# The Binac*

A. A. AUERBACH‡, ASSOCIATE, IRE, J. P. ECKERT, JR.†, ASSOCIATE, IRE, R. F. SHAW‡, SENIOR MEMBER, IRE, J. R. WEINER†, ASSOCIATE, IRE, AND L. D. WILSON†, MEMBER, IRE

*Summary*—**The Binac was the first high-speed electronic digital computer of its type to be completed. It consists of a main computing section, input-output equipment, and a mercury delay-line memory of 512-word capacity. Although designed for a specific purpose, the Binac is well adapted to any type of general mathematical computation.**

**While the number of tubes is comparatively large, the Binac is a combination of a few basic circuits. Extensive use is made of crystal diodes in switching or gating circuits. These circuits fall into three basic classifications: function tables or diode matrices, used for converting coded groups of signals such as instructions into corresponding but differently coded groups of signals to operate switching circuits; switching gates, which control the flow of information in accordance with signals received from function tables or similar sources; and reshaping gates, which, as their name implies, reshape signals that have become distorted in electric delay lines or other circuits. The use of reshaping gates, together with suitable shaping networks, makes possible the use of a "pulse envelope" or "non-return-to-zero" system of representing information, thus effectively doubling the bandwidth of circuits.**

**Crystal gates are used with electric delay lines to form a serial binary adder. A one-word register is formed by combining an electric delay line with appropriate reshaping gates, input and output gates, and amplifiers, all connected to form a closed loop. Insertion of the adder in the loop results in an accumulator. By temporarily inserting or removing a one-pulse delay in the loop, information can be made to precess, which is equivalent to multiplication by positive or negative powers of two.**

**Input data are supplied to the memory from a keyboard or magnetic tape through the use of a synchronizer which reconciles the randomly timed input signals with the accurately timed internal memory. The same synchronizer is also used for transferring data from the memory to magnetic tape, or to paper by means of an electric typewriter.**

**Extensive tests have demonstrated the usefulness of the Binac in the solution of mathematical problems. Some striking results have been obtained in connection with the solution of Poisson's equation for various boundary conditions. An example is given of the coding for a square-root routine (see Table I).**

## I. Introduction

THE BINAC, first computer of its type to be completed successfully in the United States and successor, historically, to the electronic numerical integrator and computer (ENIAC), is a high-speed, automatic digital computer, operating in serial fashion on numbers expressed in binary form. The digits occur at a repetition rate of four million per second, considerably faster than any computer yet built. Thus, it may truly be called "high speed." The computer is automatic in the sense that, properly instructed, it will perform a complicated series of computations by itself. This is in contrast to a desk calculator, for instance, which assists the human operator by performing individual elementary operations, but must be separately manipulated for each such operation.

The major instructions which the Binac can follow are the four basic arithmetic operations, instructions involving the moving of data from one storage register to another, and the so-called "conditional and unconditional transfers of control" which contribute largely to the automaton characteristic of the computer. Normally, the Binac follows instructions in their originally given sequence. A transfer of control instruction, however, may tell the computer to proceed to another instruction that does not follow sequentially (i.e., either

TABLE I

Square-Root Routine for Binac

| | | | |
|---|---|---|---|
| 000 | 25000 | | Skip |
| | | $U$ 024 | Transfer to 024 to start computation |
| 001 to | 023 inclusive | | $\mu$'s (radicands) |
| 024 | $A$ (001) | | |
| | | $H$ 057 | $\mu$ to storage register 057 |
| 025 | 23000 | | Shift right (divide by 2) |
| | | $A$ 046 | $Z_i=1/2\ (\mu+1)$ |
| 026 | $C$ 055 | | $Z_i$ to storage register 055; clear $A$ |
| | | 25000 | Skip |
| 027 | $A$ 057 | | $\mu$ to $A$ |
| | | $D$ 055 | $\mu/Z_i$ |
| 030 | $S$ 055 | | $\mu/Z_i-Z_i$ |
| | | 23000 | Shift right (divide by 2) |
| 031 | $A$ 055 | | $1/2\ (\mu/Z_i+Z_i)=Z_i+1$ |
| | | $C$ 056 | $Z_i+1$ to storage register 056 |
| 032 | $A$ 055 | | $Z_i$ to $A$ |
| | | $S$ 056 | $Z_i-Z_i+1$ |
| 033 | $S$ 054 | | $Z_i-Z_i+1-2^{-14}$ |
| | | $T$ 036 | Test sign of $Z_i-Z_i+1-2^{-14}$ |
| 034 | $A$ 056 | | $Z_i+1$ to $A$ |
| | | $C$ 055 | $Z_i+1$ to $Z_i$ |
| 035 | 25000 | | Skip |
| | | $U$ 027 | Start next iteration |
| 036 | $A$ 056 | | $Z_i+1=\sqrt{\mu}$ |
| | | $C$(755) | $\sqrt{\mu}$ to output |
| 037 | $A$ 024 | | Instruction in 024 to $A$ for modification |
| | | $A$ 052 | Modify to pick up next $\mu$ in series |
| 040 | $C$ 024 | | Modified instruction back to 024 |
| | | $A$ 036 | |
| 041 | $A$ 053 | | Modification of instruction in 036 |
| | | $H$ 036 | |
| 042 | $S$ 051 | | Subtract 051 from 036 |
| | | $T$ 024 | Test |
| 043 | $A$ 047 | | |
| | | $C$ 024 | Reset 024 to original value |
| 044 | $A$ 050 | | |
| | | $C$ 036 | Reset 036 to original value |
| 045 | 25000 | | Skip |
| | | $U$ 754 | Transfer control to 754 to stop process |
| 046 | 40000 | 00000 | 1/2 |
| 047 | $A$ 001 | $H$ 057 | Original setting of 024 |
| 050 | $A$ 056 | $C$ 755 | Original setting of 036 |
| 051 | $A$ 056 | $A$ 000 | Test word |
| 052 | 00001 | 00000 | $2^{-15}$ |
| 053 | 00000 | 00001 | $2^{-30}$ |
| 054 | 00002 | 00000 | $2^{-14}$ |
| 055 | | | Storage register for $Z_i$ |
| 056 | | | Storage register for $Z_i+1$ |
| 057 | | | Storage register for $\mu$ |
| 754 | 25000 | | Skip |
| | | 01000 | Stop |
| 755 to | 777, inclusive | | Storage for computed roots |

* Decimal classification: 621.375.2. Original manuscript received by the Institute, December 15, 1950; revised manuscript received July 16, 1951.

† Eckert-Mauchly Computer Corp., Division of Remington Rand, Inc., 3747 Ridge Ave., Philadelphia 32, Pa.

‡ Formerly with the Eckert-Mauchly Computer Corp. Now with the Electronic Computer Corp., 265 Butler St., Brooklyn 16, N. Y.

an instruction that occurred previously in the sequence, or one that lies somewhere ahead). An instruction which insists that this be done, unconditionally, is known as an "unconditional transfer." On the other hand, there may be a "conditional transfer" of control, whereupon *if* an equality condition is met, and only if it is met, will the computer deviate from the original sequence of instructions.

By virtue of the first transfer possibility, the Binac is capable of circling around (i.e., repeating a sequence of instructions or calculations). By virtue of its ability to transfer control conditionally, it can decide when a particular calculation is complete, and proceed to a new set of instructions. In this manner, by means of a "built-in" logic and power of decision, the Binac will perform a computation automatically.

The memory is the Binac component which stores the sequence of instructions as well as the numerical data that constitute the raw material of a given calculation, and also stores intermediate results and final data. As distinct from this, there is the computer proper, which, utilizing the instructions called from the memory, operates upon the numerical data. The device initially used to insert both data and instructions into the memory, as well as to inform the operator of the results of a calculation by printing them in visible form, is known as the "converter."

The Binac actually consists of one converter, or input-output unit, *two* computing units, and *two* memories. The memories are filled with identical programs (instructions and data), and the two units are run together and checked against each other. Any difference (which of course indicates an error) between information being handled by the duplicate units stops operation immediately. The existence within one computer of identical parallel units greatly simplifies the problem of finding the source of error, and also serves as an almost perfect checking device on the operation of the computer because of the extreme improbability of identical faults being present simultaneously in both units. (This has been proved true in practice. Many self-checking problems were run through the Binac, and identical faults never appeared.)

Although built to solve a specific problem, the Binac contains all the necessary elements for solving a great variety of problems. This will be treated further in Section IV of this paper.

## II. System Considerations

### *A. Representation of Information*

#### 1. *Electrical Representation of Numerical Data*

As mentioned earlier in the paper, the Binac utilizes the binary system of notation rather than the more familiar decimal system. One advantage of the binary system is its use of only two states: 0, 1; +, −; on, off; and so on. This system is particularly well adapted to the representation of information by the use of relays, flip-flops, binary counters, and other circuits having only two stable states. To this list may be added vacuum tubes, in general, which are either fully conducting or completely cut off. This, of course, reduces the problem of amplitude discrimination to an absolute minimum. A "one" is then represented in the computer by the presence of a pulse at some instant, the setting of a flip-flop or counter to the "one" position, or possibly by the energizing of a relay.

#### 2. *Word Structure*

A "word" is the basic unit for both instructions (or orders) and numerical quantities. Words are represented in the Binac as groups of pulses that occur at a basic repetition rate of four million per second. The pulse width is, therefore, one quarter of a μsec. A word consists of 42 pulse positions; the time required for its transmission (10.5 μsec) is known as a minor cycle. Thirty-one pulse positions contain information; the others, referred to collectively as the "space between words" (although they should be considered a part of the word), provide time for switching between the significant part of one word and that of the next.

For convenient reference, the pulse positions of a word are numbered from $p0$ to $p41$, inclusive. Positions $p0$ to $p6$, inclusive, are always unfilled.

The digits of a number, as shown in Fig. 1, are labeled $p7$ to $p36$, inclusive. The $p7$ digit is least significant, and represents the binary quantity $2^{-30}$. The $p36$ digit

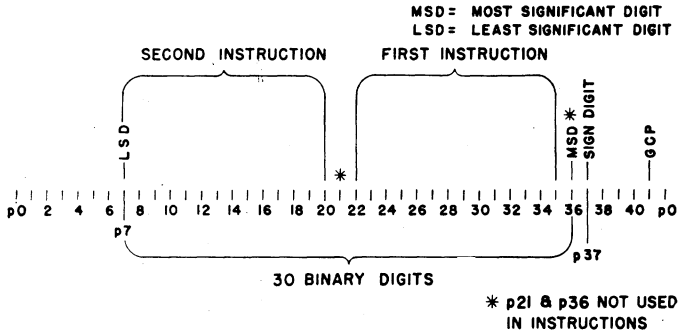



Fig. 1—Word structure.

is most significant, and represents $2^{-1}$, or $\frac{1}{2}$; similarly, the $p35$ pulse represents $\frac{1}{4}$, and so on. The least significant digit occurs first in time so that Fig. 1 indicates a number as it would appear on an oscilloscope, which is contrary to the usual positional notation used in mathematics. The $p37$ pulse indicates the sign of the number. Presence of a pulse in that position indicates that the number is negative; conversely, its absence indicates that the number is positive. The $p41$ pulse is referred to as a "gain control pulse." It has no numerical significance, existing only to stabilize the gain of the memory circuits (see Section III C4). Thus, nothing of significance exists between $p37$ and $p7$, and this area is the "space between words" referred to above.

Negative numbers exist in the Binac as complements with respect to two; in this connection, the $p37$ position may be considered as corresponding to $2^0$. It follows, then, that negative numbers will have a 1 in the $p37$ position. To represent $-\frac{5}{8}$, the sum $(2-\frac{5}{8})$ or $1\frac{3}{8}$ would be formed. The 1 exists to the left of the binary point and corresponds to a $p37$; $\frac{3}{8}$ is simply $\frac{1}{4}$ plus $\frac{1}{8}$ (i.e., a $p35$ and a $p34$). Thus, $-\frac{5}{8}$ could be written as 1.011.

The structure of instructions is shown in Fig. 1. Two general classes of instructions exist. There are those which involve transferring a word to or from the memory, and those which do not. The instruction itself may be specified by, at most, 5 binary digits. *If* the instruction involves a memory transfer, a memory position is associated with it. Since the memory holds 512 or $2^9$ words (in positions 0 to 511), 9 pulse positions are required to specify that memory position. Thus, a single instruction may consist of 14 pulse positions. On the other hand, if the instruction does not involve the memory, it will consist of, at most, 5 pulses, the other 9 positions being unfilled. The Binac uses only twenty of the possible thirty-two instructions that could be formed from the 5 digits available, and these combinations were chosen so as to minimize the number of pulses in the order group.

An instruction, then, is completely specified by, at most, a 14-pulse group. On the other hand, 31 pulse positions are available in a word. For this reason, instructions are paired, controls being arranged to execute the first and then the second instruction in the order in which they are typed.

Since instructions are stored in the memory together with numbers, the necessary gain control pulse must also be included in the $p41$ position. Since only 14 digits, at most, specify an instruction, they are separated slightly, with the first instruction occurring from $p22$ to $p35$, inclusive, the second from $p7$ to $p20$, inclusive.

### 3. *Basic Computer Cycle*

Starting with the existence of two instructions within one word, it follows directly that one step of the computer cycle must be the execution of the first instruction and the following cycle the execution of the second instruction. These cycles are referred to as $\gamma$ and $\delta$. On the other hand, since the instructions which have been stored in the memory are executed sequentially by the Binac, there are the further requirements: (a) that the pairs of instructions be called from the memory; and (b) that a count be kept, the numerical position in the sequence to be increased by one for each computer cycle.

Thus, on the alpha ($\alpha$) cycle, the numerical position of the pair of instructions *to be selected* from the memory is determined. On beta ($\beta$), the selected word is called from the memory, while the number in the memory-position counter or "control counter" increases by one. At the end of $\beta$, the first instruction in the word sets up the controls which permit its execution. On gamma ($\gamma$), this first instruction is executed, while the second instruction of the pair is set up at the end of $\gamma$. On delta ($\delta$), this second instruction is executed, the next number in the sequence being set up at the end of $\delta$ in preparation for the execution of the next alpha cycle.

This process may be modified when an instruction calls for an executed "transfer of control," conditional or unconditional. Its effect is to change the number in the memory sequence to the memory position indicated by the transfer instruction. From that point on, however, the instructions will again be taken from the memory sequentially, unless again interrupted by another transfer of control instruction. The over-all effect of this ability to "transfer control" is to add greater flexibility to the Binac, as already discussed.

## *B. Arithmetic Processes*[1]

### 1. *Addition and Subtraction*

The basic operation of binary addition utilizes the following rather simple rules:

| *Addends* | *Sum* | *Carry* |
|---|---|---|
| 0 and 0 | 0 | 0 |
| 0 and 1 | 1 | 0 |
| 1 and 1 | 0 | 1 |

In the Binac adder, the carry is fed back as an additional adder input; thus the rule is modified as follows:

| *Addends* | *Sum* | *Carry* |
|---|---|---|
| 0, 0, and 0 | 0 | 0 |
| 0, 0, and 1 | 1 | 0 |
| 0, 1, and 1 | 0 | 1 |
| 1, 1, and 1 | 1 | 1 |

As previously stated, negative numbers are represented as complements with respect to 2. To form the complement of a number, it suffices to replace its zeros by ones, and its ones by zeros, and to add a unit in the least significant position. (This is demonstrated in the appendix.)

The Binac must, of course, be capable of adding both positive and negative numbers (the latter in complement form) whose absolute value lies in the range between 0 and 1. If digits that may carry beyond the sign position are disregarded, both positive and negative numbers can be treated in the same manner. The accumulator is designed to do precisely this, carries beyond the sign digit being eliminated at the delete gate in the accumulator ($A$). For proof of the validity of this process, see the appendix.

### 2. *Multiplication*

Any multiplication is basically a series of additions separated by shifts. Multiplication in the binary system

[1] R. F. Shaw, "Arithmetic operations in a binary computer," *Rev. Sci. Instr.*, vol. 21, pp. 687–693; August, 1950.

is somewhat simplified by the fact that the multiplier digit can be only 0 or 1. Thus, additions and shifts alternate. The basic multiplication process then consists of examining successively the digits of the multiplier, starting with the least significant; of adding the multiplicand into the partial product if the digit is one; and of adding 0 if the digit is zero. After each such addition but the last (corresponding to the sign digit), the multiplicand must be added into a position more significant by one digit. This can be done by shifting the multiplicand left or the partial product right. Actually, the latter course is taken.

Since the accumulator capacity is limited to one word, this implies that of the 60 digits which would result from the multiplication of two 30-digit binary numbers, 30 digits are lost. Again, the delete gate serves the function of removing these 30 least significant digits.

Since the computer must be capable of forming correct products regardless of sign, the multiplication process must be generalized to include the cases where one or both factors are in complement form. It can be shown that two relatively simple correction steps will accomplish this; if either factor is negative, the complement of the other must be added to the result. For reasons which will become clear later (see discussion of arithmetic circuits, Section III B 4 a), the multiplier is lost during the building up of the product. The correction for a negative multiplicand is, therefore, made piecemeal as the product is built up, while the correction for a negative multiplier is made by adding the complement of the multiplicand to the product.

### 3. *Division*

To perform a division, the Binac utilizes a variation of the familiar nonrestoring method, which can be shown to be valid regardless of the signs of the operands. The method basically consists of comparing the $p37$ (sign) digit of the divisor with that of the remainder (which is the dividend in the case of the first comparison). If these digits are alike, a 1 is placed in the quotient register and the divisor is subtracted from the left-shifted remainder. If they are not alike, a 0 is put in the quotient and the divisor is added to the left-shifted remainder. This procedure is then repeated.

The consequence of this scheme is to build up a quotient of the form

$$\sum_{k=1}^{n} 2^{-(k-1)} p_k,$$

where $p_k$ is the quotient digit (0 or 1) and $k$ represents its significance; $n$, of course, is 30. It can be demonstrated that the true quotient differs from this expression referred to as the "pseudo-quotient" by the amount $(2^{-n}-1)$. Thus, the procedure in division resolves into performing 30 steps as described and making the required correction. $2^{-n}$ is then $2^{-30}$, or a digit in the least significant position. $(-1)$ can be added by simply adding a unit in the most significant, or sign, position since $p37$ acts as $-1$ in the Binac. This, incidentally, is equivalent to reversing the sign of the pseudo-quotient. As in the other arithmetic processes, any carry beyond the sign position is deleted.

### 4. *Round-off Procedure*

The procedure adopted for round-off consists of "stuffing" a unit into the least significant (or $2^{-30}$) position. That is, this digit is always made 1 by superposition beyond the adder so that no carries can occur.

This method has the advantage of not requiring the retention of the $(n+1)$st digit, which is necessary in some round-off schemes. It yields approximately the same bias as the retention method and no carries. In multiplication, it can be carried out concurrently with the correction step required where the multiplicand is negative. In division, it coincides with part of the general correction required to transform the pseudo-quotient into a true quotient.

## C. *Generalized Block Diagram of the Binac*

Sufficient introductory material has been given so that a simplified block diagram of the Binac (Fig. 2) can be presented at this point. The arithmetic units proper

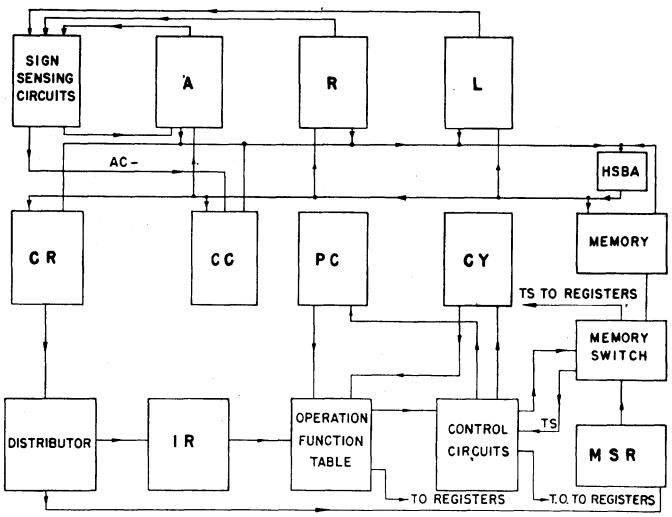



Fig. 2—Simplified block diagram of the Binac.

(by which is meant the blocks most directly involved in addition, multiplication, and the like) are the $A$, $R$, and $L$ registers. The $A$, denoting accumulator, contains an adder, and is used to compute and hold sums, differences, products in multiplication ($M$), and the dividend (subsequently the remainder) in division ($D$). The $R$ register holds the multiplier on $M$ and a "pseudo-quotient" on $D$. (The pseudo-quotient is the quotient preceding a final correction step, the correction being made in $A$.) The $L$ register contains the multiplicand on $M$ and the divisor on $D$. The memory ($My$) is also shown, and it is from this unit that the numbers operated on by the arithmetic registers are taken.

Although the memory has been treated as a single unit thus far, it actually consists of 16 separate "tanks," or channels, each channel capable of holding 32 words. A "memory switch" is used to select a desired position in the memory; it consists of two parts—a "channel selector" to select one of the 16 channels, and a "time selector" to select one of the 32 words in that channel.

The problem of selecting the next word in the memory in a sequence of instructions is handled by the control counter, previously referred to as a memory position counter. Instructions taken from the memory are converted into form suitable for controlling other circuits by means of the control register, the instruction register, the memory switch registers 1 and 2, and the memory switch tank selector, the program counter, a comparator, and the time selector counter. Also involved are the clock, the cycling unit, the cycle counter, and the high-speed bus amplifier.

The clock generates basic timing pulses at a 4-mc/s rate. The clock frequency is set by a temperature-controlled crystal, the entire computer depending on the accuracy of the clock. The cycling unit is essentially a frequency divider, generating pulses at word frequency, that is, every 10.5 $\mu$secs. These pulses occur at various points in the word. Thus, the cycling unit generates $p7$'s, $p37$'s, and the like; these are used as required throughout the computer.

The cycle counter is a two-stage binary counter whose four possible states are used to indicate whether the machine is on alpha, beta, gamma, or delta. It is assumed that a number indicating a particular memory position exists in the control counter. On alpha, this number, indicating the memory position of the next pair of instructions to be called from the memory, is transferred from the control counter to the control register and then to memory switch registers 1 and 2. As has been already indicated, this number will contain, at most, 9 digits or pulse positions. Combinations of the first 4 digits are sufficient to specify the tank called for (since there are only 16 tanks), and combinations of the last 5 digits specify the position within the tank, or time (there being 32 times or positions).

The transfer from control counter to control register is made by way of a bus or signal trunk, hereafter referred to as the "high-speed bus," which includes an amplifier (high-speed bus amplifier). Indeed, all transfers to and from the memory as well as interregister transfers proceed by this path. The existence of such a single transfer path simplifies the design of the Binac considerably. Furthermore, it provides an ideal point at which to check the operation of the two halves of the Binac. Comparison circuits, not shown in the block diagram, are connected to the outputs of the two high-speed bus amplifiers. As has been indicated, any differences between the pulse groups passing this point stop the computer immediately.

Continuing now with the problem of selecting a word from the memory, the contents of the control counter have been transferred through the control register to the memory switch registers 1 and 2. The latter consist simply of 4 and 5 "flip-flop" circuits, respectively. The first 4 digits of the number transferred then set up memory switch register 1. The term "set up" here means that a diode matrix or function table (the tank selector) is activated by memory switch register 1, with the consequence that an output, corresponding to the chosen tank, is selected. The second group of 5 digits sets up memory switch register 2. This similarly delivers an output to the comparator, the output corresponding to the position or time in the tank. With these steps completed, the alpha cycle is finished, the computer then entering into the beta cycle. This is done through the operation of the control circuits upon the cycle counter.

The beta cycle consists mainly of waiting for the selected time to arrive. The time finally selected is determined by the coincidence of the time selector counter and memory switch register 2. The former is a 5-stage continuously running binary counter whose 32 possible states correspond to the 32 words in each tank. Meanwhile, memory switch register 1 also has been delivering its output to the tank selector. Thus, the proper tank and time within that tank are chosen, and the output gate of the chosen tank opens, permitting the word to emerge from the memory. The word then passes through the high-speed bus amplifier. It feeds back through an "input gate" to the same memory position, and proceeds through the control register, setting up the instruction register and memory switch registers 1 and 2, this time in preparation for the gamma cycle. (The word is still retained in the control register, however, this being necessary for the execution of the delta cycle.) The beta cycle is now complete, the selected word having been taken from the memory.

Referring now to Fig. 1, it may be pointed out that the most significant part of the instruction pair (between $p35$ and $p22$) is first used. It is this portion of the word, then, that controls the gamma cycle. Like memory switch registers 1 and 2, the instruction register consists simply of 5 flip-flop circuits, there being one flip-flop for each digit of the instruction. The entire set of flip-flops, 14 in all, is sometimes referred to as the "static register."

On gamma, then, the instruction register sets up in a manner corresponding to the instruction itself. The function table is activated, and it selects, in turn, the various control signals required to execute the order. The control signals go to the registers involved in carrying out the instruction (see Fig. 2). One such signal is necessarily an "instruction-ending" signal, which goes to the main control unit. This signal indicates that all the required steps have been performed in executing the given instruction. The control then cycles the cycle counter from gamma to delta, initiating the next computer cycle.

It will be recalled that, at the conclusion of beta, the

word selected from the memory was retained in the control register, the first instruction of the pair being

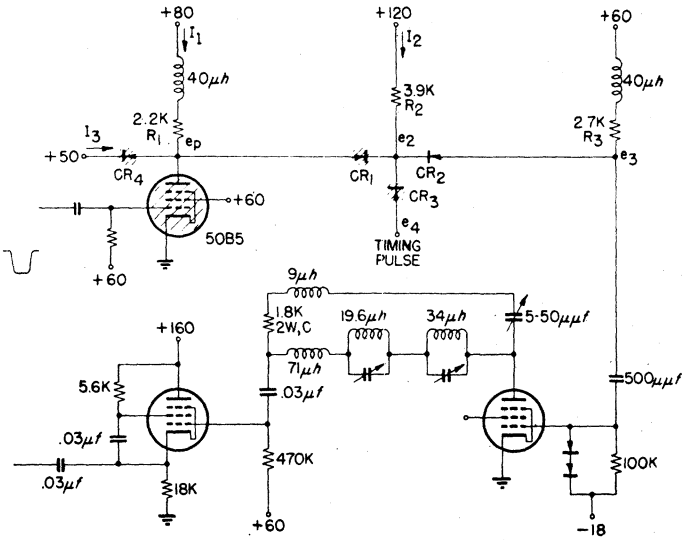



Fig. 3—Pulse former and reshaper (PFR).

executed on gamma. On delta, the second instruction of the pair is executed in exactly the same way that the gamma step was performed. The function table again delivers an instruction-ending signal so that the next alpha cycle is initiated. The basic computer cycle is thus complete. It should be noted that only on $\gamma$ and $\delta$ proper can the contents of the memory be modified. One type of "memory instruction" simply reads out the contents of a memory location, and it is retained there by being fed back through the input gate. Another type, however, inhibits this path, permitting new data to enter that location.

In the course of this discussion of the block diagram, all of the basic units, except the program counter, the synchronizing register, and the converter, have been dealt with briefly. The program counter is important only during multiplication and division. Its function, as will be seen in greater detail later, is to control the four basic steps of $M$ and the three steps of $D$. The converter and synchronizing register are used to bring original data and instructions into the memory and to remove results of a computation from the memory. Together they serve as the input-output mechanism of the Binac. It may be noted that both a typewriter and a magnetic tape are available as original sources of data as well as devices by which the contents of the memory may be recorded.

## III. Engineering Design

### *A. Basic Circuits*

#### 1. *Clock Gate and PF*

The purpose of the timing pulse gate, or "clock gate," and pulse-forming network used at numerous points in the computer is to reshape signals which have degenerated in respect to rise time and signal-to-noise ratio as a result of passing through delay lines and other circuits. The signal to be reshaped is timed (by inserting a small compensating delay, if necessary) so that the timing pulses at the clock gate fall in the center of the pulse-envelope pulses; this adjustment assures maximum gating tolerances. The output of the clock gate will be a signal having a timing pulse wherever there was a pulse-envelope pulse in the original signal. These pulses are amplified by the following tube, whose plate load is a Guillemin line. The latter is adjusted to convert each timing pulse into a flat-topped pulse having a half-amplitude width of approximately 0.25 μsec; the output of the network is, therefore, a pulse-envelope signal equivalent to the original input signal but with improved rise time and signal-to-noise ratio. Since the leading edge of an output signal corresponds to a timing pulse which was, in turn, centered with respect to the input signal, it follows that the network introduces a time delay of approximately half a pulse time. A cathode follower, to minimize capacitive loading of the network, completes the reshaping circuit. The entire circuit of Fig. 3 is represented by a block labeled "pulse former and reshaper" on the block diagrams.

#### 2. *Gates and Buffers*

The gates used for switching purposes in the Binac are, in general, crystal diode gates designed to handle negative signals and to use the circuit in Fig. 4 (a).

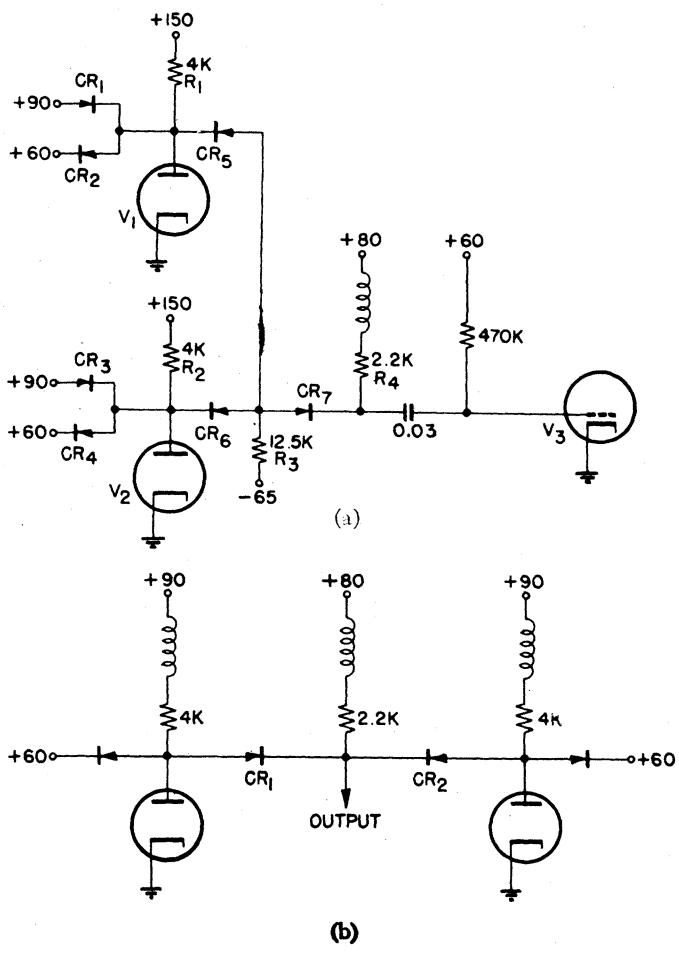



Fig. 4—Gate (a) and buffer (b) circuits.

In many cases it is necessary to supply signals to a point from various sources without having the latter react on each other. A "buffer" circuit of this sort, responsive to negative pulses, is shown in Fig. 4(b).

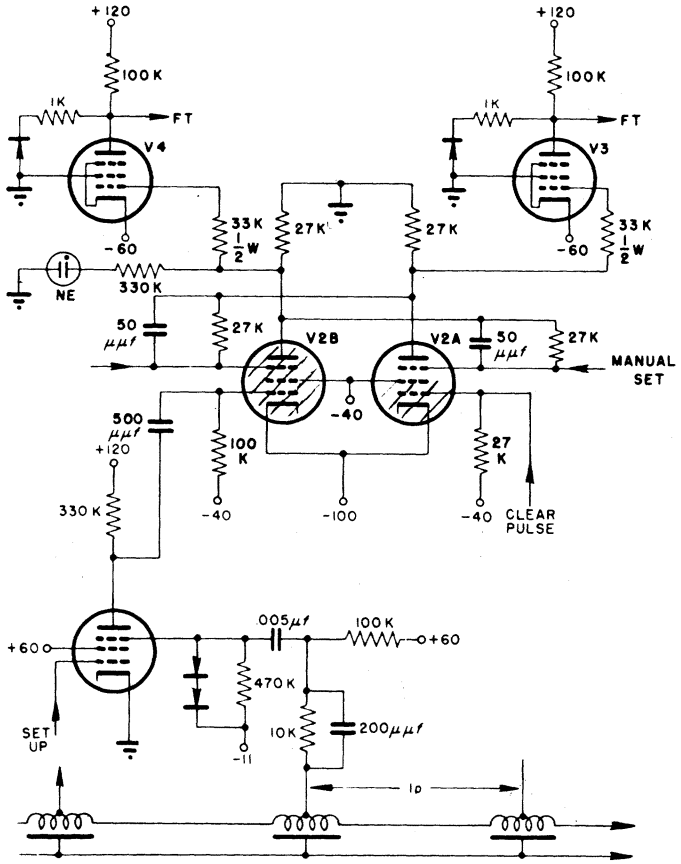



Fig. 5—Distributor and static register stage.

3. *Flip-flops, Counters*

The flip-flops used in the Binac for memory and switching purposes are, in many cases, essentially similar to those used in the ENIAC.[2] They require little comment, except to note that the static-register flip-flops (see Fig. 5 at left) use multigrid tubes, permitting the use of separate grids for internal coupling and for triggering or clearing.

Multistage binary counters are used in several places, but these also require little comment as they differ only slightly from those previously described in the literature.[3] A circuit of one of these, the cycle counter, is shown in Fig. 6.

4. *Distributor*

The distributor circuit, together with a group of flip-flops which constitute a static register, provides the means for converting information consisting of pulse patterns circulating in a delay line into patterns of steady (dc) signals more suitable for performing switching operations. A typical distributor stage is shown in Fig. 5.

Positive pulse-envelope signals are fed to an electric delay line through a pulse former and line driver. The line is tapped at points one pulse time apart. Therefore, at some particular instant the entire pattern of pulses (or unfilled pulse positions, as the case may be) will exist in such a position along the line that the voltage at each tap will be relatively positive or negative, according to whether the corresponding pulse position in the pattern

[2] A. W. Burks, "Electronic computing circuits of the E.N.I.A.C.," PROC. I.R.E., vol. 35, pp. 756–767; August, 1947.
[3] Staff of Engineering Research Associates, "High-Speed Computing Devices," McGraw-Hill Book Co., New York, N. Y.; 1950.

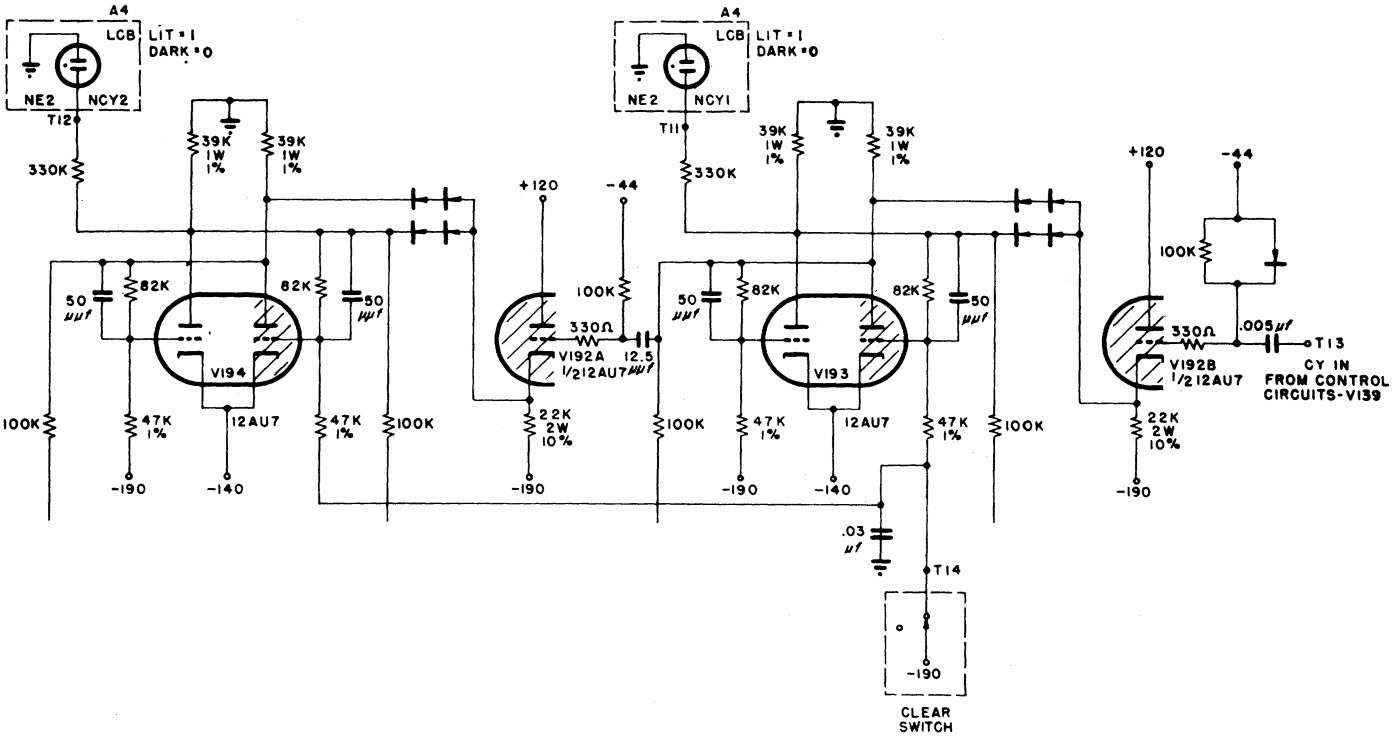



Fig. 6—Cycle counter (CY), typical of the binary counters used in the Binac.

contains a one or a zero. If the first grids of all the distributor gates are now simultaneously pulsed, those corresponding to *ones* will give output signals to set their respective flip-flops; others, receiving no signal from their respective line taps, will produce no outputs and will leave their flip-flops in the cleared position. As a result of the application of the set-up pulse to the gates, therefore, the pulse pattern is converted into a corresponding set of dc voltages. These are ordinarily amplified and applied to function tables although in the case of the time selector (see Section III B 3) they are applied to comparator gates directly (that is, without amplification).

### 5. *Asynchronous to Synchronous Devices*

Any transfer of information into the computer from the outside, whether it be numerical or program information from magnetic tape or keyboard, or simply the signal to start computing, must be timed to enter the machine in synchronism with the basic pulse rate of the computer and also at the correct time with respect to the information circulating in the delay-line registers and memory.

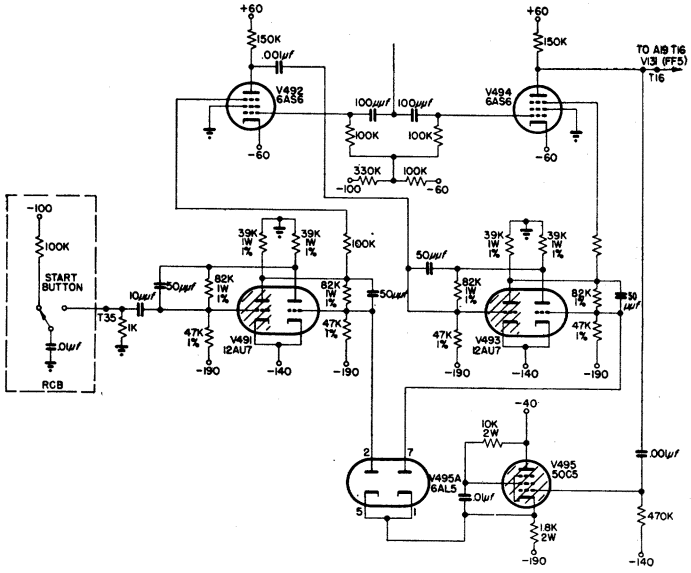



Fig. 7—Single pulse device (start circuit).

The start circuit, shown in Fig. 7, is an example of the method used to go from the asynchronous to the synchronous system.

### 6. *Shift Circuits*

It is necessary in multiplication and division to shift the data in a one-word register to the right or left, respectively. In a timed-pulse system this is equivalent to causing the pulse pattern to lag behind or advance ahead of its previous position relative to the minor cycle. A lag of one-pulse time can be produced by introducing an extra one pulse delay for a period of one minor cycle, while a lead can be produced by shortening the loop by a corresponding amount for the same period of time. A register capable of producing one-digit shifts in either direction, therefore, has a circuit like that shown in Fig. 8.

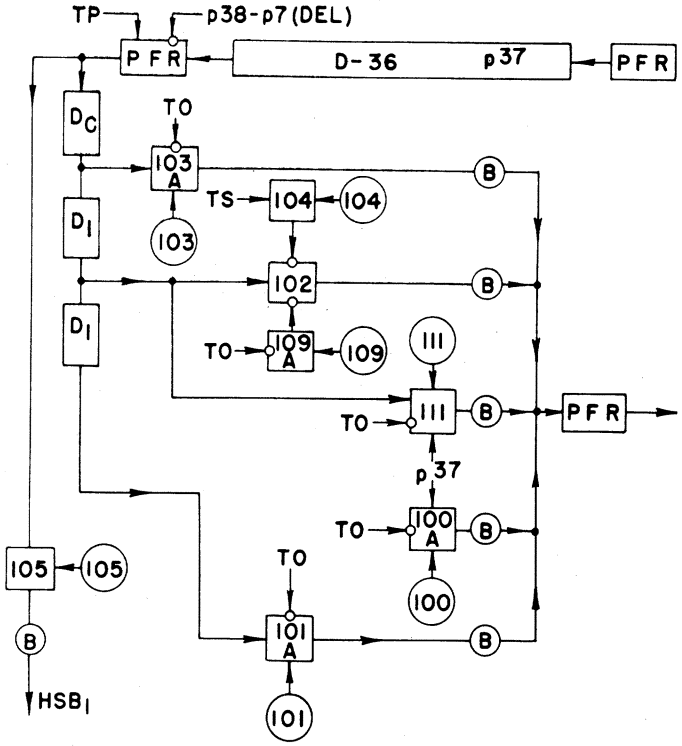
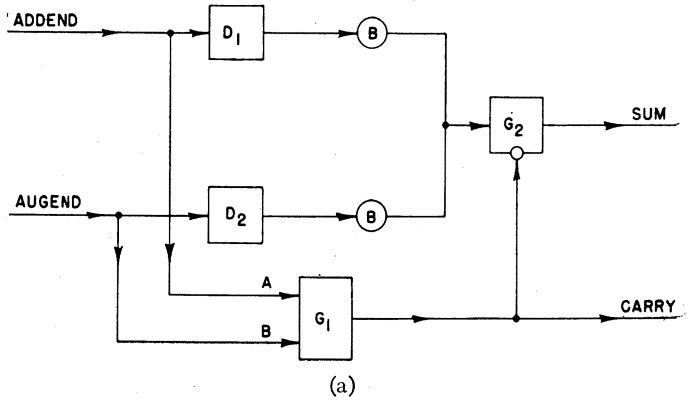


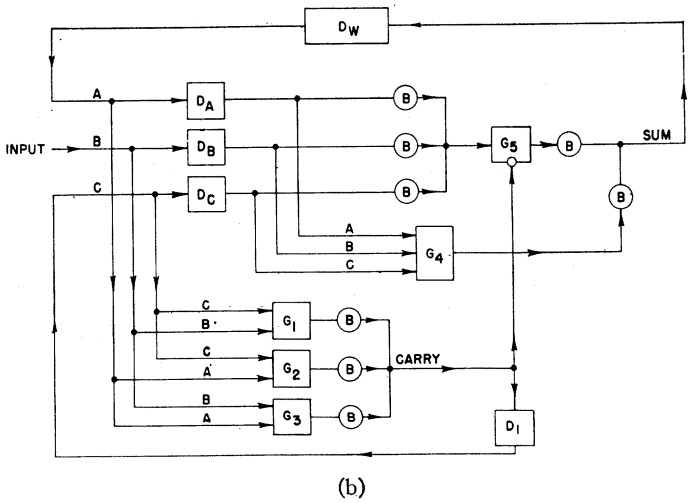

Fig. 8—Shift circuit.

### 7. *Half Adder and Adder*

In block form, a half adder may be represented as at (a) in Fig. 9 and a full or binary adder as at (b). A combination of two half adders may also be used to form



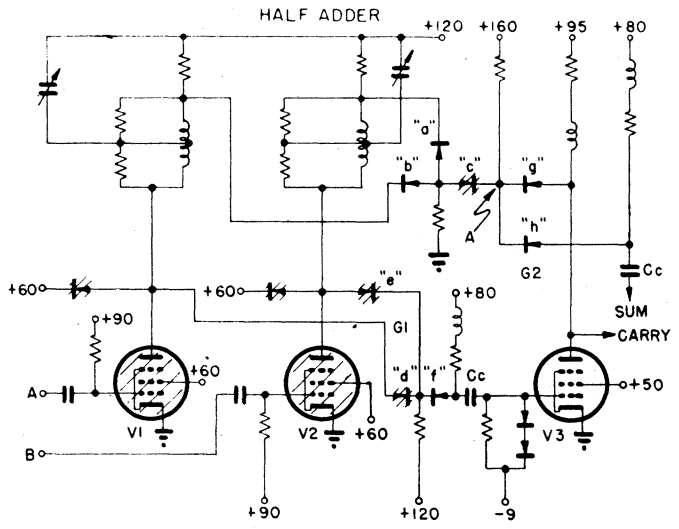

(a)



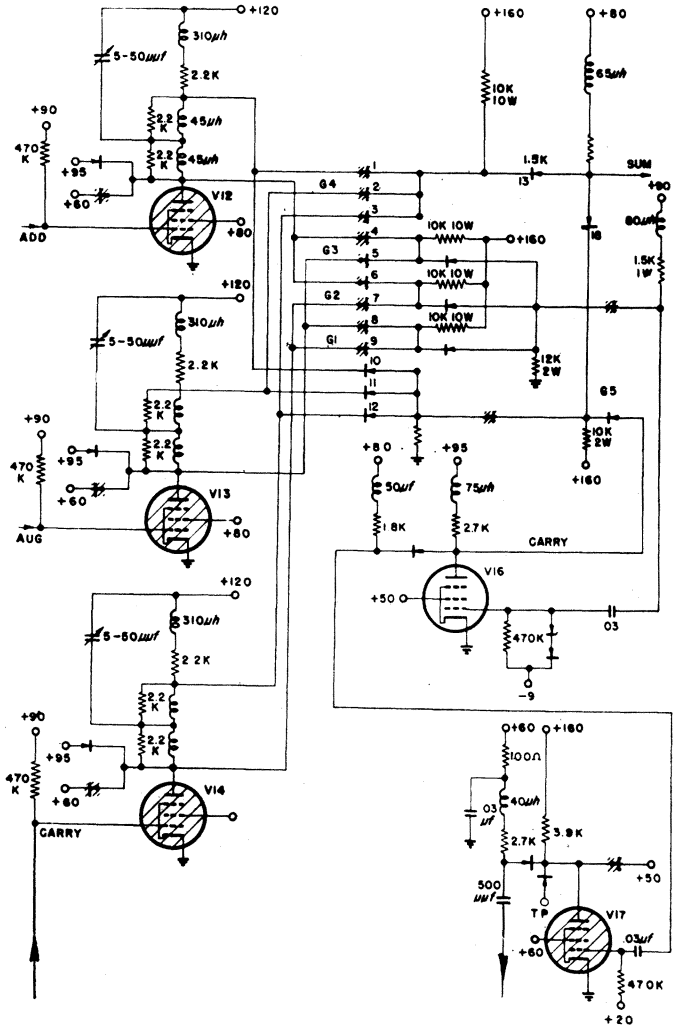

(b)

Fig. 9—Block diagram of half adder (a) and full adder (b).

an adder, but it is somewhat more economical to use the form shown. The corresponding schematics of the half adder and adder are shown in Figs. 10(a) and 10(b), respectively.




(a)




(b)

Fig. 10—Schematic diagram of half adder (a) and adder (b).

The properties required of a half adder were summarized in Section II B (1). Thus, in Fig. 9(a), if one input is present, it is buffed into *G2*, and appears on the sum line since no carry is present to inhibit this gate. If both inputs are present, the carry pulse from *G1* inhibits *G2*. Thus, the carry line is operated while no sum appears. The half adder can be used as a complementer (reversing zeros and ones) if, for the duration of the word to be complemented (which enters one input), the other input is continuously *one*. The output is then taken off the sum line. As can be readily seen, for a zero in the word entering, a 1 appears as the sum output; and for a one entering, a zero appears as the sum output.

### *B. System Considerations*

#### 1. *Synthesis of a One-Word Register*

In dealing with numbers or words, it is clear that the first requirement is a means by which the numbers may be held while manipulating them. This is done in the Binac by means of one-word registers. The general principle used is that of a recirculation loop and regeneration and retiming gate. The loop itself must contain a delay element that, together with the remainder of the loop, is sufficiently long to permit the word to exist within it. Since words are temporal sequences of pulses that appear within a 42-pulse or 10.5-μsec interval, the recirculation loop is made the same length. Any duration other than 10.5 μsec, or an exact multiple of 10.5 μsec, would result in the word precessing around the register, a fact which is employed in designing shift circuits, as previously described. Fig. 11 represents in block form a one-word register. *D*38 is the major delay element. The delay found in the complete loop, plus D38, of course, must equal 10.5 μsec.

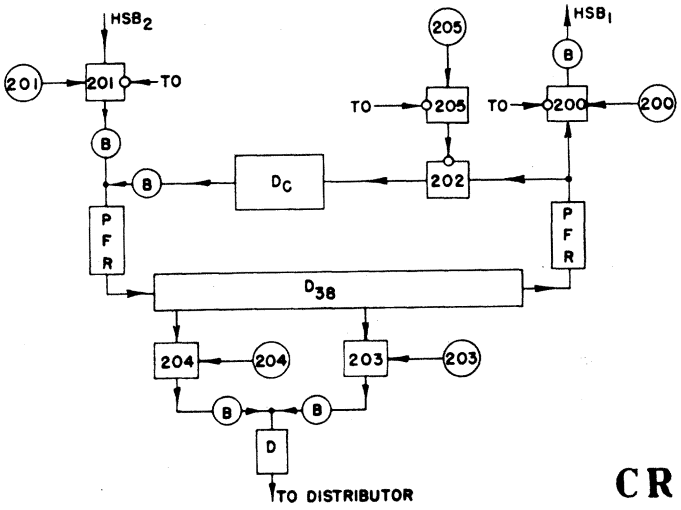



Fig. 11—Control register (CR), a typical one-word register.

To maintain this delay exactly and, also, to maintain the word in usable form is manifestly impossible using *D*38 alone. In the first place, the one-word delay element (which in the Binac is a lumped constant electric delay line using some mutual coupling between sections) is

subject to temperature drift. Even with temperature control of the lines, over a period of several hours precession of the word would be unavoidable. In the second place, the delay line both attenuates and distorts the signals. Thus, after a few circulations, the information would have deteriorated in both amplitude and shape to the point of unusability.

To eliminate the amplitude loss, the gain around the loop must be made to equal unity. Any other value means loss of the information—either because it deteriorates into random noise when the gain is less than 1, or builds up into unusable oscillation when the gain is greater than 1. The maintenance of unity gain must then be accomplished by adding precisely the required energy to the circuit by means of amplifiers and associated circuits external to *D*38. These elements are represented by the pulse former and reshaper (PFR).

Were achieving unity gain the only consideration, the block pulse former and reshaper could assume quite a variety of forms (e.g., amplifiers with swings limited on both sides). The deterioration of wave shape could be avoided by a variety of reshaping circuits. However, these alone would not prevent precession of the word; retiming as well as amplification and reshaping must be provided. The pulse former and reshaper is a clock gate and pulse former of the type previously described. The timing pulses are generated by a highly accurate "clock" whose frequency deviation is less than 20 parts per million. Since timing pulses are common to the entire computer, the effect of making timing pulses one of the gate inputs is to keep the signals in synchronism with those in other parts of the computer. The signal, in short, may be somewhat out of step at the input to the gate, but will be precisely in synchronism with the basic clock at the output.

In this paper a buffer circuit will be denoted by the symbol shown in Fig. 12(a). A gate will be denoted by the symbol of Fig. 12(b), with either the letter *G* or a gate number in the square.

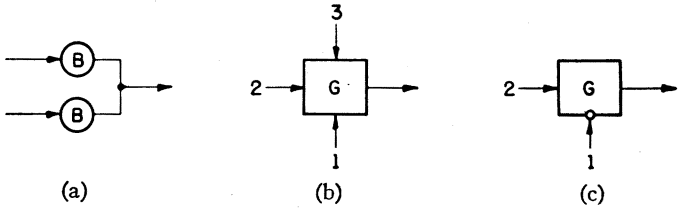



Fig. 12—Block-diagram symbols: (a) buffer; (b) gate; (c) gate with inhibitory input.

Further extending the gate symbology, an inhibitory input to a gate will be denoted by a circle (see input No. 1 of Fig. 12(c) ). Signal (2) normally passes through *G*, but when (1) is present, it inhibits the passage of (2). Actually, there is no real difference between the inhibiting gate and the normal gate. It is still an "and" circuit. The symbol indicates that when signal (2) *and* the lack of signal (1) exist simultaneously, an output is produced.

It is possible to combine gates and buffers to serve as a means of getting information into the one-word register. The inputs to the gate *G*201 are the desired information and a control signal (201), which is applied when it is desired to put the information into the register. The original information appears at the output of *G*201, and is read in through the buffer, whereupon the control signal (201) is removed. Thereafter, it continues along the recirculation loop, meets the second buffer, but continues around again through *D*38, the path through the original buffer being closed. The information is thus trapped in the register.

To take information out of the register, a similar device is used. Signal (200) must now be applied when information is to be read out of the register. In the case of reading out, if it is desired at the same time to retain the information, the pulse former and reshaper circuit must include an amplifier capable of driving both the output gate and the next stage of the normal recirculation path.

Actually, the above is still an incomplete description of a one-word register since an essential timing detail has not yet been considered. All information moves between registers by way of the high-speed bus amplifier. The high-speed bus amplifier circuits have finite delays since they are required to reshape and retime. Thus, information coming out of the high-speed bus amplifier will lag behind the information coming in. It follows from this that were information to be transferred from one register to another without compensating for this delay it would be displaced correspondingly in the transfer process. For this reason a delay $D_c$ equal to that of the high-speed bus amplifier system is placed between the input and output gates. Outgoing information is then read out at a point in time earlier than that at which information is read in. Thus, after reading from some Register I to an identical Register II, at corresponding points in these registers identical pulse patterns are observed that are not displaced in time relative to each other. Mathematically, this means that the significance of the digits has not been changed —the sign pulse, for example is still retained in the sign position. Since the over-all loop delay must still be one minor cycle, $D$ is correspondingly shorter by the amount $D_c$ or $D38+D_c+$circuit delay $=1$ minor cycle.

Other possibilities remain. Thus, it is ordinarily necessary to erase the old information in the register when new information is entered. In this case an inhibitory gate, *G*202, would be put in series with the recirculation loop. In the case considered, then, signals (201) and (205) operate together. The information into *G*202 is lost and information arriving at *G*201 is gated into the loop. If, simultaneously, *G*200 is operated, the original information would be read out. This, therefore, cannot be done if the information arriving at *G*201 is coming from another register since the two signals would be superimposed on the high-speed bus. In the case of the memory, signals corresponding to (201), (200), and (205) are all operated together in order that the tube driving *G*200 and *G*202 may never have to drive both

simultaneously; thus, when reading out of the memory, the high-speed bus amplifier forms part of the recirculation path. In this case, the one-word register receiving the signals from the memory has only gates corresponding to $G201$ and $G205$ operated. When reading *into* the memory, a gate between the memory output and the high-speed bus amplifier is inhibited in order to clear out the previous contents of the memory.

A variety of circuits are used to operate gates and buffers, drive the delay lines, and do other jobs. These, and in particular the delay $D_c$, make necessary another retiming operation. A typical register will, therefore, have more than one pulse former and reshaper. The main delay line $D$ may be tapped to provide output signals shifted in time with respect to the usual reference point (which, in the Binac, is the clock gate of the high-speed bus amplifier).

### 2. *Clock and Cycling Unit*

The timing pulses used at the clock gates are positive pulses of approximately 28-volts amplitude and 0.06-$\mu$sec duration at a 4-mc per second repetition rate. They are generated by a crystal oscillator using a type 807 tube and employing a temperature controlled 4-mc quartz crystal. The oscillator output is coupled to the grid of a Class C amplifier, also using a type 807 tube. The plate current pulses are coupled through a pulse transformer to a load consisting of the parallel inputs of eight resistance-terminated 93-ohm coaxial cables which carry the signals to all parts of the computer.

The cycling unit (Fig. 13) has several features which differ from those of the sample register just discussed.

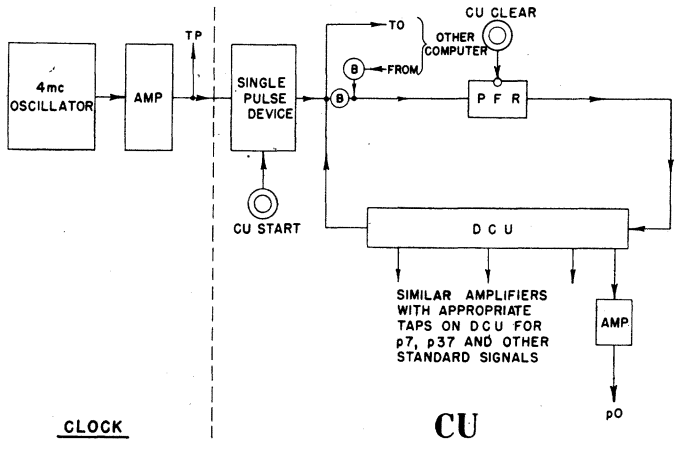



Fig. 13—Clock and cycling unit.

The only input to the register is a single pulse introduced by operation of the cycling-unit start button. Depressing the start button causes one of the clock pulses to trigger a flip-flop whose output is differentiated and shaped by means of a line-pulse former to produce an accurately timed signal input pulse for the loop. This pulse continues to circulate until interrupted by operation of the cycling-unit clear switch.

To obtain control pulses at various times during the minor cycle, signals are taken from taps along the cycling unit delay line. These signals are, in most cases, used to gate timing pulses which are fed into pulse formers, followed by amplifiers as shown. In two cases, the error gate and the delete gate, signals of several digits duration, are needed. For these signals, two taps on the register line are used, the two timing pulses gated being used to set and reset a flip-flop. The signal from the flip-flop plate provides the desired gating signal. For those cases where the timing of the desired signal is less critical, the output of the register delay-line tap was used without gating of a timing pulse. An example of such a signal is the $p0$ signal used for oscilloscope synchronization during test and maintenance operations.

### 3. *Control Circuits*

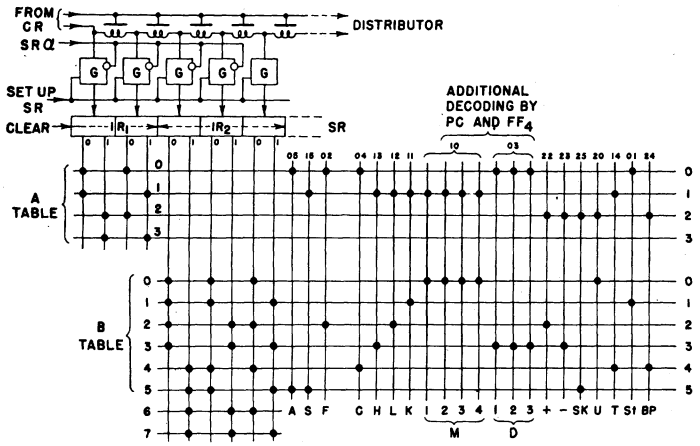
The basic operation cycle of the Binac was outlined in Section II A 3. A two-stage binary counter, the cycle counter (see Fig. 17), controls the four basic cycles; its decoded outputs are used to excite lines in the encoding function table which actuate the gates necessary for the required operations.

The control register has already been mentioned as an example of a one-word register. Its purpose is to hold the second instruction of a pair until the first has been completed; although it would be possible to return to the memory to bring out the second instruction directly into the static register, as is done with the first, the necessity for setting up the memory location of the instruction in the static register and the latency delay make this a time-consuming operation. Therefore, on the $\beta$ cycle the pair of instructions is brought into the control register, and the first instruction set up in the static register after passing through $G204$; meanwhile, the entire word continues to circulate in the control register until cleared out at the time the next pair of instructions is brought in.

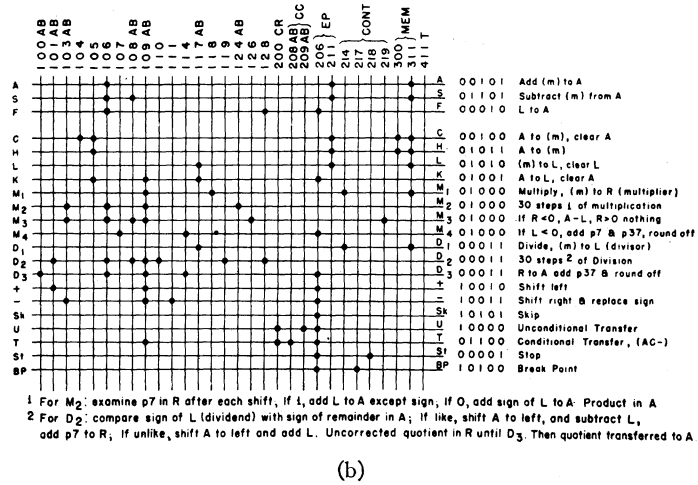
Pairs of instructions are ordinarily taken from the memory in numerical order; to accomplish this, a one-word register, the control counter, containing a unit adder (see Fig. 9(c)), is used to keep a record of the location from which the next pair of instructions is to be taken. As the word is read out of the specified location on the $\beta$ cycle, a unit is added to the number in the control counter so that on the next $\beta$ cycle the pair of instructions in the next memory location will be read out. Instructions from the control register are converted into static form by means of a distributor and static register.

For convenience in testing, push buttons are provided for setting up manually any desired combination in the static register flip-flops. Another push button can clear all flip-flops, while a different button can clear only the nine representing the memory location. All 14 flip-flops are cleared automatically by the ending pulse, which occurs at the completion of every instruction.

Of the 14 binary digits in an instruction, 9 are necessary to specify the memory location with which the instruction is concerned, leaving only 5 for the instruction proper. Since the number of gates employed by various instructions is much larger than 5, it is impossible to have a one-to-one correspondence between instruction digits and gates. The instruction must first be decoded into one of the 32 possible combinations represented by the 5 pulses (actually, not all the 32 combinations are used); this signal can then be used to operate the proper combination of gates. Electrical function tables are used for the decoding and encoding. Fig. 14(a) shows the decoding table, each dot representing a crystal diode. The output lines, which in turn form the inputs of the encoding table, are labeled at the top with their numerical equivalents, expressed in octal form, and at the bottom with alphabetic symbols, which are used for convenience in programming. Thus, for example, the instruction calling for addition is actually typed into the computer as the combination 05 followed by the memory location of the addend; in programming, however, it is more convenient to designate this operation by $A$, subtraction (15) by $S$, division (03) by $D$, and so on.




(a)




(b)

Fig. 14—Function tables: (a) decoding; (b) encoding.

The function table is so extensive that its wiring capacitances, and those of its output circuits, result in time constants considerably longer than the space between words. It can be shown that in a computer of the Binac type, where the latency time of the memory has an important bearing on the over-all speed, it is more economical to allow a full minor cycle for switching operations between instructions than to increase the space between words, and consequently the length of the minor cycle. A so-called "time-out" period is, therefore, introduced at the beginning of each operation requiring a change in the function-table signals. Some type of fast switching circuit, capable of operating in the space between words, is still needed to prevent gating of parts of words during the time the function-table signals are changing; but the "time-out" principle permits the burden of fast switching to be shifted from the numerous function-table signals to a single flip-flop which can furnish inhibitory signals to any gates which require them.

This "time-out" flip-flop (see Fig. 17) is set by the same $p0$ pulse which clears the static register in preparation for setting up a new instruction. It is also set by certain pulses into the program counter, as described later. Since "time out" is normally required for only one minor cycle, the "time-out" flip-flop resets itself by gating the next $p0$ pulse into its reset input.

Selection of one of 16 tanks in the memory is accomplished by means of a small function table which decodes the four most significant digits of the memory location specified in an instruction set up in the static register.

The 5 remaining binary digits in the memory location number specify one of the 32 words stored in the tank. Fig. 15 shows the time-selector circuit which selects the desired word. A 5-stage binary counter, the time-selector counter is driven by $p0$ pulses from the cycling unit. Since this is a scale-of-32 counter driven by minor cycle pulses, there will always be a one-to-one correspondence between the reading of the counter and the word appearing at the gates of the memory tank. Thus, if the time-selector counter, by comparison with the memory location set up in the static register, is used to control the storing of a given word in the memory, it will always be possible to extract that same word by setting up the same number in the static register and comparing it with the time-selector counter to control the operation of the output gate of the memory.

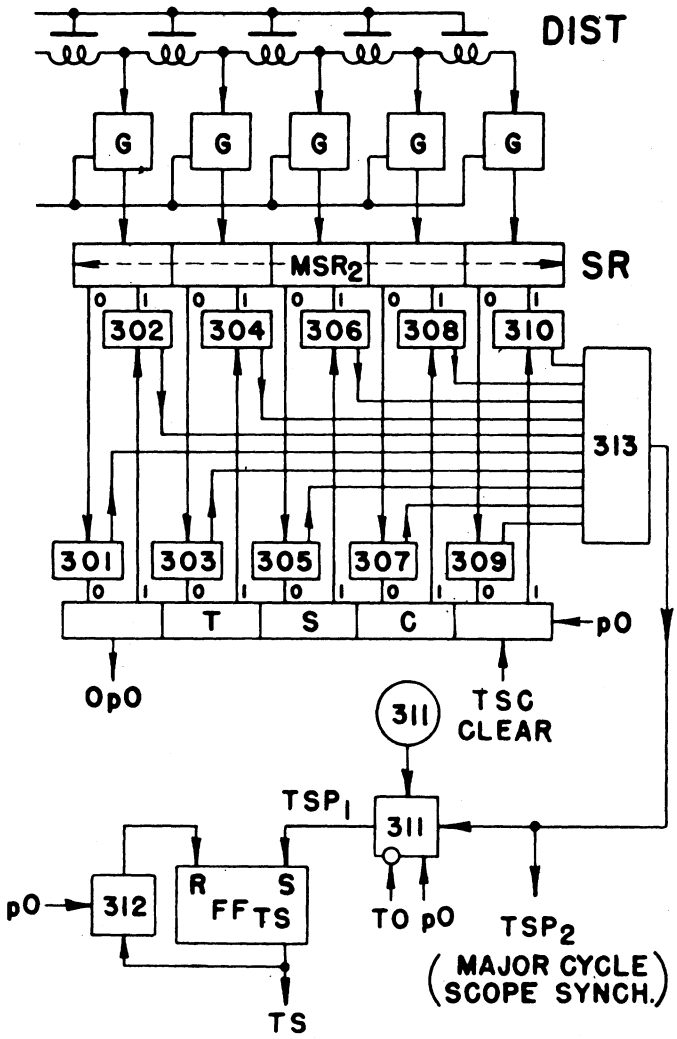



Fig. 15—Time selector (TS) circuits.

#### 4. *Arithmetic Equipment*

Section II B discussed the general arithmetic processes. Multiplication and division involved a series of repetitive additions (or subtractions) and shifts, followed by required corrections. In analyzing these operations, it is convenient to divide multiplication into four steps (designated $M1$, $M2$, $M3$, and $M4$) and division into three ($D1$, $D2$, and $D3$). The steps of multiplication are discussed individually with reference to the block diagram, Fig. 16.

*a. Multiplication:* The $M(m)$ instruction, together with another order, is transferred from the memory on the $\beta$ cycle. It sets up in the $IR$ on $\gamma_{TO}$ or $\delta_{TO}$, depending on whether it is the first or second order in the order pair. The multiplicand is assumed to be stored in the $L$ register as a result of some previous operation.

The program counter ($PC$) is used to subdivide $M$ and $D$ into their individual steps (see Fig. 17, the following page). It is a 6-stage (scale of 64) binary counter with its own decoding function table. The outputs of this table are connected into the main function table in such a way that when the $M$ line, for example, is excited by $SR$, $PC$ will determine whether $M1$, $M3$, or $M4$ is excited. Since $M2$ lasts for 30 cycles, it is more economical to allow a flip-flop ($FF4$) to take over the controlling function rather than to decode all 30 combinations of $PC$ and buff them together. With this arrangement, $FF4$ is set at the end of $M1$, and it is then only necessary to decode $PC$'s output on the 29th of the 30 repetitive steps to get a reset signal for $FF4$. A similar procedure is used on $D2$.

It will be noted that position 1 of $PC$ corresponds to the binary combination 111111 (decimal 63). By clearing $PC$ to this state rather than to 000000, carry-overs are not produced by the clearing operation, and a narrower clear pulse (a gated $p0$) can be used. Thus, $M1$ corresponds to $PC$ 63, and $M2$ starts on $PC0$ and ends on $PC29$; $M3$ and $M4$ correspond to $PC30$ and $PC31$, respectively.

The $M1$ step is used to select the multiplier and transfer it from the memory to the $R$ register. The product is to be built up in $A$. At this point, then, it is convenient to clear both $A$ and $R$. Clearing of $A$ and $R$ takes place on the first minor cycle following $\gamma_{TO}$ or $\delta_{TO}$, as the case may be. $M1$ selects $FT$ signals $109A$ and $109B$, which inhibit $G102$ in $A$ and $G115$ in $R$, respectively, when the machine goes off time out. $FT$ signal 311 is also selected to cause reading out from the memory of the word selected by the memory location part of the multiplication instruction. Since $FT$ signal 118 is activated on $M1$, the multiplier enters $R$ through $G118$. $G214$ is also energized on $M1$ during the $TS$ period. Thus, the same $p0$ that resets $FF_{TS}$ passes through $G214$ to set $FF4$ and $FF_{TO}$, and step $PC$ to $PC0$. This puts a $TO$ between $M1$ and the $M2$ step to follow. $FF4$ is used

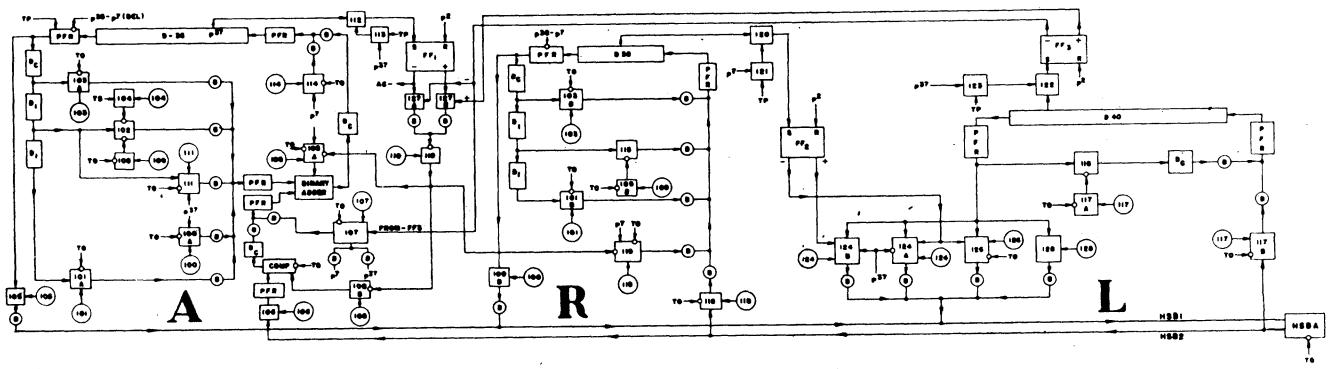



Fig. 16—Arithmetic circuits (A, R, and L registers).

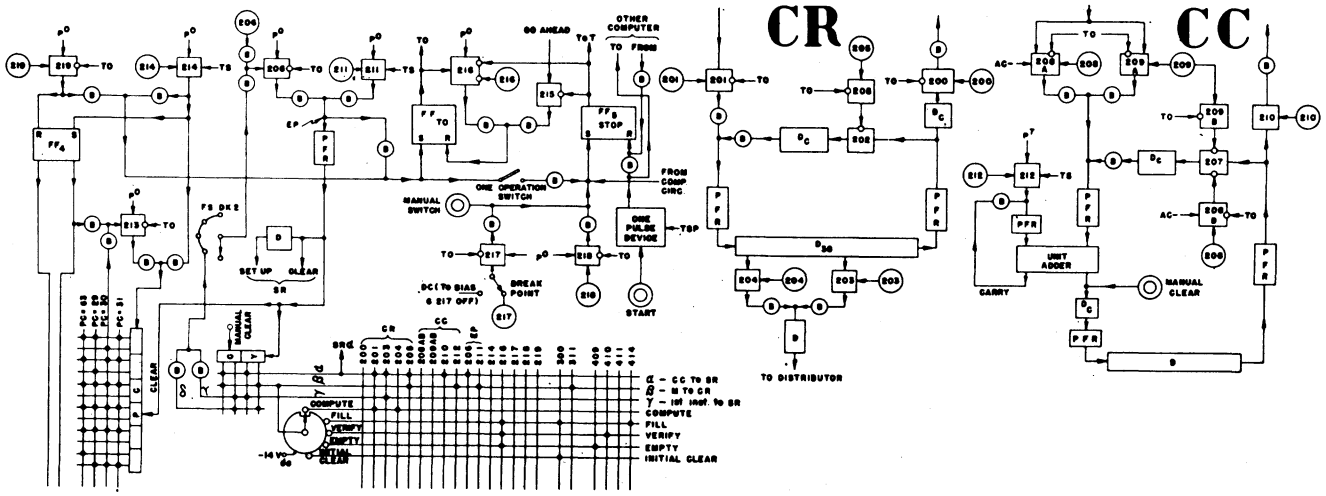



Fig. 17—Control circuits.

to control the next 30 minor cycles of *M2*. During *M2*, *PC* must keep track of the operation by counting, which it now can do by way of *G*213 (which permits *P*0's to pass, under the control of *FF*4, as soon as the *TO* inhibition is removed).

With the multiplicand and multiplier in *L* and *R*, respectively, the 30 repetitive steps are begun. In accordance with the mathematical requirements discussed in Section II B, the digits of the multiplier are examined, starting with the least significant. For a zero, only the sign digit of the multiplicand is transferred to *A*; for a 1, the multiplicand, without its sign, is transferred to *A*. A right shift in both *A* and *R* follows each transfer, permitting examination of the next multiplier digit and proper positioning for the next addition of the multiplicand. At the conclusion of 30 steps, *PC* is in position 29. The *PC*29 signal excites *G*219, which gates a *p*0 to reset *FF*4 and set $FF_{TO}$, conclude the *M2* cycle, and initiate the *TO* before *M3*.

Examination of the block diagram shows that on *M2* the *FT* signals 103A, 103B, 106, 109A, 109B, 124A, and 124B are activated. These, in turn, activate *G*103A and *G*103B (opening the right shift paths in *A* and *R*); *G*106 (permitting the transfer into *A*); *G*109A, and *G*109B (which inhibit the normal recirculation paths in *A* and *R*); and *G*124A and *G*124B (permitting read out of the multiplicand from *L* under the control of *FF*2 and *p*37). *FF*2 is set only if the *p*7 digit in *R* is a 1, in which case *G*124A operates, the *p*37 which may be in *L* being inhibited. Reset pulses are fed to *FF*2 every minor cycle, but the tap on the *R* line is so adjusted that *FF*2 is immediately set again if a pulse is present in the *p*7 position. If *FF*2 remains reset (i.e., if the *R* digit is 0), *G*124B is operated and only the *p*37 or sign is added into *A*.

On *M3*, a correction is made if the multiplier is negative. In this case, the multiplicand is subtracted from the uncorrected product in *A*. Note that the *M3* line produces *FT* signals 103A, 103B, 106, 108A, 108B, 109A, 109B, 126, and 219; 108B operates *G*108B, activating the complementer. Thus, the correction term from *L* has its ones replaced by zeros and its zeros by ones. In conjunction with this operation, 108A simultaneously operates G108A, which adds in the *p*7 required to complete the subtraction of the multiplicand. The multiplicand itself leaves *L* via *G*126. The remaining gates operate as described before. Note that *G*126 is under the control of *FF*2. Thus, *only* if *R* is negative (its sign digit has finally been shifted into the *p*7 position and can, therefore, control *FF*2) does *G*120 pass a signal which sets *FF*2 and permits *G*126 to open. *G*219 sets $FF_{TO}$ and steps *PC*, permitting the *M*4 step to be initiated.

The *M*4 cycle is used to correct the partial product if the multiplicand is negative. A partial correction has already been made in this case during *M2* since for multiplier digits equal to 0 the negative sign of the multiplicand (if present) was added into the partial product via transfers through *G*124B. The remaining step then is to add in a *p*7 and *p*37 to *A*. Simultaneously, it is convenient to round off the partial product by stuffing in a *p*7, as described in Section II B 4.

The *M*4 line activates *FT* signals 107, 114, and 206, these signals in turn operating gates 107, 114, and 206. *G*107 operates under the additional control of *FF*3. This, in turn, is set through *G*122, which receives its input from a tap on the line of *L* in such a position as to sense there the presence of a *p*37. Thus, the *p*7 and *p*37 correction in *A* is made only if the multiplicand is negative. *G*114 performs the round-off operation by superimposing a *p*7 on the partial product in *A*. With the completion of the round-off step, the final product is now held in *A*. Since *G*206 is operated, an ending pulse is passed which sets $FF_{TO}$, clears the static register, and advances *CY* to the next cycle ($\delta$ if $M(m)$ were performed on $\gamma$, and $\alpha$ if it were performed on $\delta$). At the conclusion of the *M* operation, then, the final product is in *A*, the multiplicand is still in *L*, but by reason of

the continuous right shift in $R$, the multiplier has been cleared out.

*b. Miscellaneous Operations:* Although the Binac ordinarily obtains instructions from successive memory locations, as controlled by the control counter, it is frequently necessary to depart from the normal sequence and obtain a pair of instructions from a location specified in a "transfer of control instruction." This is accomplished by reading the contents of the control register into the control counter. If the transfer instruction is placed in the right-hand position of the word, its memory position falls into the same place in the control counter as the number which was being built up there. On the next $\alpha$ step, the new location can be read back into the control register and the static register, and the desired pair of instructions brought out of the memory on $\beta$. Furthermore, the new number remains in the control counter, having a unit added to it on each $\beta$ step, and thus a new sequence of operations is started, continuing until another transfer of control instruction again changes the number in the control counter.

The transfer can be unconditional (the *Um* instruction), replacing the number in the control counter, as described above, or it may be made conditional on some situation in the computer (the *Tm* instruction). The conditional transfer allows the choice of two courses of computation to be made, dependent on the result of a previous computation. The sign of the quantity in $A$ was chosen as the determining factor in the *Tm* instruction. If the quantity in $A$ is positive, the sign sensing flip-flop prevents the reading of the new number from the control register into the control counter, and permits the original contents of *CC* to remain; but if the quantity in $A$ is negative, the operation is the same as that of a *Um* instruction.

Several instructions transfer data from one register to another; these include *Cm*, *Fm*, *Hm*, *Km*, and *Lm* (see Appendix E). The skip is generally used to fill in the left-hand position ahead of a *T* or *U* instruction where the last preceding instruction does not fall in this position. The break-point instruction can be inserted at strategic points in a program; it causes the computer to stop at each of these points if the break-point switch is set—an aid in checking new programs. With the break-point switch set to "continuous," break-point instructions are treated as skips. The stop instruction, as its names implies, stops computation by setting *FF*5, which in turn blocks *G*216 and prevents $FF_{TO}$ from being reset.

For testing and maintenance, and for checking new programs, it is essential to be able to cause the computer to perform a single operation and stop, continuing with the next operation only after manual operation of a start button. The "time-out" feature furnishes a convenient means for achieving this. Since all significant gating operations, including, in particular, the gating of the ending pulse, are inhibited during "time out," interrupted operation can be achieved by inhibiting the automatic resetting of the time-out flip-flop (through *G*216) and by providing a circuit which produces a single $p0$ pulse to reset the flip-flop when the start button is pushed. A "stop" flip-flop (*FF*5) is used to inhibit *G*216, and is set by the same pulse which sets the time-out flip-flop. But this occurs only if an operation control switch is set to "one operation," if a stop instruction is encountered, or if a break-point instruction with the break-point switch is set to "break" position. The start button is used to reset the stop flip-flop.

The discussion up to this point has referred to the Binac as if it were a single computer. Actually, the entire computer, with the exception of the input-output converter, is duplicated. The two computers are started in synchronism, and computations proceed in parallel, with a half adder in each computer making a continuous comparison of signals on the high-speed buses. Any discrepancy will produce a signal at the "sum" outputs of the half-adders; these signals are used to set the stop flip-flops of both computers. An additional check is provided by a gate circuit supplied by a cycling unit signal which corresponds to the space between words; if any signals in these pulse positions appear on the high-speed bus, the gate produces an output which sets the stop flip-flop.

### 5. *Input-Output Equipment*

Since the Binac is designed for mathematical operations in which a relatively small amount of input data forms the basis of extensive computations, which, in turn, produce a relatively small volume of output data, very simple input and output facilities are sufficient. These consist of a keyboard, a printer, a manually controlled magnetic-tape reading and recording device, and a converter for transferring data between the Binac memory and the other units.

The keyboard has eight keys representing the octal digits 0 to 7, inclusive; numbers (including instructions, which are in numerical form as previously described) typed on the keyboard are recorded by the printer as they are typed, forming a visible record for checking. A word is not transferred to the memory until its last character is typed, giving the operator an opportunity to proofread. If it is necessary to retype a word before transferring to the memory, an erase key clears it out. It is also possible to print out the contents of the memory.

A one-word synchronizing register forms the link between the converter and the main Binac memory. In typing in from the keyboard, each key stroke produces a signal corresponding to three binary digits; this signal sets up the corresponding combination in a group of three flip-flops. A transfer pulse, generated by a pulse circulating in another one-word line having a shifter network, transfers the signal from the flip-flops to the synchronizing register. Each successive group of three pulses is inserted in its proper place in the word because the pulse in the timing register, which generates

the transfer pulse, is shifted three pulse times after each transfer. When the word in the synchronizing register is completed (by typing its last octal digit), it is transferred to the main memory; the control counter controls the transfers to consecutive memory locations.

The above procedure is reversed when printing the contents of the memory; in this case words from consecutive locations are transferred to the synchronizing register, and from there are transferred one octal digit at a time to flip-flops which control the printer magnets.

The printer is an electric typewriter whose keyboard has been removed. It has been fitted with solenoid actuators for the characters 0 to 7, inclusive. An automatic carriage return is provided to operate after an integral number of words; once the "emptying" operation is initiated, the entire contents of the memory are printed out automatically.

Because most problems require a considerable amount of data (instructions and initial or boundary values) in the memory, the typing operation can become laborious. An accident, such as power failure, can destroy an hour's work in a moment; furthermore, it is frequently found that numerous computer runs of a particular problem require the same input data, with only very minor modifications which can be easily made after a standardized routine has been entered in the memory. For these reasons it is customary, after typing in a set of data, to record the latter on tape. The process is similar to that of printing out the contents of the memory, except that the recording heads on the tape equipment are connected in place of the printer magnets. The data so recorded can then be read back into the memory at any time, again using the converter, but setting up the synchronizing flip-flops with signals from the head amplifiers rather than from the keyboard.

### 6. *Memory*

The Binac memory is of the mercury delay-line type, and was described in a previous article.[4] It consists of sixteen channels, each storing 32 words. An additional channel is used for the temperature-control circuit. A memory-channel recirculation circuit is quite similar to an electric delay-line register, with the line amplifier replaced by a wide-band amplifier and detector, and the line driver operating directly out of the clock gate without the use of a pulse former (since the mercury delay line and following amplifier broaden the pulses into a pulse-envelope signal).

## IV. Applications

The Binac is suitable, within the limitations imposed by the size of its memory and the nature of its input and output operations, for the solution of a wide variety of mathematical problems. These must, of course, first be reduced to suitable form for solution by numerical methods. Solutions of Poisson's equation in two dimensions for various boundary shapes and load conditions were obtained during the testing period, and results indicated that the Binac can effect a very substantial saving in time for the solution of such problems as compared to solutions by hand computation using desk-type calculators.

The program in Table I,[5] showing the computation of the square roots of a series of 20 numbers, is presented here to illustrate the method of coding problems for the Binac. All numbers are expressed in octal form. The first column shows the memory location in which each pair of instructions is stored; the next two contain, respectively, the first and second instructions in the word. A brief description of the operation appears in the fourth column. Before typing the program into the computer, the letters in columns 2 and 3 are replaced by numerical equivalents ($05 = A$, $04 = C$, and so on). After clearing the memory and clearing the control counter to zero, the program is typed in, after which the control counter is again cleared to zero. Computation is now started, and will stop automatically when the 20 roots have been computed. At this time the control counter will read 754. The computer can then be switched to "empty" so that it will start printing with memory location 755, the first root. Printing will continue until all 20 roots have been printed, after which the operation stops.

In the program shown, 000 contains a skip followed by an unconditional transfer to the first instruction in 024. The parentheses around 001 indicate that this portion of the instruction is modified in the course of computation. The Newton-Raphson method of computing roots is used here.

$$Z_{i+1} = 1/2(\mu/Z_i + Z_i),$$

where the $Z_i$ are successive approximations and $\mu$ is the radicand. $Z_0$ is taken as 1. In line 024, $\mu$ is brought in; $Z_1$ is computed in lines 025 to 026. Lines 027 to 031 constitute the iterated process. In lines 032 to 033 the new approximation is compared with the previous one; if the difference is not sufficiently small, $Z_{i+1}$ is substituted for $Z_i$, and another iteration is performed. If the difference is small enough, however, the latest $Z_i$ is placed in one of the registers 755 to 777, inclusive, to be printed later.

Having computed the root of the first $\mu$, the computer must obtain the next $\mu$; it does this by modifying the first instruction in 024, adding 1 to its memory location. The second instruction of 036 must be similarly modified so that the next computed root will be placed in a different register for printing. These modifications are made in lines 037 to 041.

[4] I. L. Auerbach, J. P. Eckert, Jr., R. F. Shaw, and C. B. Sheppard, "Mercury delay-line memory using a pulse rate of several megacycles," Proc. I.R.E., vol. 37, pp. 855–861; August, 1949.

[5] J. J. Bartik, "Square-Root Routine," Computational Analysis Laboratory, Eckert-Mauchly Computer Corp., Division Remington Rand, Inc., Philadelphia, Pa.

Since it is necessary to know when all 20 roots have been computed, a test is made in line 042. When the last modification of 036 takes place, an overflow occurs, changing $C777$ to $A000$ (the $A$ has no significance here; actually $C777 = 04777$, which on addition of 1 becomes 05000, which happens to be $A000$). Thus, subtraction of $A056A000$ will cause a sign reversal on each iteration, except the last, and the absence of the sign reversal on the last one causes the $T$ to be ineffective. Lines 043–044 are a "patch-up" routine to restore 024 and 036 to their original settings; line 045 contains a transfer of control to 754, where a stop instruction is placed. By putting the stop in 754 rather than in 045, the control counter is left with the proper setting, and the results, which are stored in locations starting with 755, may be printed.

While the foregoing example is a relatively simple routine, it illustrates several of the principles of programming. In particular, it shows the manner in which instructions are modified (by adding constants to their memory locations) and the use of tests (employing conditional transfer instructions) to determine whether an iterative process has been completed. It is of interest to point out that, although there are 20 lines of coding (excluding constants, storage registers, and the like), only 5 (025–031) are concerned with the actual computation. This is characteristic of computers of the class to which the Binac belongs.

### Acknowledgment

The authors wish to acknowledge the contributions of their associates in the Eckert-Mauchly Computer Corporation to the work described in this paper. Appreciation is also due several others for assistance in preparing the manuscript and illustrations.

## APPENDIX A

### PROOF OF RULE FOR COMPLEMENTING

To form the complement of any $m$-digit number $n$ $(2 > n \geqq 0)$ with respect to 2, replace all zeros of $n$ by ones, and all ones by zeros; then add a unit in the extreme right-hand position.

Let $n = \sum_{-m}^{0} a_i 2^i$ and let the number $n^1$ formed by reversing zeros and ones in $n$ be $\Sigma\, b_i 2^i$. Now, for any given $i$, if $a_i = 0$, $b_i = 1$; if $a_i = 1$, $b_i = 0$. It then follows that for any $i$, $a_i + b_i = 1$, and, therefore, $n + n^1 = \Sigma(a_i + b_i)2^i = 1.111 \cdots$. Adding $2^{-m}$, $n + n^1 + 2^{-m} = 10.0$, and, therefore, (since 10.0 is the binary equivalent of 2) $n^1 + 2^{-m}$ is the complement of $n$ with respect to 2.

If the convention is now adopted that numbers lying in the range $1 > n \geqq 0$ are positive, it follows that negative numbers in the range $0 \geqq n \geqq -1$, if represented by the complements of their absolute values with respect to 2, must lie in the range $2 > n \geqq 1$. They will, therefore, always have a 1 in the position to the left of the binary point (i.e., in the $p37$ position). The notation used can, therefore, represent numbers in the range $1 > n \geqq -1$. It also follows that the complementing method is reversible as, of course, it must be if it is to be of any value.

## APPENDIX B

### ADDITION OF COMPLEMENTS

In the following discussion it is assumed that all original numbers are within range of the computer (i.e. $1 > n \geqq -1$) and negative numbers are represented by their complements, as previously described. A simple binary addition should then give the correct result regardless of whether one or both addends may be negative.

If $n_1$ is positive and $n_2$ negative, the quantities added will be $|n_1|$ and $2 - |n_2|$. Their sum will then be $2 + |n_1| - |n_2|$. Now $||n_1| - |n_2||$ must be less than 1 since $|n_1|$ and $|n_2|$ are each less than 1; therefore, if $|n_1| - |n_2|$ is positive, the $p37$ position must contain a zero. The 2 would normally occupy the $p38$ position, but it is deleted by the delete gate so that the actual result is $|n_1| - |n_2|$, which is correct. If $|n_1| - |n_2|$ is negative, it can be written $-||n_1| - |n_2||$, and the result of the addition is $2 - ||n_1| - |n_2||$, which is the complement of $|n_1| - |n_2|$; again this is correct since negative results should be expressed in complement form.

If $n_1$ and $n_2$ are both negative, they will be represented by $2 - |n_1|$ and $2 - |n_2|$, respectively, and the sum will be $4 - |n_1| - |n_2|$. Since all pulses beyond $p37$ are deleted, it follows (since $2 = p38$) that the addition of $2p$ to any number where $p$ is any positive integer greater than zero, leaves the number unchanged. Therefore, as far as the computer is concerned, $4 - |n_1| - |n_2| = 2 - |n_1| - |n_2|$. Since $2 - |n_1| - |n_2| = 2 - |n_1| + |n_2|$, the result of the addition is the complement of the sum of $n_1$ and $n_2$, and is therefore correct.

It should be noted that if the addition of two positive numbers causes a carry-over into the sign ($p37$) position this implies that the sum exceeded 1, and is therefore outside the range of the computer; such a result must be distinguished from the negative number, which it superficially resembles. Similarly, the addition of two negative numbers must be considered improper if the result does *not* contain a 1 in the $p37$ position.

## APPENDIX C

### ILLUSTRATIVE EXAMPLE OF MULTIPLICATION

1. $0.110 \times 0.101 = 0.01111$ $(3/4 \times 5/8 = 15/32)$

| $A$ | $R$ | Explanation |
|---|---|---|
| 0.000 | 0.101 | *LSD* in $R$ is found to be 1. |
| 0.000+0.110=0.110 | 0.101 | $L$ is added to $A$. |
| 0.0110 | 0.010 | $A$ and $R$ are shifted; *LSD* of $R$ is 0. |
| 0.00110 | 0.001 | There is no addition, but another shift; *LSD* of $R$ is 1 again. |
| 0.0011+0.110=0.1111 | 0.001 | $L$ is added to $A$. $A$ and $R$ are shifted; *LSD* of $R$ (originally sign) is 0. No correction is needed, and the process is complete. |
| 0.01111 | 0.000 | |

Note: Digits to the right of 0.011 are actually deleted in the Binac.

2. $1.010 \times 1.011 = 0.01111$ $(-3/4 \times -5/8 = 15/32)$

| $A$ | $R$ | Explanation |
|---|---|---|
| 0.000 | 1.011 | *LSD* of $R$ is 1. |
| 0.010 | 1.011 | Add $L$ (without sign) to $A$. |
| 0.001 | 0.101 | Shift $A$ and $R$; *LSD* again 1. |
| 0.001+0.010=0.011 | 0.101 | Add $L$ again and shift. |
| 0.0011 | 0.010 | *LSD* of $R$ now 0. |
| 1.0011 | 0.010 | Add only the sign digit of $L$ and shift. |
| 0.10011 | 0.001 | *LSD* of $R$ (formerly sign) is 1. |
| 0.10011+0.110=1.01011 | | Add $-L$ (0.110) to $A$; sign of $L$ is also 1. |
| 1.01011+1.001=0.01111 | | Add 1.001 to $A$ to get final result. |

## APPENDIX D

### INSTRUCTION CODE

In the following list, $m$ is a 3-digit octal number representing a memory location, $(m)$ means "contents of $m$."

| | |
|---|---|
| $Am$ | Add $(m)$ to $(A)$, result in $A$. |
| $Cm$ | Transfer $(A)$ to $m$ and clear $A$. |
| $Dm$ | Divide $(A)$ by $(L)$, quotient in $A$. |
| $Fm$‡ | Add $(L)$ to $(A)$, result in $A$. |
| $Hm$ | Transfer $(A)$ to $m$ without clearing $A$. |
| $Km$‡ | Clear $L$; transfer $(A)$ to $L$ and clear $A$. |
| $Mm$ | Multiply $(L)$ by $(m)$, product (rounded off) in $A$. |
| $Sm$ | Subtract $(m)$ from $(A)$, result in $A$. |
| $Tm$ | Transfer control to $m$ (i.e., replace the number in the control counter by the number "$m$") if sign of $(A)$ is negative. |
| $Um$ | Transfer control to $m$ unconditionally. |
| $01m$‡ | Stop computation, leaving computer on "time out." |
| $22m$‡ | Shift $(A)$** one binary digit to left, i.e., multiply $(A)$ by 2. |
| $23m$‡ | Shift $(A)$ one binary digit to right, i.e., divide $(A)$ by 2 without rounding off. |
| $24m$‡ | Stop computation if "break point" switch is set; otherwise proceed to next instruction. |
| $25m$‡ | Proceed to next instruction (skip). |

‡ In these instructions $m$ is not significant, and 000 is customarily used.

** The shift instructions also operate on $R$, but this is not significant except when the shift forms a part of the multiplication or division process.

### BIBLIOGRAPHY

1. H. H. Goldstine and A. Goldstine, "The electronic numerical integrator and computer (ENIAC)," pp. 97–110; July, 1946.
2. D. R. Brown and N. Rochester, "Rectifier networks for multiposition switching," PROC. I.R.E., vol. 37, pp. 139–147; February, 1949.
3. University of Pennsylvania, Staff of the Moore School of Electrical Engineering, "Progress Report on the EDVAC," University of Pennsylvania, Philadelphia; June 30, 1946.

# Logical Description of Some Digital-Computer Adders and Counters*

HARRY J. GRAY, JR.†, ASSOCIATE, IRE

*Summary*—**A number of adding and counting circuits which have a high degree of reliability are presented. Most of them have been used in a digital computer. A high-speed binary counter is described which can be cycled either forwards or backwards.**

### INTRODUCTION

WHEN THE ENIAC[1] was nearing completion, it became apparent that a reconsideration of the logic would make possible a much smaller and more flexible digital computer. In the course of a development program for such a machine, a large number of computer circuits and techniques were conceived, many of which found their way into the EDVAC.[2–4] A few of these circuits are described below. They are adding and counting circuits which, with one exception, do not use Eccles-Jordan-type trigger circuits. The one circuit which uses an Eccles-Jordan-type trigger circuit does so in an unconventional manner.

* Decimal classification: 621.375.2. Original manuscript received by the Institute, December 4, 1950; revised manuscript received, April 24, 1951.

† Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia, Pa.

[1] ENIAC: Electronic Numerical Integrator and Computer, designed and constructed by the University of Pennsylvania, Moore School of Electrical Engineering, for the Ordnance Department, Department of the Army, under Contract No. W-670-ORD-4926.

[2] EDVAC: Electronic Discrete Variable Computer, designed and constructed by the University of Pennsylvania, Moore School of Electrical Engineering, for the Ordnance Department, Department of the Army, under Contract No. W36-034-ORD-7593.

[3] G. W. Patterson, R. L. Snyder, L. P. Tabor, and I. Travis, "The EDVAC—A Preliminary Report on Logic and Design," University of Pennsylvania, Moore School of Electrical Engineering, Research Division Report 48-2; February 16, 1948.

EDVAC Staff, "A Functional Description of the EDVAC," vols. I, II, University of Pennsylvania, Moore School of Electrical Engineering, Research Division Report 50-9; November 1, 1949.

### REPRESENTATION OF INFORMATION[3,4]

In the EDVAC there are two types of information, numbers, and orders. The orders tell the EDVAC what to do with the numbers. These two types of information can be represented by "words." A word consists of 44 binary digits,[5] where the binary digit "1" is represented by a voltage pulse and the binary digit "0" is represented by the absence of a pulse. For example, a string of forty-four "1"'s would be represented by a temporal array of pulses about 0.3 microsecond wide (standard width) and spaced one microsecond apart. Since the numerical value assigned to a pulse or a "no pulse" or blank depends on its position in time, a time reference is provided by a crystal-controlled timing unit called the "Timer,"[6] which divides machine time into periodically recurring intervals of 48 microseconds duration called "minor cycles." Each minor cycle is further divided into 48 equally spaced-in-time positions called pulse positions, 44 of which can be occupied by the pulses or

[5] H. E. Buchanan and L. C. Emmons, "A Brief Course in Advanced Algebra," Houghton Mifflin Co., Boston, Mass., pp. 169–179; 1937. This book contains a lucid explanation of number bases, conversion from one base to another, and so on.

[6] R. M. Goodman, "A digital computer timing unit" (completed manuscript to be submitted for publication in the near future).