THE EDVAC

A PRELIMINARY REPORT ON LOGIC AND DESIGN

UNIVERSITY OF PENNSYLVANIA
Moore School of Electrical Engineering
PHILADELPHIA, PENNSYLVANIA

February 16, 1948

D. Charp

THE EDVAC

A Preliminary Report on Logic and Design

Prepared at the request of the Office of the Chief of Ordnance, Department of the Army, for the National Bureau of Standards, describing work under Contract W-36-034-ORD-7593.

This Report was Prepared by:

G. W. Patterson
R. L. Snyder
L. P. Tabor
Irven Travis

Research Division,
Moore School of Electrical Engineering
University of Pennsylvania.

Approved:

Irven Travis
Supervisor of Research

16 February 1948
Date

Report Serial No.
48-2

TABLE OF CONTENTS

ERRATA
Report No. 48-2 (Sheet 1 of 2)

Negative line numbers signify counting up from bottom of page.

Page	Line	Now Reads	Should Read
i	6 and 7	Perform ... 5.0 ...	Perform ... 5.0 ...
7	13	magnetiz d	magnetized
7	-11	The wires	However; the wires
7	-10	have n = 44	have n = 0, n = 44
7	-10	, however,	,
8	-4	2^{-i} c_i	$2^{-i}\ c_i$
11	-12	language - (	language (
26	-2	Reader Recorder	Reader-Recorder
34	-3	control	dispatcher
49	3	may an	may have an
50	-2	$2^{-43} - 1 \leq m \leq 0$	$2^{43} - 1 \geq n \geq 0$
50	-2	$2^{43} - 1 \leq n \leq 0$	$2^{43} - 1 \geq n \geq 0$
51	12	i =	$i = \infty$
52	14	$\overline{q^*}$,	$\overline{q^*}$.
52	16	conventions,	conventions.
53	7	$\bar{w}(q^*)$	$\bar{w}(\overline{q^*})$
53	10	d, 5 -	d, $\overline{5}$ -
53	-2	operator .	operator $\overline{\ \ }$.
54	8	$(N_2 - N_1)\ 2^{-43}$	$N_2 - N_1\ 2^{-43}$
54	12	<b	$<\lvert\bar{b}\rvert$
55	10 and 11	convention,	convention. (delete remainder of sentence)

ERRATA
Report No. 48-2 (Sheet 2 of 2)

Negative line numbers signify counting up from bottom of page.

Page	Line	Now Reads	Should Read
55	-7	$\overline{b}$ (unless $\overline{q} = 0$),	$\overline{b}$,
55	-7	$\overline{b}$ (unless $\overline{q} = 0$),	$\overline{b}$,
58	-3	Wire (W, 2 -)	(W, $\overline{2}$ -) Wire
59	-1	ust	must
61	13	3^{A}_{k}	$3^{\overline{A}}_{k}$
62	-13	2	8
63	-11	1^{Ω}	$1^{\overline{\Omega}}$
65	12	R5a	R5A
66	10	classification	clarification
67	3	$N(_{1}\overline{A}) + 1$	$N(_{1}\overline{A}) - 1$
67	-9	3 -	$\overline{3}$ -
75	-3	shifter	shifted
76	1	$_{3}\overline{A}$	$\overline{w}(_{3}\overline{A})$
76	1	24 (= 20, binary	24 (=20, decimal
76	7	tablet	table
76	-2	will any	will also any
77	-13	$\underline{1}\overline{XX0}$ fails	$\underline{1}\overline{XX0}$ (except $\underline{1}\overline{000}$) fails

1. Introduction

In the design of the ENIAC, which was constructed by the Moore School of Electrical Engineering for the Ballistic Research Laboratory of Aberdeen Proving Ground, it was necessary, because of the great need at that time for such a calculator, to freeze engineering at a very early date. As construction proceeded, however, it became increasingly obvious that by a reconsideration of the logic, it was possible and desirable to design a computer much smaller in size than the ENIAC, with even greater flexibility and better mathematical performance than the ENIAC. This conclusion was reached as a result of consideration of the computing speed possible with electronic design, the operating characteristics of the circuit elements required in electronic computers, and the nature of the mathematical problems which can be attacked economically only by large scale high speed computers. The Ballistic Research Laboratory became interested in this possibility, and in late 1944, it was agreed that, as work on the ENIAC permitted, the design and construction of such a machine should be undertaken, under the sponsorship of the Office of Chief of Ordnance, U.S. Army, by the Moore School of Electrical Engineering.

A progress report on this machine, the EDVAC, or Electronic Discrete Variable Computer, was published in September, 1945, covering a number of possible physical designs, all meeting the general logical requirements established for the EDVAC.

In a conference in early 1946, attended by Dean Harold Pender and Dr. Irven Travis of the Moore School, Col. Paul Gillon of OCO, and Dr. John von Neumann of the Institute for Advanced Study, it was decided that experience with a pilot model of the EDVAC type was urgently needed, in order to obtain information about coding problems and

operating characteristics, which could then be applied to the design of a very comprehensive calculating machine. It therefore seemed expedient that the Moore School should immediately proceed with the design and construction of a small preliminary model of EDVAC for the Ballistic Research Laboratory, while the Institute for Advanced Study should undertake a study program leading to the establishment of design requirements for a large-scale comprehensive computer.

In working out the preliminary designs for this small EDVAC, Moore School personnel worked in close cooperation with representatives of the Ballistic Research Laboratory who were then operating the ENIAC. Both Moore School and BRL personnel very naturally desired this small EDVAC to be as comprehensive as possible, and still meet the requirement for small size and simplicity. In order to obtain a mutually agreeable interpretation of the term "Small Preliminary Model of EDVAC", a conference was held at Aberdeen on 9 October 1946, attended by Dean Pender and Dr. Travis of the Moore School Colonel G. F. Powell and Mr. Samuel Feltman of OCO, Col. L. E. Simon of BRL, Dr. von Neumann of IAS, Mr. Harry Diamond of the National Bureau of Standards, and other representatives of these activities.

The Moore School presented three possible designs, briefly described as follows:

EDVAC I. A very simple binary computer, with automatic addition, subtraction and multiplication, programmed division, and no internal checking. Memory capacity 1000 words.

EDVAC II. A simple binary coded decimal computer, with fixed decimal point, all four basic arithmetic processes automatic, and automatic checking in the computer. Memory capacity 1000 words.

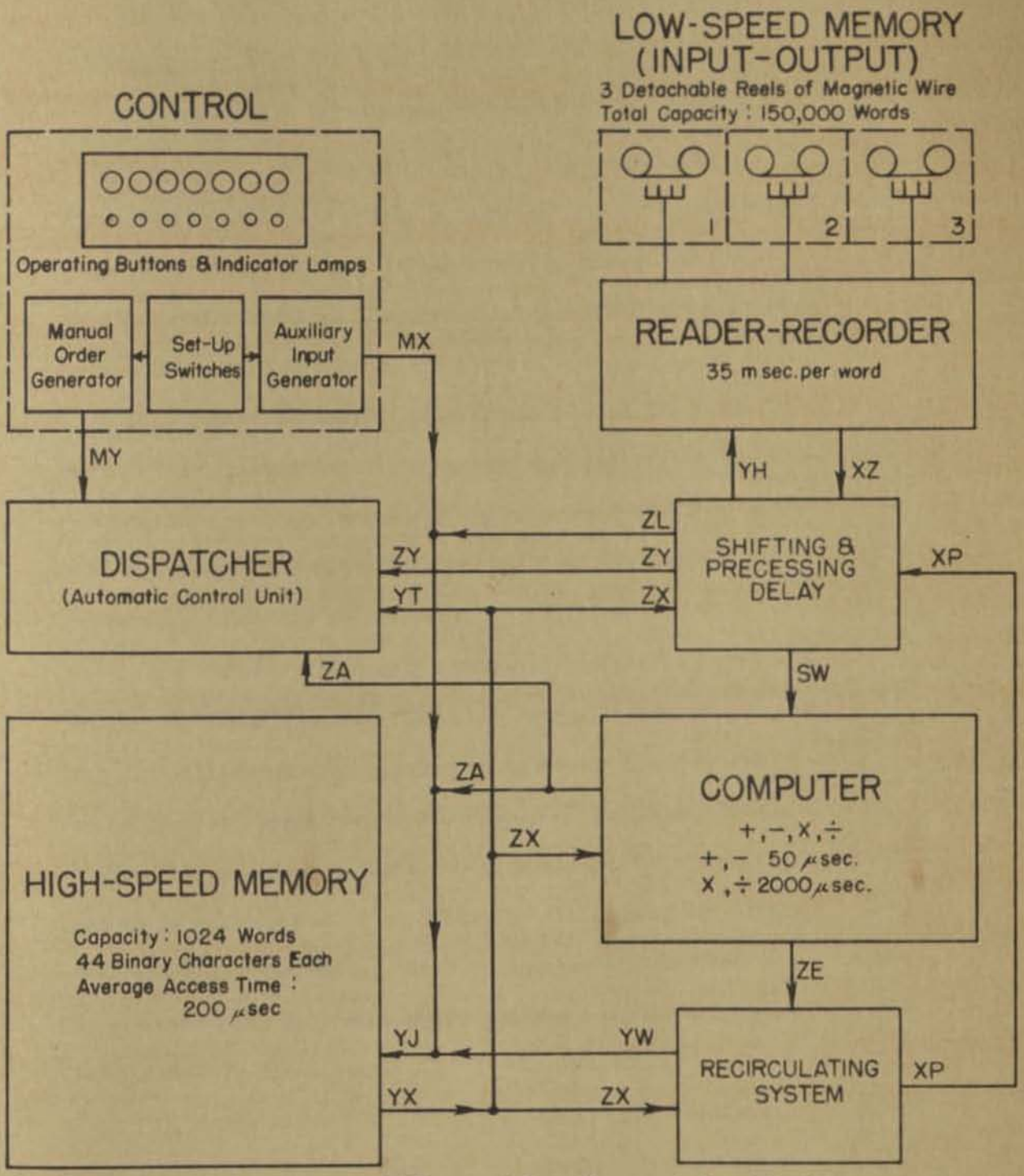
EDVAC III. A more comprehensive machine, with automatic floating decimal point, and all the automatic features listed under EDVAC II. Memory, 4000 words.

In this conference, it was decided that a binary machine based on the EDVAC I design, but with automatic division and automatic checking, was desirable. This machine was designated as the EDVAC 1.5. It was also decided that the National Bureau of Standards would design and supply the equipment for preparing and printing from the magnetic wire used in the Reader-Recorder of the EDVAC.

As the design of the EDVAC 1.5 progressed, further discussions with BRL representatives indicated the desirability of introducing certain additional orders, such as the "Extract" order, described in a later section of this report, into the coding scheme. Since these changes, together with another change necessary to give increased stability to the memory, would considerably increase the complexity of the EDVAC, another conference was held in Aberdeen on 27 May 1947, in which two alternative proposals, EDVACS 1.5A, and 1.5B, with and without the added coding flexibility, respectively, were considered. No final decision was reached at this conference, and Dr. R. F. Clippinger of BRL was appointed to consider the various possibilities, and submit a report. As a result of this report, the EDVAC 1.5B design was approved, with one minor modification. The major part of this report is devoted to a condensed description of the approved design.

In the discussion that follows, it has been necessary, in order to keep the size of this report within reasonable bounds, and to avoid excessive attention to minor details, to introduce certain simplifications, and to omit occasional exceptions to certain general statements. Emphasis is placed on the overall logic, the basic engineering design, and on the principles of operation of the EDVAC.

UNIVERSITY OF PENNSYLVANIA
MOORE SCHOOL OF ELECTRICAL ENGINEERING



NOTES : 1. Except for High-Speed Memory, All Times and Capacities are in Round Numbers
2. Components Omitted for Simplicity : (a) Timer, (b) Power Supply, (c) Neon Light Register (d) Oscilloscope, (e) Mode of Operation Switch, (f) Exceed Capacity Option Switches (g) Address A Generator, (h) Address B Generator

PRELIMINARY

APR 7 1948

FIGURE 1

DRAWN BY J.E.G. 1-28-48	EDVAC BLOCK DIAGRAM	SCALE
CHECKED BY J.W.M. 1-28-48		104-10LA-4
APPROVED BY		

2. Organization of the EDVAC

A simplified block diagram of the EDVAC is shown in figure 1(drawing 104-101A-4.) All the interconnecting lines represent single channels carrying words or characters only; none of the control connections are shown. Major physical and functional components of the EDVAC are indicated by the full width blocks in the figure.

(a) Reader-Recorder. This unit contains the three wire drives, associated servo-mechanisms, amplifiers for the magnetic reading, recording and erasing heads, and equipment required to transfer information from the heads to the precessing delay and vice versa.

(b) Control. This unit contains all operating buttons, indicating lamps (but not flip-flop neons) , control switches, and an oscilloscope for aid in maintenance. The control sends special orders, which are set up on switches, to the dispatcher in order to start the machine, and may send words, set up on another set of switches, to the high-speed memory.

The above two units, shown at the top of the block diagram, are the only ones which contain devices which are manipulated by the operator.

(c) Dispatcher. This unit decodes orders received from the control and the memory, and special data received from the precessing delay or computer, and emits control signals to the other units which are needed by them in order to properly perform their functions. It contains an elect-

rical delay memory which retains the order while it is being performed. The dispatcher cabinets also house the shifting and precessing delay which assembles the characters received at low speed from the reader-recorder and transmits them at high speed to the high-speed memory, and also dissects words received from the high-speed memory and transmits them, one character at a time, to the reader-recorder. This delay is also used in the "extract" order. The recirculating system is housed in the dispatcher. This gating apparatus permits a word being transferred out of the memory to be simultaneously copied back into the same position, and has other special functions in the wire and extract orders.

(d) High-Speed Memory. This consists of two identical units, each containing 64 acoustic delay lines and associated regeneration circuits. Each line or "tank" has a capacity of 8 words.

Each memory cabinet also houses three short tanks of one word capacity, which are connected to the computer. In addition to the memory units, the cabinets also house apparatus to decode the addresses received from the dispatcher and select the memory position whose contents are to be transferred out of the memory or to be replaced by incoming data.

(e) Computer. This unit performs the rational operations, +, -, x, ÷, on pairs of signed numbers received from the high-speed memory, and returns the result to the memory at the appropriate time. When the compare order is programmed, it transfers the result of a subtraction to the dispatcher, which uses the sign of this result to select one of two alternative courses of action. The computing equipment is in duplicate and the answers are compared, digit for digit. Any disagreement will stop the machine and give an "abnormal halt" indication. If the result of performing +, -, ÷, gives an answer whose absolute value is equal to or greater than unity, the computer emits signals to the dispatcher and control which may be used to modify the subsequent operation of the machine in various ways.

(f) Timer. This unit is not shown in the block diagram. It emits clock pulses at intervals of 1 μsec, and timing pulses at intervals of 48 μsec (1 minor cycle). There is also a major cycle pulse occurring every 384 μsec (1 major cycle = 8 minor cycles) originating in the dispatcher.

In addition to the EDVAC proper, an input-output system is being constructed by the National Bureau of Standards. This equipment, whose design is a relatively independent problem, is to consist of modified teletype equipment for inscribing keyboarded information to the magnetic wires, and for outscribing information from the magnetic wires to automatically typewritten characters on rolls of ordinary paper. The apparatus which converts keyboard data to wire recordings is known as the inscriber, that which converts wire recordings to typewritten characters, as the outscriber.

Before discussing the block diagram, it is important to consider the various languages that are employed in the EDVAC. Basically, there is a spoken and a written language. The former consists physically of timed electrical pulses. The essential function of the EDVAC timer is to interpret these signals. The spoken language is dynamic; if the power is cut off, all of its characters will disappear. The written language consists physically of magnetized regions on the wires. If power is cut off, the characters of this language are permanent , owing to the magnetic retentivity of the medium.

Both languages are binary, that is, only two kinds of characters are employed. In the spoken language the two kinds of characters are characterized by the presence of a pulse or the absence of a pulse. In other words, the absence of a signal is not a mere space between characters, but conveys information. The timing pulses are compared with the pulses in the word and when both are present, we have one kind of character, and when

only the timing pulse occurs, we have the other kind. The timing pulses are cyclic and thus serve to identify the beginning and end of a word as well as all intermediate positions. This language is thus seen to be synchronous, and not interpretable except with the aid of a clock.

In the written languages, the two kinds of characters are distinguished by regions of opposite longitudinal magnetic polarity on the wires. The reader-recorder can properly identify these regions regardless of the direction in which the wire moves, provided only that the direction in which the inscriber prepared the wire is known. This direction must of course be known at all times when the wire is used. Unmagnetized stretches occurring between magnetized areas, although nothing but blanks or spaces, have the important function of making the characters discrete and countable. If all the words had the same number of characters, n, the beginning and end of a word, as well as any intermediate character, could be identified by counting the characters from the beginning of the first word on the wire, and determining the residue modulo n. This is easily mechanized by a counter which "counts off" the characters in groups of n. However, the wires, as prepared by the inscriber, have $n \neq 0$, $n \neq 44$ or $n \neq 54$, ~~however,~~ hence the end of a word is signalled by a special kind of character or "marker" on the wire, which is an abnormally long magnetized region identifying the end of a word. The written language thus has three distinct kinds of characters, as well as blanks, and is static and hence asynchronous.

More conventional languages are used in the input-output system, employing, inter alia, the 10 decimal digits. The input-output system is capable of transliterating these characters into the binary language, but true translation, i.e. the binary-denary

(binary-decimal) interconversion is a mathematical problem which, in the absence of a special purpose computer, will be programmed on the EDVAC. For problems requiring large amounts of computation per unit of input or output (for which the EDVAC is intended), this procedure is perfectly feasible. More details on the languages employed in the input-output system will be found in Section 4. In what follows we restrict ourselves to the languages used in the EDVAC proper.

Having disposed of the characters of our languages the nature of the words requires some explanation. In the spoken language, a word consists of 44 characters. Introducing $\bar{w}$ as a word variable, $\bar{w} = (c_1, c_2, \ldots, c_{44})$, where the c's are either pulses or non-pulses. In the computer, a word is interpreted as a 43-digit binary number with sign. The two kinds of characters are interpreted as zeros and ones. The kind of character which is represented by a pulse is interpreted as a 1, and the absence of a pulse as a 0. The function $\bar{x}$ $(\bar{w})$ is introduced: $\bar{x} = \left[\sum_{i=1}^{i=43} 2^{-i} c_i\right] (-1)^{c_{44}}$, where $c_1, c_{44} = 0, 1$.

A c_{44} thus represents the algebraic sign of the number, and a pulse (1) represents -; no pulse (0), +.

Since $|\bar{x}| = \sum_{i=1}^{i=43} 2^{-i} c_i$, $\max |\bar{x}| = \sum_{i=1}^{i=43} 2^{-i} = 1-2^{-43}$, and $\min |\bar{x}| = 0$.

It should be noted that $|\bar{x}|<1$, and we have fixed the "binary" point at the left. Thus $-1+2^{-43} \leq \bar{x} \leq +1-2^{-43}$, and $\bar{x}$ can be varied only by increments of 2^{-43} or multiples thereof. Any real number, x, in the closed interval $-1+2^{-44} \leq x \leq +1-2^{-44}$ can be represented, using only one word, with a maximum error of $2^{-44} \sim 6\text{x}10^{-14}$; the absolute accuracy (and also the relative accuracy with respect to max $|\bar{x}|$) of one-word representations of numbers is about 10^{-13}. Since there are 2^{44} different kinds of words, but only $2^{44}-1$ different values of $\bar{x}$, there is one number which has a non-unique representation, namely; zero. The computer is so designed as to recognize -0 and +0 as the same number, which is not an entirely trivial problem.

The interpretation of a word in the dispatcher is quite different. The dispatcher, in effect, breaks the word up into 5 segments: c_1-c_{10} inclusive, c_{11}-c_{20}, c_{21}-c_{30}, c_{31}-c_{40}, and c_{41}-c_{44}. The first 4 segments of 10 characters each are called addresses 1, 2, 3, 4, respectively. In order to have a convenient symbolism to employ when discussing these segments, particularly in section 5, five functions of $\bar{w}$ are introduced:

$$\bar{1A} = (1A1, 1A2, \ldots, 1A10) = (c_1, c_2, \ldots, c_{10}),$$

$$\bar{2A} = (2A1, 2A2, \ldots, 2A10) = (c_{11}, c_{12}, \ldots, c_{20}),$$

$$\bar{3A} = (3A1, 3A2, \ldots, 3A10) = (c_{21}, c_{22}, \ldots, c_{30}),$$

$$\bar{4A} = (4A1, 4A2, \ldots, 4A10) = (c_{31}, c_{32}, \ldots, c_{40}),$$

$$jAi = c_{10(j-1)+i}; \quad j = 1, 2, 3, 4; \quad i = 1, 2, \ldots, 10$$

$$\bar{T} = (T_1, T_2, T_3, T_4) = (c_{41}, c_{42}, c_{43}, c_{44}).$$

Since $2^{10} = 1024$, the functions $_j\bar{A}(\bar{w})$, that is, the four blocks of 10 binary characters, have exactly 1024 functional values, which agrees exactly with the number of positions in the high-speed memory. To a first approximation, the dispatcher uses each address to select a memory position involved in a transfer of data; since there are 4 addresses per order, the order code is referred to as a 4-address code. The final block of 4 characters, the function $\bar{T}(\bar{w})$, designates the order-type; namely , the operation to be performed on the numbers involved in the transfers. (i.e., addition, subtraction, etc.) Complete details will be found in section 5. Although sixteen different kinds of order-types are possible ($2^4 = 16$),

only 11 different codings are ordinarily used (A, S, M, D, C, m, d, E, W, H, ₹). If one of the 5 unused codes enters the dispatcher, the machine will cease computing and indicate the cause. This provides a valuable safety feature. If, through some error in programming, a number is sent to the control, rather than the order, the probability is 5/16 that it will be meaningless. The probability of any sizeable number of such erroneous orders passing undetected through the dispatcher thus becomes vanishingly small, particularly when it is noted that the characters c_{41}-c_{43}, which form part of the order-type code, correspond to the least significant digits of the numbers and hence will tend to fluctuate randomly.

There is no internal characteristic of a word which distinguishes orders from numbers. The distinction is dynamic. Any word entering the dispatcher is an order; any word entering the computer is a number. The same word may have both classifications at different times. This possibility is one of the major innovations in the EDVAC, which distinguishes it from earlier large-scale computers; there is no segregation of program and numerical data.

It should be noted that the characters c_1-c_{44} inclusive, in a single word are invariably accompanied in the memory by four constant characters, c_{45}-c_{48}. These characters are all "non-pulses" and are required to give a time interval between words sufficiently long to permit the electronic circuits to operate properly. These characters are known as switching blanks. Although a word usually only contains 44 useful characters, a minor cycle is 48 μsec in length. It should also be borne in mind that the characters appear in reverse time sequence within the machine, that is, c_{44} appears first, then c_{43} one μsec later, and finally c_1. In a serial adder, which is used in the EDVAC, the carries must propagate from right to left, as in

pencil and paper addition; the least significant digit must be examined first.

The written (wire) language will now be discussed. There are really two languages, the words of which can be intermingled in an arbitrary manner. One language has 44 binary characters per word, $k_1, \ldots, k_{44}$. These correspond exactly to the 44 binary characters of the spoken language. The other language has 54 characters per word, $K_1, \ldots, K_{10}, K_{11}, \ldots, K_{54}$. The last 44 characters again correspond exactly to the 44 binary characters o f the spoken language. The first 10 characters, K_1-K_{10}, correspond to an address. This address is called a fifth address, and designates the desired location of the word K_{11}-K_{54} in the high-speed memory. When "read fifth address" is programmed, the fifth address is sent to the dispatcher, which then uses it to control the location of the word. The EDVAC reader-recorder can be programmed to read either language, but to record only the k-language. The inscriber and outscriber equipment is similarly designed. Either language can be keyboarded on the inscriber, but the outscriber can transliterate only the k-language (see section 4).

It will be recalled that in the spoken language the characters occur in reverse time sequence. On the wires, however, the characters originally appear in normal sequence. The keyboarding proceeds from left to right in the normal way, and when the wire is traveling in the forward direction (defined as the direction it moved when it was recorded in the inscriber equipment) k_1 passes under the reading heads before k_2, etc. This causes no difficulty, since when the wire is read into the machine, all of the characters are assembled in the processing delay before they are sent to the high-speed memory. It is merely necessary to assemble them in the reverse order when the wire is running forward in the EDVAC, and to

assemble them in the obverse order when the wire is running backward. Words of the k-language can thus be read with the wire running in either direction. The order of the words may be reversed, but the order of the characters within the words is not. Individual words are thus palindromic for all practical purposes. Recording on the wire can be accomplished in either direction in a similar manner. Since the processing delay can only assemble 44 characters, it is not possible to handle words of the K-language in this palindromic fashion. The first ten characters from the wire are assembled and sent to the dispatcher, and then the last forty-four are assembled and sent to the memory in accordance with action taken by the dispatcher on the first ten. The two portions of the word can be properly reversed, but not the word as a whole. Attempts to read these words backwards would result in interpreting K_{45}-K_{54} as the address and K_1-K_{44} as the word. No serial palindromic manipulation of the two separate parts will be successful in accomplishing the desired results.

We now turn to the principles involved in the routing of data within the machine. The basic transfer operation is known as an "execute." The function of an execute is to make one of the 1024 high-speed memory positions available to the rest of the machine. The 10 binary characters of an address are required for this purpose. The switching problem is partly spatial and partly temporal. The characters ${}_jA_1$-${}_jA_7$ inclusive are used to select one of 128 long tanks, (2^7=128) and characters ${}_jA_8$-${}_jA_{10}$ are used to select one of 8 positions (minor cycles) in the long tank (2^3=8). A three-stage binary counter in the dispatcher counts minor cycles modulo 8. The timing pulses are used to identify the different characters within a word, and the output of this counter is used to identify the

different words in a tank. The timer is analagous to the minute hand of a clock, and the minor cycle counter analagous to the hour hand. The dispatcher compares the status of this counter with $_jA_8$-$_jA_{10}$ each minor cycle, and when they agree, the dispatcher recognizes that the word desired is on the verge of emerging from the output end of some long tank, and emits an execute signal. $_jA_1$-$_jA_7$ are continually available at the high speed memory, ready to select the proper tank on receipt of the execute signal. As soon as the execute is received the proper long tank is opened long enough to emit the 44 characters of the word stored therein on the connection marked YX in the diagram (see figure 1), and to receive whatever information appears at YJ during the same time interval. As soon as the transfer is completed, the internal recirculating system of the tank restores the regeneration loop and the memory is completely isolated from YJ and YX until another execute occurs. It will be seen that as far as the memory is concerned, an execute merely causes the contents of the memory position identified by the address concerned to be emitted at YX, and to be replaced by whatever data are concurrently present at YJ. Whenever a word is withdrawn from the memory for use in the computer, dispatcher, or if the ultimate destination is the magnetic wire, the word emitted is simultaneously made available again to the memory at YJ, from YW, via the external recirculating system and ZX. The information is thus merely copied and not erased. On the other hand, if the result of a computation is being transferred from the computer to the memory, or a word is being read from the wire or the input generator in the control, the external recirculating system fails to make the connection between ZX and YW and the old contents of the position affected disappear. This design philosophy is followed whenever

reasonably possible; no data are erased until it becomes essential to do so. The word "erased" may be slightly misleading in this context. Within the high-speed memory, a total absence of pulses, $\bar{w}_o$, is significant since $\bar{x}(\bar{w}_o) \equiv 0$. A cleared memory position will be interpreted by the computer as a zero. The dispatcher interprets it as an "unused order".

As soon as an execute occurs, the dispatcher immediately switches to the next address to be considered. Since there are only 10 characters to examine, this can easily be accomplished serially during one minor cycle. By the time the memory contents affected by the first execute have been completely transferred, the checking of the next address is completed, and if agreement is found, the second execute follows the first after one minor cycle has elapsed. Agreement must certainly occur before 8 minor cycles have elapsed, so unless the dispatcher inhibits the execute for reasons not pertinent to this discussion (such as waiting for the computer to complete a multiplication), the second execute must follow the first by at most 8 minor cycles. Assuming random distribution of the addresses, the average waiting time will be 4 1/2 minor cycles or 216 μsec. The maximum is 384 μsec (8 minor cycles), the minimum is 48 μsec (1 minor cycle). Although the access time of the memory is stated to be 200 μsec in round numbers, it is possible in certain problems to approach the optimum waiting time of 1 minor cycle by careful programming. It should not be assumed that the waiting time is necessarily above and beyond the time required for computing. For instance, in addition, the first execute transfers one of the summands to a computer short memory tank, the second execute transfers the second summand to the computer adder, and simultaneously transfers the previously received contents of the computer short

memory to the computer adder. The two numbers thus enter the adder in parallel. The adder has a delay of only about .6 μsec which can easily be made up during a switching blank. By the time the second summand has completely left the memory the sum is ready to emerge from the adder memory where it is to be stored until the third execute occurs. If the desired location of the sum is properly arranged, the three executes may occur in three successive minor cycles. Thus from the time the first character of the first summand leaves the memory to the time the last character of the sum enters the memory, as little as 3 minor cycles may elapse. Since 3 transfers have occurred, addition requires no time whatever in addition to the waiting time.

Continuing with the addition example, the first execute causes the first addend to be sent to the computer. Characters c_{41}-c_{44}, that is, $\bar{T}(\bar{w})$, are also decoded at this time, and the results memorized on flip-flops until the entire order is completed. The execute also advances a ring counter which controls the selection of the 4 addresses in the order. The second execute transfers the second addend to the computer and starts the addition. The recirculating system returns both of these numbers to the memory simultaneously with their withdrawal. The third execute transfers the sum to the memory. The recirculating system closes for this execute. Finally, the fourth execute transfers a new order to the dispatcher. S,M,D orders (subtraction, multiplication, division, both of the latter with round off) are similarly handled. The program chain can thus be stored in the high-speed memory in any arbitrary manner; the control is not required to scan memory positions in sequence. This permits great flexibility in programming.

sub-routines can be easily integrated or additional orders interpolated when the problem is in the planning stage without requiring extensive alterations of previous coding results. Programming for minimum time is also more feasible with a four-address code. It should be pointed out that the entire order is not decoded at one time. There is only one address-decoding circuit in the dispatcher, which scans the addresses serially.

The computer performs directly (i.e., using one order) the rational algebraic operations, together with all necessary transfers. The result is always returned to the memory. In the case of multiplication and division two options are provided, one producing a round-off and the other a more accurate result suitable for problems requiring more significant digits than can be stored in a single memory position. These order-types are designated as m,d. The computer also performs a subtraction when the dispatcher reaches a "compare" order, and the result is sent to the dispatcher instead of the memory. In this order, the contents of the memory position designated by address 1 are compared with the contents of address 2. Depending on the relative magnitude of these two numbers, the dispatcher selects its next order from address 3 or address 4. The compare order is the EDVAC analogue of what are variously known as branch, discriminate, or conditional transfer orders in other terminology, and permits previously computed results to affect the course of the computation.

In addition to these 7 orders, which the computer performs, there is the wire order which transfers any selected number of words from 1 to 1024 from the corresponding number of positions ready to pass under the reading heads in either direction to a selected sequence in the high-speed memory, or vice versa.

These words must be written in the k-language (44 characters per word) previously mentioned. It also permits words to be transferred from positions ready to pass under the reading heads in the forward direction to positions identified by the characters K_1-K_{10} on the wire, until a designated memory position has been filled. These words must be written in the K-language (54 characters per word). Finally, it permits a transfer of a word set up on switches on the control panel (auxiliary input generator) into the high-speed memory. In all cases, if the information is coming from the memory, no erasure occurs, but if the information is going into the memory, the previous contents are displaced by the new data.

The extract order permits digits to be shifted, i.e., a linear transformation on the subscripts of the c_i, it also permits selected portions of a word to be replaced by corresponding portions of this shifted word. This order enables a word to be dissected, and enables replacements of word segments to be performed.

The visual order is designed for a future installation; it performs no computation, but emits the contents of 2 selected memory positions simultaneously on 2 outputs. This is designed to be used in conjunction with a long-persistence viewing scope and digital to continuous conversion apparatus which is planned for future installation.

Finally, the halt order causes the machine to cease computation, give an audible and visual signal, retain all results, and permit calculation to be resumed when desired. This is used for programming break points or indicating the conclusion of the problem. Full details on all these operations will be found in section 5.

Normal operation of the EDVAC will proceed as follows: the operator will take one or more "printed" wires and possibly some blank wires and load one or more of the three wire drives on the reader-recorder. The blank wires require preparation since even though no characters are present the markers are needed. A special order to read the "printed" wire will then be set up on the control switches; another switch (mode of operation) is set to direct the control to read this special order. The initiate button is then depressed, causing the special order to be sent to the dispatcher where it is acted upon and the information on the wire is read into the high-speed memory. The machine then halts after a few seconds. The switch (mode of operation) is then set to a continuous mode. The initiate button is depressed again. The machine starts and continues to operate until a programmed halt is reached. An audible and visual signal are given and the machine ceases computing. The operator then removes the wires which contain the recorded data, presumably one of the wires which was originally blank. Each output wire, originally prepared on the inscriber equipment, is loaded into the outscriber equipment which automatically prepares a typewritten transcript of the information on the wire.

3. Discussion of Design Problems

As pointed out in the first section of this report, the EDVAC is required to be much smaller than the ENIAC, with greater flexibility and better mathematical performance. It was tacitly assumed that the speed would be at least equal to that of the ENIAC. Reliability is an obvious requirement, and operating experience with large scale computers of various types in service today has further indicated the need for emphasis on design for maximum reliability, coupled with inclusion of an adequate checking system. These, then, are the basic design requirements for the EDVAC.

In order to obtain high overall speed in the solution of complex mathematical problems, high functional speed must be supplemented by an adequately large high-speed memory. The most influential factors in determining the overall design of a large scale computer are probably the nature and size of the high-speed memory.

In late 1946, a mercury acoustic delay memory unit had been built by the Moore School, and had operated stably and reliably for long periods. At this time, while a number of other types of memory gave promise of eventually becoming useful, it did not appear that any other type could be reduced to an engineering design in the immediate future. The mercury memory was therefore selected for the EDVAC.

In the discussion of possible problems for digital computers, requirements for a high-speed memory of as many as 10,000 words have been encountered. However, many of the problems currently under consideration for high-speed computers can conveniently be handled with a memory of 1,000 words or less without excessive loss of time because of too frequent replacements of the contents of the high-speed memory from the input mechanism or the intermediate memory. Since the EDVAC is a binary machine, selection of a number

of memory positions which is a power of 2 will simplify coding and switching problems. It turns out that a mercury acoustic memory of 1024 words requires very nearly the same amount of equipment as the rest of the machine. A reduction in memory size to 512 words, the next lower power of 2, would result in a saving of only 25% in size and cost of the machine, and would certainly slow down the solution of many problems. On the other hand, increase to 2048 words would increase the size and cost 50%, and it appears the additional memory capacity would be required by a relatively small number of problems. A high-speed memory capacity of 1024 words has therefore been selected as the best compromise between the mathematical requirement, and the requirement that the EDVAC shall be much smaller than the ENIAC.

With the memory capacity determined, it is possible to fix the word length, by also taking into account the method of coding numerical information, the method of coding orders, the necessary switching time per minor cycle, and the number of digits required for the desired degree of precision. As indicated in the preceding sections of this report, the EDVAC is a binary computer, and a four address code, with a type of operation order appended, has been selected because of its flexibility. It has been found that four digit spaces switching time is required each minor cycle. Experience with other large scale computers has indicated that the number of decimal digits per number entering or leaving the machine should be on the order of 10, and it is preferable to have a few more digit places available within the machine, to ease the scaling problem and decrease round-off errors.

Since 1024 is 2^{10}, 40 binary digits are required to specify the four addresses in the memory. Four binary digits are used in the type of operation order. Adding these digits to the switching time, we arrive at a total word length of 48 digit spaces. For reasons discussed in a later paragraph, it has been found desirable to operate at a rate of 1 binary digit per microsecond,

so that the word length is 48 microseconds. It will be noted that the 44 usable binary digit spaces per word permit the use of 10 digit decimal numbers (binary coded) with sign, in the input and output, and that within the machine, signed 43 binary digit numbers are used, corresponding approximately to signed 13 decimal digit numbers.

The amount of equipment required for a given number of words in the memory depends upon several factors. Since each mercury column must be provided with transducing, amplifying, and selecting equipment, it is economical to store as many words as possible in each tank. The attenuation per unit length of tank is small in comparison with the coupling losses. On the other hand, information stored within the tanks can only be available when it reaches the end of the tank, so that the time wasted in waiting for a word to appear at the output of the tank may appreciably decrease the speed of computation. Assuming serial operation, since the mercury memory logically operates in this manner, the average "waiting time", or time required to remove a word from a tank is $\frac{n+1}{2} w$ microseconds, where n is the number of words in a tank, and w is the length of the word in microseconds. From this point of view, it is desirable to have the tanks as short as possible. In addition to this factor, there is a change in transit time, as the temperature coefficient of increase of acoustic velocity is greater than the linear coefficient of thermal expansion of the material of the tank walls. With eight 48 microsecond words storage space in each tank, the permissible temperature tolerance is $\pm 2\frac{1}{2}^{\circ}C$, if the pulses leaving the tank are broadened by a factor of three before being used to gate clock pulses. The average waiting time with this length of tank is only 216 microseconds. With a 16 word tank, the waiting time would be 408 microseconds, and the temperature tolerance would be only $\pm 1\text{-}1/4^{\circ}C$. Since it would be extremely difficult to maintain a large assembly of tanks within this tolerance, it appears that 8 words is the maximum tank length for

thoroughly reliable operation, and the EDVAC memory therefore consists of 128 8-word tanks, located in two temperature-controlled cabinets.

The broadening of the pulses mentioned above is achieved by making the long tank lengths one and one-half microseconds less than a major cycle, and inserting a multi-tapped delay line. This delay line also makes it possible to emit a word to the recirculating system one microsecond early, so that after the delays inherent in its excursion, the word may re-enter the tank in exact synchronism.

Liquid delay lines are used, since this is a straightforward way to avoid interference from transverse waves. Quartz transducers are used because of their stability and low cost, and because X-cut quartz crystals are well suited to the production of compressional waves in the liquid. Of a large number of liquids considered, mercury was found to give the best acoustic match with the quartz crystals, and consequently is used in the EDVAC.

Mercury has three disadvantages; it is dense and expensive, and it is contaminated by most metals, producing a powdery deposit on the quartz surfaces, which greatly increases the coupling loss. In the EDVAC, contamination is completely eliminated by the use of glass tubes with tungsten electrodes. Weight and cost are reduced by use of tubes of the smallest diameter consistent with good performance. Measurements have shown that dispersion and wall attenuation effects, which are serious at very small diameters, are reduced only slightly by increasing the tube diameter above 3/8 inch, which is the size selected for the EDVAC. This small diameter also decreases crystal capacitance, which decreases the driving power required.

Tank temperature will be stabilized by enclosing the tanks in heavy extruded U-sections of Dow metal clamped to thick vertical plates of the same material. These plates are mounted back to back, with heating elements between

them. The temperature control system is designed to maintain the temperature well within the $\pm 2\frac{1}{2}^{\circ}C$ tolerance noted above. Thirty-two long tanks are mounted on each plate. The entire assembly is enclosed in a heat insulating case, and a pair of coaxial leads is brought out from each tank to its associated recirculating chassis, which is mounted outside the insulating case where it is cooled by circulation of the ventilating air. Two of these assemblies, mounted in separate cabinets, are used in the EDVAC.

Some consideration was given to the possibility of using a temperature compensating device with each tank, instead of the system for controlling the temperature of the assembly. While attractive in many ways, this would require considerable development, and would have the disadvantage of requiring 128 individual controls, the failure of any of which would incapacitate the system, as compared with the two controls required for the two-bank stabilized system in the EDVAC.

As indicated above, functional operation must be fast enough to give operational speeds comparable with the ENIAC, in spite of the fact that the serial operation of the EDVAC is inherently slower than the parallel operation of the ENIAC. An upper limit to the operation speed is set by the pulse repetition frequency. The frequency selected, one megacycle per second, is determined primarily by the characteristics of the commercially available vacuum tubes used in the machine. Experience with the ENIAC has demonstrated that for long life and reliable operation, tubes should not be operated at average currents or dissipations above half the values for which they are normally rated. Under these circumstances, it is feasible to produce pulses of at least 15 volts amplitude with a rise time of .05 microsecond. The pulse can then be made to have a base width of .3 microsecond, and a crest width of .2 microsecond. A repetition frequency of one megacycle per second then

provides an interval between pulses slightly greater than twice their duration, which is adequate for good discrimination, even when the pulses have been broadened by passage through a number of circuits. Fifteen volts is chosen as the normal pulse amplitude because this is about three times as great as the grid excursion necessary to cover the active characteristics of the tubes at the plate voltages at which they are operated. This overdrive eliminates spurious pulses due to noise or cross-talk. Reduction of factors of safety would permit higher operating speeds.

In all circuit design, emphasis has been placed on reliability, simplicity, economy of vacuum tubes, and to the limited extent to which it is practicable, standardization of circuits. Two circuits, extensively used without variation in the EDVAC, have been built into plug-in units, for ease and speed in maintenance. These are the flip-flop and mercury memory circulating units. Crystal gating circuits are extensively used, to decrease the number of tubes required. All available information indicates that the life of the germanium crystals, at the voltages and currents to which they are subjected in the EDVAC, will be many times that of the tubes they will replace.

In a field as new as electronic digital computer design, it is impossible to set up hard and fast rules of circuit design. Where guiding principles are established, they are subject to modification as more information becomes available, and occasional exceptions are inevitable in order to meet particular requirements. Subject to these qualifications, the following guiding principles in circuit design are believed to be conducive to reliable operation and long component life:

(A) All heater voltages shall be held at 6.2 $\pm$ 0.1 volts.

(B) The difference between cathode and filament potential shall in no case exceed 70 volts.

(C) All voltages for D. C. coupled inverter and buffer stages shall come from the same bleeder.

(D) Mica and paper condensers shall be used at no more than 50% of their rated values. Electrolytic condensers shall not be used at more than 50% of their rated values, and shall not be mounted near any heat dissipating elements.

(E) The current through 1N34 crystals shall not exceed 6 milliamperes.

(F) No operational pulses shall appear on the contacts of switches or relays. Where necessary to switch pulses, switches or relays may be used to control the bias of gate or inverter tubes.

(G) All pulse lines more than 2 feet long shall be terminated by their characteristic impedance.

The checking system and its use are discussed in Section 7 of this report. A number of methods of checking the algebraic operations were considered, and taking into account the maintenance problem, it was concluded that the most satisfactory method was to build two identical algebraic units, carry out all algebraic problems in both in synchronism, and compare results at five points. As described in Section 7, other checks have been provided in the Reader-Recorder and Dispatcher for detecting forbidden orders, numbers in blanks, and other coding and functional errors.

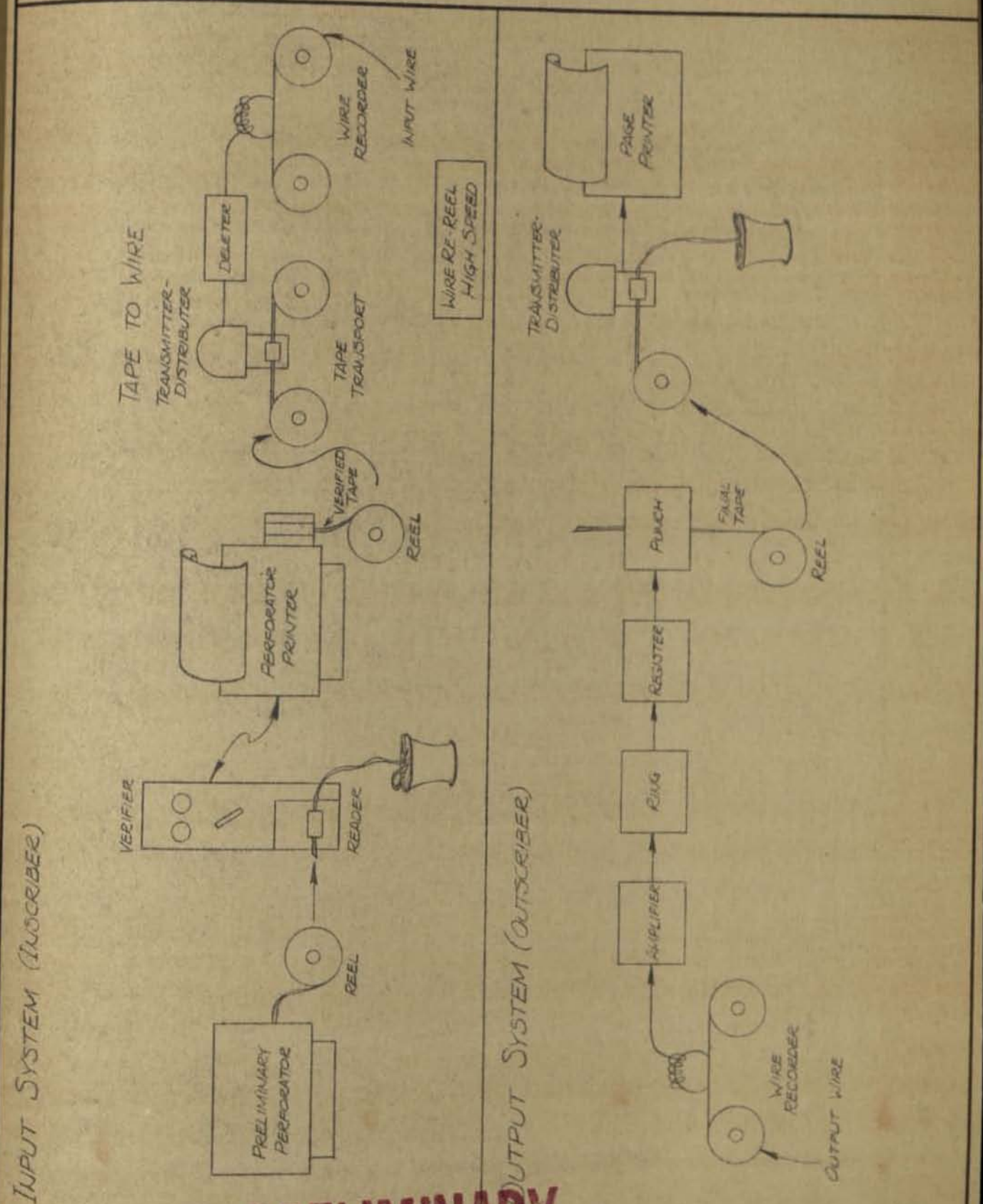
The problem of driving pulse lines has received a great deal of attention. The method used in the ENIAC, the driving of lines by means of cathode followers in parallel, is in general unsuitable for use in the EDVAC because of the large number (on the order of 40) of tubes required to drive certain heavily loaded lines. Two new methods have been devised: blocking oscillators for low duty cycle lines, and power pulse transformer coupled tube banks for high duty cycle lines. These systems require only about one-sixth as many tubes as the earlier circuits.

The use of magnetic wire as the input and output medium for the EDVAC is largely the result of a decision to make extensive use of commercially available equipment in the Input and Output Systems, which are described in the next section of this report. For convenience in coding and to eliminate excessive wire transport three wire drives are used. Although in most cases all three can be used interchangeably, it is often convenient to consider one as primarily for input data, one for use as an intermediate memory, and one for output data. Together, they furnish an auxiliary memory of more than 150,000 words, the equivalent of more than 6,600,000 binary digits.

The EDVAC is housed in steel cabinets 86 inches high. In all except the Memory Units, chassis are mounted vertically, with doors front and back for ready access to both sides of each chassis. This arrangement simplifies maintenance and is convenient for installation of the ventilating system. All air inlets will be located on the sides or backs of cabinets, which makes it possible to build a partition flush with the front panels, and completely isolate the ventilating system. This will eliminate noise and drafts in the operators' space.

Relatively few manual controls are required, since they are used only for starting and stopping the machine, examining the progress of a calculation, modifying a routine, or checking functional operation. Most of these will be mounted on the Control Unit panel, with a few on the Reader-Recorder panel for use in connection with the handling of the wire spools.

UNIVERSITY OF PENNSYLVANIA
MOORE SCHOOL OF ELECTRICAL ENGINEERING



PRELIMINARY

FIGURE 2

APR 7 1948

DRAWN BY J.E.G. 1-30-48	SCALE
CHECKED BY	104-5LA-1
APPROVED BY	

4. Input-Output System

The input-output system is the result of collaboration between the National Bureau of Standards, the Institute for Advanced Study, and the Moore School. It is designed by the Bureau of Standards and, insofar as possible, standard commercial equipment has been used in its construction, particularly Teletype equipment and standard wire recording components. Figure 2 is a block diagram of this system.

It is divided into two parts, the inscriber and the outscriber. The inscriber is used to translate the information prepared by the programmer into a code on the magnetic wire which serves as an input for the EDVAC. The outscriber prints the information which the output of the EDVAC has placed on a magnetic wire, on paper for human consumption.

Two Teletype keyboarding machines are used in the inscriber. The first is called the preliminary perforator and is used to prepare a preliminary "chadless" paper tape by typing the programmer's instructions. Chadless tape is used in order that the character may be printed over the perforations, to assist in checking. This preliminary tape is then fed (either directly, or after being reeled and rewound) into a solenoid-operated tape reader, the output of which is fed through the Verifier to the perforator-printer. This machine will be used, preferably by a different operator, to retype the same code. The output of this machine is fed back to the Verifier. If the character typed agrees with the character on the tape, it is punched in a second tape, not chadless, called the verified tape, and the tape reader advances the preliminary tape to the next character. If the two characters disagree, an indicator lamp lights on the Verifier panel, and the perforator-printer locks until the source of disagreement is eliminated. Provision is made for alteration, deletion,

or addition of a character, as required to place the correct character on the verified tape. No characters are printed on the verified tape, but a page proof is provided for checking against the original code script.

The verified tape is then fed by a tape transport device through a transmitter-distributor, the output of which is fed to the deleter. The deleter removes spaces, and converts the 5-character teletype code into the binary notation required for the input wire.

The output of the deleter is fed into the wire recorder, which records units as positive pulses, and zeros as negative pulses on the wire. It also records marker pulses .6 inch long, to mark the ends of words. These marker pulses furnish the EDVAC reader-recorder with positioning information and the dispatcher with word counting information. A space of at least .1 inch is left between the end of a marker pulse and the first digit pulse. If the 5-address code is being used, 54 digits will be recorded, and the space between the end of the word and the marker pulse will be about .17 inch. If the 4-address code is used, only 44 digits are recorded, and a wide space (about .37 inch) is left at the end of the word. The pulses themselves are about .01 inch long, with spaces of the same length. When recording is complete, a high-speed rewind prepares the wire for the EDVAC. "Blank" wires for the EDVAC auxiliary memory and output require the markers, which are placed on them by running them through the wire recorder with no input signals.

The output wire from the EDVAC has only 44 pulses per word. The pulses are slightly wider than the input pulses, and the end spaces are smaller than on the input wire. These pulses are read by a wire reader, similar to the wire recorder, amplified and assembled

in groups of four in electronic circuits. These circuits control a parallel-type punch which perforates the final tape. Since only sixteen different signals are provided, the output tape cannot provide the complete teletype (5-hole) code.

The final tape is passed through a transmitter-distributer which operates a page printer. In view of the limited code used, this printer can only print 16 different characters. Ten of these characters are the conventional decimal digits and 2 are the symbols +, -. Ordinarily an output wire to be printed, since it will contain decimal data only, needs only these 12 characters. The other four characters are arbitrary, and only needed to print out wires which have not been converted to coded-decimal data, or to detect errors in supposedly coded-decimal data.

5. Performance Details

5.0 Numerical Interpretation of Words

The method of performing the operations, or order-types, A, S, M, D, C, has already been outlined. The eleven different order-types will now be considered in more detail. It will be recalled that each order consists of 4 addresses, of ten characters each, and an order-type of 4 characters. In order to be scrupulously accurate in describing the results of performing an order, without indulging in clumsy locutions, a carefully considered symbolism is essential. We have already introduced the functions $_1\bar{A}(\bar{w})$, $_2\bar{A}(\bar{w})$, $_3\bar{A}(\bar{w})$, $_4\bar{A}(\bar{w})$ to describe the four addresses in an order, and $\bar{T}(\bar{w})$ to describe the order type.

In the sequel, a numerical interpretation of these segments is important, because certain operations on addresses performed in the dispatcher are most easily explained if the addresses are numerically interpreted. Taking a broader view, this numerical interpretation is important for two additional reasons. In the first place, both the inscriber keyboard and the manual control switches are such as to make it convenient (and in some cases, necessary) to introduce the address $(_j\bar{A})$ and the order-type $(\bar{T})$ in numerical form.

Secondly, since the EDVAC stores orders and numbers in a common reservoir, it is possible to manipulate orders by passing them thru the computer. This is the most striking difference between the EDVAC and earlier large-scale computers, since it opens up an exceedingly wide range of programming possibilities. But if orders are to be treated as numbers, the relation between the numerical and the order interpretation of a word must be made explicit.

The function $n({}_j\bar{A})$ is introduced:

$$n({}_j\bar{A}) = \sum_{i=1}^{i=10} 2^{(10-i)}\ {}_jA_i \ ; \text{ where } {}_jA_i = 0, 1; \ j = 1, 2, 3, 4.$$

This gives a numerical identification for every address and preserves the interpretation of the characters of our languages as zeros and ones. Since $0 \leq n \leq 1023$, the memory positions in the EDVAC will be considered as being identified by these values of n. The characters in an address are considered to be the successive digits of the binary expansion of n. As stated in section 2, the characters ${}_jA_1 - {}_jA_7$ identify the long tank, thus

$$l({}_j\bar{A}) = \sum_{i=1}^{i=7} 2^{(7-i)}\ {}_jA_i \ , \text{ where } {}_jA_i = 0, 1 \ ; \ j = 1, 2, 3, 4.$$

The first seven characters in an address $({}_j\bar{A})$ are thus the most significant digits of n. We now have the numerical identification for the long tanks. Since $0 \leq l \leq 127$, the long tanks in the EDVAC will be considered as being identified by these values of l.

Finally the characters ${}_jA_8 - {}_jA_{10}$ identify the minor cycle (modulo 8), thus

$$m({}_j\bar{A}) = \sum_{i=8}^{i=10} 2^{(10-i)}\ {}_jA_i, \text{ where } {}_jA_i = 0, 1; \ j = 1, 2, 3, 4.$$

The last three characters of an address $({}_j\bar{A})$ are the least significant digits of n. The numerical identification for minor cycles will follow from the fact that $0 \leq m \leq 7$. The minor cycle counter in the EDVAC is designed to count forward so that the minor cycles occur in the natural sequence 0, 1, 2, ..., 7. In planning problems, particularly for minimum waiting time, and in order to utilize the inscriber keyboard most efficiently, expansions to the base 8 will be normally used to represent memory positions. Octonary (octal) digits are denoted by

$\bar{0}$, $\bar{1}$, $\bar{2}$, $\bar{3}$, ..., $\bar{7}$. In this system, $\overline{0000} \leq n \leq \overline{1777}$; $\overline{000} \leq \ell \leq \overline{177}$; and $\bar{0} \leq m \leq \bar{7}$. Since one octonary (octal) digit is the equivalent of three binary digits, and since there are only 10 characters in an address $(j\bar{A})$, it will be convenient to replace the $\bar{1}$ by $\underline{1}$, to represent a binary digit. These characters all appear on the inscriber keyboard. In the octonary (octal) system, the least significant digit of n identifies the minor cycle, and the other digits identify the long tank.

When performing certain types of orders, the dispatcher modifies the addresses $(j\bar{A})$ by changing their characters in such a way that $n(j\bar{A})$ increases by unity. Numerical operations on orders in the computer can also be interpreted as changing $n(j\bar{A})$. This raises the problem of determining the memory position corresponding to an arbitrary integer, that is, of establishing the inverse function $n^{-1}(\phi(n(j\bar{A})))$, where ϕ is some numerical operation on n. If n is 1023 and ϕ is n + 1, then $\phi(n)$ becomes 1024, which corresponds to no address $(j\bar{A})$ since it requires 11 binary digits instead of 10. In order to take care of this, we define $N(j\bar{A})$ as the residue class of integers congruent to $n(j\bar{A})$ modulo 1024. The function $N(j\bar{A})$ now has a single-valued inverse, and the values of n will be used as the standard representation of the equivalence classes which constitute the range of the function N. For example if N = 1023, N + 1 = 0.

Finally we introduce the function P to designate the memory position identified by an address. This represents an actual position in the high-speed memory, and not a mere number.

$$P(j\bar{A}) = P(n(j\bar{A})) = P(N(j\bar{A}))$$

thus if $n(j\bar{A}) = 1336$, P is the 6th position (minor cycle) in the 91st long tank. At any fixed time $\vec{w}(P)$ is the word stored in position P.

$\overline{w}({}_j\overline{A})$ is the word identified by the address ${}_j\overline{A}$. A numerical interpretation of $\overline{T}(\overline{w}) = (c_{41}, c_{42}, c_{43}, c_{44})$ is also convenient.

$$n(\overline{T}) = \left(\sum_{i=41}^{i=43} 2^{(44-i)} c_i\right)(-1)^{c_{44}}, \text{ where } c_i, c_{44} = 0, 1.$$

The interpretation of the characters as zeros and ones is preserved. Since $-7 \leq n \leq +7$, the order types may be identified as signed octonary (octal) digits. Since the sign occurs at the right, it will be appropriate to write these digits with the sign in the same position, e.g. $\overline{7}-$, $\overline{3}+$. The ambiguity of +0 and -0 is resolved by making both of these unused orders.

The eleven fundamental order types will now be specified, roughly in order of increasing complexity. The alphabetic symbol for the order type, which appears on the control switches, and the numerical symbol, which is used when keyboarding an order in the inscriber, appear in parentheses.

5.01 (V, $\overline{1}-$) Visual Display (provision for future use)

The first execute copies $\overline{w}({}_1\overline{A})$, the word specified by the first address into a short tank, the second execute presents a copy of $\overline{w}({}_2\overline{A})$, the word specified by the second address and simultaneously with it, the memorized $\overline{w}({}_1\overline{A})$ from the short tank, on two output terminals. ${}_3\overline{A}$ is not executed, that is, used to select a memory position, but is decoded to give another signal which is intended to select one of three output scopes:

If $n({}_3\overline{A}) \equiv 1 \bmod 4$, flip flop 1 is set;

If $n({}_3\overline{A}) \equiv 2 \bmod 4$, flip-flop 2 is set;

If $n({}_3\bar{A}) \equiv 3 \bmod 4$, flip flop 3 is set;

if $n({}_3\bar{A}) \equiv 0 \bmod 4$, no flip-flop is set.

In other words, the least significant octonary (octal) digit in $n({}_3\bar{A}), D$, is intended to select the scope:

$D = \bar{1}$ or $\bar{5}$ selects scope 1,

$D = \bar{2}$ or $\bar{6}$ selects scope 2,

$D = \bar{3}$ or $\bar{7}$ selects scope 3,

$D = \bar{4}$ or $\bar{0}$ selects no scope.

The other digits in $n({}_3\bar{A})$ are thus not significant. In order to complete the installation required for performing this type of order, it will be necessary to provide 3 long persistence scopes to be turned on and off by the flip-flops, and a pair of digital to continuous conversion devices (pulse-modulation demodulators) to transform the pulses of the words into voltages for the scope deflection plates. When installed, this equipment will permit visual curve-tracing as the problem proceeds. The last execute copies the contents of the memory position identified by ${}_4\bar{A}$, i.e. $w({}_4\bar{A})$, into the dispatcher to enable the program sequence to continue. It should be noted this order does not alter the contents of the high-speed memory in any way.

5.02 (H, $\bar{6}$+) Halt

When this type of order reaches the control, the EDVAC ceases all computation and gives an audible and a visual signal. The power remains on and the machine is ready to continue. If the initiate button is depressed, the contents of the memory position identified by ${}_4\bar{A}$ (the new order) is copied into the dispatcher and the EDVAC resumes operation. ${}_1\bar{A}$, ${}_2\bar{A}$, ${}_3\bar{A}$ are dummies in the halt order and have no meaning or effect on the operation of the EDVAC. Again, it should be

noted that this order does not alter the contents of the high speed memory in any way.

5.03 (A, $\bar{2}$+) Add

This is a typical order which uses all four addresses for transfer operations and which produces a numerical result which is sent to the memory. In order to explain the nature of the computer output, we must discuss this problem: Given a number, x, what word is to represent it? The inverse question of determining what number is represented by a word has already been settled by the conventions established in section 2. First of all it is essential that x be a proper digital number $\bar{x}$, i.e., lie in the range $-1 + 2^{-43} \leq x \leq +1 - 2^{-43}$, and be of the form $x = N\,2^{-43}$, where N is some integer (not necessarily positive.) Assuming this has been properly taken care of, what sign (+ or -) are we to choose when x = 0?

The partitioning of a word into 5 segments by the dispatcher has been explained in section 2; in order to clarify the discussion of computer operation it is convenient to recognize smybolically what actually occurs in the computer circuits. The computer separates the sign from the digits of a number and handles them separately. A word is thus divided by the computer into two segments, the first containing 43 characters, and the last, a single character. (It should be continually borne in mind that first and last do not refer to the flow of characters in the spoken language, which emerge temporally in reverse order). Two new segment-functions of a word are introduced, analagously to $_1\bar{A}$, $_2\bar{A}$, $_3\bar{A}$, $_4\bar{A}$, and $\bar{T}$:

$$D(\bar{w}) = (c_1, c_2, \ldots, c_{43}),$$

$$S(\bar{w}) = c_{44}.$$

In order to be consistent with the numerical interpretation of words described in section 2, the following numerical interpretation of these word-segments is required:

$$\left|\bar{x}\right| \quad (D(\bar{w})) = \left|\bar{x}\right| (\bar{w}) = \sum_{i=1}^{i=43} 2^{-i} c_i ; \text{ where } c_i = 0, 1;$$

$$\sigma(S(\bar{w})) = \sigma(\bar{w}) = (-1)^{c_{44}} \quad \text{where } c_{44} = 0, 1.$$

The interpretation of the characters as zeros and ones is preserved, and since $\left|\bar{x}\right| (\bar{w}) = \left|\bar{x}(\bar{w})\right|$, $D(\bar{w})$ represents the absolute value of the number, that is, the unsigned digits. $\sigma(\bar{w})$ is a multiplying factor which determines the sign, or sense, of the numerical interpretation of the word:

$$\bar{x}(\bar{w}) = \left|\bar{x}\right| (\bar{w}) \cdot \sigma(\bar{w}) = \left(\sum_{i=1}^{i=43} 2^{-i} c_i \right) (-1)^{c_{44}} .$$

This is identical with the numerical word-interpretation of section 2.

Now we are prepared to settle the problem of encoding proper digital numbers into words. The digits give no difficulty, since $\left|\bar{x}\right| (\bar{w})$ has an inverse, that is, it is unique, and certainly exists if x is properly digital. There are exactly 2^{43} different values of $\left|\bar{x}\right|$, and exactly 2^{43} different kinds of word-segments D. We encode the digits of $\bar{x}$ by setting $D = \left|\bar{x}\right|^{-1} \left|\bar{x}\right|$.

The problem of determining S, given $\bar{x}$, is slightly more subtle. In a computer which operates with the usual notation instead of using complements, the double representation of -0 and +0, must be carefully accounted for in the logical design. The obvious choice for the function S'(x) is closely related to the function sgn(x):

$$\operatorname{sgn}(x) = \begin{cases} +1, & x > 0; \\ 0, & x = 0; \\ -1, & x < 0. \end{cases}$$

Unfortunately sgn (x) has three values in its range, which runs counter to our binary language principles. On the other hand, sgn x = 0 if and only if x = 0, hence the difficulty only arises in this isolated case. In order to circumvent this deficiency in our language we more or less arbitrarily throw zero in with the positive numbers; instead of the trichotomy, positive, zero, negative, we have the dichotomy, non-negative, negative. A new function, $\operatorname{sgn}_+(x)$ is defined:

$$\operatorname{sgn}_+(x) = \begin{cases} +1, & x \geq 0 \ ; \\ -1, & x < 0 \ . \end{cases}$$

which gives

$$(1 - \operatorname{sgn}_+(x))/2 = \begin{cases} 0, & x \geq 0; \\ 1, & x < 0. \end{cases}$$

in order to preserve the interpretation of the characters as zeros and ones. We now have a function which enables us to encode the sign of zero uniquely:

$$S'(\bar{x}) = \begin{cases} \text{pulse}, & \text{if } \dfrac{(1 - \operatorname{sgn}_+(\bar{x}))}{2} = 1; \\ \text{no pulse}, & \text{if } \dfrac{(1 - \operatorname{sgn}_+(\bar{x}))}{2} = 0. \end{cases}$$

If $\bar{x}$ is non-negative we assign c_{44} in such a way that $\bar{w}$ has a plus sign, if $\bar{x}$ is negative, a minus sign. The computer will decode either -0 or +0 as 0, but it will encode 0 only as +0. This is of special importance in the compare order as will be seen later. The method of encoding $\bar{x}$, a number in proper digital form is thus

$$\bar{w} = \bar{w}(\bar{x}) = |\bar{x}|^{-1} \, |\bar{x}| \, \overset{\frown}{S'}(\bar{x}),$$

where the arch signifies concatenation or justaposition.

$|\bar{x}|^{-1}(\bar{x})$ is the segment D(w) containing characters c_1-c_{43}, the "digits" of w, and $S^1(x)$ is the character c_{44}, the "sign" of w. Such a representation of a computer output will be properly decoded if the word is returned to the computer, since

$$\begin{aligned}\bar{x}(\bar{w}) = \bar{x}(\bar{w}(\bar{x})) &= |\bar{x}|(\bar{w}(\bar{x})) \cdot \sigma(\bar{w}(\bar{x})) \\ &= |\bar{x}|(D(\bar{w}(\bar{x})) \cdot \sigma(\bar{w}(\bar{x})) \\ &= |\bar{x}|(|\bar{x}|^{-1}|x|) \cdot \sigma(\bar{w}(\bar{x})) \\ &= |\bar{x}| \cdot \sigma(\bar{w}(\bar{x})) \\ &= |\bar{x}) \cdot \sigma(S(\bar{w}(\bar{x}))) \\ &= |\bar{x}| \cdot \sigma(S'(\bar{x})) \\ &= |\bar{x}| \cdot (-1)^{(1-\mathrm{sgn}_+(x))/2} = \bar{x}.\end{aligned}$$

The rules for encoding proper digital numbers having been established, we now direct our attention to the effect of performing rational operations, in this case, addition, on such numbers. When we program an add order we signify our intention of adding the contents of the memory position identified by $_1\bar{A}$ to the contents of the memory position identified by $_2\bar{A}$:

$$\bar{x}(\bar{w}(_1\bar{A})) + \bar{x}(\bar{w}(_2\bar{A})) = \sum .$$

$\sum$ is not, in general, a proper digital number. Since max $|\bar{x}| = 1 - 2^{-43}$, max $|\sum| = 2 - 2^{-42}$, and $-2 + 2^{-42} \leq \sum \leq 2 - 2^{-42}$. Since $\bar{x}$ varies only by increments of 2^{-43}, $\sum$ must do likewise. $\sum$ thus satisfies the digital requirement that it be of the form $N\,2^{-43}$, but it may lie outside of the range stipulated for $\bar{x}$, and hence fail to be a proper digital number for that reason. There are $1 + (2-2^{-42} - (-2+2^{-42}))/\,2^{-43} = 2(2^{44}) - 3$ different possible values of $\sum$. Since there are only 2^{44} different kinds of words possible in the memory, and there are almost

twice as many values of Σ, it is clearly impossible to store Σ in the memory without modifying it in some cases. If we expand Σ in the binary system:

$$\Sigma = \left(\sum_{i=0}^{i=43} 2^{-i} \delta_i \right) (-1)^{\delta_{44}},$$ where δ_i, $\delta_{44} = 0, 1$, and at least one of the δ_i is zero.

If it were not for the δ_o, Σ could be expressed as a word in the same way that $\overline{x}$ can. The computer, emulating Procrustes, trims off the δ_o. If $|\Sigma| \geq 1$, we subtract 1 from it, which is equivalent to discarding δ_o. If such is the case, the computer gives a signal to the dispatcher, known as "add-subtract capacity exceeded". The complete significance of this will be described later; this signal makes it possible to modify the subsequent operation of the EDVAC. In any case, the computer output is not modified by this circumstance and it remains to describe it analytically. In order to do this conveniently the function "greatest integer in x", $[x]$, is employed. For our purposes it is sufficient to consider only positive values of x:

$$[x] = \begin{cases} 0, & 0 \leq x < 1, \\ 1, & 1 \leq x < 2, \\ \vdots & \vdots \\ n, & n \leq x < n+1. \end{cases}$$

$|\Sigma| - [|\Sigma|]$ will give the fractional part of $|\Sigma|$ only, and this operation is the analytical equivalent of the syntactical operation of suppressing the supernumerary digit at the left, δ_o; in other words this is a mathematical expression for the physical operation corresponding to rubbing out δ_o with an eraser.

The proper digital number encoded by the computer is $\overline{\Sigma} = (|\Sigma| - [|\Sigma|]) \cdot \operatorname{sgn} \Sigma$. This number is not encoded strictly in

accordance with the principles previously stated; if $\sum = -1$, $\overline{\sum} = 0$, but $\overline{w}\left(\overline{\sum}\right)$ will be a "minus zero", that is, S'(x) is determined from $\sum$, rather than from $\overline{\sum}$. If we have an exceed capacity signal, examination of the sign (c_{44}) of the contents of the memory position identified by $_3\bar{A}$, will indicate whether the result is incorrect by +1 or -1.

The complete discussion of the conversion of $\sum$ into a proper digital number has resulted in almost losing sight of the broader outlines of the add order, which will now be summarized. The first execute sends $\overline{w}(_1\bar{A})$ to a computer short memory tank and returns $\overline{w}(_1\bar{A})$ to the memory, via the recirculating system. The second execute sends $\overline{w}(_2\bar{A})$ direct to the adding circuits, returns $\overline{w}(_2\bar{A})$ to the memory, and sends the memorized $\overline{w}(_1\bar{A})$ to the adding circuits. At the end of one minor cycle $\sum$, the algebraic sum of $\bar{x}(\bar{w}(_1\bar{A}))$ and $\bar{x}(\bar{w}(_2\bar{A}))$ has been computed, and when the third execute occurs $\overline{w}(_3\bar{A})$ is replaced by $\overline{w}\left(\overline{\sum}\right)$, thus storing the sum in the memory position identified by the third address in the order, and displacing the original contents of that position. In case $\left|\sum\right| \geq 1$, an "add-subtract capacity exceeded" signal is sent to the dispatcher, to be discussed later, but in any case, a proper digital number is stored in the memory by dropping the 1 to the left of the binary point, and storing only the "fractional" part of $\sum$. The sign of this number agrees with the sign of $\sum$, even when $\overline{\sum}$ is zero. The fourth execute normally sends $\overline{w}(_4\bar{A})$ to the dispatcher, constituting the next order in the program chain.

5.04 <u>(S, $\bar{3}$+) Subtract</u>

This operation is similar to add. If we program a subtract order

we signify our intention of subtracting the contents of the memory position identified by $_2\overline{A}$ from the contents of the memory position identified by $_1\overline{A}$:

$$\overline{x}(\overline{w}(_1\overline{A}) - \overline{x}(\overline{w}(_2\overline{A})) = \Delta \quad .$$

Δ gives rise to an extra digit at the left, exactly as $\sum$ does. The proper digital number encoded by the computer is $\overline{\Delta} = \left(|\Delta| - [|\Delta|]\right) \cdot \operatorname{sgn} \Delta$, and if $\Delta = -1$, $\overline{\Delta} = 0$, and $\overline{w}\left(\overline{\Delta}\right)$ will be a "minus zero". If $|\Delta| \geq 1$, the computer emits an "add-subtract capacity exceeded" signal, as in the previous order.

The first execute transfers $\overline{w}(_1\overline{A})$ to a computer short memory tank and returns $\overline{w}(_1\overline{A})$ to the memory, via the recirculating system. The second execute transfers $\overline{w}(_2\overline{A})$ direct to the adding circuits, returns $\overline{w}(_2\overline{A})$ to the memory, and transfers the memorized $\overline{w}(_1\overline{A})$ to the adding circuits. At the end of one minor cycle, Δ , the algebraic difference, has been computed, and when the third execute occurs, $\overline{w}(_3\overline{A})$ is replaced by $\overline{w}(\overline{\Delta})$, thus storing the difference in the memory position identified by the third address in the order, and displacing the original contents of that position. In case $|\Delta| \geq 1$, an "add-subtract capacity exceeded" signal is sent to the dispatcher, but in any case, a proper digital number is stored in the memory by dropping the 1 to the left of the binary point, and storing only the fractional part of Δ . The sign of this number agrees with the sign of Δ even when $\overline{\Delta}$ is zero. The fourth execute normally sends $\overline{w}(_4\overline{A})$ to the dispatcher, constituting the next order in the program chain.

5.05 (C, $\overline{1}$+) Compare

The first two executes accomplish the same results as in the subtract order. The result, $\overline{w}\left(\overline{\Delta}\right)$, is immediately sent to the dispatcher

as soon as it has been computed. The dispatcher takes $S(\overline{w}(\overline{\Delta}))$, (the sign of $\overline{\Delta}$) and uses it to select ${}_3\overline{A}$ (the third address), or ${}_4\overline{A}$ (the fourth address) for the next execute. If S is a pulse, $\Delta < 0$, and the third address is chosen, if S is a non-pulse $\Delta \geq 0$, and the fourth address is chosen. $\overline{w}({}_j\overline{A})$, where $j = 3, 4$ is sent to the dispatcher on the last execute, constituting the next order in the program chain. In other words, if the number in the memory position identified by ${}_1\overline{A}$ is greater or equal to the number in the memory position identified by ${}_2\overline{A}$, then the order in the memory position identified by ${}_4\overline{A}$, will become the next order, while if $\overline{x}(\overline{w}({}_1\overline{A}) < \overline{x}(\overline{w}({}_2\overline{A}))$, then $w({}_3\overline{A})$ becomes the next order. The reason for giving $\Delta = -1$ a minus sign, even though $\overline{\Delta} = 0$, is apparent. We do not wish the dispatcher to think that the numbers are equal, if their difference is exactly -1. Similarly, $\Delta = 0$ has a plus sign, since if it had a minus sign, the dispatcher would infer that $\overline{x}(\overline{w}({}_1\overline{A})) < \overline{x}(\overline{w}({}_2\overline{A}))$.

The compare order enables the EDVAC to make a choice as to the next operation to be performed, which can depend on the results of previous computation, since either or both of $\overline{w}({}_1\overline{A})$ and $\overline{w}({}_2\overline{A})$ may be the results of previous computations, and $\overline{w}({}_3\overline{A})$ and $\overline{w}({}_4\overline{A})$ may be entirely different orders. The compare order does not change the memory contents in any way, $\overline{w}({}_3\overline{A})$ or $\overline{w}({}_4\overline{A})$ is returned to the memory via the recirculating system. The last execute cannot follow the second execute until at least 2 minor cycles have elapsed, since the address ${}_3\overline{A}$ or ${}_4\overline{A}$ cannot be checked against the minor cycle counter until after $\overline{w}(\overline{\Delta})$ arrives in the dispatcher.

5.06 (M, $\bar{L}$+) Multiply with round-off

The first execute transfers $\overline{w}(_1\bar{A})$ to a computer short memory tank, and returns it to the memory via the recirculating system. The second execute transfers $\overline{w}(_2\bar{A})$ to the same computer short memory tank, sends the memorized $\overline{w}(_1\bar{A})$ to a second computer short memory tank, and begins the serial process of multiplication, which requires 43 minor cycles since there are 43 digits in the multiplier, $\overline{w}(_2\bar{A})$. The computer forms the product $\overline{x}(\overline{w}(_1\bar{A}) \cdot \overline{x}(\overline{w}(_2\bar{A})) = \overline{\Pi}$. $\overline{\Pi}$ is not, in general, a proper digital number. Since max $|\overline{x}| = 1 - 2^{-43}$, max$|\Pi| = 1 - 2^{-42} + 2^{-86}$; Π falls in the same range as $\overline{x}$, hence there is no problem of extra digits at the left, i.e., no exceed capacity problem. On the other hand Π does not satisfy the digital requirement that it be of the form $N \cdot 2^{-43}$ and hence fails to be a proper digital number because of supernumerary digits at the right. If we expand $|\Pi|$ in the binary system, 86 digits will be required, in general:

$$|\Pi| = \sum_{i=1}^{i=86} 2^{-i}\,\delta_i \text{, where } \delta_i = 0,1$$

Indeed max $|\Pi|$ is a number which actually requires the full 86 digits: if $1 \leq i \leq 42$, $\delta_i = 1$, if $43 \leq i \leq 85$, $\delta_i = 0$, and $\delta_{86} = 1$. Hence there is a significant digit at each end of the expansion. That 86 digits are always sufficient follows from the fact that the smallest non-zero increment of Π is $(2^{-43})^2 = 2^{-86}$. In order to reduce $\overline{\Pi}$ to proper digital form, a wholesale trimming of digits is necessary. In view of the standard representation of numbers, digits 44-86 must be discarded. One way of doing this is described analytically by the operation

$$\overline{|\pi|} = \left[|\pi| 2^{43}\right] \cdot 2^{-43}.$$

This corresponds to translating the "point" to a position between the 43d and 44th digits, dropping all fractional digits (by selecting the greatest integer), and then restoring the point to its original position. Since $\overline{|\pi|} = N \cdot 2^{-43}$, it satisfies one of the proper digital requirements, also, since $\overline{|\pi|} \leq |\pi|$ and $\max |\pi| = 1 - 2^{-42} + 2^{-86}$, then $1 + 2^{-43} < \overline{\pi} < 1 - 2^{-43}$, and $\overline{\pi}$ certainly falls in the proper range. As a matter of fact,

$$\begin{aligned} \max \overline{|\pi|} &= \left[\max |\pi| \cdot 2^{43}\right] \cdot 2^{-43} \\ &= \left[(1 - 2^{-42} + 2^{-86}) 2^{43}\right] \cdot 2^{-43} \\ &= \left[2^{43} - 2 + 2^{-43}\right] \cdot 2^{-43} \\ &= (2^{43} - 2)\, 2^{-43} = 1 - 2^{-42}, \end{aligned}$$

and $-1 + 2^{-42} \leq \overline{\pi} \leq 1 - 2^{-42}$. $\overline{\pi}$ is certainly a proper digital number suitable for encoding.

On the other hand, $\overline{\pi}$ has a serious deficiency. Since $\overline{|\pi|} \leq |\pi|$, this operation is biased towards zero; $\overline{|\pi|}$ is always too small. If $|\pi|$ is assumed to be a random real number lying in the range $0 \leq |\pi| < 1$, then the error is uniformly distributed in the interval $-2^{-43} < e \leq 0$, with expected value $E(e) = 2^{-44}$; least upper bound of absolute value, l.u.b. $|e| = 2^{-43}$; variance, $\sigma^2(e) = (1/12)2^{-86}$. In order to make $E(e) = 0$, to remove the bias, and to decrease l.u.b. $|e|$, it is better to add 2^{-44} before dropping digits, thus centering the error. The EDVAC computer is designed to perform the round off operation

$$\overline{|\pi|}^{*} = \left[\left(|\pi| + 2^{-44}\right) 2^{43}\right] \cdot 2^{-43} = \left[|\pi| 2^{43} + \tfrac{1}{2}\right] \cdot 2^{-43}.$$

This corresponds to translating the "point" to a position between the 43d and 44th digits, adding 1/2 (and forming all carries), dropping all fractional digits (by selecting the greatest integer), and then restoring the point to its original position. The error is now uniformly distributed in the interval $-2^{-44} < e \leq +2^{-44}$, with expected value $E(e) = 0$, $\max |e| = 2^{-44}$, and variance $\sigma^2(e) = (1/12)\, 2^{-86}$.

Other methods of rounding-off are possible which eliminate the bias but increase the variance. In the EDVAC computer, no extra time is required to add in the 2^{-44} required for round-off. The last time the partial product passes thru the adding circuits, a fictitious carry from a non-existent digit in the 2^{-45} positional value is introduced into the circuit in such a way that three numbers are added simultaneously. The operation is still performed in 43 minor cycles.

$\overline{|\pi|^*}$ is clearly of the form $N \cdot 2^{-43}$ since $\left[|\pi| 2^{43} + 1/2\right]$ is an integer, by definition. The most difficult restriction required if a number is to be proper digital has been met. But although $|\pi| < 1 - 2^{-43}$, the round-off to $\overline{|\pi|^*}$ may possibly create an extra digit at the left. Fortunately, this is not the case. Since $\max |\pi| = 1 - 2^{-42} + 2^{-86}$, $\max \overline{|\pi|^*} = \left[(1 - 2^{-42} + 2^{-86} + 2^{-44}) 2^{43}\right] \cdot 2^{-43}$ $= 1 - 2^{-42}$. Even after round-off, $\max \overline{|\pi|^*} < \max |\bar{x}|$; we never can have a complete string of ones as the result of a multiplication.

It remains to describe completely $\bar{w}(\overline{\pi^*})$, the word encoded by the computer. The proper digital number to be encoded is

$$\overline{\pi^*} = \overline{|\pi|^*} \cdot \operatorname{sgn} \pi + \overline{\pi^*}.$$

The resulting word is

$$w(\overline{|\Pi|^*}) = |\bar{x}|^{-1}\ \overline{|\Pi|}^*\ S'(\overline{\Pi\Pi^*}).$$

$|\bar{x}|^{-1}\ \overline{|\Pi|}^*$ is the segment $D(\bar{w})$ containing characters $c_1 - c_{43}$, the digits of $\overline{|\Pi\Pi|}^*$, which is $|\Pi\Pi|$ rounded off to 43 binary digits, and $S'(\overline{\Pi\Pi}^*)$ is the character c_{44}, the sign of $\overline{\Pi\Pi}^*$, which is not necessarily the sign of $\Pi\Pi$. If $\overline{\Pi\Pi}^*$ is zero, it is encoded as a +0 in accordance with the usual conventions, even though $\Pi\Pi$ may be in the interval $-2^{-44} < \Pi\Pi < 0$.

The third execute, which may occur no earlier than 43 minor cycles after the second execute, replaces $\bar{w}(\bar{A}_3)$ by $\bar{w}(\overline{\Pi\Pi}^*)$, thus storing the rounded-off product in the memory position identified by the third address in the order, and displacing the original contents of that position. Finally, the fourth execute sends $\bar{w}(\bar{A}_4)$ to the dispatcher, constituting the next order in the program chain.

5.07 (m, $\bar{4}$-) Exact multiplication

In this operation, the computer forms the product $|\Pi\Pi|$ as before. In this case, however, no round-off occurs and $|\Pi\Pi|$ is split into 2 proper digital numbers. The first is $\overline{|\Pi\Pi|}$ as previously described in connection with multiplication with round-off. $\overline{|\Pi\Pi|}$ is properly digital and contains the 43 digits of $|\Pi\Pi|$ having the greater positional value. The "point" is in the proper position in $\overline{|\Pi\Pi|}$. The other number contains the digits of $|\Pi\Pi|$ having the lesser positional value:

$$|\pi| = \left(|\Pi| - \overline{|\Pi|}\right).$$

This is not a proper digital number unless $|\pi| = 0$; it has the proper number of digits, but the point is in the wrong place. It is necessary instead to use

$$\overline{|\pi\, 2^{43}|} = \left(|\Pi| - \overline{|\Pi|}\right) 2^{43}.$$

This must be a proper digital number, a fact which can be demonstrated analytically.

$\left(|\pi| - \overline{|\pi|}\right)2^{43} = |\pi|2^{43} - \left[|\pi|\,2^{43}\right]$, and since $0 \leq x - [x] < 1$, $\overline{|\pi 2^{43}|}$ is in the correct range, provided that it does not exceed $1 - 2^{-43}$. We already know that $|\pi|$ varies only by increments of 2^{-86} and hence $|\pi|2^{43}$ varies only by increments of 2^{-43}. $\left[|\pi|2^{43}\right]$ varies only by increments of unity since it is an integer. Consequently $\overline{|\pi 2^{43}|}$ varies only by increments of 2^{-43}, and hence lies in the range $0 \leq \overline{|\pi 2^{43}|} \leq 1 - 2^{-43}$. It is thus a proper digital number.

It only remains to explain the encoding of $\overline{\pi}$ and $\overline{\pi 2^{43}}$ and to decide where they are to be stored in the memory.

$$\overline{w}(\overline{\pi}) = |\overline{x}|^{-1}\overline{|\pi|}\,S'(\overline{\pi}).$$

This number thus may be encoded as a + 0, even though π may be negative, i.e., if $2^{-43} < \pi < 0$.

$$\overline{w}(\overline{\pi 2^{43}}) = |\overline{x}|^{-1}\overline{|\pi 2^{43}|}\,S'(\overline{\pi 2^{43}}).$$

Again this number may be encoded as a +0, even though π may be negative, i.e., if π "comes out even", and digits 44 through 86 are all zero.

The first two executes, and the timing of the third, are the same as for multiplication with round-off. On the third execute, $w(\overline{\pi})$ is transferred to the memory position identified by the third address in the order, displacing the original contents, i.e., $\overline{w}(\overline{\pi})$ replaces $\overline{w}(\overline{A}_3)$. The third memory position designated in the order now contains the 43 digits of π having the greatest positional value. As soon as this execute occurs, the dispatcher, in effect adds unity to $\overline{A}_3$, more precisely it performs the operation $N(\overline{A}_3) + 1$, producing a new address

$N^{-1}(N(\bar{A}_3) + 1)$. This new address has the property $m(N^{-1}(N(\bar{A}_3)+1)) - m(\bar{A}_3) = 1 \bmod 8$, since we have effectively added 1 to the minor cycle number. This address can be immediately executed the next minor cycle, even though $\ell(N^{-1}(N(\bar{A}_3) + 1))$ is not necessarily equal to $\ell(\bar{A}_3)$, that is, the long tanks may be different. On this execute, which follows the previous execute after a lapse of exactly one minor cycle, $\bar{w}(N^{-1}(N(\bar{A}_3)+1))$ is replaced by $\bar{w}(\overline{\pi\, 2^{43}})$. The "next" memory position after the one where $\bar{w}(\overline{\Pi})$ is placed, will receive the 43 digits of Π having the lesser positional value. Roughly speaking, the most significant digits of the product go into the third address, and the least significant digits go in the next address. These two executes take place in succession and no time is wasted. If the more significant digits go into memory position 1023, the least significant go into position 0. The "next" position is defined cyclically.

Four executes have now taken place, with one of the addresses manufactured by the dispatcher. The fifth execute sends $\bar{w}(\bar{A}_4)$ to the dispatcher, constituting the next order in the program chain. Finally, it should be noted that $\Pi = \bar{x}(\bar{w}(\bar{A}_3))+2^{-43}\,\bar{x}(\bar{w}(N^{-1}(N(\bar{A}_3+1))$, so exact multiplication enables all the digits of π to be recovered.

5.08 (D, $\bar{5}$+) Divide with round-off

This operation involves both problems previously encountered, namely, extra digits at the right, and extra digits at the left. Nominally, we wish to perform the operation:

$$\bar{x}(\bar{w}(\bar{A}_1) \div \bar{x}(\bar{w}(\bar{A}_2)) = q.$$

It is well known that q is not a proper digital number. For instance .10 ÷ .01 = 10.0 (binary) and q lies outside the range of $\bar{x}$, capacity

has been exceeded, and digits created at the left. Also .001 ÷ .101 = .0011..., (binary) and this segment of four digits repeats indefinitely. Since the quotient may have an infinite number of digits, it is not possible to split it into two halves as was done with the product. Finally, if the divisor, $\overline{x}(\overline{w}(\overline{A}_2))$, is equal to zero, q does not even exist.

The division algorithm which the computer is designed to perform, lumps the cases $q \geq 1$, and q does not exist, into the same category. In either of these cases, the computer emits a signal "division capacity exceeded" to the dispatcher. Since the signal is ignorable, a logical design decision must be made concerning the word sent to the memory when capacity is exceeded. Partly for logical reasons and partly for engineering simplicity it has been decided to emit a string of pulses, as $D(\overline{w}(q))$, when capacity has been exceeded. Let us refer to this case as Q. This makes $|\overline{x}|\ (w(Q)) = 1 - 2^{-43} = \max|\overline{x}|$. When we exceeded capacity in addition and subtraction, the fractional part of the result was retained. This is not feasible in division, even with round-off, for three reasons; in the first place, the computer division process does not readily give the fractional part of Q in case capacity is exceeded, secondly, the quotient may not even exist, and thirdly, this information is of dubious value. In addition-subtraction the fractional part gives 44/45 = 98% of the information, so to speak, but in division it would give much less. The fractional part is rejected for these reasons. The selection of max $|\overline{x}|$ is reasonable, since it is the largest proper digital number we have available and hence suitable for indicating an exceed capacity. Furthermore, it will be shown that $\overline{|q|} < \max\ |\overline{x}|$

if capacity is not exceeded, so if division without round-off is programmed, the fact that capacity has been exceeded can be detected by suitable programming, if for any reason it is desired to do so. Finally, a string of pulses is easy to manufacture in the circuits.

When capacity is exceeded, we therefore send a string of ones to the memory, for the digits. The sign remains to be considered. We always follow the rule of signs, and the complete details will be given in the description of "exact division". Speaking loosely, if $|\pm a/\pm b| \geq 1$, or if $b = 0$, then the result is $\pm (1 - 2^{-43})$. The plus sign is selected if a and b have the same signs, the minus sign is selected if a and b have opposite signs.

The maximum value of $|q|$, if it exists, is $(1 - 2^{-43})/2^{-43} = 2^{43} - 1$. Hence $|q|$, expanded in the binary system is:

$$|q| = \sum_{i=-42}^{i=\infty} 2^{-i} \delta_i \text{, where } \delta_i = 0, 1.$$

The possible 43 digits to the left of the "point" have already been ruled out, their occurrence indicates capacity has been exceeded. Attention will now be directed to the useful results where $|q| < 1$. Henceforth, q will be used only when $|q| < 1$.

$$|q| = \sum_{i=1}^{i=\infty} 2^{-i} \delta_i \text{, where } \delta_i = 0, 1.$$

q is of course still not properly digital, although we have now eliminated the extra digits at the left. The fact that max $|q| <$ max $|\bar{x}|$ will now be established. Consider two positive proper digital numbers $m\, 2^{-43}$ and $n\, 2^{-43}$, where $2^{+43} - 1 \geq m \geq 0$; $2^{43} - 1 \geq n \geq a$. If $m/n = |q|$, then $m < n$, otherwise $m/n \geq 1$. For a given n, max $(m/n) = (n-1)/n$. Since

this increases monotonically with n, max (m/n) = $(2^{43} - 2) / (2^{43} - 1) =$

$(1 - 2^{-42}) \sum_{i=0}^{i=\infty} 2^{-43i} = 1 - \sum_{i=1}^{i=\infty} 2^{-43i}$. Digitally, max $|q|$ is a repeating decimal with a 43-digit repetend, consisting of 42 ones followed by a zero; in other words, the dividend "repeats".

$$\text{Max} \quad |q| = 1 - \sum_{i=1}^{i=\infty} 2^{-43i} < 1 - 2^{-43} = \max \quad |\overline{x}|.$$

The values max $\overline{|q|}$ and max $\overline{|q|}^{*}$ are of more interest. The first corresponds to merely dropping all digits after the 43d in the quotient, the second to adding 2^{-44}, (forming all carries) and then dropping all digits after the 43d. The functions $\overline{|\ \ |}$ and $\overline{|\ \ |}^{*}$ are defined exactly as they were in discussing multiplication. Now then, we merely need note that

$$0 < \sum_{i=1}^{i=\infty} 2^{-43i} < 1/2 \text{ and max } \overline{|q|} = \left[\max \ |q| \ 2^{43}\right] \cdot 2^{-43}$$

$$= \left[2^{43} - \sum_{i=1}^{i=\infty} 2^{-43i} \underline{+43}\right] \cdot 2^{-43}$$

$$= \left[2^{43} - \sum_{i=0}^{i=\infty} 2^{-43i}\right] \cdot 2^{-43}$$

$$= \left[2^{43} - 1 - \sum_{i=1}^{i=\infty} 2^{-43i}\right] \cdot 2^{-43} =$$

$(2^{43} - 2) \cdot 2^{-43} = 1 - 2^{-42}$;

$$\max \overline{|q|}^{*} = \left[\max \ |q| \ 2^{43} + 1/2\right] \cdot 2^{-43}$$

$$= \left[2^{43} - 1 + 1/2 - \sum_{i=1}^{i=\infty} 2^{-43i}\right] \cdot 2^{-43}$$

$$= (2^{43} - 1) \cdot 2^{-43} = 1 - 2^{-43} = \max \ |\overline{x}| \ .$$

Both $\overline{|q|}$ and $\overline{|q|^*}$ are properly digital. Since $\overline{q^*}$ has the more desirable round-off properties, it is selected for encoding as the most satisfactory single-word representation of q. This requires computing 44 quotient digits, or 44 minor cycles of computing time, since division is done serially. This increases the time required for division to about 2% more than that required for multiplication. It is believed that this increase is justified in order to give a round-off which has the same properties as the multiplication round-off.

It remains only to describe $\overline{w}(\overline{q^*})$, the word encoded by the computer. The proper digital number is $\overline{q^*} = \overline{|q|^*} \cdot \operatorname{sgn} \overline{q^*}$. The resulting word is $\overline{w}(\overline{q^*}) = |\overline{x}|^{-1} \overline{|q|^*} S'(\overline{q^*})$. $|\overline{x}|^{-1} \overline{|q|^*}$ is the segment $D(\overline{w})$ containing the characters $c_1 - c_{43}$, the digits of $\overline{|q|^*}$, which is $|q|$ rounded off to 43 binary digits, and $S'(\overline{q^*})$ is the character c_{44}, the sign of $\overline{q^*}$. If $\overline{q^*}$ is zero, it is encoded as a + 0, in accordance with the usual conventions.

To recapitulate, the first execute transfers $\overline{w}({}_1\bar{A})$ to a computer short memory tank, and returns it to the memory via the recirculating system. This number is the dividend. The second execute transfers $\overline{w}({}_2\bar{A})$, the divisor, to the same computer short memory tank, and sends $\overline{w}(\bar{A}_1)$ to the subtracting circuits to be used as the first remainder, and begins the serial process of division. The division algorithm is based on the non-restoring principle and requires 1 minor cycle per digit, or 44 minor cycles in order to give the extra digit for round-off purposes. The computer forms enough digits of the quotient

$\overline{x}(\overline{w}(_1\overline{A}) \div \overline{x}(\overline{w}(_2\overline{A}))$ to permit the determination of $\overline{q^*}$ or to establish the fact that capacity has been exceeded. In the latter case the "division capacity exceeded" signal is sent to the dispatcher and $\overline{w}(Q)$, corresponding to $1 - 2^{-43}$ or $-1 + 2^{-43}$, is sent to the memory in normal operation. The third execute, which may occur no earlier than 44 minor cycles after the second execute, replaces $\overline{w}(_3\overline{A})$ by $\overline{w}(Q)$ or $\overline{w}(\overline{q^*})$. It should be noted that there exists a $\overline{w}(\overline{q^*}) = \overline{w}(Q)$. Finally, the fourth execute sends $\overline{w}(_4\overline{A})$ to the dispatcher, to continue the program chain.

5.09 (d, $\overline{5}$ -) Exact division

It has already been pointed out that it is impossible to store all the digits of q, even using two memory positions. Since the numbers involved are rational, q is at worst a repeating binary fraction, in the digital sense. If advantage were taken of the fact that the numbers are also bounded, it should be possible to determine a bound to the length of the repetend. Given sufficient space in the memory, an exact division algorithm could be devised which would terminate in a bounded interval of time, provided of course that the numbers be properly digital.

A more practical arrangement is possible. Suppose we are attempting to determine $|\bar{a}| \div |\bar{b}|$, where $\bar{a}$ and $\bar{b}$ are properly digital and $|\bar{a}| < |\bar{b}|$; there is no question of exceeding capacity. Now

$$|q| = \left|\frac{\bar{a}}{\bar{b}}\right| = \overline{|q_0|} + 2^{-43}\epsilon_0, \text{ where } 0 \le \epsilon_0 < 1,$$

in view of the properties of the operator $\overline{\ |\ |\ }$. Furthermore

$$|\bar{a}| = |\bar{b}|\,\overline{|q_0|} + 2^{-43}(\epsilon_0|\bar{b}|).$$

We set $|r_o| = \epsilon_o |\bar{b}|$, and show that $|r_o|$ is a proper digital number. Since $|\bar{b}||\bar{q}_o|$ is the product of two properly digital numbers, it is of the form $\sum_{i=1}^{i=86} 2^{-i}\delta_i$ or $N_1 \cdot 2^{-86}$, where N is a non-negative integer. Now $|\bar{a}|$ is properly digital, hence $|\bar{a}| = N_2 \cdot 2^{-43}$ where N_2 is a non-negative integer. Consequently

$$2^{-43}|r_o| = N_2 2^{-43} - N_1 2^{-86},$$
$$|r_o| = N_2 - N_1 2^{-43}.$$

Therefore, $|r_o|$ can vary only by increments of 2^{-43}. Since $|\bar{b}| < 1$, $\epsilon_o < 1$, $|r_o| < 1$, hence $0 \leq |r_o| \leq 1-2^{-43}$, and $|\bar{r}_o|$ is properly digital. As a matter of fact, we will very shortly need the sharper inequality, $0 \leq |\bar{r}_o| < |\bar{b}|$. Since $|\bar{r}_o| = \epsilon_o |\bar{b}|$, and $0 \leq \epsilon_o < 1$, then $|\bar{r}_o| < |\bar{b}|$. Furthermore, since $|\bar{r}_o|$ is properly digital, max $|\bar{r}_o| = |\bar{b}| - 2^{-43}$ ($|\bar{b}| \neq 0$ since capacity has not been exceeded). Now

$$\left|\frac{\bar{a}}{\bar{b}}\right| = \overline{|q_o|} + 2^{-43}\frac{|\bar{r}_o|}{|\bar{b}|},$$

and if we perform another division by $|\bar{b}|$, which will not exceed capacity, since we have just shown $|\bar{r}_o| < |\bar{b}|$,

$$|q| = \overline{|q_o|} + 2^{-43}\left(\overline{|q_1|} + 2^{-43}\epsilon_1\right)$$
$$= \overline{|q_o|} + 2^{-43}\overline{|q_1|} + 2^{-86}\epsilon_1,$$

and ϵ_1 has all the properties that ϵ_o had, hence by iterating this process we can get $|q|$ as accurately as we like. If the quantity $|\bar{r}|$ is stored, along with $\overline{|q|}$, we can get as many digits of $|q|$ as desired. This scheme is the one familiar to operators of desk calculators. The remainder $|\bar{r}|$ is transferred from the right hand side of the accumulator register to the left hand side of the accumulator

register; the counter register is copied and cleared, and another division performed using the original divisor, and the whole process repeated as many times as necessary to get the desired accuracy.

The exact division process in the EDVAC is designed to permit this iteration to be performed when desired. $\overline{|q|}$ and $|\bar{r}|$ are stored in the memory together with the proper signs, and these remain to be considered. Unless $\overline{|q|} = 0$, we obviously assign the sign of q to $\overline{|q|}$. q must have a sign if $\overline{|q|} \neq 0$, since $\overline{|q|} \leq |q|$. If $\overline{|q|} = 0$, we assign a + sign to $\overline{|q|}$, in accordance with our convention.

We prefer, if possible, to choose the sign of $\bar{r}$ in such a way as to make $\bar{a} = \bar{b}\bar{q} + 2^{-43}\bar{r}$, regardless of the signs involved, provided this is possible. We have available the relation $|\bar{a}| = |\bar{b}|\,\overline{|q|} + 2^{-43}|\bar{r}|$. If $|\bar{r}| = 0$, there is no error in $|\bar{q}|$, since $|\bar{b}| \neq 0$, and hence $\epsilon = 0$. In this case $\bar{a} = \bar{b}\,\bar{q}$ and we assign the + sign to $|\bar{r}|$.

We now assume $\bar{a} \neq 0$, since otherwise $\overline{|q|} = 0$, $|\bar{r}| = 0$, and we have the special case just mentioned. Now assume $\bar{a} > 0$. We must have $|\bar{b}| > 0$, otherwise capacity would be exceeded. Now $\bar{q}$ must be given the same sign as $\bar{b}$, since otherwise $\bar{a}/\bar{b} = q$ would not have the same sign as $\bar{q}$. Consequently $\bar{b}\bar{q} \geq 0$, and $\bar{a} = \bar{b}\bar{q} + 2^{-43}(\,|\bar{r}|\,)$. Now assume $\bar{a} < 0$, we must again have $|\bar{b}| > 0$. Now $\bar{q}$ must have the opposite sign to $\bar{b}$. Consequently $\bar{b}\bar{q} \leq 0$ and $a = bq - 2^{-43}(\,|\bar{r}|\,) = \bar{b}\bar{q} + 2^{-43}(-|\bar{r}|)$. We thus see that, assuming $|\bar{r}| \neq 0$, if we assign the sign of $\bar{a}$ to $|\bar{r}|$, the relation $\bar{a} = \bar{b}\bar{q} + 2^{-43}\bar{r}$ is always satisfied.

As previously stated, we assign the + sign to $|\bar{r}|$ if $|\bar{r}| = 0$.

Now the encoding of q and r will be described. We are still assuming that capacity has not been exceeded.

$$\overline{w}(\bar{q}) = |\bar{x}|^{-1} \widehat{|q|\ S'(\bar{q})};$$

$$\overline{w}(\bar{r}) = |\bar{x}|^{-1} \widehat{|\bar{r}|\ S'(\bar{x}(\bar{w}(_1\bar{A})))}, \text{ if } |\bar{r}| \neq 0;$$

$$\overline{w}(\bar{r}) = |\bar{x}|^{-1} \widehat{|\bar{r}|\ S'(|r|)}, \text{ if } |\bar{r}| = 0.$$

The problem of exceeding capacity must now be treated. $D(\bar{w}(Q))$ is the same as in division with round-off, that is $|\bar{x}|\ D(\bar{w}(Q)) = 1 - 2^{-43}$. This quantity is used in place of both $\bar{q}$ and $\bar{r}$; the substitute for $\bar{r}$ will be referred to as R. $|\bar{x}|\ D(\bar{w}(R)) = 1 - 2^{-43}$. It remains to settle the question of signs. The same rule is followed for Q as in the case of division with round-off. In the case of R, we select the sign of $\bar{a}$, even though it may be a minus zero.

We define $S''(w_1, w_2) = S''(S(w_1), S(w_2))$ as follows:

$S(w_1)$	$S(w_2)$	$S''(w_1, w_2)$
No pulse (+)	No pulse (+)	No pulse (+)
Pulse (−)	No pulse (+)	Pulse (−)
No pulse (+)	Pulse (−)	Pulse (−)
Pulse (−)	Pulse (−)	No pulse (+)

This is nothing more nor less than the rule of signs. It is easily mechanized by counting the minus signs in a one-stage binary counter. If the number is odd, we emit a pulse, if even, no pulse.

In both types of division,

$$\overline{w}(Q) = |\bar{x}|^{-1} \widehat{(\max|\bar{x}|)\ S''(\bar{w}(_1\bar{A}), \overline{w}(_2\bar{A}))}.$$

The sign affixed to max $|\bar{x}|$ is in accordance with the rule of signs. $\bar{w}(R)$ is selected as follows

$$\bar{w}(R) = |\bar{x}|^{-1}(\max|\bar{x}|) \overbrace{\quad} S(\bar{w}(_1\bar{A}));$$

we choose the sign of the dividend, even if it be a minus zero. Both of these sign conventions are chosen simply because they are the results which the sign circuits happen to give. It is believed that the sign choice can afford to be arbitrary, since capacity has been exceeded, and the digits are useless as representing any correct numerical values. We have already noted in the discussion of division with round-off, that there exists $\bar{w}(\overline{q^*}) = \bar{w}(Q)$. However, since max $|\bar{q}| = 1 - 2^{-42}$, and $|x|^{-1}(\bar{w}(Q)) = 1 - 2^{-43}$, $\bar{w}(Q)$ uniquely determines Q when exact division if performed. $\bar{w}(R)$ has the same property. We have already shown that max $|\bar{r}| = |\bar{b}| - 2^{-43}$, and max $|\bar{b}|$ = max $|\bar{x}| = 1 - 2^{-43}$, hence max $|\bar{r}| = 1 - 2^{-42}$ which is less than $|\bar{x}|^{-1}(\bar{w}(R)) = 1 - 2^{-43}$. Exceed capacity can be determined from either the quotient or the remainder result, in exact division.

The results of the analysis will now be collected. In the first place, the first two executes are exactly the same as in division with round-off. Now assume that Q (exceed capacity) does not occur. Since we are computing $|\bar{q}|$ rather than $\overline{|q|^*}$, only 43 minor cycles are required to obtain $\overline{|q|}$; one cycle per digit. The third execute, which replaces $\bar{w}(_3\bar{A})$ by $\bar{w}(\bar{q})$, occurs one minor cycle earlier than in division with round-off. During the minor cycle following this execute, the EDVAC, a serial computer, is doing three things at once. First, it is transferring $\bar{w}(\bar{q})$ to $P(_3\bar{A})$. Second, it is preparing a new berth for $\bar{w}(\bar{r})$. Just as in exact multiplication, the dispatcher

in effect adds unity to $_3\bar{A}$, more exactly it forms the new address $N^{-1}(N(_3\bar{A}) + 1)$, which is checked for long tank selection and immediately executed. The third process is the preparation of $\bar{w}(\bar{r})$. The computer uses non-restoring division, and the remainder memory may contain $\left||\bar{b}| - |\bar{r}|\right| = |\bar{b}| - |\bar{r}|$, instead of $|\bar{r}|$. If this is the case then the subtracter forms

$$\left||\bar{b}| - |\bar{r}| - |\bar{b}|\right| = \left|-|\bar{r}|\right| = |\bar{r}|; \quad (\text{since } |\bar{b}| > |\bar{r}|).$$

This restores the remainder, requiring 1 minor cycle, but this is done concurrently with the above two operations.

The fourth execute, which follows the third after a lapse of exactly one minor cycle, thus replaces $\bar{w}(N^{-1}(N_3\bar{A}) + 1))$ by $\bar{w}(\bar{r})$. The remainder is stored in the memory position "next after" the quotient. If the quotient goes into position 1023, the remainder goes into position 0, the "next" position is used in the cyclic sense.

Finally, the fifth execute sends $\bar{w}(_4\bar{A})$ to the dispatcher, determining the next order. The above is varied if capacity is exceeded. As soon as the computer detects the exceed capacity, which may occur before the usual 43 minor cycles have elapsed (44 for division with round-off), the computer emits the "exceed capacity signal". This may be used to modify the subsequent events, by setting certain controls, but if this is not done the third and fourth executes occur as soon as possible after the signal is received, and the computer replaces $\bar{w}(_3\bar{A})$ by $\bar{w}(Q)$ and $\bar{w}(N^{-1}(N_3\bar{A}) + 1))$ by $\bar{w}(R)$.

5.10 (W,Z-) Wire

The EDVAC is so organized that the three spools of magnetic wire serve as an auxiliary low-speed memory and not merely an input and

output device. A true input device, which ENIAC has, can only serve to send prepared information to the machine in unlimited amounts while the machine is running, and a true output only receives information from the machine in similarly unlimited amounts. The machine can only read in material which the operator has in some manner prepared sufficiently far in advance to keep the reservoir from becoming exhausted; information which has been emitted at the output may be, and usually is, completely forgotten by the machine and in this case cannot be recalled unless the output data is manually transferred to the input reservoir.

In the EDVAC, all of the data which the machine records on the wires can be made automatically available to the machine by suitable programming. Since the wires are of finite, but large, capacity, data cannot be inserted into or received from the machine in indefinite amounts unless the machine is stopped and the wires exchanged for different ones. In view of the 50,000 word capacity, per wire spool, it is believed that this is a small price to pay for the really great advantage of providing a true memory of large capacity.

Since the wire transport is a mechanical device, the time required to reach an arbitrary position on a wire will be enormously greater than the access time of the high-speed memory. The programmer will thus wish to refer to the wire memory as infrequently as possible, and in particular to avoid referring to remote positions. The wire order is consequently arranged to transfer data in blocks of arbitrary size, from one word up to a full load for the high-speed memory, and to always read the positions on the wire immediately adjacent to the magnetic reading heads. If the wire is not in the proper position, it must be

transported to the desired location, without reading or recording. In this case the block of arbitrary size refers to the distance, measured in word positions, that the wire moves. Since we must be able to transport the wire in either direction, it is also convenient to be able to read and record on the wire in either direction, which is possible in most cases.

The above conclusions indicate that a complete wire order must contain a comprehensive pattern of information. In addition to indicating the positions in the high-speed memory which are affected (normally indicated by addresses) it must also choose one of three wires, one of the two directions forward and backward, and the number of positions to be moved.

So far, little has been said about the high-speed memory positions involved. We wish to reserve the fourth address ($\overline{A}_4$) to specify the location of the next order in the program chain. This leaves three addresses, so at most three high-speed memory positions can be indicated. If we used these addresses directly to specify the memory positions to be filled, or whose contents are to be recorded on the wire, we would be restricted to considering at most 3 positions. We have already noted that we wish to transfer information in blocks of arbitrary size from 1 to 1024 words. If we are willing to fill the memory positions in a standard pattern, say in sequence, then we need only specify the starting point in the sequence, and the size of the block to be transferred. Since the numerical equivalent of an address ranges from 0 to 1023, an address is almost exactly the proper medium for expressing the size of the block, and another address can be used to specify the starting point in the sequence of memory positions

involved. Alternatively, we can consider the two addresses as specifying the initial and terminal points of the sequence of memory positions involved. If the wire is to be transported without reading or recording, this will still specify the amount of translation, but we disconnect the transfer channels. The amount of wire to be moved, and the high-speed memory positions involved are now specified. It still remains to select the wire, etc. Before this is discussed, a general method of subdividing an order-type will be analyzed.

In section 2, the partition of a word into 5 segments by the dispatcher was explained:

$$\overline{w} = \overline{{}_1A}\,\overline{{}_2A}\,\overline{{}_3A}\,\overline{{}_4A}\,\overline{T}.$$

The discussion of the visual order required considering the least significant octonary (octal) digit in ${}_3\bar{A}$. This, in effect, constitutes a still finer dissection of the word. In the wire order and extract order, this subdivision of a word is carried out in a more comprehensive way. The following functions of addresses are introduced:

$$\left.\begin{aligned} {}_0\overline{\Omega}({}_j\bar{A}) &= (c_{10(j-1)+8}, c_{10(j-1)+9}, c_{10(j-1)+10}),\\ {}_1\overline{\Omega}({}_j\bar{A}) &= (c_{10(j-1)+5}, c_{10(j-1)+6}, c_{10(j-1)+7}),\\ {}_2\overline{\Omega}({}_j\bar{A}) &= (c_{10(j-1)+2}, c_{10(j-1)+3}, c_{10(j-1)+4}),\\ B({}_j\bar{A}) &= c_{10(j-1)+1}. \end{aligned}\right\} \quad j = 1,2,3,4$$

Therefore,

$${}_j\bar{A} = B\,\overline{{}_2\Omega}\,\overline{{}_1\Omega}\,\overline{{}_0\Omega}\,;\quad j = 1,2,3,4.$$

This corresponds to breaking up the address into three octonary (octal) digits and one binary digit. Blocks of one and three characters are given a numerical significance, one which preserves the interpretation

of the characters as zeros (no pulse) and ones (pulse):

$\nu_1(c_i) = c_i$ where $c_i = 0,1$; for any i such that $1 \leq i \leq 44$

$\nu_3(c_i, c_{i+1}, c_{i+2}) = \sum_{j=0}^{j=2} 2^j c_{i+2-j}$, where $c_{i+2-j} = 0,1$, for any i such that $1 \leq i \leq 42$.

The values of ν_1 will be denoted by 1, 0, (binary digits). The values of ν_3 will be denoted by $\overline{7}$, $\overline{6}$, $\overline{5}$, $\overline{4}$, $\overline{3}$, $\overline{2}$, $\overline{1}$, $\overline{0}$ (octonary (octal) digits). Consequently

$$\nu_1(B) = 1 \text{ or } 0 ;$$
$$\nu_3({}_k\overline{\Omega}) = \overline{7} \text{ or } \overline{6} \text{ or } \ldots \overline{0} ; \qquad 0 \leq k \leq 2.$$

The numerical interpretation of an address, is of course, expressible in terms of the numerical expressions of its binary and octonary (octal) digits:

$$n({}_j\bar{A}) = 8^3(\nu_1(B({}_j\bar{A}))) + \sum_{k=0}^{k=2} 8^k \nu_3({}_k\overline{\Omega}({}_j\bar{A})) ; \quad j = 1,2,3,4$$

The first term is either 0 or 512, the second term has any value from 0 to 511. As previously mentioned, the least significant octonary (octal) digit in an address identifies a minor cycle, and the other digits identify a long tank.

In the orders previously described, with the exception of "visual," an address $({}_j\bar{A})$ is either used to select a memory position, or else it is ignored completely. In the visual order, the least significant octonary (octal) digit of the third address, ${}_0\overline{\Omega}({}_3\bar{A})$, was used to select a scope (or no scope at all). In the wire order, all of the digits of the second address except one, and in the extract order all of the digits of the second address are used to further specify the order.

This scheme permits greater flexibility in coding; if we are willing to reduce the number of independent memory-position variables, we permit the order-type to specify a larger number of alternatives. Each address which we give up gives up to 1024 variants on the order-type. In this way, the sixteen possible order-types can be extended where desirable.

The wire order uses this scheme because we have already seen that the use of three addresses in the ordinary way is not feasible, and because additional specification of subdivisions of the order-type is imperative. We only have one address available for this treatment, but that is more than sufficient. The octonary (octal) digit at the extreme right selects the wire (or auxiliary input):

$$\nu_3({}_0\bar{\Omega}({}_2\bar{A})) = \begin{cases} 1 \bmod 4, & \text{selects wire 1;} \\ 2 \bmod 4, & \text{selects wire 2;} \\ 3 \bmod 4, & \text{selects wire 3;} \\ 0 \bmod 4, & \text{selects auxiliary input.} \end{cases}$$

The next octonary (octal) digit, ${}_1\bar{\Omega}$ $({}_2\bar{A})$ is not used at all; the dispatcher is completely indifferent to these three pulses. The first octonary (octal) digit, has the following significance, provided $\nu_3({}_0\bar{\Omega}) \not\equiv 0 \bmod 4$, i.e., if we have selected a wire.

$$\nu_3({}_2\bar{\Omega}({}_2\bar{A})) = \left.\begin{cases} 0 \bmod 4, & \text{translate (TR),} \\ 1 \bmod 4, & \text{memory to wire (MW),} \\ 2 \bmod 4, & \text{wire to memory (WM),} \\ 3 \bmod 4, & \text{read fifth address (R5A).} \end{cases}\right\} \text{To be explained subsequently.}$$

If $\nu_3({}_0\bar{\Omega}) \equiv 0 \bmod 4$, the dispatcher is not interested in ${}_2\bar{\Omega}$ at all. Finally, the binary digit determines whether the wires are to run forward or backward.

$$\nu_1(B(_2\bar{A})) = \begin{cases} 0, \text{ run wire forward;} \\ 1, \text{ run wire backward.} \end{cases}$$

A word of explanation concerning the auxiliary input is required. The auxiliary input consists of a set of 44 double-throw switches on the control panel. The switches thus effectively constitute a word of 44 characters. Whenever the auxiliary input is selected by the wire order, the word set up on these switches will be read into all the high-speed memory positions designated by the rest of the order. It is not possible to set these switches during the reading because this is accomplished at the speed of one minor cycle per word. Ordinarily these switches will be used to send only single words to the high-speed memory, primarily for testing and maintenance. If it were not for this apparatus it would be impossible to deposit even a single word in the high-speed memory without going through the inscriber routine of preparing tapes and wires, loading wires into the reader-recorder, and so forth. Since the auxiliary input can be regarded as a wire of unlimited length, containing identical words in every position, forward and backward have no particular significance; if you like, they produce identical results. Furthermore, the auxiliary input only sends data to the memory; it cannot receive anything from it whatever. Certain methods of overcoming this restriction are currently under investigation.

"Translate" (TR), refers to the process of moving the wire without reading or recording, "memory to wire" (MW) designates the process of transferring the words of the k-language (44 characters) to the wire. The memory data is returned to the memory position from which it came, and any data appearing on the wire is erased just before the recording head reaches the wire, in accordance with the convention. "Wire to

memory" (WM) designates the inverse transfer of words of the k-language from the wire to the memory; data on the wire is not erased but the old memory contents affected are destroyed. In WM, attempts to read words of the K-language (54 characters) will halt the machine and give an indication of an error in the reader-recorder. In fact, any word containing more or less than 44 characters will do likewise.

"Read fifth address" (R5A) refers to the process of reading words of the K-language from the wire to the memory, without erasing the wire, but displacing previous memory contents. The inverse operation is impossible, as already described in section 2, where the reasons why this cannot be done in the reverse direction are also explained. An order which is coded to R5A backwards will halt the machine and indicate this particular cause. The reader-recorder error will also be omitted if R5A is attempted even in the forward direction on words not of the K-language. Any word containing more or less than 54 characters will do likewise.

"Forward" means to move the wire in the same direction for as many positions as needed to satisfy the rest of the order, and "backwards" the opposite direction. A few examples will be given of the significance of different kinds of $\underline{1}\overline{2}\overline{A}$; the preferred coding is given first in each case;

$\underline{1}\overline{003}$
$\underline{1}\overline{063}$
$\underline{1}\overline{427}$

Translate wire 3 backwards.

$\overline{1303}$)
$\overline{1767}$) Read fifth addresses from wire 3 backwards (will halt machine),

$\overline{0202}$)
$\overline{0636}$) transfer data from wire 2 to memory forwards,

$\overline{0101}$)
$\overline{0565}$) transfer data from memory to wire 1 forwards,

$\overline{0000}$)
$\overline{1370}$) transfer data from auxiliary input to memory.

The method of selecting the memory positions involved needs clarification. In all save R5A, the first memory position used is $P({}_1\bar{A})$, the position designated by the first address. The dispatcher then compares ${}_1\bar{A}$ with ${}_3\bar{A}$. If they are the same, that concludes the wire order. If they are not the same, the dispatcher forms a new ${}_1\bar{A}$, which is $N^{-1}(N({}_1\bar{A})+1)$. This is the same operation employed in exact multiplication to get the next address; loosely speaking, 1 is added to the original address. The next memory position affected will be $P(N^{-1}(N({}_1\bar{A}+1)$. The dispatcher after "executing" this address, then compares $N^{-1}(N_1\bar{A})+1)$ with ${}_3\bar{A}$. If they are the same that concludes the wire order. If not, then, etc., etc., until finally for some $0 \leq p \leq 1023$, $N^{-1}(N({}_1\bar{A})+p) = {}_3\bar{A}$. Such a p always exists, since we certainly have exhausted every possible address when $p = 1023$. $p + 1$ memory positions have been affected since the word is transferred before the comparison. Clearly $1 \leq p + 1 \leq 1024$ and hence ${}_1\bar{A}$ and ${}_3\bar{A}$ permit us sequentially to fill a block of the memory of any desired size. The wire also has moved $p + 1$ positions; in translate (TR), no memory positions are affected whatever.

The simplest way to express all this is to say that ${}_1\bar{A}$ indicates the first high speed memory position involved, and ${}_3\bar{A}$ the last one,

together with all between, "following around the clock" through 1023 if necessary. If ${}_1\bar{A} = {}_3\bar{A}$ originally, 1 word is transferred; if $N({}_1\bar{A})-1 = N({}_3\bar{A})$; 1024 words are transferred. If the auxiliary input is used, all words will be identical.

Read fifth address operates somewhat differently. As soon as $K_1 - K_{10}$ (subsequently denoted by ${}_5\bar{A}$) has been assembled in the processing delay, it is sent to the dispatcher where it displaces ${}_1\bar{A}$ ${}_1\bar{A}$ has no significance in R5A. As soon as $K_{11} - K_{54}$ has been assembled, the dispatcher causes it to be sent to the memory position $P({}_5\bar{A})$. ${}_5\bar{A}$ is now compared with ${}_3\bar{A}$. If they are the same this ends the order. If not, a new word is read, a new ${}_5\bar{A}$ sent to the dispatcher, etc., etc. It must be noted, that contrary to MW, WM, and TR, there is no existence theorem showing that this operation will ever cease, except by running out of wire. Consequently the programmer must use care to see that ${}_3\bar{A}$ actually corresponds to some ${}_5\bar{A}$ in the block of K-language words being read, namely, the last one desired. ${}_4\bar{A}$, as usual, indicates the location of the next order to be executed.

5.11 (E, $\bar{3}$ -) Extract

In the wire order, ${}_2\bar{A}$ was used to subdivide the order type, primarily because it was highly desirable to do so, but partly because no other practical use for ${}_2\bar{A}$ could be found. In the extract order, an extremely versatile operation is obtained at the cost of giving up an address. Normally, in the computing operations, both independent variables are returned unchanged to the memory. In the extract order, the result is returned to the same place from whence came one of the words used to determine the result. This word does not in general survive unaltered.

If it must be preserved, a duplicate of it must be transferred elsewhere before the operation takes place. Since in most cases in which this order is used, the preservation of the word is not desired, it is felt that this is a small price to pay for the great flexibility of the order.

Speaking broadly, "extract" permits shifting the characters in a word any arbitrary amount to the right or left, and also enables selected segments of words (replacees) to be replaced by the corresponding portions of others (replacers). In general, therefore, the replacee will finally consist in part of its original characters, and in part of characters selected from the replacer. The complete operation enables both shifting and replacement operations to be performed in succession using only one order. The shifted word is used as the replacer, but since zero shift is permitted, replacement without shifting is possible; since the entire word may be chosen as a segment to be replaced, in this case the result contains only characters of the replacer, and none of the replacee. The result in this case may be said to consist of a shift only, since all the characters of the result are derived from the shifted word.

The subdivisions of the order-type must specify the amount of the shift, direction of shift, and the segments to be replaced. There are 8 possible choices of segments, which have been selected in accordance with interpretations of words by the computer and the dispatcher as a result of the following considerations.

Considering the numerical interpretation of words, one of the most important attributes of shifting is the fact that it corresponds to multiplying or dividing by powers of 2. Although it is possible to halve by multiplying by 1/2 and to double by dividing by 1/2, or to

double by iteration of addition, these are time consuming and complex methods of doing a simple operation.

If a floating "binary" point is to be programmed, or scale adjustments need to be made, it will be desirable to perform the shift more rapidly. The most useful segments of a number will be the sign only, the digits only, and the entire number (to permit pure shifts without replacement). This gives three cases. If the order interpretation of a word is considered, the most important segments are the addresses ($\bar{A}_j$). It will be convenient to replace addresses in orders by corresponding or different addresses in other orders. This gives four more cases, or a total of seven.

Up to this point, the word "shift" has been freely used without any explanation of precisely what is involved. The numerical interpretation of words gives the cue here. Since words have a fixed length, a left shift requires creation of new characters at the right, a loss of old characters at the left, and conversely for a right shift. A cyclic shift could, of course, be adopted which inserted the characters pushed out at one end into the gap opened at the other. This is completely preposterous as a numerical interpretation, which requires that all the new digits be zeros. It follows that the characters moved out will have to be given up entirely. If the shifts are interpreted as multiplication and division by powers of 1/2, then in the former case we have a round-off of the type symbolized by $\overline{|\ |}$, that is, the excess digits are discarded without adding in the appropriate 1/2, and in the case of division, we retain only the fractional part of the quotient without any exceed capacity signal. If these possibilities are significant,

they must be considered in the programming, or the slower multiplication and division orders used.

Numerical considerations require that the sign character (c_{44}) be treated differently from the digits. The minus sign is represented by a pulse, and so is a digit 1; if the sign character is shifted to the left, it will appear as a digit; if the shift is to the right, it will disappear entirely. On the other hand, if we refuse to shift the sign at all, we give up the possibility of moving an arbitrary digit into the sign position, which may be useful for special purposes, since it permits sensing of individual digits via the compare order. The only solution to this dilemma is to provide two versions of the shift, one to be associated particularly with a pure shift for numerical interpretations of words. In this case the sign is not shifted, and any other character is prohibited from occupying its position. In all other replacements, with one exception, we throw the sign away entirely, and allow its position to be occupied by any other character. This is no hardship as far as the order interpretation of a word is concerned, since the main purpose of shifting in this case will be to move addresses into the desired replacement position, and c_{44} is part of the order-type designation rather than part of an address. Furthermore, an operation which is of considerable importance in the numerical interpretation of words is that of obtaining the absolute value of a number. This can be accomplished either by throwing away the sign entirely, by replacing the digits of a number known to be positive by the digits of the number whose absolute value is desired, or by replacing the sign of the number whose absolute value is desired by the sign of a number known to be positive. The first case is the simplest since only one number needs to be con-

sidered.

It will be noted that with only these two alternatives there is no possibility of moving the sign whatever. We either don't move it or throw it away entirely. The replacement code also suffers from a lack of generality, e.g., suppose we wish to retain all the characters of a word save an arbitrary one, and replace that by an arbitrary character in another word. This is a basic operation which is not easy to visualize performing with the replacement segments already selected, corresponding to addresses, complete sets of digits, etc.

In order to simplify this operation, we provide a new kind of segment suitable for replacement. The previous ones have all been constant sets of characters in the replacee, the word which is changed by having different characters inserted in it. The new replacement segment will consist of a single character. Its location is determined by the position of the sign of the shifted word, and in this case the sign is actually shifted. Two birds are killed with one stone. In fact, any character of any word can replace any character of any other word using two extract orders. If the replacer character is already the sign, the problem is trivial, and only one order is required. If not, the replacer word has the desired character shifted into the sign position and this is then used to replace the sign of any arbitrary word. This reduces the situation to the trivial case. This new replacement segment gives a total of eight possible cases which conveniently corresponds to an octonary (octal) digit. $_{0}\overline{\Omega}\,(_{2}\overline{A})$ is chosen as the part of the order which specifies this information.

The results of the analysis will now be presented more formally:

$\nu_3(_0\bar{\Omega}(_2\bar{A})) = 1$, replace $_2\bar{A}(\bar{w}(_3\bar{A}))$ by $_1\bar{A}$ of shifted $\bar{w}(_1\bar{A})$;

verbally, replace characters 1-10 inclusive of the word in the memory position specified by the third address of the extract order by the characters 1-10 inclusive of the result of shifting the word in the memory position specified by the first address in the extract order.

$$\nu_3(_0\bar{\Omega}(_2\bar{A})) = \begin{cases} \bar{2}, \text{ replace } {}_2\bar{A}(\bar{w}(_3\bar{A})) \text{ by } {}_2\bar{A} \text{ of shifted } \bar{w}(_1\bar{A}); \\ \bar{3}, \text{ replace } {}_3\bar{A}(\bar{w}(_3\bar{A})) \text{ by } {}_3\bar{A} \text{ of shifted } \bar{w}(_1\bar{A}); \\ \bar{4}, \text{ replace } {}_4\bar{A}(\bar{w}(_3\bar{A})) \text{ by } {}_3\bar{A} \text{ of shifted } \bar{w}(_1\bar{A}). \end{cases}$$

The above can be collected as one case:

$\nu_3(_0\bar{\Omega}(_2\bar{A})) = j$, replace $_j\bar{A}(w(_3\bar{A}))$ by $_j\bar{A}$ of shifted $\bar{w}(_1\bar{A})$;

$j = 1, 2, 3, 4$.

The first four cases are intended to be used primarily for orders.

$\nu_3(_0\bar{\Omega}(_2\bar{A})) = \bar{5}$, replace $S(\bar{w}(_3\bar{A}))$ by S of shifted $\bar{w}(_1\bar{A})$;

verbally, replace c_{44} (sign character) of, etc., etc.

$\nu_3(_0\bar{\Omega}(_2\bar{A})) = \bar{6}$, replace $D(\bar{w}(_3\bar{A}))$ by D of shifted $\bar{w}(_1\bar{A})$;

verbally, replace c_1-c_{43} (digits) of etc. etc.

$\nu_3(_0\bar{\Omega}(_2\bar{A})) = \bar{7}$, replace $w(_3\bar{A})$ by shifted $w(_1\bar{A})$

$\nu_3(_0\bar{\Omega}(_2\bar{A})) = \bar{0}$, replace whatever character (if any) in $\bar{w}(_3\bar{A})$ corresponds to $S(\bar{w}(_1\bar{A}))$ after the shift has occurred, by $S(\bar{w}(_1\bar{A}))$.

$\bar{6}$, $\bar{7}$ are intended to be used primarily with numbers; $\bar{5}$ with a zero shift gives a convenient method of obtaining absolute value, and $\bar{0}$ and the other versions of $\bar{5}$ permit generalized operations where signs and digits require interchanging.

The nature of the shift depends on the replacement code. In cases $\overline{1} - \overline{6}$, inclusive, the sign of $\overline{w}(_1\bar{A})$, the replacer, is discarded entirely, and a digit (character) may be shifted into its position; in case $\overline{7}$, the sign is retained and the digits (other characters) prohibited from occupying its position; in case $\overline{0}$, only the sign of $\overline{w}(_1\bar{A})$ is retained, but it may be shifted into any position whatever.

The subdivision of the order-type by using $_2\bar{A}$ to select the amount and direction of shift has not been specified. The first binary digit of $_2\bar{A}$ is naturally selected for the direction, since only two alternatives are possible.

$$\nu_1(B(_2\bar{A})) \begin{cases} = \underline{0}, & \text{selects a left shift (divide by } (1/2)^n; \\ = \underline{1}, & \text{selects a right shift (multiply by } (1/2)^n. \end{cases}$$

It must be borne in mind in numerical interpretations that we are shifting the digits and not the binary point. In a fixed binary point computer, the point is always considered, from the machine's eye point of view, to be immediately in front of c_1, the most significant digit. Scale factors are part of the programming.

The amount of shift can be specified by $_2\overline{\Omega}$ $(_2\bar{A})$ and, $_1\overline{\Omega}$ $(_2\bar{A})$, the first and second octonary (octal) digits, respectively, of the second address. This gives 64 possibilities (0-63) which are more than adequate; a shift of 44 or more accomplishes the same thing, destruction of all characters (or at least all digits, in replacement code $\overline{7}$). It is most convenient in designing the circuits to permit shifts only up to and including 47. If a greater shift is called for in the coding the circuit will perform a shift which is 16 less than coded. We note that zero shift gives the same result, whether left or right; $+0 = -0$.

The numerical interpretation of the two octal (octonary) digits concerned is:

$$N_E = \sum_{i=1}^{i=2} v_3(i\bar{2}(2\bar{A}))8^{i-1}, \quad 0 \leq N_E \leq 63.$$

The amount of shift, Amt, is

$$Amt = \begin{cases} N_E, & 0 \leq N_E \leq 47; \\ N_E - 16, & 48 \leq N_E \leq 63. \end{cases}$$

Ordinarily, the order will be restricted to $N_E \leq 47$, since larger values produce results which are already included in the case $N_E \leq 47$. The existing formal symbolism gives complicated results when applied to the result of a shift, so the following description is informal.

For replacement codes $\overline{1}$ - $\overline{6}$, a right shift produces a word which appears as

0000...0X...XXX,

where the number of zeros on the left is equal to the amount of the shift, Amt. If Amt = 0, then the word appears as

XXXXX ... XO,

since the sign is deleted. The X's indicate character of the replacer $\overline{w}(1\bar{A})$. For a left shift, the word appears as

XXXX ... XO ... 000,

Where the number of zeros on the right is one more than the amount of the shift, since the sign is deleted. If Amt = 0, we get the same result as in the left shift,

XXX ... XO.

For replacement code $\overline{7}$, a right shift produces a word which appears as

000 ... 0XXXXS,

where S is the sign of $w(_1\dot{A})$. For a left shift, the word appears as

XXX...XO...000S.

In both cases the number of zeros corresponds to the amount of the shift. For replacement code $\overline{0}$, a right shift produces a complete blank, unless Amt = 0, in which case the word appears as,

000...0S.

A left shift produces

000...0S0...0,

where the number of zeros on the right is equal to the amount of the shift. When the replacement occurs, the character in the replacee corresponding to S, is replaced by S. If the shift is to the right, and Amt> 0, no character will be replaced whatever, and the replacee survives intact.

Some examples will be explained to clarify the discussion, using informal symbols.

$$1^{\overline{A}}\ \overline{1\,2\,4}\ 3^{\overline{A}}\ 4^{\overline{A}}\ E.$$

Suppose we have the above order, where E represents the order-type extract. On the first execute, the $\overline{w}(_1\overline{A})$ is sent to the recirculating system where its sign is deleted, it then passes into the shifting and processing delay, over the channel marked XP in figure 1. The digits $\overline{1\,2\,4}$ select the amount and direction of shift in the shifting delay. The shifted word then is stored in a computer short tank, travelling over the channel marked SW. One minor cycle later, and subsequently thereafter, it is available in the recirculating system viz ZE. When $3^{\overline{A}}$ is executed, the digit $\overline{4}$ selects characters 31-40 of the shifted word, and 1-30; 41-44 of $\overline{w}(_3\overline{A})$ and the result immediately replaces $\overline{w}(_3\overline{A})$. Since 31-40 of $\overline{w}(_3\overline{A})$ corresponds to $_4\overline{A}(\overline{w}(_3\overline{A}))$, we have replaced

the fourth address of $\overline{w}(3^{\overline{A}})$. Since the shift is 24 (= 20, decimal / binary notation), to the right (1), the second address of $\overline{w}(1^{\overline{A}})$ was shifted into the fourth address position. The net result of the order is to leave $\overline{w}(1^{\overline{A}})$ unchanged, but $4^{\overline{A}}$ of $\overline{w}(3^{\overline{A}})$ now equals $2^{\overline{A}}$ of $\overline{w}(1^{\overline{A}})$; the fourth address of the replacee has been replaced by the second address of the replacer. Shifts of zero, 1o, 20, 30, ($\overline{00}$, $\overline{12}$, $\overline{24}$, $\overline{36}$) will generally be used with the first four replacement codes. The following table shows how to replace an address in the replacee by any desired address in the replacer:

		Replacee			
		$1^{\overline{A}}$	$2^{\overline{A}}$	$3^{\overline{A}}$	$4^{\overline{A}}$
Replacer	$1^{\overline{A}}$	$\underline{1}$ $\overline{000}$ / $\underline{0}$	$\underline{1}$ $\overline{122}$	$\underline{1}$ $\overline{243}$	$\underline{1}$ $\overline{364}$
	$2^{\overline{A}}$	$\underline{0}$ $\overline{121}$	$\underline{1}$ $\overline{002}$ / $\underline{0}$	$\underline{1}$ $\overline{123}$	$\underline{1}$ $\overline{244}$
	$3^{\overline{A}}$	$\underline{0}$ $\overline{241}$	$\underline{0}$ $\overline{122}$	$\underline{1}$ $\overline{003}$ / $\underline{0}$	$\underline{1}$ $\overline{124}$
	$4^{\overline{A}}$	$\underline{0}$ $\overline{361}$	0 $\overline{242}$	$\underline{0}$ $\overline{123}$	$\underline{1}$ $\overline{004}$ / $\underline{0}$

Some examples of the other replacement codes follow.

$1^{\overline{A}}$ $\underline{1}$ $\overline{225}$ $3^{\overline{A}}$ $4^{\overline{A}}$ E.

This replaces the sign of $\overline{w}(3^{\overline{A}})$, $S(\overline{w}(3^{\overline{A}}))$, by the 26th (44-$\overline{22}$=44-18=26) character of $\overline{w}(1^{\overline{A}})$.

$1^{\overline{A}}$ $\underline{1}$ $\overline{005}$ $3^{\overline{A}}$ $4^{\overline{A}}$ E,

will delete the sign of $\overline{w}(3^{\overline{A}})$, $S(\overline{w}(3^{\overline{A}}))$, as will also any second address of the type $\underline{0}\overline{XX5}$.

$$1^{\bar{\mu}}\ \underline{1}\ \overline{246}\ 3^{\bar{\mu}}\ 4^{\bar{\mu}}\ E,$$

will replace the "digits" of $|\bar{x}|\ (\bar{w}(3^{\bar{\mu}}))$, i.e. $D(\bar{w}(3^{\bar{\mu}}))$, by the "digits" of $|2^{-20}\ |\bar{x}|\ (\bar{w}(1^{\bar{\mu}}))|$, i.e. $D(\ |\bar{x}|\ -1\ \overline{|2^{-20}\ |\bar{x}|\ (\bar{w}(1^{\bar{\mu}}))|}\ )$; the multiplication is performed with the $\overline{|\ |}$ round-off. The result has the "sign" of $\bar{w}(3^{\bar{\mu}})$, i.e., $S(\bar{w}(3^{\bar{\mu}}))$, in other words the operation replaces $\bar{w}(3^{\bar{\mu}})$ by

$$D(\ |x|\ -1\ \overline{|2^{-20}\ |\bar{x}|\ (w_{1^{\bar{\mu}}})|}\ )\ S(\bar{w}(3^{\bar{\mu}})).$$

$$1^{\bar{\mu}}\ \underline{1}\ \overline{247}\ 3^{\bar{\mu}}\ 4^{\bar{\mu}}\ E,$$

will multiply $1^{\bar{\mu}}$ by 2^{-20}, preserving the original sign. The formal representation is similar to the above, except that $S(\bar{w}(3^{\bar{\mu}}))$ should be exchanged for $S(\bar{w}(1^{\bar{\mu}}))$. Note that in this case a minus zero may result.

$$1^{\bar{\mu}}\ \underline{0}\ \overline{010}\ 3^{\bar{\mu}}\ 4^{\bar{\mu}}\ E,$$

replaces the 43d digit ($43 = 44 - \overline{01}$) of $\bar{w}(3^{\bar{\mu}})$ by the sign of $\bar{w}(1^{\bar{\mu}})$ (informal notation). Any second address of the type $\underline{1}\ \overline{XX0}$ (except $\underline{1}\ \overline{000}$) fails to change $\bar{w}(3^{\bar{\mu}})$ at all, and nothing is accomplished except to direct the dispatcher to $P(4^{\bar{\mu}})$ for its next order.

After the replacee has received its new characters (if any) and immediately returns to the memory, the last execute sends $4^{\bar{\mu}}$ to the dispatcher to continue the program chain.

This concludes the discussion of the eleven order-types:

V,	$\bar{1}$ −,	Visual Display	m	$\bar{4}$ −	Exact multiplication
H,	$\bar{6}$ +,	Halt	D	$\bar{5}$ +	Divide with round-off
A,	$\bar{2}$ +,	Add	d	$\bar{5}$ −	Exact division
S,	$\bar{3}$ +,	Subtract	W	$\bar{2}$ −	Wire
C,	$\bar{1}$ +,	Compare	E	$\bar{3}$ −	Extract
M,	$\bar{4}$ +,	Multiply with round-off			

The unused order-types are $\bar{0}+$, $\bar{0}-$, $\bar{6}-$, $\bar{7}+$, $\bar{7}-$. If the dispatcher receives any of these, the machine will halt and give the indication "unused orders". A complete blank has significance to the computer (zero) but not to the dispatcher. The EDVAC will not run automatically on empty tanks; it will not even start with an empty dispatcher. It thus becomes necessary, after initial clear, to insert an order which is not a complete blank into the dispatcher to start up the problem.

5.2 Speed of Operation.

The time required to perform an operation, where the EDVAC is running continuously is defined as follows: It is the time elapsing from the instant the order under consideration just starts to leave the memory to the instant the next order just starts to leave the memory. This time, for all the order-types, except H and W, can be precisely defined as a function of $1\bar{A}$, $2\bar{A}$, $3\bar{A}$, $4\bar{A}$, $\bar{T}$, and exhibited by a set of tabulated values. In what follows, however, the $j\bar{A}$ will be assumed:

(1) to be optimally distributed, giving minimum time;

(2) randomly distributed, giving average time;

(3) distributed in the worst possible way, giving maximum time.

Since H involves a cessation of continuous operation the definition does not apply; suffice it to say that the dispatcher will send the halt signal to the alarm circuit within 500 μsec after receiving the order. The wire order involves asynchronous operation and does not give unique values of speed. The present design of the wire drives will permit a word to be read from or recorded on the wire in about 35 milliseconds.

In the following table, the number of minor cycles is given in parentheses, and the first number is the time in microseconds.

Order-Type	Time required		
	Minimum	Average	Maximum
V, C, E,	192 (4)	696 (14.5)	1200 (25)
H	Not applicable		
A, S,	192 (4)	864 (18)	1536 (32)
M	2208 (46)	2880 (60)	3552 (74)
m, D, d,	2256 (47)	2928 (61)	3600 (75)
W	About 35 milliseconds per wire position moved		

The ENIAC adds in 200 μsec, and multiplies in 2800 μsec; division is much slower. The EDVAC, with more flexibility and much less equipment, has about the same operating speed as the ENIAC.

5.3 Controls

Figure 3 (drawing 104-7MD-1) shows the main control panel. At the extreme top we have a bank of 10 neons; the initial address register. This is provided to enable the machine to be shut down in the middle of a problem without losing any results. The initial address register stores the location of the next order. Immediately beneath the initial address register is the neon light register, which is a bank of 44 neons capable of holding a complete word and intended to be used to read the contents of the machine visually when desired. Below the two registers is the oscilloscope, which can be connected to any memory position in the EDVAC, including computer short tanks, dispatcher memory and shifting and processing delay. Immediately beneath the scope and on both sides are its controls (indicator selectors).

The next two rows contain the most important operating controls and buttons. The mode of operation switch determines whether the machine is to operate continuously or stop after performing single operations or parts of operations, and the memory bank switch permits, in an emergency, the operation of the machine when only half of the memory is functioning. The "start" button turns on the power and produces an initial clear, the "stop" button turns off all power and the "D.C. off" button turns off everything but the tube heaters, thus decreasing the probability of tube failures. These are all power controls and have only an obvious indirect connection with the computing operations. The "clear" button restores the EDVAC to the initial state, all memory positions empty, including the dispatcher, and all flip-flops reset except those where this is not essential. The "halt" button causes the EDVAC to finish any order in process, retain the order in the dispatcher memory, and wait for further instructions. The "initiate" button permits the dispatcher to proceed with the computation by obtaining its new order, either from information contained in $\overline{A}_4$ of the order it contains (or possibly $\overline{A}_3$ in compare) or in case the machine is being started from scratch, from the special order switches. These two alternatives are the normal and special modes of operation indicated on the mode of operation switch. The "read out" button is used for shutting down the EDVAC without losing any data. A row of pilot lights, just above the operating buttons, indicates the status of the machine, whether it is running or halted, power on and off, and other information of a similar nature.

The next two rows comprise the auxiliary input, previously described in the section on the wire order, and the excess magnitude option switches.

There is one switch for addition and subtraction, and one for division. The setting of these switches determines what happens if capacity is exceeded. A pilot lamp, immediately above each switch, indicates that capacity has been exceeded.

The switches which control the manual order generators are located beneath all the other controls. The set marked "Address A" is used to program break points without inserting a halt order into the program, and its use is controlled by the mode of operation switch. The set marked "Address B" is associated with the excess magnitude option switches. Finally, the special order switches, located in the bottom rows, are used primarily for inserting the first order into the dispatcher when the EDVAC is in the cleared state.

5.31 Mode of Operation.

This switch has two indexes on the dial, "normal" and "special", and five indications on the panel:

(1) To completion,

(2) To address A,

(3) One order,

(4) One execute,

(5) One cycle.

It is physically impossible to set the special index to the first two indications, which refer to continuous operation. Only the bounded modes of operation are possible on "special". All five indications are accessible to the normal index.

"To completion" indicates that it is desired to run until a programmed halt enters the dispatcher, at which time a gong rings and a pilot light

comes on. "To address A" indicates that it is desired to run until an order is called for which is stored in the memory position identified by address A. The address is set in on the address A switches. This arrangement permits break points to be inserted into a program without using any halt orders, and using the halt order as a unique indication of the completion of the problem. The break points can be established even after the problem is on the machine. When address A is reached, the EDVAC halts and gives an indication on a neon lamp. If it is desired to continue, it is only necessary to depress the initiate button.

The above are the unbounded or continuous modes of operation. The bounded or step-by-step modes always require the EDVAC to halt as soon as the current order is completed, or even before. "One order" requires the EDVAC to halt as soon as the current order is completed; "One execute" requires it to halt as soon as one execute has occurred (or a reading of the special order switches into the dispatcher or one wire position has been moved). "One cycle" is the same as one execute, except in multiplication and division. In these cases, as soon as the arithmetical operations start, the machine halts as soon as one partial product or one quotient digit is computed. Except for "one order" these modes are intended for testing, and are somewhat analogous to "one add time" on the ENIAC.

The normal mode of operation permits the dispatcher to obtain its next order in the usual way, as described in section 5.1. The special mode of operation interferes with this, by insisting that the dispatcher obtain its next order from the special order switches. These switches permit any arbitrary order to be set up. Since nothing useful would be accomplished by performing an identical order ad infinitum, continuous

or unbounded modes are not permitted on special. The orders are identical to the regular programmed orders, except that it is not possible to set up any unused order-types on the switches.

5.32 Excess magnitude options.

In connection with the orders A, S, and D, d, the possibility of an exceed capacity signal was mentioned. Ordinarily this indicates an error in programming. If this is the case, both switches are set to "halt". If the signal reaches the dispatcher the EDVAC halts, and a pilot light over the switch comes on. If for special reasons, it is desired to ignore the signal entirely, the switches are set to "normal". If it is desired to automatically take account of the occurrence of the signal, two more options are provided "special" and "address B". If the switch is set at "special" the dispatcher will read the order set on the switches instead of its regular next order, and the machine continues without stopping. If the switch is set at "address B", the dispatcher will read the order stored in the memory position designated by address B, and continue without stopping. One option can be chosen for A,S, and the other for D,d.

5.33 Memory bank switch.

Three positions are provided. Normal (LR), "L-1", and "R-0". The high-speed memory is divided into two halves, one located at the left end of the EDVAC, and one at the right. Each of the duplicate computers has its short memory tanks mounted in one of the temperature stabilized cases which contains half of the memory. All of the memory positions selected when $\nu_i\,(B(\overline{A}_j)) = \underline{1}$ are in the left case, (hence L-1), and all of the memory positions selected when $\nu_i\,(B(\overline{A}_j)) = \underline{0}$ are in

the right case, hence R-O. In other words, the first, binary, digit of the address chooses between the right and left memory banks. In the event of failure of temperature stabilization in one case, the switch can be set to indicate the half still working. The dispatcher will then ignore the first digit of every address and refer all executes to the memory which is unimpaired. Special programming will usually be required, but only one rather than two distinct programs. One of the duplicate computers will be out of service, so all checking of duplicate results is discontinued.

5.34 Read out.

The read out button is used to shut down the machine without losing the place in the problem. When the button is depressed, the current order is completed, and then a set of latching relays is unlatched. The address of the next order to be performed is sent from the dispatcher to the latching relays which then latch and retain the information until another read out occurs. The occurrence of a programmed halt or an exceed capacity is similarly stored. As soon as this is accomplished, enough information is available to restart the problem if the memory can be refilled with an exact duplicate of its current condition. In order to do this, the control sends a wire order to the dispatcher, which when executed reads the entire contents of the high-speed memory to wire 3, moving in the forward direction. This order is then carried out, and the contents of the memory is permanently recorded. All power is shut off after this is accomplished. The latching relay indications are read on lamps which will go out, but when the power is turned on, will read the status of the latched relays correctly.

6. Example of operation

Figure 4 (drawing 104-3LD-4) is a block diagram of one-half of the double computer, less the unduplicated part which contains the checking circuits. Most of the equipment used in multiplication and division has been removed in order to avoid confusion. The operation of (algebraic) addition which is basic to the computer will be described briefly, with reference to the diagram. Symbols in parenthesis, such as (D1) refer to the "map coordinates" at the top and left of the drawing.

The input to the computer is the hexagon symbol IN(C1). Hexagons indicate remote connections to other components or perhaps to other places on the same drawing. The symbol ASC indicates that the connection is energized when either of those orders is being performed, and similarly for SC. -C denotes a connection which is energized when C is not being performed. (In the propositional calculus we would use AvSvC, SvC, $\sim$ C; "and" is not symbolized by juxtaposition. The operations are mutually exclusive).

The computer performs algebraic addition by noting the signs of the numbers involved, and whether they are to be added or subtracted, and then performs the appropriate arithmetic (signless) operation in the adder-subtractor. If arithmetic subtraction occurs, a complement may result. This information is needed to determine the sign of the result, and is fed to an output sign circuit. The digits are then decomplemented before the sign is attached.

The rules are tabulated as follows:

Order	Sign of augend or minuend (first word)	Sign of addend or subtrahend (second word)	Arithmetic Operation Required	Number of Minus Signs
A (+)	+	+	Add	0
A (+)	-	+	Subtract	1
A (+)	+	-	Subtract	1
A (+)	-	-	Add	2
SC (-)	+	+	Subtract	1
SC (-)	-	+	Add	2
SC (-)	+	-	Add	2
SC (-)	-	-	Subtract	3

It will be noted that if the number of minus signs is even, the arithmetic operation is add, if odd, subtract. The one stage binary counter located at (DO) performs the count. When execute 2 occurs, if S or C is the order, a pulse is fed to the counter via X1, A4a, A5, X4. X4 is shut off except on ASC since the counter is used in a different way in multiplication and division. The counter counts the sign in the first column of the table. When the first word, $\overline{w}(\overline{A}_1)$, appears at LN, its sign is detected by a timing pulse P_1, at X2, and sent to the counter via A4b, A5, X4. When the second word, $\overline{w}(\overline{A}_2)$, appears at LN, its sign is detected by the same circuit. The counter is initially cleared to zero by RF, before each operation, so if the number of signs is odd, it energizes the output LSG, (L subtract gate) and even.LAG (L add gate). This constitutes the input sign circuit, or arithmetic operation selector.

The digits of the first word, $\overline{w}(\overline{A}_1)$ are selected by DP (digit

pulses) at X3(C1) and sent to the computer short tank via d7 (a timing delay), B1b, C1, (gate X6 is closed at this time) d10, etc., to the short memory system (B2). This has just been cleared by flip-flop G2 and memorizes the digits of the first word by means of the regeneration loop X20, d8. When the second word, $\overline{w}(\overline{2})$ appears at LN, another execute has occurred, and the regeneration loop closes at X20, thus clearing the tank. The gate X14(A2) opens however and the digits of the first word are sent to the adder-subtractor via X12 (A3) and d6 (F1). Remote connections 1, 2, 3 are energized while the first, second and third addresses, respectively, are being checked. Immediately after the execute occurs, the next in the sequence becomes energized. The execute counter has already advanced to 2, when the first word appears at LN, and has already advanced to 3, when the second word appears at LN. The second word just arriving has its digits selected by DP at X3(C1) and replaces the first word in the memory tank. By this time however, gate X6(C3) is open and the digits of the second word reach the adder-subtractor via d5(E1). Both sets of digits now appear in synchronism at the adder-subtractor input. The adder-subtractor is a relatively simple circuit containing 16 tubes, which performs M + S serially if its add gates are energized or either M-S (if $M \geq S$) or $2^n - M + S$ (if $M < S$) when the subtract gates are energized; M and S are non-negative numbers. The latter two cases can be distinguished by the fact that a pulse (1) turns up abnormally late, corresponding to the string of 9's produced at the left by a desk calculator.

The analysis of the sign of the answer will be subdivided into

the two cases, arithmetic add and arithmetic subtract; in the circuit these are distinguished by whether LAG or LSG is energized.

Case 1 - Arithmetic add (LAG).

The following is a table which shows how the sign of the result is determined:

Order	Sign of augend or minuend (first word)	Sign of addend or subtrahend (second word)	Sign of Result
A (+)	+	+	+
A (+)	-	-	-
SC (-)	-	+	-
SC (-)	+	-	+

In this case the sign of the result is determined in advance and no further information is needed. The sign of the first word, and only the first, passes through X23(D3) into another binary counter, the output sign counter (D4). Only the sign of the first number gets through because of the 2 on gate X23, and the other input to X24 from L11a and X34 (E3) is closed because LSG is not energized. Triangles with lines perpendicular to the base, continuing the input leads, represent or gates, otherwise an and gate. This counter is initially cleared to zero prior to each operation by RF. If this counter is on 1, then the answer is negative, and if on 0 the answer is positive. This is accomplished by "counting" the minus sign of the first number (plus is no pulse; minus, a pulse).

Case 2 - Arithmetic subtract (LSG)

The table shows how the sign of the result is determined:

Order	Sign of Augend or minuend (first word	Sign of addend or subtrahend (second word)	Sign of result	
			M ≥ S	M < S
A (+)	-	+	-	+
A (+)	+	-	+	-
SC (-)	+	+	+	-
SC (-)	-	-	-	+

In this case the sign of the result is not determined in advance. However, if we count the sign of the first word only as was done in case 1, then we get the correct result if $M \geq S$, and the incorrect result if $M < S$. If $M < S$, a "borrow" propagates to the left in the adder-subtracter, which is detected at X34, since LSG is now energized: M and S are non-negative numbers. A pulse then appears at the output sign counter which reverses it. The sign of the result can now be read at this counter in the same way as in case 1.

The sign having been determined, the digits remain to be considered.

Case 1 - Arithmetic add (LAG)

In this case, the digits do not require any complementing but we may have a carry to the left, indicating $M + S \geq 1$; M and S are non-negative numbers. This carry is detected at X35, where it sets flip-flop G10 (F3). Flip-flop G10 emits the exceed capacity signal (LASCE) to the dispatcher, and permits a pulse P_{44} to be emitted at X36. This occurs nearly a minor cycle later than the occurrence of the original carry, but the digits pass through the short memory system (F4) meanwhile, so P_{44} and the extra carry arrive at the half-

adder G9(F6) simultaneously. The half-adder is a circuit which emits a pulse if an odd number of pulses are presented on its two inputs, otherwise not. It consequently adds single pairs of digits but cannot propagate a carry. The P_{44} and the extra carry thus cancel out at the half-adder and the superfluous digit is eliminated. The digits begin to appear at LA(G9) via F8, d20, X57, D12a, C12, one minor cycle after the second number starts to enter at LN. The connection to LAT(F9) is to a checking circuit which is comparing the digits formed in the other half of the double computer. The digits also return to the adder via d22, X39, C76, and X11, as long as X39(G4) permits them to pass through. Nothing is present at the other adder input, so the digits pass through unaltered. This loop serves as a memory until the dispatcher is ready for the digits. As soon as that occurs X39 closes because 3 is deenergized and the memory regeneration loop clears. a is always energized on ASC. The process of cancelling the carry and producing exceed capacity signal is not performed on compare, since -C is deenergized on compare and X35(F3) closes.

Case 2 - Arithmetic subtract

In this case, no exceed capacity can possibly occur, since $\max |M-S| \leq \max(M,S) < 1$. On the other hand, the result of the adder-subtractor must be decomplemented if $M < S$; M and S are non-negative numbers. This event is detected at X34, where it sets flip-flop M11(E3). This permits clock pulses to pass through X33(E5) as soon as L10 is set. This occurs immediately after the first 1 digit (pulse) reaches d18. Clock pulses now reach G9 simultaneously with the digits of the complement. The least significant unit (the first, reading

from right to left) and all zeros to the right of (earlier than) it, are unchanged, but all digits to the left of (later than) it, are changed from 0 to 1 and vice versa. This gives a 2's complement at the output of G9. The digits now have the proper value and are free to leave or recirculate as in case 1.

Flip-flop N10(F4) serves as a zero detector. If no 1's are present, then N10 fails to be set, and the sign (minus) cannot get through X59 (F9). This requires all zeros to have a plus sign. The sign comes from the counter via X29(D4) which is read by a pulse at "sign time" every minor cycle. The sign is compared with the other half of the double computer via LSC. LB(F9) is a switching voltage which cancels the output when the EDVAC has one memory tank out of service.

Figure 5 (drawing 104-10LB-1) is a collection of standard symbols used in EDVAC block diagrams. A ring counter has as many stable states as it has stages, being so constructed that one and only one stage can be "set", the others are always "reset". A counting pulse causes the next higher stage to become set. "Higher" is interpreted cyclically. In a binary counter, each stage can set or reset independently of the others; it has 2^n stable states, where n is the number of stages. Crystal combinations are circuits containing 1N34 germanium diodes, whose outputs become positive (negative) if one or more (all) of the inputs are positive (negative).

7. Aids to Maintenance

Small and compact as the EDVAC is, compared to the ENIAC, its approximately 3000 vacuum tube circuits present a fairly complex maintenance problem. It is therefore desirable to include in the desig as many aids to maintenance as are consistent with the requirement for compactness and simplicity. On the other hand, an unduly elaborate checking system introduces its own maintenance problem, and if carried too far, becomes difficult for the operator and maintenance man to interpret.

In the EDVAC, therefore, the following aids to maintenance have been included:

A. Provision for examination of words in various parts of the machine by means of indicators on the Control panel.

B. Provision, by means of the "Mode of Operation" switch on the Control panel, for short period operation for various convenient intervals, in addition to the normal modes of operation. For example, computation of one quotient digit or one partial product summation, or the transfer of a single word.

C. An abnormal halt indicator on the Control.

D. Indicator lamps on the Dispatcher panel, which are actuated by:

(a) Error detecting circuits in the Dispatcher.

(b) An error detecting circuit in the Reader-Recorder.

(c) Exceed capacity circuits in the Computer.

(d) Error detecting circuits in the Computer.

E. Neon lamps on all flip-flops.

By means of switches on the Control panel, it is possible for the operator to observe the word in any position in any of the long tanks, in any of the short tanks, or in the dispatcher memory. As this informa-

tion is recurrent, it may be displayed on either the oscilloscope or the neon lamp register. The in and out memory buses also may be examined, but only by means of the neon lamp register, since this information is transient. Oscilloscope sweeps are provided for examination of the entire contents of a long tank, or of a single word. The sweep may be expanded for detailed examination of a part of a word.

The "Mode of Operation" switch on the Timer panel permits operation for 1 order, 1 execute or 1 cycle. Operation may be in accordance with the programmed routine, or in accordance with a special order. In this manner, any desired operation may be observed in steps short enough to permit localizing of a source of malfunction.

An example may help to indicate how the various aids to maintenance are used.

Let us suppose that the EDVAC is operating with the Mode of Operation switch set at "Normal to address A". Suppose now that an audible signal is given out, and the operator observes that the light over the "Halt" button is on. This will indicate that one or more of the following events have occurred:

A. A halt has been programmed by means of an order on the input wire.

B. Address "A" has been reached in the computation.

C. A coding error has been detected.

D. A malfunction has been detected.

The operator should first look at the "Abnormal halt" indicator on the Control panel. If this light is off, the operator should set the indicator selector switch to "Control." If the last four digits read 1100, event A has occurred. If the fourth address agrees with address "A", event B has occurred. In either case, the operator will

proceed in accordance with the problem instructions.

However, if the "Abnormal halt" neon indicator is on, either event C and/or event D has occurred, and the operator should look at a group of six halt-type neons on the control panel. If the "Wire code error" neon or the "Unusable Order" neon is on, a coding error has been detected. These errors can be studied by means of the oscilloscope, and corrected by means of the manual controls. The "Wire code error" neon indicates that the order "Read 5th address backward", which cannot be executed, has been programmed. The "Unusable Order" neon signifies that a type of operation, other than the eleven (A, S, C, E, etc.) for which the machine is designed, has been programmed.

If the "Reader-Recorder halt" neon is on, it indicates either that pulses have been erroneously recorded in the spaces on the wire, or that the wrong number of digits per word has been recorded.

If the "Pulses in Blanks" neon is on, it indicates that a malfunction has occurred, introducing unwanted pulses in the blanks in words in the "In" or "Out" memory buses.

If either the "Exceed Capacity Halt" or "Computer Error" neon is on, it is necessary to refer to flip-flop lamps on the Computer panel.

In the Computer, algebraic operations are carried on in parallel in two identical units. Operation of these units is compared by means of half adders and flip-flops at five points:

A. The adder loops.

B. The quotient loops (including the quotient round-off circuits).

The sum, difference, result and quotient sign circuits.

C. The remainder sign circuits.

D. The shuttle tanks.

The half adders are arranged to set the flip-flops if any difference is found between words at corresponding points in the two algebraic

units. This arrangement will also detect unwanted pulses in the blanks in either unit.

If the arithmetic capacity is exceeded, reference must be made to "Add-Subtract Capacity Exceeded" and "Divide Capacity Exceeded" flip-flop neons in the Algebraic Units. These indicate coding errors, which may be corrected by the operator.

The flip-flop neons enable these procedures to be extended to the determination of the particular circuits in which errors are arising, so that detailed checking may be limited to a small number of components.

To facilitate maintenance, the memory amplifier and gate units are in the form of readily replaceable plug-in units, as noted in an earlier section of this report. The flip-flops also are small plug-in units.

Cabinets have been designed to give convenient access to tubes and other circuit components, and a marking system will facilitate rapid location of particular circuit components.

8. Indicated Improvements.

8.1 Design Policy.

As work under contract W36-034-CRD-7593 progressed, it became necessary from time to time to freeze logical planning and design characteristics in order that the project should result in the completion of an EDVAC rather than merely the establishment of the specifications and techniques always representative of the current state of the art. Certain design characteristics which will be contained in the completed equipment will be representative of the state of the art two years ago, certain characteristics will be representative of the state one year ago and certain others representative of the current state. As so often happens in development projects of this nature, the freezing of design has occurred later than had been anticipated, partly because early design decisions were unsound and partly because it is never possible to prevent research-minded personnel from continuing to incorporate desirable improvements resulting in a better instrument but accompanied by a later completion date. It is estimated that a delay of approximately one year has been incurred on these accounts. On the other hand, the instrument will be a full-scale device of performance and capacity far beyond that envisioned in the preliminary report or in the contract and its supplements. The equipment will in many ways be representative of the current state of digital computer

art. There are a number of features, however, reflecting early design decisions which, if redesigned, would admit of considerable improvement; there are also certain features not included in the equipment, having to do with logic and scope, which would add substantially to its usefulness. The Moore School recommends that serious thought be given to such modifications in connection with the planning for additional EDVAC type equipments. Specifically, these features are set forth in the following paragraphs:

8.2 Recommended Improvements in Design.

The input-output equipment is designed to use magnetic wire. The reader-recorder unit of the EDVAC proper contains a wire handling equipment designed to use magnetic clutches in the servomechanisms. The equipment external to the EDVAC (the inscriber and outscriber) being developed by the Bureau of Standards consists primarily of modified teletype equipment. This equipment prepares two paper tapes, the first of which is used to check the second in a verifier. The second tape in turn prepares the magnetic wire for insertion into the reader-recorder unit of the EDVAC. At the time decisions were made to build equipment along the lines indicated above, the desired early date of completion was the dominant factor. It was believed that modification of teletype equipment and the use of wire rather than magnetic tape would represent the most satisfactory solution consistent with minimum development time. In retrospect,

this decision seems to have been unwise. It is recommended that any new EDVAC equipment use a multiple channel tape as the input-output medium rather than wire. It is recommended that suitable inscriber and outscriber equipment be custom designed for the purpose without prejudice on account of efforts to use available equipments. Elimination of intermediate tapes can, with proper design, effect a great simplification at no sacrifice in provisions for checking accuracy. It is recommended that the wire handling equipment in the reader-recorder unit use servo systems not containing clutches so as to eliminate the objectionably high maintenance attendant to the use of clutches.

It is believed that by the use of RF links and RF flip-flops, equivalent performance could be obtained with substantially simplified equipment. It is expected that there would be a saving in tubes, a decrease in the number of D.C. levels required and a decrease in the power requirements.

It is recommended that higher repetition rates than one megacycle be considered. It is recognized that in order to obtain acceptable rise times at higher frequencies it is necessary to increase the power delivered to the electronic components with attendant operation at higher percentages of normal ratings and resultant increases in casualty rate. Since dependability is of

paramount importance, substantial increases in repetition rate are considered undesirable but an increase to two megacycles is certainly feasible. Three megacycles are considered marginal, but with careful design may prove to be practicable.

8.3 Functional Improvements which Merit Consideration.

The following features are considered of sufficient utility that their incorporation in contemplated new EDVACS should be considered.

(a) Provision of coded decimal to binary and binary to coded decimal converters as part of the input-output system to speed operation by removing the necessity for performing these relatively slow operations within the EDVAC proper and to eliminate special coding. It seems that this could be done most expeditiously by means of relays incorporated in the Inscriber and Outscriber.

(b) Provision of both relative and absolute numbering systems in both tape and high speed memory.

(c) Provision of additional reading and recording mechanisms which would permit unlimited input or output of data without stopping the equipment.

(d) Provision of additional "Order Types" to give greater flexibility and to facilitate coding.

(e) Provision of an accumulator to permit rapid Sigma or Pi operations.

(f) Addition of magnetic drum intermediate speed memory in order that external memory need not be as slow as magnetic tape would demand.

8.4 Special Purpose Features

There are two functions not necessary to a general purpose EDVAC and therefore not included in 8.3 which are regarded as highly desirable for special purpose machines. It is recommended that consideration be given to

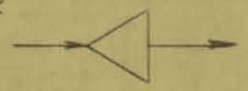
(a) Provision of facilities for sorting of an alphabetical nature and for printing out of alphabetical characters.

(b) Provision of a random number generator for use in statistical problems.

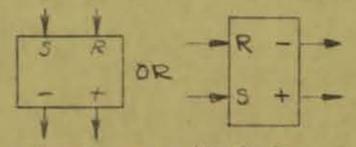
8.5 Coded Decimal Versus Binary Computer.

No report on EDVAC would be complete without reopening the highly controversial point as to whether the reduction in equipment effected by the use of a binary computer represents real economy. There is no doubt that trouble shooting within the binary part of the EDVAC and programming of problems for solution on the EDVAC will demand a knowledge of the binary system. Further, the extent to which this may cause loss of time in consultation with potential users not familiar with binary arithmetic is not known. No experience exists for evaluating difficulties which may arise on this account since no binary machine has been built. It is believed the binary EDVAC will be a reasonably satisfactory instrument but maintenance and programming experience is probably the only means for obtaining a categoric answer to the question.

AMPLIFIER

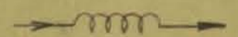


FLIP FLOP

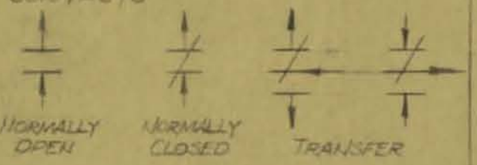


SYMBOLS R, S & +,− RESPECTIVELY MAY BE ARBITRARILY INTERCHANGED. OUTPUT POLARITIES ARE THOSE EXISTING WHEN RESET; (NEON OFF). OPPOSITE POLARITIES EXIST WHEN SET.

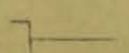
RELAY COIL



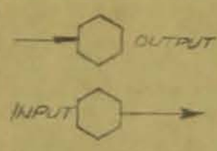
RELAY CONTACTS



MAGNETIC RECORDING, READING, ERASING HEAD

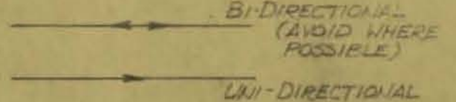


REMOTE CONNECTION

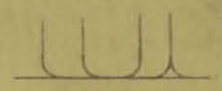


PULSE TRANSFORMER

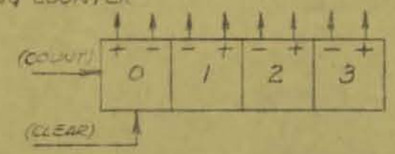
LINES



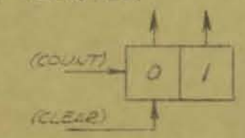
CABLES



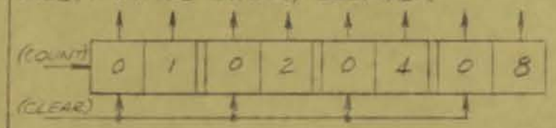
RING COUNTER



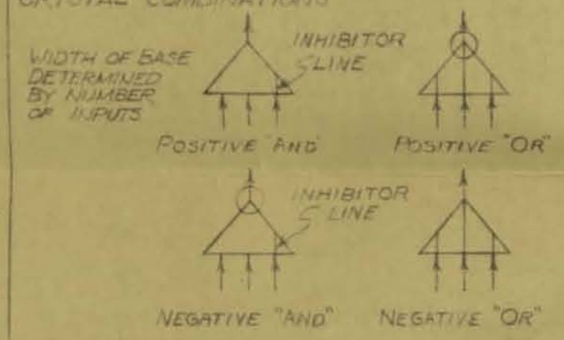
BINARY COUNTER



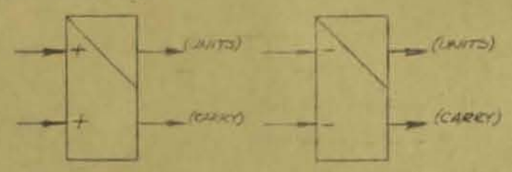
MULTI-STAGE BINARY COUNTER



CRYSTAL COMBINATIONS



HALF ADDERS

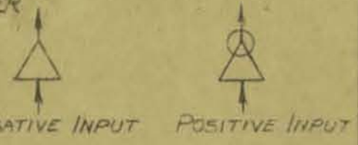


POSITIVE INPUTS — NEGATIVE INPUTS

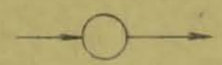
OMIT CARRY OUTPUT LEADS NOT USED

SEE DRG. 104-10LA-5

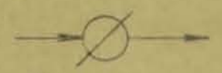
CRYSTAL BUFFER



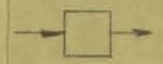
OFF TUBE



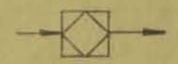
ON TUBE



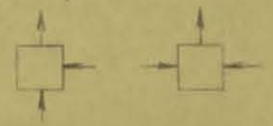
CATHODE FOLLOWER



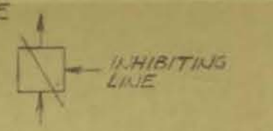
BLOCKING OSCILLATOR



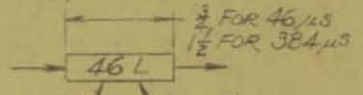
GATE



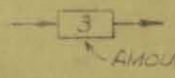
INHIBITOR GATE



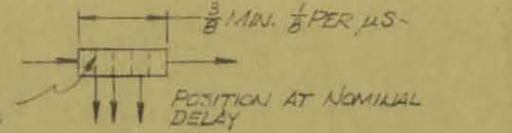
MERCURY DELAY



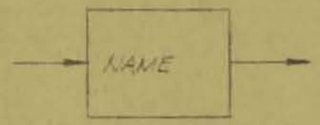
UNTAPPED ELECTRICAL DELAY



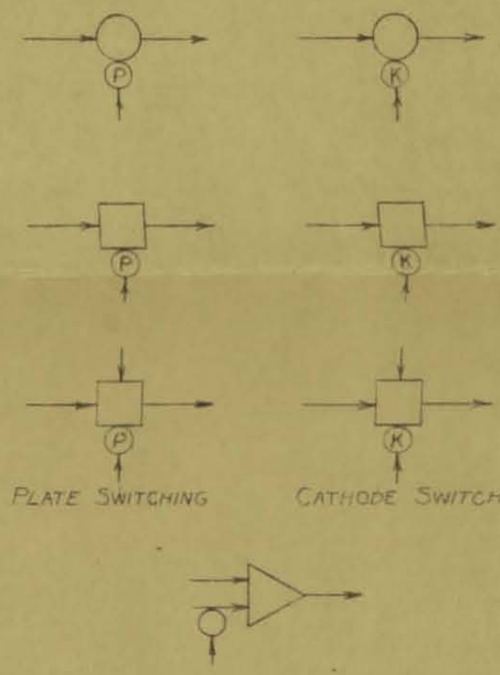
TAPPED ELECTRICAL DELAY



CATCH ALL (DRAW LARGER THAN FLIP-FLOP)



SWITCHING BY CONTACTS



POTENTIAL SWITCHING ON CRYSTAL COMBINATIONS

NEW SYMBOLS ADDED C.R.Th 9-26-47	1	
DELETED SUBTRACTOR INSERTED REMOTE CONNECTION	11-6-47	2
BLOCKING OSCILLATOR ADDED	12-1-47	3
REVISED HALF ADDERS	12-22-47	4

APR 9 1948

PRELIMINARY

FIGURE 5

MOORE SCHOOL OF ELECTRICAL ENGINEERING
UNIVERSITY OF PENNSYLVANIA

STANDARD SYMBOLS FOR BLOCK DIAGRAMS

SCALE FULL SIZE

DRAWN BY	CHECKED BY	APPROVED BY	
J.E. GRAVEL 9-10-47	C.R.Th		104-10LB-1

D7a
X12
A3C
3
X14
M4a
M4b
M7
LC0
d11
LZE
LC0
LC1
T4
E1
F1
E2
SHORT MEMORY SYSTEM (45)
DRWG 104-1L
ZJ
S +
G2
R −
F2a
F2b
P45
D3
d35
X20
D2
d10
C1
B1b
ASC
X 21
CP
X7
C1a
C1b
X8
CP
E36
E4
S
R
d15
P44
X6
C4
SC
1
EX
X1
A4a
LN
P1
X2
A4b
DP
X3
B3a
d8
d7
A5
A3C
X4
RF
0
1
A7
B8
LAG
LSG
A6, B6
B7, C6
M (INPUT 1)
S (INPUT 2)
d5
d6
X11
D5
LSA
BINARY
ADDER SUBTRACTOR
DWG. 104-3LC-3
LMA
L±
2
EX
A3C
X38
C3
T5
LB1
SHORT MEMORY SYSTEM (45)
DRWG 104-1L
LB2
2
A3C
X23
L11b
X24
K11a
0
1
H10
J10, N11
K11b
P1A
RF
K10b
A3C
X29
J12a
H12a
H13
T2
S +
M11
R −
M10b
P46
RF
A3C
MTH
Dd
P48
L11a
X34
P47
LAG
LSG
-C
A3C
X35
F10
P46
T1
RF
R +
G10
S −
G11
P44
LAB CE
X36
J9
CP
X33
S −
N10
E +
N11
M10a
LA1
LA1
SHORT MEMORY SYSTEM (45)
DRWG 104-1L
LA0
LA0
L10
E
S
L1P
S1P
P45
C7b
X39
α
3

NOTES

For right algebraic w
all remote connection de
with "L" and replace "LEFT"
it occurs in the Remote Connection Table

REMOTE CONNECTION TABLE		
INPUTS	FROM DRAWING	DESCRIPTION
2	104-2LD-1	Check 2rd address
3		Check 3rd address
4		Check 4th address
Ex		Execute
-YA		
ZJ	104-2LC-2	
ASC	104-2LD-5	Add or subtract or comp
C-		Not compare
D		Divide with round off
d		Exact Divide
Dd		Divide
MDmd		Multiply or divide
Mm		Multiply
-Mm		Not multiply
SC		Subtractor Compare
-SC		Neither Subtract nor comp
CP1	104-3LC-1	
CP4		
CP5		
CP6		
LM		Left minuend
LS		Left Subtrahend
LSW		Left Shifted Word in
DP		Digit pulses
P1A		
P1B		
P2A		
RF		Reset Flip Flops
L±	104-3LC-3	Left sum or difference
a	104-4LD-1	Non-precess
b		Precess
CN		Computing interlock
LB	Memory Bank SW. No Drwg Available (See 104-3LC-1)	
OUTPUTS	TO DRAWING	DESCRIPTION
LA	104-3LC-1	Left adder readout
LAT		Left adder tank
LQ		Left quotient readout
LQT		Left quotient tank
LRS		Left remainder sign
LSC		Left sign circuit
LST		Left shuttle tank
LZE		Left shifted word out
LAG	104-3LC-3	Left add gate
LMA		Left minuend arithmetic
LSA		Left subtrahend arithm
LSG		Left subtract gate
LDCE	Memory bank switch - See LB above	Left division capacity exce
LASCE		Left add-subtract capacity exceeded
LA1	Left memory unit no drwg available	to left short tank A
LB1		" " " " B
LC1		" " " " C
INPUTS	FROM DRAWING	DESCRIPTION
LA0	Left memory unit no drwg available	from left short tank A
LB0		" " " " B
LC0		" " " " C

Note:- This drawing IS a partial reproduction of 104-3LD-2 with equipment used only for multiplication and division omitted

PRELIMINARY

APR 8 1948

FIGURE 4

MOORE SCHOOL OF ELECTRICAL ENGINEERIN
UNIVERSITY OF PENNSYLVANIA

EDVAC ADD - SUBTRACT
BLOCK DIAGRAM

NOTES

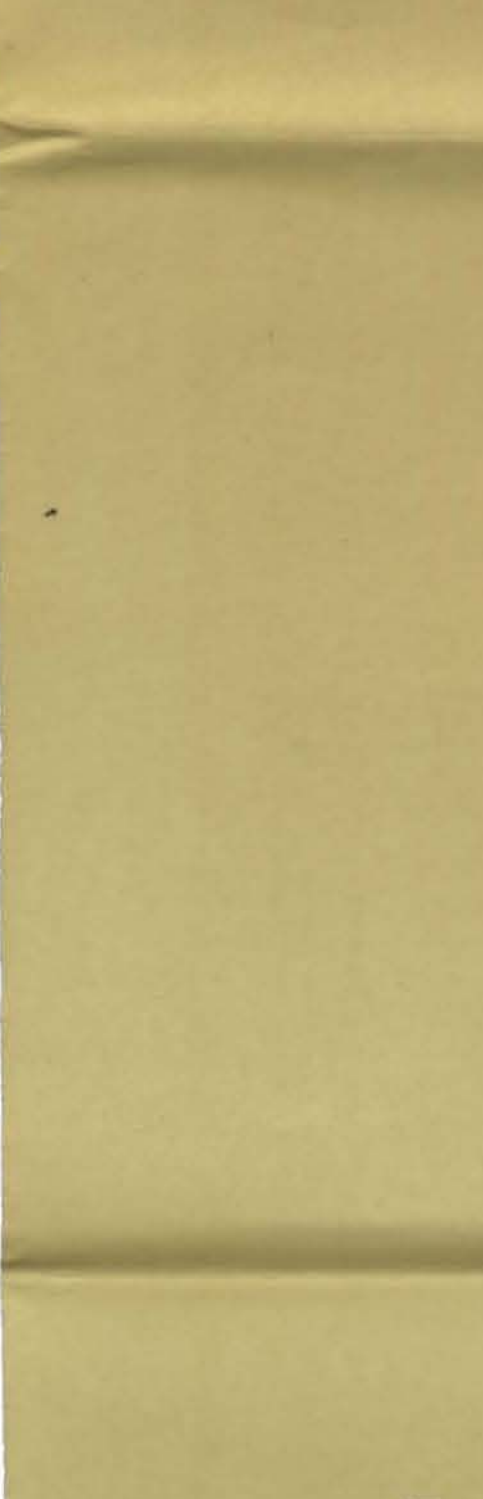
For right algebraic unit, replace "L" by "R" on all remote connection designations beginning with "L" and replace "LEFT" by "RIGHT" everywhere it occurs in the Remote Connection Table

REMOTE CONNECTION TABLE

INPUTS	FROM DRAWING	DESCRIPTION
2	104-2LD-1	Check 2nd address
3		Check 3rd address
4		Check 4th address
EX		Execute
-YA		
ZJ	104-2LC-2	
ASC	104-2LD-5	Add or subtract or compare
C-		Not compare
D		Divide with round off
d		Exact Divide
Dd		Divide
MDmd		Multiply or divide
Mm		Multiply
-Mm		Not multiply
SC		Subtractor Compare
-SC		Neither Subtract nor compare
CP1	104-3LC-1	
CP4		
CP5		
CP6		
LM		Left minuend
LS		Left Subtrahend
LSW		Left Shifted Word in
DP		Digit pulses
P1A		
P1B		
P2A		
RF		Reset Flip Flops
L±	104-3LC-3	Left sum or difference
a	104-4LD-1	Non-precess
b		Precess
CN		Computing interlock
LB	Memory Bank SW. No Drwg Available (See 104-3LC-1)	

OUTPUTS	TO DRAWING	DESCRIPTION
LA	104-3LC-1	Left adder readout
LAT		Left adder tank
LQ		Left quotient readout
LQT		Left quotient tank
LRS		Left remainder sign
LSC		Left sign circuit
LST		Left shuttle tank
LZE		Left shifted word out
LAG	104-3LC-3	Left add gate
LMA		Left minuend arithmetic
LSA		Left subtrahend arithmetic
LSG		Left subtract gate
LDCE	Memory bank switch - See LB above	Left division capacity exceeded
LASCE		Left add-subtract capacity exceeded
LA1	Left memory unit no drwg available	to left short tank A
LB1		" " " " B
LC1		" " " " C

INPUTS	FROM DRAWING	DESCRIPTION
LA0	Left memory unit no drwg available	from left short tank A
LB0		" " " " B
LC0		" " " " C



Note:- This drawing is a partial reproduction of 104-3LD-2 with equipment used only for multiplication and division omitted

FIGURE 4

PRELIMINARY

APR 8 1948

MOORE SCHOOL OF ELECTRICAL ENGINEERING
UNIVERSITY OF PENNSYLVANIA

EDVAC ADD-SUBTRACT BLOCK DIAGRAM

SCALE

DRAW

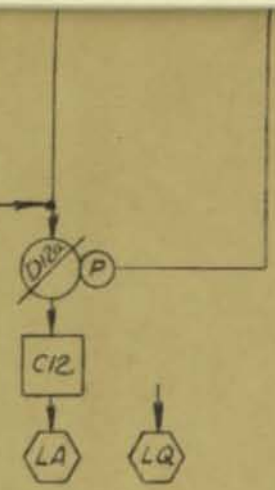


FIGURE 4

PRELIMINARY

APR 8 1948

MOORE SCHOOL OF ELECTRICAL ENGINEERING
UNIVERSITY OF PENNSYLVANIA

EDVAC ADD - SUBTRACT
BLOCK DIAGRAM

SCALE			
DRAWN BY: J.E. GRAVEL 1-29-48	CHECKED BY	APPROVED BY	104-3LD-4

APR 9 1948

RESTRICTED

PRELIMINARY

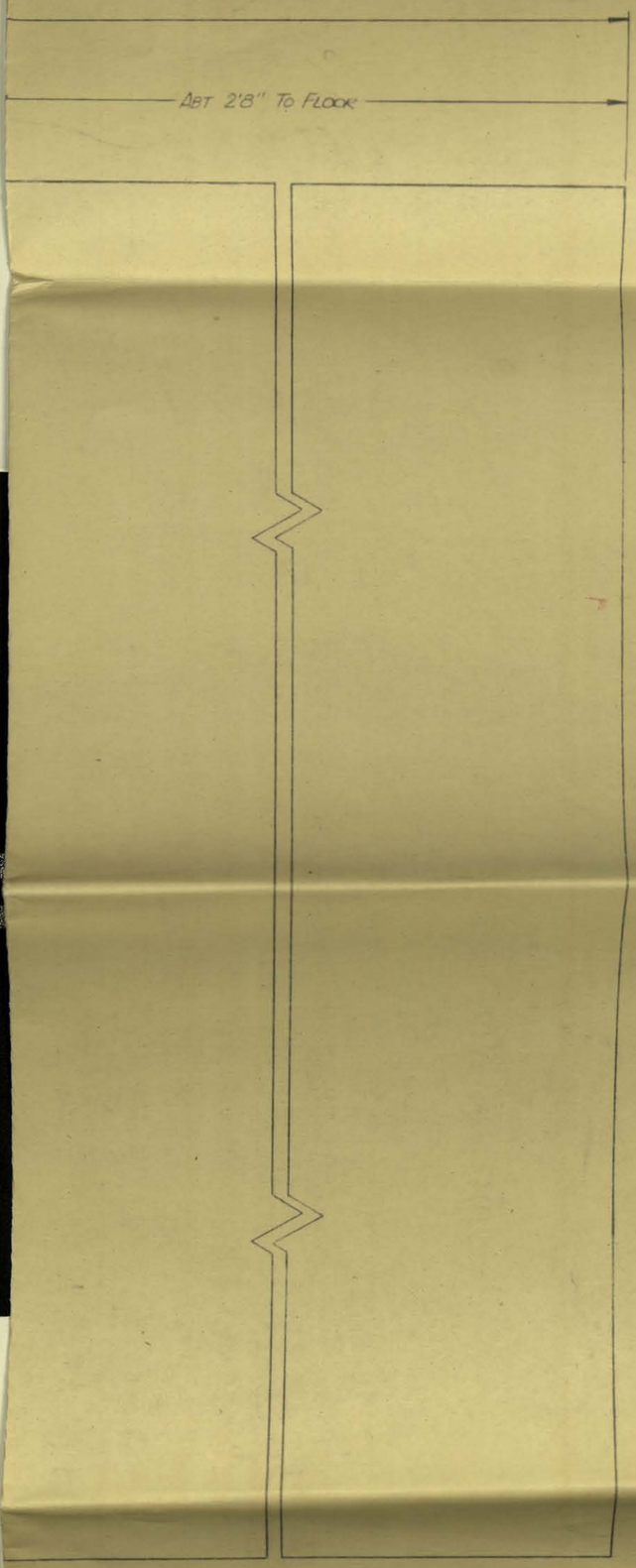
FIGURE 3

MOORE SCHOOL OF ELECTRICAL ENGINEERING
UNIVERSITY OF PENNSYLVANIA

EDVAC CONTROL
PANEL ASSEMBLY.

SCALE 6" = 1'0"

DRAWN BY	CHECKED BY	APPROVED BY	
J.E.GRAVEL 10-15-47			104-7MD-1



APR 9 1948

RESTRICTED
PRELIMINARY

FIGURE 3

MOORE SCHOOL OF ELECTRICAL ENGINEERING
UNIVERSITY OF PENNSYLVANIA

EDVAC CONTROL
PANEL ASSEMBLY.

SCALE 6" = 1'0"

DRAWN BY	CHECKED BY	APPROVED BY	
J.E.GRAVEL 10-15-47			104-7MD-1

VERTICAL

HORIZONTAL

OFF 384 48 0

SWEEP

STROBE

INTENSITY

FOCUS

INDICATOR SELECTORS

MODE OF OPERATION

MEMORY BANKS

OPAL AMBER GREEN BLUE RED OPAL RED

START STOP D.C. OFF CLEAR INITIATE HALT READ OUT

POWER

OPERATION

OCTONARY (OCTAL) DIGIT

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

ADDRESS 1 ADDRESS 2 ADDRESS 3 ADDRESS 4 TYPE

AUXILIARY INPUT

AMBER AMBER

EXCESS MAGNITUDE OPTION

A or S D or d

ADDRESS A

ADDRESS B

ADDRESS 1 TYPE ADDRESS 2

SPECIAL ORDER

ADDRESS 3 ADDRESS 4

ABT. 5'2" TO FLOOR

CYCLING UNIT