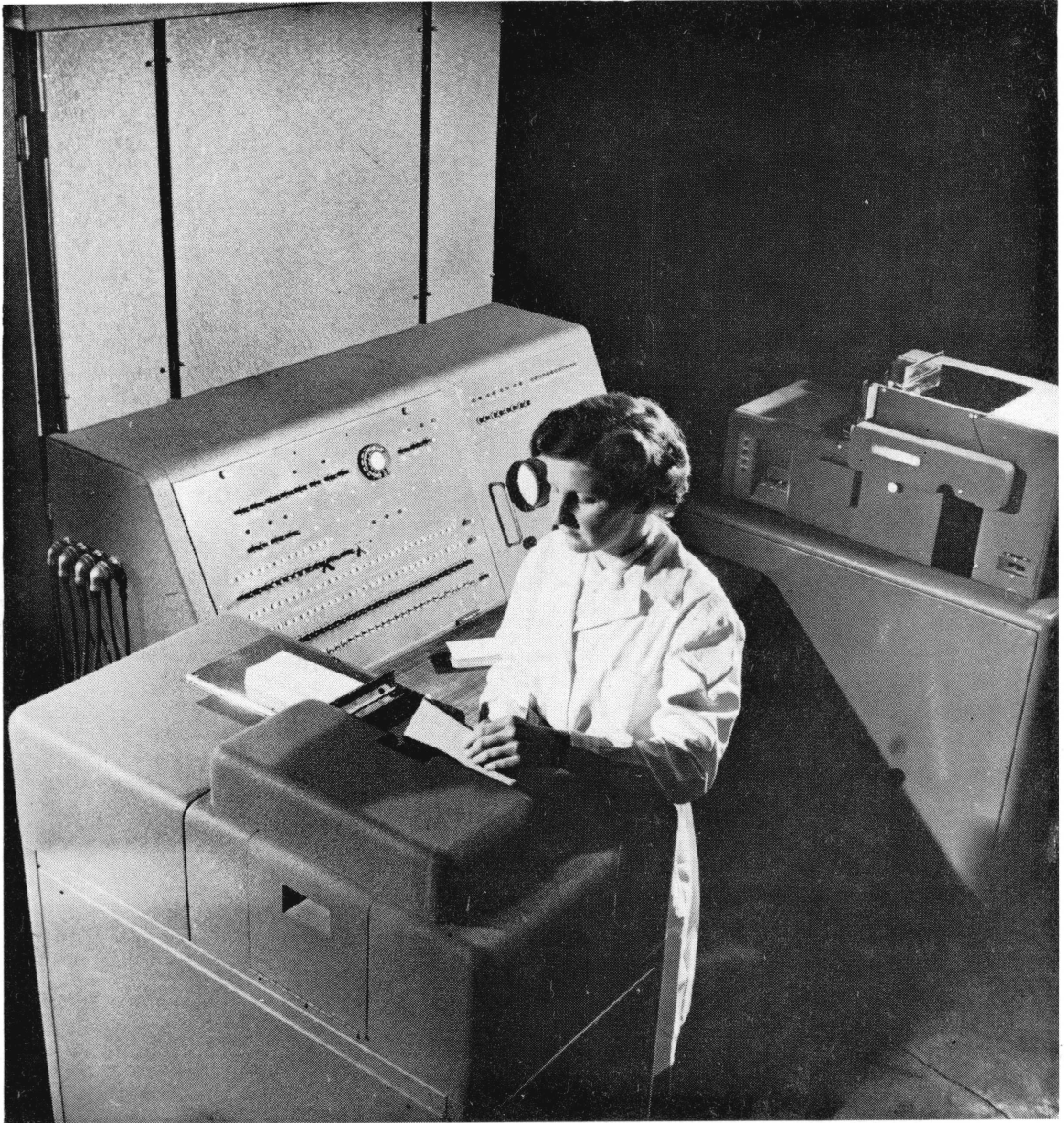


'ENGLISH ELECTRIC'

D.E.U.C.E. PROGRAMMING MANUAL

The ENGLISH ELECTRIC Company Limited



C O N T E N T S

1. INTRODUCTION
 2. REPRESENTATION OF NUMBERS
 3. METHOD OF STORAGE
 4. PROGRAMMING FOR 'DEUCE'
 5. THE INSTRUCTION WORD
 6. OTHER SOURCES AND DESTINATIONS
 7. SUB-ROUTINES
 8. INITIAL INSTRUCTIONS
- APPENDIX I - Hollerith Timing
- APPENDIX II - List of Sources and Destinations

INTRODUCTION

1. INTRODUCTION.

The DEUCE is a universal electronic digital computer capable of carrying out automatically and rapidly a sequence of arithmetical or other operations comprising a computation. In using the machine the problem to be solved is broken down into a series of elementary operations, which will be performed in the correct order by the machine. A 'programme', consisting of the appropriate instructions directing the machine to perform this sequence of operations, is fed into the computer together with the initial arithmetic data, and the computation then proceeds without further intervention from the operator.

The process of making out a programme may take a few weeks for a complicated problem, considerably longer than would be occupied in carrying out the steps individually on a desk calculating machine.

The utility of the DEUCE lies in the element of repetition, both of individual instructions within a programme and of numbers of computations of exactly the same type. In the solution of a set of simultaneous linear equations, for example, the process of eliminating one variable between two equations consists of subtracting a fixed multiple of the coefficients of one equation from the corresponding elements of the other. Only one copy need be made of the instructions dealing with a pair of coefficients; the required number of repetitions is specified by a few extra instructions and carried out automatically by the Machine. There is a further stage of repetition in eliminating a particular variable from all but one of the equations, and again in eliminating successive variables from successively smaller numbers of equations to reduce the set to triangular form. Thus the whole process is built up from one small group of instructions, with extra groups to ensure the correct number of repetitions at each stage.

Once a programme has been made, it is retained in permanent form on punched cards and may be run into the DEUCE in a few seconds. As a result of the two or three weeks originally spent in making a programme for the solution of a set of simultaneous linear equations, any set may now be solved in a matter of minutes.

An account is given in this report of the basic principles of construction of programmes for the DEUCE, reference being made also to such features of construction of the machine as are necessary to an adequate understanding of its use.

It must be emphasised that construction of programmes without subsequent testing on the machine is of little value, and for a programme of any size the testing and elimination of errors can be effectively carried out only by the programmer himself. However, the testing facilities and methods of tracing errors have not been described in this report, as a demonstration is thought to be much more satisfactory.

Reference is made to the library of 'sub-routines' or elementary programmes available for incorporation in the programmes of larger computations, but as this library is extensive and still expanding it is considered that a list of available sub-routines could not usefully be included in the report and is issued separately.

This manual is intended primarily for initial training purposes and therefore is not up to date with particular techniques as they are developed. Such techniques are recorded regularly in DEUCE NEWS.

The DEUCE was developed as a direct result of work at the National Physical Laboratory, Teddington, on the Pilot ACE. Consequently, many programmes made for use on that machine can be translated for DEUCE, if it is not considered worthwhile to revise them.

REPRESENTATION OF NUMBERS

2. THE REPRESENTATION OF NUMBERS

2.1. BINARY NOTATION.

In an electronic equipment it is inconvenient to represent digits in the decimal scale, as this requires the use of circuits having ten possible states. Such circuits can be produced, but compare unfavourably in reliability and economy of equipment with circuits having but two possible states. For this reason, the DEUCE, in common with many other computing machines, is constructed to operate in the binary scale. Programmes are, however, normally constructed in such a manner that the machine translates input data presented to it in decimal notation into binary form during the process of taking in such data, and in a similar manner converts the results of computations into decimal form before feeding out the solutions.

In the familiar decimal scale, the significance of a digit is multiplied by ten by each displacement to the left; ten different symbols are required for the digits from 0 to 9. In the binary scale each displacement to the left multiplies the significance of a digit by two. Here, only two symbols are needed, for the digits 0 and 1, since the number 2 appears as '10', and any one digit may be represented by the condition of an element having only two possible states. The successive natural numbers are written 0, 1, 10, 11, 100, 101, 110 etc.

Example.

The decimal number 185 means $1 \times 10^2 + 8 \times 10 + 5$. In binary this number is represented as 10111001, or

$$1 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2 + 1$$

The decimal number 13.25 means $1 \times 10 + 3 + 2 \times 10^{-1} + 5 \times 10^{-2}$.

In binary, this would be 1101.01, the last digit signifying 1×2^{-2} . It will be observed that any fraction whose denominator is not a power of two will recur when expressed in binary form.

More digits are in general required to express a number in binary form than in decimal. For large numbers, the ratio is approximately $\log_{10} 2$, or about 10:3.

In a desk calculating machine, the size of the number which can appear on any of the registers is limited to a fixed number of digits. This is true of any digital calculating apparatus. Throughout the DEUCE, the limit is 32 binary digits covering any number that can be expressed in nine decimal digits. If two numbers being added together are so large that there is a carry from the 32nd place and 33 digits are required to express the sum, the extra digit is lost, and only the last 32 digits of the true sum appear.

Example.

For the purpose of illustration, the number length will be taken as six binary digits. In this case, the sum of 59 and 43, both within the six-digit limit, is 102, which is outside the limit.

111011	59
101011	43
(1)100110	38

The answer given is less than the true sum by 64, the value of the missing digit.

Care must be taken in programming either to keep within the allowed limit or to give an indication of failure if it is exceeded (though it is always possible to extend the accuracy by arranging the programme to work with double-length numbers, or even more). The fact that extra digits generated in an addition are lost is not entirely a disadvantage; it is used in forming a convention in which both positive and negative numbers may be expressed. In the following paragraphs reference is again made to a machine having a six digit number length. The extension to a machine using 32 digit numbers will be apparent.

With six binary digits, it is possible to express 2^6 different numbers. These have so far been taken as running from 0 to $2^6 - 1$, represented respectively by 000000 and 111111. This convention ('Unsigned') is useful for some types of work, particularly in the theory of numbers. For general computation, however, both positive and negative numbers must be represented. No extra provision is made for indicating the sign of a number, so that with six digits a range of numbers more or less equally spaced about zero must be represented. The greatest possible positive number will be only half what it was, but an equal negative number will also be represented.

Several different "Signed conventions" are possible. The one used on the DEUCE allows exactly the same process of binary addition to be used as in the Unsigned convention; the Machine may thus be used for either signed or unsigned arithmetic, the result of a given operation being interpreted according to the convention being used in that particular programme.

2.2 SIGNED CONVENTION

- (a) The positive numbers from 0 to $\frac{1}{2} \cdot 2^6 - 1$ (= 31) are represented by the same binary numbers as in the Unsigned convention.
- (b) The negative numbers '-n', from $-\frac{1}{2} \cdot 2^6$ (= -32) to -1 are represented by the binary numbers which previously stood for " $2^6 - n$ " (= 64 - n).

Examples.

Binary Number	Unsigned Convention	Signed Convention
100000	32	-32
100001	33	-31
100010	34	-30
111101	61	- 3
111110	62	- 2
111111	63	- 1
000000	0	0
011111	31	31

Adding 1 to '111111' produces the sum '000000', since the first digit of the true sum is lost. This result is false in the unsigned convention, but true in the other, where "111111" represents "-1". Similarly, in the example given above, the first two numbers could represent "-5" and "-21", and the sum "-26", in which interpretation the result is correct.

A few more examples will be given of addition in the Signed convention.

000101 = 5	000101 = 5	000011 - 3
111011 = -5	111101 = -3	111011 - -5
000000 = 0	000010 = 2	111110 - -2

So far, numbers have been treated as integers. In dealing with fractional numbers, the same conventions of sign are applied, but the programmer must bear in mind the position of the binary point at each stage of the computation.

In the Signed convention, we are limited to numbers of 31 binary digits (with the single exception -2^{31}). This is still sufficient to cover any number of nine decimal digits. Since all positive numbers have the first digit 0 and all negative numbers the first digit 1, the most significant digit is known as the "sign digit", and is used as the criterion whenever the sign of a number has to be examined. To change the sign of a number, the ones and zeros are interchanged and 1 is added to the result.

Example.

Take a binary number, say	000101 =	5
Interchange ones and zeros	111010 =	-6
Add one to the result	111011 =	-5

That this result is true in general is apparent, since the sum of the first two rows is 111111, or -1; the sum of the first and third rows is therefore 0.

2.3. MULTIPLICATION AND DIVISION BY 2

Facilities are provided for multiplying or dividing a number by 2. This is mainly a matter of shifting the whole number a single place to the left or right respectively. A complication is introduced by the fact that these results are arranged to be correct in the Signed Convention, and false in the Unsigned convention if the two results differ.

Example.

Original		Multiply by 2		Divide by 2
000110	= 6	001100	= 12	000011 = 3
000011	= 3	000110	= 6	(a) 000001 = 1
111010	= -6	(c) 110100	= -12	(b) 111101 = -3
111101	= -3	(d) 111010	= -6	(e) 111110 = -2
100111	= -25	(f) 001110	= 14	(g) 110011 = -13
010111	= 23	(h) 101110	= -18	(i) 001011 = 11

(The decimal equivalents are derived in accordance with the Signed convention).

NOTES.

- (a, e, g, i.) If an odd number is divided by 2, the last digit is lost, giving an error of $\frac{1}{2}$.
- (b, e, g.) If a negative number is divided by 2, a digit "1" is automatically inserted at the beginning to give a result true in the Signed but false in the Unsigned convention.
- (c, d.) These results are also true for signed numbers and false for unsigned.
- (f.) This result is false in any convention; the original number was too big to be multiplied by 2 without exceeding the digit capacity.
- (h.) This is false in the Signed convention, for the same reason; in the Unsigned, however, it is true, since unsigned numbers go up to 63 instead of only 31.

2.4. MULTIPLICATION.

The DEUCE has an automatic multiplier. Two 32-digit numbers are multiplied together, giving a product which may have up to 64 digits, though usually only 32 of these digits are taken on to the next stage of the computation. The result as given is true in the Unsigned convention; corrections must be made if either of the factors is to be treated as a negative number. In the Signed convention for double-length numbers, the number '-n' is represented by " $2^{64} - n$ ".

2.5. DIVISION.

There is also an automatic divider. This gives the quotient of two 32-digit integers to 31 binary places. If the divisor and dividend are taken to represent fractions with two different numbers of binary places, the number of binary places in the quotient will differ from 31 by the difference between the number of binary places in the two input quantities. The divider works with dividend and divisor of either sign and in all four cases produces a correct, correctly signed quotient within the sign

convention. The quotient produced is single-length; the dividend must therefore not exceed the divisor in absolute value or the quotient will exceed the 32-digit capacity.

2.6. STATIC REPRESENTATION.

Although binary notation has been adopted principally to facilitate construction of reliable and economical electronic circuits, advantage is taken in other parts of the machine of the fact that a digit may be represented by a mechanical or electrical element having two possible states. These elements may, for example, be lamps, switches or holes punched in a card.

A single number which is required in a computation, such as the order of a set of linear equations, may be set up in binary form on a row of 32 lamps on the control panel, each of which is controlled by a key. Those lamps corresponding in position to digits "1" in the number are put on; those corresponding to digits "0" are left off. They are known as the Input Dynamiciser lamps. There is also on the control panel a second row of 32 lamps, on which single numbers arising from a computation may be displayed in binary form in a similar way. This second row of lamps is called the Output Staticiser.

The principal method of reading numbers in and out of the Machine is by Hollerith Punched cards. Each card has 80 columns and 12 rows; only 32 of the 80 columns are used by the DEUCE; they are columns 21 to 52, numbering from the left-hand end of the card. The rows are numbered, from the top of the card, "Y, X, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9". This is the order in which the rows pass through the machine to be read or punched; the various holes in any particular row are all read or punched simultaneously. The pattern of holes punched on a card may represent either decimal or binary numbers. In the reading of initial data and the punching of final results, decimal notation is used; each column carries only one hole, which is in the row carrying the number of the corresponding digit. A positive or negative number may be indicated by a hole punched respectively in the Y or X row of a column reserved for this purpose. Thus the number +59378 could be represented by a card punched in row Y of the left-hand column and in rows 5, 9, 3, 7, 8 of the next five columns.

Where the computation is too large to be dealt with all at once, it is done by stages, with a separate programme for each stage. The intermediate results are usually punched in binary form. In this case, only one row is required for each number, since the holes may represent "1"s and the blanks "0"s, and twelve numbers may be punched on each card. As will be explained below, binary numbers are always punched with their least significant digit on the left.

2.7. SERIAL REPRESENTATION.

In the examples previously considered, the digits of a number have been assumed to be present simultaneously, and in these circumstances a set of 32 units, such as switches or holes in a card, is needed to represent each number. Within the DEUCE, however, the digits of a number appear one after another at fixed intervals of time. A single unit is used to display all 32 digits of number in succession. The difference is similar to that between a number printed on a page and one spoken aloud. The two systems are known respectively as 'Parallel' and 'Serial' representation.

A number is represented by a sequence of 32 electrical pulses occurring at intervals of one microsecond. (This is not strictly true, since only those pulses corresponding to unit digits are present, the pulses corresponding to zero digits being omitted). A number thus occupies a time of 32 microseconds.

Some confusion is caused by the fact that the first digit to appear is always the least significant. This convention is necessitated by the process of addition, since a particular digit of the sum of two numbers may be affected by a carry from a less significant place. Since, in physics and engineering, the time scale is normally drawn from left to right, this convention has led indirectly to the system of representing binary numbers on the DEUCE with the least significant digit on the left, in contrast with the normal convention. This is true of the Input Dynamiciser and Output Staticiser, and of the representation of binary numbers on Hollerith cards.

There is some difference of opinion, in work and reports on the DEUCE as to whether binary numbers should be written with the least significant digit on the right or the left; to avoid confusion, it must always be remembered whether "forward" or "backward" binary is being used.

Example.

The number 13 ($1 + 4 + 8$) is represented by the following signal:-

1st. Microsecond	a pulse appears
2nd. Microsecond	nothing happens
3rd. & 4th. Microseconds	two more pulses
and 28 more inactive microseconds.	

METHOD OF STORAGE

3. METHOD OF STORAGE

3.1. DELAY LINE STORES.

In order to carry out a computation, both the numbers involved and the instructions specifying the computation are stored in the machine in the form of sequences of 32 electrical pulses. Such a sequence of pulses, which may represent either a number or an instruction, is known in general as a "word".

(Incidentally, the fact that numbers and instructions are stored in exactly the same form permits the use of the arithmetic facilities to modify or create an instruction. This facility is a vital feature of this type of machine).

The method of storage is based on a special delay element which may be regarded as having two terminals. Any pulse applied to the input terminal appears at the Output terminal after a delay of 32 microseconds (or more - see below). To store a number or instruction, the sequence of pulses is applied to the Input of such a delay element; it will appear unchanged at the Output 32 microseconds later. Since the last digit occurs only 31 microseconds after the first, there will be a time when the last pulse has reached the Input, but the first pulse has not yet appeared at the Output. At this moment, a connection is made between the two terminals. Immediately after this, the first digit emerges at the Output terminal, and is at once returned to the Input, to reappear after a further 32 microseconds. It is followed in succession by the other 31 digits. So long as the back connection, or "circulation path" is maintained, the complete sequence of pulses will continue to repeat its appearance at the Output terminal of the delay element once every 32 microseconds.

In order to replace the word being stored by a new word, the circulation path is temporarily broken and the new sequence of pulses applied to the Input terminal. During this time, any pulses appearing at the output are lost, since this terminal is disconnected. After 32 microseconds the whole of the old word has gone, all the digits of the new word have been applied successively to the Input of the delay, and its first digit is due to appear at the output. At this point, the circulation path is re-established, and the required replacement has taken place.

A word in such a storage position is repeated indefinitely; the last digit is always followed immediately by a copy of the first, and there is no indication in the stream of pulses of where the word begins and ends. In order to keep track of this point, the operation of the DEUCE is controlled by a sequence of special pulses occurring at intervals of 32 microseconds. Every operation begins and ends at the time marked by one of these pulses. The period of 32 microseconds between two of these pulses is known as a 'minor cycle' or "m.c.".

There are 22 such storage positions in the DEUCE each with its own delay element. The delay is not the same in all cases, for in the same way

that 32 digits may be stored by introducing a delay of 32 microseconds, 1024 digits are stored in a delay of 1024 microseconds. In this case, however, the 1024 digits are regarded as making up 32 quite independent words of 32 digits each. The 32 words stored in such a position, of course, appear successively at the output terminal, so that a particular word is presented only once in every 32 minor cycles. In order to replace a single word, the circulation path is broken for only 1 m.c., while the new word is applied to the input, and is then restored. The other 31 words in the same storage position are thus not disturbed.

As well as the storage positions holding one and 32 words respectively, there are three positions with delays of 64 microseconds, each holding two 32 digit words and two delays of 128 microseconds, each holding four 32-digit words. The former are useful in double-length arithmetic and for holding the 64-digit product of two 32-digit numbers. The storage system may now be set out in detail.

3.1.1. Temporary Stores.

There are five of these, holding one word each. One, T.S.COUNT, forms part of the Control mechanism, which interprets and obeys instructions, and is not available for the storage of numbers. The other are TS13, TS14, TS15 and TS16.

3.1.2. Double-length Stores.

There are three of these, each holding two words. They are DS19, DS20 and DS21.

3.1.3. Quadruple Stores.

There are two of these, each holding four words. They are QS17 and QS18.

3.1.4. Delay Lines.

There are twelve of these, each holding 32 words. They are numbered DL1 to DL12 inclusive. Numbers are generally stored in DLs and moved into the smaller stores when they are required for arithmetic processes. The object in designing the layout of the store was to combine adequately rapid access to numbers in immediate use with economy of storage equipment.

3.2. NUMBERING OF MINOR CYCLES.

So long as the DEUCE is switched on, the controlling pulses continue to be generated at intervals of 32 microseconds, dividing time into sections of one "minor cycle" each. For convenience, the minor cycles are referred to by numbers, from 0 to 31 and then starting again at 0. Suppose we send a word into DL1 during m.c. 5, by breaking the circulation path for this minor cycle only while the sequence of digits is applied to the Input terminal. Each digit will appear at the Output of DL1 1024 microseconds after it is sent to the Input. In other words, the whole word will

emerge, just as it went in, during the 32nd minor cycle after that in which it was inserted. Unless the circulation path is broken during this minor cycle, the same word will re-emerge after a further 32 minor cycles, and again in every 32nd minor cycle until it is replaced. Every one of these minor cycles will bear the number "5". We say that the word has been stored in "DL1, m.c. 5", or simply in "15"

The notation for a QS or DS is slightly less obvious. Suppose, for example, that a word has been sent to QS17 in m.c. 25; it will reappear in m.c. 29, again in m.c. 1 and in m.c. 5, m.c. 9, etc. The notation uses the lowest number from this list; the word is said to be stored in 17₁. The other three words in the QS are in 17₀, 17₂ and 17₃.

The two words in a DS are said to be stored in m.c.2 and 3; the numbers 0 and 1 are rejected to avoid confusion with a previous, now obsolescent, notation. A word in a TS is available in every minor cycle and is simply said to be stored, for instance, in 'TS15' or in '15'

This numbering of minor cycles has no particular significance in the operation of the Machine. Before a programme is run in, the entire storage (except magnetics - see below) is cleared by pressing a key. The instructions of the programme are then read from Hollerith cards and placed in the storage. The minor cycle in which the first word is sent to a Delay Line is given the number "0", and the others are numbered successively. Between programmes, while the store is empty, the minor cycles may be considered anonymous.

During any period when the Machine is inactive (i.e. when the circulation paths of all delay elements are closed and no transfers or arithmetic operations are in progress) the state of all parts of the machine is repeated exactly at intervals of 32 minor cycles, the delay period of the Delay Lines. This period of 32 minor cycles (1024 us) is referred to as a 'Major Cycle'.

3.3 MAGNETIC STORE.

In addition to the high-speed Delay Line Store described above, the DEUCE has an auxiliary Magnetic Drum Store. This holds 8192 words altogether, but has the disadvantage that information takes longer to get in and out.

The drum is a continuously rotating cylinder with a surface of magnetic material. Digits are represented by small magnetic dipoles induced in this surface by a writing head held close to the drum. As the drum revolves, this head writes a circumferential track of information. There are 256 tracks each holding 32 words, the contents of one DL. The information is read from the Drum by a reading head held close to it opposite the same track.

There are actually sixteen writing heads held solidly together in a block which is capable of movement into anyone of sixteen positions along the Drum. Information is transferred to and from the Magnetics only in blocks of 32 words, that is one track. To write such a block on to a particular track, two operations are in general necessary. The first is to shift the write head block into the appropriate position (this is necessary only if it is not already known to be there); the second is to write on the Drum through the appropriate head. The tracks may be considered as numbered from 0 to 255, those numbered 0 to 15 being reached with the head block in position 0, those from 16 to 31 with it in position 1, and so on. If the track number is written as an eight-digit binary number the most significant four digits represent the head block position, numbered from 0 to 15, and the bottom four digits represent the particular head to be used. In the normal notation a track number is written in the form "a/b", where "a" represents the head-block position and "b" the head number.

There is a similar arrangement of sixteen reading heads in a block which is moveable into any of sixteen positions. The two mechanisms are independent, so that an operation which requires repeated reading from tracks 48, 49 and 50 (reading head block position 3) and repeated writing on tracks 80, 81 and 82 (writing head block position 5) does not need any shifting operations.

Information is always written from and read into DL11. There are thus four basic operations concerned with the Magnetics, shift read heads into a specified position, shift write heads, write the contents of DL11 on to the track now under a specified writing head and read the track now under a specified reading head into DL11.

PROGRAMMING FOR DEUCE

4. PROGRAMMING FOR THE DEUCE.

4.1. REPLACEMENT TRANSFER.

The DEUCE works by carrying out a sequence of instructions in a prescribed order. In the simplest case, an instruction requires merely a replacement transfer from one to another of the twenty-one storage positions on the Machine.

Example.

The instruction "13 to 15" replaces the content of TS15 with that of TS13, leaving TS13 unchanged. There are thus now two copies of the former content of TS13. An instruction is normally written in the form "13 - 15" (the sign being a dash and not a minus).

4.2. FORM OF INSTRUCTION.

It will be seen that an instruction must specify both a Source number and a Destination number. Each of these may take any value from 0 to 31. Of these, 21 Source numbers and 21 Destination numbers relate to the storage location listed above, the number of the location (with the appropriate prefix TS, DS, QS or DL) being the same as its Source and Destination numbers. The remaining 11 Sources and 11 Destinations are used for various purposes including the arithmetic operations and the operation and control of the Hollerith Read and Punch machines and the Magnetic Store. There is in general no particular relationship between the Source and the Destination bearing the same number, apart from those corresponding to the various storage positions.

4.3. ADDITION AND SUBTRACTION.

Addition and subtraction may be performed in only two of the storage positions, TS13 and DS21. Each of these four facilities has its own special destination. Destinations 25 and 26 are respectively the additive and subtractive inputs to TS13; destinations 22 and 23 are respectively the additive and subtractive inputs to DS21.

Example.

The instruction "14 - 26" subtracts the content of TS14 from that of TS13, leaving TS14 unchanged and the difference in TS13.

The sequence of instructions:-

15 - 13
16 - 25
13 - 15

replaces the content of TS15 with the sum of those of TS15 and TS16 leaving TS16 unchanged and a copy of the sum in TS13. These three instructions form an elementary "Programme".

4.4. FUNCTIONAL SOURCES.

Destinations 22, 23, 25 and 26 may be called "functional destinations". There are also certain functional sources, associated with TS14, TS15 and DS21. For instance, Sources 23 and 24 give the contents of TS14 respectively divided by two and multiplied by two (in the signed convention).

Example.

"24 - 25" adds twice the content of TS14 into TS13, leaving TS14 unchanged and the sum in TS13. "23 - 14" halves the content of TS14, leaving all other storage positions unchanged.

4.5 FIXED NUMBERS.

Also available at Sources are certain useful fixed numbers which are built into the machine. Thus the numbers 0, 1 and -1 (in the least significant place) are obtained from Sources 30, 27 and 31 respectively. Source 31 in fact gives a continuous sequence of "ones", of which the 32 occurring in any given minor cycle represent -1 in the signed convention.

Example.

"30 - 16" clears TS16. Either "27 - 26" or "31 - 25" subtracts 1 from the content of TS13.

4.6. OTHER FACILITIES.

With a few examples already given of this form of instruction, specifying a Source and a Destination, it is possible to carry out several different types of operation including replacement transfer, addition, subtraction and multiplication or division by 2. In each case, the particular type of operation, as well as the storage positions involved, is specified by the chosen Source and Destination numbers. One of the Sources and seven of the Destinations do not fall into any of the classes which have yet been mentioned. These are used to give further types of operation with the same form of instruction.

4.7. DESTINATION TRIGGERS.

In some cases, an Instruction does not involve the transfer of a sequence of digits, as all the previous examples do, but merely requires that something shall happen or start happening. This 'something' may be the passage of cards through the Hollerith Reader or Punch, or a process of multiplication or division. Most of these operations (twelve, in fact) are grouped under one Destination, D24, which is called "Destination Triggers". Which of the twelve more or less independent operations is intended is specified by the Source Numbers. Only one example will be given here, the others being mentioned in appropriate subsequent paragraphs including the one immediately following.

Example.

"0 - 24" initiates the multiplication operation associated with DS21 and TS16. This operation replaces the contents of DS21 with the product of the two 32-digit numbers previously in TS16 and in DS21₃, leaving the number in TS16 unchanged (remember that the two words in DS21 are said to be stored in 21₂ and 21₃). More will be said about multiplication later, and also about division.

4.8 INPUT AND OUTPUT

The Instruction "10 - 24" causes a sequence of cards to start passing through the Hollerith Punch; this by itself gives no output of information. The 21 storage positions may be completely full of numbers, but none of them will be punched on the card unless the number to be punched is sent to the output, which is Destination 29. The sequence of cards is stopped by the Instruction "9 - 24".

Example.

The three-instruction programme "10 - 24
15 - 29
9 - 24" causes one card to pass through the Punch and the contents of TS15 to be punched upon it.

When the Punch is idle, Destination 29 cannot send a number to a card. Instead of wasting the Destination, its function is in these circumstances transferred to the upper row of lamps on the control panel, known as the Output Staticiser: e.g. "16 - 29" displays the content of TS16 on the OS lamps (provided the Punch is not running).

A similar mechanism applies to the input of information. The Hollerith Read machine is started by the instruction "12 - 24", and subsequently Source 0 supplies the information being read from the current row. When the Read machine is not running Source 0 supplies the number which has been set up manually at the Control Panel by means of the I.D. switches. The sequences of cards passing through the Reader is stopped by the instruction "9 - 24". The same stopping instruction does for both Read and Punch since they are never in practice used simultaneously.

The method of input and output is complicated by the difference in speed between the DEUCE and the Hollerith Machines; this problem will be dealt with later.

4.9. CONSTRUCTION OF A SIMPLE PROGRAMME.

Example.

Although several facilities remain to be described, it is already possible to make a programme for a simple problem. The following programme forms the squares of successive natural numbers, starting at 1, from the property that the second difference is constant. This second difference is stored in TS14, the first difference in TS15 and the square itself in TS16. The only arithmetic facility used is the additive input to TS13 (Destination 25). Once the initial values are in place, the arithmetic is done by the following cycle of instructions; the content of each TS is given for two typical cycles.

Instructions.

	<u>First Time</u>				<u>Second Time</u>			
	<u>TS</u>	<u>14</u>	<u>15</u>	<u>16</u>	<u>13</u>	<u>14</u>	<u>15</u>	<u>16</u>
14 - 13	2	3	4	2	2	5	9	2
15 - 25	2	3	4	5	2	5	9	7
13 - 15	2	5	4	5	2	7	9	7
16 - 25	2	5	4	9	2	7	9	16
13 - 16	2	5	9	9	2	7	16	16

Extra instructions must be inserted in the loop to punch the answers obtained, and at the beginning to set up the appropriate initial values. The complete programme might be as given below, though one or two instructions could be saved by taking more advantage of the fact that all storage positions are cleared at the start of a programme and by feeding in the constant second difference with the instructions rather than building it up in the programme.

Instructions.

	<u>TS</u>	<u>14</u>	<u>15</u>	<u>16</u>	<u>13</u>
27 - 13		0	0	0	1
27 - 25		0	0	0	2
13 - 14		2	0	0	2
27 - 15		2	1	0	2
27 - 16		2	1	1	2
10 - 24 (start punch)					
16 - 29 (Punch content of TS 16)					
14 - 13		2	2n-1	n ²	2
15 - 25		2	2n-1	n ²	2n+1
13 - 15		2	2n+1	n ²	2n+1
16 - 25		2	2n+1	n ²	(n+1) ²
13 - 16		2	2n+1	(n+1) ²	(n+1) ²

An important feature of this programme is the repeated loop. By means of twelve instructions, all the squares less than $2^{32} - 1$ (about 60,000 in number) are punched out. As the programme stands, in fact, it will not stop at this point; the 32 least significant digits will be punched of higher and higher squares, of which the upper digits have been lost, and this was hardly the result intended. Before dealing with the facilities which enable a programme to stop at a prescribed point, a little more must be said about the programme in its present form.

4.10 THE NEXT INSTRUCTION.

The operation of a programme is determined as much by the order in which instructions are obeyed as by the individual instructions. These must therefore be linked together in some way, to ensure the required sequence. For this reason, each instruction is made to specify, as well as its Source and Destination numbers, the storage location of the instruction which is to follow it.

Instructions are initially read from Hollerith cards and stored in locations decided by the programmer. The "Next Instruction" section of an instruction cannot be filled in until its successor has been assigned to a storage location.

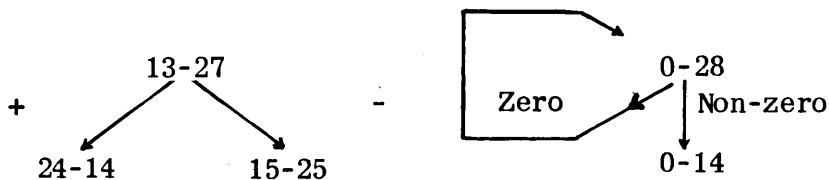
Example.

Suppose the first two instructions in the previous example are to be stored in DL1, in minor cycles 3 and 5 respectively. The first instruction would then read '27 - 13', Next instruction in DL1, 2 m.c. on from this one. (Minor cycles are counted from the one in which the current instruction is stored). If the first instruction were placed instead in DL3, m.c. 3, it would remain unchanged, provided that the second instruction were still in 15.

4.11. DISCRIMINATION FACILITIES.

In most cases, the location of the Next Instruction is specified uniquely. An instruction which sends a number to Destination 27 or 28 however, may take its successor from either of two locations, depending on the number being sent. With D27, the choice is between positive and negative numbers; that is, between numbers whose 32nd digit is respectively 0 and 1. (In this convention, zero is treated as a positive number). With D28, it is between zero and non-zero numbers.

Example.



In the first case, the number in TS13 is examined to see whether it is positive or negative.

If it is positive, the next instruction is '24 - 14' which shifts up the number in TS14 (i.e. multiplies it by 2); if negative, the next instruction is '15 - 25', which adds the content of TS15 to that of TS13. These few instructions form part of one method of programming a square root calculation. In the second example, the number on the Input Dynamiciser is examined to see whether or not it is zero; so long as there is no digit on the I.D. this instruction is repeated indefinitely; as soon as any digit is inserted by depressing the appropriate switch,

the machine proceeds to the next instruction, transferring this digit into TS14. These are the first two instructions in a programme for reading decimal numbers from the I.D., one digit at a time.

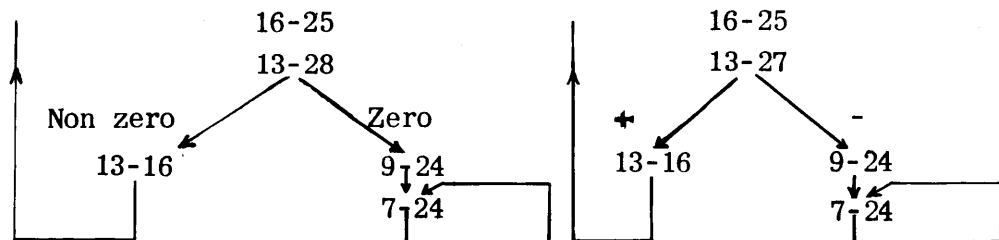
4.12 THE ALARUM.

In completing the Successive Squares programme, an additional facility will be used. This is the Alarum which sounds a buzzer and lights a special red lamp on the Control Panel. The Alarum is started by the instruction "7 - 24" and may be stopped either by the instruction "6 - 24" or by pressing a special key on the Control Panel.

The Alarum is most often used to indicate the failure of some check which has been incorporated in the programme. In the present case, however, it will indicate the completion of the required number of results.

Example.

Two simple ways of completing the Successive Squares programme will be given, in each case inserting a "discrimination" between the last two instructions of the loop.



The right-hand path in each case indicates that all the required squares have been punched; "9 - 24" stops the stream of cards through the Punch and "7 - 24" sounds the alarum.

For both discriminations, TS13 contains the number $(n+1)^2$, the result due to be punched during the next repetition of the loop. This is never zero for any result within the digit capacity; the discrimination in the first example will then take the left-hand path, completing the loop and punching all the valid results. The first excessive result is 2^{32} , the square of 2^{16} ; the first 32 digits of this are zero and the 33rd, a "1", is lost, leaving the number zero in TS13. At this point, the right-hand alternative is chosen, and the buzzer sounds. In the second case, the loop is repeated until the number in TS13 has a "1" as its 32nd digit; i.e. until the value to be punched exceeds 2^{31} . The words "positive" and "negative" are misnomers in this case, but they are normally used in this situation, even when working, as here, in the Unsigned convention. The second method produces fewer results, but is less specialised, since it works despite the fact that the number 2^{31} is not a perfect square. The "next instruction" part of an Instruction cannot be left blank; there must always be a successor. Since, in this case, no useful action is required after "7 - 24", this Instruction is made to

specify its own storage position as the location of the next Instruction. It is therefore repeated indefinitely until stopped by external action.

4.13 LONGER TRANSFERS.

An instruction specifies the transfer of information from a Source to a Destination. So far, this information has in each case consisted of one word; the connection between, say, the Output of one DELAY element and the Input of another has been maintained for just one minor cycle, long enough for 32 digits to pass between the two storage positions. If required, this connection can, by one instruction, be maintained for any number of minor cycles up to 32.

Examples.

"15 - 25 (3 m.c.)"; since the same number appears at the Output of a TS in every minor cycle, this instruction simply adds the content of TS15 into TS13 three times. In other words, it adds three times the number in TS15 to the number in TS13.

"13 - 25 (5 m.c.)"; in the first m.c., the number in TS13 is added to itself, and thus doubled; in the second m.c., it is doubled again, and so on. The net effect of the transfer is to multiply the number in TS13 by 32.

"1 - 25 (32 m.c.)"; 32 different numbers appear at the Output of a DL before the first one is repeated. This instruction therefore adds together into TS13 all the 32 numbers stored in DL1.

"16 - 15 (2 m.c.)"; after one m.c., the contents of TS15 and TS16 are identical; the second minor cycle, in which the content of TS15 is again replaced with that of TS16, is therefore redundant.

4.14. TIME OF OPERATION

It is sometimes important to know the time occupied by a sequence of Instructions, particularly when using the Hollerith Punch or Read. In the above examples, the transfers occupy respectively 3 m.c., 5 m.c., 32 m.c., and 2 m.c.

To calculate the total time occupied by an Instruction, one must add to the period of transfer a minor cycle which is occupied in setting up the Source and Destination connections, and also the duration of any pauses which the Instruction may specify both before and after the period of transfer. The question of timing will be reconsidered when the Instruction word is studied in detail.

4.15 TRANSFER FOR A PARTICULAR MINOR CYCLE.

In a large computation most of the numbers involved are stored in Delay Lines, and only transferred into a Temporary Store when they are required for an arithmetic operation. In order to select the desired one of the 32 different numbers stored in a DL, the instruction must specify not only the appropriate Source but also the particular minor cycle in which transfer is to take place. In fact, every instruction does specify the minor cycle or cycles of transfer but for transfers between Temporary Stores it is immaterial, and is therefore not mentioned.

Examples.

"1₅ - 16" transfers the number in DL1, m.c.5, into TS16. The Output of DL1 is connected to the Input of TS16 for m.c. 5 only.

"3₁₁ - 4₁₁"; it is not possible to transfer directly from one m.c. of one DL to a different m.c. of another DL. This may, however, be done in two instructions, using a vacant T.S. as an intermediate store. Similarly, a word cannot be transferred directly from any even m.c. of a DL to the odd half of a double-length Store, or vice versa. Transfers to or from a QS are subject to similar restrictions.

"21₃ - 27" discriminates on the sign of the number in the odd half of DS21.

4.16. COUNTING CYCLES OF REPETITIVE LOOPS.

In a previous example, the termination of a programme for punching successive squares was indicated by sounding the buzzer when the capacity of a storage location was about to be exceeded. A method of more general application is to count directly the number of repetitions of a closed loop of instructions, an alternative path being followed after the cycle has been performed the required number of times.

Example.

A second version will now be given of the Successive Squares programme; this time, just twenty results will be punched before sounding the alarm. To achieve this, the number '20' will be placed in the even half of DS21 before entering the repeated loop, and '1' subtracted from it once in each repetition.

Instructions.

Contents of Storage.

		TS15	TS16	TS13	DS21 ₂
'20'	- 21 ₂	0	0	0	20
27	- 15	1	0	0	20
		<u>First Time</u>		<u>General</u> (only given at changes)	
15	- 25	1 0 1 20	2n-3	(n-1) ²	2n-1 21-n
13	- 15	1 0 1 20	2n-1		
16	- 25	1 0 1 20		n ²	
13	- 16	1 1 1 20		n ²	
10	- 24				
16	- 29				
27	- 23 ₂	1 1 1 19			20-n
21 ₂	- 28				
Non Zero "2"	13	1 1 2 19			2
Zero	9 - 24				
	7 - 24				

The programme above contains one or two other ideas not previously mentioned. The numbers '20' and '2' are punched on the programme cards and fed into the storage along with the instructions; the Source number in '20' - 21 will be that of the storage position where the number '20' has been 'planted'. This will normally be in a DL and in this case the number must be planted in an even minor cycle in order that a transfer may be made to DS21₂. This method contrasts with the previous version of the programme, where the number "2" was built up in TS13. There is often more than one way of producing a required result; the choice of method is made so as to reduce the amount of storage required and the time (in minor cycles) of operation. These two requirements may conflict in a particular example; in the present case, however, the 'planted number' technique wins on both counts. An additional advantage of planting the number "2" is that TS14 need not now be used; if this programme were part of a larger one, TS14 could be storing some parameter of the main computation.

The repeated loop is entered at a different point in the two versions. In the second version, some of the instructions in the loop are used in the first repetition to set up the initial values of the variables; an instruction is thus saved in the part of the programme before entering the loop.

4.17. STOP INSTRUCTIONS.

There is a little more to be said about the output of results on to cards or the Output Staticiser lamps. In the programme above, for instance, successive results will be calculated and sent to the output by the Instruction "16 - 29" and cards will pass through the Punch presenting their rows successively beneath the punching knives; so far, however, there is no provision for the synchronisation of the two trains of events. As it stands, in fact, the programme will calculate all 20 results and try to punch them before the first row of the first card has reached the punching knives.

This synchronisation depends on the fact that any Instruction may be made a Stop Instruction or "Stopper". When a Stopper is reached in the course of a programme, the Machine stops and there is no further action until an external stimulus is supplied called a "Single-Shot" which causes the Stopper to be obeyed and to call in its successor. The programme then proceeds until it comes to another Stop Instruction, when it waits for a further Single-Shot.

A Stopper is indicated in the programme by an X after the Instruction.

4.18. SINGLE-SHOTS.

A Single-Shot may be supplied by pressing a special key on the Control Panel. When either the Punch or Reader is running, however, this key is no longer operative. Instead, the Reader or Punch (whichever is running - they are never used simultaneously) supplies a succession of Single-Shots, one for each row of a card, occurring just as the row comes into position to be read or punched.

Example.

To synchronise the Successive Squares programme with the Punch, we need only make the Instruction "16 - 29", which sends the successive squares to the output, a Stopper ("16 - 29X"). In each repetition of the loop, this Instruction can now only be obeyed as a row of a card comes into position under the punching knives and the Punch gives a Single-Shot. The Programme then hurries round the loop until it reaches this Instruction again; by this time the next result has been calculated and placed in TS16. The DEUCE then waits in suspended animation until the arrival of the next row at the punching knives enables this next result to be punched in its turn.

The same technique is used in reading from cards. In some cases it is also necessary when reading from the ID. keys or displaying on the OS lamps on the Control Panel; in these cases, of course, the Single-Shots come from the Control Panel key, not the Reader or Punch.

4.19. STOP KEY.

It may be worth mentioning here that there is a key on the Control Panel which in effect makes all Instructions behave like Stoppers. It is then possible to trek through the whole programme, one Instruction at a time, by the Single-Shot key on the Control Panel, having a good look at each Instruction and at its effect. This can be very useful in finding mistakes in programmes. (Every programmer makes just one programme which is correct without the need for such testing - usually his first).

THE INSTRUCTION WORD

5. THE INSTRUCTION WORD.

A general view of the operation of the DEUCE has now been given, but several points remain to be explained before it can be fully understood. The most important of these is the precise form in which an Instruction is represented by a 32-digit word.

5.1. CONTROL.

The section of the DEUCE which interprets instructions is called "Control". It incorporates a special storage position, TS COUNT, in which is stored the instruction being obeyed. "Obeying an Instruction" implies two operations, first carrying out the required transfer, and then taking the next instruction into TS COUNT. The first of these may occupy up to 64 minor cycles, the second takes just one.

5.2. INSTRUCTION REQUIREMENTS.

The details of an instruction are contained in a word of 32 digits. This word must specify the Source, Destination and minor cycle or cycles of transfer, whether the instruction is a Stopper, and the storage position and minor cycle in which the next instruction is stored. The choice of the storage position from which the next instruction is to be taken is limited to the first eight DLs. There is thus rapid access to a maximum of 256 Instructions. Larger programmes can of course be accommodated by keeping further instructions in the magnetic store and bringing them into one of the first eight DLs as they are required.

5.3. THE INSTRUCTION WORD.

The instruction word comprises seven separate numbers of up to five digits each, with a few digits spare. Each of these numbers occupies a particular place in the word, and has its own function and range of values. These are set out briefly below, but the precise operation of some of the numbers will be explained in later paragraphs.

<u>Digits.</u>	<u>Name and Range.</u>	<u>Function.</u>
2 to 4	Next Instruction Source (NIS) 0 to 7	Number of DL (1 to 7) in which instruction is stored; NIS "0" means DL8.
5 to 9	Source (S) 0 to 31	Number of selected Source
10 to 14	Destination (D) 0 to 31	Number of selected Destination
15,16	Characteristic (C) 0, 1 or 2.	Determines length of Transfer (explained below).
17 to 21	Wait Number (W) 0 to 31	Specifies first m.c. of Transfer (explained below).
26 to 30	Timing Number (T) 0 to 31	Specifies m.c. of next instruction, and sometimes also last m.c. of Transfer (explained below).
32	Go Digit (G) 0 or 1	If G is 0, the instruction is a Stopper, and waits for a One-Shot; if G is 1, the instruction is obeyed immediately.

Digits 1, 22 to 25 and 31 are not used in the instruction word; their value is ignored by Control and has no effect on the operation.

5.4. ADDRESS SECTION.

The numbers NIS (sometimes called simply "N"), S and D are said to form the "Address Section" of the instruction. As soon as a new instruction begins to emerge from TS COUNT, Control interprets these numbers, digit by digit, to select the required Source, Destination and NIS. Connection between the selected Source and Destination, and from the selected NIS to TS COUNT, will not, however, be made until the minor cycles determined by the other numbers in the instruction.

5.5. SET-UP MINOR CYCLE.

The process of reading the Address Section and setting up the appropriate connections occupies the first half of a minor cycle, which is therefore called the "Set-Up minor cycle". Nothing else must happen in this minor cycle, since any words transferred would come partly from each of two sources (those of the old and new instructions) and would therefore be meaningless. The Set-up m.c. immediately follows that in which the new instruction enters TS COUNT; if this new instruction comes from a Delay Line in the normal way it can, of course, enter TS COUNT only in the minor cycle in which it has been stored.

Example.

Before explaining the rest of the instruction word, a demonstration will be given of the simplest type of operation, taking as an example the first few instructions of the Successive Squares programme (first version) on Page 16.

These instructions were:	A	27 - 13
	B	27 - 25
	C	13 - 14
	D	27 - 15
		etc.

We will suppose that Instruction 'A' enters TS COUNT in m.c. 0. During this minor cycle, the transfer ordered by a previous instruction may still be in progress. The precise timing is then, in the simplest case, as follows:-

Minor Cycle	TS COUNT	Action
0	takes in instruction "A"	(? transfer by previous instruction)
1	holds "A"	"A" set up
2	takes in "B"	transfer 27 - 13 ("A")
3	holds "B"	"B" set up
4	takes in "C"	transfer 27 - 25 ("B")
etc.		

During each even m.c. in the present example, two "transfers" take place, the main transfer specified by the instruction and the "transfer" of the next instruction into "TS COUNT". Instructions are normally stored in a Delay Line, and must be stored in the proper minor cycles to be available to enter TS COUNT when required. In this case, successive instructions would be stored in successive even m.c. of a DL. In a programme of this size, all the instructions could be kept in one DL: we might use DL1 for the purpose, in which case they would be stored in successive even minor cycles of DL1 and would all have NIS number "1". In the example, all odd m.c. are set-up m.c. during which there is no action except the setting up of the N, S and D connections.

5.6. TIMING SECTION.

The numbers C, W and T are said to form the timing section of the Instruction. The Characteristic C divides Instructions into three types, those requiring a Single Transfer ($C = 0$) a Long Transfer ($C = 1$) and a Double Transfer ($C = 2$). The Single Transfer will be described first.

5.7. SINGLE TRANSFER ($C = 0$)

If $C = 0$, transfer takes place for just one minor cycle. If the Instruction enters TS COUNT in m.c.m, transfer is in m.c. $m + W + 2$ and the next Instruction enters TS COUNT in m.c. $m + T + 2$. In the above example,

all the Instructions could have had $C = W = T = 0$; in each case, transfer would occur in m.c. $m + 2$ and the next Instruction would enter TS COUNT in the same minor cycle. In other words, each successive Instruction would take effect, and also would call in its successor, two minor cycles after it entered TS COUNT. This corresponds with the scheme given above.

Example.

An Instruction stored in 1_{17} is required to transfer TS13 to 3_{25} and take its successor from 5_{29} . The details of the Instruction would be

N	S	D	C	W	T	G
5	13	3	0	6	10	1

The detailed operation of this Instruction would be

	m.c.	
	m (17)	Instruction enters TS COUNT
	m + 1 (18)	N, S and D connections set up
	m + 2 (19)	} Pause of W(6) m.c.
	to	
	m + W + 1 (24)	
	m + W + 2 (25)	Transfer from TS13 to DL3
	m + W + 3 (26)	} Pause of T - W - 1 (3) m.c.
	to	
	m + T + 1 (28)	
	m + T + 2 (29)	Next Instruction enters TS COUNT from DL5
	m + T + 3 (30)	N, S and D connections of next Instruction set up.

The Instruction would normally be written "5, 13-3, 6, 10". An instruction is assumed to be single-transfer ($C=0$) unless otherwise specified. This is done by inserting a symbol between the "D" and "W" numbers; the symbol is "l" or "1" for a long transfer ($C=1$), "2" or "d" for a double transfer ($C=2$) and "3" or "q" for $C=3$ if this is ever used. Single transfer is indicated by "0" or "s", but this, as mentioned above, is usually omitted in the coding. The numbers are preferred to the letters. A Stopper ($G=0$) is indicated by an "X" at the end.

5.7.1. W Greater than T.

An Instruction clearly must not call in its successor before it has been obeyed itself. If W exceeds T, the value of T is effectively increased by 32 and the next Instruction enters TS COUNT not in m.c. $m + T + 2$ but in m.c. $m + T + 34$.

Example.

The Instruction "0,4 - 16,10,1" stored in 2_{27} would transfer from DL4 to TS16 in m.c. 39, i.e. m.c. 7 of the next Major Cycle and would call its successor from DL8 in m.c. 62, i.e. m.c. 30 of the next Major Cycle. During the operation of the Instruction, the next Instruction emerges twice from the delay element of DL8, but is refused admission to TS COUNT on its first attempt.

5.8. DOUBLE TRANSFER.

If $C = 2$, transfer is for two successive minor cycles, $m + W + 2$ and $m + W + 3$. Everything else is as for the case of Single Transfer. There is one difficulty; if $W = T$, $m + W + 3 = m + T + 3$, the Set-up minor cycle of the next Instruction. To overcome this, it is arranged that where $C = 2$ the value of T shall be increased in effect by 32 when $W = T$ as well as when $W > T$. This gives plenty of room for the two-minor-cycle transfer.

Example.

"4,20 - 17,2,1,5" stored in 1_{13} transfers from DS20 to QS17 in m.c. 16 and 17 and takes the next Instruction from 4_{20} . This replaces 17_0 and 17_1 with a copy of the contents of DS20.

"3,21 - 11,2,0,0" stored in 3_4 transfers from DS21 to DL11 in m.c. 6 and 7 and takes the next Instruction from 3_6 in the next Major Cycle.

5.9. LONG TRANSFER.

If $C = 1$, transfer is for a sequence of $T - W + 1$ minor cycles from m.c. $m + W + 2$ to $m + T + 2$ inclusive. The next Instruction still enters in m.c. $m + T + 2$, in the last minor cycle of transfer. If W exceeds T , transfer is for $T - W + 33$ minor cycles, from $m + W + 2$ to $m + T + 34$ and the next Instruction enters in m.c. $m + T + 34$.

Example.

"7, 15 - 25,1,29,1" stored in 6_5 adds TS15 into TS13 for five minor cycles from m.c. 36 to 40 inclusive i.e. from m.c. 4 to 8 of the next Major Cycle. It takes the next Instruction from 7_8 at the second time of asking. The effect is to add five times the contents of TS15 into TS13; the particular minor cycles of operation didn't matter, and this is often true of a long Transfer.

5.10. C = 3.

It is not intended that the Characteristic should ever be given the value 3. This might happen by accident, however, and the result had better be stated to avoid possible confusion.

If W and T differ, $C = 3$ has just the same effect as $C = 1$; that is, a Long Transfer for $T - W + 1$ minor cycles from m.c. $m + W + 2$ to m.c. $m + T + 2$ inclusive, taking the next Instruction from m.c. $m + T + 2$. (If W exceeds T , read $m + T + 34$ for $m + T + 2$ and $T - W + 33$ for $T - W + 1$).

When $W = T$, $C = 3$ increases the effective value of T by 32. Transfer is for 33 minor cycles from $m + W + 2$ to $m + W + 34$ inclusive (remember $W = T$) and the next Instruction enters during m.c. $m + W + 34$.

5.11 TIME OF COMPUTATION.

The total time taken by a piece of computation must sometimes be known quite accurately. When a card is running through the Hollerith punch, for instance, the rows succeed each other at an interval of about 38 Major Cycles. Where computation is done between rows, it must be kept within this limit. An instruction, in effect, occupies the time between entering TS COUNT itself and calling in its successor. This period, in minor cycles, is $T + 2$ if T exceeds W . If $T = W$ and $C = 0$ or 1 it is also $T + 2$; in all other cases it is $T + 34$.

Examples.


The first few instructions of the programme on page 21 will be coded. The positions of the instructions and planted numbers are arranged to allow the greatest economy of time while using only one of the Delay Lines.

<u>Flow Diagram</u>		<u>Corresponding Coding</u>		
Position DL(m.c.)	Instruction	Position DL(m.c.)	Instruction (or number)	Position of next Instruction
1 ₀	1 ₂ - 21 ₂	1 ₀	1, 1 - 21, 0, 1	1 ₃
1 ₃	27 - 15	1 ₂	"20"	
1 ₅	15 - 25	1 ₃	1, 27 - 15, 0, 0	1 ₅
		1 ₅	1, 15 - 25, 0, 0	1 ₇

An instruction is referred to by the position in the storage which it occupies. In the Flow Diagram, only the Source, Destination and if necessary the m.c. of transfer are mentioned. Instruction 1₀ is single-transfer, so that transfer from S1 to D21 takes place only in m.c. $0 + W + 2$ i.e. m.c. 2; the content of DL1, m.c. 2, is thus transferred to the even m.c. of DS21; the next instruction is taken from DL1 in m.c. $0 + T + 2$, i.e. m.c. 3.

The next example shows what may happen towards the end of coding a large programme, when only a few scattered spaces remain in the

storage and the coding must be arranged accordingly. It is required to punch the contents of m.c. 0 and 1 of DL10 on the first two rows of a card and then to sound the Buzzer to show that computation is complete.

<u>Flow Diagram</u>		<u>Corresponding Coding</u>		
Position DL(m.c.)	Instruction	Position DL(m.c.)	Instruction	Position of next Instruction
2 ₂₇	10 - 24 (Stim PUNCH)	2 ₁₅	2, 7 - 24, 0, 30	2 ₁₅
4 ₅	10 ₀ - 29 X	2 ₂₇	4, 10 - 24, 0, 8	4 ₅
3 ₈	10 ₁ - 29 X	3 ₈	5, 10 - 29, 23, 26 X	5 ₄
5 ₄	9 - 24 (Clear PUNCH)	4 ₅	3, 10 - 29, 25, 1 X	3 ₈
	↓	5 ₄	2, 9 - 24, 0, 9	2 ₁₅
2 ₁₅	 7 - 24 (Stim ALARUM)			

5.12. MODIFICATION OF INSTRUCTIONS.

It will be observed that instructions do not necessarily occur in the same order in the coding as in the flow diagram, and that two instructions in the last example lead to 2₁₅ as next instruction, both 5₄ and 2₁₅ itself. For the precise problem stated, this is the best method of coding. If many more m.c. of DL10 had to be punched, or if some arithmetic operation had to be performed on each number before punching, this would no longer be true. In this case, there would be a sequence of instructions which is repeated exactly except that in one of the instructions the m.c. of transfer must move on one for each repetition. This involves a change only in the Wait number of that instruction.

The method used is to store the instructions once only and arrange the Timing numbers to form a closed loop; in this loop, extra instructions are included which use the arithmetic facilities to modify the instruction in question. In the present case, the modification required is simply an increase of 1 in the Wait number. Since this is often needed, the digit which represents a unit in the Wait number position, the 17th digit of the word, is made available at Source 28 as one of the 'useful constant numbers' already referred to.

The instruction to be modified is transferred to TS13 and modified during each repetition of the loop by the instruction "28 - 25". There remains the problem of getting it back into the programme, since TS13 is not available as a Next Instruction Source. This could be achieved by transferring from TS13 to a vacant space in a DL which would later be named as the next instruction position. This method, however, is wasteful of space, and an alternative is provided in the facility associated with Destination 0.

5.13. DESTINATION 0.

Destination 0 provides an entry to TS COUNT which overrides the function of the NIS selection.

Example.

"15 - 0,1" takes as its next Instruction the word in TS15. The NIS part of the Instruction is ignored.

A transfer to DO must in general be Long (C = 1). The reason for this is that the words from the selected NIS are replaced with those from the selected Source only so long as transfer is taking place.

Example.

"5, 15 - 0,1, 4" stored in m.c. 7 has no effect. Transfer occurs in m.c. 10 which means that for this minor cycle the word from the NIS (5₁₀) is replaced with that from the Source (TS15). This has no effect, however, since the word from the selected NIS would not be entering TS COUNT anyway in this minor cycle. From m.c. 11 to m.c. 13, the words from DL5 are again being presented at the entrance to TS COUNT. In m.c. 13, the word from DL5 is let in to TS COUNT and then forms the next Instruction.

"X, 15 - 0, 1, 0, 4" in m.c. 7 takes its next Instruction from TS15 in m.c. 13 and the NIS number is irrelevant. Transfer, which replaces the words from DLX with copies of the word in TS15, continues from m.c. 9 to m.c. 13 inclusive; in the last of these, m.c. 13, the word presented to TS COUNT is allowed in.

"X, 15 - 0, 3, 3" in m.c. 7 does take its successor from TS15 in m.c. 12. Since W = T, C = 0, gives an identical effect with C = 1.

Example.

To punch the contents of successive minor cycles of DL10 on successive rows of one or more cards, starting with m.c. 0.

	Flow Diagram	Corresponding Coding
1 ₀	1 ₂ - 13	1 ₀ 1, 1 - 13, 0, 1
1 ₃	10 - 24	1 ₁ 1, 28 - 25, 0, 2
	↓	1 ₂ (1, 10 - 29, 23, 24 X)
1 ₅	13 - 0	1 ₃ 1, 10 - 24, 0, 0
Q ₇ OBEY	(10 _n - 29 X)	1 ₅ (0) 13 - 0, 1, 0, 0
1 ₁	28 - 25	

When instruction 1_5 is reached for the first time, TS13 contains the instruction '1, 10 - 29, 23, 24'. Since 1_5 has zero Timing number, the planted instruction enters TS COUNT in m.c. $5 + 0 + 2$, i.e. m.c. 7. Its Timing and Wait numbers must therefore be calculated as though it had been stored in m.c. 7 of a DL. The notation ' Q_7 ' stands for "Quasi 7" meaning that the Instruction must be coded as if stored in m.c. 7. In successive repetitions of the loop, instruction 1_1 modifies the content of TS13 to "1, 10 - 29, 23 + n, 24". At the first two repetitions, this instruction occupies a time of 26 m.c. ($T + 2$), subsequently it occupies 58 m.c. since W is now greater than T . The programme could have been arranged to save some of this time, but there would have been little point in this as the loop takes at most two Major Cycles and instruction Q_7 can be repeated only at intervals of about 38 major cycles as the successive rows of the card pass the punching station.

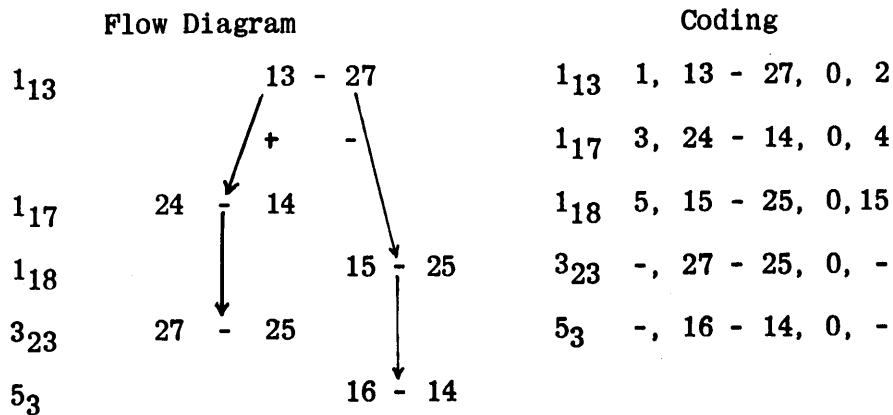
The problem of time assessment is greatly simplified by the fact that a repeated loop must occupy an exact number of Major Cycles. The details of all the above examples should be closely studied to see how the required effects are achieved by the given values of the instruction parameters.

5.14. DISCRIMINATION INSTRUCTION.

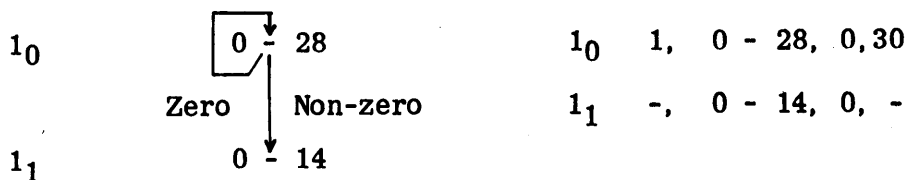
The last example includes no provision for leaving the repeated loop when the required number of values has been punched from DL10. To meet this requirement, use may be made of the Discrimination facilities, which are closely linked with the Control mechanism. It will be remembered that a transfer to D27 or D28 may be followed by either of two next instructions depending on the number being transferred.

If an instruction uses Destination 27 or 28 the next instruction may enter TS COUNT one minor cycle later than normal, that is, in m.c. $m + T + 3$. This will happen only if the number being transferred is negative in the case of D27 and non-zero when using D28. Otherwise, the operation is normal. There is no other effect. The result is the selection, as next instruction word, of either that in DL "N", m.c. $m + T + 2$ or that in DL "N", m.c. $m + T + 3$.

Example.



The N and T numbers of instruction 1₁₇ and 1₁₈ depend on the instruction which follows each of them.



In each case, the two instructions which may follow a discrimination are in adjacent minor cycles of a DL. The timing number is that which would, in any other type of instruction, always select the first of this pair. In the above examples, the discrimination transfer was for only one minor cycle. Where a succession of different numbers is transferred to D27 or D28, the next instruction comes from m.c. $m + T + 3$ if any of them is negative, or non-zero, and from m.c. $m + T + 2$ only if all are positive or zero.

OTHER SOURCES AND DESTINATIONS

6. OTHER SOURCES AND DESTINATIONS.

The facilities associated with the various Source and Destination numbers will now be described in detail. Some of them have already been briefly mentioned.

6.1. USEFUL CONSTANTS.

Five of the Sources give constant numbers. For reference, a word consisting of 31 zeros and a "1" in the nth digit is called "Pn". The five Sources are:-

Source	Output	
S27	P1	- the least significant digit
S28	P17	- this is the units digit in the Wait number (see above).
S29	P32	- The most significant digit.
S30	zeros	- a transfer from this source to a storage position clears the latter.
S31	ones	- represents -1 in the Signed convention.

6.2. LOGICAL OPERATIONS.

Four Sources give various functions of the numbers in TS14 and TS15. Two of them depend only on TS14, the other two on both TS14 and TS15. For brevity, the symbol "TS14" is here used to mean 'the word in TS14'. For instance, "TS14 \div 2" means 'one half (in the Signed convention) of the number in TS14'.

Source.

S23	TS14 \div 2	The digits P1 to P31 are copies, respectively of the digits P2 to P32 of the word in TS14; P32 is a copy of P32 in TS14. This is true division (without round-off) in the Signed convention.
S24	TS14 x 2	The P1 digit is zero; digits P2 to P32 are copies, respectively, of the digits in TS14, P1 to P31. This is true multiplication, in either the Signed or Unsigned convention, provided that the capacity of the machine is not exceeded.
S25	TS14 & TS15	This has a digit '1' only in positions where both TS14 and TS15 contain a "1", and zeros everywhere else.
S26	TS14 \neq TS15	This has a digit "1" wherever the digits in TS14 and TS15 are different and a '0' where they are the same. (spoken as '14 not equivalent to 15').

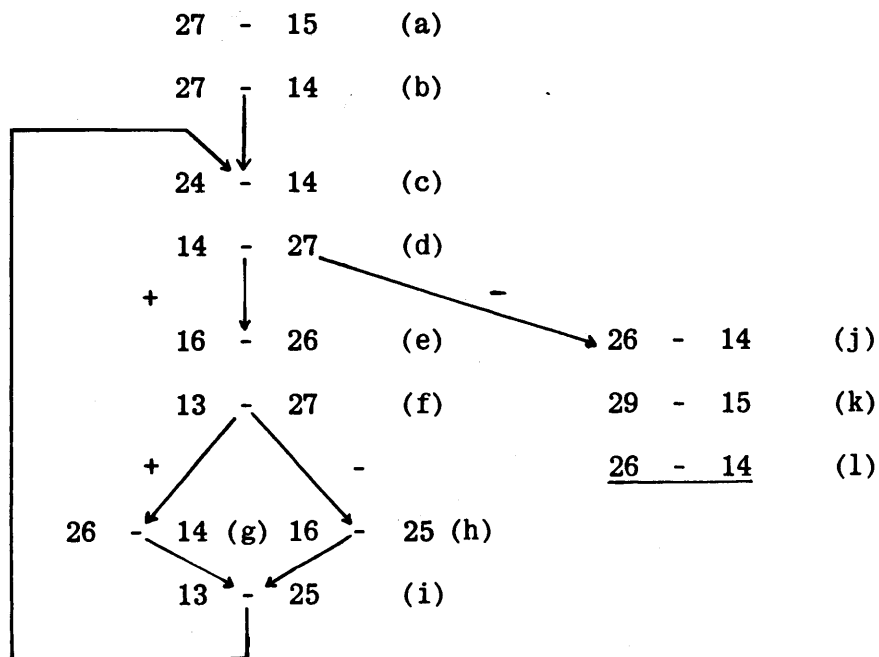
An example will be given of the words at the various Sources for particular contents of TS14 and TS15; only the first eight and last four digits are shown, (the least significant digit is on the left).

Word in TS14	11000111	0011
Word in TS15	10101010	1110
S23 (TS14 ÷ 2)	1000111	00111
S24 (TS14 x 2)	011000111	001
S25 (TS14 & TS15)	10000010	0010
S26 (TS14 \oplus TS15)	01101101	1101

Example.

The following programme divides the number in TS13 by that in TS16 giving the quotient in TS14. Both initial numbers must be positive, the dividend being less than twice the divisor. A few added instructions would enable the programme to deal with numbers of either sign. This programme is given purely for illustration, since division is normally done by using the automatic divider.

The quotient will have 31 digits, one above the binary point and 30 below; its 32nd (top) digit will always be zero.



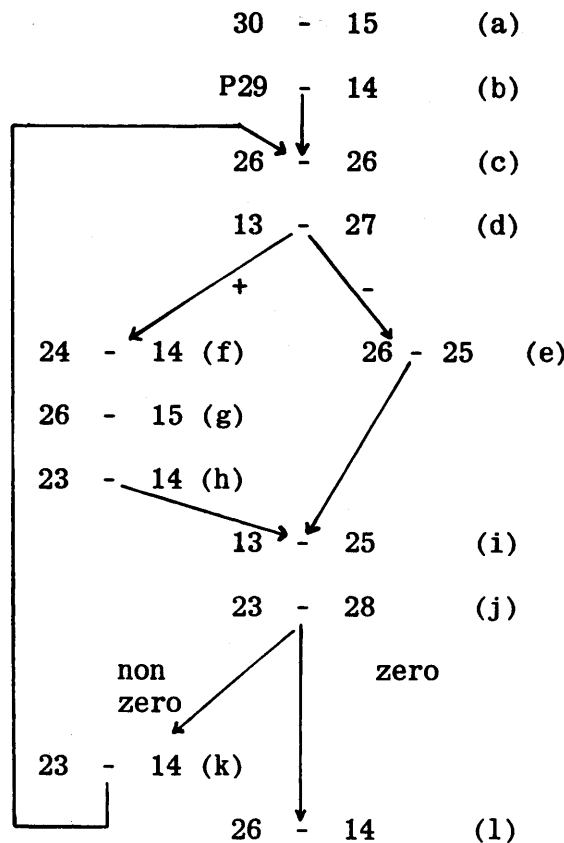
Notes.

- (a) Put P1 in TS15; this remains throughout the division, and is used to generate the successive digits of the quotient.
- (b) Put P1 in TS14; this is the "marker digit". It is shifted up one place as each digit of the quotient is formed; when it reaches the P32 position, the division is complete.
- (c) Move the contents of TS14 up by one place. TS14 contains the marker digit and the digits so far found of the quotient. These digits are found successively, from the top, each being inserted in the P1 position; the partial quotient is then shifted up, by this instruction, to make room for the next digit.
- (d) Has TS14 a "1" in the P32 position? Not until the marker digit has been shifted 31 times, which is after 30 repetitions of the loop. At this stage, the marker digit is in P32, and the top quotient digit is in P31, as required for the final answer.
- (e) Subtract the divisor from what remains of the dividend.
- (f) Is the difference positive or negative?
- (g) If positive, the divisor "goes" into the remainder, and the current quotient digit is "1". This instruction has the effect of adding P1 into TS14, in which the P1 digit is always zero at this stage. TS15 has no other non-zero digits, so the other digits of TS14 are undisturbed.
- (h) If negative, the divisor does not "go" into the previous remainder; the current quotient digit is "0". This instruction "restores" the remainder ready for finding the next quotient digit.
- (i) This doubles the remainder, thereby shifting it up one place. When dividing on paper, one effectively lets the remainder stand and shifts the divisor down one for each digit of the quotient, but the effect is the same as this.
- (j) At this stage, TS14 contains the marker digit in P32, the top 30 digits of the quotient in P31 to P2, and the P1 digit is zero. This instruction merely changes the P1 digit to "1". Previously, the number in TS14 represented the quotient with a possible error ranging from 0 to -2 in the lowest place; now the range of error is from 1 to -1. It is desirable, wherever possible, to make the range of error symmetrical about zero. In this case, a further repetition of the loop would have halved the range of error, but to make it symmetrical would then have required several extra instructions.
- (k) Put P32 in TS15.

- (1) Remove the marker digit. TS14 now contains the quotient, and we can proceed to the next stage of the main programme.

Example.

A more practical example is the calculation of a square root, again by the schoolroom method. In this case the marker digit is shifted down at each stage and the answer kept still, instead of keeping the marker digit at P1 and shifting up the answer, as was done in the division. The remainder is again shifted up at each stage. This reflects the fact that in normal square root process the answer shifts up one place and the remainder two places at each stage. The process actually finds $\sqrt{2^{30}x}$ rather than just \sqrt{x} , because the latter would have only half the number of significant digits of x itself. Again, the answer is given rounded to the nearest odd integer. The arithmetic process of square root is more complicated than that of division; the reader is advised to verify the programme with a few examples. The number x is originally in TS13 and the answer ($\sqrt{2^{30}x}$ to the nearest odd integer) is produced in TS14. The process will work only for values of x less than 2^{30} .



- (a) Clear TS15, in which the answer is to be built up.
- (b) TS14 contains the marker digit; this sets its initial position at P29.
- (c) S26 gives $14 \neq 15$; TS14 holds the marker digit and TS15 the partial answer which at any stage contains only those digits of the answer above and not including the marker digit. Thus S26 gives the sum of the partial answer and the marker digit; this sum is subtracted into TS13, which contains the remainder (in the first instance, the original number x).
- (d) Is the result positive or negative?
- (e) If negative, restore the remainder.
- (f) If positive, shift the marker up one place ---
- (g) --- add it to the partial answer ---
- (h) --- and shift the marker digit down again. Instructions (f), (g) and (h) have the effect of adding twice the marker digit to the partial answer.
- (i) Shift up the remainder by one place.
- (j) Would $TS14 \div 2$ give the answer zero? This will happen for the first time when the marker digit has been shifted down to P1 by repeated applications of instruction (k). By then, instructions (c) to (i) will have been repeated 29 times.
- (k) Shift the marker digit down one place and proceed to calculate the next digit.
- (l) At this stage, TS15 contains the digits of the answer from P30 to P2 and TS14 contains P1. This instruction adds P1 to the previous answer and puts the result in TS14, completing the round off as specified.

6.3. OPERATION OF THE PUNCH AND READER

The instruction "10-24" causes a succession of cards to start passing through the Punch. As each successive row comes into position under the punching knives, there is punched on columns 21 to 52 of that row the configuration currently on the Output Staticiser. The OS is automatically cleared on starting the Punch and immediately after punching each row; the configuration to be punched on the next row must be sent to D29 some-time after this clearing of the OS and the arrival of the next row. To assist in the timing, a Single-Shot signal is sent from the Punch to the DEUCE just before the arrival of each row at the punching knives. The normal procedure is to make all these transfers to D29 stoppers, so that not more than one such transfer is made for each row. There remains the

danger that there is so much computation between some two adjacent stoppers that the Single-Shot signal which should have operated the second has passed and gone before the instruction concerned has arrived in TS COUNT. This danger must be guarded against by the programmer.

The twelve Single-Shots relating to a given card follow each other at intervals of not less than 38 Major Cycles. The time between the last Single-Shot of one card and the first Single-Shot of the next is not less than 116 Major Cycles. These times, of course, determine the maximum amount of computation which can be done between rows and between cards. A complete schedule of relevant timings on the Punch and Reader is given as an appendix. The Reader, by the way, runs about twice as fast as the Punch.

It is useful to be able to distinguish the last row of a card from any other. For this purpose, a signal is provided called TIL (Twelfth Impulse Line), which lasts from shortly before the arrival of the last row at the knives to some time afterwards. TIL is used by the instruction "2 - 24" which has the effect of sending TIL signal to D28; if TIL is present, the next instruction is taken in m.c. $m + T + 3$, otherwise in m.c. $m + T + 2$, where the instruction "2 - 24" enters TS COUNT in m.c. m and has timing number T . This enables a different course to be followed after the last row from that followed after any of the others.

The Punch may be stopped at any time by the instruction "9 - 24". If this is done in the middle of a card, the rest of the card will run through before the motor stops, but there will be no more Single-Shots or TIL signal and no further punching on that card will be possible. If punch is called again by "10 - 24" while this card is running out, it will take effect only at the end of the card, and will then cause another card to follow with full Single-Shot, TIL and punching facilities.

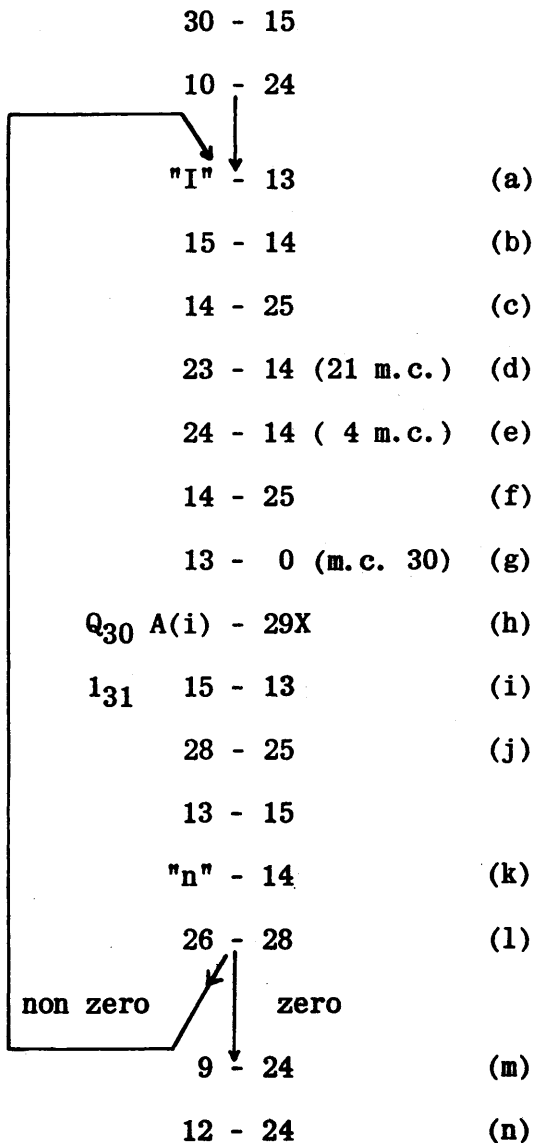
If punch is called when there are no cards in the Punch, it will take effect only when cards have been run in by the operator.

The operation of the Reader is very similar. It is called by "12 - 24" and stopped by "9 - 24". The same instruction is used to stop both the Punch and Reader because they are never in practice used simultaneously. On starting to read, Source 0 is switched from the Input Dynamiciser lamps on the Control Panel to the reading brushes; at the end of reading, SO reverts to the ID which still has the same configuration as at the beginning (unless it has been changed manually in the meantime). The Single-Shot and TIL facilities are identical with those on the Punch.

The Reader has the additional facility of being stopped by a hole punched in column 54 of the card being read. As such a hole passes the reading brushes it stops the Reader just as would the instruction "9 - 24"; the Single-Shot occurs for that row (the one in which P34 is punched), but not for any subsequent ones. There is no corresponding facility for the Punch.

Example.

To punch out results, up to 128 in number, on successive rows of several cards and then proceed to read in more data. The number of results is determined by a parameter "n", and they are stored in 7₀, 7₁, --- 7₃₁, 8₀ etc. up to 10₃₁. They will be referred to as A(0), A(1), --- A(i), --- A(n-1), the number "i" being stored in TS15.



Read more
data

Notes.

- (a) "I" is "1,7 - 29, 0,31,X".
- (b) Place "i" in TS14, which has shifting facilities.
- (c) Add "i" to the Wait number. Since the maximum value of "i" is 127, reaching the 23rd digit, this addition can never spill over into the Timing number, which starts in the 26th digit.
- (d) Shift down 21 places, wiping out the lowest five digits of "i". What is left must be added to the Source number.
- (e) Shift the result of instruction (d) into the Source number position.
- (f) Add it to the instruction.
- (g) The modified instruction enters TS COUNT in m.c. 30, so that its Wait and Timing numbers are respectively the minor cycles of transfer and of the next instruction.
- (h) The modified instruction must be "1, (7 + a) - 29, b,31,X", where "i" has reached the value (32a + b) "b", in fact, is the lowest five digits of "i", and "a" the remaining digits, reduced to units of P5.
- (i) The storage location of this instruction determines the N and T numbers of "I".
- (j) Add P17 to "i". This happens in every cycle, so that TS15 contains "i" units of P17, ready to add into the Wait number without the need for any shifting.
- (k). Place "n" in TS14. Since this is to be compared with the "i" x P17 in TS15, "n" must originally have been punched (or set on the I.D.) in units of P17. This is very common practice.
- (l) S26 gives zero only if the numbers in TS14 and TS15 are equal; that is if "i" and "n" are equal. The machine therefore proceeds back to instruction (a) if there are more results to be punched, and otherwise to instruction (m).
- (m) Stop the Punch.
- (n) Start the Reader.

6.4. MAGNETIC INSTRUCTIONS.

Operations involving the magnetic store are initiated by instructions using D30 and D31. D31 is used for head shifts and D30 for reading or writing.

The Characteristic of the instruction determines whether the reading or writing apparatus is intended. An even characteristic gives reading (D30) or shift reading heads (D31); an odd characteristic gives writing (D30) or shift writing heads (D31).

Examples.

"5 - 31(0)" shifts the Reading Heads into position 5.

"9 - 31(1)" shifts the Writing Heads into position 9.

"15-30(1)" writes the contents of DL11 on to the track now under writing head 15.

" 0-30(0)" reads the contents of the track now under reading head 0 into DL11.

To read from track 121 (7/9) first do "7 - 31(0)" which shifts the reading heads into position 7, and then "9 - 30(0)" which reads into DL11 the track now under reading head 9.

Having been initiated by one of these instructions, each operation takes a fixed time to complete, 13 Major Cycles for reading or writing and 64 Major Cycles for a head shift. During this time, any sequence of instructions may be obeyed which do not use DL11 or the Magnetics. If during the course of a magnetic operation, however, an instruction enters TS COUNT which uses S11, D11, D30 or D31, this instruction is held suspended in TS COUNT without being obeyed, as though it were a stopper; it is obeyed only on completion of the current magnetic operation. This feature is known as "Control-Magnetics Interlock" or CMI.

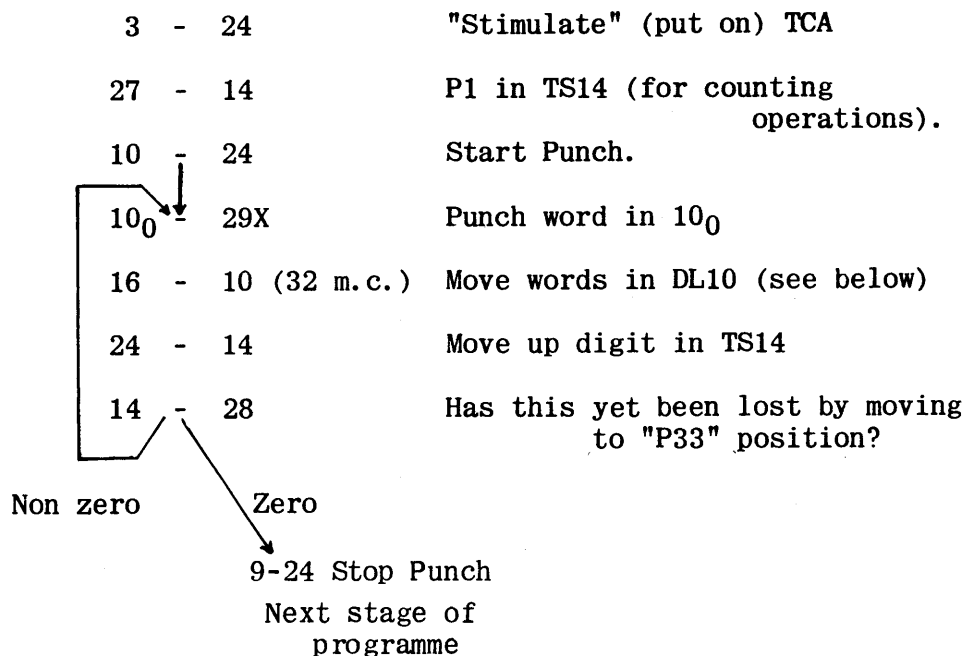
6.5. TRIGGER TCA.

Normally S16 is the Source of the word in TS16, but it has an alternative function. Instead of the successive digits in TS16, it may give the digits in DL10, delayed by one minor cycle. When it is performing its second function, the words in DL10 are available at two Sources, but in different minor cycles. The word stored in DL10 in m.c. 5, for instance, also appears at S16 in m.c.6. In this case, TS16 no longer acts as a storage position, and any word previously stored in it is lost.

The alternative functions of S16 are controlled by an electronic switch called "TCA". This is said to be "off" when S16 gives the word in TS16, and "on" when it gives those in DL10 a minor cycle late. TCA is put on by the instruction "3 - 24". There is no point in putting it off until TS16 is again to be used as a storage position. TCA is therefore automatically cleared at the beginning of any transfer to TS16 and no other provision is made for clearing it.

Example.

In a previous example, the facilities associated with S28 and D0 were used to carry out an operation on the numbers stored in successive minor cycles of DL10. This operation was that of punching the numbers on successive rows of a card. TCA provides an alternative method. When TCA is on, the transfer "16 - 10 (m.c. 5)" replaces the word in 10₅ with that in 10₄, leaving two copies of the latter and none of the former. Extending this, the transfer "16 - 10 (32 m.c.)" replaces the word in each m.c. of DL10 with that formerly stored in the previous minor cycle.



Moving the words in DL10 puts that formerly in m.c. 31 into m.c. 0, and so on. On successive repetitions of the loop, the words punched on successive rows are those originally in 10₀, 10₃₁, 10₃₀, etc., round to 10₁. This reverses the order obtained by the previous method. However, the numbers must first have been placed in DL10, after calculating them, by using either TCA or S28 and D0. So long as the same one of the two methods was used both for putting them in and taking them out, the numbers will be punched in the same order as that in which they were calculated.

The first method given of selecting successive words in a Delay Line is more flexible; the order of selection may be reversed by subtracting one from the Wait number of the transfer instruction, instead of adding it, and the method applies equally to any DL, whereas TCA is connected only with DL10, and always works backwards. Where the repeated loop is more complicated, and uses TS16, TCA must be Stimulated separately each time the move round is required.

Example.

3 - 24 This pair of instructions may be inserted in any
 16 - 10 (32 m.c.) repeated loop; the word previously stored in
 TS16 is lost, however.

6.6. DOUBLE-LENGTH ARITHMETIC.

The 64 digits in a DS may be treated as comprising either one double-length number or two single-length numbers. To allow for this, the operation of arithmetic facilities associated with DS21 is modified by an electronic switch called "TCB"; this is put on and off respectively by instructions "5 - 24" and "4 - 24". When TCB is off, the arithmetic is double-length; when it is on, single-length.

No special provision is required for simple transfers; the instruction "20 - 21 (2 m.c.)", for instance, transfers the 64 digits in DS20 into DS21, whether they comprise one or two separate numbers. The only distinction, in fact, lies in the treatment in arithmetic operations of the 32nd (most significant) digit of the two minor cycles. In double-length numbers, the less significant half is always assumed to be in the even minor cycle.

Numbers transferred to D22 are added into DS21. If TCB is on, DS21 acts as two separate single-length accumulators; any carry from the 32nd digit of either minor cycle is lost, and is not added to the 1st digit in the next minor cycle. If TCB is off, however, any carry generated from the 32nd digit of the even m.c. is added to the first digit of the odd m.c.; carry from the 32nd digit of the odd m.c. is lost in either case. Numbers transferred to D23 are subtracted into DS21 with the same provision for single-length or double-length arithmetic controlled by TCB.

For both addition and subtraction, there is a special provision for operating on the bottom half of a double-length number. This provision effectively extends to double-length a single-length number added or subtracted into the even minor cycle of DS21 when TCB is off; putting this another way, if TCB is off and a single-length number is transferred to either D22 or D23 in an even minor cycle, 32 copies of the sign digit of this number are respectively added or subtracted into DS21 in the following odd minor cycle. This provision also comes into effect if the transfer lasts for several minor cycles ending with an even one.

Examples.

20 - 22 (2 m.c.). If TCB is on, this adds the two single-length numbers in DS20 to those in DS21. If TCB is off, it is intended to add the double-length number in DS20 into DS21; if the even P32 digit in DS20 is a "1", however, and the transfer is for an odd m.c. followed by an even one, the special provision mentioned above will result in an unwanted "1" being subtracted into the top half of DS21. To make sure of getting the correct result, the transfer must be for an even minor cycle followed by an odd; this is indicated by writing the instruction "20 - 22 (2 m.c. e,o)".

15 - 23 (odd m.c.). The number in TS15 is subtracted into the odd m.c. of DS21, which may contain either a 32-digit number or the top half of a 64 digit number.

15 - 22₂. In this case, the state of TCB affects the operation. If it is on, so that DS21 contains two single-length numbers, carry from the 32nd digit of the addition is suppressed, and does not interfere with the number in the top half of DS21. If TCB is off, DS21 contains one double-length number, and the process of addition effectively extends the number in TS15 to double-length.

19 - 22 (6 m.c. e,o), (TCB off). This adds three times the (double-length) number in DS19 to that in DS21.

21 - 22 (2 m.c. e,o). (TCB off). This doubles the number in DS21.

21 - 22 (4 m.c. e,o). (TCB off). This quadruples the number in DS21, i.e. shifts it up by 2 places.

6.7. SOURCE 22.

It will be remembered that S23 gives the number in TS14, divided by two in the Signed convention. S22 is similar in operation; it gives the number or numbers in DS21, divided by two in the Signed convention. If the digits in 21_2 are $A_1, A_2 \dots A_{32}$ and those in 21_3 are $B_1, B_2 \dots B_{32}$, the digits at S22 will be $A_2, A_3 \dots A_{32}, X$ in the even m.c. and $B_2, B_3 \dots B_{32}, B_{32}$ in the odd m.c. "X" depends on the state of TCB. If TCB is off, X is B_1 ; if it is on, X is A_{32} . Thus, with TCB on, the two words at S22 are independent, and give respectively one-half of the words in DS21. With TCB off, S22 gives one half of the double-length number in DS21, each digit being shifted down by one except for the 64th, which is repeated.

Examples.

If TCB is off,

22 - 21 (2 m.c. e, o) halves the number in DS21.

22 - 21 (2n m.c. e, o) divides the number in DS21 by 2^n .

Transfer must always start with an even m.c. and end with an odd m.c.

22 - 22 (2 m.c. e, o) multiplies the double-length number in DS21 by 1.5.

6.8. MULTIPLICATION.

The DEUCE contains an automatic multiplier associated with DS21 and TS16. The answer produced is the 64-digit product of two 32-digit numbers, true in the Unsigned convention. The two factors are the words previously in TS16 and DS21₃. DS21₂ must be empty at the start of multiplication. Multiplication is started by the instruction "0 - 24", which must be obeyed in an odd m.c. The process of multiplication occupies 65 m.c. or just over two Major Cycles. During this time, D16, D21, D22 and D23 must not be used, and S21 and S22 will give sequences of digits representing a partly completed product. TCB is automatically cleared at the start of a multiplication, and is left off at the end. TCA is automatically cleared when the multiplicand is sent to TS16 and must not be switched on again before completion of the multiplication. The completed product appears in DS21.

Example.

To multiply the positive numbers in TS16 and TS14 and send the result to DS19.

	m. c.		m. c.	
(a)	0	30 - 21 ₂	0	30 - 21, 0, 0
(b)	2	14 - 21 ₃	2	14 - 21, 1, 1
(c)	5	0 - 24 (odd m.c.)	4	30 - 29, 1, 0
(d)	4	30 - 29	5	0 - 24, 0, 29
(e)	6	21 - 19 (2 m.c.)	6	21 - 19, 2, 0, 1
	9	etc.	9	next instruction

Notes.

- (a) (b) Before starting multiplication, 21₂ is cleared and the multiplier sent to 21₃.

- (c) Multiplication is controlled by an electronic switch called MULT. This is stimulated at the start of the transfer 0 - 24 and then remains on for a period of 65 m.c. after which it is automatically cleared. Further or continued transfer 0 - 24 during multiplication is ineffective. Instruction (c), even if it had characteristic 1, would give transfer only from m.c. 7 to m.c. 4 of the next Major Cycle, inclusive; MULT remains on till the end of m.c. 7 of the Major Cycle after this.
- (d) The product is not yet available, and a dummy instruction is inserted to use the time until it is. Since W exceeds T, the instruction occupies 34 m.c., which added to the 31 m.c. occupied by instruction (c), gives the 65 m.c. for multiplication.
- (e) Transfer is in m.c. 8 and m.c. 9. This is the tightest possible timing since multiplication finished only at the end of m.c. 7. The period of multiplication may be used for useful instructions if the programme permits. For instance, if the factors are liable to be negative, certain corrections must be added to the product to give the correct Signed result. These may be calculated during the multiplication.

6.9. SIGNED MULTIPLICATION.

A negative number "-a" is represented by 2^{32-a} . If we multiply such a number by a positive number "b", the "product" will be $(2^{32-a})b$, or $2^{32}b-ab$. The double-length product of "-a" and "b" is represented in the Signed convention by 2^{64-ab} . A correction must therefore be added equal in value to $2^{64-2^{32}}b$, or $2^{32}(2^{32}-b)$. This is equivalent to adding the single-length signed representation of "-b" to the top half of the "product".

If two negative numbers are multiplied, the immediate result is $(2^{32-a})(2^{32-b})$, or $2^{64-2^{32}a-2^{32}b+ab}$. The required addition to the top half is here $(a + b)$, or $-(-a)-(-b)$, since only "-a" and "-b" are immediately available. The correction for any pair of signs is made up as follows; if either factor is negative, the other is subtracted from the top half of the result of the multiplication.

Example.

To multiply the numbers in TS14 and TS16, allowing for the sign of either.

<u>DL2. m. c.</u>		<u>Coding.</u>
(31)	30 - 21 ₂ (a)	m. c.
(19)	14 - 21 ₃ (b)	19 2, 14 - 21, 0, 0
(21)	0 - 24 (c)	20 2, 16 - 27, 0, 1
(25)	14 - 27 (d)	21 2, 0 - 24 0, 2
	+ - (e)	22 2, 13 - 23, 1, 2
(29)	30 - 13(30) 16 - 13 (f)	23 2, 30 - 29, 0, 29
	+ - (g)	24 2, 14 - 25, 0, 28
(20)	16 - 27 (g)	25 2, 14 - 27, 0, 2
	+ - (h)	26 2, 21 - 22, 1, (32-2n), 31
(23)	(h) dummy (24) 14 - 25 (i)	27 1, 29 - 23, 1, 1
	+ - (j)	28
(22)	(j) 13 - 23 ₃ (after mult)	29 2, 30 - 13 0, 21
(26)	(k) 21 - 22 (2n m. c. e, o)	30 2, 16 - 13 0, 20
(27)	(l) 29 - 23 ₂	31 2, 30 - 21, 1, 18
1 ₃₀	Next part of programme.	

Notes.

- (a), (b), (c) Start multiplication as before.
- (d) to (i) Build up the sign correction ready to subtract into the top half of the product after multiplication. The correction consists of zero if both factors are positive, one factor if only the other is negative, and the sum of both factors if both are negative.
- (h) The coding must allow a time of at least two major cycles between (c) and (j). If the dummy were omitted, a third major cycle would elapse between (g) and (j) if the number in TS16 were positive.

- (j) Subtract in the correction. This happens in m.c. 25 (see coding), whereas multiplication started in m.c. 23 two Major Cycles before. In other words, we are just all right.
- (k) Shift the product up n places. n can be any number from 1 to 16, and is determined by the Wait number of the instruction (in m.c. 26). This allows for binary places. When working to 27 b.p., for example, the product of two numbers with 27 b.p. each has 54 b.p. The binary point after multiplication comes between P22 and P23 of the top half. Shifting up by five b.p. brings the point between P27 and P28 of the top half. Taking this top half as the answer leaves us still with 27 b.p.
- (l) Subtract P32 into 21_2 . Well, that's what it looks like. The actual effect is to add P32 into 21_2 . Remember that a transfer to D22 or D23 in an even m.c. only, with TCB off, of a number which has a "1" in P32 position, is automatically followed by the transfer of 32 "1" s to the same destination. This means that P32 (even) when sent to D22 or D23 looks just like -P32 (even) so that in order to add it you have to subtract. The object is to balance the error, as was done at the end of the division and square root examples. It is assumed that only the top half of the product is to be taken on to the next stage of the calculation. Without this preliminary addition of P32 (even) the error caused by dropping the bottom half would vary from zero to -1 in the bottom binary place. With the round off, the resultant error varies from $-\frac{1}{2}$ to $+\frac{1}{2}$ in the bottom place.

The coding of this example may appear a little eccentric, but there is method in the madness. The salient points are that all the instructions are crammed as tightly as possible into bottom of DL2 (except for the gap at 2₂₈ which will be explained) and that the last instruction of the multiplication, 2₂₇, leads on to 1₃₀ for the first instruction in the next part of the programme. The reason for both of these manoeuvres will appear in section 7.1.

6.10. DIVISION.

There is also an automatic divider associated with TS16 and DS21. This accepts divisor and dividend of either sign and produces a correctly signed quotient. First, the divisor is sent to TS16 and the Dividend to DS21₃; 21_2 should be cleared. Division is started by the instruction "1 - 24" which must be obeyed in an odd minor cycle; the process takes 66 minor cycles, one more than for multiplication. The quotient appears in 21_2 , and a modified form of remainder in 21_3 .

The process needs more careful definition than does multiplication. The quantities involved are:-

Dividend	A
Divisor	B
Quotient	Q
Remainder	R
Modified form of remainder	r

These are linked by the relationship:-

$$2^{31} A = Q B + R$$

$$\text{for } B > 0, 0 \leq R < B$$

$$\text{for } B < 0, B \leq R < 0$$

$$Q = 2^{31} \frac{A}{B} = 10_{32} \frac{A}{B}$$

(except this is not allowed!)

The limitations are imposed that A, B and Q must all be single-length within the signed convention. Q being single-length implies that $|A| < |B|$ and that $|A|$ and $|B|$ can be equal only if A is negative and B is positive.

$$\text{We thus have } |A| < |B| < 2^{31}.$$

It will be observed that Q is always algebraically less than or equal to the true quotient; where an exact quotient is possible, Q will reach this value only if B is positive and will otherwise fall short of it (algebraically) by one unit in the lowest binary place. Since the result is truncated, a few further instructions are required to give a balanced round-off where this is required.

If the true remainder must be found as well as the quotient, an additional limitation is imposed that $|A| < |B| < 2^{30}$. The modified form of remainder, r, actually found in 21_3 after division, is related to the true remainder R by:-

$$r = 2 (2 R - B)$$

$$\text{So that } R = \frac{1}{2} (\frac{1}{2} r + B)$$

It will be clear that if $2^{30} < B < 2^{31}$, r may be out of single-length in which case R will not be recoverable. This is the reason for the extra limitation.

The divider gives the quotient to 31 binary places of two integers (or two quantities with an equal number of binary places). Shifts may be necessary after division to restore the position of the binary point and before division to ensure that $|B| \geq |A|$

The detailed construction of the divider imposes one slightly awkward limitation; this is that the divisor must not be sent to TS16 in the odd

minor cycle immediately preceding the odd minor cycle in which division is started. TCB, by the way, is stimulated automatically at the start of a division and left on at the end.

Examples.

To divide the number in TS14 by that in TS16, knowing the latter to be larger in absolute value, giving the quotient in TS16 to 30 b.p. with balanced error of $\pm \frac{1}{2}$ in the lowest place.

14 - 21 ₃	(a)
30 - 21 ₂	(b)
1 - 24 (odd m. c.)	(c)
dummy	(d)
27 - 22 ₂ (after division)	(e)
22 ₂ - 16	(f)

Notes.

- (a) dividend to 21₃
- (b) clear 21₂
- (c) start division
- (d) waste time to cover division process
- (e) before this instruction, the quotient has an error ranging from 0 to -1. We are about to drop the bottom digit (instruction (f)). Before doing so, the result should be brought to the nearest even integer (ignoring the bottom digit). If the bottom digit is "1", the rest of the quotient needs increasing by one (i.e. P2); if the bottom digit is zero, the rest of the quotient should be taken as it stands. Adding one in the bottom place carries out this requirement.
- (f) Shift the result of (e) down one place, dropping the P1 digit (remember TCB is on).

To divide the number in TS16 by the number in TS14 to 31 binary places knowing the latter to be the greater and less than 2^{30} , and to find the exact remainder.

16 - 21 ₃	(a)
30 - 21 ₂	(b)
14 - 16	(c)
1 - 24 (odd m. c.)	(d)
dummy	(e)
22 - 21 ₃ (after div.)	(f)
16 - 22 ₃	(g)
22 - 21 ₃	(h)

- (a) dividend to 21₃
- (b) clear 21₂
- (c) divisor to TS16
- (d) start division. N.B. if (c) transfers in an odd m.c., coding must be such that start of division is not in the immediately following odd minor cycle.
- (e) waste time to allow for division
- (f) shift down modified remainder ---
- (g) --- add in divisor ---
- (h) --- and shift down again. (f), (g) and (h) generate the true remainder R from the modified remainder r by the formula $R = \frac{1}{2} (\frac{1}{2} r + B)$.
The quotient to 31 b.p. is in 21₂.

SUB-ROUTINES

7.0. SUB-ROUTINES.

7.1. A SUB-ROUTINE.

In the example above in 6.9, a programme of twelve instructions was given for multiplying two numbers. At this rate, a computation involving much multiplication would soon fill the Machine with instructions solely for that purpose. However, it is possible to store this group of instructions just once, and to call on them whenever a multiplication is required.

Example.

To replace the number in TS14 with its fourth power.

(a) 14 - 16

"MULT"

(b) 21_3 - 14

(c) 14 - 16

"MULT"

(d) 21_3 - 14

The word "MULT" represents the group of 12 instructions from the previous example. It has the effect of forming in DS 21_3 the product of the numbers initially in TS16 and TS14. Its first instruction lies in m.c. 31 of DL2, and its last in m.c. 27. Thus both instructions (a) and (c) are followed by instruction 2_{31} . This is easy. However, instruction 2_{27} must be followed in the first case by (b) and in the second by (d). This is achieved by placing first (b) and then (d) in the storage location specified by 2_{27} as containing its successor.

Main Programme		MULT Sub-Routine	
1 ₀	14 - 16	2 ₂₈	13 - 1 ₃₀
1 ₂	1 ₄ (Link A) - 13	2 ₃₁	30 - 2 ₁₂
		2 ₁₉	14 - 2 ₁₃
2 ₂₈			
to	MULT		and so on, as in previous example
2 ₂₇			
1 ₃₀ (link A)	21_3 - 14	2 ₂₆	21 - 22 (2n m.c. e, o)
1 ₁	14 - 16	2 ₂₇	29 - 2 ₃₂ (a)
1 ₃	1 ₆ (Link B) - 13	1 ₃₀	Link
2 ₂₈ enter	MULT		
1 ₃₀ (link B)	21_3 - 14		

Note (a)

227 specifies, as its next instruction position 130, which now contains the word which was in TS13 at the start of MULT.

A group of instructions for performing a particular operation, such as multiplication, is called a "Sub-Routine". If it is to be used more than once in the programme, the instruction which follows it in each case is called a "link". In order to be used, the link must be placed in a position fixed by the last instruction of the sub-routine; this fixed position is usually 130. Before entering the sub-routine, the link is placed in a vacant TS; it is transferred to 130 by the sub-routine itself. The alternative method of placing the link directly in 130 from m.c. 30 of another DL would not work if the sub-routine were to be used more than eleven times, and is therefore discarded in the interests of uniformity and ease of programming. The use of m.c. 30 leads to some simplification of arithmetic, since the Wait and Timing numbers of an instruction stored there are precisely those of the first and last minor cycles of the specified transfer.

The advantages of the method can be seen from the example. The 12 instructions of multiplication are called upon by only one instruction, that which plants the link in TS13. The cost is the addition to the process of one instruction "13 - 130", and the forfeiture of 130 as a normal storage position. This last, however, will do for any number of sub-routines, since all use 130 as their link position.

If the position of the binary point varies in the programme, the shift after multiplication must also be altered. In this case, instruction 226 must be modified before each entry to the sub-routine.

In the example, the sub-routine takes a total time of four Major Cycles. This would have been reduced by planting the link from TS13 to 130 only after multiplication had been started; the time occupied by planting the link would then become part of the 2 Major Cycles of multiplication, which have to be wasted anyway.

Furthermore, the coding given takes the same time for any other number of places of shift as for the maximum (16 places). If the number of places were known in advance, it might be possible to recode so as to save still more time. It is generally worth taking a great deal of trouble in coding sub-routines, since these are the parts of a programme which are most often repeated.

7.2. OTHER SUB-ROUTINES.

As well as for multiplication, sub-routines may be made for Division, Square Root, Double-Length Multiplication, Division and Square Root, and other simple operations. These are called "First-Order Sub-Routines". From these, more complicated Sub-Routines may be built up.

A Sub-Routine may be made for the Summation of Polynomials. This incorporates a Multiplication Sub-Routine, and is therefore called a "Second-Order Sub-Routine". This, in its turn, may be part of a Sub-Routine for taking Logarithms, which would be a Third-Order Sub-Routine.

A Second-Order Sub-Routine plants its Link in l_{31} , since l_{30} is used during its operation by the inner Sub-Routine. In general, an Nth-order Sub-Routine plants its link in l_{29+N} .

7.3. LIBRARY SUB-ROUTINES.

The technique of Sub-Routines has a further great advantage. Once a Sub-Routine has been made for, say multiplication, and punched on cards to occupy half of DL2, it may be used in exactly the same form in any programme where it is required. A library is kept of all Sub-Routines as they are made for a particular programme, and these are available for any future user. For the sake of flexibility, several copies are made of each, storing the instructions in different Delay Lines or halves of Delay Lines. For instance, there are fourteen copies of MULT, using the top and bottom halves of DL2 to 8. The Sub-Routines are not punched for storage in DL1, since this is usually occupied by the main Programme.

Fitting library sub-routines together is facilitated by the convention of coding them in a block at one end of a DL, as was seen in the MULT sub-routine above. Larger sub-routines are coded to occupy completely as many DLs as necessary with any extra instructions in a block at one end of a further DL.

As the Library increases, Programming may come to consist mainly of placing words in the positions required by Sub-Routines and planting links. The Sub-Routines are collected from the Library, and copied into the programme by means of a Hollerith Reproducer.

INITIAL INSTRUCTIONS

8. INITIAL INSTRUCTIONS

8.1. INITIAL INPUT.

In reading in a programme, the words to be placed in one DL are punched in order on three cards, leaving the first four rows of the first card free for the special input instructions. Taking DL2 as an example, the three cards would be as follows:-

Row	First Card	Second Card	Third Card
Y	blank	2 ₈	2 ₂₀
X	2, 0 - 2, 1, 26, 25X	2 ₉	2 ₂₁
0	2, 0 - 2, 30, 31X	2 ₁₀	2 ₂₂
1	1, 0 - 2 30, 31X	2 ₁₁	2 ₂₃
2	2 ₀	2 ₁₂	2 ₂₄
3	2 ₁	2 ₁₃	2 ₂₅
:	:	:	:
9	2 ₇	2 ₁₉	2 ₃₁

Depression of the initial input key on the Control Panel has the effect of clearing the whole Delay Line store, and then calling a 'Read' just as it is called by the instruction "12 - 24". The three cards are now run into the Hollerith Read.

TS COUNT is cleared along with the others, and therefore initially contains the instruction "0, 0 - 0, W, T X"; the numbers W and T may have any arbitrary values, because of the organisation of Control. The operation of this instruction is ambiguous. It will be remembered that a transfer to Destination 0 will generally work only if it is a long transfer; otherwise, transfer will have ceased before the TS COUNT gate is opened to let the next instruction in, and this next instruction will come in the usual way from the DL named in the NIS number. However, if $T = W + 1$, the transfer to DO will work for a double transfer and (which is the point here) if $W = T$ it will work for a single transfer instruction ($C = 0$). The instruction is in any case obeyed on the Single-Shot which marks the first row of the first card. If, by accident, $W = T$ after clearing store, this row will be placed in TS COUNT; otherwise, zeros will be placed in TS COUNT from some minor cycle of DL8 (the DL specified by NISO). The only way to overcome the ambiguity is to leave blank the first row of the first card read in after clearing store. This is made a general rule for DEUCE programmes. It is usually achieved on an Initial Card which is also required for other purposes (see below). In the present case, the first row is blank anyway, so no more bother is needed.

Returning to the example, the instruction "0, 0 - 0, W, T X" is obeyed on the first row of the first card. This has the effect of placing "0, 0 - 0, 0, 0 X" (Zeros) in TS COUNT, either from the first row or from some minor cycle of DL8. This is obeyed in turn at the second row, and sends "2, 0 - 2, 1, 26, 25 X" to TS COUNT. At the "0" row, this fills DL2 ~~DL1~~ with 32 copies of "2, 0 - 2, 30, 31 X", and takes one of these as the next Instruction.

The effect of "2, 0 - 2, 30, 31 X" entering TS COUNT in m.c. m is to send the word on the next row to 2_m and take the next instruction from 2_{m+1} . In other words, each of these instructions replaces itself with a row from the card and then proceeds to the next. The first of them is replaced with "1, 0 - 2, 30, 31 X", the second with 2_0 , the third with 2_1 and so on, until the 32nd is replaced with 2_{30} . The next instruction is now "1, 0 - 2, 30, 31 X" which replaces itself with 2_{31} and takes the next instruction from 1_0 . This is still empty, and TS COUNT now contains "0, 0 - 0, 0, 0 X", ready to start on a similar triad of cards relating to, say, DL3.

While the storage is empty, the minor cycles are considered anonymous. The m.c. in which the word "2₀" arrives in DL2 is called "m.c. 0". We will now consider the minor cycles in which the subsequent operations take place.

Minor Cycle.	Operation
0	"2 ₀ " placed in DL2
1	"2 ₁ " ditto
30	"2 ₃₀ " ditto
31	"2 ₃₁ " ditto Instruction has NIS "1".
0	"0, 0 - 0, 0, 0 X" enters TS COUNT from 1_0
2	ditto from card.
4	"3, 0 - 3, 1, 26, 25 X" enters TS COUNT from card.
0 to 31	32 copies of "3, 0 - 3, 30, 31, X" enter DL3 from card.
also in 31	one of these enters TS COUNT.
in a later 31	"1, 0 - 3, 30, 31 X" enters DL3
0	"3 ₀ " placed in DL3
1	"3 ₁ " placed in DL3

etc.

It will be seen that, with the given Wait and Timing numbers, words occupying corresponding position in the triads relating to different Delay Lines will enter their respective DLs, in the same minor cycle. In general, the first four rows of the triad filling DL"N" are:-

blank

"N, 0-N, 1, 26, 25 X"

"N, 0-N, 30, 31 X"

"1, 0-N, 30, 31, X"

There are two exceptions to this. Firstly, in the triad filling the last DL used in the programme, the fourth row is replaced with "M, 0-N, 30, m-1, X" where the first instruction of the programme to be obeyed is in DL"M", m.c. "m". Secondly, since the method uses the fact that each DL has an NIS, a modification is needed, to fill DLs 9 to 12. In these cases, one of the first eight DLs, not yet filled itself, is used as an auxiliary. The first four rows of a triad which filled DL9 by using DL7 might be:-

1, 0-7, 1, 29, 28, X

7, 0-9, 30, 31, X

7, 0-7, 27, 28, X

1, 0-9, 30, 31, X

The first row fills DL7 with copies of the second row; the third row replaces one of these with the fourth row. The other 31 are obeyed in order, placing successive rows in DL9; the fourth row is now obeyed, completing DL9 and returning to NIS "1" for filling the next DL, which might be DL7 itself.

The reader is advised to work through the operation of this group of input instructions in the same way as set out above for the filling of DL2. He will find that DL1 is asked to supply a word of zeros on two occasions and that the Wait and Timing numbers have been cunningly arranged to bring both these references at minor cycle 0. A blank minor cycle somewhere in the machine is always required for this method of filling a DL (as well as a spare DL if DL9, 10, 11 or 12 is being filled). By fiddling the NIS and T numbers, this blank minor cycle may be taken anywhere in the first eight DLs; 1₃₀ is often a convenient minor cycle, though 1₀ has been used in both the examples.

8.2. INITIAL INPUT TO THE MAGNETIC DRUM STORE.

With a large programme, it is often useful to read part of it initially on to the Magnetic Drum and the rest of it in to the D.Ls. A system is

therefore needed for reading triads of 32 instructions on to specified tracks on the Drum. In this case, the first four rows of each triad are insufficient to accommodate the special input instructions. The space is used instead to designate the number of the track on which the subsequent 32 rows are to be stored, and a special routine is used to see them home.

The following card will plant that routine in DL8.

Row	Y	Blank	
X		0, 0-8, 1, 27, 25 X	
0		0, 0-8, 30, 2 X	
1		0, 0-8, 30, 31 X	
2		0, 8-13, 6, 10	Stored 8 ₃
3		0, 0-8, 20, 26 X	8 ₇
4		7, 0-7, 30, 31 X	8 ₁₁
5		0, 13-7, 1, 3, 2	8 ₁₅
6		7, 8-7, 10, 11	8 ₁₉
7		0, 7-11, 1, 3, 2	8 ₂₃
8		0, 8-0, 0, 0	8 ₂₇
9		0, 0-7, 30, 22 X	8 ₃₁

Its own introductory 4 rows differ from normal in that

- (a) X Row. Wait No. 27 instead of 26: to ensure zero in 8₀
- (b) 0 Row. Timing No. 2 instead of 31: After replacing itself, each filler instruction jumps 4 minor cycles instead of the usual one minor cycle getting back to 8₃₁ after 8 jumps instead of 32. The essential instructions are thus stored (as indicated) and TS COUNT returned to zero after one card only.
- (c) 1 Row. NIS 0 instead of 1 : because 8₀ is left blank.

This subroutine will read into track a/b of the drum the last 32 rows of a triad which bears the introductions.

Y - row	Blank
X - row	0, a-31, 1, 0, 1 X
0 - row	Blank
1 - row	0, b-30, 1, 0, 1

The detailed operation is as follows:-

Control contains $Q_0, 0, 0 - 0, 0 0 X$

One shot from:	Instruction obeyed		Action
1st. card Y-row	Q_0	$0, 0 - 0 \quad 0, 0 X$	Y-row enters TS COUNT
X-row	Q_2	$0, 0 - 0 \quad 0, 0 X$	X-row " "
0-row	Q_4	$0, a - 31, \quad 1, 0, 1 X$	Appropriate head block shift started
1-row	8_7	$0, 0 - 8_{29} \quad X$	Files "writing instruction" in 8_{29}
	8_3	$8_1 \Gamma 13$	} 32 copies of "filler" instruction to DL7.
	8_{15}	$13 - 7 \quad (32 \text{ mc.})$	
	8_{19}	$8_3 \Gamma 7_{31}$	Final filler to 7_{31}
2-row	7_0	$0 - 7_0 \quad X$	} DL 7 filled with 32 words from the cards.
3-row	7_1	$0 - 7_1 \quad X$	
etc. to			
3rd. card 9-row	7_{31}	$0 - 7_{31} \quad X$	
	8_{23}	$7 - 11 \quad (32 \text{ mc.})$	DL7 to DL11
	8_{27}	$8_{29} 0$	} DL 11 to track a/b.
	Q_{29}	$b - 30 \cdot 1$	
	8_0	$0 - 0 \quad X$	

The buffer delay line (in this case DL7) has to be used because after giving the head block shift instruction (a - 31 l) DL11 may be inaccessible for a period of about 66 m.s.

Since the return after each triad is to 8_0 , triads for drum and DL's may be in any order provided only that DL8, 11 and 7 are not filled until drum filling is complete. It is usual to read into the drum first and fill the DL's afterwards.

8.3. INITIAL CARD.

To ensure uniformity of conditions, an Initial Card which provides a leading blank row and clears TCB, is placed in front of each programme. Because even and odd minor cycles are still distinguished even when the Store is empty, the Initial Card is also used to ensure that the first word shall be stored during an even minor cycle. One version is :-

Row of Initial Card.	Function
Y blank	Initial blank row
X blank	
0 4 - 24, 0, 0, X	Clear TCB
1 blank	
2 31 - 21,1,0,1,X	Fill DS21 with "1"s
3 blank	
4 27 - 22, 0,0,X	add 1 into DS21. This leaves DS21 either empty or half full, depending on whether it is done in an even or odd minor cycle.
5 blank	
6 21 - 28,1,0,1,X	take next instruction into TS COUNT either 3 or 4 m.c. later depending on whether or not DS21 is now empty.
7 blank	
8 30 - 21,1,0,1,X	clear DS21
9 blank	

8.4. RUNNING IN.

Having made all the necessary triads, some copied from the Library, they are packed together with an initial card and placed in the Hollerith Reader. A key on this is pressed which performs all the functions of clearing Store, calling for a Read and running in the cards. The last card is punched on row 9 in column 54, and this stops the Reader when all have been read in.

8.5. LARGE PROGRAMMES.

A programme too large to be contained in the storage may consist of a sequence of separate operations based on the same initial data. In this case, blocks of new instructions may be read in at the completion of each stage. The method is to punch cards just as for the Initial Input. The pair of instructions "12 - 24;1, 30 - 8, n, n," will then call in the new information; the number "n" is chosen to call the next instruction in m.c. 0.

APPENDIX 1

HOLLERITH TIMING

In order to link the operation of the DEUCE with the Hollerith Read and Punch, it is useful to know the time in Major Cycles between various operations on the two Machines. A chart of these is attached. The column referring to the Reader will be explained, that for the Punch being very similar.

- (a) From the time of the Hollerith one-shot, Source 0 continues to give the current row for a period of 2 Major Cycles. From 6 Major Cycles onwards Source 0 gives zeros, representing the space between rows.
- (b) Not less than 15 and not more than 21 Major Cycles elapse between successive one-shots from the same cards. Thus, computations which take place between rows should not last more than 15 Major Cycles if one-shots are not to be missed.
- (c) When cards are following each other at maximum speed, the first one-shot of one card follows the last one-shot of the previous card after an interval of not less than 80 and not more than 100 Major Cycles. Computations which take place between cards should not last more than 80 Major Cycles if first one-shots on each card are not to be missed.
- (d) Between the 11th one-shot and the beginning of TIL there is a period of at least 11 Major Cycles. From the beginning of TIL to the 12th one-shot there are not less than 2 and not more than 7 Major Cycles. After the 12th one-shot TIL lasts for not less than 18 and not more than 23 Major Cycles. Also, after a P54 on the 12th row, or 9-24X using the 12th row stopper, TIL will certainly be off after 23 Major Cycles. For continuous reading, the first one-shot of the next card is at least 57 Major Cycles after the end of TIL. Source 0 does not revert to the I.D. between cards, provided cards are running continuously. Otherwise, Source 0 reverts to the ID lamps within 13 Major Cycles after the 12th one-shot.
- (e) When the Read has been idle, there is a period of at least 130 Major Cycles between calling for a Read ("12-24") and the arrival of the first one-shot.
- (f) When the reader is cleared by a P34 on the last row of a card it may be recalled immediately by the instruction 12-24 (using the single-shot from the last row if desired).
- (g)

If however the reader is cleared, (either by a P34 or by an instruction 9-24) before the last row of a card it should not be recalled within 2 Major Cycles, or further rows of the same card will be read as if the reader had not been cleared. (This means that single-shots and the TIL signal will not be inhibited).

APPENDIX 1 cont'd

HOLLERITH TIMING

- (h) If the reader is cleared within 40 Major Cycles after the S.S. from the last row of a card, no further cards will be fed.
If the reader is cleared between 40 and 70 Major Cycles after the S.S. from the last row of a card, a further card may be fed, but will not be read and will give a 'MISSED CARD' alarm.
If the reader is cleared more than 70 Major Cycles after the S.S. from the last row of a card, but before the first S.S. of the next card, a further card will be fed, and not read, without giving a 'MISSED CARD' alarm.
- (j) In order to inhibit the S.S. from the subsequent row of a card, reader or punch must be cleared within 13 and 35 Major Cycles respectively of the stop instruction for a given row.
- (k) If a transfer to destination 29 is made more than 20 Major Cycles after the stop instruction for any row other than the last, the word sent will be successfully punched on the next row, unless an instruction 8-24 precedes the time of the next row.
- (l) This is similar to a 'MISSED CARD' on the Reader. If the punch is not cleared within 20 Major Cycles of the last row of a card, another card will pass through the punch. This card will be blank in the DEUCE field but will carry punching in the Job and card numbering sections.
- (m) If a programme finishes with the punch and then uses D29 for the O.S. lights there is a possibility that the first transfer to D29 after the punch is cleared will punch the next row of the card. This will not occur if the punch is cleared within 24 Major Cycles from the last row on which punching is required.

APPENDIX 1 CONT' D
HOLLERITH TIMING TABLE

Event from which time is measured.	Time		Requirement after this time.
	Reader	Punch	
a Stop instruction for any row of a card.	2 6	4	Successful read or punch. Read zeros.
b Stop instruction for any of first 11 rows.	15 21	38 46	Catch S.S. for next row. Miss S.S. for next row.
c Stop instruction for 12th row.	80 100	116 135	Catch 1st S.S. of next card when running continuously. Miss 1st S.S. of next card when running continuously.
d Stop instruction for 11th row. Start of TIL. Stop instruction for 12th row. Read cleared by P34 or 9-24X on any row. End of TIL.	11 2 7 18 23 23 23 13 57	33 2 9 35 50 - 30 - 75	TIL not yet on. Catch 12th S.S. Miss 12th S.S. TIL not yet off. TIL off. TIL off after P34 on 12th row. TIL off after 9-24X on 12th row. I. D. switches can be used. Catch 1st S.S. of next card when running continuously.
e Call READ or PUNCH when machine has stopped (or re-call at least 1 second after end of TIL).	130	180	Catch 1st S.S. of card.
f Read cleared by P34 on any row.	0	-	Recall read by 12-24.

APPENDIX 1 CONT' D

HOLLERITH TIMING TABLE

Event from which time is measured.	Time		Requirement after this time.
	Reader	Punch	
g Clear READ or PUNCH by 9-24.	2	2	Not read or punch same card on re-calling.
h Stop instruction for 12th row of a card.	40	-	Clear READ and not pass extra card through reader.
	70	-	Clear READ and pass extra card through reader without giving 'MISSED CARD' alarm.
j Stop instruction for any row of a card.	13	35	Clear READ or PUNCH by 9-24 and not get next S. S.
k Stop instruction for any row of a card.	-	20	Successful punch on following row.
l S. S. on last row of card.	-	20	Clear PUNCH and not pass extra card through punch.
m Stop instruction for any row of a card.	-	24	Clear PUNCH and not punch on next row of card.

APPENDIX 2

LIST OF SOURCES AND DESTINATIONS FOR THE DEUCE.

<u>Source</u>	<u>Destination</u>	<u>N. I. S.</u>
0 ID or Reader	TS COUNT	DL 8
1 DL 1	DL 1	DL 1
2 DL 2	DL 2	DL 2
3 DL 3	DL 3	DL 3
4 DL 4	DL 4	DL 4
5 DL 5	DL 5	DL 5
6 DL 6	DL 6	DL 6
7 DL 7	DL 7	DL 7
8 DL 8	DL 8	
9 DL 9	DL 9	
10 DL10	DL10	Triggers
11 DL11	DL11	0 - 24 Stim. Mult.
(DL11 Associated with magnetics)		1 - 24 Stim. Divide.
12 DL12	DL12	2 - 24 T. I. L. - Discrim. (Zero)
13 TS13	TS13	3 - 24 Stim. T. C. A.
14 TS14	TS14	4 - 24 Clear. T. C. B.
15 TS15	TS15	5 - 24 Stim. T. C. B.
(a) 16 TS16	(a) TS16	6 - 24 Clear. Alarum
(or DL10 delayed by 1 m. c.)		7 - 24 Stim. Alarum
17 QS17	QS17	8 - 24 Clear. O. P. S.
18 QS18	QS18	9 - 24 Clear. Punch & Read.
19 DS19	DS19	10 - 24 Stim. Punch.
20 DS20	DS20	12 - 24 Stim. Read.
21 DS21	DS21	
(b) 22 DS21 ÷ 2	(b) DS21 ⁺	
23 TS14 ÷ 2	(b) DS21 ⁻	
24 TS14 x 2	Triggers	
25 14 & 15 ⁿ⁾	TS13 ⁺	
26 14 ≠ 15	TS13 ⁻	
27 P1	Discrim. (Sign)	
28 P17	Discrim. (Zero).	
29 P32	O. P. S. or Punch.	
30 Zeros	Magnetics read or write.	
31 Ones.	Magnetics head block shift.	

(a) Modified by T. C. A.

(b) Modified by T. C. B.

n) $14 \vee 15 = (14 \& 15) + (14 \neq 15)$

INDEX

	<u>Page.</u>	<u>Para.</u>
Addition and subtraction.	13, 44	4.3, 6.6
Address Section.	25	5.4
Alarm.	18	4.12
&"and"	34	6.2
Binary Notation.	3	2.1
Binary Point	5, 50, 49	2.2, 6.10
Carry.	44	6.6
Characteristic.	26-29	5.7 - 5.10
Characteristic of Magnetics Instructions.	41	6.4
Constants, useful.	34	6.1
Control-Magnetics Interlock (C.M.I.)	42	6.4
Counting cycles of repetitive Loops.	20	4.16
Delay-Line Storage.	9	3.1
Delay-Lines.	10	3.1.4
Non Instruction Source (see DL 9-12)	58	8.1
Destinations, List of. Appendix 2.		
Destination 0	31	5.13
Destination Triggers.	14	4.7
Discrimination.	17, 32	4.11, 5.14
Division.	6, 49	2.5, 6.10
division by 2.	5, 34, 45	2.3, 6.2, 6.7
Double-length Arithmetic.	44	6.6
Double-length Stores.	10	3.1.2
Double Transfer.	28	5.8
Dummy Instruction.	46	6.8
Fixed numbers.	34, 14	6.1, 4.5
Form of Instruction.	13, 24	4.2, 5.3
Functional Sources.	13, 34, 45	4.4, 6.2, 6.7
Hollerith Timing, Appendix 1.		

	<u>Page.</u>	<u>Para.</u>
Initial Card.	60	8.3
Initial Input.	56	8.1
Initial Instructions.	56	8
Input and Output.	15	4.8
Operation of Punch and Reader.	38	6.3
Input Dynamiciser (I.D.)	15, 39	4.8, 6.3
Initial Input to Magnetic Drum Store.	58	8.2
Instruction, dummy	46	6.8
form of	13, 24	4.2, 5.3
Initial	56	8
Magnetic Drum	11, 41	3.3, 6.4
Modification	30	5.12
Next	17	4.10
Requirements	24	5.2
Stop	22, 38	4.17, 6.3
Word	24	5.3
Introduction.	1	1
Large Programmes.	61, 58	8.5, 8.2
Library Sub-routines.	55	7.3
Link.	53	7.1
Logical Operations.	34	6.2
Long Transfer.	19, 28	4.13, 5.9
Loop.	15	4.9
Counting cycles of repetitive loops.	20	4.16
Magnetic Drum Instructions.	41	6.4
Magnetic Store.	11	3.3
Marker digit.	36	6.2(b)
Minor Cycles.	10	3.2
Modification of Instructions.	30	5.12
Multiplication,	6, 46	2.4, 6.8
signed	47	6.9
by 2	5, 34, 44	2.3, 6.2, 6.6

	<u>Page.</u>	<u>Para.</u>
Next Instruction.	17	4.10
Non Instruction source (see DL 9-12)	58	8.1
≠ "Not equivalent".	34	6.2
Numbering of minor cycles.	10	3.2
"Ones" (Source 31).	34	6.2
Operation of Punch and Reader.	38	6.3
Order of a Sub-Routine.	54	7.2
Output.	15	4.8
Output Staticiser.	38	6.3
P ₁ , P ₁₇ , P ₃₂ . (Source 27-29).	34	6.1
P ₅₄	39	6.3
Parameter.	40	6.3
Programme Examples,		
Successive Squares.	15, 20	4.9, 4.16
Programmed division.	35	6.2
Square root.	37	6.2
Punch out 128 results & read in more data.	40	6.3
Signed Multiplication.	47	6.9
Division using automatic divider.	49	6.10
Quadruple Stores.	10	3.1.3
'Quasi' minor cycle numbers.	31	5.13
Remainder after division.	49	6.10
Replacement Transfer.	13, 9	4.1, 3.1
Representation of Numbers,		
Binary Notation.	3	2.1
Signed Convention.	4	2.2
Static Representation.	7	2.6
Serial Representation.	8	2.7
Round off.	49	6.9 (1)
Running In.	61	8.4

	<u>Page.</u>	<u>Para.</u>
Serial Representation.	8	2.7
Set-up Minor Cycle.	25	5.5
"Shift".	5, 34, 44 45, 49	2.3, 6.2, 6.6, 6.7, 6.9 (k)
Sign Correction.	47	6.9
Signed Convention.	4	2.2
Signed Multiplication.	47	6.9
Single Shots.	22, 38	4.18, 6.3
Single Transfer.	26	5.7
Source 22.	45	6.7
Source 23-31.	34	6.1, 6.2
Source and Destinations, List of, Appendix 2.		
Static Representation.	7	2.6
Stop Instructions.	22, 38	4.17, 6.3
Stop Key.	23	4.19
Storage, Delay-Line.	9	3.1
Delay Lines.	10	3.1.4
Temporary Stores.	10	3.1.1
Double Length Stores.	10	3.1.2
Quadruple Stores.	10	3.1.3
Magnetic.	11, 41	3.3, 6.4
Sub-Routines.	53, 54	7.1, 7.2
Subtraction.	13, 44	4.3, 6.6
Temporary Stores.	10	3.1.1
T.I.L.	39	6.3
Time of Computation.	29	5.11
Timing Section.	26	5.6
Transfer, double.	28	5.8
for a particular minor cycle.	20, 26	4.15, 5.7
long	19, 28	4.13, 5.9
replacement.	13, 9	4.1. 3.1
single.	26, 20	5.7, 4.15
Triggers.	14	4.7
Trigger T.C.A.	42	6.5
T.C.B.	44	6.6
T.S. Count.	10, 24	3.11, 5.1
Wait Number.	26, 30	5.6 - 5.11
"Zeros" (Source 30)	34	6.1

A Short Description of the Automatic Instruction Modifier on "DEUCE"

Up to now, the usual method of modifying a DEUCE instruction has been by means of one of the accumulators. This has the disadvantage that the accumulator used is not available for the calculation in progress at the time of modification. Furthermore, a modifying instruction is necessary and in many cases a modifying word. The A.I.M. facility has been developed to overcome these difficulties.

Operation

When an instruction of the form 17-0 or 18-0 is executed, the first minor cycle of Q.S.17 or Q.S.18 in the transfer period is modified automatically according to the rules given below. Modification does not take place in subsequent minor cycles of transfer. If the transfer of the word in this minor cycle to COUNT actually takes place (i.e. if $c = 0$ or 1 and $w = T$) the modified word is taken as the next instruction.

Operation of "Destination 0" is unchanged in that the normal next instruction is replaced by a word from the quadruple store only if transfer is still taking place when the timing number has counted down. Hence, if it is required that the word modified by the transfer enters control as the next instruction, it is necessary that the last minor cycle of transfer should also be the first, or differ from it by an exact multiple of four minor cycles.

Rules for Specifying Modification

1. For Q.S.17 (instruction 17-0)

(a) If $c = 0$ in the instruction 17-0, then if in this instruction the first two digits are considered as a binary number n , when

$n = 0$, P5 is added to the word in Q.S.17.

$n = 1$, P10 is added to the word in Q.S.17.

$n = 2$, P17 is added to the word in Q.S.17.

$n = 3$, P18 is added to the word in Q.S.17.

Note that if $W \neq T$ (as when 17-0 is a dummy instruction used for automatic counting) the use of the P2 digit imposes a restriction on the Next Instruction Source.

(b) If $c = 1$, then the first two digits have the same effect as above, with the following additional facilities:

if $P3 = 1$, P22 is also added into Q.S.17,

if $P4 = 1$, all digits controlled by P1, P2, P3 are subtracted instead of being added.

2. For Q.S.18 (instruction 18-0)

Operation is as for Q.S.17, but addition and subtraction operations are interchanged.

"Carry"

If a modification to an instruction in Q.S.17 or Q.S.18 includes a P22 there will be no spill from the wait number to the J digits.

There is no carry from the modified minor cycle of the Q.S. to the next minor cycle.

Request Stop

Request Stop should not be used on instructions 17-0 or 18-0. If it is, the appropriate modification will occur repeatedly, once per major cycle, so long as the Request Stop condition holds.

Flow Diagrams

It has been decided that in flow diagrams of published programs and subroutines 17-0 and 18-0 instructions will be written in the same way as 13-0 instructions in the past, with three additions as follows: (nP1) between the location and the 17-0, with the characteristic and modification digits following the 17-0. For instance,

$$2_{28}(6P_1) 17_2 - 0, 1 (P_{17} + P_{22})$$

Subroutines and programs which use the A.I.M. have an M in the alphabetic code number.

Example of Use of Instruction Modifier

A detailed flow diagram is given below of a routine for storing data on the magnetic drum. This illustrates four different types of modification. The coding of the Destination 0 instructions is of interest thus:

- 2_{13} is $(8P_1, 18 - 0, 1, 0, 0)$ (shift)
- 2_2 is $(6P_1, 18 - 0, s, 0, 1)$ (count)
- 2_{27} is $(10P_1, 18 - 0, 1, 0, 0)$ (assemble)
- 2_3 is $(12P_1, 18 - 0, 1, 1, 1)$ (transfer)

It should be noted in this example that the use of Q.S.17 has been avoided in order that it can be used as a normal store in the calculation if desired. All instructions of the routine have been confined to D.L.2 for the sake of compactness.

The parameters P_0 , H_0 and N may be varied by coding 26, 27 and 2_{12} as indicated.

To produce N numbers and store them consecutively on the Magnetic Drum, starting at m.c.0 of track P_0/H_0

