Sakharov, A. D. 1948 *J. Exp. Theor. Phys.* **18**, 631.
Schafroth, R. 1949 *Helv. Phys. Acta*, **22**, (iv), 392.
Schwinger, J. S. 1949 *Phys. Rev.* **76**, 790.
Streib, J. F., Fowler, W. A. & Lauritsen, C. C. 1941 *Phys. Rev.* **59**, 523.
Stuckelberg, E. C. G. 1938 *Helv. Phys. Acta*, **11**, 225.
Thomas, R. 1940 *Phys. Rev.* **58**, 714.
Tomlinson, E. P. 1941 *Phys. Rev.* **60**, 159 (A).
Ward, J. C. 1950 *Phys. Rev.* **78**, 182.

The diagnosis of mistakes in programmes on the EDSAC

By S. Gill
Mathematical Laboratory, University of Cambridge

(*Communicated by D. R. Hartree, F.R.S.—Received* 13 *December* 1950)

This paper describes methods developed at the Cambridge University Mathematical Laboratory for the speedy diagnosis of mistakes in programmes for an automatic high-speed digital computer. The aim of these methods is to avoid undue wastage of machine time, and a principal feature is the provision of several standard routines which may be used in conjunction with faulty programmes to check the operation of the latter. Two of these routines are considered in detail, and the others are briefly described.

1. Introduction

Two kinds of mistakes, or blunders, arise in the use of an automatic digital computing machine: (i) those resulting from faults in the machine itself, and (ii) those arising because the orders or data presented to the machine are not those required to obtain the results sought. This paper is entirely concerned with mistakes of the second kind, and describes methods employed for dealing with such mistakes on the EDSAC at the Cambridge University Mathematical Laboratory. Although it is written with special reference to this machine, much of the subject-matter is in principle more generally applicable.

Programmes are presented to the EDSAC in the form of punched tape, the entries on which are converted into orders and numbers by the machine as the tape is read. This process has been described in a paper by Wheeler (1950), and it will be assumed that the reader is already acquainted with that paper and the various technical terms employed therein. The order code of the machine is repeated here in appendix 1 for convenience.

It is natural at first to dismiss mistakes in programming as an inevitable but temporary evil, due to lack of experience, and to assume that if reasonable care is taken to prevent such mistakes occurring no other remedy is necessary. However, experience with the EDSAC has shown that although a high proportion of mistakes can be removed by preliminary checking, there frequently remain mistakes which could only have been detected in the early stages by prolonged and laborious study.

Some attention has, therefore, been given to the problem of dealing with mistakes after the programme has been tried and found to fail.

The difficulty lies not in detecting the presence of a mistake, but in diagnosing it. In practice its presence is nearly always obvious, for the character of most programmes is such that even a slight error will usually have an extensive effect. If its presence is not immediately apparent, it will be detected by the arithmetical checks which must be incorporated in every calculation.

Every large calculation which is to be performed on the EDSAC is first broken down into its major components, for example, the evaluation of elementary functions such as square roots, each of which can conveniently be carried out by means of a sub-routine. Wherever possible, use is made of sub-routines already available in the library. The use of a library sub-routine not only saves considerable time and effort; it also eliminates the risk of mistakes in that part of the programme. There are at present some eighty sub-routines in the EDSAC library, and the number is steadily increasing. Every new sub-routine which is constructed for the library is tested alone in a simple programme before being applied to a serious calculation, and routines for individual purposes should also be tested before use.

When each programme has been drawn up it is checked, preferably by another person, for obvious mistakes. A tape is then prepared and the punching is checked. In due course the tape is presented to the machine. At some stage it may become apparent that the machine is not behaving as was anticipated. If there is any doubt as to the serviceability of the machine, it can be settled by applying standard test tapes which test every part of the machine thoroughly, and by applying the programme tape repeatedly to see whether consistent results are obtained (although this may not be convenient if the programme fails only after several minutes). When a machine fault arises, the programme is abandoned until the fault has been cured.

The commonest symptom of failure in the EDSAC is stoppage of the machine; this is nearly always the result of the control unit trying to obey some 'order' which is not included in the machine's order code.* When this occurs, the operator notes the position of the 'order' causing stoppage. Such stoppage may even occur before the tape has been fully read, if the punching is wrong. Another common symptom is 'looping', in which the machine is seen to be obeying repeatedly a small cycle of orders, many more times than was intended. Dislocation of the programme may also affect the signals sent to the teleprinter, thus changing the lay-out of the printing or inserting unusual symbols. If the EDSAC is working properly, any of these symptoms indicates that the programme has failed in such a way that the machine is not carrying out the correct orders in the sequence intended by the programmer. Such failures are termed 'order failures'.

It is possible for the machine to carry out the orders as planned by the programmer, and yet to arrive at the wrong numerical results. This may be due to errors in the data or constants, to the capacity of a storage register being exceeded,

* Most machines would not stop in such a situation. The fact that the EDSAC does stop is often valuable, since it rapidly arrests the machine in the event of serious dislocation of the programme.

or to a fault in the mathematical theory. If arithmetical checks are included in the programme they should indicate such a failure, otherwise it will only be detected by inspecting the printed results. These failures are called 'numerical failures'. There is no clear distinction between the two types of failure because the course of the calculation is nearly always affected by the results obtained, but the classification is a useful guide in the early stages of investigating a mistake.

If the programmer is present in person when his programme fails, he may be able to obtain some useful information by noting the timing of events, or by watching the monitors attached to the arithmetical unit and the store, but it is not usually possible to get relevant data in the short time available. Otherwise the facts reported by the operator, together with any printing which might have been produced, are the only evidence available. Before consuming any further machine time, the programme is re-examined in the light of this evidence. Often the mistakes can be diagnosed at this stage. On the other hand, the programmer might still be unable to deduce exactly what must have happened inside the machine during the execution of the programme. In this case it is necessary to return to the machine to obtain more detailed information about the operation of the programme by repeating the calculation, perhaps in a modified form. Ways in which this may be done are discussed in the next section.

2. Methods of checking

All high-speed computers are equipped with a means of causing orders to be obeyed singly, at the press of a button, to enable the progress of a calculation to be followed by eye. However, this facility is not very suitable for the purpose of checking a programme. The most serious disadvantage is the extravagant waste of machine time involved. In the EDSAC, orders are normally obeyed at the rate of about 500 per second. Even if it is possible to follow them by eye at the rate of one every second, it requires 8 min. to work through the equivalent of 1 sec. of normal operating time, and the machine is computing for only 0·2 % of the time. This disadvantage could be reduced slightly by the provision of special facilities such as conditional stop orders (which stop the machine only if a manual control is operated) or slow-motion operation (in which the machine obeys about 10 orders per second).

Any record of the information obtained in push-button operation must be made by hand, while the machine is idle; moreover, the operator must decide immediately which information to record. The task is rendered more difficult by the fact that in the EDSAC, as in most machines, numbers are displayed on the monitors in a coded form and must be interpreted by the operator. In addition to consuming time, this introduces a serious risk of human errors. All these facts make the process an extremely inefficient one for the checking of programmes. Single-order operation is a useful facility for the maintenance engineer, but the programmer can only regard it as a last resort.

It would no doubt be possible to obtain photographically a more or less complete record of the contents of the machine throughout the operation of a programme,

with little reduction in the speed of operation. Such a record would probably enable any mistake to be traced in a comparatively short time. The equipment, with its maintenance requirements and large consumption of film, would, however, be very expensive. It would in fact be a means of high-speed output from the machine; if it were to be used solely for checking programmes its use would hardly be justified.

There exists, however, the possibility of obtaining the required information via the machine's normal output channel (the teleprinter in the case of the EDSAC). This can be done without any alteration whatever to the machine, and, moreover, it utilizes the machine at a speed limited only by the teleprinter. It is not practicable to print as much information as is often desirable, and the type of information to be printed must be chosen according to the circumstances. However, this method is much faster than following the calculation by eye, and it has the great advantage of providing an immediately legible and intelligible record, which can, moreover, be taken away from the machine and studied at leisure.

One way of obtaining such a record is to insert into the machine, after the operation of the original programme has stopped, a second programme causing the teleprinter to print the contents of relevant parts of the store. This method has come to be known as the 'post-mortem' technique. It gives the programmer a static picture of the machine, which may have been stopped in order to apply the check, or may have stopped automatically on encountering a meaningless order. Such a check may be useful when investigating either order failures or numerical failures. Routines which print information from the store are simple in design, and will not be described here. Tapes carrying such routines are kept available near the EDSAC, for use as and when required.

Alternatively, the original programme may be modified in such a way as to cause the printing of suitable information, and the modified form presented to the machine in place of the original. It will be shown that valuable results may be obtained by modifications which are not extensive or difficult to make. This method is more commonly useful than the post-mortem technique, since it can provide information relating to successive stages in the operation of the original programme. A simple example of this procedure is the alteration of a programme to cause the printing of a function at smaller intervals than are ultimately required, the additional values serving purely as checks. This may assist in locating a numerical failure. An order failure may be investigated by inserting, at suitable points in the programme, extra print orders to cause the printing of distinctive symbols. For example, a convenient system is to place such a print order at the head of each sub-routine. This procedure may be difficult to adopt after a programme has been made up, and is more often used as an anticipatory measure, being incorporated in the original programme and later removed (since it is much easier to remove orders from a programme than to insert them). The insertion of the print orders can in many cases be carried out by a special form of assembly routine. (An assembly routine is one which may be used to organize the input of a whole programme, marshalling the various sub-routines in suitable places in the store, and making the necessary adjustments in the orders. The

method of inserting extra print orders was originated by M. V. Wilkes, who also constructed the first assembly routine.)

Methods like the foregoing are too limited in scope to deal with many of the questions that arise. In such cases a considerable modification of, or addition to, the original programme is necessary. It has been found possible to construct general sub-routines which incorporate all these modifications and additions, and which can be applied to any original programme in order to provide certain standard types of information. Such sub-routines are called checking routines.

3. Principles of checking routines

There are at present seven checking routines in the EDSAC library. These have been so designed that they can be applied in as simple a manner as possible, and with the least modification of the programme to be checked; in general, it is merely necessary to change a few symbols at the end of the original programme tape, and to add the checking routine. This simplicity of use, important for any sub-routine, is doubly important for a checking routine because a mistake that is made in the application of a checking routine might lead the programmer into even worse difficulties than those he is trying to resolve.

A checking routine can of course only be used if room is available to accommodate it in the store. This limitation is not so great as may at first appear. Comparatively few programmes require so much storage space that an extra sub-routine cannot be added, and such programmes are always composed of sub-routines which can be tested individually. Moreover, as will be explained later, it is usually possible to put a checking routine in place of the print routine in the original programme.

There are two distinct techniques upon which checking routines can be based. In the simplest of these a single order, known as a blocking order, is inserted into the original programme. The blocking order is an *E* order (see appendix 1), which when obeyed causes control to be transferred to the checking routine. The latter then prints some selected information about the state of the calculation, and returns control to the original programme at the order immediately following the blocking order. Thus if the blocking order is situated in a cycle in the original programme, the checking routine will be called into operation once each time the cycle operates. The technique is suitable for investigating numerical failures, particularly in studying the convergence of an iterative process, but is of little value for checking the order sequence.

The second technique will be referred to as the 'step-by-step' technique. In this, the control unit of the machine never obeys any of the orders of the original programme directly. The machine remains under the control of the checking routine, which is so constructed that the orders of the original programme are examined one by one, and carried out in exactly the same manner and sequence as if they were being obeyed directly. If this were all that the checking routine accomplished, it would be merely a means of carrying out machine operations in slow motion—slowed down, in fact, by a factor of the order of 10. The reason for

adopting this type of operation is that it is now open to the author of the checking routine to insert additional orders into it, causing it to print useful information as it proceeds. This information may be chosen so as to assist in the investigation of either order or numerical failures.

In some checking routines these two techniques are combined: a blocking order is planted, and the original programme allowed to operate directly until it encounters this order. This switches control to the checking routine, which then continues the calculation by the step-by-step process.

The actual forms taken by checking routines must depend to a great extent on the design of the machine, both on its logical plan and on the output organs available. The design of the EDSAC has made possible some very useful forms, which have played a large part in facilitating the use of the machine. These are described in the following sections.

4. Example of blocking order technique

The simplest checking routine in the EDSAC library employs the blocking order technique alone. It is known as *C* 1, and its purpose is to print the number standing in a given location in the store, whenever the blocking order is obeyed.

The use of a blocking order in a programme for the EDSAC is attended by some difficulties which must now be discussed. First, in the order code of the EDSAC there is no order which causes a transfer of control unconditionally; there are only the *E* and *G* orders (see appendix 1), which transfer control if the number in the accumulator is non-negative and negative respectively. If, therefore, the sign of the number in the accumulator were unknown, it would be necessary to use two blocking orders, an *E* order and a *G* order, to ensure transfer. If, however, we restrict the blocking order to a point at which the number in the accumulator is known to be non-negative, only an *E* order is required. A further advantage is gained by restricting the number in the accumulator to zero, for the following reason. The accumulator is required for use by the checking routine; if it is not empty its contents must be stored somewhere until the checking routine has finished its work. The storing, and subsequent recovery, of the entire contents of the accumulator is a rather lengthy operation for the EDSAC, and is avoided by the stipulation that the accumulator shall contain zero.

The blocking order may be inserted either in place of one of the orders of the original programme, or between two such orders. In the latter case some rearrangement of the original programme would be necessary to make room for the blocking order; this might involve alterations to many other orders, a tedious and error-provoking task. The former method is therefore adopted. In order that the operation of the original programme shall be unaffected, it is necessary to reproduce in another part of the store the order which is replaced by the blocking order, so that the original order may be carried out after the checking routine has operated, and before control is returned to the original programme. *C* 1 itself carries out these preparatory operations, before directing control to the entry point of the original programme; the programmer merely has to specify where the blocking order is to be placed.

In addition to the condition that the accumulator shall be empty when the blocking order is obeyed, the programmer must also observe that the order which is to be replaced by the blocking order must not be an order which is altered during the operation of the programme, or which is taken into the arithmetical unit. If the second condition is not satisfied, then when checking, the blocking order will be either destroyed or used in the arithmetical unit in place of the original order, and the programme will fail. In practice these restrictions do not seriously limit the application of $C1$. There are frequent points throughout any programme at which the accumulator is known to be empty, and comparatively few orders are altered or used arithmetically. However, there does exist a danger of $C1$ being misapplied and yielding erroneous results.

The routine occupies a total of 44 storage locations. The check numbers are printed to 11 decimal places, preceded by a sign, in a single column at the left-hand side of the sheet. Numbers printed by the original programme normally appear to the right-hand side of the previous check number. Preset parameters* are used to specify the location whose content is printed, the position of the blocking order, and the position of entry into the original programme.

The actual orders of $C1$ are given in appendix 2.

5. Example of step-by-step technique

Routine $C11$ forms the basis of all the step-by-step routines in the EDSAC library, and will therefore be treated in some detail. The purpose of this routine is to print the function letters† of the orders in the original programme in the sequence in which they would be obeyed. This information is valuable when investigating an order failure.

$C11$ occupies 32 storage locations. Its speed of operation is determined almost entirely by the teleprinter, and it examines about five orders of the original programme per second. Difficulties arise when print orders in the original programme are encountered (see appendix 3, note 5).

The letters are normally printed in line across the page; a new line of printing is started whenever a transfer of control occurs. Thus each line of printing corresponds to a sequence of orders in consecutive locations. This fact assists the programmer in identifying the orders which have been obeyed, and in practice it is very rare for any ambiguity to arise. An earlier routine indicated the sequence in which orders were obeyed by printing the actual locations to which and from which each transfer of control occurred, thus removing all ambiguity. It was found, however, that the information provided by $C11$ was more useful to the programmer, for when a programme is drawn up its orders are usually numbered only with respect to the beginning of the routine to which they belong, and their absolute positions can only be deduced by performing an addition, whereas their function letters remain invariant. Moreover, $C11$ is a much simpler routine as it involves no binary-to-decimal conversion.

* See Wheeler (1950, §6).

† For example, *A* for add, *S* for subtract, etc. See appendix 1.

The manner in which *C* 11 operates is best explained by means of a flow diagram (see figure 1). The functions of the components of this diagram will now be described.

I. *Select next order.* The 'current order', that is, the order in the original programme which is to be examined, is taken into the checking routine via the accumulator. The order in I which extracts the current order from the store is called the 'select order'.

II. *Print function letter.* The function digits of the current order are read out to the teleprinter from the checking routine.

V. *Carry out current order.* The current order is obeyed, being read by the control unit from the checking routine, not from the original programme.

VIII. *Advance address in select order.* Since orders are normally obeyed in serial sequence, it is necessary to arrange that the next order to be examined shall be extracted from the following location in the store. This requires that the address specified in the select order shall be increased by 1.

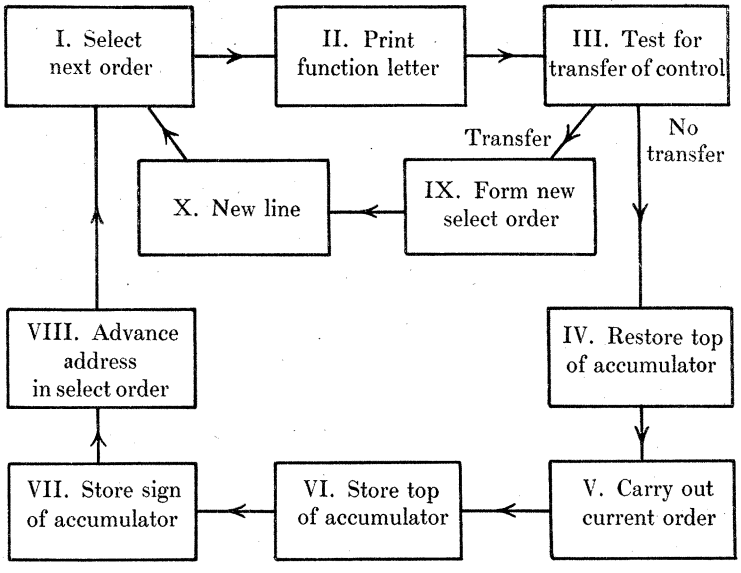

FIGURE 1.

III. *Test for transfer of control.* If the current order is such as to cause a transfer of control, it must not be obeyed, because that would direct control to the original programme and the checking routine would no longer operate. Control must therefore be switched to IX and X.

IX. *Form new select order.* The new select order, after a transfer of control, must specify the address to which control is transferred, that is, the address specified in the last current order.

X. *New line.* Suitable instructions are sent to the teleprinter.

VI. *Store top of accumulator.* Most of the above operations require the use of the accumulator, which must therefore be cleared after each current order has been

obeyed. In fact it is only necessary to clear the 17 most significant binary digits. These must be stored elsewhere, while the above operations are carried out.

IV. *Restore top of accumulator.* Before each current order is obeyed, the accumulator must be restored to the exact state which it occupied immediately after obeying the previous current order.

VII. *Store sign of accumulator.* In III, the criterion for transfer of control involves the sign of the number in the accumulator, that is, the number which was in the accumulator after obeying the last current order. This sign could be obtained from the 17 digits stored by VI, but it is convenient to record the sign separately in a suitable coded form, ready for use in III.

Appendix 3 contains the orders of $C11$, with an example of its application.

6. More elaborate checking routines

$C11$ suffers from two disadvantages. First, it may fail when it encounters printing orders in the original programme. Secondly, if information is only required concerning a small part of the original programme, time is wasted in checking through other parts which are known to be correct.

The first disadvantage could be overcome by causing the routine to react in a special way to O and F orders (see appendix 1), besides transfers of control, in the original programme. However, it is rarely necessary to check print routines, which can usually be taken from the library. The best course is therefore to remove the printing orders from the original programme. Usually all the printing in a programme is carried out within a single closed sub-routine, which can be put out of action without affecting the progress of the calculation. To do this it is merely necessary to replace the print routine in the store by a short sub-routine, usually known as a 'dummy print routine', which does nothing but return control to the master routine (otherwise called the main programme).

A dummy print routine consists of very few orders, so that most of the storage space originally occupied by the print routine is now vacant. This space can be used to store the checking routine. In fact, the dummy print routine and the checking routine can be provided together on the same length of tape. This procedure is adopted in $C12$, which like $C11$ prints the function letters of orders as they are obeyed. $C12$ has one other distinguishing feature: it employs a blocking order to provide a delayed start of checking. The original programme operates at full speed until the blocking order is obeyed for the nth time. The blocking order is then replaced by the order originally in that position, and checking commences, function letters being printed. The number n and the position of the blocking order must be specified by preset parameters. The whole routine occupies 40 locations.

There are two other checking routines, $C7$ and $C9$, which check the order sequence by printing function letters. $C7$ is so constructed that, although it examines every order of the original programme in turn, it only prints the function letter in certain cases, namely, if the order is drawn from certain regions of the store, specified in advance. The user can therefore specify the regions containing those orders whose operation he wishes to check. Other parts of the programme

are worked through by the checking routine at a higher speed, since no printing is involved.

When using $C7$, the store may be divided into four regions, orders in two of which have their function letters printed. The regions are specified by preset parameters. The speed of operation is about 5 orders per second when printing function letters, and about 30 orders per second when not printing. Print routines in the original programme must be arranged to lie in regions from which function letters are not printed. Characters printed by such routines appear as figures. A new line of printing is commenced at each transfer of control; a clear line is left where orders have been obeyed silently, unless such orders themselves cause printing to appear on this line. The routine $C7$ occupies 61 locations.

$C9$ is used when the orders to be examined are all within the master routine. It provides for a cessation of checking whenever a closed sub-routine is encountered. By convention, closed sub-routines in the EDSAC are always called into operation in a certain distinctive way;* $C9$ recognizes such an event and takes appropriate action. This action results in the sub-routine being obeyed at full speed. The fact that the sub-routine has been obeyed is indicated by printing the letter Q; checking then recommences. The process is arranged so that sub-routines with one programme parameter operate correctly, but $C9$ must not be applied to programmes containing sub-routines which have more than one programme parameter. A blocking order is used to provide a delayed start of checking.

$C8$ and $C10$ employ the step-by-step technique to provide numerical checks. The principal difficulty in devising a numerical checking routine lies in arranging for the routine to provide sufficient useful information, without using a large number of preset parameters to specify what information is required. $C1$ is very limited in scope; it only checks the content of one storage register at one point in the programme. To extend this method many preset parameters would be required, to specify the positions of several blocking orders, or several registers whose contents were required to be known.

Instead, $C8$ and $C10$ provide systematic checks, throughout the calculation, of quantities that are most likely to be useful to a programmer. They print the number that is transferred from the accumulator by each T order (see appendix 1). Except during input and output, every number in the store is first formed in the accumulator, so that the numerical progress of a calculation can be followed if the behaviour of the accumulator is known. Furthermore, the only order in the EDSAC code which clears the accumulator is the T order, so that normally any error in the content of the accumulator persists until the next T order is obeyed. This system of checking is therefore the most generally useful, and since the accumulator is known to be empty after a T order, its content does not have to be stored during printing.

The checking routine must examine every order of the original programme to detect T orders, and therefore the step-by-step technique must be employed. $C8$ and $C10$ are similar, except that $C10$ ceases checking during the operation of closed sub-routines, and $C8$ only prints when it encounters T orders referring to

* See Wheeler (1950, §9).

addresses less than a specified number. Usually orders are placed in the upper part of the store and numbers at the lower end, and therefore the latter provision enables the user to cut out checks on those parts of the calculation which are concerned purely with such things as counting and the alteration of orders. Both routines incorporate a dummy print routine, and use a blocking order to provide a delayed start. As an example, some details of $C10$ are given below.

Title. Numerical check with delayed start, dummy print routine, and suppression of check during closed sub-routines.

Description. May be applied to a programme to print $C(Acc)$ before obeying T orders. May not be applied to programmes containing sub-routines with more than one programme parameter.

Notes. (1) When inserted into the store, $C10$ is split into two parts. One occupies 37 locations, the first of which must bear an even address. The other occupies 51 locations; it includes the dummy print routine, and may therefore be put in place of a print routine in the original programme. (The reason for splitting $C10$ is that the whole routine cannot be accommodated in the space normally occupied by a print routine.)

(2) A new line of printing is commenced at each transfer of control. A blank line is left when a sub-routine is encountered.

(3) Checking commences the first time the blocking order is encountered.

(4) The number of digits to be printed, the location of the second part of $C10$, and the location of the blocking order, are specified by preset parameters.

The author is indebted to D. J. Wheeler, who contributed many useful suggestions and was in particular responsible for the development of routines $C8$, $C9$, $C10$, and the 'post-mortem' test, and also to A. Glennie and K. N. Dodd, who assisted in the preparation of some of the routines. The author also wishes to acknowledge the award of a Maintenance Allowance by the Department of Scientific and Industrial Research.

Appendix 1

The EDSAC order code

$A\ n$	Add the number in storage location n into the accumulator.
$S\ n$	Subtract the number in storage location n from the accumulator.
$H\ n$	Copy the number in storage location n into the multiplier register.
$V\ n$	Multiply the number in storage location n by the number in the multiplier register and add the product into the accumulator.
$N\ n$	Multiply the number in storage location n by the number in the multiplier register and subtract the product from the accumulator.
$T\ n$	Transfer the contents of the accumulator to storage location n and clear the accumulator.
$U\ n$	Transfer the contents of the accumulator to storage location n and do not clear the accumulator.

C n	Collate the number in storage location n with the number in the multiplier register and add the result into the accumulator; that is, add a '1' into the accumulator in digital positions where both numbers have a '1', and a '0' in other digital positions.
*R D**	Shift the number in the accumulator 1 place to the right; that is, multiply by 2^{-1}.
R $2^{p-2}F$†	Shift the number in the accumulator p places to the right; that is, multiply by 2^{-p} $(2 \leqslant p \leqslant 12)$.
R F	Shift the number in the accumulator 15 places to the right; that is, multiply by 2^{-15}.
*L D**	Shift the number in the accumulator 1 place to the left; that is, multiply by 2^{1}.
L $2^{p-2}F$†	Shift the number in the accumulator p places to the left; that is, multiply by 2^{p} $(2 \leqslant p \leqslant 12)$.
L F	Shift the number in the accumulator 13 places to the left; that is, multiply by 2^{13}.
E n F	If the number in the accumulator is greater than or equal to zero, transfer control to n, i.e. execute next the order which stands in storage location n; otherwise proceed serially.
G n F	If the number in the accumulator is less than zero, transfer control to n; otherwise proceed serially.
I n	Read the next row of holes on the input tape and place the resulting integer, multiplied by 2^{-16}, in storage location n.
O n	Print the character now set up on the teleprinter and set up on the teleprinter the character represented by the five most significant digits in storage location n.
F n	Place the five digits which represent the character next to be printed by the teleprinter in the five most significant places in storage location n, clearing the remainder of this location.
*X**	Ineffective; machine proceeds to next order (previously used as a round-off order).
*Y**	Round-off the number in the accumulator to 34 binary digits; that is, add 2^{-35} into the accumulator.
*Z**	Stop the machine.

Appendix 2

Library sub-routine C 1

Preset parameters:

code letter	parameter	significance
H	*P h D*	*hD* is the location whose content is to be printed
N	*P n F*	position of blocking order
M	*P m F*	point of entry into original programme

The first order of *C* 1 must be placed in an even-numbered location.

* The addresses in these orders need not be zero.

† The addresses in these orders *may* be $k.2^{p-2}$, where k is odd, provided that the addresses do not exceed 2047.

location with respect to first order	order		notes
	T	***Z***	control combination
0	(*O*	41 *θ*)	(i) fig. shift (ii) work space
1	(*A*	*N*)	(i) *C*(*n*) to 41
2	(*T*	41 *θ*)	
3	(*A*	6 *θ*)	(ii) store *C*(*R*)
4	(*T*	*N*)	(i) plant blocking order (ii) count digits
5	*E*	*M*	enter original programme
6	*E*	13 *θ*	blocking order
7	*J*	*F*	= 10/16
8	Δ	*F*	line feed = − 1/2
9	*ϕ*	*F*	space
10	*θ*	*F*	carriage return
11	*Z*	*F*	+
12	*P*	11 *F*	number of digits
N → 13	*O*	10 *θ*	carriage return
14	*N*	8 *θ*	
15	*N*	8 *θ*	store *C*(*R*)
16	*T*	2 *πθ*	
17	*O*	8 *θ*	line feed
18	*H*	7 *θ*	10/16 to multiplier register
19	*A*	*H*	*C*(*hD*) to accumulator
20	*E*	25 *θ*	test sign
21	*O*	19 *θ*	print '−'
22	*T*	*πθ*	change sign negative
23	*S*	*πθ*	
24	*E*	26 *θ*	
20 → 25	*O*	11 *θ*	print '+' positive
24 → 26	*T*	*πθ*	
27	*S*	12 *θ*	set digit count
38 → 28	*T*	4 *θ*	
29	*V*	*πθ*	
30	*U*	*θ*	
31	*O*	*θ*	
32	*F*	*θ*	
33	*S*	*θ*	digit print cycle
34	*L*	4 *F*	
35	*T*	*πθ*	
36	*A*	4 *θ*	
37	*A*	2 *F*	
38	*G*	28 *θ*	
39	*O*	9 *θ*	space
40	*H*	2 *πθ*	restore *C*(*R*)
41	(*π*	*F*)	order from *n*
42	*E*	1 *N*	return to *n*+1
43	*G*	1 *N*	

Notes

(1) The following notation is employed:

Transfers of control are indicated on the left of the page by means of arrows opposite the order to which the transfer occurs.

Unconditional transfers: a horizontal line is shown underneath every E or G order which is intended to produce a transfer of control each time it is encountered.

Pseudo-orders, namely, entries which are written as orders but which are intended never to be obeyed as orders, are indicated by vertical lines immediately to their left.

Variable orders, i.e. orders and pseudo-orders which are to be changed during the course of the calculation, are shown in brackets.

The content of storage location n is written $C(n)$, and that of the multiplier register $C(R)$. The long location h is now referred to as hD, not h' as in the paper by Wheeler (1950).

(2) The multiplier register is used by the checking routine in performing the binary-to-decimal conversion for printing. While this is being carried out, the number originally held in the multiplier register must be stored elsewhere in the machine: positions 2 and 3 of $C1$ are used.

(3) All working space, including space for the storage of $C(R)$, must be within $C1$ itself, to avoid interfering with numbers required by the original programme.

(4) When the original order in location n is copied by $C1$ it is copied into a position at the end of $C1$, so that it will be obeyed at the appropriate time. After it has been obeyed, control must be returned to the original programme by means of an E order and a G order, since the state of the accumulator is unknown.

Appendix 3

Library sub-routine C 11

location with respect to first order		order		notes
		G	K	control combination
0	‖	$(P$	$F)$	storage for sign of $C(A)$ } (note 2)
1	‖	$(P$	$F)$	storage for $C(A)$ } (note 2)
2	‖	θ	F	
3	‖	Δ	F	$= -1/2$ } constants
4	‖	A	F	$= -1/4$ } constants
5	‖	Q	F	$= +1/16$ } constants
18 → 6		A	$4\,\theta$	form new select order (note 4) IX
7		O	$2\,\theta$	teleprinter carriage return } X
8		O	$3\,\theta$	teleprinter line feed } X
31 → 9		U	$11\,\theta$	} place select order } VIII, IX
10		S	$11\,\theta$	} place select order } VIII, IX
11		$(Z$	$F)$	select order } I
12		U	$22\,\theta$	place current order } I
Enter → 13		O	$22\,\theta$	print function letter (note 6) II
14		S	θ	$3/16 \leqslant x < 4/16$ } (note 4) III
15		A	$4\,\theta$	$-1/16 \leqslant x < 0$ } (note 4) III
16		E	$19\,\theta$	} (note 4) III
17		A	$5\,\theta$	$0 \leqslant x < 1/16$ } (note 4) III
18		E	$6\,\theta$	} (note 4) III
16 → 19		U	θ	} clear top 17 digits of accumulator } IV
20		S	θ	} clear top 17 digits of accumulator } IV
21		A	$1\,\theta$	restore $C(A)$ } IV
22		$(K$	$3000\,F)$	current order (note 6) V
23		U	$1\,\theta$	store $C(A)$ VI
24		E	$26\,\theta$	test sign of $C(A)$ } VII
25		A	$3\,\theta$	add $-1/2$ if $C(A) < 0$ } VII
24 → 26		S	$1\,\theta$	subtract $C(A)$ } VII
27		U	θ	} store sign of $C(A)$ (note 2) } VII
28		S	θ	} store sign of $C(A)$ (note 2) } VII
29		A	$11\,\theta$	select order to accumulator } VIII
30		A	$2\,F$	add unity to address } VIII
31		G	$9\,\theta$	} VIII
		E	$13\,Z$	

Followed on tape by

E	m	F

punched by user

When this has been read, control is switched to order 13 of this routine, with $E\,m\,F$ in the accumulator.

Notes

(1) The notation described in appendix 2, note 1, is used.

(2) As in $C\,1$, all working space must lie within the routine itself. This includes a location for the storage of the 17 digits, referred to as $C(A)$, which would be at the top (most significant end) of the accumulator if the original programme were operating directly, and also a location for recording separately the sign of $C(A)$. The latter is coded thus: 0 if $C(A) \geqslant 0$; $-1/2$ if $C(A) < 0$.

(3) The whole of the accumulator, except the top 17 digits, must remain undisturbed from the obeying of one current order to that of the next. Hence

a T order cannot be used in $C11$, as this would clear the whole accumulator. When it is necessary to clear the top 17 digits (as at orders 10, 20 and 28) an S order following a U order is used, so that these digits are subtracted from themselves.

(4) Operation III to test for a transfer of control is carried out as follows. The numerical value of the function letter E is 3/16, so that with the address part added, an E order is numerically equal to a number in the range $3/16 \leqslant x < 4/16$. Similarly a G order lies in the range $-5/16 \leqslant x < -4/16$. Consideration of the effect of E and G orders (appendix 1) shows that by adding 1/2 to any order if, and only if, the accumulator contains a negative number, then the necessary and sufficient condition for a transfer of control can be stated thus: that the modified order shall lie in the range $3/16 \leqslant x < 4/16$.

The current order is placed in the accumulator by the select order, and remains there until order 14 is obeyed; this modifies the current order as above. The remainder of the test merely determines whether x lies in the necessary range. Opposite orders 14, 15 and 17 are shown the conditions satisfied by the number in the accumulator at each stage if transfer is to occur. It will be seen that control is switched to order 6 under these conditions.

On arriving at order 6, the function digits of the order have been reduced to zero, but the address part has not been changed. To form the new select order it is therefore only necessary to insert the function letter A.

(5) There are two ways in which $C11$ may fail if printing ordered by the original programme is encountered. If an output order in the original programme shifts the teleprinter to print figures, all subsequent function letters will appear as their figure equivalents in the teleprinter code. Also if the original programme contains an F order (appendix 1) this order will place in the store, when $C11$ is used, not the symbol last set up on the teleprinter by the original programme, but the last symbol set up on the teleprinter, which is F (set up by order 13 of $C11$).

(6) The first time that order 13 is obeyed it prints not a function letter, but a symbol that is initially placed in position 22 to set the teleprinter on letter-shift. Then C11 proceeds as though it had just selected the order $E\,m\,F$: it sets the teleprinter to begin a new line, and puts $A\,m\,F$ in the select order position, so that the first current order is drawn from position m.

As an illustration, the effect of applying $C11$ to the initial orders will be shown. This is not a practical example, since the initial orders are fixed and their correctness established. However, they afford an excellent example of the possible complexity of a programme, and have already been described in detail by Wheeler (1950); appendix III to that paper gives the present initial orders in full, and may be referred to in following this example.

The initial orders will be examined in the process of reading a tape which is punched as follows:

T 60 K
G K
A 8 $\pi\theta$
T 310 D
etc.

In order to carry out this example, $C11$ would first have to be placed in the store and then directed to examine the initial orders, starting at a suitable point, say order 34. If $C11$ is placed in locations from 100 onwards, the complete tape would consist of

T 100 K

sub-routine $C11$

E 34 F

followed by the symbols to be read during the test.

Below is shown the result. Only the letters on the left are actually produced by the machine, the other columns being given for the guidance of the reader, but it will be seen that the letters themselves are in fact sufficient to show the course of the programme. The time required, including tape input, to obtain the results shown is less than a minute.

symbols printed by teleprinter	positions of corresponding orders	symbols read from tape by I order
IARTE	34 to 38	*T*
TIASG	8 to 12	6
ARVLTIASG	4 to 12	0
ARVLTIASGLSESATAE	4 to 20	*K*
AE	30 to 31	
TE	25 to 26	
IARTE	34 to 38	*G*
TIASGLSESATAE	8 to 20	*K*
AEATIARTE	30 to 38	*A*
TIASG	8 to 12	8
ARVLTIASGLSESATAE	4 to 20	π
AE	27 to 28	
TIASGLSE	8 to 15	θ
ATAAATAATE	17 to 26	
IARTE	34 to 38	*T*
TIASG	8 to 12	3
ARVLTIASG	4 to 12	1
ARVLTIASG	4 to 12	0
ARVLTIASGLSE	4 to 15	*D*
ATAAATAATE	17 to 26	

etc.

References

Wheeler, D. J. 1950 *Proc. Roy. Soc.* A, **202**, 573.

Wilkes, M. V. 1949 *J. Sci. Instrum.* **26**, 217.

Wilkes, M. V. 1950 *Appl. Sci. Res.* **B1**, 429.

Wilkes, M. V. & Renwick, W. 1949 *J. Sci. Instrum.* **26**, 385.

Wilkes, M. V. & Renwick, W. 1950 *M.T.A.C.* **4**, 61.

Wilkes, M. V., Wheeler, D. J. & Gill, S. 1951 *The preparation of programs for an electronic digital computer, with special reference to the EDSAC and the use of a library of subroutines.* Cambridge, Mass: Addison-Wesley Press Inc.