MA/17/1024

DEPARTMENT OF SCIENTIFIC AND INDUSTRIAL RESEARCH

NATIONAL PHYSICAL LABORATORY

Progress Report

on the

Automatic Computing Engine

Mathematics Division

April, 1948

CONFIDENTIAL. MA/17/1024.

DEPARTMENT OF SCIENTIFIC & INDUSTRIAL RESEARCH.

NATIONAL PHYSICAL LABORATORY.

PROGRESS REPORT ON THE AUTOMATIC COMPUTING ENGINE.

MATHEMATICS DIVISION.
April, 1948.

PROGRESS REPORT ON THE AUTOMATIC COMPUTING ENGINE

- By -

J. H. WILKINSON.

- -

SUMMARY.

This report gives a general introduction to the work carried out by Dr. Turing and his collaborators during the last two years at the National Physical Laboratory.

The report covers the general principles of both the design of the machine and the method of programming adopted for it. It includes a number of schematic diagrams of the circuits needed for the logical control and the arithmetic units. The circuits are given in some detail for a small pilot model containing some dozen delay lines and using a very simple logical control, and an indication of the extensions necessary for the full scale machine of some two hundred long delay lines is sketched. The programming of a number of problems is given, beginning with one or two simple examples which are complete in themselves, and proceeding to the development of routines which are of general application as components of large scale computing problems. As an example of a problem which requires the use of a number of standard routines, solution of a set of simultaneous linear algebraic equations by two different methods is given.

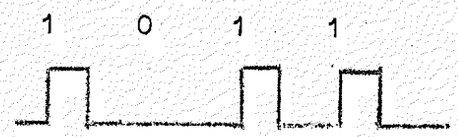
1. Introduction.

The A.C.E. belongs to the class of calculating machines usually known as "digital" machines. This class of machine consists of those which operate upon numbers directly in their digital form, as distinct from those which operate on physical quantities of which the numbers are the measures; the latter machines are usually called "analogue" machines. The fundamental advantage which digital machines possess over analogue machines, is their extreme flexibility. Typical digital machines are the ordinary desk calculating machines such as the Brunsviga, Friden and Marchant, and any of these may, in theory, be used for any problem which can be reduced to numerical computation. The limiting factor, in practice, is the time taken to perform the computation. This limiting factor makes itself felt, even in problems which, at first sight, may appear to be very simple. Consider, for example, the problem of solving a set of linear algebraic simultaneous equations by direct methods of computation. A typical method of solution is by the successive elimination of the variables, often known as pivotal condensation. For a set of equations of order n, the number of multiplications required for the solution is of the order $n^3/3$; for a set of order 32 this already amounts to several weeks work, while a set of order 100 would probably take more than a year. It is the attempt to remove this restrictive time factor which has lead to the consideration of electronic computing machines.

In almost all calculating machines which have been constructed up to the present time, numbers have been represented in the conventional decimal scale. In the A.C.E., however, we intend to represent all numbers in the binary system. The explanation of this break with tradition is that it results in an enormous simplification in the equipment necessary for carrying out arithmetic operations. Every number may be represented in the binary system by a sequence of digits each of which is either a zero or a one, and this provides us with a particularly simple method of

representing/

representing a number electrically. If we consider a sequence of electrical pulses at a fixed time interval apart, then to represent any number we may take a pulse pattern in which, corresponding to each one in the binary representation of the number, there is a pulse, and, corresponding to each zero, there is the absence of a pulse. Thus the number eleven, which is equivalent to 1011 in the binary system may be represented by the pulse pattern below



The number of digits necessary to represent a number in the binary scale is, of course, more than that necessary to represent that number in the decimal scale by a factor $\frac{\log_2 10}{}$; thus to represent all ten digit decimal numbers a minimum of thirty-four binary digits is required. The simplicity of the arithmetic operations in the binary scale will be readily appreciated; in the process of multiplication for example, we have only to consider the two simple cases of multiplication by unity and by zero. In the A.C.E. the time between successive pulses will be 1 microsecond so that it will take only 34 microsecs. for a number of ten decimal digits to pass any given point on a conducting line. The fact that the successive digits of a number appear as a temporal sequence of pulses, characterises the machine as a "serial" operating machine. This is in contrast to a machine in which all the digits of a number are available simultaneously; such machines are known as "parallel" operation machines.

The employment of electronic equipment reduces the times needed for the fundamental operations of arithmetic by an enormous factor. Addition of two 32 digits binary numbers, for example, can be performed in 32 microsecs. on a serial machine with a pulse interval of 1 microsec, and the time of multiplication of two such numbers can be reduced to 64 microsecs. This enormous increase in speed necessitates an entirely different method of control

in/

in an electronic computer. It would clearly be absurd, for example, to use the machine to perform a multiplication in the same way as is done on a desk calculating machine, since the time of reading the multiplier and multiplicand, communicating these values to the machine and writing down the answer would completely swamp the time taken for the actual multiplication.

If we return to the problem of the solution of a set of equations of order n, it will be seen that although approximately $n^3/3$ intermediate numbers are obtained in the course of the computation, only the 'n' numbers, corresponding to the solution of the set, are of interest to the computer. In a high speed computer there must be some means of storing such intermediate numbers obtained during a computation, in a form in which they are available at a speed comparable with that at which the machine performs the arithmetic operations. The possession of a high speed "memory" of considerable capacity is thus seen to be a basic requirement of any efficient electronic computer. In the A.C.E. the high speed memory will consist of a number of elements known as acoustic delay lines with a total storage capacity equivalent to approximately 6,000 twelve digit decimal numbers.

There is a further requirement before the machine can take full advantage of its speed. It must be possible to supply the instructions which the machine is to carry out at the same high speed. This may be achieved by representing these instructions in a coded numerical form and storing them in the memory in the same way as the numbers themselves; the machine must therefore possess a "logical control" which is capable of interpreting the coded instructions and causing them to be obeyed.

There is one final requirement which the electronic computer must meet. There must be some means of communicating with the outside world, both for the purpose of receiving initial data and instructions, and for communicating results. In its earliest version at least, the A.C.E. will use Hollerith punched card equipment to perform these input and output

functions/

functions. It will clearly be essential that it should be possible for the machine to accept data in decimal form and to provide solutions in the same form if needed, since the conversion from one system to the other is quite a lengthy process. Hence two most important instruction tables which the machine must be capable of interpreting are tables for decimal-binary and binary decimal conversion.

It might be imagined that a problem such as the solution of a set of linear equation of order 32 which required 10,000 multiplications would need so many instructions that the whole of the storage capacity would be completely filled with instructions. Fortunately this is not true because most computational problems can be reduced to a small number of cycles of operations each of which is repeated a great number of times. The machine must, however, have some means of discriminating whether one of these cycles has been repeated the requisite number of times, and as a result of this discrimination move to the next cycle of operations or repeat the current cycle.

The means by which the A.C.E. fulfils all the above requirements are the subject of the following paragraphs.

The account may be divided into two main sections. The first section deals with the design of the A.C.E. and the development of the appropriate code. It is not considered expedient at this stage to decide upon the details of the A.C.E. since many points will only be definitely determined in the light of engineering and coding experience gained from constructing and computing with a small pilot model. For this reason the details of the design and the code of such a pilot model are given first, and then a sketch only of the extensions envisaged in the full scale machine is given. This sketch is merely intended to be sufficient to form a background for the description of the type of code which the full scale A.C.E. will employ. The second section deals with the problem of programming, that is, of

preparing problems for computation on the machine. Although the detailed coding of problems has been given in terms of the code which has been adopted for the A.C.E., many of the considerations of this section are independent of any specific code. Before a large-scale problem can be coded for solution on an automatic computer a considerable amount of preliminary work is necessary even if the problem is already expressed in mathematical form, i.e. if the problem of translating a physical problem into a mathematical problem is excluded. The computational process appropriate to the mathematical problem has first to be decided upon and if this involves approximations, such as replacing derivatives by finite difference expressions, the limits of the error thereby introduced must be investigated. When a process which is satisfactory in this respect has been determined, an investigation into its stability with respect to the "rounding off" errors, which are inherent in digital computations, must be undertaken. Further, it is necessary that all the numbers derived during the course of the computation should be provided with scale factors which ensure that they do not grow out of the range permitted by the storage space allotted to them. This problem is somewhat more serious in automatic computation than in ordinary desk computation, because in the latter appropriate scale factors may be introduced by the operator as they become necessary and it is not important to anticipate when the necessity will arise. When all these points have been cleared up it is convenient to map out a general scheme of the method of computation and it is only after the completion of this scheme that the detailed coding begins. If the coding of each problem were attempted 'ab initio', the time taken to prepare a problem for computation might seriously reduce the effective speed of the machine. For this reason a considerable part of the section on programming is devoted to the problem of constructing tables of some generality for the standard routines which occur as elements of large scale problems. Generality is achieved by providing these standard routines with a number of parameters each

of/

of which will be assigned a specific value, appropriate to the problem in hand, when the routine is used. In this way it is possible to prepare the greater part of any large scale computing problem for the machine by building it up from prefabricated units. Since these units are to be of very wide application, far more time and thought may be given to their preparation than would otherwise be justifiable. For this reason we have made a general principle of carrying out, concurrently with the mathematical investigation and the coding of a routine, a practical investigation on desk calculating machines. This is not because we regard the ordinary weapons of mathematical analysis as in any way inadequate for the task, but because the practical method often brings to light special points which are easily overlooked in a strictly theoretical approach. It should be emphasised that none of the instruction tables given in the second section are regarded in any respect as representing finished products. It is very unlikely that any of them will remain unaltered in the light of experience gained in using the machine. In spite of this we regard the preparation of these tables as being of real value since we believe that the successful use of electronic computing equipment will depend more upon the development of an efficient method of organising routines and their integration into large scale computing problems than upon any other single factor.

2. Symbols used in schematic circuits.

A number of special symbols are used in our schematic circuit diagrams, and for convenience these are collected and described below.

In the schematic circuits we consider the transmitted signals to be sequences of digits, each of which is a zero or a one. The actual electrical signal corresponding to a 'zero' or a 'one' will be different in different parts of a circuit. For instance, in a number of parts of the circuit, an unbroken sequence of ones may in fact be a D.C. voltage and an unbroken sequence of zeros, a different D.C. voltage. In the case where a one is a pulse, the centre of the pulse may occur at different times in different parts

of/

of the circuit and the width and shape of the pulse may also vary. A full consideration of such points would be out of place in a report of this nature, however, and it is sufficient to assume in the schematic circuits that we are dealing with signals consisting of zeros and ones which are perfectly synchronised. It is convenient for this purpose to think of time, not as being continuous, but as coming in a sequence of ticks; corresponding to each tick the signal at each point of a circuit is defined as 0 or 1. With this understanding we proceed to the definition of our schematic symbols. It should be recognised that these symbols do not necessarily possess a unique equivalent in the actual electronic circuitry. A particular symbol may correspond to a number of entirely different electronic circuits; all that is implied is that the logical effect of the true circuit is the same as that in the schematic circuit.

Direction of propogation.

An arrow on a line indicates the direction in which signals are propagated. Thus in fig.1 the signal is travelling from A to B.

Simple delays.

If the arrow is converted into a triangle as in fig.2 there is a delay of one unit in passing from A to B, so that if a_n is the signal at A at time n, and b_n that at B, then $b_n = a_{n-1}$.

Long delays.

A delay of a number of units is shown as an elongated rectangle with the delay within it. In fig.3, for example, there is a delay of four units in passing from A to B.

Change of polanity (negation).

A short stroke across a transmission path indicates a change of polanity i.e. zeros and ones are interchanged. Hence in fig.4 $b_n = 1 - a_n$.

Junction of transmission paths.

A transmission path may be allowed to bifurcate, in which case the signal is transmitted simultaneously on both forks. E.g. in Fig.5 $a_n = b_n = c_n$.

Transmission/

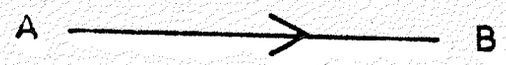


FIG. 1.

FIG. 2.

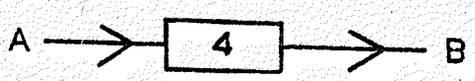


FIG 3

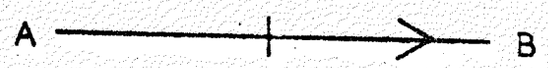


FIG. 4.

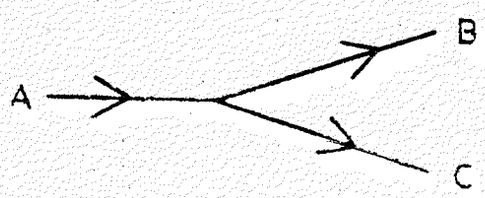


FIG. 5.

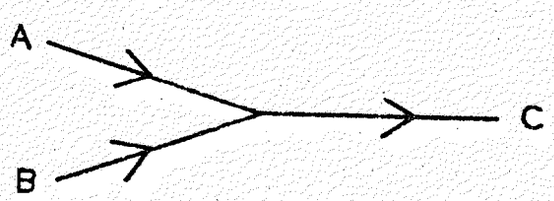


FIG. 6.

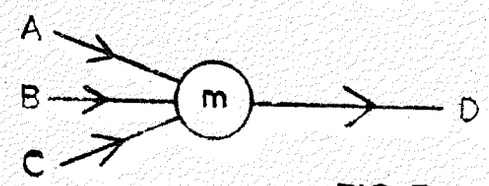


FIG. 7.

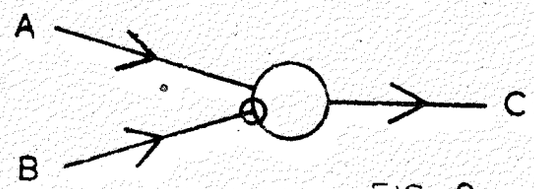


FIG. 8.

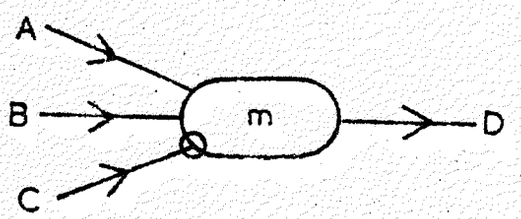


FIG. 9.

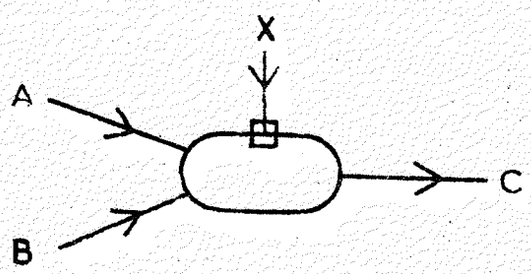


FIG. 9A.

FIG. 10.

FIG. 11.

FIG. 12.

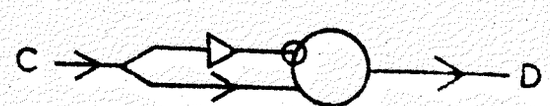


FIG. 13.

FIG. 14.

Symbols used in

Schematic Diagrams

Transmission paths are sometimes shown converging as in Fig.6. This is not normally done unless the circuit is such that it is not possible for a one to appear simultaneously on more than one of the tributary converging lines. The signal in the main stream is then defined to be a one if the signal in any tributary is a one, and to be a zero if the signals in all tributaries are zero.

Limiters.

A limiter is shown as a circle containing a number m, known as the threshhold of the limiter, with any number of inputs and outputs (Fig.7). The limiter gives an output of one on all output lines if the input is 'm' or more ones, and an output of zero otherwise. In a limiter of threshhold 1 the 1 is omitted.

Inhibitive connections.

A limiter may have a second type of input known as an inhibition, e.g. the input B in Fig.8. The output of the limiter is to be a zero if the input from B is a one, regardless of the other inputs. An input of zero from B is disregarded.

Triggers.

A trigger is unlike a limiter in that its output at any time depends on its previous history as well as its current inputs. A trigger of threshhold 'm' is shown as an ellipse with the figure 'm' within it. (Fig.9). It has any number of inputs and outputs and at least one inhibition. The trigger is "tripped" when it receives m or more simultaneous ones, and from then onwards its output is a one until such time as it receives a one on an inhibitive connection. It is then "reset" i.e. returns to a condition in which its output is zero. The trigger is said to be 'on' when it is emitting ones, and 'off' when it is emitting zeros. A trigger may have another type of input known as a 'changeover'. Such an input is shown as X in Fig.9A. If the trigger receives a one on a 'changeover' input its state is changed i.e. if it was emitting ones it begins to emit zeros and vice-versa.

Differentiators.

A circle containing an E (Fig.10) denotes an element which emits a one after the

termination/

termination of a sequence of ones on its input. Thus in Fig.10 $d_n = \text{Max}\ (0, a_{n-1} - a_n)$. Fig. 12 shows an equivalent circuit. A circle containing a B (Fig.11) denotes an element which emits a one at the beginning of a sequence of ones at its input. Fig.13 shows an equivalent circuit. These elements will normally be realised with "differentiator" circuits. The 'E' and 'B' are intended to suggest the ideas of "end" and "beginning".

Manual Controls.

A manual press button is shown as a circle containing an 'M' with an output, Fig.14. It is to be understood that such an element emits a sequence of ones starting at some unpredictable time, shortly after the pressing of the button. This is actually a slightly troublesome condition to attain, but for the purposes of the schematic diagrams it is sufficient to assume that it has been reached.

Major and minor cycles.

The individual unit of information is, of course, the digit, which may take the values 0 and 1, but it is convenient to consider also units consisting of aggregates of digits. In the pilot model of the A.C.E. a group of 32 digits has been taken as one of our fundamental units of information. An aggregate of 32 binary digits is capable of representing numbers of rather less than ten decimal digits ($2^{32} = 4.295 \times 10^9$). This suggests that we should measure time in groups of 32 'ticks'. These groups are known as 'minor cycles'. It may be helpful to think of the minor cycles as being equivalent to minutes, with ticks at each second. The information contained in the signal appearing at any point during a minor cycle is called a 'word'. Each word then, consists of 32 digits.

This division of time into minor cycles is provided by a circuit known as a ring counter. It is, effectively an electronic commutator and its schematic circuit is shown in Fig.15. It consists of a group of 32 elements in a ring, each of which is separated from the two adjacent elements by a unit delay; the 32 elements are known as P1, P2, - - P32. At the beginning of operations the CLEAR push button is operated; this removes any stray pulses in the ring

counter/

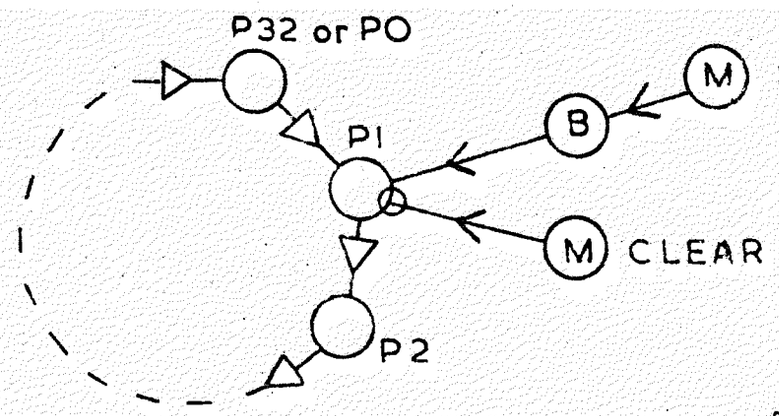


FIG.15.
RING COUNTER

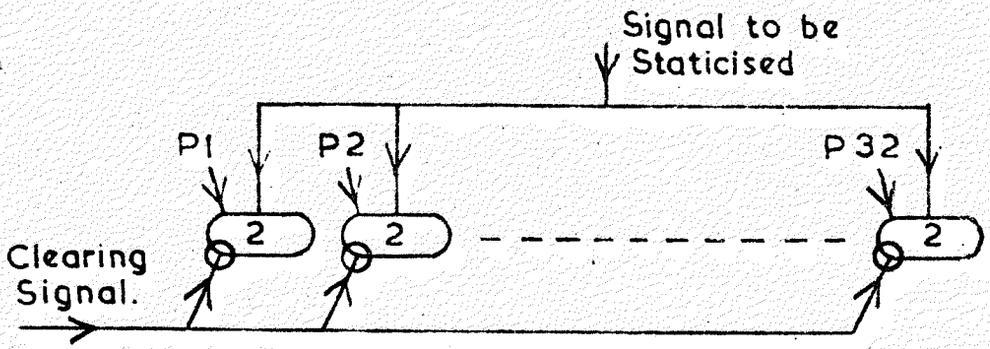


FIG.16.
STATICISER

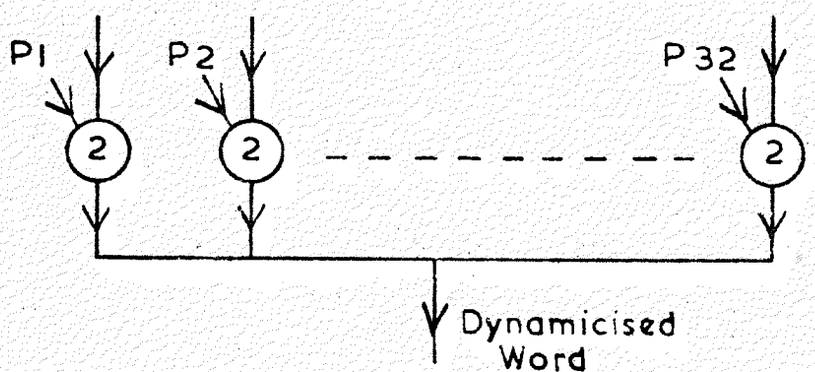


FIG.17.
DYNAMICISER

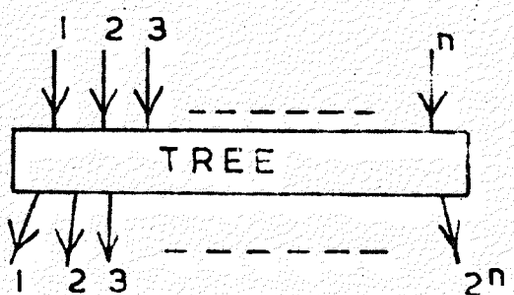


FIG.18.
TREE

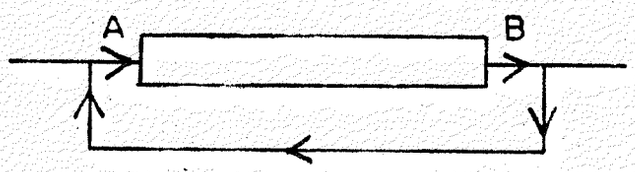


FIG.19.
DELAY LINE

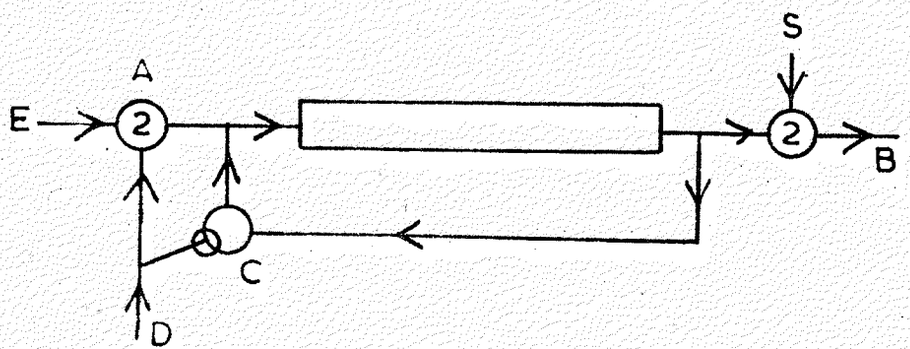


FIG.20.
DELAY LINE
WITH INPUT
AND OUTPUT

counter; the START push button is then operated and this puts a single pulse into P1. At any subsequent tick one and only one of the Pn's will be emitting a pulse.

The elements of the ring counter are frequently used in the control circuits to supply ones at definite points in a minor cycle. The element P1 e.g. supplies a one at the beginning of each minor cycle and thereafter 31 zeros. The pulses supplied by the element Pn are usually referred to as "Pn pulses", and the notation, Pn, is also used to denote the sequence of digits supplied by the element Pn in any minor cycle. This use of "Pn" may appear to be somewhat ambiguous, but in practice the meaning is always abundantly clear from the context.

A period of 1024 ticks or 32 minor cycles is known as a major cycle. There is no need to have anything corresponding to the ring counter for telling the time within major cycles. This is achieved, effectively, by the positioning of the coded instructions, as will appear later.

Staticisers and Dynamicisers.

In a number of positions in the circuits it is necessary to convert information from the dynamic form, as a temporal sequence of digits or pulses, into a static form consisting of D.C. voltages on a number of lines, corresponding to the number of pulses. In Fig.16 the circuit used for "staticising" a word of 32 digits is given. It consists of a group of 32 triggers of threshold two, each of which receives the signal to be staticised and one of the Pn pulses. The n^{th} digit of the signal to be staticised trips the n^{th} trigger if it is a one, and fails to trip it if it is a zero. There is also a lead for resetting all the trigger circuits; this must be pulsed before a second word can be "set up" on the staticiser.

Conversely, we may wish to convert a number of static signals into the dynamic form. The static signals may come from triggers or be obtained from a set of switches. The circuit for performing this operation is shown in FIG.17. It consists of a set of 32 limiters of threshold two, each of which

is/

is supplied with a Pn and one of the static signals. The outputs from the limiters are combined together to produce the static signal, in dynamic form, once per minor cycle.

Trees.

If we consider a set of n triggers then there are 2^n possible states in which they can be, since each one may be either "on" (emitting ones) or "off" (emitting zeros). It is frequently necessary in our schematic circuits to use a set of n triggers to provide an unbroken stream of ones on one of 2^n different lines. There are a number of different electronic circuits for performing this selection, of which perhaps the most common is a resistance network. Any circuit which performs this operation is referred to, in this report, as a "tree of order n", see Fig.18.

Whenever we have n different sources of D.C. voltage each of which may take one of two different voltages we may use these n sources to select one of 2^n different lines, i.e. as the basis of a tree of order n. Triggers are the most common examples of such sources of D.C. voltages.

Delay lines.

The account given in this report is based on the assumption that the storage medium for the numbers and coded instructions will be acoustic delay lines, though little modification would be needed for any other truly serial method of storage. An acoustic delay line consists essentially of a straight cylindrical tube filled with mercury and with a piezo-electric crystal at each end. It is shown schematically in Fig.19. If a pulse is fed into the delay line at A, the crystal at A sends out a sonic wave into the mercury which travels down the delay line at the velocity of sound in mercury; at point B this disturbance is reconverted into a pulse by the second crystal. If this pulse is now amplified and reshaped it may be fed back into A and in this way the pulse may be preserved indefinitely. Two standard lengths of delay lines are used in the A.C.E. The first type, usually called a long tank is of such a length that a sound wave takes 1024 (i.e. one major cycle) to travel down the

tube/

tube. Since our pulses are a microsecond apart, a long tank may be used to store 1024 digits or 32 words. The configuration of pulses in a long tank recurs once per major cycle. The other type of delay line, usually called a short tank or temporary storage (T.S.), is capable of storing 32 digits only i.e. exactly one word. The configuration in a short tank recurs once per minor cycle. In addition to the circuitry needed for the recirculation, each tank is equipped with an input and an output device. This is shown in Fig.20. If an unbroken stream of ones is supplied along the line marked S, a copy of the contents of the tank is obtained at B, the circulation of the content of the tank being undisturbed. If, on the other hand, an unbroken stream of ones is supplied along the line marked D and a new sequence of digits is fed in at E, this new sequence passes into the tank, whilst the former contents are obliterated at C.

3. Arithmetical and logical operations.

Conventions used in the representation of numbers.

It has already been stated that numbers are stored in the binary system. As far as the units which perform the arithmetic operations are concerned these numbers are treated as though they were integers. The true position of the binary point will vary from problem to problem to suit convenience and its correct manipulation in any given problem will be taken care of by the programming. It is possible that some problems will arise which will necessitate the use of a floating binary point. This means that each number will be represented by an ordered pair of numbers m and n, the first of which is an integer (positive or negative) and the second satisfies $-1 \leqslant n < 1$. The number corresponding to the ordered pair (m,n) is $2^m n$. In such problems the basic arithmetic operations will also be programmed; in general it may be said that a floating binary point will only be resorted to, when it proves impossible to estimate the necessary scale factors without undue effort. In most problems its use may be eliminated by adjusting the binary point from time to time during the computation, as this becomes necessary.

If for the moment we agree to consider our numbers as integers and make one word available for each number, then we can represent all integers 'n' such

that/

that $-2^{31} \leq n < 2^{31}$ by the following convention. The integer 'n' is to be represented, modulo 2^{32}, by a number, n', such that $2^{32} > n' \geq 0$. In this convention all negative numbers have a 1 in the extreme left hand position and all non-negative numbers, a 0. For this reason -2^{31} is included and $+2^{31}$ is excluded. Each number, then, may be said to consist of 31 digits and a sign digit.

If we allow our numbers to be two words in length, then we may represent all integers n such that $-2^{63} \quad n \quad 2^{63}$. A number is now represented by 63 digits and a sign digit. Similarly, if we allow m words for our numbers, each number is represented by 32 m-1 digits and one sign digit in the extreme left hand position. A small number, regarded as a number represented in the m word convention, will begin with a long string of zeros if positive, and a long string of ones if negative.

Arithmetic units.

The Arithmetic units of the A.C.E. are more closely integrated with the logical control than is the case in most other electronic computing machines which have been designed, but it is convenient to consider the basic circuits for performing these operations as separate units at this stage.

Addition.

The schematic circuit for addition is shown in Fig.21. To understand how it performs addition let us suppose two sequences of digits a_n, b_n corresponding to numbers A and B are fed in to the adder on the lines on the left hand side. These two sequences must of course be in phase, i.e. the digits in the least significant position must arrive simultaneously. For each digit of the sum (A + B) except the first, we have to consider the problem of adding the two current digits of A and B and the carry over from the previous digit, and forming from them the current digit of (A + B) and the carry over to the next digit. The current digit of (A + B) is fed out on the line marked (A + B) and the carry digit is borne by the line marked C. The carry digit is delayed one unit before it arrives at D. At any stage of the addition, then, the current

digits/

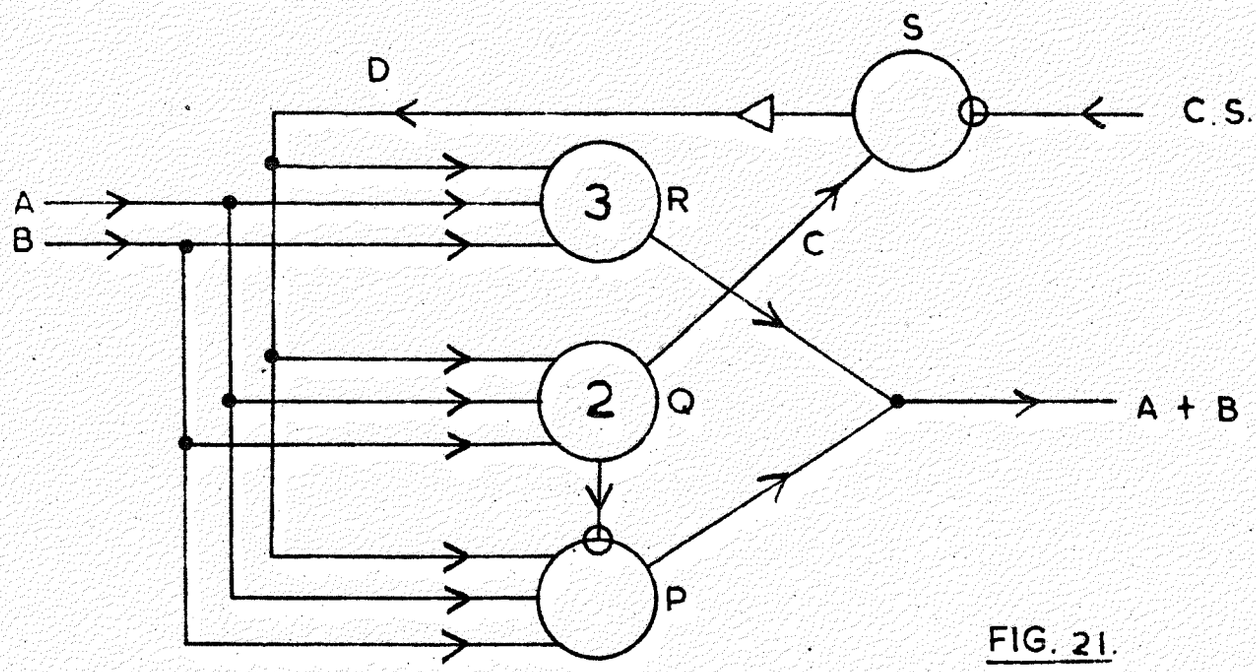


FIG. 21.

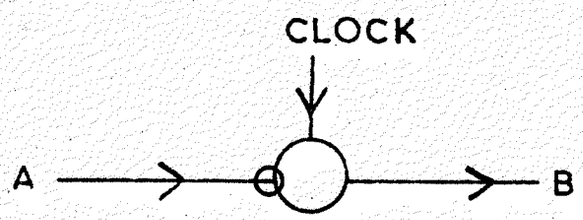


FIG. 22.

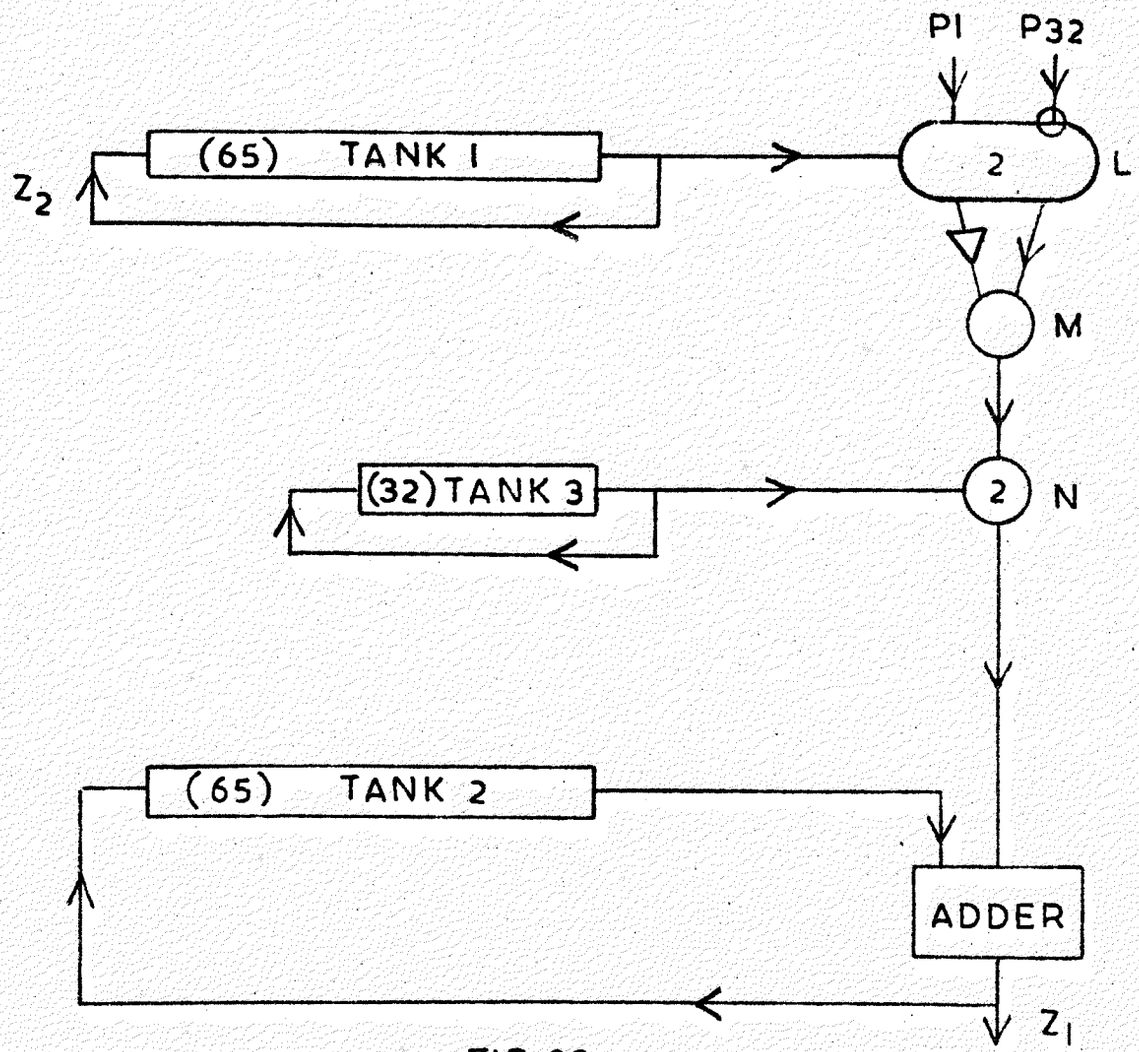


FIG. 23.

digits of A and B and the carry over from the previous stage are each fed to all three of the elements P, Q and R. It is now sufficient to consider four simple cases.

(i). P, Q and R are each fed with three zeros, In this case a zero is emitted for the current digit of (A + B) and for the carry digit.

(ii). P, Q and R are fed with two zeros and a one. In this case Q and R are fed with two zeros and a one. In this case Q and R emit zeoos and P emits a 1 on the (A + B) line. The carry digit is therefore zero and the current digit of (A + B) is a 'one'.

(iii). P, Q and R are fed with one zero and two ones. In this case R emits a zero but P and Q both have sufficient inputs to emit a one. The one emitted by Q however inhibits P and the net effect is that a zero goes out from P to give a zero for the current digit of the sum; and a one goes out on the carry line from Q.

(iv). P, Q and R are fed with three ones. In this case the threshhold of all three elements is reached. Again P is inhibited by the output of Q and therefore puts out a zero but R emits a one as the current digit of the sum, and Q gives a carry digit of one on the line C.

It will be seen that each of the above cases gives the result required by the rules of binary addition. The first digit does not require any special consideration. Clearly there is no carry from the "previous stage" and hence there is a zero on the line marked D.

So far we have tacitly assumed that our numbers are indefinitely long and have omitted all reference to the element marked S with the inhibiting connection marked C.S. Suppose now we restrict our numbers to be one word in length. If we are to use our convention for numbers with a sign, then, even in the simplest case of the addition of two positive numbers, we cannot expect the adder to give the correct answer unless it lies in the permitted range. E.g. consider the

addition/

addition of $2^{30} + 1$ to 2^{30} shown below. The answer given by the adder interpreted by the

```
0100 .........  001 +
0100 .........  000
-------------------
10............  001
-------------------
```

same convention is negative since it has a 1 in the extreme left hand position; it is in fact $-2^{31} + 1$ according to this convention. Hence if we are to use the convention all the time the answer to this addition is only given correctly, modulo 2^{32}. Provided the true sum is such as to lie in the permitted range, however, it will be exact. Consider now the addition of 6 and -5. The number 6 will be represented by 6 but the number -5 will be represented by $2^{32} - 5$. The addition as performed by the adder is shown below.

```
0000 .........  00110  representing   6
1111 .........  11011  representing  -5
-----------------------------------------
1 0000 .......  00001
```

The adder will produce first a 1 and then 31 zeros but it will also produce a carry one from the thirty second digit. This carry over at the thirty second digit is suppressed by feeding a P32 on the lead marked C.S. (carry suppression) at the end of each minor cycle. By this means the adder can perform correctly the addition of any two numbers, either positive or negative, which lie in the range permitted by the convention, provided the true sum also lies in this range. Otherwise the sum given by the adder, interpreted in our convention, will only be correct modulo 2^{32}. If we consider the addition of double length numbers a similar problem arises, but now we must suppress the carry over at the P32 position corresponding to the sign of the number and must not suppress it at the P32 position occuring in the middle of the number. This is achieved by means of a source which gives a P32 at the end of alternate minor cycles. Full

consideration/

consideration of this point and the addition of multiple length numbers is given later.

Subtraction.

The operation A-B is performed effectively as $A + (2^{32} - B - 1) + 1$. The number $2^{32} - B - 1$ may be obtained from B by interchanging zeros and ones i.e. by a polarity change. If we assume we have a standard clock which is emitting an unbroken stream of ones, then this polarity change may be achieved by the circuit in Fig. 22 which operates as follows. If the digit in line A is a one, then the clock is prevented from passing a one to B and hence we obtain a zero at B; if the digit in line A is a zero then line B receives a one from the clock. A-B, then, is formed by sending A and $2^{32} - B - 1$ to the adder and feeding a P1 pulse in the carry line to account for the extra 1. As in the formation of A + B, we obtain the correct answer to A-B, within the convention, provided the true answer is in the permitted range. A and B may be either positive or negative. For double length numbers, the P1 has to be added in the first P1 position of the number but obviously not in the second P1 position. This is a similar problem to that of the suppression of the carry digit in double length addition.

Multiplication.

It is convenient, for the sake of simplicity, to ignore our sign convention for the time being and to consider the multiplication of two positive numbers A and B each of which satisfies the inequalities

$$0 \leqslant A < 2^{32}$$

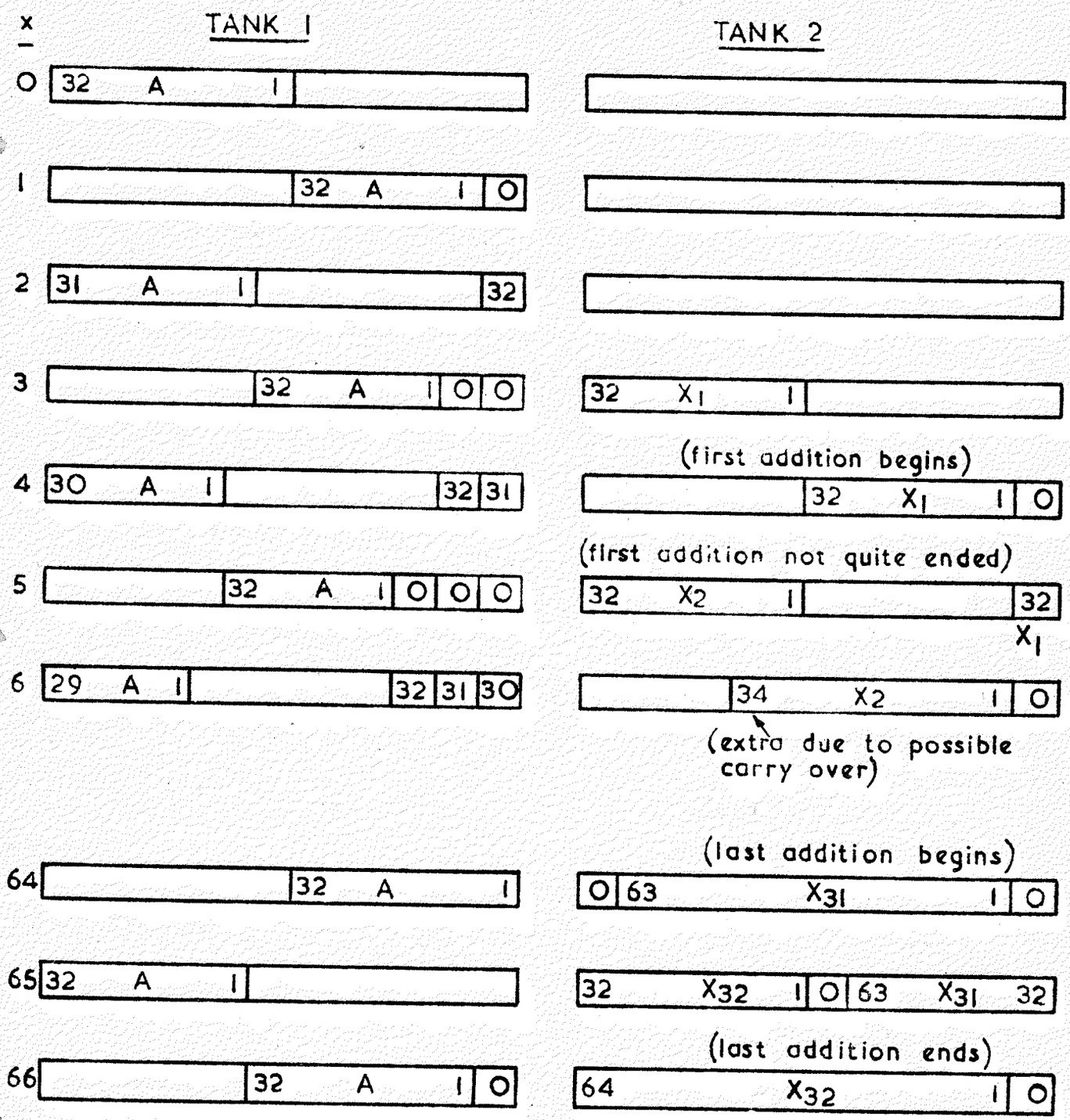
We define a_n and b_n to be the n^{th} binary digits of A and B, a_1 and b_1 being the least significant and there being 32 binary digits in each number. If either of the numbers is small there will be a number of zeros in the more significant places of that number. The product X, of the two numbers, may be regarded as a 64 digit number of which any number of the more significant digits may be zeros. If the number A be regarded as the multiplier and B as the

multiplicand/

FIG. 36A.

WORKING OF MULTIPLIER

Showing content of Tanks 1 and 2 at end of minor cycle m+x, for x=0,---,66.



multiplicand then the product may be formed in the following steps

$$X_1 = a_{32} B \qquad (1)$$

$$X_2 = 2X_1 + a_{31} B \qquad (2)$$

$$X_3 = 2X_2 + a_{30} B \qquad (3)$$

- - - - - - -

$$X_n = 2X_{n-1} + a_{33-n} B \quad (n).$$

- - - - - - -

$$X = X_{32} = 2X_{31} + a_1 B \quad (32).$$

The number X_i may be called the i^{th} partial product. To form X_{i+1} from X_i, A and B, we multiply X_i by 2 and add to it B or zero according as a_{32-i} is one or zero. The circuit for performing this operation is shown in Fig. 23. It consists of two tanks of length 65 units and one ordinary short tank. Let us suppose that initially the two 65 digit tanks are empty and the 32 digit tank contains B, and that in minor cycle number 'm' the multiplier A is sent to tank 1 (i.e. the upper 65 digit tank). Nothing will happen until the trigger L is tripped and this can only occur if tank 1 emits a one at time P1. Throughout the multiplication, tank 1 will contain the multiplier and 33 zeros. The first opportunity L has to be tripped is at the P1 time of minor cycle (m+3). which we denote by $(m+3)_1$,; at this time the a_{32} digit is received by L and it will therefore be set up or not according as a_{32} is a 1 or 0. When L is set up, M emits 32 consecutive ones and this enables the contents of tank 3 (i.e. B) to pass through N. During minor cycle (m+3) then, $a_{32}B$ passes into the adder and is added to the contents of tank 2 (i.e. zero). to form X_1. In (m + 4), one of the 33 zeros is omitted from tank 1 and therefore L is not tripped. Since tank 1 is of length 65 its contents are delayed one unit for each circulation and therefore in $(m + 5)_1$, tank 1 emits a_{31}. Hence the adder receives a_{31} B from M. The content of tank 2 will also be delayed one unit and will therefore be transmitting $2X_1$ to the adder. Tank 2 therefore receives $(2X_1 + a_{31} B)$ i.e. X_2 during minor cycle (m + 5) and the 1st tick of (m + 6) if X_2 should be

as/

as long as 33 digits. Similarly in $(m + 6)_1$ the trigger L receives one of the 33 zeros and in $(m + 7)_1$ it receives a_{30}. Hence in $(m + 7)$ and $(m + 8)$ tank 2 receives X_3. The process proceeds with L receiving the successive digits of A and one of the 33 zeros in alternate minor cycles, and tank 2 receiving the successive partial products. It is evident that tank 2 receives X_i from the adder during minor cycles $(m + 2i + 1)$ and $(m + 2i + 2)$ and therefore the product X or X_{32} is available at Z_1 in minor cycles $(m + 65)$ and $(m + 66)$. The multiplier has been described with two tanks of length 65 for clarity; the second tank of length 65 can easily be dispensed with in the following manner. The partial product X_i cannot contain more than $32 + i$ digits, and at the stage at which it is formed we have already made full use of i digits of the multiplier. If the successive digits of the multiplier are obliterated immediately after use, then the partial products may be stored in the same tank as the remaining part of the multiplier and there will, in fact, always remain one zero at least between the partial product and this remainder of the multiplier. Hence tank 2 is removed, the output of tank 1 is fed into the adder and the output of the adder back to the input of tank 1. Tank 1 now has no recirculation except via the adder. The only other alteration (in addition to the simple device for removing the successive digits of the multiplier as they are used) is that the trigger L, must now be supplied with alternate P1's only. This is because in the simplified diagram we made are of the fact that tank 1 contained the multiplier and 33 zeros, and one of these zeros reached L at alternate P1 times. The space occupied by those zeros is now replaced by the partial products and these must not be allowed to trip L.

So far we have considered the multiplication of positive numbers only and have ignored our sign convention. Suppose we use the multiplier as it stands to form the product of plus six and minus five. These numbers will be represented by 6 and $2^{32} - 5$ and hence it will produce the answer $6(2^{32} - 5)$. To produce the correct answer regarded as a signed number of two word length we must add 2^{32} $(2^{32} - 6)$ using a two word adder; this gives $2^{64} - 30$ which is

the/

the correct answer within the convention. For the multiplication of minus six by minus five, the multiplier produces $(2^{32} - 6)$ $(2^{32} - 5)$. If we add to this result $2^{32} \times 6$ and $2^{32} \times 5$ using a two word adder then we will obtain the true answer, 30, since the extra 2^{64} will be the carry which is suppressed. It is easy to see from these examples that, in order to obtain the true answer for the product of two signed numbers in the standard convention for two-word signed numbers, the value given by the multiplier must be corrected according to the following rule. If either of the factors is negative then we must add $2^{32} \times (2^{32} -$ other factor) using a two-word adder.

The basic circuit for multiplication may be extended in a number of ways. Its speed may be doubled, for example, simply by multiplying by two digits of the multiplier for each circulation. This would require the following alterations. The tank of length 65 units must be extended to 66 units, the elements L, M and N must be duplicated and the second trigger corresponding to L supplied with a P2 instead of a P1, and two adders must be used instead of one. Similarly, by using four adders, the speed may be quadrupled and so on. Since adders are quite simple in a serial machine this does not involve very much extra apparatus: The multiplier may also be extended to put in the corrections required for 'signed' multiplication automatically. Finally it could be made to produce a 32 digit answer correctly rounded off, if this should prove advantageous. The form of multiplier we intend to use will be discussed later.

Division.

An automatic dividing unit may be constructed, either using the same tanks as are used for multiplication, in which case one digit of the quotient is obtained every two minor cycles, or using short tanks only, in which case one digit per minor cycle can be obtained: In this report, however, we shall assume that the A.C.E. has no automatic divider; division will always be programmed, using a standard table of instructions.

The logical operations.

In addition to the fundamental arithmetic operations we used a number of

logical/

logical operations; the circuit for performing these operations may be expressed very simply by means of our standard symbols. The logical operations are defined below.

(i) A & B = C. The operation A 'and' B on two sequences of digits, A, B, produces a sequence of digits, C, such that c_n is a one if both a_n and b_n are ones, and is a zero otherwise. (Fig. 24).

(ii) A v B = C. The operation A 'or' B produces a sequence of digits, C, such that c_n is a 'one' if a_n or b_n is a 'one' or if both are ones, and c_n is a zero only if a_n and b_n are both zeros (Fig. 25).

(iii) $\sim$ A = C. The operation, 'not' A, produces a sequence of digits C such that $c_n = 1-a_n$ i.e. zeros and ones are interchanged (Fig. 26).

(iv) A $\not\equiv$ B = C. The operation A 'not equivalent to' B, produces a sequence of digits C such that $c_n = 1$ if $a_n \not\equiv b_n$, $c_n = 0$ if $a_n = b_n$ (Fig. 27).

4. Logical Control of the Pilot Model.

General considerations on the coding of instructions on the A.C.E.

Although the logical control of the full scale A.C.E. is much more comprehensive than that of the pilot model, it may be regarded as a natural extension of the latter. For this reason we describe the logical control of the pilot model in detail and then sketch the extensions made in the full scale A.C.E.

A fundamental feature of the control of the pilot model is that every coded instruction is regarded as a transfer of a sequence of digits from one of a number of positions in the machine known as 'sources', to one of a number of positions known as 'destinations'. We shall consider, first of all then, the problem of transferring numbers from one tank to another.

Suppose we have a number of short tanks and long tanks, each with an input and output device of the type described in Section 2 and each of the outputs of the tanks is connected via a common line called 'highway' to each of the inputs. Such a system consisting of two short tanks and three long tanks is shown in Fig. 28. Each of the tanks is assigned a number. Suppose we wish to transfer a number from tank no. 1 to tank no. 2 both being short tanks. We have only to

supply/

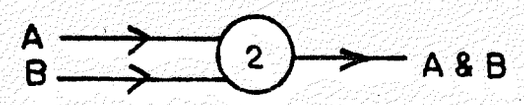


FIG. 24.

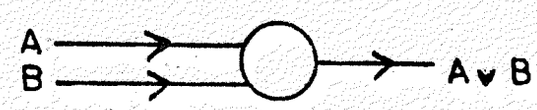


FIG. 25.

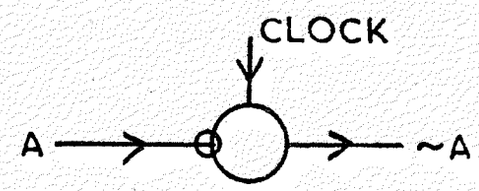


FIG. 26.

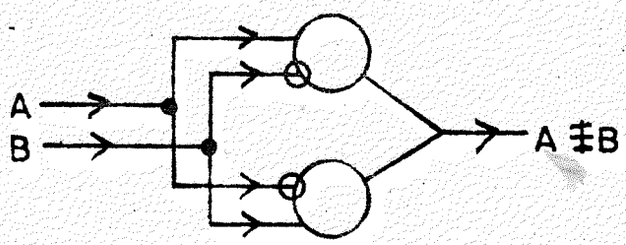


FIG. 27.

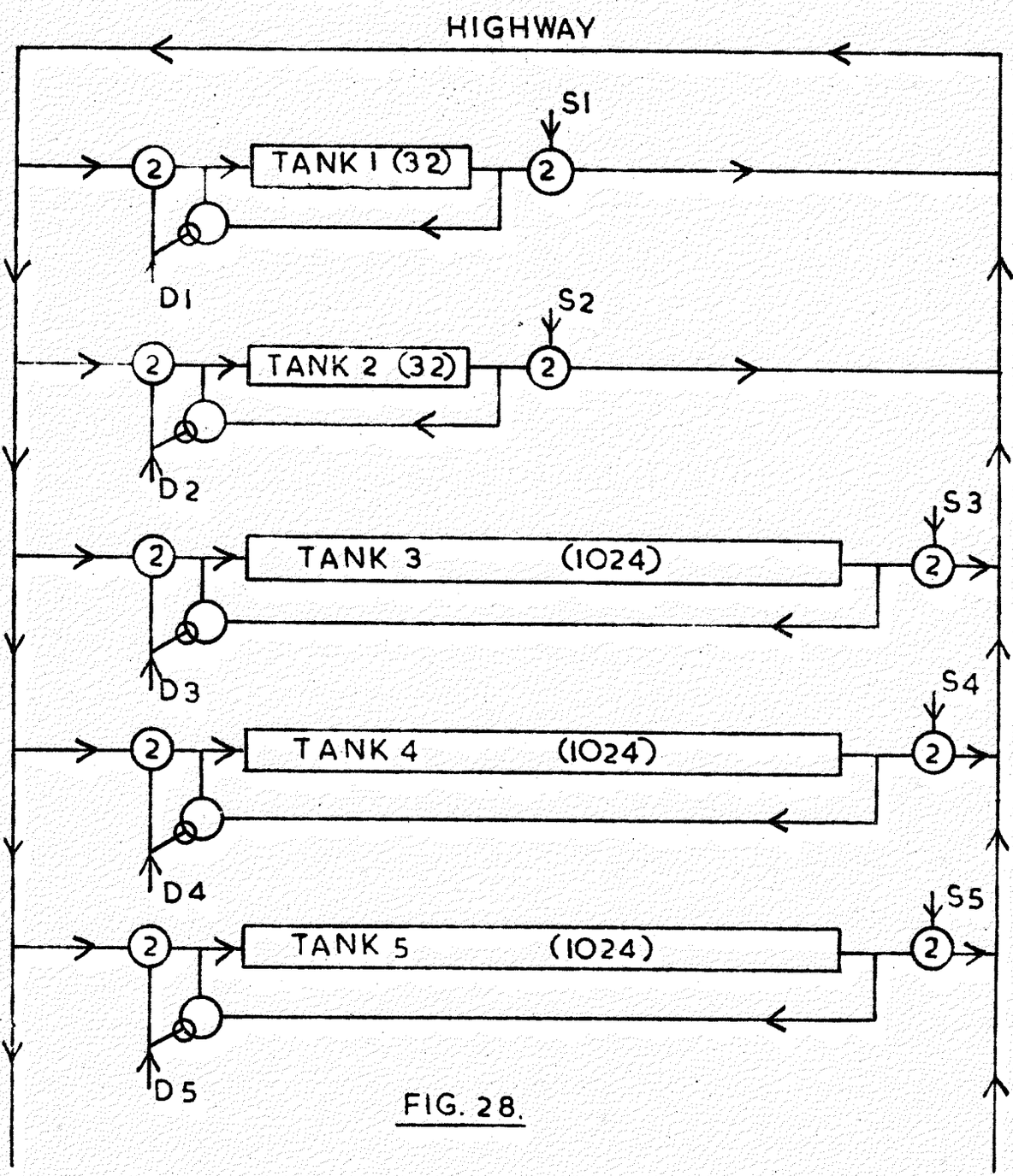


FIG. 28.

supply an unbroken stream of ones on the lines marked S1 and D2 for a period of time not less than 32 microsecs.; a copy of the contents of tank no.1 will then pass out into highway and the ones on the line D2 will enable them to pass into tank 2. After the operation has been carried out for 32 microsecs. its continuation serves no further purpose. As long as we confine ourselves to short tanks, any transfer may be described by an order consisting of two numbers e.g. the operation of transferring the contents of tank 1 to tank 2 may be written, 1 → 2. If we consider the problem of transferring a number of words from one long tank to another long tank then, in addition to specifying the two tanks involved, two further numbers are required which specify, in some manner when the transfer begins and when it ends. Unless the transfer begins and ends at the correct time the required words will not be transferred. Two remarks are pertinent at this stage; first it has been assumed that the words in two tanks are in phase; this requires the use of the ring counter; secondly the transfer from one long tank to another can only replace words of the receiving tank by the corresponding words of the source tank. The instruction needed has now become of the form S → D, T_1, T_2. Where T_1, T_2 are numbers required to time the transfer correctly.

So far the instruction has been used to effect a pure transfer, but it can be made to perform arithmetic and logical operations in quite a simple matter. Suppose we place an adding unit in the circulation of a short tank, as shown in Fig.29, and, as a second input, we have a line connected to highway via a destination 'gate'. We may give this destination gate a number, just as we assigned a destination number to each of the tanks. The result of transferring a number from any of the sources to this new destination is to add that number to the number already contained in the tank with the adding unit: this tank has now become an "accumulator". Since the adder is associated with a tank of length 32 the adder will need carry suppression at the end of every minor cycle. As an example of the type of operation which a single

transfer,

transfer can perform, we consider the transfer of m consecutive words of a long tank to the accumulator; the accumulator will form the algebraic sum of the m numbers plus its original contents. The tank plus adder will now have effectively two inputs. An input through the adder for adding numbers to its original contents, and a normal input for replacing its original contents by the number received. The accumulator may be used for subtraction by providing it with another destination which receives the complement of the number travelling in highway and transmits it to the adder. This is shown as destination 7 in Fig.29. The ones in the line D7 must also be used to insert a P1 in the carry position when subtraction is performed. This may be done as is shown at element A in Fig.29. Logical operations can be effected by a transfer in a similar manner. Fig.30. shows how we can form the sequence A & B from two sequences A and B. Suppose the outputs of two short tanks are fed continuously, into a limiter of threshhold two; the output of this limiter will be A & B if A, B are the contents of the two tanks. If this limiter is supplied with a source gate and is assigned a number then A & B may be called upon by a transfer in the same way as any other source. In Fig.30 the "and" facility has been provided on tank 9 and tank 10 and it has been given source number 20. It will be seen that Tank 9 'and' Tank 10 is being produced continuously at the output of the limiter C, but is only passed into highway when a sequence of ones is applied on the line marked S20. The operatinns A v B may be provided in a similar manner. In addition to the natural sources and destinations and the functional types, a number of rather special sources and destinations may be provided. The pattern emitted by the P1 element, for example, which consists of a one in the least significant position followed by 31 zeros is frequently required and may be used as a source, by supplying the P1 element with a source gate and a source number; the P32 pattern is also very useful. Similar sources are the clock, which provides 32 ones, and a blank, which supplies 32 zeros,

With this extension of the sources and destinations it will be seen that

any/

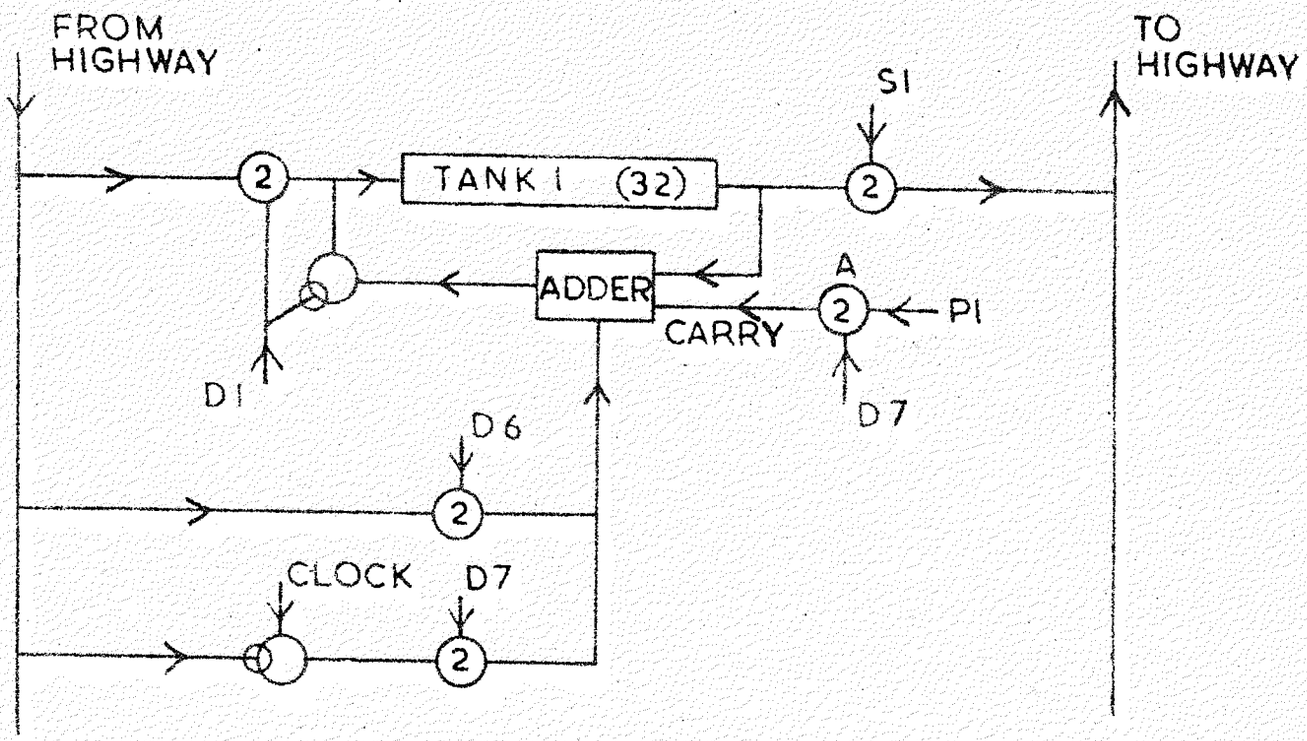


FIG. 29 ACCUMULATOR

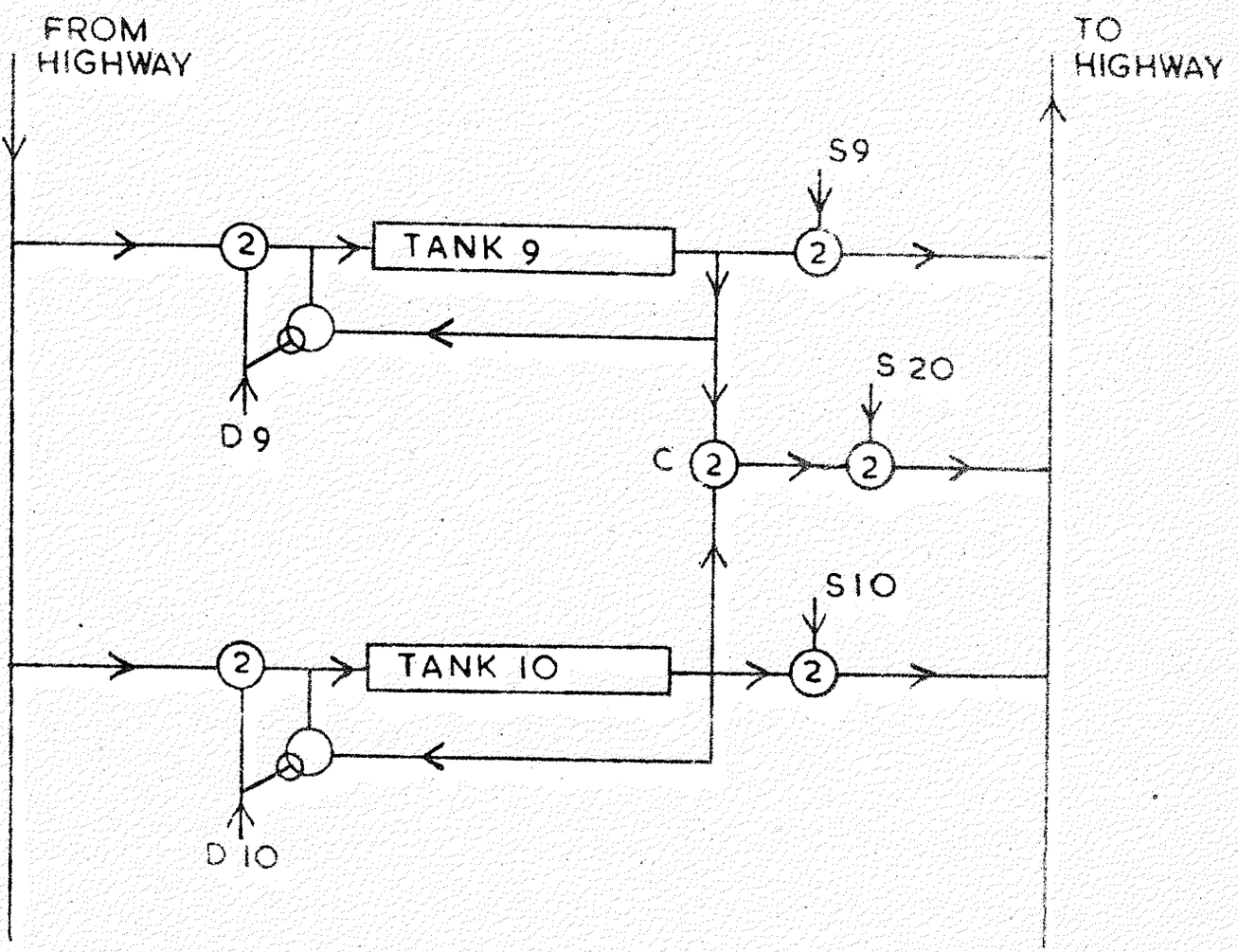


FIG. 30. LOGICAL OPERATION

any computation can be reduced to a sequence of coded instructions each of which is of the form $S \rightarrow D, T_1, T_2$. where S is any source, D is any destination and T_1 and T_2 are two timing numbers. Our coded instruction contains one other element which serves to determine which instruction is to be obeyed next. The most obvious way of arranging the coded instructions in the memory is to let each instruction comprise one word, and place successive instructions consecutively in long tanks. This has the disadvantage that after the control has read one instruction it cannot receive the next instruction until one complete major cycle afterwards. Since most of the manipulation will be performed on numbers which are being stored temporarily in short tanks this involves an unnecessary waste of time. The alternative adopted, is to space the instructions in such positions in the instruction tanks, that when one instruction is completed, the next is in the correct position for being obeyed. This will be explained more fully when the schematic circuits for the logical control are described. Using this method of spacing, it is sufficient for each instruction to give the number of the tank containing the next instruction; its position in that tank need not be specified since the programming is arranged in such a manner that this next instruction is available when the current instruction is completed.

The standard instruction in the pilot model therefore assumes the form, $S \rightarrow D, N, T_1, T_2$ where N is a number selecting the tank containing the next instruction. Before we can describe the schematic circuit of the logical control which interprets the code we must know the exact position of the elements S, D, N, T_1 and T_2 in the code word. The pilot model has 32 sources and 32 destinations; hence five binary digits are needed to represent each. Of the thirteen tanks used for storage (7 long and 6 short) only eight are used for storing instruction (7 long and 1 short) and hence three digits only, suffice to specify the tank containing the next instruction. Each of the timing numbers T_1 and T_2 is permitted to vary between 0 and 63 and hence 6 binary digits are required for each. The position in the word occupied by each of these

elements/

elements is as follows:-

P3	to	P7	Source number.
P8	to	P12	Destination number.
P13	to	P15	Next instruction number.
P19	to	P24	T_1
P26	to	P31	T_2

All other digits in the instruction words are spares and will be zeros.

Details of the components of the logical control.

The schematic circuit for the logical control is given in Fig.31. Before its functioning as a complete unit is described, we will consider the more important components separately.

(i) Staticisers for source, destination and next instruction source.

The digits detailing the source, destination and the next instruction source are staticised on the thirteen triggers numbered CI3 to CI15 in the bottom right hand side of the control diagram. (The notation CIn refers to the n^{th} digit of the "current instruction"). Triggers CI3 to CI7 control a tree of order five and thereby provide an unbroken sequence of ones of the source gate selected by the instruction. Similarly the triggers CI8 to CI12 select the required destination by means of a tree of order 5 and the triggers CI13 to CI15 select the next source of instruction by a tree of order 3. All these triggers retain their information throughout the transfer and are reset or "cleared" just before the next instruction is staticised.

(ii) The instruction delay line (length 32) INST.

During the time that a transfer is being performed the instructions from the tank which is to supply the next instruction, feed continuously into the short tank marked INST. The instruction which is trapped in INST when a transfer ceases, becomes the next instruction to be obeyed. The method by which the next source of instruction is selected and directed to INST is shown in Fig.32. Although the number, N, which selects the next source of instruction varies

from/

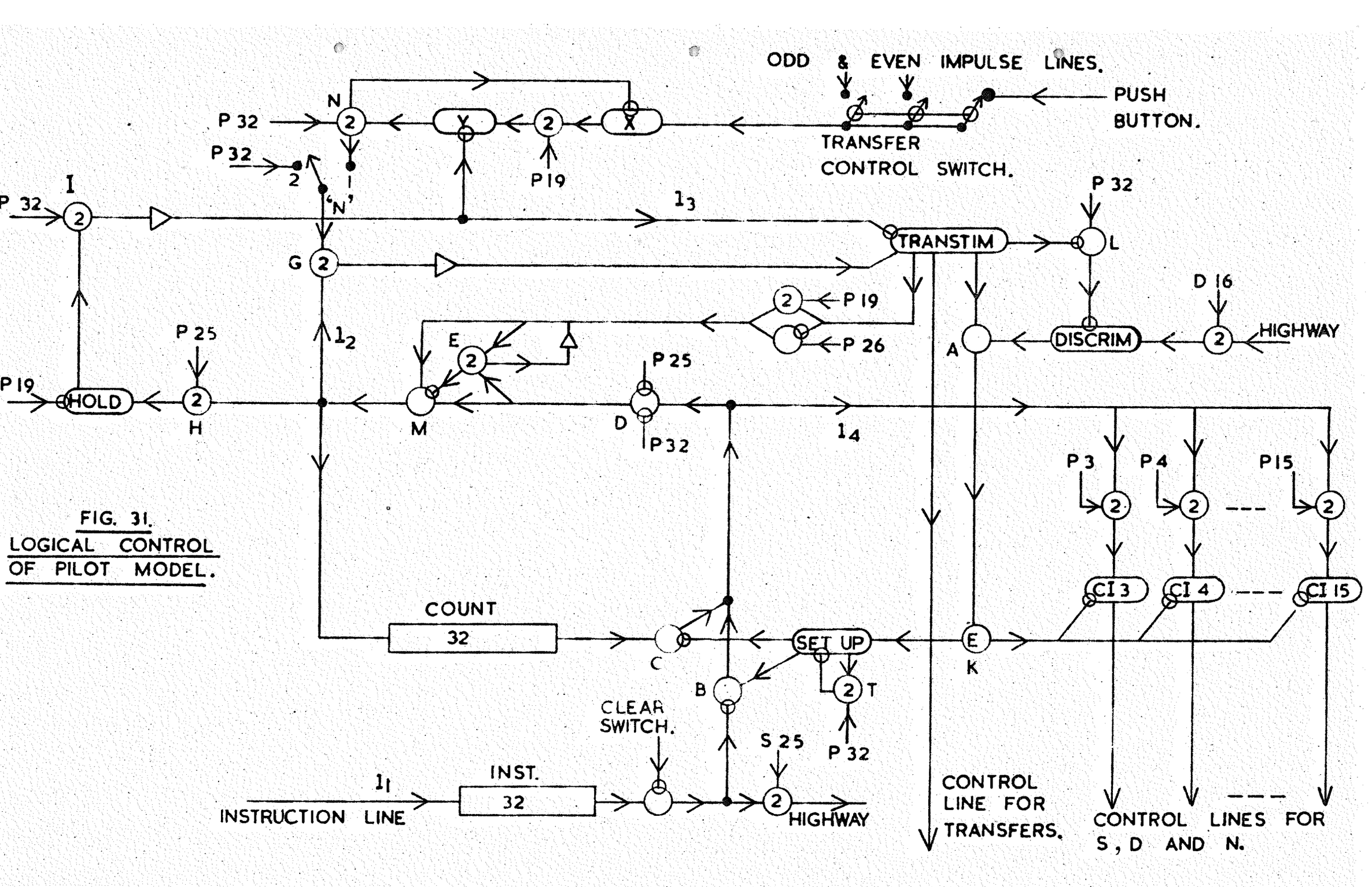
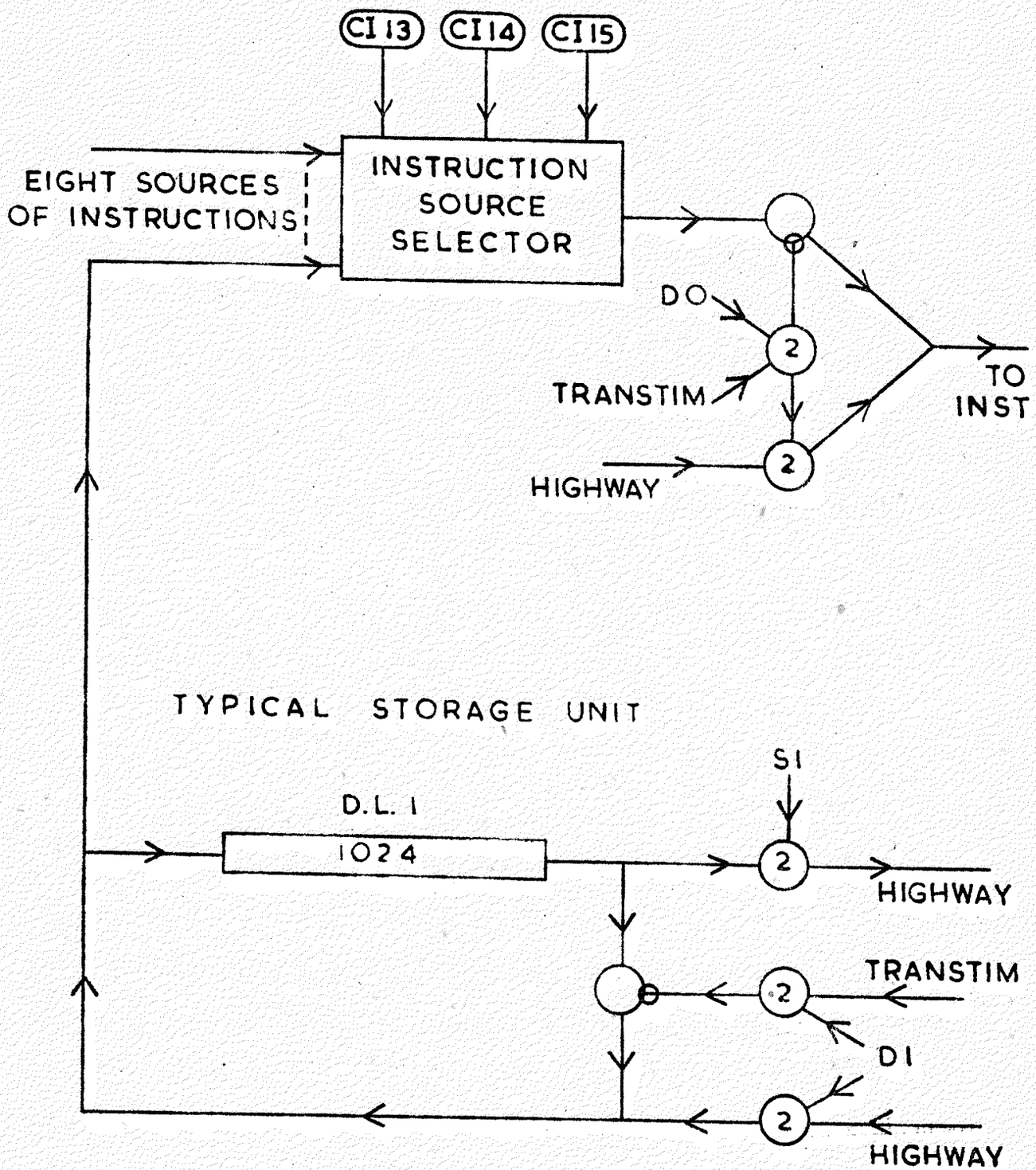


FIG. 31.
LOGICAL CONTROL OF PILOT MODEL.

FIG. 32.



CONNECTION OF INSTRUCTION SOURCES FOR PILOT MODEL.

from 0 to 7, the numbers of the tanks which hold the next instruction are 1 to 8. This is because source 0 and destination 0 have to be used for rather special purposes connected with the "initial input" problem. For this reason when the next source of instruction number is N, the next instruction comes from tank number 'N + 1'.

(iii) The trigger TRANSTIM

The trigger, TRANSTIM, is the heart of the control since it determines the actual period during which transfer from the source to the destination takes place; a transfer is taking place if and only if TRANSTIM is on. Up to the present it has always been assumed that provided a source and a destination were being supplied with ones a transfer was taking place. This was convenient before the control circuit was described, but is not, in fact, strictly true. There is a main gate in highway which is operated by TRANSTIM; until TRANSTIM is on, nothing can pass along highway although a source gate and a destination may be receiving ones. Similarly, there are provisions to ensure that a delay line does not have its contents replaced by the number in highway, even though its source gate is supplied with ones, until TRANSTIM is on. These are shown for a typical tank, in Fig.32.

(iv) The counter.

The counter consists essentially of a short tank with an adder in its circulation and also an element D which removes any ones in the P25 or P32 positions. If TRANSTIM is off the number in the tank has a P26 added to it once per circulation until a digit is formed in the P32 position; this P32 digit then puts TRANSTIM on (assuming switch 'N' is at position 2) and is immediately removed at D. When TRANSTIM is on, the number in the tank has a P19 added to it once per circulation until a one is formed in the P25 position. This is detected by the element H, remembered by the trigger "HOLD" until the next P32 time when it puts TRANSTIM off again. It will be seen that the counter acts as a timing mechanism for determining when TRANSTIM is put on

and/

and off. Since the adder is merely called upon to add the single digits P19 or P26 to the number in COUNT, there is no stage at which we have two current digits and a carry over; hence the adder in the circuit may be of the simplified type usually termed 'a half adder'.

(v) The trigger SETUP.

The trigger SETUP is put on for a period of one minor cycle after each transfer. It supplies the next instruction, with ones and zeros interchanged, to the counter and then puts itself off via the gate T.

Operation of logical control.

The description of the operation of the control as a whole may now be given. It will be assumed for the time being that switch 'N' is set at position 2 so that triggers X and Y may be ignored; trigger DISCRIM may also be neglected at this stage.

Let us suppose that the previous instruction was completed at the end of minor cycle number m, and that at the time when it was completed the instruction 'trapped' in the delay line INST (bottom of figure) was

$$S \longrightarrow D; \quad N; \quad T_1, \quad T_2.$$

Since this ran into INST during minor cycle m, it must have occupied the m^{th} position in the instruction tank from which it was derived. On the termination of the transfer TRANSTIM goes out and (since we are ignoring the effect of DISCRIM for the moment) the end element K sends out a one which clears the staticisers and puts SETUP on. SETUP will therefore be on during the whole of minor cycle (m + 1). During this minor cycle (m + 1) SETUP will delete the present contents of COUNT, at the element 'C', and will feed into the COUNT circuit, the new instruction with its polarity changed, via the element B. The result of the polarity change is that the instruction

$$31 - S, \quad 31 - D, \quad 7 - N, \quad 63 - T_1, \quad 63 - T_2$$

passes via D into count. Positions P25 and P32 of the instruction, before polarity change, were blank i.e. zeros and hence they are ones when they go

into/

into element D. At D the P25 and P32 are removed. Since TRANSTIM is off, the adder will be adding P26's, and hence at the end of minor cycle (m + 1) the instruction in count will be

$$31 - S, \quad 31 - D, \quad 7 - N, \quad 63 - T_1, \quad 64 - T_2.$$

During the same minor cycle, (m + 1), the parts of the instruction relating to S, D and N will be staticised on CI13 to CI15; it will be noted that 31-S, 31-D and 7-N are staticised instead of S, D and N but this is of no significance since this merely determines the particular manner in which the triggers CI13 to CI15 are to be used to perform the selection. By the end of minor cycle (m + 1) the source, destination and next instruction source are statisiced and the number in COUNT has $63-T_1$ and $64-T_2$ in the positions occupied by the timing numbers.

In each subsequent minor cycle the number $64-T_2$ is augmented by one, due to the addition of P26 until finally in minor cycle $m + 1 + T_2$ it becomes 64 and thereby provides a one in the P32 position. This passes via line l_2 through the gate G and, after being delayed 1 μs, to TRANSTIM, which is put on. The one in the P32 position is obliterated at D on the next circulation. Immediately TRANSTIM is put on the gate on HIGHWAY is open and the transfer begins to take place. The number T_2 therefore determines how long the control 'waits' until it permits the transfer to start. Since TRANSTIM is now on, COUNT will have a P19 added in each subsequent minor cycle; this means that the number which was originally $63-T_1$, will be augmented by one in each minor cycle until in minor cycle $m + 1 + T_2 + T_1 + 1$ i.e. $m + T_1 + T_2 + 2$ it will become 64 and will provide a one in the P25 position. This P25 passes through the gate H and puts on the trigger HOLD. The trigger HOLD sends out ones from P25 onwards until the P32 line, when one of these ones is permitted to pass the element I whence it passes via l_3 after a unit delay, to TRANSTIM, which it puts off. Hence TRANSTIM goes off at the beginning of minor cycle $m + T_1 + T_2 + 3$ after having been on for minor cycles $m + T_2 + 2$, $m + T_2 + 3$, ... , $m + T_2 + T_1 + 2$ i.e. for $T_1 + 1$ minor cycles; this means that the transfer from the source S

to/

to destination D is continued for $T_1 + 1$ minor cycles (N.B. it is necessary to use T_1 for a transfer of $T_1 + 1$ minor cycles to span transfers from 1 to 64 minor cycles while using numbers 0 to 63 for the timing number). The numbers T_1 and T_2 are known as the 'transfer' and the 'wait' numbers, respectively. For the whole of the time after minor cycle $m + 1$, the instructions from the instruction source corresponding to the number N have been running into INST. When TRANSTIM is put off in the beginning of minor cycle $m + T_1 + T_2 + 3$ the instruction which is trapped in INST is the instruction whose position in N is word number $m + T_1 + T_2 + 2$; this is therefore the next instruction to be obeyed and in making up our instruction table, our next instruction must be placed in this position. Two points remain to be clarified (a) the source, destination and next instruction source are staticised digit by digit during minor cycle $m + 1$, and therefore at various stages during this minor cycle different numbers are set up on the staticiser. This means that before the staticiser reaches its ultimate value, it registers a varying source, destination and source instruction number. It is however of no consequence, because no transfer takes place in this minor cycle and the instruction of mixed origin which passes into INST is never used. (b) The parts 31-S, 31-D, 7-N of the instruction in COUNT pass to the staticiser during every minor cycle of the 'wait' and 'transfer' period; this clearly leaves the staticiser unaltered.

Discrimination.

We may now consider the modification produced by the trigger DISCRIM on the operation of the control. The trigger DISCRIM is a device used to effect discrimination. From time to time during a computation it will be necessary to follow one of two courses depending on the result of previous computations. Such a discrimination is achieved in the following way. The trigger DISCRIM is one of the 32 possible destinations, and an instruction is given which either sends an unbroken sequence of zeros to this trigger or a word containing at least one 'one', according to which course the computation should take. If an unbroken sequence of zeros is sent to DISCRIM it will not be put on, but if it receives at least one 'one'

it/

it will be 'tripped'. It will be seen from Fig. 31 that if an instruction is given which puts DISCRIM on, then after TRANSTIM is put off (i.e. the transfer ordered ceases), DISCRIM remains on for one further minor cycle before it is put off via L and provides a sequence of ones to the element A. The effect of this is that the end element K gives out the pulse which clears the staticiser and trips SETUP, one minor cycle later than would normally be the case. Hence if the instruction in position in a long tank is S→DISCRIM, N, T_1, T_2, the instruction next obeyed will be number $(m + T_1 + T_2 + 2)$ or $(m + T_1 + T_2 + 3)$ of tank 'N + 1' according as the instruction tripped DISCRIM or not. Its action on DISCRIM will therefore determine which of two instructions will be obeyed next and thereby affects a discrimination.

Input and Output.

The Input and Output requirements of the pilot model will be met by using a Hollerith reproducer unit, both for reading in numbers punched in binary form or decimal form and for punching out information in binary or decimal form. Printed decimal information will be obtained by printing from the punched cards. The decision to use Hollerith equipment for the external organs of the A.C.E., in the earlier versions, at least, in spite of the rather low rate of input and output (a Hollerith reader passes 100 cards per minute), was taken for the following reasons. The Hollerith equipment is already completely developed and needs very little modification for use in conjunction with the A.C.E. and it will prove very convenient, particularly in the early stages, to be able to use the standard Hollerith equipment for sorting, collating and other similar purposes. Moreover, there is quite a considerable range of problems which give rise to a large volume of internal high speed computation for quite a small input and output, and for these problems the slow nature of the Hollerith equipment is of little consequence. If, later, the A.C.E. is required to perform computations for which high input and output rates are essential the modifications necessary to change over to another form of external equipment should not prove very extensive.

A/

A Standard Hollerith card has 80 columns and twelve rows, the latter being numbered Y,X,0,1,2, .. 9. Of the eighty columns only 32 will be used directly in combination with A.C.E. equipment, namely columns 40 to 71. The other columns will be used for various other purposes, such as recording the identification number of the problem to which it relates and other similar information.

A card bearing binary numbers may have one 32 digit binary number in each of its twelve rows, but only one decimal number will be punched on each card. A decimal number will be punched in standard Hollerith code and since we are not interested in general, in numbers of more than ten decimal digits such numbers will be punched in columns 62 to 71 with a sign indication in the 0 row (zeros may be omitted in punching the number).

The Hollerith equipment is co-ordinated with the A.C.E. during normal operation by the provision of a number of special sources and destinations which are described below.

Input Dynamiciser.

As each row of a card passes the reading station, the thirty two reading brushes associated with columns 40 to 71 make contact with the Hollerith reader contact roll in each position where a hole has been punched, (Fig.33). Consequently each of the lines associated with the brushes, is at a positive D.C. voltage (100 volts) or zero, according as there is a hole or not in the corresponding column of the card. These thirty two voltages are dynamicised, and the dynamic sequence obtained is provided with a source gate. For reasons connected with the problem of "initial input" (i.e. the input necessary to get the machine 'under way' when it is started up) the source associated with this input dynamiciser has been made source 0. At any time when a row is being read then, source 0 provides us with the word contained in that row, in dynamic form, once per minor cycle. It will be seen from Fig.33 that the input dynamiciser may also be connected to a set of 32 manual switches; this means that numbers may be set up by hand and fed into the machine.

READ/

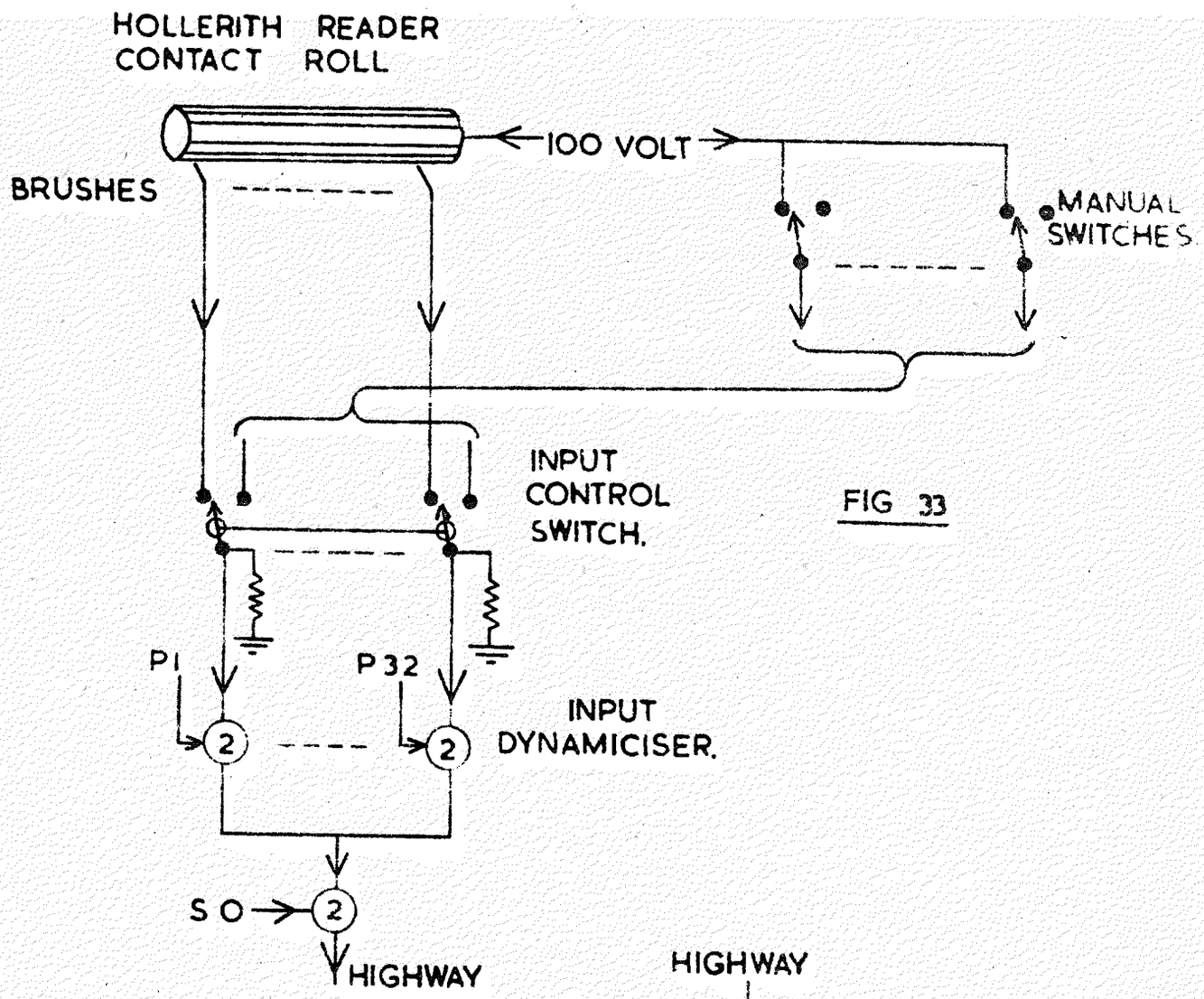


FIG 33

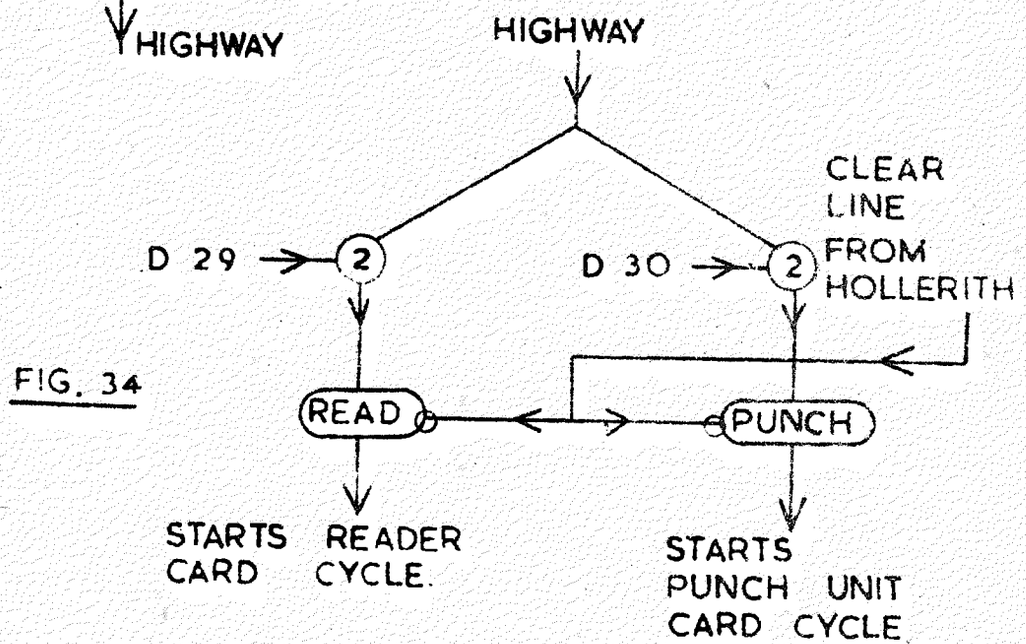


FIG. 34

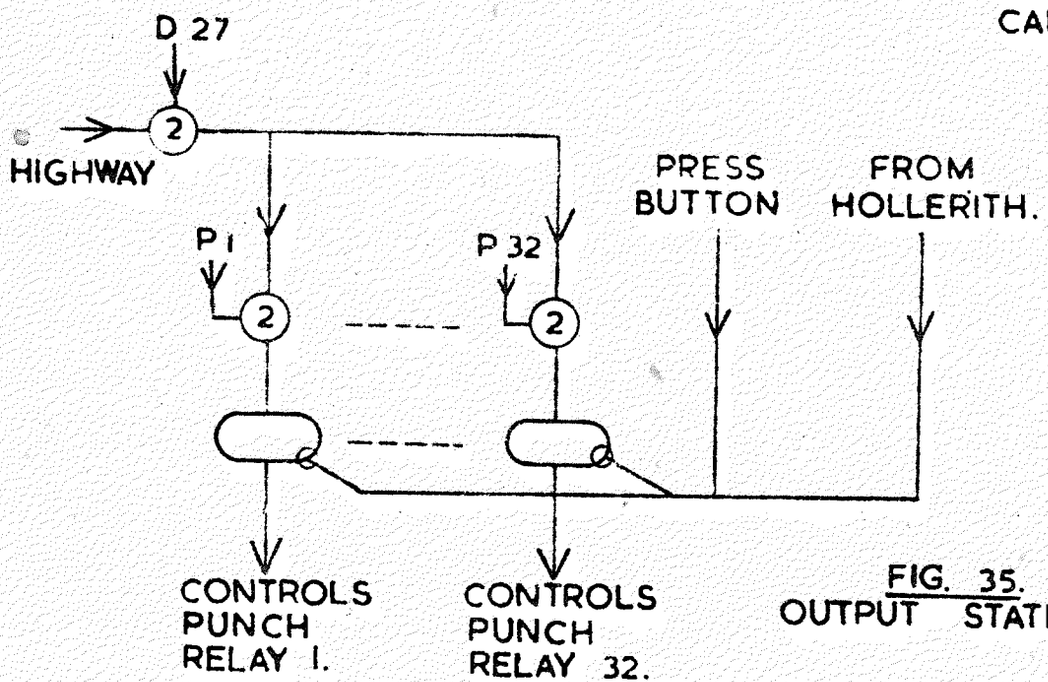


FIG. 35.
OUTPUT STATICISER.

READ and PUNCH triggers.

There are two special triggers associated with the input and output functions by means of which the A.C.E. may initiate reading and punching. (Fig.34). The READ trigger has destination 29 associated with it, and is such that when a signal containing at least one 'one', is sent to destination 29, READ is put on and starts a card cycle (i.e. the passage of one card). The drive clutch holds automatically until the last row of the card is past the reading station. The trigger READ is put off automatically by the Hollerith equipment as soon as a card cycle starts. If a further card is required immediately, the drive clutch should remain engaged to avoid loss of time; in order to ensure this, READ must be put on again some time between the reading of the first row and about 60 major cycles after reading the last row. From the point of view of programming the input, all that is essential is that when a second card is to be read, READ must be put on again after reading the first row in order to make certain that this restimulation follows the automatic clearing of READ. The trigger PUNCH, which is associated with destination 30, is connected to the Punch Unit so that when it is put on, a card cycle is initiated in the latter and automatically completed, PUNCH being put off by the Hollerith as soon as the cycle begins.

Odd and even impulse line. The rows of the Hollerith are each in the reading position for a number of major cycles (about 6) and between each of the rows there is a somewhat greater number of major cycles; the same is also true of punching. There must therefore be some means of synchronisation between the A.C.E. and the Hollerith which not only makes certain that each row is read (or punched) but also ensures that no row is read twice. This co-ordination is provided by the 'odd and even impulse lines' which are sources 29 and 30 respectively. When any odd row of a card is in position for reading or punching the Hollerith provides a sequence of ones on the odd impulse line; these ones are available at source 29 and thereby we have a method of discriminating whether an odd row is in position or not. Similarly for even

lines/

lines, the 'even input line' provides the same facility by means of source 30.

The outline of operations for reading a card is therefore of the form.

(i) Stimulate READ.

(ii) Send source 29 (odd impulse line) to DISCRIM the next instruction being to repeat (ii) or go on to (iii) according as DISCRIM receives zeros or ones.

(iii) Read input i.e. odd row of card.

(iv) Send source 30 (even impulse line) to DISCRIM, the next instruction being to repeat (iv) or go to (v) according as DISCRIM receives zeros or ones.

(v) Read input i.e. even row of card.

The above cycle is repeated from (ii) until all the rows of a card are read. If a further card is required READ is restimulated after the first row has been read.

The outline of the operations for punching a word are similar, viz.

(i) Stimulate PUNCH.

(ii) Send source 29 (odd impulse line) to DISCRIM, the next instruction being to repeat (ii) or go to (iii) according as DISCRIM receives zeros or ones.

(iii) Set up odd row on staticiser.

(iv) Send source 30 (even impulse line) to DISCRIM, the next instruction being to repeat (iv) or go to (v) according as DISCRIM receives zeros, or ones.

(v) Set up even row on staticiser.

etc.

The instructions of type (ii) and (iv) are usually given briefly as "Wait for odd impulse line" and "Wait for even impulse line".

The method by which the rows and cards are counted during input and output will be fully described later.

External Control of the machine by push button and Hollerith.

For checking purposes it is convenient to have some means by which the control mechanism may be put into "slow motion". The nature of the dynamic storage used in the A.C.E. makes it virtually impossible to do this by

reducing/

reducing the pulse frequency but it is possible to make the control obey the orders of any instruction table one by one, on the successive stimulation of a push button, in quite a simple manner. It will be remembered that the initiation of transfer in the automatic control is by means of a P32 pulse which has been gated at the element G (Fig.31). This P32 pulse arises from the slow counter when the counter has raised the number 63 - T_2 to 64. If this P32 pulse is not gated at G the number in COUNT in the wait position immediately reverts to zero because the 64 (which is represented by the P32) is deleted at the element D. Failure to gate the P32 pulse at G means that TRANSTIM is not put on, and hence COUNT is still counting in the "wait" position; it will therefore count for another 64 major cycles before it sends out a further P32 pulse to G. As long as G is not supplied with the gating P32 pulses this process of counting up to 64 and then reverting to zero will be carried on indefinitely. If at some subsequent period, gate G is permitted to function, the instruction originally staticised will be executed although it will occur some multiple of 64 major cycles later than in normal operation. Since the configurations in all the tanks (except those of the multiplier) are repeated every major cycle the delay will not effect the instruction in any way. This is the basis of the push button operation of control, which will now be explained.

Let us suppose that the TRANSFER CONTROL SWITCH has been switched to PUSH BUTTON (Fig.31 TOP) and the 'N' switch is set at position 1. This means that the gate G is only supplied with ones when the trigger Y is on. Let us suppose further that Y is off and an instruction has been staticised. Until the push button is operated the counter will continue to count up to 64 and then revert to 0 repeatedly. As soon as the push button is operated, a single pulse will be sent to trigger X at some arbitrary time; trigger X will send out ones, and at time P19 one of these will put on trigger Y; from then onwards the gate N will pass P32's to the gate G and also to the inhibiter on X. The first of these will put X off again. The next time the counter sends a P32 pulse to G it will therefore pass through and put TRANSTIM on. The transfer required will then take place, and on its completion TRANSTIM will be put off. The pulse

which/

which puts off TRANSTIM will also inhibit Y. Hence the operation of the push button will ensure the performance of one operation only, after which the control will revert to its previous inert state until the push button is operated again. During the inert state the latent transfer will be staticised and therefore the source chosen will be sending out its signal into HIGHWAY, where it will be obstructed because the main HIGHWAY gate will be closed. If a scope is fitted on the HIGHWAY before the main gate, the number whose transfer is to be ordered by the instruction may be seen before the instruction is actually carried out. PUSH BUTTON operation therefore provides us with two very useful checking features.

The external control of the machine by means of the Hollerith is achieved in a similar manner. The TRANSFER CONTROL SWITCH is set to 'Hollerith' and as a result of this the trigger X receives a pulse whenever an odd or even row of a card is in the reading position. Thus the control executes one transfer for each row which is read.

Multiple-purpose components.

For economy of equipment a number of components of the pilot model are used for dual purposes. One such unit is the short tank, TS11. This tank is normally used as a simple temporary storage, but by setting a trigger its recirculation may be inhibited and its input permanently connected in series with the long tank DL1. The trigger which effects this transformation is shown as TC32 in Fig.36. The trigger TC32 is set by sending a signal to destination 22 and is reset by sending a signal to destination 23. During a period when TC32 is set, source 11 provides the contents of delay line 1, delayed by a minor cycle. The facility is of very frequent use in repetitive cycles of operations when the words of a long tank are required in succession, one for each cycle of operation. If an order is contained in the cycle which delays the whole contents of a delay line by one minor cycle, then the successive words of that delay line may be transferred by an instruction which always take place in the same minor cycle.

Another/

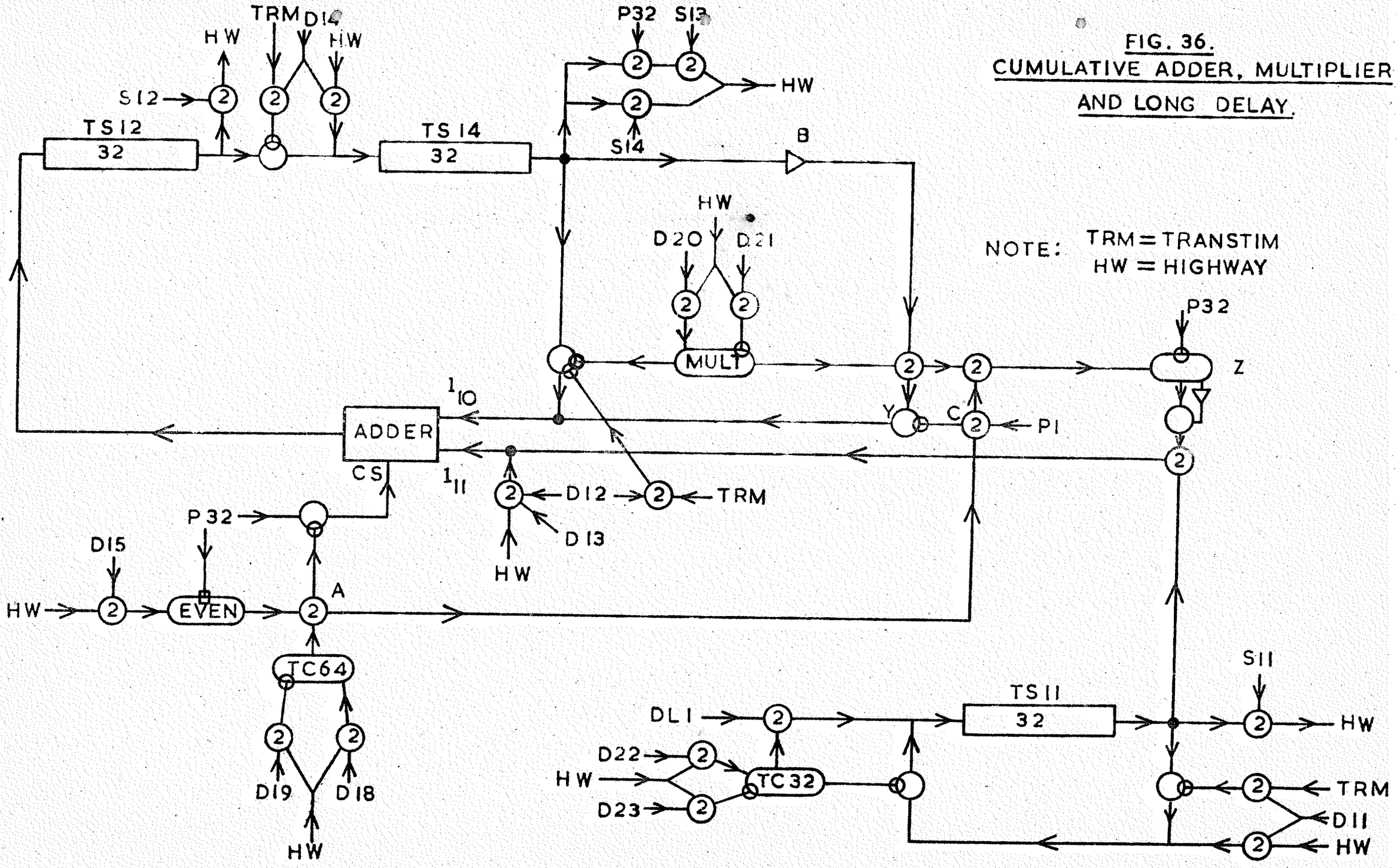


FIG. 36.
CUMULATIVE ADDER, MULTIPLIER AND LONG DELAY.

Another multi-purpose piece of equipment is that associated with the accumulator and multiplier (Fig.36). The accumulator consists of two short tanks TS12 and TS14 in series with an adder. In normal operation each of these short tanks may be used as an accumulator for numbers of one-word length. Numbers may be added to either by sending them to destination 13 at the appropriate time, while each of them has an ordinary destination gate (destinations 12 and 14) by means of which their contents may be replaced by other numbers. When these short tanks are being used as one word accumulators the carry suppression takes place at the end of each minor cycle. The trigger TC64 enables the tanks 12 and 14 to be used as a single two word accumulator with carry suppression at the end of odd minor cycles. This transformation makes use of the trigger marked EVEN which is on in even minor cycles and off in odd minor cycles. If TC64 is put on by a signal to destination 18, triggers TC64 and EVEN together have the effect via gate A of reducing the carry suppression to the end of odd minor cycles only.

A third trigger, MULT, converts the accumulator into a multiplier, for which the multiplicand is stored in TS11. It will be seen that when MULT is put on by a signal to D20, the normal recirculation of TS12 and TS14 is inhibited, and circulation takes place via a unit delay B. The combination TS12, TS14 and the unit delay become effectively the tank of length 65 described in the basic multiplier circuit. Multiplication can only be performed when TC64 is on since it is essential that gate 1 should pass alternate P1's only. (This was explained in the description of the basic multiplier circuit). To perform a multiplication the accumulator must be cleared and the multiplicand sent to TS11. MULT must then be put on and the multiplier sent to the accumulator. When the time taken for multiplication is completed, which will be 65 or 66 minor cycles later according as the 'multiplier' is sent to destination 14 or destination 12, MULT must be put off and the product will be left circulating in TS12 and TS14. An appropriate shift must be given to the product to position the decimal point correctly and the round off may then be applied by adding a P32 to the minor half of the resulting number.

Miscellaneous/

Miscellaneous facilities in pilot model.

Trigger TC98. The trigger TC98 is used for "remembering" one binary digit. It may be put on by a signal to destination 28 and put off by a signal to destination 24. Typical use of this trigger is for remembering the sign of a number while a number of computations are performed on the absolute value of that number. The trigger may be put on if the number is negative and left off if the number is positive; the state of the trigger may then be used to discriminate between operations which depend on the sign of the number.

INST. The short tank INST which forms an important unit of the control may be used as a source or a destination. It's source number is 25 but its destination number is 0. The explanation of this is given in the section below on initial input. One of its chief uses as a source occurs at positions in an instruction table, where some constant of the computation is required. Such constants are planted among instructions as though they were themselves instructions, and are called upon by using source 25 at the appropriate time. It is also used as a source when, during the course of computation, it is necessary to construct an instruction, some of the elements of which will be determinate when the instruction table is made but other elements of which will depend upon the results of the computation itself. The fixed elements of such an instruction will then be planted among the instructions and the complete instruction built up from these elements at the appropriate time. Numerous examples of this use will occur in the section on coding.

INST is used as a destination occasionally in the course of automatic computation but mainly in the initial input stage of operation. When an instruction is sent to destination 0 it passes into INST, at the same time overriding the next source of instruction which is selected by the current instruction (Fig.32).

Temporary storage 9 delayed one unit.

The contents of temporary storage 9 are available, delayed by one unit at source 19. By using an instruction with transfer no. n which transfers source 19 to destination 9 the contents of 9 may be moved round n positions. It should

be/

be noted that by this means a number which consists originally of digits $a_{32}\ a_{31} \cdots a_2\ a_1$ (a_1 being the least significant) becomes $a_{32-n}\ a_{31-n} \cdots a_2\ a_1\ a_{32}\ a_{31} \cdots a_{33-n}$. The problem of multiplying a signed number by 2^n is in general different from this. The multiplication of a signed number by a factor 2^n is obtained by sending the number to the accumulator and then adding the accumulator to itself repeatedly by using the appropriate transfer number. This gives a simple method of multiplying single and double word numbers by powers of two.

Failure Bell.

At various stages of the computation, checks of various types may be programmed and discriminations performed on the findings of these checks. A Failure Bell is provided which may be stimulated by sending a signal to destination 31. In the pilot model it is assumed that there is no destination by means of which the A.C.E. may put the failure bell off. Failure of a check is assumed to require human intervention.

Sign detection in the accumulator.

The number TS14 & P32 is available at source 13. This source will consist of a 1 in the 32nd position if the number in TS14 is negative and a zero otherwise. It may therefore be used to detect the sign of either of the numbers in the accumulator.

A full table of the possible sources and destinations in the pilot model is given below. The explanation of the choice of somewhat incongrous numbers, such as in TC98 etc., is that these numbers are derived from source numbers given in the original version of the full scale model.

ACE. (PILOT MODEL). INDEX.

	SOURCE	DESTINATION	
0	INPUT	INST	
1	DL	DL	
2	DL	DL	
3	DL	DL	
4	DL	DL	
5	DL	DL	Available as Instruction Sources
6	DL	DL	
7	DL	DL	
8	TS	TS	
9	TS	TS	
10	TS	TS	
11	TS	TS	TC 32 connects TS11 to the output of DL1 as a delay of length 32
12	TS	TS	
13	TS14&P32	TS12 through Cumulative Adder	TS12 and TS14 are in series with an adder, forming a tank of length 64.
14	TS	TS	
15	~TS11	EVEN	
16	TS9≠TS10	DISCRIM	
17	TS9&TS10		
18		TC64	
19	TS9▷1	Clear TC64	
20	TS9&P32	MULT	
21	$2^{32} - 1$ (Ones)	Clear MULT	
22	P1	TC32	
23	P16	Clear TC32	
24	P32	Clear TC98	
25	INST	TS12 through cumulative adder after change of sign (i.e. for subtraction).	
26	DL	DL	
27		Output Staticiser	
28	TC98	TC98	
29	Odd Hollerith Impulse Line	READ	
30	Even Hollerith Impulse Line	PUNCH	
31	0(Zeros)	Failure Bell	

Initial Input.

The initial operation of the machine, when power has just been switched on, presents special problems because the configurations of the various units will be largely unpredictable. The only extra facility used in this initial stage is a CLEAR switch on INST. When the clear switch is put on all the digits of the instructions passing out of INST become zero. i.e. it is as though the instruction from INST had been 0→0, 0, 0, 0. The initial procedure is as follows

(i) Set the TRANSFER CONTROL switch to PUSH BUTTON and switch 'N' to 1. This stops the machine transferring, but leaves an arbitrary word on the staticiser waiting to be obeyed.

(ii) Set 'clear' switch to 'on'.

(iii) Operate the push button. (ii) and (iii) have the effect that the arbitrary instruction is obeyed and 0→0; 0; 0; 0 is set up on the staticiser. The meaning of this instruction is that source 0 i.e. the input dynamiciser is to be passed to destination 0 i.e. INST for one minor cycle, the use of destination 0 overriding the choice of the next source of instruction which is delay line 1.

In case the useless instruction obeyed had the effect of stimulating READ, PUNCH or FAILURE BELL the next operations are

(iv) Stop possible FAILURE BELL.

(v) Wait until possible reader and punch unit card feeds stop i.e. until any initiated cycles are completed.

(vi) Set CLEAR switch to off.

All the externally controllable organs are now in a state for use, and we proceed to the second part of the initial input drill, using the Hollerith reader.

(vii) Put the Initial Input pack of cards into the Reader hopper.

(viii) Switch the Transfer Control Switch to Hollerith.

(ix) Start the Reader card feed and leave the card feed on.

So far we have the word 0→0; 0; 0; 0 set up in the Control waiting to be

obeyed/

obeyed as soon as trigger X receives a signal. This occurs when the first row reaches the reading station. The execution of the transfer 0→0; 0; 0; 0 has the effect that the first row of the card is sent to INST, and staticised in the following minor cycle, ready to be obeyed when the next row of the card arrives. When this first row is obeyed however, the next instruction is taken from one of the storage units and is probably quite arbitrary as the tanks are not necessarily empty. To avoid this the first row of the Initial Input pack of cards contains the word 31→1; 0; 31; 0 which orders delay line 1 to be filled with zeros and the next instruction taken from it. This next instruction (0→0; 0; 0; 0) will then again transfer the word in the next row of the card to INST. As the cards of the Initial Input pack feed in, the A.C.E. will perform the transfers punched in alternate rows of the card, and the fixed instruction 0→0; 0; 0; 0 alternately. The first few instructions put MULT, TC64, TC32, and TC98 off (their previous state is, of course, unknown) and then stimulate EVEN so that the reference datum for numbering minor cycles can be fixed relative to it. These operations take up exactly one card. The instructions for the standard table to be used for automatic input may now be fed in, in a similar manner; it should be noted that the position in which the first of these instructions is placed in a long tank determines which word of that tank (and therefore of all others) shall be the first i.e. it determines the reference datum for minor cycle numbers; it must be placed in such a position as to agree with the state of EVEN.

(x) When all seven cards have been read the TRANSFER CONTROL SWITCH is set to PUSH BUTTON and the A.C.E. is ready to enter the Input Table as soon as we switch to automatic operation. This completes the Initial Input drill and to start automatic operation we must switch 'N' to 2.

5./

5. Extensions of the coded instruction word in the full scale A.C.E.

Although a number of problems have been coded for the pilot model, they are not included in this report. The limited nature of the storage in the pilot model makes it necessary to resort to a number of special tricks in the coding, as a result of which the instruction tables are not so readily intelligible to anyone who has not made a special study of the machine.

Quite apart from the artificial limitations imposed by the small storage capacity, experience in programming for a machine using the type of instruction word imployed by the pilot model reveals a number of shortcomings in this instruction word itself. The association of the arithmetic and logical operations with special units means that quite a number of instructions are worded in directing numbers to these units and then back again to their 'normal' storage positions. A more flexible type of instruction would be one which gave an order of the type. 'A function B to C' i.e. had two sources a function and one destination. Typical instructions of this type are.

$$A + B \rightarrow C$$

$$A - B \rightarrow C$$

$$A \,\&\, B \rightarrow C$$

Single source instructions may be represented in this system by instructions of the form.

A + zero $\rightarrow$ C.

In the large scale A.C.E. the coded words and the logical control have been extended to cover orders of this type. The full instruction in fact is written in the form.

Source one, function, Source two, destination, characteristic, timing number, next source of instruction.

It/

It is assumed that there will be 256 possible sources and destinations and therefore 24 digits are needed to cover source one, source two and the destination; the total number of functions has been limited to 16 and the next source of instruction may be drawn from any of 32 tanks (31 long, and 1 short). This requires another 9 digits. The timing number may take any value between 0 and 31 and the characteristic one of four possible values 0 to 3. The total number of digits in the instruction is therefore 40. Up to the present stage it has always been assumed that the words were 32 digits long. From now on they will be 40 digits long and consequently a short tank will be 40 digits long and a long tank 40 x 32 = 1280 digits long, i.e. a long tank will still hold 32 words. The major and minor cycles now become 1280 microsecs and 40 microsecs respectively. This naturally requires that the ring counter now have 40 elements numbered P1 to P40 and a number of other obvious changes of this type. Since 2^{40} is rather greater than 10^{12} we can now cover numbers of 12 decimal digits.

The only other alteration in the instruction word is that the two timing numbers have been replaced by one timing number and a 'characteristic' which may take one of four values. The combined effect of the timing number and the characteristic for the four possible values of the characteristic are defined as follows.

(1) and (2). The first two values of the characteristic given rise to what are known as 'immediate transfers'. An immediate transfer with timing number 'n' takes place for n + 1 minor cycles immediately after it is staticised and the next instruction is the one in INST at the end of the transfer. In order that immediate transfers may take place for any period from 1 to 64 minor cycles two values of the characteristic are used to cover this type of transfer. If the first value of the characteristic is used the transfer takes place for a number of minor cycles one greater than that given

by/

by the timing number but if the second value is used the transfer takes place for 33 minor cycles longer than indicated by the timing number. In writing instruction tables the actual period of transfer is given and not the timing number 'n' which is used in the instruction word. An immediate instruction is written e.g. as

A + B → C imm. 10, N.

A + B → C imm. 40, N.

without direct reference to the characteristic. The timing number of the first of these instructions would be 9; while the timing number of the second of these instructions would be 40 - 33, i.e. 7 and the second value of the characteristic would be implied.

(3) The third value gives rise to what are known as 'deferred' instructions. A deferred instruction with timing number 'n' waits n minor cycles after it is staticised and then takes place for one minor cycle only. It is written e.g. as

A + B → C def n, N.

The next instruction is the one in INST at the end of the transfer. Such an instruction is used for picking out one word from a long tank.

(4) The fourth value gives rise to what are known as serial instructions. A serial instruction with timing number n waits for n minor cycles after it is staticised and then transfers for one minor cycle, but it differs from the other instructions in that the next instruction to be obeyed is not the one in INST at the end of the transfer, but the instruction from the tank selected by the next source of instruction which occupies the next minor cycle to the current serial instruction itself. Thus the instruction obeyed after an instruction occupying position 'm' given by

A + B → C ser n, N

is the instruction occupying position (m+1) in instruction tank no N and is therefore independent of n. A serial instruction is used to pick out a single

word/

word from a long tank whose position is not known in advance but will depend upon the course of the computation itself. The number n can be assigned during the course of a computation without effecting the instruction which follows the serial instruction. For each of the other three characteristics the position of the next instruction obeyed is directly dependent on the value, 'n', of the timing number.

The timing of the transfers, using timing number and characteristic is achieved in a very similar manner to the timing of instructions in the pilot model. Again the principal unit involved is the short tank COUNT and its associated half adder.

The use of two sources and one destination together with a function requires considerable extensions of the control however, and a schematic diagram and description of this part of the circuit is given (FIG.37).

The parts of the instruction which have to be staticised now, are the two sources, the destination, the function and the next source of instruction; corresponding to these elements of the instruction there are three trees of order eight, one of order four and one of order five respectively. Each of the sources is provided with two source gates the first of which may be opened by the tree corresponding to source 1 and the second by the tree corresponding to source 2. All the lines from the first set of gates pass into a common line called "HIGHWAY 1" and all the lines from the second source into a common line called "HIGHWAY 2". Each of these highways is gated by TRANSTIM and then passes into a set of function boxes which produce functions of the sequences of digits on the two lines. Thus whenever TRANSTIM is on, the function box marked BO will be producing the number HIGHWAY 1 + HIGHWAY 2 with carry suppression at the end of each minor cycle. Following each of the function boxes there is a gate which is controlled by

the/

FIG. 37

HIGHWAYS AND FUNCTION BOXES

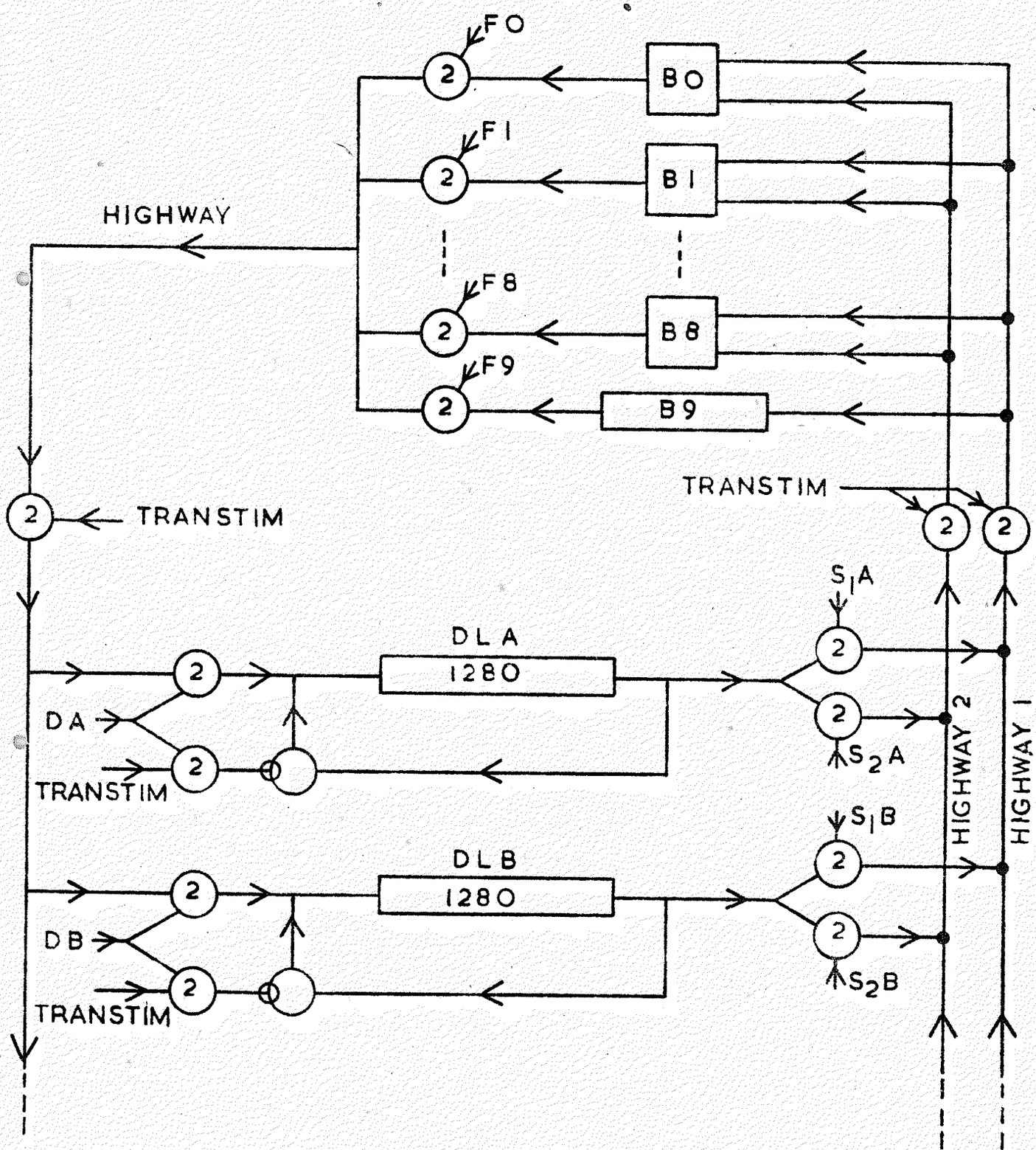
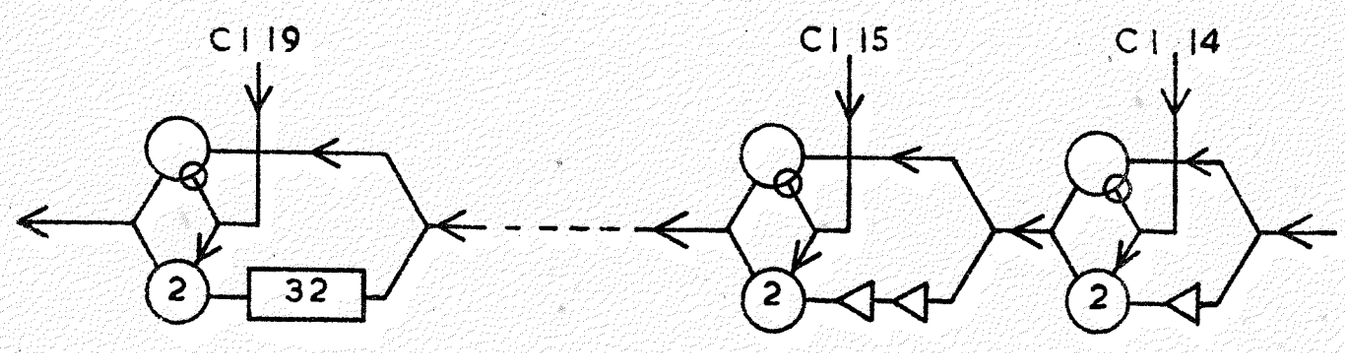


FIG. 38. DELAY UNIT



the tree associated with the function element of an instruction. The outputs of all the function gates are connected to a common line which is called HIGHWAY; this line also has a gate controlled by TRANSTIM. As in the pilot model all the destination gates are connected to HIGHWAY. The following functions are derived exactly in this way (there are also a numbers of functions which are derived somewhat differently).

0 SOURCE 1 + SOURCE 2 carry suppression at the end of all minor cycles.

1 SOURCE 1 + SOURCE 2 " " " " " " odd " "

2 SOURCE 1 + SOURCE 2 no carry suppression.

---

3 SOURCE 1 - SOURCE 2 carry suppression at the end of all minor cycles

4 SOURCE 1 - SOURCE 2 " " " " " " odd " "

5 SOURCE 1 - SOURCE 2 no carry suppression

---

6 SOURCE 1 V SOURCE 2

7 SOURCE 1 & SOURCE 2

8 SOURCE 1 ≢ SOURCE 2.

Although the six function boxes associated with addition and subtraction are shown separately, there will in fact be one adder only for all six functions. The particular line from the function tree which is being stimulated will modify the functioning of the adder to give addition or subtraction with the required type of carry suppression. The notation $+_A$, $+_O$, $+_N$ is used to denote addition with carry suppression at the end of all, odd or none.

It is convenient to be able to delay any sequence of pulses by a number of microseconds or even by a whole minor cycle. The delay facility is made one of the available functions but it is provided as a function of 'source one' only. An instruction ordering such a delay has the number of microseconds by which 'source one' is to be delayed, given in the position normally occupied by

the/

the second source. It is usually written in the form of which the following instruction is typical.

A delayed m→C, charact, timing no, next source of inst. The result of this transfer is that the sequence of digits available at A during the transfer is sent to destination C after suffering a "shift to the left of m places", the gap of m places at the least significant end being filled by m zeros. Assuming the delay facility is given as function 9, the method by which it is produced is shown in (Fig.37). It will be seen that HIGHWAY 1 only, passes into the delay network. The form of the delay network is shown in Fig.38; the delay which source 1 undergoes before passing to HIGHWAY, and thence to its destination, is determined by the tree associated with source 2. Although the tree associated with the second source sets up a delay in the delay network even when the current instruction is not ordering a delay, this is of no significance because in such cases the gate controlled by F 9 is not open. Since the delay element of an instruction occupies the same position as that normally occupied by the second source element, it has eight digits available and therefore could be made to select any delay between 0 and 255 units. In the examples in this report no delay of more than 40 units has been used and it is sufficient to assume that the delay uses only 6 digits, thereby providing delays between 0 and 63 units.

Multiplication and division. It is assumed in the instruction tables which follow, that the large scale A.C.E. will contain an automatic multiplier but no automatic divider; division will be programmed from the elementary operations of addition and subtraction. The automatic multiplier will provide the following facilities.

It will multiply two one-word numbers which may be regarded either as 40 digit positive numbers (unsigned multiplication) or as two 39 digit numbers with the usual sign convention (signed multiplication), according to which multiplication is required. The multiplier performs whichever of these types of multiplication

is/

is required and, when the multiplication is completed, the 80 digit product is added to an 80 digit accumulator which has carry suppression at the end of odd minor cycles. The timing necessary to determine when the multiplication is complete is performed by counting in the short tank COUNT in the digits which are not occupied for counting the transfer timing. Multiplication is ordered by instructions of the type.

A x_s B - imm 1, N

A x B - imm 1, N

A x B - def m, N

A x B - ser m, N etc.

In these instructions 'x_s' and 'x', which stand for signed and unsigned multiplication are two further possible functions; since the product is added into a double length accumulator the destination is redundant in a multiplication instruction. The multiplication functions route HIGHWAY 1 and HIGHWAY 2 to the multiplier as soon as TRANSTIM is on. It is assumed that the multiplier is of the same basic type as has been described fully before, i.e. it takes 2 minor cycles to multiply by each digit and therefore 3200 microsecs. for the complete multiplication. This may ultimately be regarded as too long, in which case the time of multiplication may be cut down by having two or more adders, but for the purpose of this report the full 3200 microseconds has been taken. To determine the position in instruction source N of the instruction following a multiplication, the position of the next instruction on the basis of an ordinary current instruction (i.e. one which does not order multiplication) and then 81 minor cycles is added (actually 17 is added because the result is only needed modulo 32). This allows time for the multiplication to be completed.

Special Facilities provided by coded instructions.

There are a number of features in the type of coded instruction used on the A.C.E. which are sufficiently different from any displayed in the coded instructions of other projected calculating machines, to make a brief survey

of/

of the type of facilities which our instruction provides worth while. The first important difference is that instruction do not occupy successive positions in the memory. The position of the instruction following each of the basic types of instruction is shown below. -

Instruction position a.	A FUNCTION B → C imm m N (1 m 64)
Next instruction is in position	(a + m+1) in N.
Instruction position a	A FUNCTION B → C def m N (0 m 31)
Next instruction is in position	(a + m+2) in N.
Instruction position a	A FUNCTION B → C ser m N (0 m 31)
Next instruction is in position	(a + 1) in N i.e. is independent of m.

If the function involved is multiplication 81 must be added to the position of the next instruction in all the above cases.

The feature of our instructions which makes them differ most markedly from other instructions is the fact that the immediate type of instruction may be used to transfer a number of consecutive words. By means of the instruction

$$A +_A B \rightarrow C \text{ imm } m, N$$

for example, where A, B and C correspond to long tanks, m consecutive words of tank A may be added to m consecutive words of B and the result sent as m consecutive words to C, i.e. the instruction gives vector addition of two vectors each with m components; the round carry at the end of all minor cycles ensures that the carry from the end of one component, (which may occur with positive and negative numbers even when each component of the vector sum lies in the required range), does not 'run into' the next component. If B is made equal to A the original vector is doubled. By using the accumulator the immediate instruction may be used to give a remarkable variety of results.

$A +_A B \rightarrow acc_+$ imm m, N, (The definations of acc_+ etc. are given in the last paragraph of this section) for example will take m consecutive numbers from delay lines A and B and add all 2m numbers into the accumulator. If A and B are short tanks then the same numbers are sent to the accumulator each

minor/

minor cycle and therefore m(A+B) is added to the accumulator. Hence the immediate instruction may be used for multiplication of the sum of two numbers in short tanks by integers m up to 64 in one instruction.

The instruction

$$\text{acc} +_A \text{zeros} \rightarrow \text{acc}_+ \quad \text{immed m, N}$$

has the effect of multiplying the number n the accumulator by 2^m giving the correct result for positive or negative numbers provided this result still lies in the permitted range. The explanation of this is that if the original number in the accumulator is 'a', at the end of the first minor cycle of the transfer it contains a + a, i.e. 2a, at the end of the second it contains 2a + 2a i.e. 4a and so on.

In a similar way the instruction

$$\text{acc} +_A \text{acc} \rightarrow \text{acc}_+ \quad \text{imm m, N}$$

has the effect of multiplying the number in the accumulator by 3^m, since its contents will be trebled for each minor cycle of the transfer. The function of addition with carry suppression at the end of odd minor cycles only, may be used in a similar way to deal with double length numbers. Thus vectors with double length components may be added (and, similarly, subtracted), numbers in the double length accumulator may be multiplied by powers of 3 etc. Another, and perhaps more important, property of the immediate instruction, is that it enables us to perform operations on numbers containing a large number of digits (up to 1280 in fact). Two long numbers in two long tanks may be added together, provided they are in phase, by using the addition with no carry suppression; the fact that when positive and negative numbers are added together there may be a carry from the last digit (i.e. the sign digit) is of no consequence because this carry is left behind in the adder when the transfer terminates and is lost during the period when the next instruction is being staticised. Such a carry digit can therefore do no harm.

The combined use of the delay facility and the accumulator gives a simple method of multiplying positive numbers by powers of $(2^n + 1)$.

The instruction

acc ▷ 4 → acc_+, imm n, N. (' ▷ ' is written for 'delayed')

multiplies the contents of the accumulator by $(2^4 + 1)^m$ i.e. by 17^m since each transfer produces $(2^4 \text{ acc}) + \text{acc}$.

The & and operator may be used for selecting particular digits of a word by using the P1 and P40 sources. Its most frequent use is for detecting the sign digit of a number. Its use may be illustrated by the example:-

A & P1 → acc_+ imm m N

This transfer will perform the & operation on m consecutive numbers of delay link A, and corresponding to each of them which has a 'one' in the P1 position (i.e. corresponding to each of them which is an odd number) a P1 will be sent to the accumulator. Hence the number added to the accumulator as the result of this instruction is equal to the number of odd numbers among the 'm' numbers examined. Such examples of the use of the instruction word could be multiplied indefinitely, but the above will suffice to illustrate the nature of the immediate type of instruction.

Index and notation for full-scale A.C.E.

Since the exact structure of the A.C.E. is not yet considered to be finally decided upon, the various facilities it provides have not yet been allotted definite source and destination numbers. To facilitate reading tables of instructions, the various sources and destinations used in the instructions, have been given an abbreviated code. This will be described below.

The/

The A.C.E. will contain:

COMPONENT	ABBREVIATED CODE.
(1) Approximately 200 long tanks	D.L.X. (Delay line). The n th word of D.L.X. will be denoted by X_n.
(ii) Approximately 32 short tanks	T.S.X. (Temporary storage)
(iii) A few triggers for remembering binary digits.	T.C.X.
(iv) A trigger DISCRIM.	DISC
(v) A short tank.INST.	INST
(vi) An accumulator of length 40.	When used as a source "acc". When used as a destination "acc" or "acc_+"according as number sent there replaces, or is added to current content.
(vii) An accumulator of length 80	When used as a source "Acc". When used as a destination "Acc" or "Acc_+"according a number sent there replaces or is added to current contents.
(viii) Odd and even impulse lines associated with input and output	O.I.L. (odd impulse line) E.I.L. (even impulse line)
(ix) Triggers READ and PUNCH	READ, PUNCH.
(x) Input Dynamiciser	I.D.
(xi) Output Staticiser	O.S.
(xii) One short tank for instructions.	T.S.i (Temporary Storage for instructions).

The characteristics imm, def, and ser. are abbreviated to i,d, and s in the later examples. The notation $D.L.X_n$ is used to denote the nth word in delay line X, it is frequently abbreviated to X_n when this will not lead to confusion.

The instruction is written in the form.

Source 1; function; Source 2; Destination; Characteristic; Timing number;

Next source of instruction number.

this enables the instruction to be read in the form in which it most readily conceived. The order of the elements of the instruction will not, however, be as written above. Since in a number of instruction tables given later, instructions have to be built up during the operation of the table it is necessary to know the true order of the elements of an instruction. They have been chosen to be

Timing/

Timing number, Source One, Source Two, Destination, Function, Characteristic, Next instruction Source, P1 to P5, P6 - P13, P14 - P21, P22 - P29, P30 - P33, P34 - P35, P36 - P40.

The timing number precedes the source number, so that we may pass naturally from the end of one long tank to the beginning of the next when we are using a serial instruction, by adding one repeatedly to the timing number. When the timing number passes 31, it adds one to Source one and itself reverts to zero.

6. Examples of Coding.

Problem 1.

The derivation of the squares and cubes of all integers from 1 to m.

This problem is treated in the simplest possible manner and is merely intended to show how an instruction table is constructed. The squares and cubes are derived successively, and each is destroyed when the next is formed. No binary decimal conversion is given since this is treated later.

General Plan of Table.

It is assumed that the integer m is given in TS1. In order to have all numbers as accessible as possible the values are stored in temporary storages. The numbers n, n^2, n^3 are obtained in TS2, TS3, TS4 respectively for successive values of n from 1 to m and $(n+1)$, $(n+1)^2$, $(n+1)^3$ are derived from them in the following steps:

Form $(n^2 + n)$ in TS3.
Form $n^3 + 3(n^2 + n)$ in TS4
Form $(n + 1)$ in TS2
Form $(n^2 + n) + (n + 1)$ i.e. $(n + 1)^2$ in TS 3.
Form $n^3 + 3(n^2 + n) + 1$ i.e. $(n + 1)^3$ in TS4.

This cycle of operations is performed repeatedly and after each repetition a discrimination is performed to see if the number in TS2 has become identical

with/

with that in TS1 i.e. to see if 'n' has reached 'm'. If it has not, the cycle is repeated, but if it has, the computation is complete and we move on to a new instruction table. In addition to the cycle of operation there are a few preparatory instructions which put 0, 0^2, 0^3, (i.e. zero) in each of the tanks TS2, TS3, TS4. This 'sets the stage' for the repeated cycle. It is assumed that all the instructions are in the long instruction tank A and occupy the positions given in the table. The time taken is 1 millisec per step.

A1	zeros+ zeros → TS2 Imm 1, A	Preliminary process for obtaining $0, 0^2, 0^3$ in TS2, TS3, TS4
A3	zeros+ zeros → TS3 Imm 1, A	
A5	zeros+ zeros → TS4 Imm 1, A	
A7	TS2 + TS3 → TS3 Imm 1, A	i.e. $n^2 + n \rightarrow$ TS3
A9	TS3 + TS4 → TS4 Imm 3, A	i.e. $n^3 + 3(n^2 + n) \rightarrow$ TS4
A13	TS2 + P1 → TS2 Imm 1, A	i.e. $n + 1 \rightarrow$ TS2
A15	TS3 + TS2 → TS3 Imm 1, A	i.e. $(n^2 + n) + (n + 1) \rightarrow$ TS3
A17	TS4 + P1 → TS4 Imm 1, A	i.e. $n^3 + 3(n^2 + n) + 1 \rightarrow$ TS4
A19	TS2 ≠ TS1 → DISC Imm 18, A	i.e. Discrimination on the equivalence of new 'n' and 'm'. If not equivalent, return to A7, otherwise go on to A6.
A6	END	

The only instruction which calls for any comment is A 19. The result of this instruction is that for each digit of TS2 and TS1 which differ, a one is sent to DISCRIM and therefore DISCRIM will be put on unless TS2 and TS3 contain identical numbers. The timing number of A19 has to be chosen so that if DISCRIM has been put on, the next instruction will be A7, i.e. the beginning of the repeated cycle; if DISCRIM has not been put on the next instruction is A6, which will start on a new problem.

Problem 2./

Problem 2.

The derivation of values of sin n d and cos n d for values of n from 1 to m. (It is assumed that $\frac{\pi}{2} > m\ d > \frac{\pi}{4}$ so that neither sin n d cos n d takes the value 1·0).

This problem is again treated quite simply and there is no section to deal with the binary-decimal conversion and the punching of results. Routines are given later which will show how this would be done. Temporary storages are again used for the numbers throughout. Since all the values obtained are numerically less than 1·0 the binary point may be placed between digits 39 and 40 without running into any difficulties which require the use of scale factors. It is assumed that sin d and cos d are given in TS1 and TS2 also with the binary point between digits 39 and 40. Sin n d and cos n d are derived successively and stored in TS3 and TS4. The next two values are determined from these by the formulae.

$$\sin (n + 1)\ d = \sin n\ d \cos d + \cos n\ d \sin d$$

$$\cos (n + 1)\ d = \cos n\ d \cos d - \sin n\ d \sin d$$

sin (n + 1) d is formed first and it is stored temporarily in TS6, because the value of sin n d is still required for forming cos (n + 1) d. When cos (n + 1) d has been formed and put in TS4, sin (n + 1) d is moved from TS6 to TS3. It is assumed that the number m is given in TS5 originally. Each time the cycle of operations deriving the (n + 1) d values from those for n d is performed, a P1 is subtracted from TS5. Since the table starts by putting the given values of sin d and cos d in TS3 and TS4, there are only (m - 1) cycles to be performed and hence the process is complete when the number in TS5 has been reduced to 1 i.e. when it is equivalent to P1. The test for the completion of the work is therefore done by discriminating whether or not TS5 is identical with P1. The formation of sin (n + 1) d and cos (n + 1) d are carried out in the double length accumulator. For the formation of sin (n + 1) d the process in detail is this:

The/

The Accumulator is cleared

sin n d cos d is sent to it as an 80 digit number with binary place between digits 78 and 79.

cos n d sin d is added to the accumulator to form sin (n + 1) d as an 80 digit number with binary place between 78 and 79. The answer is then shifted one place to left to give a number with the binary point between 79 and 80 and finally a P40 is added into the 40^{th} digit of this number to round off. The left hand half of the resulting number is then sin (n + 1) d regarded as a 40 digit number with binary point between 39 and 40. This half is sent to TS6.

The multiplication is given in full here to make the process quite clear; it should be noted that the two 80 digit numbers are added before they are rounded off. This procedure is adopted throughout these examples, for the summation of products. A point of general interest in table making should be mentioned here; insofar as sin d and cos d are not exactly representable as 39 digit binary numbers we will be using incorrect values for these two numbers, which will bring a systematic error into the calculated values of sin n d , cos n d , over and above the "random" rounding-off errors. It is therefore advisable to choose a value of d such that sin d and cos d will be particularly accurately representable to 39 binary digits. If e.g. d is chosen to be 2^{-19}

$$\sin d = 2^{-19} - \frac{1}{6} 2^{-57} + \text{etc.},$$

and would therefore be represented by 2^{-19} with an error of less than $\frac{1}{6} \times 2^{-57}$. Similarly we have

$$\cos d = 1 - \tfrac{1}{2} 2^{-38} + \frac{1}{24} 2^{-76} - \text{etc.}$$

and therefore cos d will be representable as $(1 - 2^{-39})$ with an error of less than $\frac{1}{24} 2^{-76}$. In this way the consistent error due to incorrect values of sin d and cos d is kept very low. When values of sin n d and cos nd are

required/

required for a small value of d which is to be a power of 10^{-1} it is better to use 2^a sin d and 2^b (1-cos d) where a and b are chosen, so that these values are still less than unity. The products obtained when forming sin (n+1) d and cos (n+1) d will then need an appropriate shift in order to obtain the binary point in the correct place. The general plan of the instruction table is given below, and the table itself follows. Each step of the computation takes 11 millisecs.

1. Send sin d to TS3 } Initial process
2. Send cos d to TS4 }

---

3. Clear Acc.
4. Form sin n d cos d and add to Acc.
5. Form cos n d sin d and add to Acc.
6. Correct position of binary point in Acc.
7. Round off result
8. Send sin (n + 1) d to T.S.6

Forming sin (n + 1) d in TS6

9. Clear Acc.
10. Form sin n d sin d and add to Acc.
11. Reverse sign of contents of Acc (-sin n d sin d)
12. Form cos n d cos d and add to Acc.
13. Correct position of binary point in Acc
14. Round off result
15. Send cos (n + 1) d to TS4
16. Send sin (n + 1) d to TS3

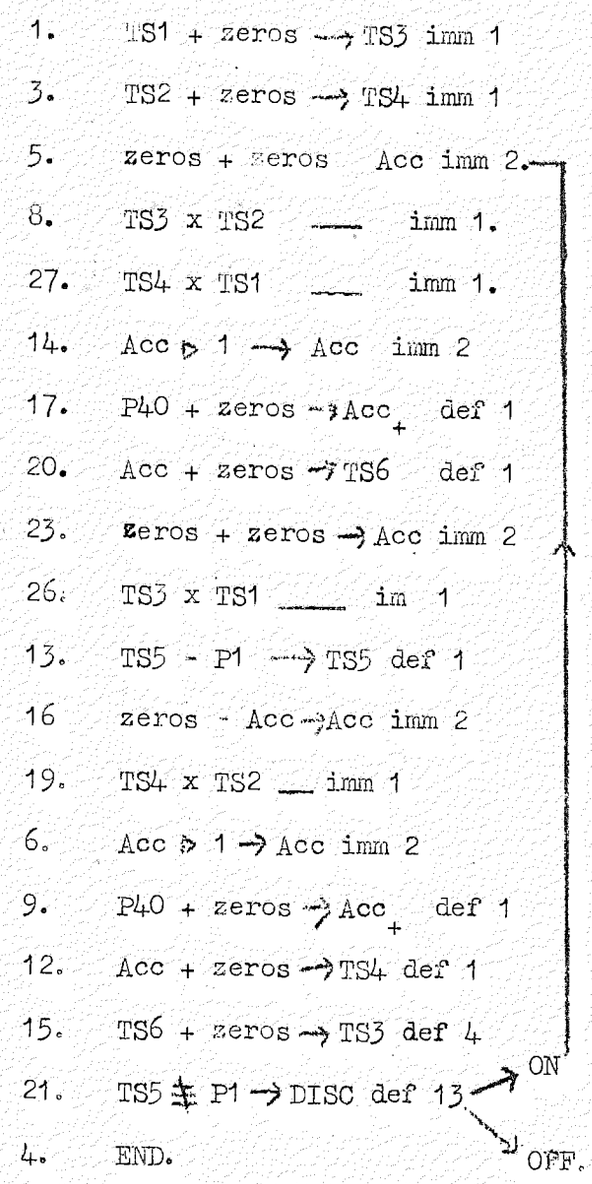
Forming cos (n + 1) d

17. Subtract one from TS5
18. Detect whether or not TS5 = P1. If so go to 19. If not return to 3.
19. End.

The table itself follows.

Table/

Table of sines and cosines.



Problem 3.

Derivation of the square root of an integer $< 2^{39}$ to 1000 binary digits.

This example is designed to illustrate the use of the logical control for dealing with numbers of a considerable number of digits. The restriction on the size of the integer could easily be removed, but, since the example is merely meant to be illustrative, it has been taken in its simplest possible form.

General description of the method.

It is clearly sufficient to consider the problem of finding the square root of numbers 'm' such that $2 > m \geqslant \frac{1}{2}$ expressed as numbers with 38 binary places. It is assumed that the problem has been reduced to this form, and that the number m is in the long tank DL A in the 1st position, i.e. at the beginning of a major cycle the position of the number in the tank will appear as in the figure.

zeros	zeros	zeros		zeros	m
31	30	29		1	0

The method of calculating the successive digits is essentially the binary equivalent of the standard method of "extracting square roots". It is given below for the square root of 2 for convenience.

```
                1.01101
               ---------
               10.
                1
 1001          -----
               10000
                1001
 10101         -----
                11100
                10101
  1011001      -------
                 1110000
                 1011001
   1011010      --------
                  1011100
```

Ignoring all reference to binary points the process may be described as follows. Let us suppose that n digits of the square root have been extracted, and let u_n be the integer represented by these digits and r_n the remainder. To obtain the next digit, we form the numbers $4r_n$ and $4u_n$. We then form $4u_n+1$ and if this

is/

is less than or equal to $4r_n$ the next digit is a 1 otherwise it is a zero. u_{n+1} will be $2u_n + 1$ or $2u_n + 0$ respectively in these two cases.

This process is programmed as follows.

r_n is stored in D.L. A $(r_0 = m)$

u_n is stored in D.L. A+2 $(u_0 = 0)$

$4u_n$ is derived in D.L. A+1

The single digit P39 is stored in the first position of D.L. A+3 i.e. $(A+3)_0$ and is used for forming $4u_n + 1$ and $2u_n + 1$ from $4u_n$ and $2u_n$ respectively. Since we are determining the square root to 1000 binary places only, none of the intermediate values obtained in the computation will occupy more than 26 minor cycles. Hence for positive numbers all the long tanks will have 6 words at least consisting entirely of zeros. In order to determine whether or not $4u_n + 1$ is greater than or equal to $4r_n$, a subtraction will be performed. In the cases where the result is a negative number the last six words in the long tank containing this number will consist entirely of 1's and hence any of these digits could be used as a sign digit. In most additions (and subtractions), the addition with no carry suppression must be used, and the transfers involving them must obviously start exactly at the least significant end of the number and terminate exactly at the most significant end. However in forming $4u_n + 1$ or $2u_n + 1$ from $4u_n$, $2u_n$ and the tank containing the single P39 any form of addition may be used and only the minor cycle containing the P39 need be covered by the transfer since there cannot be any carry overs. In fact this addition is performed in the table by the use of the logical operations of the type (A+1) v (A+3), since this operation gives us the greatest freedom. Even if the transfer were of such length that it went through the minor cycle in which the P39 is contained twice, the result would still be to add the two numbers together correctly since the repetition of the logical operations produces no further result! The operation 'or' can be used to replace addition because the P39 is added to a zero, The discrimation on the sign of $4r_n - (4u_n + 1)$ is performed by sending A & P40 to DISCRIM in minor cycle 29; the effect of this

is/

is that DISCRIM is put on if the 40th digit of DLA_{29} is a 1. This will only be true if $4r_n - (4u_n + 1)$ is negative. Although the table contains few instructions it is necessary to use more than one instruction storage tank because a number of instructions have to be placed in the same minor cycle. The storage tanks for the instruction are called B, C and D. It is assumed that the number 1,0000 in binary form is given in TS1. Each time a digit of the square root is determined a P1 is subtracted from TS1 and the discrimination for the completion of the process is therefore performed by testing whether or not the contents of TS1 have become zero.

TABLE.

B1	zeros + zeros → A + 3	imm 32,B	Clears D.L. A+3
B2.	P1 ▷ 38 → A + 3	dif 28,B	Puts P_{39} in $(A+3)_0$
B0	zeros + zeros → A + 1	imm 46,B	Clears D.L. A+1
B15	zeros + zeros → A + 2	imm 45,B	Clears D.L. A+2 i.e. supplies $u_o = 0$.
B29	A + 2 ▷ 1 → A + 2	imm 32,B	Replaces u_n by $2u_n$ in D.L. A+2
B30	A + 2 ▷ 1 → A + 1	imm 32,B	Send 2 x $2u_n$ i.e. $4u_n$ to D.L. A+1
B31	(A + 1) v (A + 3) → A + 1	imm 30,C	Forms $4u_n + 1$ in D.L. (A + 1)
C30	A - $_N$ (A + 1) → A	imm 32,C	Subtracts $4u_n + 1$ from $4r_n$ in D.L. A.
C31	A & P40 → DISCRIM	def 28,D	This discriminates on the 40th digit of D.L. A_{29} if it is a 0 the next instruction is D29, if a 1 next instruction is D30.
D29	(A + 2) v (A + 3) → A + 2	imm 33,D	forms $2u_n + 1$ in A + 2
D30	A + $_N$ A + 1 → A	imm 32,D	Reforms $4r_n$ in A
D31	TS1 - $_A$ P1 → TS1	def 26,D	Subtracts 1 from T.S.1.
D27	A ▷ 2 → A	imm 32,D	Form $4r_{n+1}$ in D.L. A.
D28	1 + zeros → DISCRIM	imm 31,B	Discriminates of content of T.S.1. If it is not zero the next instruction is B_{29} (i.e. beginning of cycle). If it is zero the next instruction is B28 and the process is complete.
B28.	End.		

It should be noted that the first step in the cycle is to double u_n. It is permissible to perform this step first because on the first occasion we enter the cycle $u_n = u_o = 0$.

The time taken to determine the thousand digits is about $11\frac{1}{2}$ secs.

Problems 1, 2 and 3 give rise to self contained instruction tables but this is only because they have been constructed in an artificial manner. The simple problem of forming squares and cubes of numbers would require some form of output table, possibly incorporating binary decimal conversion, so that the results could be obtained on punched cards or printed. The table for binary decimal conversion and punching would, in fact, be needed for almost any problem and it would therefore be convenient to have a standard routine for dealing with this problem. If we regard this routine as a simple table which will take a given binary integer and convert it into a decimal integer in the form required by the Hollerith equipment, then before any table could make use of this routine it would have to provide the routine with the binary number to be converted, and also with an instruction which ensured that the control returned to the correct instruction position when it had finished using the routine. The first of these requirements has been called 'a parameter' of the routine, and the second the 'Link' instruction. Another example of the use of a standard routine is provided by the problem of multiplying two square matrices or order m. This problem may be treated by dividing it up into the problem of finding m^2 scalar products of vectors of order m. If we have a routine which forms the scalar products of two vectors, then the programming of the matrix multiplication could be reduced to using the routine for scalar products m^2 times. The routine for scalar product would need a number of parameters which together described the position of the two vectors which were to be used and the order of these two vectors. The function of the main matrix multiplication table would then consist of repeatedly supplying the scalar product routine with the correct parameters, and a link to bring the control back to the main table. A little experience in programming soon reveals that one of the main problems to be solved in programming is that of the general organisation of these routines and the method of employing them. A simple method of organisation, which suffices for quite complex problems, is described in the following section of the report.

Elementary/

Elementary Organisation of the use of sub-routines.

In this method of organisation the parameters and link used by any routine are stored in fixed temporary storages. Let us suppose that a simple routine, which itself makes no use of sub-routines (such a routine might be called a routine of the 1st order) has m parameters and uses n temporary storages for other purposes; then its parameters will be stored in TS1, TS2, ... TSm and the link will be placed in TS(m + n + 1).

Before such a routine can be used the particular values assumed by the parameters must be put in TS1 to TSm, the link in TS(m + n + 1), and the control then directed to the first instruction (usually known as the 'Entry') of the routine. The routine is then carried out; the last instruction of each routine is an order transferring the Link instruction to the Instruction temporary storage T.Si and making the instruction in T.S.i. the next instruction to be obeyed. This brings us back to the original table which made use of the routine. It is convenient to have routines which themselves make use of subroutines. Such routines are organised as follows. We first make a general plan of the instructions which the routine will require and determine which subroutines it will use. Suppose the highest temporary storage used by any of the subroutines is TSn, then the parameters of the routine are placed in TS(n+1), TS(n+2)...TS(n+n_1) where n_1 is the total number of its own parameters. The link for the main routine is put in the TS next higher to the highest T.S used for all other purposes. It will be seen that the routine itself organises the placing of the parameters and links in any subroutine which it uses. It will now be realised why it was necessary to make the final instruction of any routine an order placing the link for that routine in T.S.i. It is not possible to place the link directly in T.S.i. before the routine has been used, because that routine might wish to use T.Si itself or may have a subroutine of its own in which case it would lose its own link when it provided the subroutine with a link. A routine which uses subroutines of the first order only may be called a routine of the second order. Similarly we may have routines of any higher order.

Each/

Each of the routines used in the A.C.E. is given a name, usually of five or six letters which gives a brief indication of the function which it performs. Thus the routine MATMUL might perform matrix multiplication, and SOLVEQ might solve a set of linear algebraic equations. In order to make the organisation of a complex computing problem simpler, a record of each routine is kept, which gives a brief description of the function it performs, lists its subroutines if any, lists its parameters and the temporary storages they occupy and gives the T.S. occupied by the LINK and the minor cycle in which it is obeyed. The routines will each occupy a definite position in the memory used for storing instructions. Since not all the routines will be required in every problem it will be convenient to arrange the position of routines in the memory so that all routines of a similar type are fairly close together. The main table for a fairly complex problem would consist almost entirely of a set of instructions whose function was to supply the various routines used with the appropriate parameters and links and to keep account of the position reached in the computation. If the storage space allotted to instructions happened to be inadequate for all the instructions of the main table, then the surplus instructions would have to be stored in the remaining part of the memory and from time to time an instruction would be needed to feed these external instructions into the instruction delay lines. For very large problems it might even be necessary to keep some instructions on the Hollerith cards temporarily.

Using the above method of organising routines a computational problem is reduced to the following steps.

(i) Make a general plan of the computation to determine which of the standard routines it will employ.

(ii) Consult the record of routines to find out where the parameters and links in these routines are kept, and the storage the instructions they use.

(iii) Construct the main table and punch it and any data (tables etc.) needed by it on a set of cards.

(iv) Collect/

(iv) Collect together the punched cards corresponding to each of the routines used and all their subroutines (these subroutines will be listed in the record) and assemble these cards with the pack for the main table.

(v) Supply the 'input table' with the necessary parameters for taking in this complete pack of cards and the link to take it to the entry of the main table.

Routine ININT. (Input of integers).

This Routine gives the standard table of instruction for taking in a number, punched on a Hollerith card in decimal form, and converting it into a binary number. The input table may either perform the conversion, regarding the number on the card as an integer (or a number with its decimal point in some fixed position) or it may have a parameter which gives the position of the decimal point in the number. If the table performs the former conversion then it may be necessary to multiply the binary number it produces by a factor after using the table; if it performs the latter it will merely be necessary to supply the table with the appropriate parameter before use. To illustrate both these methods the table which performs the input and decimal-binary conversion takes the former course and regards the number on the card as an integer. The table which does the binary-decimal conversion and the output of numbers takes the second course and has a parameter which deals with the decimal point. It should be noted that for many linear problems the position of the decimal point in the original data is not important (e.g. the coefficient in a set of linear equations may be multiplied by a power of 10 without altering the solutions) and therefore the table ININT will suffice for the input of the numbers. In nearly all cases the position of the decimal point in the solution will be really significant. It might quite well happen, therefore, that it will be most economical to have the two tables in the two different forms as given.

The function of the table is to read in a signed decimal integer $\leq 2^{39}-1$ (549,755,813,887) from a Hollerith card, and convert it to binary form. The table occupies D.L.B and D.L.C and makes use of DLA. It uses as temporary

storage/

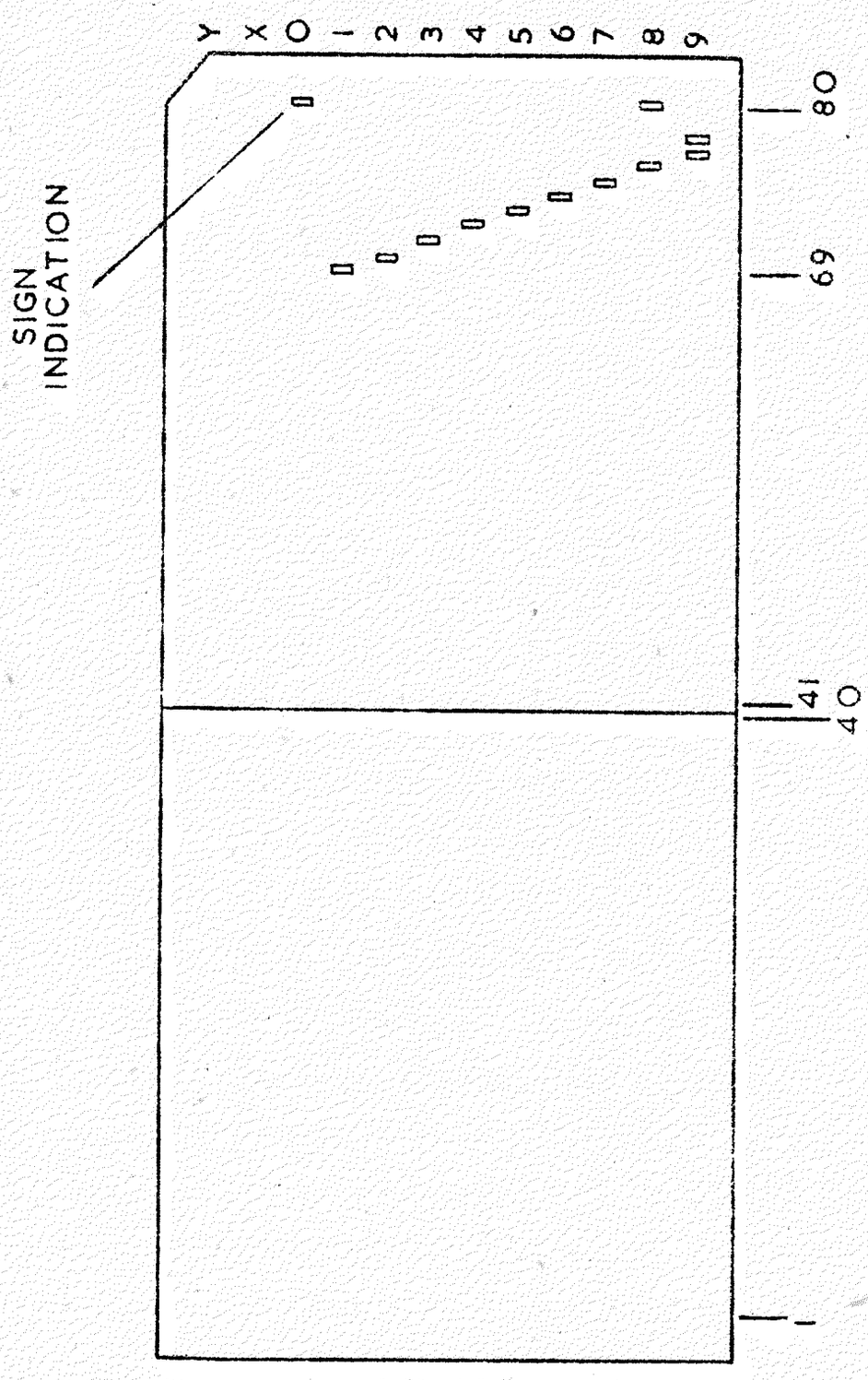
storage TS's 1 to 7. The link is in TS8. The converted number is obtained in T.S.6.

The decimal number is punched on the Hollerith card in columns 69 to 80 i.e. it is regarded as a 12 digit decimal integer, smaller numbers being made up to this size by zeros in the more significant positions. The most significant digit is punched in column 69 and the least significant in column 80. The card is read into the machine, nines first, columns 41 to 80 corresponding to P1 to P40 respectively. The zeros of a number are not punched but a hole is punched in some position in the 0 row if the number is negative. A card punched with the number -123456789908 is shown in fig 39 for illustration.

The table organisesthe intake of the ten rows of the card which are significant, by means of a cycle which is repeated five times. Each time this cycle is repeated, an odd row and an even row are fed into the machine and stored in A_2 and A_1 respectively. The method of keeping count of the number of times the cycle is completed is the following. Before the cycle is performed for the first time a P36 is put in TS7. At the end of each cycle a test is made to see if the digit in TS7 is a P40. If it is not, TS7 is delayed one microsecond, the contents of DL A are shifted back two minor cycles and then the cycle is repeated; if it is a P40 then the above cycle must have been performed five times (since the check is made before TS7 is delayed by 1 μs) and therefore all ten rows of the card have been read. The ten rows of the card will therefore be stored in A_1 to A_{10} the nines row being in A_{10} and the sign row (or 0 row) in A_1.

It is evident then that A_r $(2 \leqslant r \leqslant 10)$ contains a one in each of the positions where the decimal integer, as represented on the Hollerith card, contained a digit $(r-1)$. If a_{ir} is defined as 1 if the i^{th} decimal digit of the number is an "r", and as 0 otherwise, then if N is the integer to be converted.

FIG. 39.



HOLLERITH CARD

$$|N| = \sum_{r=1}^{9} r \sum_{i=1}^{12} a_{ir} 10^{12-i}$$

The method of conversion is to form $S_r = \sum_{i=1}^{12} a_{ir} 10^{12-i}$ for

$r = 9.8,\ldots 1$

in succession in TS3,

to form S_9, $S_9 + S_8$, $S_9 + S_8 + S_7$, ... , $S_9 + S_8 + S_7 + \ldots + S_1$

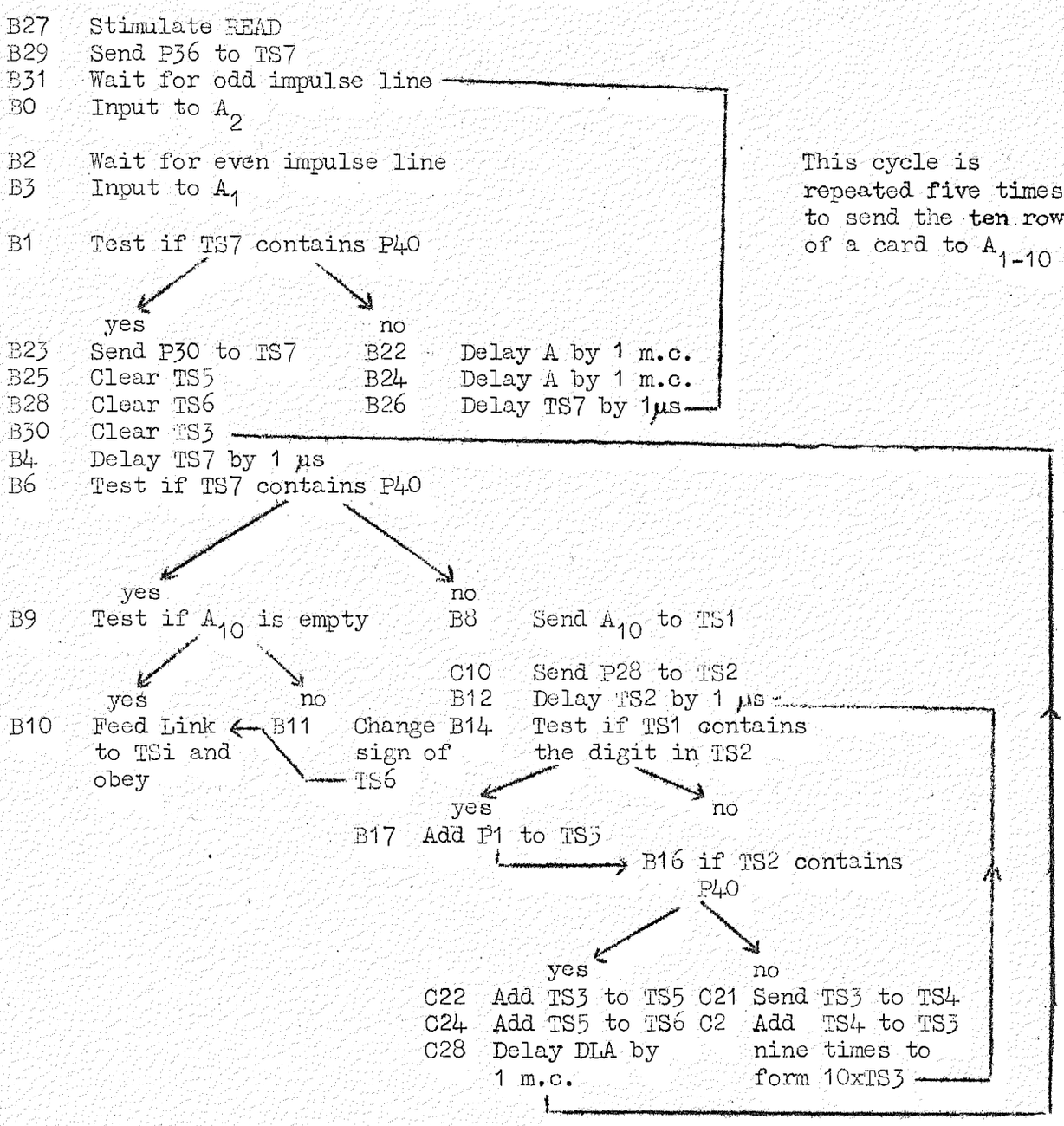
in succession in TS5

and to form S_9, $2S_9 + S_8$, $3S_9 + 2S_8 + S_7$, ... , $9S_9 + 8S_8 + \ldots + S_1$

in succession in TS6. The final value obtained in TS6 is therefore $|N|$. The value N is obtained by discriminating on whether A_1 is empty or not. The details of the formation of each of the above sums are as shown.

The/

The General Plan of the table is given below.



The table is given in full below. A number of abbreviations are used as follows. TSn is given as 'n', in the source and destination element.

The source of zeros is given as 0's.

Immediate, deferred and serial are written as i, d and s respectively.

"A delayed n microseconds" is written A ▷ n.

At the end of each instruction, in addition to giving the number of the

delay/

delay line containing the next instructions the position of the instruction in that delay line is given in brackets. This is to facilitate following the table through; the actual position in the delay line will not be part of the true instructions, but is, of course, determined by the position of the current instruction, its characteristic and its timing number.

ROUTINE ININT.

	D.L.B.	D.L.C.
1	7 & P40 → DISC. d 19 B(22 or 23)	
2	EIL + 0's → DISC. i 31 B(2 or 3)	4 + 3 → 3 i 9 B(12)
3	ID + 0's → A d 28 B(1)	
4	7 ▷ 1 → 7 i 1 B(6)	
5		
6	7 & P40 → DISC i 1 B(8 or 9)	
7		
8	A + 0's → 1 i 1 C(10)	
9	A + 0's → DISC d 31 B(10 or 11)	
10	8 + 0's → TSI i 1 TSI (LINK)	P1 ▷ 27 → 2 i 1 B(12)
11	0's - 6 → 6 d 29 B(10)	
12	2 ▷ 1 → 2 i 1 B(14)	
13		
14	1 & 2 → DISC i 1 B(16 or 17)	
15		
16	2 & P40 → DISC i 4 C(21 or 22)	
17	P1 + 3 → 3 d 29 B(16)	
18		
19		
20		
21		
22	A ▷ 40 → A i 33 B(24)	3 + 0's → 4 d 11 C(2)
23	P1 ▷ 29 → 7 i 1 B(25)	3 + 5 → 5 i 1 C(24)
24	A ▷ 40 → A i 33 B(26)	
25	0s + 0's → 5 d 1 B(28)	5 + 6 → 6 d 2 C(28)
26	7 ▷ 1 → 7 d 3 B(31)	
27	P1 + 0's → READ i 1 B(29) ENTRY.	
28	0's + 0's → 6 i 1 B(30)	A ▷ 40 → A i 33 B(30)
29	P1 ▷ 35 → 7 i 1 B(31)	
30	0's + 0's → 3 d 4 B(4)	
31	OIL + 0's → DISC i 31 B(31 or 0)	
0	ID + 0's → A i 1 B(2)	

Routine/

Routine NUMOUT (Output of numbers).

This routine gives the table of instructions for converting a number in binary form into decimal form, and punching out twelve decimal digits of this form. The table is supplied with a parameter which deals with any scale factor which the number may contain.

The detailed description of the function of the table is as follows.

Given a signed number x. in TS1, a number y in TS2, which will be a power of 10, multiplied by a power of 2 (either powers being positive or negative), and a number d in TS3, the table finds the integer nearest to $\frac{xy}{2^d}$, converts it to the decimal form required for the Hollerith and punches it on a card. The decimal number is punched in columns 69 to 80 and the 0 row has a hole in the column 80 if the number is negative. By a suitable choice of y and d we can deal with any scale factor in the number x. In problems where the position of the binary point is fixed throughout, the choice of y and d will be made when the programming of the main problem is done; if however, the binary point is adjusted continuously in the problem then it will be part of the function of the main table to determine suitable values of y and d.

Description of the method.

The table forms the product xy in the 80 digit accumulator, and then by means of an instruction which is built up from the number 'd', the contents of the accumulator are shifted (40-d) positions to the left. A 'round off' P40 is then added to the accumulator to give the nearest integer to $\frac{xy}{2^d}$ (x and y being treated as though they were integers in this expression) in the most significant half of the accumulator. This integer is placed in TS1. The process of finding the successive decimal digits of this integer, is that of subtracting 10^{12} from it repeatedly until it becomes negative, restoring after the final subtraction, multiplying the remainder by 10 and

then/

then repeating the process. The successive digits are determined by counting the number of subtractions necessary at each stage. One of the advantages of the table is that it always gives the answer exactly if the original number x were an integer since in this case y may be taken as 2^{38} and d = 38; (y may in fact be taken to be 2^n and d = n for any value of n less than 39) the round off will produce no carry. It has been made a general principle of our routines that they should always produce an "exact" answer where such an answer exists; this is convenient if the machine is working with integers on problems connected with Number Theory. The twelve decimal digits are stored in DLA_{2-10} in the following form; the digit in the P_n ($29 \leqslant n \leqslant 40$) position of A_r, is a 1 if the (n-28)th digit of the decimal number is (r-1); This is exactly the form required by the Hollerith equipment. The tables uses TS1, TS2 and TS3 for its parameters and TS4 and TS5 for other purposes. Its link is therefore in TS6.

An interesting feature of the table is that it provides examples of the need for constructing instructions during the course of a table. Thus instruction B.1. is used to build up the instruction

Acc ▷ 40-d → Acc i 2 B from the instruction

Acc ▷ 40 → Acc i 2 B which is stored in B.3. The constructed order is obeyed immediately after its formation. Similarly the instruction C28 is used to build up the instruction A + 3→A ser i C from the instruction A + 3→A ser 0 C and the number i, the latter having been derived during the course of the table's operation.

The general plan of the instruction table and its detailed coding are given below.

General Plan of Instructions.

B26 Send 10^{12} to TS4
B29 Delay TS3, i.e. 'd', by 13 μs (puts d in the position occupied by the delay element of an instruction)
B31 Clear DLA (1 to 10)
B10 Detect sign of TS1, i.e. 'x', and put P40 in TS5 if x is negative,
B12 Discriminate on content of TS5. 0 if x is non-negative.

TS5=0

B14 Multiply x by y
B1 Build up the instruction (Acc ▷ 40-d → Acc i 2 B) from B3 and put in B4
B4 Obey this instruction, i.e. move Acc (40-d) to left
B7 Round off number in Acc
C10 Send left hand half of Acc to TS1
B13 Put P28 in TS3
C15 Clear TS2

TS5=P40

B15 Change sign of number in TS1

B17 Test if TS3 contains P40

yes

B20 Stimulate PUNCH
¥B22 Send number '4' to TS2
B27 Send TS5 to A_1, i.e. put P40 in A_1 if x is negative
C1 Wait for odd impulse line
C2 A_{10} to output
C3 Wait for even impulse line
C4 A_9 to output
C9 Test if TS2 contains 0

no

C12 Delay DLA by 1 m.c.
C14 Delay DLA by 1 m.c.
C16 Subtract 1 from TS2.

yes

C11 Send LINK, i.e. TS6, to TS1

no

B19 Delay TS3 by 1 μs
B21 Subtract 10^{12} from TS1
B23 Test if No. in TS1 is negative

yes

C26 Add 10^{12} to TS1
C28 Build up the instruction A+3 → A ser n C where n is the value of the digit which has just been determined and send this instruction to C31

C31 Obey this instruction. This will put a P(28+i) in A(n+1) if the ith decimal digit of the number is an 'n'
C0 Clear acc (length 40)
B2 Send TS1 to acc 10 times, i.e. multiply TS1 by 10
C13 Send acc to TS1

no

C25 Add P1 to TS2

¥ by adding P1 to TS2 for four minor cycles; TS2 contains zero before this instruction. The number '4' might equally well have been obtained by delaying P1 by 2 microsecs.

NUMOUT.

	D.L.B.				D.L.C.			
1	INST - 3 →DLB	d	1	B(4)	OIL + 0's →DISC	i	31	C (1 or 2)
2	1 + 0's →ac	i	10	C(13)	A + 0's→ OUT	s	6	C (3)
3	Acc ▷ 40 →Acc	i	2	B(7)	EIL + 0's→ DISC	i	31	C (3 or 4)
4					A + 0's→ OUT	d	3	C (9)
5								
6								
7	P40 + 0's →Acc	d	1	C(10)				
8								
9					2 + 0's → DISC	i	1	C (11 or 12)
10	1 & P40 →5	i	1	B(12)	Acc + 0's →1	d	1	B (13)
11					6 + 0's →TSI	i	1	TSI (LINK)
12	5 + 0's→ DISC	i	1	B(14 or 15)	A ▷ 40 →A	i	33	C(14)
13	P1 ▷ 27 → 3	i	1	C(15)	ac + 0's → 1	i	1	C(15)
14	1 x 2 -	i	1	B(1)	A ▷ 40 → A	i	33	C(16)
15	0's - 1→1	d	29	B(14)	0's + 0's→2	i	1	B(17)
16					2 - P1 →2	d	15	B(1)
17	3 & P40 →DISC	i	1	B(19 or 20)				
18								
19	3 ▷ 1 → 3	i	1	B(21)				
20	P1 + 0's →PUNCH	i	1	B(22)				
21	1 - 4→1	i	1	C(23)				
22	P1 + 2 → 2	i	4	B(27)				
23	0's + 0's→Acc	i	2	B(26)=ENTRY	1 & P40 → DISC	i	1	C(25 or 26)
24								
25					P1 + 2 → 2	d	26	B(21)
26	INST + 0's → 4	d	1	B(29)	1 + 4 →1	i	1	C(28)
27	5 + 0's → A	d	4	C(1)				
28	(10^{12})				INST + 2→D.L.C.	d	1	C(31)
29	3 r→13 → 3	i	1	B(31)				
30					A + 3 →A	s	0	C(0)
31	0's + 0's→A	i	10	B(10)				
0					0's + 0's→ac	i	1	B(2)

The/

The final section of this report gives the coding corresponding to the solution of sets of linear algebraic simultaneous equations by two different methods. The standard routines used by the instruction tables for these problems are given first. For convenience of reference in this report, the routines and main tables have been placed in the instruction delay lines DL1, DL2, DL3, etc., in the order in which they are derived. These positions do not therefore represent their permanent storage positions; these would be determined when a more complete set of standard routines had been worked out.

ROUTINE SCAPRO (scalar product).

Given the parameters X,Y,r and s and a link instruction in TS1, TS2, TS3, TS4 and TS5 respectively, the table produces the scalar product of order (r+1) of the vectors in D.L.X positions 0 to r and D.L.Y positions 0 to r in the two word accumulator, shifts this scalar product s places to the left, and places the 40 most significant digits (correctly rounded off) of the resulting number in TS4.

Restrictions on the parameters. The parameter s, which is used to adjust the binary point, must lie in the range $0 \leqslant s \leqslant 39$. The parameter r which corresponds to a scalar product of order (r+1) may take any positive integral value $\leqslant 31$. The two vectors must be in phase. These restrictions could, of course, be removed by a slight modification of the routine.

This table could run into difficulties if any of the partial sums obtained during the formation of the scalar product were too large for the accumulator, or if the final value were too large to permit a shift of s places to the left. The table has been programmed to give a "failure indication" in either of these cases. Where a 'failure indication' has been obtained we could proceed in two ways according to whether we were considering a continuous adjustment of the binary point or not. If a continuous adjustment was intended then, all numbers would be shifted an appropriate number of places to the right and some record would be kept of this adjustment. Otherwise, a failure indication would imply that over preliminary mathematical planning, which should have ensured that numbers did not "grow out of range" had been at fault; in this case human

intervention/

intervention would be needed and the failure bell should be rung to indicate this. A convenient method of dealing with such a failure would be to have a standard FAILURE routine which had as its parameter, a code word which indicated the nature of the failure. The function of the FAILURE routine would be to punch out this code word on a card, and then go into a repetitive cycle in which, for example, it might wait for a signal from the odd impulse line and, when it finally received one, read in the row of a card and obey it. The machine would therefore remain in a state of 'suspended animation' until the operator started a card feed and fed in an appropriate instruction on a card. For this second course, the table which had given the failure indication would have to provide FAILURE with an appropriate code word in its parameter position, and then direct the control to the 'entry' of FAILURE. In the routine SCAPRO each failure gives rise to an instruction "Enter failure table". These instructions would have to provide FAILURE with the appropriate parameters corresponding to the failure indication and these parameters would be 'planted' in the table SCAPRO itself. They are omitted in the table given below.

The process consists of clearing the two word accumulator and then forming $X_n \times Y_n$ for $n = 0,1,2,\ldots,$ r and adding them into this accumulator. To check when the process is complete, TS3 is tested to see if it contains zero after each product has been formed. If it does not, P1 is subtracted from it and a further product formed. If it does, all (r + 1) products have been formed. The final value in the accumulator is "rounded off" in the correct position (i.e. by a 1 in position 41 + s from the left), shifted s places to the left, and the more significant half sent to TS4. To check for size the partial sums obtained in the accumulator are subjected to a test to discover if they lie in the range.

$$-2^{78} \text{ to } 2^{78} - 1.$$

If they do not a failure indication is given. They lie in this range if and only if the two most significant digits in the accumulator are the same. If the partial sum lies in the given range then after adding the next product,

the/

the value in the accumulator will still lie within the range permitted by the convention viz. -2^{79} to $2^{79}-1$°, if x is the partial sum and a is the next product we have

$$-2^{78} \leq x \leq 2^{78}-1$$

$$-2^{39}(2^{39}-1) \leq a \leq 2^{39}\ 2^{39}$$

and hence $-2^{79} + 2^{39} \leq a + x \leq 2^{79} - 1$

i.e. a + x is in the permitted range. The test is unnecessarily stringent, but it requires many more instructions to give an exact test. Similarly before the final value is moved up s places a check is made to see if this will produce a number outside the permitted range.

General plan of tube.

1.7 Clear Accumulator

1.10 Put X in 1st source position in TS1.

1.14 Put Y in 2nd source position in TS2.

1.18 Form number with X in 1st source position, Y in 2nd source position in TS1.

1.25 Put P_{39} in 2.

1.27 Form the instruction (X x_s Y - s m 1(16)) from 1_{29} (where m will take the value 0, 1, 2, ... r for successive passages through the cycle) and place this instruction in 1_{30}

1.30 Obey the instruction formed by 1_{27} i.e. form X_m x Y_m and add to the Accumulator.

1.16 Send most significant digit of Acc to P40 position of TS5

1.19 Send second most significant digit of Acc to P39 position of TS6

1.21 Delay TS6 by 1 μs.

1.23 Test equivalence of TS5 and TS6. If equivalent go to 2_{28} otherwise 2_{29}.

2.28 Test if TS3 contains zero, If so go to 1_{31}, otherwise 1_0.

2.29 Enter failure.

1.0 Add 1 to TS1

1.2 Subtract 1 from TS3 and return to 1_{27}

1.31/ Delay

1.31 Delay TS4 by 13 μs. i.e. put 's' in 'delay' position.

1.1 Form the instruction ($P_1 \triangleright 39 - s \rightarrow 1$ i 1 1(16)) from 1_3 and put it in 1_4.

1.4 Obey instruction formed by 1_1 i.e. put P_{40-s} in T.S.1.

1.6 Add this P_{40-s} to Acc to produce "round off".

1.8 Test sign of number in Acc. If + ve go to 1_{11}, if -ve go to 1_{12}

1.11 Send $Acc_1 - P_{40-s}$ to TS1 and go to 1_{13})
1.12 Send $Acc_1 + P_{40-s}$ to TS1 and go to 1_{15}) Acc_1 is the more significant
1.13 Send TS1 + P_{40} to TS1 and go to 1_{15}) half of the Accumulator.

1.15 Test if no in TS1 contains a P_{40}. If so go to 2_{18} otherwise 2_{17} (The above instructions are to test if it is permissable to shift the Acc 's' places to the left)

2.18 Enter FAILURE.

2.17 Form the instruction (Acc $\triangleright$ s $\rightarrow$ Acc i 2 2(25)) from 2_{21} and send it to 2_{22}

2.22 Obey instruction formed by 2_{17} i.e. delay Acc s μs.

2.25 Send 40 most significant digits of Acc to TS4.

2.27 Send link to TSi and obey it next.

It will be seen that the complete scalar product and round off is in the accumulator when SCAPRO is completed, and the rounded off value is in TS4 so that either is available.

In the following tables instructions which are not used as they stand, but are used in the formation of instructions, are given in brackets. The positions in the instruction delay lines in which such manufactured instructions are placed are denoted by a dotted line.

SCAPRO.

	DL.1.	DL.2
0	1 + P_1 → 1 i 1 1(2)	
1	INST - 4 → DL 1 d 1 1(4)	
2	3 - P_1 → 3 d 23 1(27)	
3	(P_1 ▷ 39 → 1 i 1 1(6))	
4	----------	
5		
6	1 + 0's → Acc_+ i 1 1(8)	
7	0's + 0's → Acc i 2 1(10) ENTRY	
8	Acc & P40 → DISC d 1 1($^{11}_{12}$)	
9		
10	1 ▷ 5 → 1 d 2 1(14)	
11	Acc - 1 → 1 i 1 1 (13)	
12	Acc + 1 → 1 d 1 1 (15)	
13	P_{40} + 1 → 1 i 1 1(15)	
14	2 ▷ 13 → 2 d 2 1(18)	
15	P40 & 1 → DISC i 1 2($^{17}_{18}$)	
16	Acc & P40 → 5 d 1 1(19)	
17		INST + 4 → DL2 d 3 2(22)
18	1 + 2 → 1 d 5 1(25)	ENTER FAILURE.
19	Acc & 2 → 6 i 1 1(21)	
20		
21	6 ▷ 1 → 6 i 1 1(23)	(Acc ▷ 0 → Acc i 2 2(25))
22		----------
23	5 ≠ 6 → DISC d 3 2 ($^{28}_{29}$)	
24		
25	P_1 ▷ 38 → 2 i 1 1(27)	Acc + 0's → 4 i 1 2(27)
26		
27	INST + 1 → DL.1 d 1 1(30)	5 + 0's → TS i i 1 TS i LINK.
28		3 + 0's → DISC d 1 1($^{31}_{0}$)
29	$0x_s$ 0 - s 0 1(16)	Enter FAILURE.
30	----------	
31	4 ▷ 13 → 4 i 1 1(1)	

ROUTINE COPY. Given the parameters X, Y, and s in TS1, TS2, TS3 the table sends $(X + r)_{o-31}$ to $(Y + r)_{o-31}$ for r = 0,1, ... s-1. i.e. it makes a copy of the contents of the X tanks in the Y tanks. The link is in TS_4. The table is stored in part of instruction D.L.10.

This routine needs little explanation, the instruction $\big((X + r) + 0's \rightarrow (Y + r)\ i\ 32N)\big)$ is built up and obeyed for values of n from 0 to s - 1.

This completes the problem.

General plan of instructions.

10.0 Put X in 1st source position in TS1.

10.2 Put Y in destination position in TS2.

10.4 Add TS1 and TS2 together to give X in 1st source, Y in destination position in TS1 (for subsequent passages through this pt we will have X + r and Y + r.

10.6 Send to TS2 a number with a 1 in 1st source and a 1 in destination position using 10_8.

10.9 Form the instruction $\big((X + r) + 0's \rightarrow (Y + r)\ i\ 32\ 10(13)\big)$ from 10_{11} and send it to 10_{12}.

10.12 Obey instruction formed by 10_9 i.e. send $(X + r)_{o-31}$ to $(Y + r)_{o-31}$

10.13 Subtract one from TS3.

10.15 Test if TS3 contains zero. If not return to 10_4; otherwise to 10_3.

10.3 Send link instruction to TSi and obey.

ROUTINE INTERC (Interchange). Given X and Y in TS1 and TS2 the contents of DL X and DL Y are interchanged, using a fixed tank, for intermediate storage. (This is assumed to be D.L.200 which will be used for all "odd jobs" of this nature). The link is in TS3. The table is stored in part of instruction D.L.10. The contents of DL X are sent to D.L.200, the contents of D.L.Y to D.L.X and the contents of D.L.200 to D.L.Y.

General/

General plan of instructions.

10.7 Send X to 1st source position in TS1.

10.10 Send Y to 1st source position in TS2.

10.14 Form the instruction (X + 0's → D.L 200 i 32 10(18)) from 10_{16} and send it to 10_{17}.

10.17 Obey the instruction formed by 10_{14} i.e. send contents of D.L.X to D.L.200.

10.18 Send X to destination position in TS1.

10.20 Add TS2 to TS1 to give Y in 1st source and X in destination position.

10.22 Form the instruction (Y + 0's → X i 32 10(26)) from 10_{24} and send to 10_{25}.

10.25 Obey the instruction formed by 10_{22}, i.e. send contents of D.L.Y to D.L.X.

10.26 Send Y to destination position in TS1.

10.28 Form the instruction (200 + 0's → Y i 33 10(1)) from 10_{30} and send it to 10_{31}.

10.31 Obey the instruction formed by 10_{28}.

10.1 Send link to T.S.i and obey it.

ROUTINE MAX. M.O.C.

Given X, n, and s in TS1, TS2 and TS3 the routine finds which of the numbers in positions X_s, $(X + 1)_s$, ... , $(X + r-1)_s$ has the greatest modulus, and if it is $(X + k)_s$ it puts (X + k) in T.S.3. The routine uses TS4 to T.S.7 for other purposes and the link is therefore in T.S.8. The routine is stored in D.L.11. It may be regarded as the equivalent of finding the element of maximum modulus in a column vector and hence it is called MAX.M.O.C. (Maximum modulus in a column).

The process consists of 'n' repetitions of the following cycle with m equal to 0,1,2,...,r-1 successively.

Obtain the modulus of the number in $(X + m)_s$ and send it to TS6. Compare TS6 with TS4, and if TS6 is greater, replace the number in TS4 by the number in TS6 and send (X + m) to TS3.

Before the repeated cycle is started, X is sent to TS3 and 0 to TS4. If a number of the values examined are equal, the first of those which are greatest,

is/

is retained in TS4 and its position in TS3. If all the numbers are zero we obtain X in TS3.

General Plan of instructions.

11.19 Form the instruction (0's + 0's → 6 s s 11(31)) from 11_{21} and send it to TS7.

11.22 Clear TS4 (In general TS4 will contain the greatest of the numbers examined).

11.24 Put X in 1st source position in TS1 (In general it will contain (X + m)

11.26 Send contents of TS1 to TS3

11.28 Form the instruction ((X + m)+ 0s → 6 s s 11(31)) from TS7 and TS1 and send it to 11_{30}.

11 30 Obey the instruction formed in 11_{28} i.e. send $(X + m)_s$ to T.S.6.

11.31 Test the sign of number in TS6. If +ve go to 11_1, if -ve go to 11_2.

11.1 Go to 11_4 (redundant instruction)) In either case

11.2 Change sign of number in TS6) |TS6| is put into TS6.

11.4 Subtract TS6 from TS4 and send answer to TS5.

11.6 Check sign of TS5 if +ve go to 11_8 if -ve to 11_9.

11.8 Redundant instruction. Go to 11_{13}.

11.9 Send TS6 to TS4.

11.11 Send TS1 to TS3 (TS3 contains (X + m) which gave greatest modulus to date).

11.13 Subtract 1 from TS2.

11.15 Test if TS2 is empty. If not go to 11_{18}, otherwise 11_{17}.

11.18 Add 1 to TS1 in 1st source position and return to 11_{28} (this changes content of TS1 from X + m to X + m + 1)

11.17 TS3 is advanced 5 μs effectively by "delaying it by 35 μs". This gives the (X + m) which gave the number of greatest modulus, in the least significant position. The modulus of the greatest number is in TS4.

11.20 Send link to TSi and obey.

COPY, INTERC and MAXMOC.

	DL10				DL11
0	1 ▷ 5 →1	i	1	10(2) ENTRY for COPY	
1	3 + 0's→ TSi	i	1	TSi LINK for INTERC	6 + 0's→6 i 2 11(4)
2	2 ▷ 21 →2	i	1	10(4)	0's - 6→6 i 1 11(4)
3	4 + 0's TSi	i	1	TSi LINK for COPY	
4	1 + 2 1	i	1	10(6)	
5					
6	INST + 0's→2	d	1	10(9)	
7	1 ▷ 5 →1	d	1	10(10) ENTRY for INTERC	
8	(*)				6 + 0's→6 d 3 11(13)
9	INST + 1→D.L.10	d,1,10		10(12)	6 + 0's→4 i 1 11(11)
10	2 ▷ 5 →2	d	2	10(14)	
11	(0's + 0's→0	i	32	10(13))	1 + 0's→3 i 1 11(13)
12	.....				
13	3 - P_1 →3	i	1	10(15)	2 - P_1 →2 i 1 11(15)
14	INST + 1→D.L.10	d	1	10(17)	
15	3 + 0's→DISC	d	18	10(3)(4)	2 + 0's→DISC i 1 11(17)(18)
16	0's + 0's→200	i	32	10(18)	
17	.....				3 ▷ 35 →3 i 2 11(20)
18	1 ▷ 16 →1	i	1	10(20)	INST + 1→1 d 8 11(28)
19					INST + 3→7 d 1 11(22) ENTRY
20	1 + 2→1	i	1	10(22)	8 + 0's→TSi i 1 TSi LINK
21					(0's + 0's→6 s 0 11(31))
22	INST + 1→DL10	d	1	10(25)	0's + 0's→4 i 1 11(24)
23					
24	(0's + 0's→0	i	32	10(26))	1 ▷ 5 →1 i 1 11(26)
25	.....				
26	2 ▷ 16 →1	i	1	10(28)	1 + 0's→3 i 1 11(28)
27					
28	INST + 1→D.L.10	d	1	10(31)	7 + 1→D.L.11 i 1 11(30)
29					
30	(200 + 0's →0	i	33	10(1))	.....
31	.....				P_{40} & 6→DISC i 1 11(1)(2)

* Instruction consisting of one in source position, one in destination position and zeros elsewhere.

ROUTINE SCAVEC. (Multiplication of a vector by a scalar).

Given X, x, r and s in TS1, 2, 3 and 4 the routine produces a vector, in $D.L.200_{0-r}$, whose components are obtained by from those of the fector in $X_{0...r}$ by multiplying each of them by the number x, obtaining the product in the 80 digit Accumulator, shifting this product s places to the left and then taking the upper half of the resulting number (rounded off). The number 's' must lie in the range $0 \leq S \leq 39$. The link instruction is in TS8 and the table occupies instruction D.L.14.

The table is quite straightforward, and consists of the following cycle which is repeated $(r + 1)$ times for m = 0, 1,...r: The number x. X_m $(0 \leq m \leq r)$ is formed in the Accumulator. A test is made to check if this number lies in the range $-2^{79-S} \leq y < 2^{79-S}$. If it does not a failure indication is given since it would then be permissable to shift it s places to the left. The "round off" digit is added and the left shift performed. The most significant 40 digits in the Accumulator are sent to $D.L.200_m$.

General Plan of routine.

14.28 Put X in '1st source' position in TS1.

14.8. Put s in 'delay' position in TS4.

14.12 Send 14_{14} to TS.5 i.e. (7 + 0's 200 s 0 14(31))

14.15 Form the instruction (P$_1$ ▷ 39-s → 6 d 2 14(24)) from 14_{19} and TS4 and send it to 14_{20}

14.20 Obey the instruction formed by 14_{15} i.e. send P_{40-s} to TS6.

14.24 Clear Acc. ←

14.27 Form the instruction (X x_s 2 → s m 14(16)) from 14_{29} and T.S.1 and send it to 14_{30}. The first time this is done m' will be 0. It will increase by 1 for each passage through the cycle.

14.30 Obey the instruction formed by 14_{27} i.e. form $X_m \cdot x$ in the Acc.

14.16 Add round off using P_{40-s} which is in TS6.

14.18. Discriminate on sign of Acc.

14.21./

14.21.) These instructions determine whether or not the number in the Acc. lies
14.22.) in the range $-2^{79-s} \leq y < 2^{79-s}$. If it lies outside the table proceeds
14.23.) to 14_1 and is directed to failure. Otherwise it does to 14_0.
14.25.)

14_0. Form the instruction (Acc ▷ s → 7 i 2 14(9)) from 14_5 and send it to 14_6.

14_6 Obey the instruction formed by 14_0 i.e. send most significant 40 digits of Acc. to TS7. after a left shift of s places.

14_9 Send TS5 to D.L.14_{30} and obey it.

14_{30} The instruction obeyed is (7 + 0's → 200 s m 14(31)) which was obtained in TS5 with m=o by instruction 14_{12} above. This instruction has the value m modified for each repetition of the cycle.
The effect of the instruction is to send T.S.7 to 200_m.

14_{31} Test if TS3 contains zero; if not go to 14_4; otherwise 14_3.

14_4 Subtract one from TS3 (TS3 thereby keeps count of number of cycles).

14_7 Add one to TS1.

14_{16} Add one to TS5 and return to 14.24 to repeat cycle with increased value of m.

14_3 Send link to TSi and obey it.

SCAVEC/

SCAVEC.

```
 0   INST + 4 ⇒ D.L.14   d   4   14(6)
 1   ENTER FAILURE.
 2
 3   8  +  0's ⇒ TSi   i  1   T.Si        LINK EXIT
 4   3 - P1 ⇒ 3  d 1      14(7)
 5   (Acc ▷ 0 ⇒ 7  i  2   14(9))
 6   ------------------------
 7   1 + P1 ⇒ 1   d   1    14(10)
 8   4 ▷ 13 ⇒ 4   d   2    14(12)
 9   5  +  Os  ⇒ D.L.14 d 19 14(30)
10   5 + P1 ⇒ 5   d  12   14(24)
11
12   INST  + Os' ⇒ 5   d  1  14(15)
13
14   (7  + Os' ⇒ 200  s  0  14(31))
15   INST  -  4 ⇒ D.L.14  d  3  14(20)
16   6  +  Os ⇒ Acc+  i  1  14(18)
17
18   Acc & P40 ⇒ DISC  d  1  14(21)(22)
19   (P1 ▷ 39 ⇒ 6  d  2  14(24))
20   ------------------------
21   Acc - 6 ⇒ 7   i  1  14(23)
22   Acc + 6 ⇒ 7   d  1  14(25)
23   P40 + 7 ⇒ 7   i  1  14(25)
24   0's  +  Os' ⇒ Acc  i  2  14(27)
25   P40  &  7 ⇒ DISC  d  5  14(1)(0)
26
27   INST + 1 ⇒ D.L.14  d  1  14(30)
28   1 ▷ 5 ⇒ 1   d  10   14(8) ENTRY
29   (0xs  2  -  s  0  14(16))
30   ------------------------
31   3  +  0's ⇒ DISC   d  2  14(3)(4)
```

ROUTINE/

ROUTINE DIVII 1. Two possible routines for division are given. The first of these performs the following operations.

Given a, b and h in TS1, TS2 and T3 respectively, it produces $(2^h\, a/b + e)$ in TS4 where $|e| \leq \frac{1}{2}$ and gives a failure indication if $2^h\, a/b < 2^{39} - \frac{1}{2}$ or $> 2^{3d} - \frac{1}{2}$. The link is in TS9. Roughly speaking this table regards the two numbers a and b in TS1 and TS2 as integers and produces the quotient a/b to h binary places in TS4, with the usual "rounding off rule".

The table is based on the following method of binary division. Suppose we have two non-zero binary numbers A and B in TS1 and TS2 respectively such that $|A| \leq 2\,|B|$ and we perform the following operation repeatedly. (in what follow (1) denotes the number in TS1 etc). Send $2^{40} - 1$ to TS4 (i.e. a number consisting entirely of 1's). (X) If (1) and (2) have opposite signs send (1) + (2) to T.S.1; if (1) and (2) have the same signs send (1) - (2) to TS1 and add 1 to TS4.

(Y) Send 2 x (1) to TS1.

(Z) Send 2 x (4) to TS4 and return to X.

Suppose X,Y and Z are repeated (r + 1) times, then X and Y are performed once and finally X once again.

Suppose (1) is s at the end of this process, and $e_r = 1$ or -1 as 1 is added to (4) or not when X is performed for the r^{th} time, then

$$2^{r+2}\, A = s + B(e_{r+3} + 2\, e_{r+2} + 2^2\, e_{r+1} + \cdots + 2^{r+2} e_1)$$

i.e. $$2^r\, A = \frac{s}{4} + B\left(\frac{e_{r+3}}{4} + \frac{e_{r+2}}{2} + e_{r+1} + 2\, e_r + \cdots + 2^r\, e_1\right).$$

If $y_r = 1$ or 0 according as 1 is added to (4) or not then

$$\text{Final (4)} = 2^{r+1}\, y_1 + 2^r\, y_2 + \cdots + 2\, y_{r+1} + y_{r+2} + y_{r+3} + 2^{r+1}\,(2^{40} - 1) \quad \text{modulo } 2^{40})$$

$$= 2^r\,(e_1+1) + 2^{r-1}\,(e_2+1) + \cdots + (e_{r+1}+1) + \tfrac{1}{2}\,(e_{r+2}+1) + \tfrac{1}{2}\,(e_{r+3}+1) - 2^{r+1}$$

since $y_r = \frac{1}{2}\,(e_r + 1)$

$$= 2^r\, e_1 + 2^{r-1}\, e_2 + \cdots + e_{r+1} + \tfrac{1}{2}\, e_{r+2} + \tfrac{1}{2}\, e_{r+3}.$$

$$2^r\, \frac{A}{B} - (\text{final (4)}) = \frac{s}{4B} - \frac{e_{r+3}}{4}$$

Since/

Since $|A| \leq 2\,|B|$ we know that $|(1)| \leq |B|$ each time immediately after (X) has been performed and in particular $|s| \leq B$.

Hence $$\left|2^r \frac{A}{B} - \text{final } (4)\right| \leq \frac{1}{4} + \frac{1}{4} \leq \frac{1}{2}.$$

(It is easy to see that the equality can only hold if after the penultimate performance of operation X, (1) = 0 and then $2^r \frac{A}{B}$ - final (4) = $-\frac{1}{2}$ if $B > 0$ and

$2^r \frac{A}{B}$ - final (4) = $\frac{1}{2}$ if $B < 0$.)

The process given above will therefore produce A / B to r binary places with the correct round off, provided A and B satisfy the conditions stated. The general problem of dividing a by b where $-2^{31} \leq a < 2^{31}$ and $-2^{31} \leq b < 2^{31}$ is reduced to the above case by the following procedure. This effectively replaces a by $2^k a$, and h by (h-k) or by $2^k b$, and h by (h + k), (where $k \geq 0$ in both cases), so that if A, B and H are the final values of a, b and h

$|B| \leq A < 2\,|B|$ if a and b have the same signs.

$|B| < A \leq 2\,|B|$ if a and b have opposite signs.

and gives a failure indication if $H > 38$ because the answer in this case is too great to be represented in the usual one word convention. The procedure of "moving the smaller number up" until it is comparable with the larger, rather than "moving the larger number down" until it is comparable with the smaller, has been adopted in spite of the fact that it is slightly more difficult, (owing to the difficulties arising through the possibility of losing the sign digit), so that we can deal with cases when a and b (or b only) are exact, when we may obtain more significant figures in the quotient than are contained in the smaller of a and b. Special precautions have been taken

so/

so that all the cases of equality in $-2^{39} \leq a < 2^{39}$, $-2^{39} \leq b < 2^{39}$ (and similarly with 2^h a/b) may be covered. This involves a little extra work both in the tables and the mathematical analysis.

(P) Test if b(i.e. (2)) = 0 → YES give a failure indication.

↓ NO

Test if a (i.e. (1)) = 0 → YES send zeros to TS4 and finish.

Test if $|(1)| < |(2)|$ if (1) and (2) have the same signs

and $|(1)| \leq |(2)|$ if (1) and (2) have different signs.

↙ NO ↘ YES Send 2 x (1) to TS1 Double a.

Send (3) -1 to TS3. Replace h by h - 1, return to P

The tests take a different form according as (1) and (2) have the same or different signs because it is correct to code them in this way and it also happens that they are suitable tests.

(Q) Send 2 x (2) to TS8.

Send (3) + 1 to TS3.

Test if $|(8)| \leq |(1)|$ if (1) and (2) have same signs.

and $|(8)| < |(1)|$ if (1) and (2) have opp. signs.

↓ NO ↘ YES

Send (8) to TS2 i.e. double b. are return to Q.

(R) Test (3) ≥ 0. (It will by this time contain H + 1).

↓ ↘ NO. Send zeros to TS4 and finish.

YES

Test (3) > 39 → YES give failure indication. (Answer too great).

↓ NO

(S)./

(S). Test if (1) = 2^{40} → YES put -2 in T.S.4 and proceed

↓ NO to division cycle.

proceed to division cycle.

The test R is to check if $H + 1 \geqslant 0$. If it is not, $H < -1$ and then $|2^h a/b| = |2^H A/B| \leqslant 2^{-2} \times 2 = \frac{1}{2}$ (since if $H < -1$ it must be $\leqslant -2$). In this case zeros are sent to TS4 and the solution is finished.

It should be noticed that the final values of A and B can now be numerically greater than 2^{39}; the signs will always be those of a and b which are detected before they are multiplied. The test S is to deal with the special case when $A = 2^{40}$ which can only happen if the original value of b is -2^{39}. There is no corresponding case with $A = -2^{40}$ because the biggest positive value b can have is $2^{39} - 1$ and in this case the final value of A is certainly $\leqslant 2(2^{39} - 1)$. It will soon be verified that in the special case when test S gives the answer 'YES' we obtain the correct result if -2 is sent to TS4 before the division process, instead of -1. Tests P and Q are performed as follows.

If	$a \geqslant 0$	$b \geqslant 0$	$2^l a$ is put in TS4.
	$a \geqslant 0$	$b < 0$	$-2^l a$ is put in TS4
	$a < 0$	$b \geqslant 0$	$-2^l a - 1$ " " "
	$a < 0$	$b < 0$	$2^l a - 1$ " " "

l=0,1,2 etc. for successive applications of (P)

Then for test P we test if (4) - (2) $\leqslant 0$ (if $b \geqslant 0$) and > 0 (if $b < 0$)

for test Q we test if (4) - (8) $\geqslant 0$ (if $b \geqslant 0$) and < 0 (if $b < 0$).

Since all these tests are done by considering the 40^{th} digit it is essential to be sure that

$$-2^{39} \leqslant (4) - (2) < 2^{39} \quad \text{at every stage.}$$

or $$-2^{39} \leqslant (4) - (8) < 2^{39}$$

This is assured for all possible cases as follows.

We/

We know initially $-2^{39} \leqslant a < 2^{39}$ $\qquad -2^{39} \leqslant b < 2^{39}$

and also the cases $a = 0$ and $b = 0$ have been eliminated.

Suppose $a > 0$ $b > 0$. Then initially $2^{39} - 2 \geqslant a-b \geqslant 2-2^{39}$

If $a - 2^l b \geqslant 0$ then

$$a - 2^l b > a - 2^{l+1} b = a - 2^l b - 2^l b \geqslant 0-a$$

If for a given l.

$$2^{39} - 1 \geqslant a - 2 b \geqslant -2^{39} + 1$$

Then $a - 2^{l+1} b < a - 2^l b \leqslant 2^{39} - 1$

and $a - 2^{l+1} b \geqslant -a > -2^{39}$

$$a - 2^{l+1} b \geqslant -2^{39} + 1.$$

$$2^{39} - 1 > a - 2^{l+1} b \geqslant -2^{39} + 1.$$

Hence the same inequality holds for $(l + 1)$. Since it holds for $l = 0$, by induction it holds for all the values of l which we shall require.

If $2^l a - b < 0$ then $2^l a - b \leqslant -1$ and $2^l a \leqslant b - 1$

$$2^{la} - b < 2^{l+1} a - b = 2^l a - b + 2^l a$$

$$2^{la} - b \leqslant b - 2 < b - 1$$

hence by induction $-2^{39} + 1 < 2^{l+1} a - b < 2^{39} - 2$ for any l that we require.

When $2^{k+1} a - b \geqslant 0$ and $2^k a - b < 0$, $-1 > 2^{k+1} a - 2b \geqslant -b$

$$\geqslant -2^{39} + 1.$$

Suppose $a > 0$ $b < 0$. Initially $2^{39} - 1 \geqslant -a-b \geqslant 2-2^{39}$.

If $-a - 2^l b < 0$ then $-2^l b < a$ and hence $-2^{l+1} b < 2a - 1$

$$-a - 2^l b < - a - 2^{l+1} b < a - 1.$$

and/

and hence by induction $1-2^{39} \leq -a-2^{l+1}\ b \leq 2^{39}-2$ for any l that we require.

If $-2^{l}\ a - b \geq 0$.

$-2^{l}a - b > -2^{l+1}\ a - b \geq b$.

and hence by induction $1 - 2^{39} \leq -a - 2^{l+1}\ b \leq 2^{39} - 2$ for any l that we require.

If $-2^{l}\ a - b \geq 0$

$-2^{l}\ a - b > -2^{l+1}\ a - b \geq b$.

and hence by induction $2^{39} > -2^{l+1}\ a - b \geq -2^{39}$ for any l that we require.

When $-2^{k+1}\ a - b < 0$ and $-2^{k}\ a - b \geq 0$.

$$0 \leq -2^{k+1}\ a - 2b < -b \leq 2^{39}$$
$$0 \leq -2^{k+1}\ a - 2b < 2^{39}$$

Suppose $a < 0$ $b > 0$. Initially $2^{39} - 2 \geq -a - b\ 1 \geq 1 - 2^{39}$

If $-a - 2^{l}b - 1 \geq 0$.

$-a - 2^{l}b - 1 > -a - 2^{l+1}\ b - 1 \geq 1 + a$.

and hence by induction $2^{39} - 1 > -a - 2^{l+1}\ b - 1 \geq 1 - 2^{39}$ for any l that we require.

If $-2^{l}a - b - 1 < 0$.

$-2^{l}\ a - b - 1 < -2^{l+1}\ a - b - 1 < b$ and hence by induction $-2^{39} < -2^{l+1}\ a-b-1 < 2^{39} - 1$ for any l that we require.

When $-2^{k+1}\ a - b - 1 \geq 0$ and $-2^{k}a - b - 1 < 0$

then $0 > -2^{k+1}\ a - 2b - 1 \geq -b \geq -2^{39} + 1$

Suppose $a < 0$ $b < 0$. Initially $-2^{39} \leq a - b - 1 \leq 2^{39} - 2$

If/

If $a - 2^{1} b - 1 \leq 0$.

$a - 2^{1} b - 1 \leq a - 2^{1+1} b - 1 \leq -a$.

and hence by induction $-2^{39} - 1 \leq a - 2^{1+1} b - 1 \leq 2^{39}$ for any 1 that we require.

If $2^{1} a - b - 1 \geq 0$

$2^{1a} - b - 1 \geq 2^{1+1} a - b - 1 \geq 1 + b$

and hence by induction $2^{39} - 1 \geq 2^{1+1} a - b - 1 \geq -2^{39}$ for any 1 that we require

When $2^{k+1} a - b - 1 \leq 0$ and $2^{k} a - b - 1 \geq 0$

$1 \leq 2^{k+1} a - 2b - 1 \leq -b \leq 2^{39}$

Since the table has been discussed in considerable detail only a brief description of the plan of instructions is given and the table follows. The division cycle of instruction and the preparatory cycle are duplicated so that two complete cycles are performed per major cycle i.e. the table works at twice the speed which is attained without duplication. For this reason the table appears to be quite long. If the extra speed is not necessary the table reduces to about 48 instructions.

The actual form taken by the table has to some extent been determined by the fact that the instructions of the division cycle have been duplicated. If this duplication is omitted then it becomes possible to perform the failure discrimination (i.e. testing whether the answer is too great) in the division cycle. It is only necessary to test immediately before (Y) is performed that the 40^{th} digit of (4) has not changed since the previous time (Y) was reached, and if it has, give a failure indication unless:-

(a) It has changed from 1 to 0 when (Y) is reached for the last time, for then $2^{h} a/b + e = 0$.

or (b) It has changed from 1 to 0 on the penultimate time when (Y) is reached and has changed back again to 1 on the last time when (Y) is reached, for then $2^{h} a/b + e = -2^{39}$.

If/

If the failure discrimination is done in this way it is only necessary for the initial preparation to ensure.

$|A| \leq 2|B| \leq 2^{40}$ and $A \neq 2^{40}$. The simplest preparation is now the following.

Test if 40^{th} and 39^{th} digit of (2) are the same. ←

↙ NO Proceed

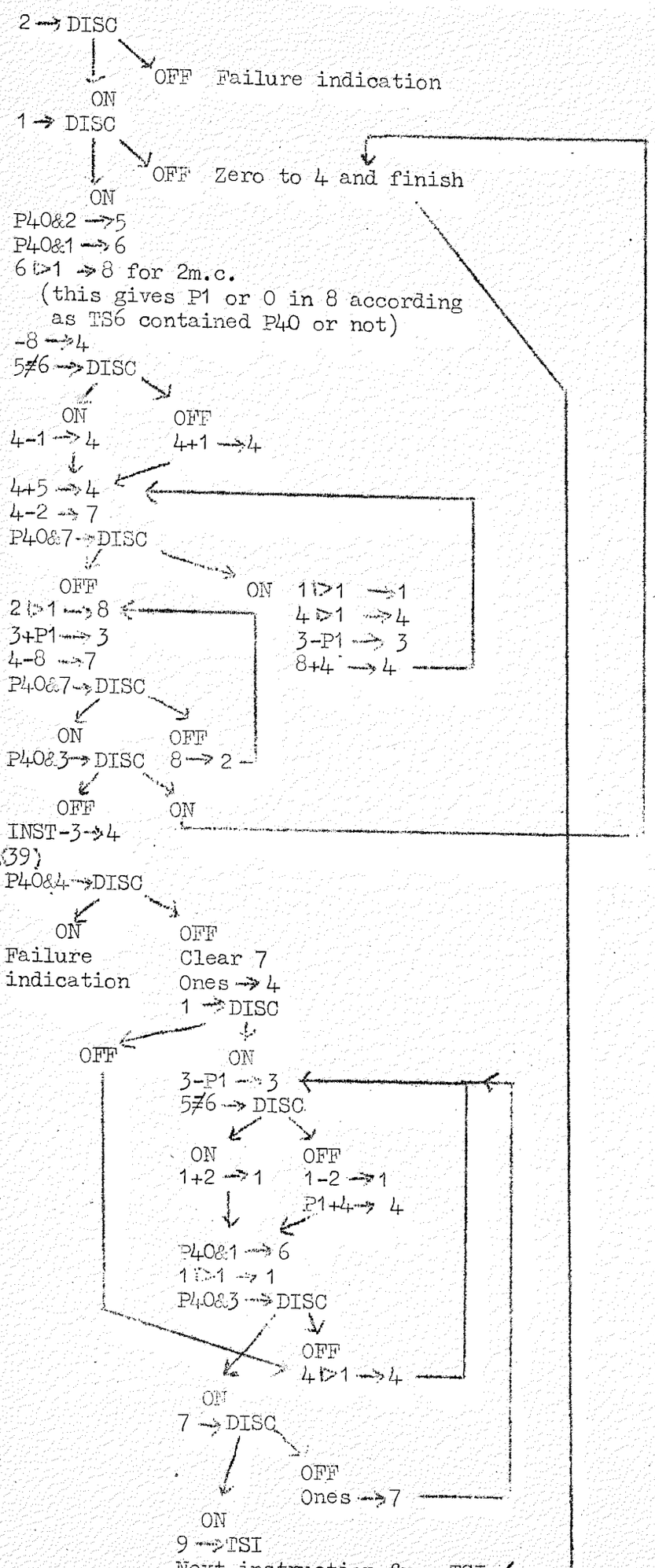
↘ YES. Double the contents (2)

Send 1 + (3) to T.S.3 and repeat. ——

This ensures $2^{38} < |B| < 2^{39}$ or $B = 2^{38}$ or $B = -2^{39}$, but it may involve an unnecessary waste of time in "moving b up". The actual process adopted when the division cycle is not duplicated will be found in the next routine DIVID 2.

General/

General Plan of Instructions



DIVID 1/

DIVID 1.

	DL3	DL4
1	3-P1 → 3 i 1 3(3)	4 + 5 → 4 i 1 4(3)
2		
3	5≠6 → DISC i 1 3(5)(6)	4 - 2 → 7 i 1 4(5)
4		
5	1 - 2 → 1 i 1 3(7)	P40 & 7 → DISC i 1 4(7)(8)
6	1 + 2 → 1 d 1 3(9)	
7	P1 + 4 → 4 i 1 3(9)	2 ▷ 1 → 8 i 1 4(9)
8		1 ▷ 1 → 1 i 1 4(10)
9	P40 & 1 → 6 i 1 3(11)	3 + P1 → 3 i 1 4(11)
10		4 ▷ 1 → 4 i 1 4(12)
11	1 ▷ 1 → 1 i 1 3(13)	4-8 → 7 i 1 4(13)
12		3-P1 → 3 i 1 4(14)
13	P40 & 3 → DISC i 1 3(15)(16)	P40 & 7 → DISC i 1 4(15)(16)
14		8 + 4 → 4 d 1 4(17)
15	4 ▷ 1 → 4 i 1 3(17)	8 + 0's → 2 d 6 4(23)
16	7 + 0's → DISC i 1 5(18)(19)	P40 & 3 → DISC d 16 5(2)(3)
17	3 - P1 → 3 i 1 3(19)	4 + 5 → 4 i 1 4(19)
18		
19	5 ≠ 6 → DISC i 1 3(21)(22)	4 - 2 → 7 i 1 4(21)
20		
21	1-2 → 1 i 1 3(23)	P40 & 7 → DISC i 1 4(23)(24)
22	1+2 → 1 i 1 3(25)	
23	P1 + 4 → 4 i 1 3(25)	2 ▷ 1 → 8 i 1 4(25)
24		1 ▷ 1 → 1 i 1 4(26)
25	P40 & 1 → 6 i 1 3(27)	3 + P1 → 3 i 1 4(27)
26		4 ▷ 1 → 4 i 1 4(28)
27	1 ▷ 1 → 1 i 1 3(29)	4 - 8 → 7 i 1 4(29)
28		3 - P1 → 3 i 1 4(30)
29	P40 & 3 → DISC i 1 3(31)(0)	P40 & 7 → DISC i 1 4(31)(0)
30		8 + 4 → 4 d 1 4(1)
31	4 ▷ 1 → 4 i 1 3(1)	8 + 0's → 2 d 6 4(7)
0	7 + 0's → DISC d 16 5(18)(19)	P40 & 3 → DISC i 1 5(2)(3)

DIVID 1. (Contd).

DL5

1	1 + 0's→DISC i 1	5(3)(4)
2	INST - 3→4 d 2	5(6)
3	0's + 0's→4 d 14	5(19)
4	P40 & 2→5 d 15	5(21)
5	(39	(8)
6	P40 & 4→DISC i 1	5(9)
7		
8	0's + 0's→7 i 1	5(10)
9	Enter failure table.	
10	1's + 0's→4 i 1	5(12)
11		
12	1 + 0's→DISC i 1	(14) 5(15)
13		
14	255 + 0's→255 d 15	3(31)
15	255 + 0's→255 i 1	3(17)
16		
17		
18	1's + 0's→7 d 13	3(1)
19	9 + 0's→TSi i 1	TS i.
20		
21	P40 & 1→6 i 1	5(23)
22		
23	6 ▷ 1→8 i 2	5(26)
24		
25		
26	0's - 8→4 i 1	5(28)
27	ENTRY 2 + 0's→DISC d 3	(0) 5(1)
28	5≠6 → DISC i 1	5(30) (31)
29		
30	4 + 1→4 d 1	4(1)
31	4 - 1→4 i 1	4(1)
0	ENTER FAILURE TABLE.	

DIVID 2/

DIVID 2. The second form of division table deals with a double length dividend. Given a,b,h in the Accumulator, TS1 and TS2 respectively the table produces $2^{h-40}\ a/b + e$ in TS4, where $|e| \leq \frac{1}{2}$ and gives a failure indication if b=0 or $2^{h-40}\ a/b < -2^{39} - \frac{1}{2}$ or $> 2^{39} - \frac{1}{2}$. (The numbers a and b are regarded as integers).

The link is in TS8 and h must satisfy $h \geq 0$.

With this form of division table it is not possible to fit the division cycle twice into a major cycle to obtain double speed. The basis of the method is similar to that of DIVID 1,though the initial preparation is now simpler and consists essentially of the following process.

(P) Test if $\left|\frac{a}{2^{40}}\right| \leq |b|$ if b is positive (where integral part of $\frac{a}{2^{40}}$ is taken).

$\left|\frac{a}{2^{40}}\right| < |b|$ if b is negative

NO

YES

(Q) Test if $\left|\frac{a}{2^{40}}\right| \leq 2|b|$ if b is positive

$\left|\frac{a}{2^{40}}\right| < 2|b|$ if b is negative

Proceed

YES

Proceed

NO

Double b; add 1 to h and return to test (Q).

To do these tests

If $a \geq 0$ $b > 0$ Put the 40 most significant digits of -a in TS4

" $a \geq 0$ $b < 0$ " " " " " " " a in "

" $a < 0$ $b > 0$ " " " " " " " a in "

" $a < 0$ $b < 0$ " " " " " " " -a in "

Then/

Then for test (P) we test if the 40^{th} digit of (4) + (2) is 0, if $b > 0$,

or 1, if $b < 0$.

and for test (Q) we test if the 40^{th} digit of (4) + (2) 2 is 0, if $b \quad 0$,

or 1, if $b \quad 0$.

These tests ensure $\left|\frac{a}{2^{40}}\right| \leq 2|b|$ if b is positive

$< 2|b|$ if b is negative.

provided that (4) + (2) or (4) + 2 (2) satisfies $-2^{39} \leq \begin{matrix}(4) \\ (4)\end{matrix} + \begin{matrix}(2) \\ 2\ (2)\end{matrix} \leq 2^{39} - 1$

This is checked by the following analysis.

<u>If $a \geq 0 \quad b > 0$.</u> Initially $2^{79} > 2^{40} \quad b-a > 1-2^{79}$

And if $2^{1+40} \quad b-a < 0$.

$$2^{1+40} \quad b - a < 2^{1+41} \quad b - a < a - 1.$$

and hence by induction $1-2^{79} < 2^{1+41} \quad b - a < 2^{79} \quad -2$ for any 1 that we require.

<u>If $a \geq 0 \quad b < 0$.</u> Initially $2^{79} > 2^{40} \quad b + a \geq -2^{79}$.

And if $2^{1+40} \quad b + a \geq 0$.

$$2^{1+40} \quad b + a > 2^{1+41} \quad b + a \geq -a.$$

and hence by induction $2^{79} > 2^{1+41} \quad b + a \geq 1 - 2^{79}$ for any 1 that we require.

<u>If $a < 0 \quad b > 0$.</u> Initially $2^{79} > 2^{40} \quad b + a^{7} > - 2^{79}$

and if $2^{1+40} \quad b + a < 0$

$$2^{1+40} \quad b + a < 2^{1+41} \quad b + a < -a-1.$$

and hence by induction $-2^{79} < 2^{1+41} \quad b + a < 2^{79}$ for any 1 that we require.

<u>If $a < 0 \quad b < 0$.</u> Initially $2^{79} > 2^{40} \quad b - a > -2^{79}$

and if $2^{1+40} \quad b - a \geq 0$.

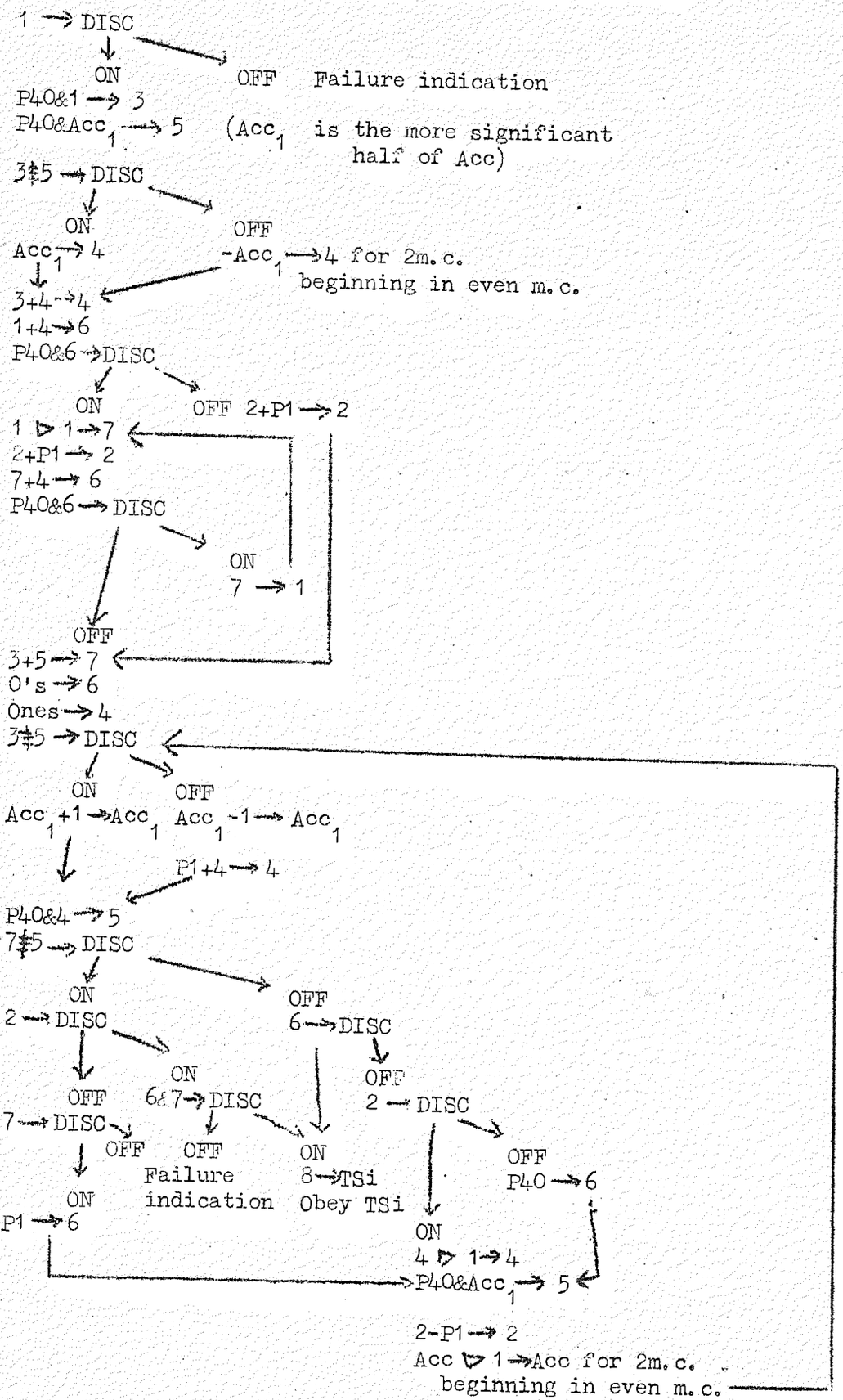
$$2^{1+40} \quad b - a > 2^{1+41} \quad b - a \geq a.$$

and hence by induction $2^{79} > 2^{1+41} \quad b - a \geq -2^{79}$ for any 1 that we require.

A brief indication of the general plan follows together with the instruction table.

General/

General Plan of Instructions.



DL12/

	DL12				DL13
1	3 ≠ 5 → DISC	i	1	12(3)(4)	
2					2 + P_1 → 2 i 1 13(4)
3	Acc -1 → Acc	i	1	12(5)	
4	Acc + 1 → Acc	d	1	12(7)	7 + 4 → 6 i 1 13(6)
5	P1 + 4 → 4	i	1	12(7)	
6					P46 & 6 → DISC i 1 13(8)(9)
7	P40 + 4 → 5	i	1	12(9)	P40 & Acc → 5 d 2 13(11)
8					3 + 5 → 7 d 16 12(26)
9	7 ≠ 5 → DISC	i	1	12(11)(12)	7 + 0's → 1 d 5 13(16)
10					
11	6 + 0's → DISC	i	1	12(13)(14)	3 ≠ 5 → DISC d 1 13(14)(15)
12	2 + 0's → DISC	d	5	12(19)(20)	
13	2 + 0's → DISC	i	1	12(15)(16)	
14	8 + 0's → TSi	i	1	TS.i EXIT.	0's -$_0$ Acc → 4 i 2 13(17)
15	P40 + 0's → 6	d	1	12(18)	0's + Acc → 4 i 1 13(17)
16	4 ▷ 1 → 4	i	1	12(18)	1 ▷ 1 → 7 i 1 13(18)
17					3 + 4 → 4 d 7 13(26)
18	P40 & Ac → 5	d	1	12(21)	2 + P1 → 2 i 1 13(20)
19	7 + 0's → DISC	d	1	12(22)(23)	
20	6 & 7 → DISC	d	2	12(24)(25)	7 + 4 → 6 i 1 13(22)
21	2 - P1 → 2	d	7	12(30)	
22	Enter Failure Table.				P40 & 6 → DISC i 1 B(24)(25)
23	P1 + 0's → 6	d	25	12(18)	ENTRY 1 + 0's → DISC d 4 13(29)(30)
24	255 + 0's → 255	d	28	12(22)	3 + 5 → 7 i 1 12(26)
25	255 + 0's → 255	d	19	12(14)	7 + 0's → 1 d 5 13(0)
26	0's + 0's → 6	i	1	12(28)	1 + 4 → 6 i 1 13(28)
27					
28	Ones + 0's → 4	d	3	12(1)	P40 & 6 → DISC d 1 13(31)(0)
29					Enter failure table.
30	Acc ▷ 1 → Acc	i	2	12(1)	P40 & 1 → 3 d 7 13(7)
31					2 + P1 → 2 d 7 13(8)
0					1 ▷ 1 → 7 i 1 13(2)

PROBLEM 4/

PROBLEM 4.

The solution of a set of linear algebraic equations by an iterative process.

A simple iterative method for solving n linear equations

$$a_{m+1,1}\ x_1 + a_{m+1,2}\ x_2 + \cdots\cdots + a_{m+1,n}\ x_n = b_{m+1} \quad m=0,\ldots,n-1$$

in the n variables $x_1, \ldots x_n$ is by repeating the following cycle of operations. The cycle starts with a given approximation to the solution $y_1, y_2, \ldots y_n$ and derives the next approximation $z_1 z_2, \ldots z_n$ as follows.

With $x_2, x_3, \ldots x_n = y_2, y_3, \ldots y_n$ respectively a value z_1 is obtained for x_1 from the first equation. With $x_1 = z_1$ and $x_3, x_4, \ldots x_n = y_3, y_4, \ldots y_n$ respectively a value, z_2, is obtained fro x_2 from the second equation. In general with $x_1, x_2, \ldots x_{m-1} = z_1, z_2, \ldots z_{m-1}$ and with $x_{m+1}, x_{m+2}, \ldots x_n = y_{m+1}, y_{m+2}, \ldots y_n$, a value z_m is obtained, for x_m, from the m^{th} equation. Similarly from the values $z_1, \ldots z_n$ a further set of values for $x_1 \ldots x_n$ is obtained. The process is complete at the end of the first cycle for which $|y_n - z_n| < E$ for $r = 1, \ldots n$ where E has been chosen to be small enough for the desired accuracy in the solution. The initial values assumed by $x_1 \ldots x_n$ for the first cycle are all zeros.

The above cycle always converges if the matrix of the equations is positive definite, though the convergence may be extremely slow. It is most satisfactory when the diagonal terms are considerably greater than the other terms of their rows. The method, might however be quite valuable in the solution of hyperbolic partial differential equations by the method of characteristics where for each step of the integration it is necessary to solve a set of equations of quite low order (4, 5 or 6 usually). Since the integration is usually performed with a fairly small mesh length, the values of the variables obtained for the previous step will be a good approximation to the values for the current step, and may therefore be taken as the first approximation in the above cycle instead of a set of zeros.

Programming/

Programming of the iterative process.

The method of programming given below applies to sets of order not greater than 31, but a simple modification would make it applicable to sets of any order (provided, of course, the memory was sufficiently extensive).

The coefficients b_{m+1}, $a_{m+1,1}$, $a_{m+1,2}$, ... $a_{m+1,n}$ of the equations are stored in positions $(A + m)_{0..n}$ m=0,1,..,n-1 and the solutions are derived in positions $B_{1...n}$. The coefficients are assumed given to an accuracy of 5 decimal digits, there being at least one coefficient in each row of the equations which is large enough to require the full five digits, so that in the binary equivalents the most significant digit will be in the Pa position where $a \leq 17$. The solutions x_1 ... x_n are given to 20 binary places, (i.e. 20 binary places are given after the binary point) and since they are not permitted more than 40 binary digits the table is programmed to give a failure indication if any x at any stage is greater than 2^{19}. These restrictions on the range of the numbers may seem rather harsh, but since this table would only be used for fairly 'simple' sets of linear equations there is probably no justification for making the table more comprehensive. A much more general method of solving sets of equations is given under PROBLEM 5. Suppose at any stage of the process we have as values of x_1, x_2, ... x_n, the numbers y_1, y_2, ... y_n to 28 binary places, stored in B_1 to n, and we are about to redetermine x_{m+1}.

We have

$$-a_{m+1,m+1}x_{m+1}=b_{m+1}(-1.0)+a_{m+1,1}y_1+\ldots+a_{m+1,m}y_m+a_{m+1,m+2}y_{m+2}+\ldots+a_{m+1,n}y_n \quad (1)$$

In order to simplify the computation of the expression on the right of equation (1) we use the following artifice.

The number -1.0 (i.e. -2^{20} if we take the integral interpretation, since the binary point is between positions 20 and 21) is stored in position B_o throughout the iteration, and when we are about to calculate from B_m a new value for x_{m+1} the current value of y_{m+1} is deleted from B_m so that B_o to n now contains

$$-1.0,\ y_1\ y_2\ \ldots\ y_m\ \ 0\ \ y_{m+2}\ \ y_{m+2}\ \ldots\ y_n\ ;$$

hence/

hence the right hand side of (1) may be calculated by using the routine SCAPRO to form the scalar product of order (n + 1) of $B_{o\ ..\ n}$ and $(A + m - 1)_{o\ -\ n}$ since the latter contains

b_{m+1}, $a_{m+1,1}$; $a_{m+1,2}$; $a_{m+1,m}$; $a_{m+1,\ m+1}$; $a_{m+1,m+2}$, $a_{m+1,m+3}$, ... $a_{m+1,n}$

The new value of x_{m+1} can therefore be obtained by dividing the appropriate part of the scalar product by "$-\ a_{m+1,m+1}$" using the simple DIVIDE table. The above discussion shows that the only routines needed for this table are the scalar product table SCARPRO and the simple divide table DIVID 1. The highest temporary storage used by either of these routines is TS9 which contains the link in DIVID 1; hence the temporary storages used by the main table for solving the equations must be T.S.10 and above. The parameters for the main table are the numbers A, B and n which describe the positions of the matrix of the equation, the delay line holding the solution and the order of the equations. These are stored in TS12, TS13 and TS14 respectively, and the link is in TS15.

The general plan of the instructions is given below, followed by the detailed table. It will be seen that the table consists of two main cycles. Corresponding to each passage through the outer cycle, a complete set of values x_1 .... x_n is determined and, corresponding to each passage through the inner cycle, a new value is determined for one of the variables x_{m+1}. The outer cycle therefore contains 'n' repetitions of the inner cycle. When x_{m+1} is being recalculated in the inner cycle TS10 contains (m) and each time this cycle is completed TS10 is increased by 1. Hence in order to determine when the inner cycle has been completed 'n' times the contents of TS10 are compared with 'n' which is in TS14. In order to determine whether another outer cycle is necessary the following method is used. When, in the inner cycle, x_{m+1} is being recalculated, the current value is stored in TS11. If the difference between the current x_{m+1} and the new x_{m+1} is greater than 2^{-19} a P40 is sent to TC1. Therefore at the end of the 'n' repetitions of the inner cycle, TC1 will be on, if at least one of the values of x has changed by more than 2^{-19}. If TC1 is on a further tour of the outer cycle is necessary and the

contents/

contents of TS10 are restored to zero, so that x_1 is calculated during the first tour of the inner cycle.

General Plan of table.

7.12 Send B to TS1 in position of 'destination' element

7.15 Form the instruction (0's + 0's → B i 32 7(22)) from 7_{20} and put it in 7_{21}.

7.21 Obey the instruction formed by 7_{15}, i.e. clear D.L.B.

7.22 Form the instruction (ones ▷ 20 → B d 2 7(0)) from 7_{27} and put it in 7_{28}.

7.28 Obey the instruction formed by 7_{22}, i.e. put -1·0 (20 binary places) in B_0.

\+ 7.0 Clear TS10 (In general TS10 contains 'm' when x_{m+1} is recalculated).

x 6.17 Send B to TS1 in position of 'first source' element.

6.21 Send B to TS2 in position of 'destination' element.

6.23 Add m in the "timing position" to TS2.

6.25 Add m in the "timing position" to TS1.

6.27 Form the instruction (B + 0's → 11 s m + 1 6(31)) from 6_{29} and put it in 6_{30}.

6.30 Obey the instruction formed by 6_{27} i.e. send B_{m+1} TS11 (this puts the current value of x_{m+1} in TS11)

6.31 Form the instruction (0's + 0's → B s m + 1 7(31)) from 7_{29} and put it in 7_{30}.

7.30 Obey the instruction formed by 6_{31} i.e. put zero in B_{m+1}, which is the position normally occupied by x_{m+1}.

7.31 Form the instruction (4 + 0's → B s m + 1 8(31)) from 8_{29} and put it in 7_{30} (This is obeyed later)

8.30 Form (A + m) in TS1

6.0 Send (A + m) to "second source" position in TS.2.

6.2 Add 'm' in the timing position to TS2.

6.4 /

\+ Beginning of outer cycle.

x Beginning of inner cycle.

6.4 Form the instruction (0's - (A + m)→9 s m+1 9(31)) from 9_{29} and put it in 9_{30}.

9.30 Obey the instruction formed by 6_4 i.e. send - $(A + m)_{m+1}$ to T.S.9 (i.e. - the coefficient $a_{m+1,\ m+1}$)

9.31 Send B to TS2.

7.1 Put the number '20' in TS4

7.4 Plant link for SCAPRO in TS5. The link is (9 + 0's→2 d 12 7(11)) which returns the control to instruction 7_{11} of the main table.

7.7 Put 'n' in TS3 and go to the entry of SCAPRO

These instruction together with 8_{30} put
(A + m) in TS1
B in TS2.
n in TS3.
20 in TS4
LINK in TS5
TS4 gives adjustment of binary point.

SCAPRO is now obeyed (and produces the scalar product in TS4) and the effect of obeying its link is to put $-a_{m+1,m+1}$ in TS2 followed by

7.11 Send the number 20 to T.S.3

7.14 Plant link for DIVID 1 in T.S.9. The link is (4-11→11 d 0 7(23)) which returns to control to instruction 7_{23} of the main table.

7.17 Put scalar product result into TS1 and go into DIVID 1.

These instructions put Scalar product in TS1
$-a_{m+1m+1}$ in TS2.
20 in TS3
LINK in TS9

DIVID 1 is now obeyed and produces the new value of x_{m+1} in TS4. The effect of obeying the link is to put the difference between this new value of x_{m+1} and its old value in TS11 and return the control to 7.23.

7.23 Discriminate if this number in TS11 to + ve or - ve, go to 7(25) if + ve, and to 7(26) if -ve.

7.25) Form P2 - difference between old and new x_{m+1} in TS11 and go to 7_9

7.26) Form P2 + difference between old and new x_{m+1} in TS11

7.9/

7.9 Stimulate TC1 if TS11 contains negative number (i.e. if difference between old and new x_{m+1} is greater than 2^{-19})

7.30 Obey the instruction which was formed by 7.31 i.e. send TS.4 (new x_{m+1}) to B_{m+1}

8.31 Add 1 to TS10.

6.1 Test whether TS10 now contains 'n'; if not return to 6.17 to repeat inner cycle with next value of m; if TS10 contains 'n' however go to 6.16 below.

6.16 Test if T.C.1 is on. If so, another outer cycle is required; we go to 7_{19} which clears T.C.1. and then return to 7_0. If not, the process is finished and we go to the link of the main table.

D.L.6

```
0     1 ▷ 13→2  i  1  6(2)(16)
1     10≠14 → DISC  d  13  6(17)
2     10 + 2→2  i  1  6(4)
3
4     INST + 2→DL.9 d  24 9(30)
5
6
7
8
9
10
11
12
13
14
15
16    TC1 + 0's→DISC i 1 7(18 19)
17    13 ▷ 5→1  d  2  6(21)
18
19
20
21    13 ▷ 21→2  i 1 6(23)
22
23    10 + 2→2  i 1 6(25)
24
25    10 + 1→1  i 1 6(27)
26
27    INST + 1 → D.L.6  d  1  6(30)
28
29    (0 + 0's →11 s 1 6(31))
30    ----------
31    INST + 2→DL7 d 29 7(30)
```

D.L.7./

	D.L.7.	D.L.8	D.L.9
0	0's + 0's→10 d 15 3(17)		
1	INST + 0's→4 d 1 7(4)		
2			
3	(20)		
4	INST + 0's→5 d 1 7(7)		
5			
6	(9 + 0's→2 d 12 7(11))		
7	14 + 0's→3 d 30 1(7)	into SCAPRO	
8			
9	P40 & 11→T.C1 d 19 7(30)		
10			
11	INST + 0's→3 d 1 7(14)		
12	13 ▷ 21→1 d 1 7(15)	ENTRY	
13	(20)		
14	INST + 0's→9 d 1 7(17)		
15	INST + 1→DL7 d 4 7(21)		
16	(4-11→11 d 0 7(23))		
17	4 + 0's→1 d 8 5(27)	into DIVID 1	
18	15 + 0's→TSi i 1 T.Si	LINK	
19	P1 + 0's→$TC1_{off}$ d 11 7(0)		
20	(0's + 0's→0 i 32 7(22))		
21	----------		
22	INST + 1→D.L.7 d 4 7(28)		
23	P40 & 11→DISC d 1 7(25/26)		
24			
25	INST - 11→11 d14 7(9)		
26	INST + 11→11 d13 7(9)		
27	(ones ▷ 20→0 d 2 7(0))		
28	----------		
29	(0's + 0's→0 s 1 7(31))	(4 + 0's→0 s 1 8(31))	(0's-0→9 s 1 9(31))
30	----------	12 + 10→1 i 1 6(0)	----------
31	INST + 2→DL7 d 29 8(30)	P1 + 10→10 i 1 6(1)	13 + 0's→2 i1.7(1)

Problem 5. The solution of a set of linear algebraic equations by pivotal condensation.

The method of "pivotal condensation", or 'elimination', used in this report may be described briefly as follows.

Consider the solution of the set of n equations given by

$$b_{m+1} = a_{m+1,1}\ x_1 + a_{m+1,2}x_2 + \cdots + a_{m+1,n}x_n \quad m=0,1,..,n-1.$$

The variable x_n may be eliminated from the last (n-1) equations, using the first equation, by multiplying the first equation by $\frac{-a_{m+1,n}}{a_{1n}}$ and adding it to the m^{th} equation for each of the values of m. This will leave us with the first equation unaltered and (n-1) equations which contain $x_1\ x_2 \cdots x_{n-1}$ only. This process may be described as 'reducing' a set of n equations in n unknowns, to a set consisting of one equation in the complete n variables, and (n-1) equations in (n-1) variables. In this reduction the first equation is the 'pivotal row' and the element, a_{1n}, is the 'pivot'. The set of equations in the 'n-1' unknowns $x_1 \cdots x_{n-1}$ may now be reduced to a set consisting of one equation in the variables $x_1 \cdots x_{n-1}$ and (n-2) equations in the variables $x_1 \cdots x_{n-2}$. By repeating the reduction process we finally obtain a single equation in the single unknown x_1. If we collect together all the equations which have formed the successive pivotal rows, together with the final equation which contains x_1 only, we have a set of the following type.

1 equation containing $x_1 \cdots x_n$
1 equation containing $x_1 \cdots x_{n-1}$
1 equation " $x_1 \cdots x_{n-2}$
.....
.....
1 equation containing x_1.

Such/

Such a set is called a 'triangular' set. From the last equation x_1 can be evaluated and, using this value of x_1, x_2 can be calculated from the last one. Similarly all the values of the unknowns can be obtained in succession from these n equations. The above description gives the mathematical basis of the method of solution only. In practice a number of difficulties arise when using this method. (REF.(1), (2) and (3). The first is due to the fact that in computational processes it is necessary to represent each number by a finite number of digits. In so far as numbers arising during the computation do not admit of an exact representation in this form it is necessary to use a truncated version of their digital representation with the result that errors are introduced into the computation. In the method of pivotal condensation it is possible for the accumulation of these truncation errors to lead to serious inaccuracies in the final solution. The second difficulty which arises is that for certain sets of equations, usually called "ill conditioned" equations, the coefficients of the derived equations have a progressively diminishing number of significant figures; this again gives a loss of accuracy. A third difficulty, is that of ensuring that all the numbers obtained during the computation remain within the range permitted by the storage space they are allotted: This difficulty is, of course, common to all problems which are programmed for an automatic computing machine, but it arises in a rather more acute form with this particular problem than for most.

It is possible to produce an instruction table which will overcome all these difficulties satisfactorily, but since it involves very lengthy

considerations/

considerations it is felt that its inclusion in this introductory report would scarcely be justifiable. It will therefore be given in a subsequent report. An unsatisfactory feature of such a table however, is that in order to guard against all contingencies, it is very much longer than is necessary for most sets of equations, both in respect of the number of instructions it contains and the time it requires to obtain a solution.

The table given in this report has been designed to give a solution, of any desired accuracy, very quickly for any set of equations which does not behave "too unreasonably". The meaning of "too unreasonably" will appear in the subsequent discussion.

It is interesting to note that the table given below would have been more than adequate for the solution of all sets of equations, many of them very ill-conditioned and of orders up to twenty, which we have had to solve in the Mathematics Division of the National Physical Laboratory by direct methods. (This excludes rather special sets of equations of orders much higher than 20 which arise when solving elliptic partial differential equations by finite different methods and sets arising in survey problems. These have always been solved by Relaxation or by iteration methods and therefore their behaviour when treated by direct methods such as pivotal condensation has not been observed. In general, direct methods have been used on those equations which did not yield readily to relaxation, and hence those solved by direct methods have nearly always been of an ill-conditioned type).

The instruction table as it stands solves sets of equations of order ≤ 31 and provides a solution corresponding to one set of values for the constants $b_1 \ldots b_n$. It can be easily extended to sets of orders greater than 31 by adding a few instructions, and a simple modification would make it capable of solving the set of equations for any number of sets of values of the constants $b_1 \ldots b_n$; in particular it could be made to invert a matrix by taking as values of $b_1 \ldots b_n$ the n sets (1, 0 0 0 .. 0), (0, 1, 0000)...(00...01). respectively.

The/

The process used is as follows.

The coefficients b_{m+1}; $a_{m+1,1}$; $a_{m+1,2}$; ... ; $a_{m+1,n}$ are stored in positions $D.L(A + m)_{0-n}$, for m = 0,1, ... n-1. It is assumed that the greatest coefficient in any row of the equations is given to an accuracy of 10 decimal digits. (if this was not true of the original equations it is assumed that, where necessary, equations have been multiplied by an appropriate power of ten in order to satisfy this condition) so that the most significant digit in each binary equivalent will be in position Pa where $a \leqslant 34$. These coefficient are regarded as integers (the position of the binary point is of no importance provided it is the same in all coefficients). For the purpose of finding the residuals corresponding to the first solution obtained, a copy of the original coefficients is made in positions $(C + m)_{0-n}$ m=0,1,...n-1. using the routine COPY. The variables x_n, x_{n-1}, ... x_2 are now eliminated successively.

The elimination of x_n is performed as follows.

The equation which contains the numerically greatest coefficient of x_n is located (using ROUTINE MAXMOC) and the coefficients of this row are then interchanged with those of the first equation using routine INTERC. We thus obtain the coefficients of the original set of equations in the same storage position but arranged so that the equation with the maximum coefficient of x_n is in D.L.A. This row is now used as the pivotal row; and x_n is eliminated from each of the other rows.

This is done by repeating the following process 'n-1' times with m=1,2 .. n-1 successively. (In what is written below it is assumed that the coefficients were renamed after the interchanging so that the coefficients in the top row are still called b_1, a_{11}, a_{12}, .. a_{1n})

(i) Calculate $-a_{m+1,n}/a_{1n}$ to 38 binary places using routine DIVID 2 (each of these values is numerically less than or equal to Unity since a_{1n} is the greatest coefficient of x_n)

(ii) Multiply the first row by this number, using routine SCAVEC, and add it to the row in which x_n is being eliminated, using vector addition.

When/

When this is completed the D.L's A, A + 1, ... A + n-1 will contain the coefficients of n equations, of which that in A will contain the n variables $x_1 \ldots x_n$ and those in the other delay lines will contain the n-1 variables $x_2 \ldots x_n$. It will now be seen that these last (n-1) equations and their positions in the memory will bear the same relation to the numbers A + 1 and n-1, as the original set bore to the numbers A and n. A reduction can therefore be performed on these in which x_{n-1} is removed from (n-2) of these equations, using that one of them which contains the largest coefficient in x_{n-1} as the pivotal row and placing this row in the D.L (A + 1) by the interchange process.

If we define the original reduction which eliminates x_n as a reduction with respect to A and n, then the complete reduction will consist of

(1) A reduction with respect to A, and n

(2) " " " " " A + 1 and n-1

(n-1) A reduction with respect to A + n-2 and 2.

We shall be left finally with an array of numbers in

$(A)_{0-n}$; $(A + 1)_{0-(n-1)}$; $(A + 2)_{0-(n-2)}$; ... ; $(A + n-1)_{0-1}$ corresponding to a triangular set of equations in $x_1 \ldots x_n$; $x_1 \ldots x_{n-1}$; $x_1 \ldots x_{n-2}$; ... ; x_1 respectively.

The various multipliers of which $-a_{m+1,n}/a_{nn}$ is typical will alway lie between -1 and +1 and will therefore be representable to 38 binary places without difficulty. The coefficients of the derived equation could, however, give trouble. Since at each stage we are multiplying the coefficients of one equation by a number less than or equal to unity and adding it to another equation, after m stages of the reduction it is possible for the derived equations to have coefficients which are greater than those of the original equation by a factor 2^m. To avoid the possibility of these coefficients becoming greater than 2^{39} it

would/

would be necessary to check on their size after each reduction and if they had reached 2^{38}, divide the equations by 2 or use some other similar device. However, in nearly all practical problems the difficulty which arises is that the successive coefficients become smaller, not greater, and therefore for this comparatively elementary table it is assumed that it is not necessary to diminish the coefficients in this way. Since the original coefficients had at most 34 significant digits only, no difficulty will arise unless an increase by a factor of greater than 2^5 is obtained. Similarly it is assumed that the coefficients do not become so small that the last pivot has no significant figures. These assumptions have been satisfied in all sets which have been solved at N.P.L., though of course, it is not difficult to construct rather special sets for which the assumptions would not be true.

The values $x_1, x_2, \ldots x_n$ are next calculated successively from the triangular set of equations.

Suppose we have already calculated $x_1 \ldots x_k$ and we wish to calculate x_{k+1}. The solutions $x_1 \ldots x_k$ have been calculated to 20 binary places (i.e. to 20 binary places after the binary point) and stored in B_{1-k} and the number -1.0 to 20 binary places is in B_o.

If we denote the equation in $x_1 \ldots x_{k+1}$ by

$$d_{k+1} = c_{k+1,1}\ x_1 + c_{k+1,2}\ x_2 + \cdots + c_{k+1,k}\ x_k + c_{k+1,k+1}\ x_{k+1}$$

Then $x_{k+1} = -d_{k+1} + c_{k+1,1}\ x_1 + c_{k+1,2} x_2 + \cdots + c_{k+1,k}\ x_k / c_{k+1,k+1}.$ (2)

Since $d_{k+1}; c_{k+1,1}; c_{k+1,2}; \cdots ; c_{k+1,k}$: are stored in D.L $(A + n-k-1)_{0-k}$

and $-1{\cdot}0;\ x_1;\ x_2; \ldots; x_k$ are stored in D.L$(B)_{0-k}$ to twenty binary places, the numerator of the expression for x_{m+1} given in equation (2) may be calculated by using SCAPRO for a scalar product of order (k+1). x_{k+1} can then be calculated from the double length value of the scalar product by using DIVID 2. Since each of the solutions is calculated to 20 binary places, and only one word is allotted

for/

for its storage, values of the variables up to 2^{19} are permissable. If any of the solutions is greater than this the table gives a failure indication. The table could be made to give a continuous adjustment so that if it "failed" for solution of 20 binary places it attempted to find solution to say 16 places and so on. Alternatively it could be made to calculate for itself exactly how many binary places were permissable and to adjust the binary point in the scalar product etc. accordingly. In this table we have assumed that it is known for physical or mathematical reasons that the solutions are unlikely to be greater than 2^{19} and therefore that a failure indication is required if this is not true.

Another point where difficulty might arise is in the formation of the scalar products themselves because any of the partial sums had become too large. This danger is guarded against by SCAFRO itself which provides a failure indication whenever this happens. As a result of "rounding off" errors, the values of $x_1 \ldots x_n$ obtained will probably not be correct to the full 20 binary places and it may be necessary to derive a more accurate solution. This may be done by substituting the values of x obtained originally in the set of equations and deriving the residuals. If the set is now solved again with the residuals multiplied by a scale factor in place of $b_1 \ldots b_n$, a correction to the original solutions is obtained. This process may be repeated indefinitely to give any desired degree of accuracy. Hence when all the values of $x_1 \ldots x_n$ have been calculated they are substituted in the copy of the original equations to find the residuals. Assuming still that the coefficient of the original equations are integers the residuals are calculated to 20 binary places (i.e. they are calculated exactly since the coefficients are integers and the solutions have been calculated to 20 binary places). The calculation of the residuals requires the use of the routine SCAFRO. The values $-1\cdot0, x_1 \ldots x_n$ are in D.L. B_{0-n} and the scalar products of order $(n + 1)$ of this vector with $(A + m)_{0-n}$ for $m = 0, 1, \ldots m-1$, give the n residuals. Hence it is intended that the equations should now be solved again using the values of the residuals in the

place/

place of the original values $b_1 \ldots b_n$ it is necessary that the residuals obtained should have no more than 34 significant binary figures. Therefore as each residual is formed, a test is made to see if it exceeds 2^{14} numerically. If any residual does exceed this value the residuals are recalculated to 18 binary places and then again to 16 binary places and so on. A count of the number of binary places finally retained in the residuals is kept in a blank space of the Instruction table used for solving the equations. It has been our experience that in general the residuals are "quite small" and usually it will be permissable to form them to 20 binary places, (i.e. exactly, since the original solutions were to 20 binary places). On a desk calculating machine it is possible to perform the second solution (i.e. the solution corresponding to the residuals) much more quickly than the original solution since in this second solution, all the coefficients of the variables in the reduced equations will be the same as before. To do this it is necessary to keep a record of the factors of the type $-a_{m+1,n}/a_{n\,n}$ which were used in the first reduction. It is possible that this is not worth while on an electronic computer it being more satisfactory to repeat the whole of the original reduction process, rather than going to the trouble of keeping a store of the multiplying factors and of making the second table appropriate to the modified procedure.

The above discussion reveals that the only routines used by the table are COPY, MAXMOC, SCAVEC, SCAPRO, INTERC and DIVID 2. Since the highest T.S. used by any of these routines is TS8 which contains the link of DIVID 2, the table may use T.S.9 and above. The tables use T.S.9 and T.S.10 for general purpose and therefore its parameters are in TS11, TS12, TS13 and TS14. They contain A,B,C and n respectively. The link is in T.S.15. The whole table may be used as a routine by another main table merely by providing the appropriate values of A, B, C and n and a link in TS12, 13, 14 and 15, and then entering the table. For all the previous tables the plan of instructions has been given in considerable detail. This has been done to make it easier for the reader to understand the

function/

function of the table. After a little experience it becomes unnecessary for the person coding a problem to go through such elaborate preparations; the type of procedure usually adopted should be clear from the following, though this is still somewhat more elaborate than is usually necessary. A plan of the following type is first drawn up.

Copy $(A + m)_{o-31}$ in $(C + m)_{o-31}$ for m = 0, 1, ..., n-1. (routine COPY).

Clear TS10 which will contain i during that part of the triangular reduction when x_{n-i} is being eliminated.

(A) Test if i = n-1 → YES Reduction to triangular form is complete. Proceed to (D)

↓ NO Find the position of that number which is numerically greatest of those stored in

$(A + i)_{n-i}$ ... $(A + n-1)_{n-i}$. If it is Y_{n-i} put Y in TS3 (routine MAXMOC).

Interchange D.L.s $(A+i)_{o-31}$ and Y_{o-31} (routine INTERC).

Send P_1 to TS9, which will contain j during the single reduction corresponding to one value of i, with j running from 1 to n-i

(B) Test if j = n-i → YES. Reduction with this value of i complete. Proceed to (C).

↓ NO Form $2^{38}\, p_{,j}$ = -no in $(A + i + j)_{n-i}$ x 2^{38}/No. in $(A+i)_{n-i}$ in T.S.4 (we know $|P_{ij}| \leq 1$) using DIVID 2

Form p_{ij} x vector in $(A + i)_{o-(n-i-1)}$ in D.L. 200 using SCAVEC. Add D.L 200_{o-31} to $(A+i+j)_{o-31}$

Add P_1 to TS9 i.e. replace "j" by "j + 1" and return to (B)

(C) Add P_1 to TS10 i.e. replace "i" by "i + 1" and return to (A)

(D) Clear TS10. We are now about to determine the values of x_1 ... x_n

When x_{k+1} is being determined TS10 contains k.

Send -1(x 20^{20}) to B_o i.e. -1.0 to 20 binary places.

(E) Form/

(E) Form scalar product of order $(k + 1)$ of B_{0-k} and $(A + n - 1 - k)_{0-k}$ in the Acc and divide result by the number in $(A+n-1-k)_{k+1}$; thus form $2^{20}\, x_{k+1}$ and send to B_{k+1} (Using SCAPRO and DIVID 2). Test if $|x_{k+1}| \leqslant 2^{19}$ and if not give a failure indication.

Add P1 to TS10 i.e. replace k by k+1
Test if new value of k=n

NO Return to (E)
YES Clear TS10 which in future will contain '1'

when the residual corresponding to the (1+1)th equation is being formed.

(F). Form $2^{y}x$ scalar product of order $(n+1)$ of B_{0-n} and $(C+1)_{0-n}$ in TS4. Take y=0 initially and test each residual as formed to see if it exceeds 2^{14} numerically (when y=0 the residuals have 20 binary places). If any residual is too great start again with y = -2 and so on (y = 4, -6 etc.) The final value of y plus 20 is stored among the instructions of the table in $(DL16)_{12}$.

Send TS4 to $(A+1)_0$

Add P1 to TS10 i.e. replace 1 by 1+1

Test if 1=n

NO Return to (F)

YES All residuals are in $(A)_0$ $(A+1)_0$ ... $(A+n-1)_0$ and we could arrange to solve again.

From the above plan the details of the instructions are drawn up as follows, while the appropriate instructions are written on sheets with rows numbered 0 to 31 as in the final table.

16.3 11 → 1 A to COPY) Planting parameters and
16.5 13 → 2 C to COPY)
15.8 Inst→4 Plant link for COPY) link in COPY.
15.15 14 → 3 n to COPY and enter ENTRY of COPY.)
T.S.I (link of COPY) 0's → 10 (original value of i is 0)
16.17 14-P1 → 1 (n-1) ←
16.19 10≠1 → DISC

OFF → To 15.26

ON

15.27 11+10 → 1 (A+i) to MAXMOC) Planting parameters
15.2 14-10 → 2 (n-i) to MAXMOC) and link in
15.5 INST → 8 Plant link for MAXMOC)
15.12 2 → 3 (copy of n-i in 3) and go into MAXMOC) MAXMOC.
TSI (link of MAXMOC) 3 → 2 (send the result of MAXMOC to TS2 for use in INTERC)
16.25 INST → 3 Plant link in INTERC)Planting parameters
17.3 11+10→1 (A+i) to INTERC and go into INTERC)and link in INTERC
TSI (link of INTERC) Send P1 to 9 (first value of j is 1))
16.11 14-10 →1 (n-i)
16.13 9≠1 → DISC

OFF (next value of i needed)
10.15 P1+10 →10 (i+1 to i) ——→

ON

16.16 11+10→ 5 (A+i)
16.18 5+9→3 (A+i+j)
16.20 5 ▷13→5 (A+i) in 2nd source position
16.22 3 ▷13→3 (A+i+j) in 2nd source position
16.24 3 ▷8→4 (A+i+j) in destination position
16.28 3+4→4 (A+i+j) in destination and 2nd source positions
16.2 1+5→5 (n-i) in timing position, (A+i) in 2nd source position
16.4 1+3→3 (n-i) in timing position, (A+i+j) in 2nd sce posn.
16.6 (INST+3→DL15
15.31 (0's+0→6 s -2 15(1)) Form $(A+i+j)_{n-i}$ → 6 and obey
15.0 - - - -
15.1 (INST→2
15.3 (38 Send 38 to 2 (For DIVID2)
15.4 INST→8 Plant link for DIVID 2
15.7 INST+4→$DL16_{10}$ i.e. plant 200+(A+i+j) → A+i+j in $DL16_{10}$

15.9 (200 + 0→0 i 32 16(11)) (obeyed later at 16_{10})
15.10 INST+5→DL16
16.31 (0's+0→1 s -2 16(1)) Form $(A+i)_{n-i}$ →1 and obey
15.0 - - - - - (parameter for DIVID2)
16.1 6→Acc i.e. $(A+i+j)_{n-i}$ to Acc and into DIVID2

TSI (link of DIVID2) -4 →2 i.e. $2^{38}p_{ij}$ (defined above) for SCAVEC)
15.18 11+10→1 (A+i) (for SCAVEC)
15.20 14-10→3 (n-i) (for SCAVEC)
15.22 INST→8 Plant link for SCAVEC
15.25 P1▷1 → 4 i.e. P2→4. This is because the products obtained
by SCAVEC need to be shifted two places to the left.
Go into SCAVEC.
TSI (link of SCAVEC) P1+9 →9 (j+1 → j)
16.10 Obey instruction formed by 15.7, i.e. 200+(A+i+j)→(A+i+j) i 32.
(vector addition) and return to 16.11

From 16.19/

From 16.19 Reduction is now complete.

15.26 12 ▷ 21 → 9 B in destination position
17.6 (INST+9 → DL17 Form (Ones ▷ 20 → B d 3 17(0))
17.26 (Ones ▷ 20 → 0 d 3 17(0) and obey it. This puts -1.0
17.27 (------ to 20 binary places in B_0.

17.0 Clear 10 (will contain k when x_{k+1} is being evaluated)

17.5 9+10 → 1 (k in timing position, B in destination position)
17.7 INST+1 → DL17 (form the instruction $4 \rightarrow B_{k+1}$ and send it to $DL17_{30}$ to be obeyed later)
15.30 12 → 1 (B)
17.1 11+14 → 2 (A+n)
16.7 2-P1 → 2 (A+n-1)
17.9 2-10 → 2 (A+n-1-k)
17.11 2 ▷ 13 → 4 (A+n-1-k) in 2nd source position
17.13 4+10 → 4 (A+n-1-k) in 2nd sce. position and k in timing posn.
17.15 10 → 3 (k)
17.17 INST+4 → $DL18_{30}$ Form the instruction $-(A+n-1-k)_{k+1} \rightarrow 1$ and put it in $DL18_{30}$ to be obeyed later
16.30 INST → 5 Plant link for SCAPRO
16.9 Clear TS4 (The scalar product table will require a shift of zero places)
Go into SCAPRO
TS1 (link of SCAPRO) (INST → 2
17.23 (40 i.e. send 40 to 2
17.24 Ones ▷ 39 → Ac_+ taking place for two minor cycles; the first an even minor cycle and the next an odd. This adds a double length number consisting of 39 zeros in the less significant positions and 41 ones in the more significant positions. The addition of this number cancels the 'round off'.

16.27 INST → 8 Plants link for DIVID2
18.30 Obey the instruction formed by 17.17, i.e. $-(A+n-1-k)_{k+1} \rightarrow 1$
18.31 Go into DIVID2
TSI (link of DIVID2) P1+10 → 10 (k+1 → k)
17.30 Obey instruction formed by 17.7, i.e. $4 \rightarrow B_{k+1}$

17.31 10 ≠ 14 → DISC
↓ ↘ ON k ≠ n and therefore return to 17.5
OFF $x_1 \ldots x_n$ have been evaluated. We now find residuals.
17.4 (INST → 9
18.9 (-20) Send (-20) to 9
18.10 20 → $DL16_{12}$

18.12 Clear TS10 (TS10 will contain 'l' when the 'l+1'th residual is being formed)
15.17 12 → 1 (B)
15.19 13+10 → 2 (C+l) (for SCAPRO)
15.21 INST → 5 Plant link for SCAPRO
17.29 16_{12}+9 → 4 i.e. initially 0 is sent to TS4 (for SCAPRO)

17.12/

```
17.12   14 → 3        'n' (for SCAPRO)
                      Go to SCAPRO
TSI   (link of SCAPRO)  (INST ≠ 9 → DISC
18.2                    (-20
                                        ↘
                                         ON   Go to 18.4
                 ↓
                OFF
18.3   4   →   DISC  ↘ ON  Go to 18.6
                 ↓
                OFF
18.5   P40+Acc → 4
18.4   P1 ▷ 34 → 1
18.7   P40&4 → DISC
            ↓         ↘ ON  18.14   4+1 → 1
           OFF
18.13  4-1 → 1            These instructions test if
17.16  P40+1 → 1             residual < 2^34 if residual is + ve.
17.18  P40&1 → DISC       and - residual ≤ 2^34 if residual is -ve.
                 ↓     ↘ ON  17.22  Go to 18.6
                OFF          18.6   (INST → 9
17.21  11+10 → 1   (A+1)     17.19((20)          Send 20 to TS9
16.23  1 ▷ 21 → 1            17.20( DL16_12+INST → DL16_12
         (A+1) in dest.            (                       Subtract 2
               position      18.11((-2)                    from DL16_12
17.25(INST+1 → DL18   )
18.27(4 → 0  d  2  18)    Form   4 → (A+1)_0, send it
18.28(Obey .          )    to 18.28 and obey it
18.0   P1+10 → 10 (1+1    1)
15.13  10 ≠ 14 → DISC
            ↓        ↘ ON  Return to 15.17 to find next residual
           OFF
15.16  15 → TSI   Send link to TSI
```

The instructions for finding the residuals call for some explanation. The residuals are found by using the routine SCAPRO and on a first attempt their values to 20 binary places after the binary point are calculated. If this produces residuals which are too great, then the values to 18,16,... binary places are found in succession until they become of permissible size. As a result of this it would be necessary to make the parameter which controls the binary point in SCAPRO equal to 40,38,36,... successively. Unfortunately SCAPRO can only deal with parameters less than 40 and therefore the first case has to be treated differently from the others. The first time round when we attempt to find the residuals to 20 binary places, the parameter in SCAPRO is given the value zero instead of 40. The residual is therefore in the least significant/

significant 40 digits of the accumulator (it may overlap into the upper 40 digits if it is large enough). The instruction in DL18.3, which is only obeyed when the attempt is being made to calculate the residuals to 20 binary places, tests whether the value supplied by SCAPRO in TS4 is zero. It will only be zero if the residual is less than 2^{19} and therefore if TS4 is not zero, it will certainly be necessary to calculate the residuals to less than 20 binary places. The control therefore obeys 18.6 next if TS4 is not zero and this gives a reduction of 2 in the number of binary places to which the residuals are calculated. In all subsequent attempts which are necessary, the routine SCAPRO has parameters 38,36 etc., and therefore it produces the required part of the accumulator without special precautions.

Solution/

SOLUTION OF EQUATIONS BY PIVOTAL CONDENSATION.

D.L.15

```
1   INS + 0's →2 d 1 15(4)
2   14-10→2 d 1      15(5)
3   (38                  )

4   INST + 0s →8 d 1  15(7)
5   INST + 0's→8 d 5 15(12)
6   0-4→2    i   1   15(18)
7   INST + 4 →D.L16 d 1 15(10)
8   INST + 0's →4 d 5 15(15)
9   200 + 0→0  i 32  16(11)
10  INST + 5→D.L16 d 20 16(0)
11  3 + 0's→2 d 1    16(25)
12  2 + 0's→3  d 5   11(19)
13  10≠14→DISC d  1  15(16)(17)
14  0's + 0's →10 d 10 16(17)
15  14 + 0's →3 d 15  10(0)
16  15 + 0's →TSI i 1  TSI
    ---------------------
17  12 + 0's→1 i 1 15(19)
18  11 + 10→1 i 1  15(20)

19  13 + 10 →2  i 1   15(21)
20  14-10→3  i  1  15(22)
21  INST + 0's →5 d 6 17(29)
22  INST + 0's →8  d 1 15(25)
23
24  P1 + 9→9  d  3   16(10)
25  P1 ▷ 1→ 4  d  1   14(28)
26  12 ▷ 21→ 9  d  10   17(6)
27  11 + 10 →1 d 5  15(2)
28
29  (4 + 0's →0 s 1 17(31))
30  12 + 0's → 1 d 1 17(1)
31  (0's + 0→6 s - 2 15(1))
0   ---------------------
```

D.L.16

```
6 + 0's→Acc d 20 13(23)
1 + 5→5   i   1  16(4)
11 + 0's→1 i  1  16(5)
                 ENTRY
---------------------
1 + 3→3  i   1   16(6)
13 + 0's→2 d 1   15(8)
INST + 3→D.L15 d 24 15(0)
2 - P1→2  i  1   17(9)
INST + 0→2  d 25 17(24)
0's + 0s'→4 d 28  1(7)
---------------------
14-10→1   i 1    16(13)
---------------------
9≠1→ DISC i 1 16(15)(16)

P1 + 10 →10 i 1 16(17)
11 + 10 → 5  i 1 16(18)

14 - P1 →1 i 1  16(19)
5 + 9→3   i  1  16(20)
10≠1→DISC d 5 15(26)(27)
5 ▷ 13→ 5  i  1 16(22)

3 ▷ 13→3  i  1  16(24)
1 ▷ 21→1  i  1  17(25)
3 ▷ 8→ 4  d  2  16(28)
INST + 0's →3  d  8   17(3)

INST + 0s'→8 d 1  18(30)
3 + 4 → 4  d  4   16(2)
(0's - 0 →1   s  1  18(31))
INST + 0's→5 d 9 16(9)
(0's + 0→1    s - 2 16(1))
-----------------------
```

Solution/

SOLUTION OF EQUATIONS BY PIVOTAL CONDENSATION. (Contd).

	D.L.17	D.L.18.
1	11 + 14→2 d 4 16(7)	
2	P1 + 0's→9 d 6 16(11)	(-20)(5)
3	11 + 10 → 1 d 2 10(7)	4 + 0's→DISC i 1 18(6)
4	INST + 0's→9 d 4 18(10)	P1 ▷ 34→1 d 1 18(7)
5	9 + 10→1 i 1 17(7)	P_{40} + Acc→4 d 29 18(4)
6	INST + 9→D.L17 d 19 17(27)	INST + 0's→9 d 12 17(20)
7	INST + 1→D.L17 d 21 15(30)	P40 & 4→DISC d 4 18(13)(14)
8		
9	2 - 10→2 i 1 17(11)	(-20)
10		0's - 9→D.L.16 i 1 18(12)
11	2 ▷13→ 4 i 1 17(13)	(-2)
12	14 + 0's→3 d 25 1(7)	0's + 0's→10 d 3 15(17)
13	4 + 10→4 i 1 17(15)	4 - 1→1 d 1 17(16)
14		4 + 1→1 d 2 17(18)
15	10 + 0's→3 i 1 17(17)	
16	P40 + 1 →1 i 1 17(18)	
17	INST + 4→D.L18 d 11 16(30)	
18	P40 & 1→DISC d 1 17(21)(22)	
19	(20)	
20	D.L 16 + INST→DL16 d 22 18(12)	
21	11 + 10→1 i 1 16(23)	
22	255 + 0's→255 d 14 18(6)	
23	(40)	
24	1's ▷ 39→Acc$_+$ i 2 16(27)	
25	INST + 1→D.L18 d 1 18(28)	
26	(ones ▷ 20→0 d 3 17(0))	
27	----------	(4 + 0's→0 d 2 18(0))
28	(INST≠9→DISC d 24 18(3)(4))	----------
29	D.L16 + 9→4 d 13 17(12)	(P1 + 10→10 d 12 17(30))
30	----------(4)	----------
31	10≠14→ DISC d 3 17(5)	255 + 0's→255 d 22 13(4)
0	0's + 0s' →10 d 3 17(5)	P1 + 10→10 d 11 15(13)

References.

(1) L. Fox, H.D. Huskey and J.H. Wilkinson. Notes on the solution of linear simultaneous algebraic equations. Shortly to be published in the Journal of Mechanics and Applied Mathematics.

(2) A.M. Turing. Rounding off errors in Matrix Processes. Shortly to be published in the Journal of Mechanics and Applied Mathematics.

(3) J. von Neumann and H.H. Goldstine. Numerical Inverting of Matrices of a High Order. Bull. Amer. Math. Soc. Vol.53. No.11. Nov.'47.