\uparrow$ control the numerical result of the expressions in the subscript place. The sign $\downarrow$ has the effect that the current calculating operations have to be interrupted. In the meantime, the operation is carried out for the first subscript position. Each incoming sign ";" or $\uparrow$ represents the end of a subscript position and has therefore the meaning of a result sign for the subscript expression because the result of the subscript operation must be determined in order to enable the selection of the quantity carrying the subscript. The sign $\uparrow$ gives at the same time the final signal for the subscript operation and the starting signal for the quantity selection, which will not be described.

It would be possible but very complicated to operate with a constant store of subscript quantities which might be designated by letters as heretofore, having up to two subscript positions with the values 0 to 99. This would mean a waste of storage places, so that this solution is not suitable.

It is therefore preferable to adapt a storage with a limited number of places to any special calculation, so that its capacity is economically used and only such a number of places is kept available as is required by the actually used subscript combinations. The design of an appropriate storage searching device for such a storage, which has to be adjusted for each calculation, is a purely technological problem which needs no further description.

The same object may be reached in another way when the storage places are not designated by quantities but numbered as is usual in known machines. (If no subscripts appeared in the formulas it would be unnecessary to use the irritating addressing of storage places by numbers in known machines operating with constant length of numbers.) The selection of the place characterized by the numerical value of the subscript within the multitude of places characterized by the same quantity letter can be carried out arithmetically. A magnitude pre-storage 24 should be arranged before the numbers storage 22, operating with letter symbols of the quantities as input, and containing the characterizing quantities for the arithmetical evaluation of subscripts and the reference place numbers connected with the quantities in the numbers storage 22.

As now the storage may additionally be utilized in a convenient manner for numbers of different length, the most simple manner of determining the places is by means of "rectangular storage" as is usual in conventional programming.

The following characterizing quantities must be accessible at the input of the prestorage 24: Maximum length p of numbers, the highest first subscript q , highest second subscript r , etc., and the initial number, a , of the set of quantities of equal letter symbol as reference place number.

In the operation, the expression

$$\downarrow i; k; 1 \uparrow$$

can be replaced by

$$((ixp+k)xq+1)xr+\alpha \quad (8)$$

The same holds good for any other number of subscripts. The calculation to be carried out for operating with the subscripts may be effected by calculating with the above-mentioned formula by means of the components of the computer already described exactly in sequential order, including evaluation of the individual subscript positions.

A comparison between the formula

$$A\downarrow_{i+1;k-1;1}\uparrow \quad (9)$$

with the formula

$$(((i+1)xp+k-i)xq+1)xr+\alpha \quad (10)$$

shows that

the first ;-sign means $)xp+$
the second ;-sign means $)xq+$
the final sign $\uparrow$ means $)xr+\alpha$

This requires that the signs $\downarrow$; and $\uparrow$ be introduced into an auxiliary control device 28, which effects the corresponding above-mentioned operations by means of the above-described devices and control operations (FIG. 2) which are available for evaluating formulas in general. Furthermore, connections 29 and 29' lead from the quantity pre-storage 24 to the numbers cellar and from the numbers cellar to the scanning device 27 respectively. It may be remarked that, in the case of quantities having no subscripts, it is only necessary to look for the place α under the respective input of the pre-storage. (In type III of the system it would be possible to use numbers of different length by means of a scanning storage containing only the addresses of the numbers storage).

The pre-storage is set by the first explicit storage command for a quantity, for instance for an input quantity

$$37.284=>a$$

with the number length 5, or for an intermediate result with the full number of digits of the result. With quantities having subscripts it is necessary to determine by the first storage command the full sequence of all subscripts and the common length of the numbers.

$$0.0000=>a_{6;9}$$

for instance, means that the quantity $a_{i;k}$ has not more than $6 \times 9 = 54$ components, which may each have up to 5 digits. Such a reservation by means of a "dummy" is in general necessary for quantities having subscripts on account of the subscript sequence and also for quantities without subscripts when the number of digits is to be limited.

As such, a pre-selected storage and the storage of input data need not be repeated, it is useful to provide a possibility for suppressing the notation in the formula storage. This requires a symbol—for instance $\diamond$ (and a corresponding code sign). The suppression is to be terminated when the first formula label appears.

Example:

$\diamond 34=>f$ is not entered into the formula storage.
 $66=>q$
 $\longrightarrow 99^*$ is postponed (forward jump)
any formulas $\left\{ \begin{array}{l} (1^*) \text{ stored only (} 1^* \text{ eliminates } \diamond) \\ - \text{ not carried out immediately} \\ - \text{ (on account of forward jump)} \end{array} \right.$
formula group $99^* \left\{ \begin{array}{l} 99^* \text{ is stored, forward jump} \rightarrow 99^* \text{ is carried out.} \\ - \\ - \text{ is stored and carried out} \\ \rightarrow 1^* \end{array} \right.$

When the jump $\rightarrow 1^*$ appears, the calculation proper with the beginning 1^* is carried out.

The main numbers storage is successively occupied by named numbers beginning with the first place. The numbers from the numbers storage having quantity symbols are "named." The "unnamed" numbers are numbers from the numbers cellar appearing in the free places of the numbers storage. A separate register, which may also be arranged in the quantity prestorage 24 (FIG. 4A), holds the address of the first free digit location. This register is designated as register F, "beginning of free storage." The free storage is always available as numbers cellar for unnamed intermediate results of a formula expression. All named numbers appearing in a formula can be introduced into this part of the numbers storage serving as numbers cellar. Then the above-mentioned method can be used for the calculation as in the system of FIG. 6, if the position of the momentary free numbers storage (=top of the sequence in the numbers cellar) is held fixed. On the other hand, the numbers cellar may be replaced by an address cellar, whereby the existing number contents of the numbers cellar are stored in the main numbers cellar 22 and the addresses appearing in the address cellar are substituted for the numbers to be selected by operations f), e), n), FIG. 2. This eliminates a transfer of the named numbers appearing in the formula. Instead thereof only their addresses must be held in the address cellar. Intermediate results are built up in the "free storage"=N.C. and their addresses, marked as N addresses, are introduced into the address cellar in the respective position. Thus, during formula evaluation, "free storage" is occupied consecutively from its beginning. The "momentary free storage" adjacent the last existing intermediate result may be used for building up new N.C. intermediate results.

The scanning unit for the storage is then served by three independent counting registers alternatively. The first two registers are occupied by the uppermost two addresses of the address cellar and ask for the two operands, digit by digit. The third register gives the position of the "momentary free storage" and controls the digit-wise storage of the result. For division, it is necessary to transfer the dividend, before the start of the calculation, into the "momentary free storage." If unnamed intermediate results are involved in the operation, the result is transferred to the place of the first (the lowermost in the address cellar) of the intermediate results involved and consumed by this operation. The final result of the formula is therefore always positioned at the "beginning free storage" (register F) and this can at the same time be used as a test.

The usual arithmetical registers, with their inherent limitation to a certain number of digits, are thus dispensed with, and the only limitation is given by the number of digit places available in the entire free storage. A "parallelization" of the computation may be reached by enlarging the adding and multiplying unit and by multiplying the reading and writing connections from and to the storage. The numbers are then dealt with in blocks of several digits. Care must be taken that the end characters of numbers block the operation with digits belonging to the two numbers adjacent the respective operands.

FIGURES No. 7, 8, 9, 10 and 11, connected as indicated in FIG. 12, show in detail the main components of the computing machine represented schematically in FIG. 1. In these figures, certain simple components have been numbered for the purpose of this description. Whenever such components are part of one of the larger aggregates shown schematically and numbered in FIG. 1, the number given this component in FIGS. 7 through 11 consists of three digits, the first of which is identical with the number of the block in FIG. 1 containing the component in question.

FIG. 7 shows keyboard 1, from which input information, coded in a 5-bits code, is fed through 5 lines to the input of code converter (predecoder) 500, which is a

code converting matrix of well known construction. This converter is connected by pulse lines to flip flops 521, 522 . . . 532 in such a way that whenever one of these flip flops is set, all others are reset at the same time. Each of these flip flops is associated with the formula character indicated, the asterisk representing more flip flops associated with characters which are used in the more advanced versions of the computer described. Flip flops 521 through 530 form decoding register E referred to in Table I. Decimal figures (digits) are passed to numbers converter 7, from which 4 pulse lines carry them in excess-3 code to arithmetic unit 10 (FIG. 10).

From set position output of each of flip flops 521 through 530, a voltage line leads into operations converter 8 and ends finally in decoding converter 800. These lines together with a second series of voltage lines coming from converter 801, each of these lines being associated with the formula character indicated, form an operation decoding matrix (operations converter 8 of FIG. 1), which is shown schematically in FIG. 2. The input to converter 801 consists of three voltage lines coming from set side outputs of flip flops 120 (FIG. 8) which together form the uppermost story of operations cellar 12, which story is identical to discriminating register D referred to in Table I. Converter 801 is of known construction, converting the 3-bits input to a one-out-of-seven output of voltages.

Keyboard 1 may be a conventional keyboard, such as that described in Tompkins et al, "High Speed Computing Devices," McGraw-Hill Book Co., Inc., New York 1950, page 399, fourth full paragraph.

Numbers converter 7 and decoding converter 800 may be constructed in accordance with the converter illustrated in FIGURE 4-3b, page 43 of Tompkins et al.

Converters 500 and 801 may be constructed in accordance with the converter illustrated in FIGURE 4-3a, page 42 of Tompkins et al.

The operations decoding matrix consists of a number of AND-gates with two voltage inputs, one pulse input and one pulse output. Voltage inputs of these gates are connected to the voltage lines coming from register E (flip flops 521-532) and converter 801 in such a way that

- (a) for each gate, one of these inputs is connected to one of the lines coming from register E and the other input to one of the lines coming from converter 801.
- (b) every combination of one line each from register E and converter 801 is connected to an AND-gate.

The input pulse of each of these AND-gates comes through a delay line from keyboard 1. The pulse outputs of the gates are connected to operations signal lines 841 through 854, each of these lines being marked by a circle with a roman letter inscribed.

For the AND-gates of the matrix drawn explicitly in FIG. 7, a circle with roman letters inscribed indicates the operation signal lines connected to the output of the gate. For the sake of lucidity, only seven of the gates are shown explicitly. Connections for all gates are shown in the diagram below, which is a more elaborate version of the scheme given in FIG. 2.

Especially, the arithmetic signals briefly calling "a" and "r" in FIG. 2 and in Table I are broken up into the following component signals.

"a" (FIG. 2): "a_x," "a₊," "a₋" according to content X, +, - of register D and signal "c" followed by signal k.

"r" (FIG. 2): "a_x," "a₊," "a₋" as above and signal "c" followed by signal "w" which last means just the "repeat" part of the original signal "r."

The row designated by "=>" in FIG. 2 is split up into the two special cases "=" and "=>." The signal "y" means "prepare for typing."

The additional signal "f" means "prepare transfer to numbers storage 22" and is necessary when character "=>" is used in the meaning "store result," described above but included in FIG. 2, this figure corresponding to the most simple version of the computer.

Operations signal lines "w," "n," "f," "e," "a₊," "a₋," "a_x," "a₊," "a₋" and "s" lead to arithmetic unit 10 (FIG. 10) (and numbers cellar 11 (FIG. 9)), "f" leading also to numbers converter 7. Signal lines "c" and "k" lead to operations cellar 12 (FIG. 8), "k" leading also through a delay to decoding converter 800.

This decoding converter 800, with voltage inputs from flip flops 521 through 530, has as outputs three pulse lines leading to flip flops 120 of operations cellar 12.

FIG. 8 shows operations cellar 12, abbreviated OC. Operations cellar 12 is a flip flop register containing a number of "stories" consisting of three flip flops in parallel. The register is used for temporarily holding formula characters (B X + - and ⊗ as above described) as code signals consisting of three binary digits, each story being able to hold one such code signal. The uppermost story, which is identical to discriminating register D, consists of flip flops 120. Voltage outputs of these flip flops lead to converter 801, and their set inputs are connected by pulse lines to decoding converter 800.

The operations cellar is equipped with facilities for shifting its content upwards as well as downward. For this purpose, a series of pulse lines coming from delay line 126 is connected to the reset inputs of all flip flops of the register. The trigger output of every flip flop (121, for instance) is connected to a delay (127, for instance), from which one pulse line leads through a pulse gate (128) to the set input of the flip flops above (129), a second pulse line leading through a second gate (129) to the set input of the flip flop below (122). All gates on pulse lines leading upwards (that is, analogous to gate 128) are connected to the voltage output in set position of a flip flop 124; gates leading downward (analogous to gate 129) are correspondingly connected to flip flop 125.

Operations signal line 853 ("c") leads to set input of flip flop 124, to reset input of flip flop 125, and to delay line 126. Operations signal line 854 ("k") leads, through a delay line, to reset input of flip flop 124, to set input of flip flop 125, and to delay line 126.

A "lower OC" pulse coming from operations converter 8 on line 854 sets flip flop 125, thereby opening all gates

Table II

v (B X + -) => =	nf k kn k k k k st sy	nf k kn k k k c	f ke k k k a_{ncw} a_{ncw} a_{ncw} a_{ncw} a_{ncw}	f ke k k k a_{xck} a_{xck} a_{xck} a_{xck} a_{xck}	f ke k k k a_{+ck} a_{+ck} a_{+ck} a_{+ck} a_{+ck}	f ke k k k a_{-ck} a_{-ck} a_{-ck} a_{-ck} a_{-ck}	f ke k k k a_{-ck} a_{-ck} a_{-ck} a_{-ck} a_{-ck}
E / D	⊗	(	B	X	÷	+	-

leading downwards (as, for instance, gate 129); resets flip flop 124, closing all gates leading upwards (such as gate 128); and, after passing through delay 126, resets all flip flops of the operations cellar proper, such as flip flops 121. Of these flip flops, all those which have been reversed by this reset pulse having been in set position before (that is, having contained the binary digit 1), emit a trigger pulse which passes through the corresponding delay (127), and through the gate (129) opened by flip flop 125 and set the flip flop below (122), whereas no pulse is emitted by flip flops which were in reset position before. In this way, every flip flop of operations cellar 12 is set or reset according to the set or reset state of the flip flop above before the pulse "lower OC." The uppermost flip flops 120, however, are set by the outputs of decoding converter 800, when triggered by a pulse coming from 854 through a delay D.

Except for an exchange of roles of flip flops 124 and 125, which leads to opening of the gates leading upwards such as 128, the same chain of events takes place when a "raise OC" pulse enters on line 853. In this case, the content of the operations cellar is shifted upwards, the previous content of the uppermost flip flops 120 being discarded in the process.

FIG. 9 shows numbers cellar 11 (abbreviated NC). Numbers cellar 11 is a flip flop register for temporarily holding decimal numbers as above described. Except for a trivial difference in numbers of flip flops in parallel, the register proper is identical in construction to the register of operations cellar 12, the "story" of the operations cellar being replaced by the "decimal digit position" of the numbers cellar, which, to hold decimal digits in binary code, requires four flip flops in parallel per decimal digit. Decimal digits may be represented by signals in the so-called excess-three code. Since numbers held in the numbers cellar may consist of an arbitrary number of digits, a special "end" character is needed for separating consecutive numbers, which character may be represented by the combination "0; 0; 0; 0;" of binary digits; that is, all four flip flops of the respective digit position are in reset position. Furthermore, three code signals representing the signs "+" and "-" and the decimal point are needed, which, together with 10 different decimal figures and the "end" signal, gives fourteen different combinations out of the sixteen possible with four binary digits.

The numbers cellar has the same facilities for upwards and downwards shifting as operations cellar 12. The terms "upwards" and "downwards" however, are used in this description in the sense in which they have been introduced in Table I in conformity to the configuration of FIG. 2, whereas, in FIG. 9, for reasons of compactness and clearness of arrangement of the drawings, directions in the numbers cellar had to be inverted. Thus, "upwards" in this description, in the drawing of FIG. 9 means the downwards direction, and correspondingly "downwards" in this description, means the upwards direction in FIG. 9. Flip flops 114, being located in the lowest position in FIG. 9, are referred to in this description, as the "uppermost digit position" of numbers cellar 11.

Direction of shifting of the content of the numbers cellar is given by one flip flop 111 in such a way that all gates leading upwards are connected to the set output of flip flop 111, and gates leading downwards are connected to its reset output. The set input of flip flop 111 is connected by a pulse line to the output "raise NC" of the arithmetic unit 10, the reset input of 111 being connected to the output "lower NC" of the arithmetic unit. Both pulse lines are joined afterwards to line 113 leading to a series of pulse lines which are connected to the reset inputs of all flip flops of the numbers cellar proper, such as flip flop 114. Thus, a pulse passing through shift line 113 effects a one digit shift of the

content of the entire numbers cellar in the direction prescribed by the state of flip flop 111.

Since, in the numbers cellar, complete numbers are to be shifted upwards or downwards, reset outputs of all flip flops 114, representing the uppermost digit of the numbers cellar, are connected to a comparator circuit 112, which furthermore has one pulse input connected through a delay to shiftline 113, and two pulse outputs. One of these outputs, meaning "end character," leads to the input designated "end shift" of arithmetic unit 10. The other output of comparator 112, meaning "not end character" leads back to shift line 113. Consequently, the shifting circuitry of the numbers cellar as a whole possesses two signals inputs "raise NC" and "lower NC" and the one pulse output "end character." Comparator circuit 112 may be constructed in accordance with FIGURES 14—14, page 377 of Tompkins et al.

The set inputs of flip flops 114 of the uppermost digit position of the numbers cellar are connected by four pulse lines to outputs of arithmetic unit 10, to outputs of numbers converter 7, and to numbers storage 22 (not represented). Furthermore, shift outputs of flip flops 114 in the upwards direction are connected by four pulse lines to inputs of arithmetic unit 10.

If now, a pulse is emitted from output "raise NC" of arithmetic unit 10, flip flop 111 is set, thereby opening all gates of the numbers cellar leading upwards, and closing all gates leading downwards. Furthermore, the pulse through line 113 resets all flip flops of the numbers cellar proper causing a one digit shift upwards of the entire content of the numbers cellar, the content of uppermost digit position 114 being transferred to arithmetic unit 10. The character entered into flip flops 114 from below by this shift switches comparator 112 in such a way that, upon arrival of the pulse entering the pulse input of comparator 112 through delay 117 from line 113, a pulse is emitted from the output meaning "not end character" if the character represented by flip flops 114 is not the end character mentioned above, whereas a pulse is emitted from the output meaning "end character" in the contrary case.

Thus, if the character now contained in flip flops 114 is not an "end" character, the pulse emitted from comparator 112 and passing through line 113 again causes a one digit upward shift of the entire numbers cellar, the character contained in flip flops 114 again being transferred to arithmetic unit 10. This process repeats itself as long as the character newly entered into flip flops 114 is not an "end" character. When finally an "end" character is shifted into flip flops 114, the output pulse of comparator 112 is, from output "end character," directed to input "end shift" of arithmetic unit 10, and the upwards shift is terminated.

In a completely analogous way, a number is shifted from the arithmetic unit 10 into the numbers cellar by a cyclic process initiated by a pulse emitted from output "lower NC" of arithmetic unit 10, which resets flip flop 111 and passes into shift line 113.

In a similar way, numbers coming digit for digit from keyboard 1 are entered digit for digit into the numbers cellar, the initial shift pulse coming from operations converter 8 on operations signal line 843 ("f"), the other shift pulses being drawn from the shifted information directly on line 108. This is controlled by gate 106 operated by flip flop 107.

FIG. 10 shows arithmetic unit 10. This is a well known device for adding, subtracting, multiplying and dividing two given numbers, described extensively in the literature, and is, for reasons of compactness and clearness, only represented schematically. Arithmetic unit 10 may be a conventional arithmetic unit such as that illustrated in U.S. Patent No. 2,604,262 to Phelps et al. The connection, illustrated in FIGURE 10 of the drawing, labeled "raise NC" is equivalent to "take first operand" and "take second operand" of a conventional

21

arithmetic unit; "lower NC" is equivalent to "store results." The signals entering arithmetic unit 10 via the 7 lines entering from the right (which derive from the terminals 844 to 850 of FIGURE 7) correspond to the lines carrying instruction pulses in Phelps.

The arithmetic unit possesses pulse inputs connected to operation signal lines 844 ("e"), 845 ("a₊"), 846 ("a₋"), 847 ("a_x"), 848 ("a₊"), 849 ("a_B"), 850 ("s"). More inputs of this type may be provided for if further operations, as for instance R which means "reciprocal of number," or square root are wanted. Four pulse lines are connected to trigger outputs of flip flops 114 (FIG. 9), serving for entering numbers from numbers cellar 11. The input "end shift" has been described in connection with shifting facilities of numbers cellar 11 (FIG. 9). The same holds for outputs "raise NC" and "lower NC." In addition, there is a pulse output "end signal arithmetic operations" which is connected, through gate 103, to keyboard 1 and to reset input of flip flop 104, and is further connected, through gate 101, to the pulse line coming from delay 52, and serving all pulse inputs of the AND-gates of operations converter 8, and to the reset input of flip flop 102 which switches gates 101 and 103. Finally, four output lines lead from arithmetic unit 10 to flip flops 114 of numbers cellar 11 (FIG. 9), serving for transferring results from arithmetic unit 10 into numbers cellar 11.

If the arithmetic unit, on one of the arithmetic operations signal lines 844 through 849, receives a pulse from operations converter 8, it first causes the transfer of two numbers in the case of operations "a₊", "a₋", "a_x", "a₊", addition, subtraction, multiplication and division, and one number in the case of operation "a_B" "absolute value," into the arithmetic unit, by emitting twice in the case of "a₊" through "a₋", and once in the case of "a_B", a pulse from output "raise NC." The arithmetic unit then carries out the arithmetic operation ordered and returns the result into numbers cellar 10 by emitting one pulse from output "lower NC." After receipt of the corresponding end signal by a pulse to input "end shift," the arithmetic unit finally signals the end of the arithmetic operation ordered by emitting a pulse from output "end signal arithmetic operations," which passes either through gate 103 to keyboard 1, permitting thus entering of a new character from the outside or passing through gate 101 to operations converter 8 and causing in this way the repetition of the operations converting process indicated by operations signal w, which, by setting flip flop 102, had opened gate 103 at the beginning of the arithmetic operation just terminated. In this last case, the pulse passing through gate 101 immediately afterwards resets flip flop 102.

Operation signal "e" causes the arithmetic unit to feed a zero into numbers cellar 11. Operation signal "n" eliminates the zero. It is controlled by flip flop 104, which is reset from "end signal arithmetic operations," and which is set by the first pulse passing the gate 105, which closes this gate, thereby avoiding an unwanted repetition of the elimination when the digits of a number are fed from the keyboard 1 (FIG. 7) to the numbers cellar 11 (FIG. 9). Operations signal "s" merely causes transfer of a number from 11 to 10; it is followed by operation signals "i" or "y," which control output to number storage 22 or typing device 2 by means of flip flop 605 or 604 (FIG. 11) and the dependent gates 603 or 602 (FIG. 11) respectively. Gate 602 is an AND-gate, which is also controlled by flip flop 531 (FIG. 7). The output voltage "□" of 531 also is connected to the input "digit requested" of the arithmetic unit 10 (FIG. 10) which causes output of the next digit through gate 602 to the output decoder 601 and typing device 2 (FIG. 11).

FIG. 11 also shows connections to a number storage, which is of well known construction.

In FIGS. 9 and 10, an example was given of numbers

22

cellar 11, and arithmetic unit 10 was assumed to be of conventional design. FIGS. 13 through 17 as an alternative give in more detail an example of a numbers cellar and arithmetic unit combined as described above in connection with FIG. 6.

FIGURE 13 replaces FIGS. 9 and 10, connections with FIGS. 7, 8 and 11 being given in FIG. 12a. FIGURE 13 contains schematic representations, numbered by Roman numerals I through VIII, of the combined numbers cellar-adding network, of the different switching circuits controlling arithmetic and input-output operations initiated by pulses coming through operations signal lines 841 through 850 from operation converter 8, and of connections between the components represented and operations converter 8, number converter 7, output control 6, and numbers storage 22.

Numbers cellar-adding network I (see FIG. 14) possesses control inputs numbered 10 through 112. Each of these inputs, when pulsed from one of control circuits II through VIII, or from numbers storage 22 through line 146, causes a certain switching or shifting operation as listed in the description to FIG. 14. Control outputs 1a through 1s serve for returning signals, likewise listed in the description to FIGURE 14, to the control circuits. Input 171 is the numbers input connected by four parallel pulse lines to numbers converter 7, and numbers storage 22. Output 161 is the numbers output connected by four pulse lines to output control 6 . . . , and numbers storage 22.

Operations signal lines (FIG. 7) 841 ("w") through 850 ("s") and 853 ("c"), 854 ("k") are connected to the inputs of the corresponding control circuits as shown in FIG. 13. Thus, a pulse coming from operations converter 8 or one of these lines activates the corresponding control circuits II-VIII in such a way as to cause and control execution in I, of, the operation ordered, by emitting pulses to control inputs 10 through 112 of I, and receiving, and evaluating, return signals from control outputs 1a through 1s of I.

From outputs 113 through 1113 of control circuits II through VII, the "end signal arithmetic operations" is given through line 145 to input 1113 of the operations cycle control contained in VIII (as shown in FIG. 17) whenever the operation controlled by one of circuits II through VII is completed. All arithmetic signal lines 845 ("a₊") through 849 ("a_B") are furthermore connected to input 1115 of VIII. Output 1116 is connected to keyboard 1 and output 1117 of the pulse line serving all AND-gates of operations converter 8 (see FIG. 7). A pulse is emitted through one of these two outputs by the operations cycle control whenever an "end signal arithmetic operations" arrives. Output 1118 finally leads to numbers converter 7. A pulse is emitted on this line whenever, by operations "f," a digit of a number is entered into I from keyboard 1.

Finally, operations signal lines "i" and "y" coming from operations converter 8, and leading to output control 6 (FIG. 11) and the lines "digit requested" from flip flop 531 "□", leading to input 12 of I, and to output control 6, cross FIGURE 13.

FIGURE 14 shows the numbers cellar-adding network, which consists of the main body of the numbers cellar here designated "NC" the multiplicand register "MD" which at the same time is the uppermost story of the numbers cellar, the digit adder 135, counter 131, two comparators 132, 133, and a number of flip flops with gates attached which control the different shifting and transfer operations possible.

Both NC and MD are shift registers of the same type as operations cellar 12 (FIG. 8) and numbers cellar 11 (FIG. 9) with 4 flip flops (binary digits) per digital position. In the case of FIG. 14, however, each of the two registers is assumed to form a complete loop, which means that upper and lower ends are connected, and a shift always is a circular movement of the contents of all digital positions of the register shifted.

Direction of shifts is given by flip flops 123, 125 in the case of NC, 124, 126 in the case of MD. Contrary to the situation of FIG. 9, directions "upwards" and "downwards" in FIGURE 14 are in agreement with definitions given in Table I. Gates leading upwards in NC are open, and gates leading downwards are closed, when flip flop 123 is set, and 125 is reset, by a pulse coming from input 12 "raise NC." Correspondingly, gates leading upwards in MD are open, and gates leading downwards are closed, when flip flop 124 is set, and 126 is reset, by a pulse coming from input 14 "raise MD." Conversely, gates in NC are switched for downwards shifting by a pulse resetting 123, and setting 125, from input 15 "lower NC," and gates in MD are so switched by a pulse resetting 124, and setting 126, from input 16 "lower MD." Input 11, "shift NC," delivers the actual shift pulse to all flip flops of NC causing a one digit shift; input 12 "shift MD," correspondingly delivers the shift pulse to MD.

Between digital positions 150, 151 of MD, and 152, 199 of NC and adder 135, various connections exist which are controlled by gates 137 through 141, switched by flip flops 127 through 130. Inputs of these flip flops are connected to inputs 17 through 110 of I in such a way that, whenever one of the flip flops is set from one of the inputs 17 through 110, all others are reset at the same time.

Flip flop 127 is set by input 17 "loop" and opens gates 137, 140 and 141. This makes closed loops with independent shifting of MD and NC by completing connections between 150, and 151 of MD on the one side, and between 152 and 199 of NC (through adder 135 in the downwards direction) on the other side.

Flip flop 128 is set by input 18 "join" and opens gates 138. This joins MD and NC by completing the connection between 151, and 152, and makes a continuous chain beginning with 150 through 151, 152, and ending in 199, of MD and NC, thus permitting numbers to be shifted from MD into NC and vice versa.

Flip flop 129 is set by input 19 "add" and opens gates 139 and 141. This connects inputs leading downwards from 151, and 199, to adder 135, which has a single numbers output leading downwards to 152 of NC. This permits adding of two numbers located in 151 and above in MD, and 199 and above in NC by synchronous downward shifting. The result is entered into 152 and downward of NC.

Flip flop 130 is set from input 110 "multiply." This opens gates 139, 140 and 141. Thus in addition to the "add" configuration effected by gates 139, 141 MD is "loop" position, and, in the process of addition of NC and MD as in "add" configuration, the content of MD is shifted cyclically through the register and is not lost. This, of course, is necessary for multiplication.

Signal input 10 "clear MD," enters "end" characters consisting of four binary 0's into all positions of MD. Signal input 112 "enter digit" sets flip flop 131, which in turn opens gate 141 controlling numbers input 171, which is connected to digit position 151, of MD. Digits are entered into MD by operation "f," by means of an upwards shift of MD, and the digit in numbers converter 7. Therefore, numbers in MD and NC, always have their most significant digit in the uppermost position. Digit position 150 finally is connected to numbers output 161. Therefore, by shifting MD upwards, the content of MD is removed through output 161 again with leading figure first.

Digit position 150 of MD and 152 of NC are connected to comparators of known design, numbered 132, and 133 respectively and switch them in such a way that the comparator emits a pulse to one of several outputs, each according to the contents of the corresponding digit positions. Outputs shown are, in the case of comparator 132 and digit position 150: I_e "end character MD," I_f "decimal point MD," I_g "decimal digit MD" and in the case of comparator 133 and digit position 152: I_a "end

character NC," I_b "decimal point NC" and I_γ "decimal digit NC."

Digit position 152 is furthermore combined with a counted 134, which has a pulse input 111 "count MR" and two pulse outputs I_η "count ended" and I_θ "count continued." This counter counts the decimal digit contained in 152 down by one. If the digit before the count was 1 or larger, the count is executed, and output I_θ is pulsed. If however the digit was 0 before the count, it is transformed into the "end" character, and output I_η is pulsed.

The adder 135 is a simple two digits adder with two digits inputs, one carry input, and one result output, which is well known. In addition it treats the "end" character as a figure "0." Furthermore whenever a decimal point is entered from either MD or NC, an additional shift pulse is immediately emitted to the corresponding shift input, 1 or 2, respectively, and addition is retarded until meaningful after entering of the next digit from the side from which the point has been entered. Alternatively, the method of handling the decimal point described in the preceding paragraph or the method described above (column 10 of the specification) in connection with the description of FIGURE 5, may be employed.

FIGURE 15 shows multiplication control V as the first and most important of the operations control circuits. This circuit consists of a series of flip flops V21 through V32 representing the different stages executed consecutively during multiplication. These flip flops each switch a gate (or several gates) which control a pulse line leading from one of the outputs I_a through I_θ of I to one or several of inputs 10 through 112 of I. In each stage of multiplication, the pulse terminating it resets the corresponding flip flop, and sets the flip flop controlling the next stage, the first stage being set initially by the multiplication signal coming through operations signal line 847 (" α_x " FIG. 7).

Multiplication of the number in MD and the uppermost number in NC which consists of digits in positions from 152 downwards to the next end character, (position 152, too, containing an end character) is executed by consecutively adding MD into NC as many times as the momentarily uppermost digit of NC indicates, then shifting NC upwards by one digit relative to MD, thus eliminating the multiplier figure, and repeating the process until the number in NC is exhausted. Thus, the initial pulse coming from 847 raises NC by one digit, moving the uppermost digit of NC into 152, and sets the first stage flip flop V21. In this stage a digit in 151 causes a count in 134, and the setting of the second stage, V22. A result "count continued" in the second stage means that the digit in 152, being the present multiplier (MR) figure, is not yet zero, and addition of MD into NC is prepared by setting the "multiply" connection of MD and NC and shifting MD and NC downwards synchronously until the last digit of MD has passed position 150.

The first end character appearing in 150 shows that the entire multiplicand has been added into NC. It therefore sets the "loop" connection in NC and MD, ends stage 2, and sets the third stage, V23, in which both registers MD and NC are shifted downwardly synchronously in "loop" connection, thus permitting carries to be executed in NC and still retain correspondence of digit positions in NC and MD. An end character appearing in 152 means that the highest position of the partial products in NC has been reached and causes a reversing of the direction of shifting of both registers, since the present addition is completed. The next two stages, V24, V25, contain the upwards shift, with an extra shift of MD for the decimal point of both registers which must be broken up into two stages to insure terminating the shift at the correct moment, that is, when the "end" character following the number in MD enters 152, and

avoiding the ambiguity possibly resulting from the double length (with respect to MD) of the product.

Stage 5 and with it the inner cycle of the multiplication process, is terminated by the appropriate "end" signal from I32, which at the same time restarts the cycle by setting stage 1 flip flop V21.

If now, the count made in stage 1 results in a "count ended" signal, the present MR digit is exhausted, and the inner cycle of the process is finished for this digit. Therefore, the initial multiplication pulse is repeated, shifting NC upwards once and setting stage 1. This is the outer cycle of the process, and restarts the addition cycle as long as a digit reappears in I52. If the character "decimal point" enters I52, it causes an immediate repetition of the second cycle starting with the initial multiplication pulse. At the same time, it sets flip flop V40, thus indicating that the multiplier in NC contains a decimal point (which is lacking in the case of whole numbers). Thus, the partial product now contains a decimal point in the position of the original multiplier decimal point. If, however, at the moment of setting of stage 1 an "end character" appears in I52, indicating that the entire multiplier is exhausted, the double cycle of addition is completed, and the transfer of the result from the uppermost place of NC into MD is started. At the same time, the decimal point must be inserted into the appropriate position of the product, which still contains the decimal point of the multiplier in its old position. This is done in stage 6, V56, by first shifting NC downwards in "loop" connection until the point enters I52. Stage 6 is omitted whenever V40 shows that the multiplier did not contain a decimal point. Then both NC and MD are shifted downward synchronously (stage 7, V57) until the point of MD enters I50. One upwards shift in both registers then moves the point of MD into I51. Now both registers are shifted upwards in "join" position (stage 8, V58) until the point of NC arrives in I52. One more shift upwards in "loop" connection in NC alone carries the point into I99.

In stage 9 (V59), the result digits transferred into MD in stage 8 are returned into NC by shifting NC and MD downwards in "join" connection until the point above said digits enters I52. In this way the point is moved upwards, from its original position in the multiplier, by as many positions as MD contained below its point, which obviously gives the correct result. Stages 7 through 9 are omitted when flip flop V41; which is reset by a signal "decimal point" in MD, shows (by being reset) that MD contains no point.

Stage 10 (V60) shifts NC downwards in "loop" connection until an "end" character showing the end of the number appears in I52. This signal clears MD and sets stage 11 (V61), which by shifting NC and MD upwards in "join" connection, enters the result into MD. If the length of the result fits into MD, the end signal arriving in I52 terminates the shift process. If the result has too many digits, a digit appearing in I50, indicating overflow, shifts MD and NC downwards by one position, thus returning the "end" character into I50, and sets stage 12 (V62) which eliminates the excessive digits of the result by shifting NC upwards in "add" connection until the "end" character appears in I52.

The "end" character appearing in either stage 11 or stage 12 terminates the entire multiplication process, and causes a pulse "end signal multiplication" from multiplication control output V13 to input VIII13 of VIII (FIGS. 13, 17).

FIGURE 16 shows the addition control circuit as a second example of an arithmetic control circuit. Overall design is analogous to the case of multiplication. The initial pulse "a+" coming from 845 (FIG. 7) sets the "loop" connection, causes one upwards shift in NC, and sets the first stage flip flop VII20.

In this first "preparative" stage, the content of MD is

shifted upwards until the highest digit appears in I50. This sets the second or point aligning stage represented by set position of VII21 and VII22. In this stage both MD and NC are shifted upwards until the first decimal point (either in MD or NC), or alternatively, the first "end" character in the case of whole numbers, arrives in either I50 or I52. This sets the third stage, in the first (decimal point) case by setting VII23, in the second (whole number) by setting VII24. In this stage, the register in which the point or "end" character is already identified and in position remains unaltered, whereas upward shift in the other register is repeated until the second point, or "end" character appears. This ends stage 3, with decimal points and/or "end" characters aligned in any case, and, in the case of two decimal points only, sets stage 4 flip flop VII25. This causes both registers MD and NC to be shifted upwards synchronously in "loop" connection until the first "end" character appears in I50 or I52.

This terminates stage 4, which is skipped if, after stage 3, at least one "end" character is in I50 or I52, that is, if at least one of the summands is a whole number. In every case the numbers in MD and NC are in correct position for addition, which is started by the pulse terminating stage 4, or stage 3, in the case of whole numbers, and setting the "add" connection of MD and NC and fifth stage flip flop VII26. In this stage, addition of MD into NC is executed by shifting both registers downwards synchronously until an "end" character appearing in I52 indicates that the addition is finished. This sets sixth stage flip flop VII27, in which result is transferred from NC into MD by shifting both registers upwards in "join" connection. If the result does not exceed the capacity of MD, the shift is terminated by the appearance of the "end" character in I52, which causes stage 7 flip flop VII28 to be set. In this stage, MD is shifted upwards in "loop" connection until the "end" character appears in I50, which terminates this final shift. If, however, the result exceeds the capacity of MD, this is indicated by the appearance, during stage 6, of a decimal digit in I50. This first causes a downward shift in the "joined" MD and NC, thus returning the end character into I50, and sets flip flop VII 29 of stage 8. In this stage, excessive result digits are eliminated from NC by shifting NC upwards in "add" position until the next "end" character appears in I52.

The end character in I50 in stage 7, or I52 in stage 8, indicates the end of the entire addition process, and causes a pulse "end signal addition" from addition control output VII13 to input VIII13 of VIII (FIGURES 13, 17).

Operation control circuits IV (division) and VI (subtraction) being nearly identical to multiplication and addition respectively, are not shown explicitly in order to avoid unnecessary length. The same holds for the well known treatment of negative numbers, represented by an "L" sign, and the digits giving "absolute value." This requires evaluation of sign only in the case of multiplication, and division, whereas, in the case of addition and equal signs of the operands, the number contained in MD must be complemented by a well known procedure, and the result be de-complemented if in complementary form, which can be carried out by the adding unit.

Operations control circuit III "absolute value" only means setting the sign in MD to "+" in all cases.

Control circuit II is activated by operations line 850 ("s"), and controls the preparation for output, which means only that MD must be set in "upwards" "loop" condition, and shifted upwards by pulses to input I2 until a decimal digit appears in I50.

FIGURE 17 shows control circuit VIII, which contains controls for operations "e," "f," "n," and the operations cycle control, including the operation "w."

Operation "e" is intended to enter a zero into MD, this being the uppermost position of the numbers cellar.

This means that the number contained in MD at this

moment must be shifted into NC by setting "join" connection "downwards" condition in MD and NC by pulsing 15, 16 and 18, which is done by the initial pulse coming from 844 ("e"). At the same time flip flop VIII20 is set, and through two gates VII21 and VIII22 causes a synchronous downwards shifting of MD and NC by pulses coming from outputs I_β and I_γ. When a pulse coming from I_α shows that the transfer is complete, and MD is filled with "end" characters arithmetically equivalent to zero, flip flops VIII20 and VIII28 are reset, and an end signal is given through gate VIII23, which at this moment, is open to the line "end signal arithmetic operations" coming from the input VIII13.

Operation "f," entering a new digit into MD from numbers converter 7, for the first digit of a number, has to execute exactly operation "e," that is, to transfer the content of MD into NC before entering the digit into NC. The first pulse coming from operations signal line 843 "f" after completion of any operations different from "f" passes through gate VII25, connected to reset position of flip flop VIII24, which is reset by each "end signal arithmetic operations." Thus, gate VIII25 is open at this moment, the pulse sets flip flop VIII24, closing gate VIII25, and starts operation "e" which is executed as above with the difference that the terminating pulse coming from output I_α through gate VIII22 now finds gate VIII23, connected to reset side of VIII24 closed. Instead, gate VIII27, connected to set side of VIII24, is open, and the terminating pulse passing through the gate moves to numbers converter 7 and shifts the digit to be entered into position I51 of MD, at the same time shifting upwards MD once.

All pulses from "f" following this initial pulse, and signaling further digits of the same number, find gate VIII25 closed, and pass through now open gate VIII26 (connected to set side of VIII24) immediately to numbers converters 7 and I, causing the shift entering the new digit into MD. This operation is finished by any operations signal differing from "f."

Operation "n" is intended to eliminate a zero entered unnecessarily by an immediately preceding Operation "e." This is done in the following way: The first pulse coming from operations signal line 842 "n," after a preceding operation "e," finds open gate VIII29 connected to set side of flip flop VIII28. It resets this flip flop thus shutting off unwanted repetitions, and sets flip flop VIII24, thereby eliminating the shift of MD into NC caused by the first operation "f" connected with the first digit of the following number.

Operation "w" causes repetition of the operations converting process after the end of an arithmetic operation. A pulse coming from line 841 sets flip flop VIII30, which in turn opens gate VIII31. Through gate VIII31, the line coming from VIII13, and carrying all "end signals arithmetic operations" is connected to the pulse lines pulsing the AND-gates of operations converter 8. The next pulse which, coming from VIII13, passes through gate VIII31, therefore causes, after having reset VIII30, a repetition of the operations converting process. If VIII30 is reset, however, gate VIII32 is open and the pulse from VIII13 passes to keyboard 1, requesting new information to be entered from the outside.

Operations "c" and "k" finally are connected to the line from VIII13 through input VIII14 and a delay gate VIII33, which is connected to reset side of flip flop VIII34. This is set by any arithmetic operation, which thereby prevents simultaneous signals from "C" or "k" to cause an "end signal" to the keyboard, or to operations converter 8.

We claim:

1. A formula controlled computer, comprising an input device for generating a series of machine characters including a machine character representative of each numerical and operational symbol of a mathematical expression to be evaluated, means for separating numerical

and operational characters generated by said input device, storage means for numerical characters and storage means for operational characters, each said storage means being arranged to make said characters available for operations on a last-in, first-out basis, computer means, output means for delivering the results of computation, and control means comprising a matrix for producing specific control impulses responsive to specific combinations of incoming and last-in operational stored characters, including impulses directing the introduction of an incoming operational character into said storage means and the stepping of an already stored character to an adjacent position to make room for such incoming character, impulses directing the removal of the newest operational character stored in said storage means and the stepping of the next newest stored character into the position vacated by the removed character, impulses directing the transfer of numerical characters from said numerical storage means to said computer means, impulses specifying the nature of the operation to be performed by said computer means, and impulses directing the printing out of information by said output means.

2. Data processing equipment controlled by information in the form of a series of characters of n types, representing respectively n of the operational symbols of mathematical notation interspersed, according to the laws of mathematical notation, with characters representing numerical symbols of mathematical notation, said equipment comprising computing sub-units and sub-units for storing characters of said n types, in which storage sub-unit the last of the stored characters remaining therein is available for identification as a specific one of said n types, and means for controlling said sub-units, responsive to different combinations of a specific incoming character with a specific available stored character of one of said n types to control different specific sub-units.

3. Data processing equipment controlled by information in the form of a series of characters of n types, representing respectively n of the operational symbols of mathematical notation interspersed, according to the laws of mathematical notation, with characters representing numerical symbols of mathematical notation, said equipment comprising computing sub-units and a sub-unit for storing characters of said n types, in which storage sub-unit the newest of the stored characters remaining therein is available for identification as a specific one of said n types, means for identifying incoming numerical characters collectively as a further type $n+1$, means for identifying an incoming operational character as being a specific one of said n types, and control means for producing different specific control impulses for controlling different specific sub-unit functions, said control means including a matrix responsive to each specific combination of an incoming character of specific type and available stored characters of one of said n types to control a specific sub-unit function.

4. Data processing equipment controlled by information in the form of a series of characters of n types, respectively representing n of the operational symbols of mathematical notation and interspersed, according to the laws of mathematical notation, with other characters of v types representing numerical symbols of mathematical notation, said equipment comprising storage means for characters of said n types, in which means the last-entered of the characters remaining in said storage means is available for identification as a specific one of said n types, means for identifying incoming characters according to type and capable of distinguishing between different ones of said n types and of distinguishing said v types from characters of said n types to produce $n+1$ specific responses, and means for controlling specific equipment functions including operations of said storage means, including matrix means for generating a specific control impulse responsive to each different combination of one of

said $n+1$ responses with one of said available stored characters of one of said n types.

5 5. Data processing equipment controlled by information in the form of a first series of characters of n types, representing respectively n of the operational symbols of a mathematical notation interspersed, in a manner and order determined by the laws of such mathematical notation, with a second series of characters of v types representing numerical symbols of mathematical notation, said equipment comprising computing sub-units and storage
10 sub-unit means for characters of said n types in which the last entered of the characters remaining therein, if any, is stored in a first identification means capable of producing any one of $n+1$ different outputs, n of said outputs being representative of the presence of a character of one of said n types, and the remaining output being representative of an empty identification means, said storage sub-unit means also including a storage into which characters can be stepped from said first identification means and from which characters can be stepped to said first
15 identification means, second identification means for identifying an incoming character and capable of producing any one of $n+1$ outputs, n of said outputs being representative of the arrival of a character of one of said n types and the remaining output being indicative of the arrival of a character of any of said v types, and means for controlling specific sub-units functions responsive to each specific combination of one of said $n+1$ outputs from said first identification means taken with one of said $n+1$ outputs of second identification means to control a
20 specific equipment function.

6. Equipment for processing data supplied thereto in the form of a series of characters of a plurality of kinds, comprising storage means having an entrance position and a series of successively adjacent positions, each having a capacity of one character, and from one to another of which characters may be successively stepped, and means for generating impulses of a plurality of kinds for respectively controlling a plurality of respective data handling equipment functions one of said functions being the stepping of a character from one to another of said positions, said impulse generating means including a device for producing different specific control impulses responsive to different specific combinations of an arriving character and a character occupying said entrance position.

7. Equipment for processing data supplied thereto in the form of a series of characters of a plurality of classes, each class including characters of a plurality of types, said equipment comprising computing sub-units and a data processing sub-unit including register means for accommodating a single character of a first class and storage means for a plurality of characters of said class, into which characters may be stepped from said register and from which characters may be stepped into said register, decoder means for separating characters supplied to the equipment by classes and for separating characters of said first class by types, said decoder means having a separate output for characters of each type of said first class and an additional output for indicating the receipt of characters of a second class, circuit means for supplying characters of said first class to said register from said decoder, said register means being responsive to the type-characteristics of a character therein and having a separate output for characters of each type of said first class and a further output for indicating an empty register, and control means having inputs connected to the outputs of said register and said decoder and arranged to control any one of a plurality of sub-unit functions responsive to corresponding ones of the possible combinations of one output of said decoder taken with one output of said register.

8. Equipment in accordance with claim 7, for processing data whose characters of said first class represent operational symbols of mathematical notation and whose

characters of said second class represent numerical symbols interspersed therewith in accordance with the laws of a mathematical notation, and including a further sub-unit comprising register and storage means, similar to the
5 aforesaid register and storage means, for characters of said second class; further circuit means for supplying characters of said second class to said further register means; and an output sub-unit; the functions controlled by said control means including the transfer of characters among
10 said two register and storage means, said decoder, said computing sub-unit, and said output sub-unit, each such transfer being determined by the nature of the combination of a decoder output and a register output existing just prior to the occurrence of such transfer.

9. Equipment in accordance with claim 8, including an input instrumentality generating characters of a third class representative of input instrumental operations without mathematical significance but with significance to said output sub-unit; and circuit means for supplying characters
15 of said third class to said output means.

10. Equipment in accordance with claim 9, in which said input instrumentality is a keyboard-controlled character generator whose keys are identified by the symbols of mathematical notation, in which said output sub-unit
20 is a character-controlled printer having type for producing the symbols of mathematical notation, and in which said control means comprises a matrix including a plurality of AND-gates, each of which has a first input connected to an output of said register, a second input connected to an output of said decoder, and an output connected to a sub-unit function control circuit, the pattern of connection of said matrix being such as to control a
25 sequence of equipment functions responsive to successive combinations of incoming and stored characters which sequence is in accordance with the mathematical laws governing the order of mathematical operations.

11. A formula translator for a data processing system of the kind having a data processing element responsive to machine instructions, said translator being responsive to a series of operational and numerical machine characters representative of a series of operational and numerical language characters to be translated, comprising operational and numerical machine character storage means, each arranged to store a plurality of character
30 on a last-in, first-out basis, decoder means responsive to certain specific combinations of a specific incoming character and a specific last-in stored operational character to produce specific machine instructions for control of said element and responsive to specific combinations of a specific incoming character and a specific last-in stored operational character to produce specific control signals to effect specific operations of said storage means.

12. A translator in accordance with claim 11 in which said decoder means is responsive to one specific combination of an incoming and stored character to introduce an operational character into said storage means, and responsive to another specific combination of an incoming and stored operational character to remove the newest operational character in said storage means.

13. In a high speed computer of the kind having a plurality of arithmetic computing sub-units, an operational character cellar for storing operational characters representative of the operational symbols of a conventional language, said cellar comprising a plurality of storage
35 positions, one of said positions being an entrance position, means for introducing an operational character into said entrance position, means for shifting a stored character occupying said entrance position to an adjacent position to make room for a character being entered simultaneously with said entry and for similarly stepping a character in any position to an adjacent position to make room for a character being stepped thereinto, means for removing a character from said entrance position, means
40 for stepping characters in a reverse direction to occupy

31

positions vacated by a removed or stepped character, means for identifying the character stored in one of said positions, a decoder for identifying incoming operational characters, and control means connected to both said decoder and said identification means, and being differentially responsive to specific combinations of a specific incoming character and a specific character occupying said identification position to produce a plurality of different control impulses, including control impulses for stepping characters from one position to another in said cellar. 5 10

14. Equipment for processing information entering the equipment as a series of characters from a fixed character vocabulary comprising:

means for storing a plurality of characters of a plurality of types including characters corresponding to characters which enter the equipment; 15

means for comparing the last character stored in said storage means and still remaining therein with the last character entering the equipment;

the comparing means having output means for producing a plurality of different output signals, including a signal for controlling the said storage means. 20

32

References Cited in the file of this patent

UNITED STATES PATENTS

2,344,885	Kozma et al.	Mar. 21, 1944
2,604,262	Phelps	July 22, 1952
2,797,862	Andrews et al.	July 2, 1957
2,801,406	Lubkin	July 30, 1957
2,805,409	Mader	Sept. 3, 1957
2,815,168	Zukin	Dec. 3, 1957
2,834,543	Burkhart	May 13, 1958
2,848,166	Wagner	Aug. 19, 1958
2,856,595	Selmer	Oct. 14, 1958
2,874,901	Holmes	Feb. 24, 1959
2,933,720	Newhouse et al.	Apr. 19, 1960

OTHER REFERENCES

Engineering Research Associates (Tompkins et al.), High Speed Computing Devices, McGraw Hill Book Co., Inc., 1950, note FIG. 13-25, and page 299.

Wilkes et al.: Micro-Programming and the Design of the Control Circuits in an Electronic Digital Computer, Cambridge Philosophical Society Proceedings, vol. 49 (1953), pp. 230-238.

UNITED STATES PATENT OFFICE
CERTIFICATE OF CORRECTION

Patent No. 3,047,228

July 31, 1962

Friedrich Ludwig Bauer et al.

It is hereby certified that error appears in the above numbered patent requiring correction and that the said Letters Patent should read as corrected below.

Column 11, line 55, for "results" read -- result --; line 65, for "zz" read -- Zz --; column 14, line 8, after "carried" insert -- out --; line 62, for "subscrips" read -- subscripts --; line 71, for "lengtht" read -- length --; column 19, lines 51 and 52, for "confiuration" read -- configuration --; column 23, line 8, for "input I2" read -- input I3 --; column 24, line 4, for "counted" read -- counter --; line 43, for "monmentarily" read -- momentarily --; column 28, line 65, for "chacaters" read -- characters --; column 30, line 19, for "charatcers" read -- characters --.

Signed and sealed this 20th day of November 1962.

(SEAL)

Attest:

ERNEST W. SWIDER

Attesting Officer

DAVID L. LADD

Commissioner of Patents