PROGRAMME DESIGN FOR THE C.S.I.R.O. MARK I COMPUTER

II. PROGRAMME TECHNIQUES

By T. PEARCEY* and G. W. HILL*

[*Manuscript received December* 5, 1952]

Summary

The general structure of typical programmes is considered in relation to the structure of the computer, in particular to its facilities which tend to render programmes invariant with regard to their position in the store. Full-scale programmes are constructed by piecing together by a master programme a suitable selection of items from a library of " standard routines ". The library contains " sub-routines " which are completely self-contained and require no additional information for their operation and are invariant with respect to their position in the store, and " routines " which are not so invariant and frequently require additional data to be provided during their entry into the store. The entry of special data for routines and the simplification of the construction of the master programme are facilitated by a special routine used whilst the entire programme is being entered into the store.

I. INTRODUCTION

Part I (Pearcey and Hill 1953) dealt largely with the preliminaries to the design of full-scale programmes. It described the computer, its address system, and the system of notation used for designing and recording programmes on paper prior to the stage of translating them onto a medium for machine input. The written programme is recorded onto punched tape with the aid of a specially modified teleprinter keyboard. The written programme permits direct translation onto tape without further writing or special effort.

Part II includes a general discussion of the nature of programmes, the main features being illustrated with regard to the use of the present machine. Basic groups of commands or " sub-routines " are selected from a library of such groups and only the connections between them, together with any special operations, are included in the part of the programme specially designed for the current problem. Commands may be modified upon entering the computer in a number of ways which involve use of a special routine which is entered initially ; this routine is described in detail.

II. THE ELEMENTS OF A PROGRAMME

The ability of a computer to change the commands held in the store in an organized manner permits a great saving in the effort needed for programme design and also in the number of commands required to perform a required calculation. In practice only a small proportion of the commands of a

* Division of Radiophysics, C.S.I.R.O., University Grounds, Sydney.

programme is actually devoted to making the calculation; the greater proportion organizes the course of calculation and frequently involves continued adjustment of some commands.

A typical example of a programme having the normal structure is given in Table 1. The construction of the table may be compared with those for a similar example given for the EDSAC (Wilkes, Wheeler, and Gill 1951).

Suppose it is required to find the sum of the content of all storage locations from 100 to 149 inclusive, to store the result in register A, and then stop. The notation described in Part I is used in Table 1, and the programme is entered at the first command in the list placed at location m.

TABLE 1

TYPICAL PROGRAMME STRUCTURE

Location of Command	Command		Action
m	$(A)\overset{-}{\rightarrow}A$		Clears A, $(A)'=0$
$m+1$	$(149)\overset{+}{\rightarrow}A$		Addition starts with (149)
$m+2$	$(m+1)\rightarrow C$		Reduces $(m+1)$ to $(149-r)\overset{+}{\rightarrow}A$; $r=0, 1, 2, \ldots$
$m+3$	$p_{11}\overset{-}{\rightarrow}C$		
$m+4$	$(C)\rightarrow m+1$		
$m+5$	$100\overset{-}{\rightarrow}C$		Reduces (C) by $100\,p_{11}$
$m+6$	$s(C)\overset{c}{\rightarrow}S$		Tests sign of (C)
$m+7$	$m+1\rightarrow S$	$[m+1]$	If <50 addition, i.e. $r<50$
$m+8$	$p_{20}\rightarrow T$		$r=50$; additions complete; stop

The programme possesses the following features which illustrate the normal structure of most programmes:

(1) The programme is partly repeated, being re-traversed from the command in $m+1$ to that in $m+7$.

(2) During each traverse of the repeated commands only one command, that in $m+1$, is directly devoted to the calculation which is the objective of the programme.

(3) The command performing the desired arithmetical function is changed during each traverse by the computer so that in the following traverse it refers to a different, although adjacent, storage location.

(4) Three of the six repeated commands adjust the command in location $m+1$, i.e. those in $m+2$, $m+3$, and $m+4$.

(5) A count is made of the number of repetitions, and is tested during each traverse to decide whether or not the full number of additions has been made.

(6) If the summation is incomplete, control is shifted " backwards " to $m+1$, otherwise a " stop " command, as in $m+8$, is reached.

(7) One command is not repeated, i.e. that at m. This is devoted to the preparatory function of clearing register A.

In a complete programme the command in location $m+8$ would be replaced by one which would cause a shift of control to continue with further calculation.

From Table 1 it is seen that a programme consists of :

(*a*) A preparatory group of commands.

(*b*) A group performing the functions which are the objective of the programme.

(*c*) A group of commands which adjusts the programme for successive immediate repetitions.

(*d*) A group keeping a tally of the number of repetitions.

(*e*) A test on the tally to decide if the repetition shall cease and another part of the programme be adopted.

In some cases these sections of a simple programme may not be distinct and may even be absent. Thus, in an iterative process, as distinct from a repetitive process illustrated by Table 1, the groups (*c*) and (*d*) may be omitted, whilst (*e*) is present in the form of a test at each stage on the result from the group (*b*).

In the example given, (*a*) corresponds to the command in m, (*b*) to that in $m+1$, whilst $m+2$, $m+3$, and $m+4$ correspond to (*c*). The command which, at each stage, appears in $m+1$ is held in register C and is used as the " counter " for the repetitions.

Normally, programmes will contain a number of groups of commands similar in structure to that of Table 1. In fact, the main controlling programme will frequently be of this typical type in which its component parts are themselves smaller groups possessing similar structure and so on.

III. Standard Routines

Construction of each programme *ab initio* upon its individual merits would become very tedious. To save much of the labour of programme design as much use as possible is made of information accumulated and made available from previous programmes in the form of " standard routines " or groups of commands.

It frequently occurs that computations involve somewhat complex operations of standard types, such as evaluation of particular functions like exponential or circular functions, interpolation of a function table, numerical quadrature and so on. Each such operation requires its special programme which, if suitably designed, may be used as a standard routine for use in other calculations which require these types of operations.

Operations such as division, taking square roots, and even reading and recording fresh information and converting data to and from decimal scale may be so standardized. Such standard routines have the effect of providing the computer with additional and more complicated functions. Thus, for instance, an inversion routine may be considered as having the logical meaning, "substitute into register A the reciprocal of its present content ".

Two factors are necessary for successful use of standard routines in construction of programmes. Firstly, standard routines should be useful over as wide a

field as possible, that is, routines should not be too specialized otherwise the list of standard routines eventually recorded and stored becomes overwhelming and confusing. Secondly, there must be a simple means of incorporating them into programmes.

Standard routines frequently, although not always, possess a structure like that of Table 1 and may frequently involve more than one loop. For greatest usefulness it should be possible to place a routine in any part of the store without complication to the organization of the remainder of the programme. It should preferably be invariant with regard to its place in the store. Occasions will arise, particularly in use of routines for specialized use, when a routine must be restricted to a special position in the store and will not be invariant.

Further, a routine may require different data when used in a different context and may do so even when used more than once by the same programme. Such data are commonly called " parameters " and are supplied to the programme during its insertion into the machine, as will be described later.

Quantities used by a routine which differ every time the routine is entered are stored in standard positions, e.g. registers A, B, C, H, and D, just before the entry into the routine. These quantities are called " variables ".

IV. Position Invariance

A special class of standard routines, which includes a large proportion of the most useful and frequently used routines, can be made invariant with regard to store position by use of the normal functions of the machine. The principal device used for this is the command of the type

$$n \overset{\pm}{\rightharpoonup} S,$$

which causes a relative shift of control by $n+1$ locations. In the example of Table 1 the command $m+1 \rightharpoonup S$ could be replaced by $-7 \overset{\pm}{\rightharpoonup} S$ (where -7 is considered modulo 1024). Within any routine the various values of n defining internal relative shifts of control are independent of the position of the routine in the store.

Numbers such as working constants, e.g. e^{-1}, $\pi/4$, etc., could be stored at the end of the routine by which they are used, that is in locations depending on the position of the routine in the store. This location reference can be avoided by storing such constants in two parts within two successive commands. Thus the sequence of commands

$$\begin{aligned} n &\rightharpoonup A, \\ m &\rightharpoonup H_u, \\ (H_l) &\overset{\pm}{\rightharpoonup} A, \end{aligned}$$

places $(n2^{10}+m)2^{-19}$ in register A. Quantities may be similarly assembled into register C, but not in D, since the digit space occupied by n includes also the space specifying the location of D and may be incompatible.

By means of the further device of initially storing variables in various standard registers most of the simpler routines can be made invariant with regard

to store position. Output routines and routines for division, square root, and simple transcendental functions are members of this class and are called " sub-routines ". Those which require the use of parameters are called simply " routines ".

Not all functions which would help in rendering routines invariant have been included as functions of the machine. Another would be a function which would allow a " transfer to the store location now indicated by the sequence register " to take place. By this means direct reference to positions within the routine could be made, so that a command constructed with the aid of a position reference value drawn from the sequence register could be adopted for immediate use. Such a procedure would involve a considerable number of additional commands.

V. Parameters

The use of standard routines may be greatly increased by use of parameters which may be given different values in different programmes. Sub-routines do not possess such parameters. Standard routines which possess internal references and cannot *conveniently* be made invariant may be provided with the store position of a particular command contained within it, usually its leading command, as one of its parameters.

Consider the example listed in Table 1. The values depending upon the particular context are (*a*) 149, the first location summed, (*b*) 100, the last location summed, and (*c*) m, the location of the leading command of the routine. There are three parameters, only one of which is determined by the way in which the routine is entered into the computer.

At this stage it is essential to distinguish between commands as they appear in the store after entry and the way in which the input tape is punched. Commands $m+1$, $m+2$, $m+4$, $m+5$, and $m+7$ may be entered in a special way dependent upon an extra character punched so as to distinguish them from the remaining commands. This character may precede or follow the command affected and cause the value of the required parameter to be added to the command actually punched. This may be done by use of a specially designed programme which must be placed in the store before the entry into the store of the programme. If α, β, γ, and δ denote special characters, associated each with a parameter and with the commands of Table 1 into which the parameters are introduced, the programme designed for punching on the tape becomes that shown in Table 2.

The quantities entered into the second and sixth commands by β and γ must previously have been entered into the store. The character α will have the effect of adding the location of the leading command of the routine to the sub-address immediately following. This location number will have been stored upon reading the character δ at the head of the routine.

The details of the procedure whereby these operations are performed is described later.

The use of parameters, although somewhat complicating the entry of routines into the store, greatly increases their flexibility and renders them invariant with respect to position in the store.

VI. Variables

Variables are normally held in the main arithmetical registers prior to entry by the programme into the routine or sub-routine which uses them. Thus the value x, of which the square root is required, is placed in register A before entry into a " square root routine ". Such a routine replaces x by $\sqrt{x}$.

Consider in particular the adaptation of the routine of Table 1 to the use of variables in place of parameters. Suppose, for instance, 149 be stored in A and 100 in H. Additional commands placed ahead of the command $(A)\overline{\rightarrow}A$ place the appropriate values for later use by the routine.

Table 2

INCORPORATION OF PARAMETERS

Example from Table 1

Relative Location of Command		Commands as Punched on Tape			Action of Additional Symbol
0	δ ;	$(A)\overline{\rightarrow}A$			δ records location m of first command
1	β ;	$[(0)\overset{\pm}{\rightarrow}A]$			Command enters as $(149)\overset{\pm}{\rightarrow}A$
2	α ;	$(1)\rightarrow C$			Command enters as $(m+1)\rightarrow C$
3		$p_{11}\overline{\rightarrow}C$			
4	α ;	$(C)\rightarrow 1$			
5	γ ;	$0\overline{\rightarrow}C$			Command enters as $100\overline{\rightarrow}C$
6		$s(C)\overset{c}{\rightarrow}S$			
7	α ;	$1\overset{+}{\rightarrow}S$		[1]	Command enters as $m+1\rightarrow S$
8		$p_{20}\rightarrow T$			

The programme so arranged is listed in Table 3 where the only parameter is the location of the leading command. The command $(149)\overset{\pm}{\rightarrow}A$ is formed from the constant or pseudo-command stored at the foot. The latter is never itself adopted as a command. The command thus formed is substituted in place of the $[p_{20}\rightarrow T]$. This is a " stop " command which, if by some mistake not changed, when adopted stops the operation and acts as a warning. The procedure follows as before except that H stores the other variable, 100, and the command $(H_u)\ \overline{\rightarrow}C$ replaces $100\overline{\rightarrow}C$.

Routines may be designed in which the roles of parameters and variables are interchangeable. By entering the routines at one or other of two possible points one or other type of data may be adopted. Such routines are used in cases where data may be fixed for one particular programme and then entered as parameters, but may vary during the course of a different calculation and must be set as variables by the programme.

If many variables are required by a routine, beyond the capacity of the arithmetical registers, standard locations in the store may be used, e.g. locations 0, 1, 2, etc. or $m+1$, $m+2$, etc.

VII. The Connection of Routines into Programmes

A complete programme consists primarily of a collection of standard routines selected from a library, together with any routines specially constructed. This part is called the " detail programme ". New routines, whenever possible, are constructed in such a manner that they may later be incorporated into the library. The various parts of the detail programme are related to each other by a specially constructed " master programme ". The main function of the master programme is to control the course of the calculation and the passage to and from the various routines.

Table 3

USE OF PROGRAMME VARIABLES

Example from Table 2

Relative Location of Command		Command as Punched on Tape			Action
0	δ, α	$(10)\overset{+}{\rightarrow}A$			Stores head location m and command enters as $(m+10)\rightarrow A$ and makes $(A)'=(149)\rightarrow A$
1	α	$c(A)\rightarrow 2$			Enters as $c(A)\rightarrow m+2$, causes $(m+2)'=(149)\rightarrow A$, leaving $A'=0$
2		$[p_{20}\rightarrow T]$			
3	α	$(2)\rightarrow C$			} Has effect of increasing r by unity
4		$p_{11}\overset{+}{\rightarrow}C$			
5	α	$(C)\rightarrow 2$			
6		$(H_u)\overset{-}{\rightarrow}C$			Subtracts $100p_{11}$ from C
7		$s(C)\overset{c}{\rightarrow}S$			
8	α	$2\rightarrow S$		[2]	Return if $r\leqslant 49$
9		$p_{20}\rightarrow T$			Stop if $r=50$
10		$<(0)\overset{+}{\rightarrow}A>$			Constant or pseudo-command

It is usual for most routines to be separated from the master programme which calls them into use. This is essential for routines which are frequently used or which are used more than once in different contexts by one particular programme.

Some routines, frequently special routines, may be incorporated directly into the master programme, that is, the master programme leads into and out of such routines without special shifts of control. Following EDSAC terminology, such routines are called " open routines ".

" Closed routines ", into which class the great majority of library routines fall, are entered and left in a special manner involving shifts of control. A convention is adopted, namely, that a closed routine, called into use by one command, finally transfers control back to the command immediately following in the store that from which it was entered. This is not, however, an inflexible rule.

The D register provides storage space for the " linking " data which organize the return of control to the master programme. Normally the lower-numbered locations in D are reserved as working space used by routines, e.g., D_0, D_1, D_2, and D_3 say. The higher-numbered locations, D_{11}–D_{15} say, are left free to store linking data.

Thus a routine entered at location t by the command at location m, transfers control back to $m+1$ from, say, location $t+r$. The set of commands organizing these shifts of control is shown in Table 4.

TABLE 4
STANDARD LINK PROCEDURE

Location	Programme Transferring to Routine	Location	Routine
$m-1$	$(S)\rightarrow D_{15}$	$\longrightarrow t$	$p_{11}\overset{+}{\rightarrow}D_{15}$
m	$t\rightarrow S$	—	—
$m+1$	next command $\leftarrow$	$t+r$	$(D_{15})\rightarrow S$

The command in $m-1$, known as the " plant " command, places mp_{11}, the content of the sequence register at the instant of transfer, into D_{15}; the next command, the " cue ", transfers control to the routine. The command in t, the " adjustment " command, changes the value of (D_{15}) to $\overline{m+1}\ p_{11}$ whilst the return to $m+1$ is controlled by the " link ", which is the command in $t+r$ by which (D_{15}) is substituted into the sequence register. As written, the adjustment is the leading command, but this may be placed anywhere convenient within the routine before the link.

Four commands only are needed of which only one, the cue, requires a special datum, the location to which to shift. All others are standard. An advantage of this device for linking to and fro is that no arithmetical register normally used for storing variables is required to perform the shifting so that all variables may be set before the cue.

The scheme adopted is very flexible, for control may easily be returned to one of a set of alternative locations. For instance, before $m-1$ is reached D_{15} may receive a special datum, np_{11} say, which may depend upon the previous course of the calculation. The command in $m-1$ may be replaced by $(S)\overset{+}{\rightarrow}D_{15}$ so that control is returned by the routine to $m+n+1$, where n may take one of a number of values. Further, routines may sometimes be entered at one of a number of positions and return of control may depend upon the position of entry or upon a variable. In such cases the adjustment command is replaced by a command which sets the required value in D_{15}.

Some routines may call upon other or auxiliary routines during their operation. The frequent use of standard routines as auxiliaries to other standard routines helps to reduce the size of a library of routines, since the auxiliary routines may also be used individually. The method of linking in such cases

may be extended to the auxiliary routines without special reference by the master programme. Those routines which do not use auxiliary routines use D_{15} for linking. Such routines are said to be of the " first " or " lowest order ". Those which use only one additional routine at any instant are of the " second order ". Second order routines may use more than one auxiliary routine but always return control back to the main routine before entering the next auxiliary routine.

Linking into a second order routine is made from the master programme by the use of D_{14}, and such a routine places successive links into D_{15} upon entering each auxiliary routine. " Third order " routines are linked via D_{13} to the master programme and use auxiliary routines linked via D_{14}, which in turn use further routines of first order linked via D_{15}. This procedure may be extended and once the master programme transfers to the highest order routine no reference to the master routine is made during the operation of the routine or its succession

TABLE 5

CONTRACTED LINK PROCEDURE

Location of Command	Command	Action
m	$(S) \rightarrow D_{15}$	Plants $(m+1)p_{11}$ into D_{15}
$m+1$	$p \rightarrow S$	Cue to 1st routine
$m+2$	$q \rightarrow S$	,, ,, 2nd ,,
$m+3$	$r \rightarrow S$	,, ,, 3rd ,,

of auxiliary routines. A routine of lower order may be placed in a standard position relative to the routine using it, e.g. at the end of the latter routine, or the linking data may be supplied to it as a parameter.

Some programmes require the use of a sequence of routines and sub-routines in immediate succession. For a group of routines of the same order, as frequently occurs with first order routines, the linking data may be planted once only for the group by the master routine before entry into the first of the group. Thus, for instance, if a group of first order routines is entered at positions p, q, and r the master programme will contain the set of commands shown in Table 5.

On return of control to $m+2$ the link datum is already stored in D_{15} and is changed to $m+3$ by the adjustment in the routine entered at q and so is already set, upon return to $m+3$, for immediate entry into the third routine at r. This allows of considerable simplification of a master programme.

VIII. Functions Required During Input of Data

During the input of routines some of the data entered must be adjusted to make the routine suited to the programme of which it is a part. The special functions required can be stated.

If a standard routine is available in a medium suitable for machine use, e.g. punched tape, it should be incorporated into the programme as it stands. It

should thus have in it such characters punched as will allow parameters to be introduced. These characters are called " control designations ". Parameters must be supplied to the machine prior to the entry of the routine which requires them and may be stored in groups of standard locations in the store, from which they will be extracted one by one according to the designations detected.

Normally commands will enter sequential store locations. At the commencement of a group of parameters the normal manner of assembly and storage of data must be broken off and the parameters entered into their appropriate positions, after which the normal process must be restored, fresh data being set into the store in locations immediately following the last routine inserted or at any chosen place.

It must also be possible to store the number of the location into which the data being assembled would normally be inserted, and to treat this number as a parameter.

At some stage in the input of a programme it may be necessary to break off the input procedure and to start all or part of the programme so far inserted. This is best done by adopting the last datum assembled from tape as a command.

IX. The Primary and Control Routines

The assembly and storage of data not associated with special designations are performed under the control of the " primary routine ". This is a set of 20 commands which is inserted whenever the computer is started from the " all cleared " condition via a group of stepping switches, where they are permanently wired. With the primary routine stored in locations 0–19 inclusive and the sequence register cleared to zero, data are transferred from punched tape in groups of 10 binary digits per row. A hole is registered as unity and absence of a hole as zero.

As described in Part I each row of the tape possesses 12 positions, 10 binary digits and two controlling positions called X and Y. An X punch is transferred to the input register as a unit p_{19} and a Y as a unit p_{20}.

The action of the primary routine, which is listed in Appendix I, is to add together successive rows which are not X-punched, the partial sums being placed at each stage in the p_{11}–p_{20} group of register A. Upon reading a row possessing an X punch the accompanying 10 digits are added into the p_1–p_{10} group of digits in register A from where it is placed into store.

Before placing a datum into store the next row of holes is " read " and if it possesses no Y punch (A) is then transferred to the store location indicated by the sub-address of the command in location 6. This command is commonly called the " transfer command ". Upon transfer to store the sub-address of (6) is increased by unity by commands 7–9. If, however, a Y is encountered, control is shifted from 3 to 20, the first place beyond the primary routine.

Simple programmes and those which involve only sub-routines can frequently be constructed without the use of parameters, so that the assembly and storage is straightforward. The first command entered is placed in location 20 and this may be the first command of the programme or one causing a shift of control,

and a Y punch, placed on the last row read, will cause control to be shifted to 20 and thus start the programme.

Where parameters are required the first routine entered is the " control routine ". This is assembled in the standard manner and possesses no Y punches and is an open routine. It is entered whenever a Y-punched row is encountered by the primary routine. It is also listed in Appendix I and consists of only 12 commands which are always stored in the locations 20–31 inclusive. Each Y-punched row is accompanied by a group of holes which determines the action taken with regard to the particular parameter already entered or to follow.

Control will be transferred to 20 on the detection of a Y punch. The first command of the control routine clears D_0 of any p_{20} digit it may contain. The next command, 21, stores the sub-address of the transfer command in H. At this stage the p_{20} due to the Y has been removed by the command in 1 and the accompanying digits are *counted* into the sequence register by the command in 22. One of the following actions will take place :

(*a*) If no units are associated with the Y the control designation is zero ; the next command *adds* (31) to (*A*) and then shifts control to 9. Then (*A*) is placed in location 6, which therefore becomes a new transfer command. The sign of D_0 is now zero and control is passed to zero via 11 and so to read the next row of holes. Suppose (*A*), at the stage of command 13, is np_{11}. Then the addition of (31) causes (*A*) to become $c(A)\rightarrow n$. Eventually the transfer command becomes $c(A)\rightarrow n$ and the next data will enter locations n, $n+1$, etc. The number np_{11} may be entered into A from a row preceding the Y-punched row. (*A*) must be zero before reading n as it will be if the previous command had been normally compiled and stored.

(*b*) If the Y is accompanied by a single unit, then 23 is omitted and 24 causes the transfer command to be replaced by (*A*), a datum which may have been compiled from preceding tape rows. It should be noted that the transfer command may be changed by this designation from one causing a storage operation to any other operation, in particular a shift of control. Thus, for instance, the tape command $75\rightarrow S$ followed by this designation will cause (6) to become $75\rightarrow S$. This command will be performed if the next two rows are X-punched. These X punches are needed because the primary routine always reads one column " beyond " the data assembled in order to detect the possible presence of any control designation which may follow and affect the datum just assembled. The same applies to case (*a*).

(*c*) When the Y punch is associated with two or more units, one of a different set of operations occurs. In the case of two units control is transferred to 25. This command adds the pseudo-command $<(0)\rightarrow T>$ to (*A*) followed by the addition of $(31)\pm A$; (30) and (23) added together form the command $(H_u)\rightarrow 31$. If previous rows have formed np_{11} in A and control moves to 25, then 27 fills 28 with the command $(H_u)\rightarrow 31+n$, which is immediately obeyed, and control passes to 29, and thence to read the next row of holes. If n is replaced by a pseudo-command 28 will imply a different operation, e.g. a command demanding a transfer of control.

The example quoted, in which (28) becomes $(H_u) \rightarrow 31+n$, causes the sub-address of the transfer command to be placed into location $31+n$. This corresponds to a store location n places beyond the pseudo-command in 31. Locations immediately following 31 are those normally used for storing references of this type which are usually the leading locations of routines.

(*d*) A *Y* punch associated with three units causes control to pass to 26, omitting 25. This causes (28) to become $(31+n) \stackrel{+}{\rightarrow} A$, where (A) is np_{11} at the detection of this control designation. Register A is cleared by command 27 and thus the next row is read with (A) equal to $(31+n)$. By this means the data stored in a group following a designation of the type (*a*) or (*c*) may be selected at will from the store. Thus a command following punched as $25 \rightarrow S$, if preceded by np_{11} and a designation of this type, will be accumulated in A as $\overline{25+m} \rightarrow S$, where mp_{11} is set by a designation of type (*c*).

Designations of this kind may be used also to refer to parameters set after a designation of type (*a*).

(*e*) When a *Y*-punched row contains four units control is shifted to 27. Then (A), whatever at this stage it may be, is obeyed as a command since (A) is transferred to 28 by 27. A command may thus be compiled in A by the primary routine and may be obeyed directly upon detection of a designation of this type. In particular this function is useful for producing immediate transfers of control without returning to read further rows, and for rearranging sets of parameters, a function sometimes necessary when the same parameters are needed by more than one routine.

(*f*) A *Y* and five units transfers control immediately to 28. This causes the sequence to stop if (28) has been unchanged. The command last placed in 28 by a designation of the types (*c*), (*d*), or (*e*) may be repeated by subsequent designations of this type.

(*g*) A *Y* and six units transfers control back to zero and recommences reading with no further action.

(*h*) A *Y* and seven units transfers control to 30 and therefore stops the sequence.

(*i*) A *Y* and eight units causes control to pass to 31 and thus replaces the first command by whatever is in A.

The cases of nine and ten units would shift control outside the control routine. However, if extra functions are required a special routine may be entered following 31. Allowance for this must be made in placing parameters. Designations of types (*g*), (*h*), and (*i*) are rarely used.

The control routine discussed here is not unique and may be replaced by any other control routine specially constructed. It is usual to adopt that illustrated since designations in standard routines are adopted for this control routine.

X. Use of the Control Routine

The system of control designations resembles the " control combinations " used for programming the EDSAC* and are used in a similar manner. Certain

* The two systems were developed independently.

designations are used more frequently than others and designations may follow each other on the tape without the interpolation of commands between them.

Programmes are written in the form in which they are to be punched onto tape, with special symbols representing control designations. Designations are written in the column to the left if preceding, or in the column to the right if following, a datum to which they refer. Sequences of control designations are written in lines as convenient. They are punched in the order they are read from the programme sheet.

The symbols used are as follows :

(*a*) mT : The symbol m may be replaced by a group of symbols such as a pseudo-command if necessary, but usually will be an integer less than 1024. Thus mT implies changing (6) to $c(A)\rightarrow m$.

(*b*) R : This symbol is placed after a command or pseudo-command and implies the replacement of (6) by the datum preceding it.

(*c*) nS : This designation causes the sub-address of the transfer command to be placed in $31+n$ where $n \geqslant 1$.

(*d*) nA : This causes the addition into register A of $(31+n)$. It usually precedes the datum which it eventually affects. The number n may be replaced by a pseudo-command which may, when added to the command $(31)\overset{+}{\rightarrow}A$, form a different command to be obeyed in location 28.

(*e*) D : This is a single symbol following the datum to which it refers and implies that datum to be obeyed as a command immediately.

(*f*) This is denoted by U and follows a datum or control designation. It causes repetition of the last nS, nA, or D function performed.

In all these designations the component m or n is punched onto tape as a binary number without an X punch if to be used as a sub-address, or with an X if as a pseudo-command, and immediately precedes the designation proper which is represented by the letter symbol. The letter symbol corresponds to the Y punch and the set of control digits corresponding to the operation required. Thus T is punched as a single hole in the Y position only, R as a Y and one unit and so on. The same keyboard is used for punching both commands and designations.

As an example of the manner in which control designations are used consider the routine listed in Table 2. Clearly the parameters $149p_{11}$ and $100p_{11}$ must precede the routine on the tape. Location 32 is normally reserved for the storage of the head location of the routine entering and 33 onward reserved for other parameters and data such as the locations of heads of other routines. The programme corresponding to Table 2, written in a form suitable for transfer to tape, is shown in Table 6.

Here $m-1$ is the location last filled before the entry of the routine. The first four rows on the tape store m and change (6) so that the two parameters following are entered into locations 33 and 34. These have been written in numerical form and not as pseudo-commands. Then follow two control designations which place mp_{11} in A and change (6) back to $c(A)\rightarrow m$ so that the next command is placed immediately following the last routine, that is, in m.

The routine itself follows on the tape and is now in a standard form. The initial $1S$, which is always placed at the head of each self-referring routine, again places mp_{11} in 32. This value is added into all commands immediately following $1A$ designations. The parameters in 33 and 34 are entered into the commands immediately following the $2A$ and $3A$ designations respectively.

At the foot of the routine the stop command shown in Table 2 is replaced by adjustment and link commands in the standard manner.

TABLE 6

USE OF CONTROL DESIGNATIONS

Location Filled	Tape Entry				Action
		$1S$, $33T$			Store m in 32, changes (6) to $c(A)\rightarrow 33$
33		$149p_{11}$			} Parameters
34		$100p_{11}$			
		$1A$, $0T$			Places mp_{11} in A and changes (6) back to $c(A)\rightarrow m$
$m+0$	$1S$	$(A)\overline{\rightarrow}A$			Routine is entered here by programme, mp_{11} sent to 32 again
$m+1$	$2A$	$(0)\overset{+}{\rightarrow}A$			Adds (33) to command and enters as $(149)\overset{+}{\rightarrow}A$
$m+2$	$1A$	$(1)\rightarrow C$			Adds (32) to command and enters as $(m+1)\rightarrow C$
$m+3$		$p_{11}\overline{\rightarrow}C$			
$m+4$	$1A$	$(C)\rightarrow 1$			Adds (32) to command and enters as $(C)\rightarrow m+1$
$m+5$	$3A$	$0\overline{\rightarrow}C$			Adds (34) to command and enters as $100\overline{\rightarrow}C$
$m+6$		$s(C)\overset{c}{\rightarrow}S$			
$m+7$	$1A$	$1\rightarrow S$		[1]	Enters as $m+1\rightarrow S$ but may be replaced by $-7\overset{+}{\rightarrow}S$
$m+8$		$p_{11}\overset{+}{\rightarrow}D_{15}$			
$m+9$		$(D_{15})\rightarrow S$			

XI. THE ORGANIZATION OF COMPLETE PROGRAMMES

Once the main scheme for carrying out a calculation has been decided upon, the necessary sub-routines and routines may be chosen from the library and special routines may be constructed. Any of these may require parameters and control designations.

The manner and order in which the routines are to be entered into the store are then fixed and a list made of the routines, together with the necessary control designations and parameters. This list describes the detail programme. The master programme must then be designed. This will be largely devoted to the

organization of the course of the programme from one routine to another. The master routine will frequently require a number of control designations associated with it, particularly those giving the positions of the routines of the detail programme. It must be entered into the store after the detail programme and will follow the detail programme on the programme tape.

The construction of a complete programme is illustrated by the example which follows. Suppose we have a tape containing a sequence of decimally punched values of a variable x $(0 \leqslant x < 1)$ and that the corresponding values of $\cos^{-1}[\exp(-\sin \pi x/2)]$ are required to be printed for each entry on the tape. The following routines, which are briefly described, would be required from the library for the detail programme:

(1) Decimal Input A: Sub-routine; closed. Length; 22 commands. Entry; position 0. Link; D_{15}.

This reads a single positive or negative decimally punched fraction from the tape and converts it to binary form leaving the result in register A.

(2) Exponential A: Sub-routine; closed. Length; 13 commands. Entry; position 0. Link; D_{15}.

This replaces (A) by its negative exponential, i.e. $(A)' = \exp[-(A)]$.

(3) Sincos A: Routine; closed; 1st order. Length; 33 commands and coefficients. Entry; position 0 for sine, 1 for cosine. Link; D_{15}. Parameters; nil.

This replaces (A) by $\sin \pi(A)/2$ or $\cos \pi(A)/2$ according as the entry is made via the first or leading, or the second command respectively. The routine is headed by the control designation $1S$.

(4) Arcos A: Routine; closed; 2nd order. Length; 19 commands. Entry; position 0. Link; D_{14}. Parameters; location of leading command of sincos A. This routine replaces (A) by its inverse cosine, i.e. $(A)' = 2/\pi.\cos^{-1}(A)$, $-1 \leqslant (A)' < +1$. It uses sincos A which is placed immediately ahead of this routine into which it is linked via D_{15}.

(5) Print A: Sub-routine; closed. Length; 33 commands. Entry; position 0. Link; D_{15}.

This sub-routine converts positive or negative numbers in A considered as integers (i.e. multiples of 2^{19}) into a series of binary-decimal tetrads which it prints as decimal characters, together with a sign, onto paper. No layout commands are included.

The detail programme is now arranged as in Table 7.

The first control designations cause the first routines to enter from location 38 onwards thus leaving 32–37 free for parameters which are locations of the various leading commands of the routines. These are used by the master programme which is now constructed as illustrated in Table 8.

The first two commands of the master programme enter unaffected and demand a line feed and carriage return of the output printer for which the printer code numbers happen to be 29 and 30. Command No. 3 is changed by the $2A$ designation and shifts control to the decimal input sub-routine. In order to

retain the number x entered, it is stored in D_2. Control is moved to No. 4, and x is printed out by the shift to the print routine following. The value of x in A is destroyed by the printing operation and is replaced by command No. 7. Two routines are entered by 9 and 10 without further plant commands. The second order link is planted in D_{14} by 11 and the inverse cosine routine is entered by 12. Commands 13–15 form the product of the computed value in A with the content of the 19th position which converts the result from the grade scale to degree scale. The result is printed and control is returned to the head of the master programme, where the next value is read from the tape.

TABLE 7

DETAIL PROGRAMME

Routines and Control Designations	Action upon Entry
Control routine	Enters positions 20–31
$38T$, $2S$	Changes (6) to $c(A)\rightarrow 38$, stores $38p_{11}$ in 33
Decimal input A	Enters positions 38–59
$3S$	Stores $60p_{11}$ in 34
Exponential A	Enters positions 60–72
$4S$	Stores $73p_{11}$ in 35
Inverse cosine A	Enters positions 73–91
Sincos A	Enters positions 92–124; it is headed by a $1S$ designation
$5S$	Stores $125p_{11}$ in 36
Print A	Enters positions 125–157

It will be noticed that the print routine is used twice, the value from the tape and the corresponding computed value of the function being printed alongside each other. Spacing control is made by the first two commands.

Further, the sincos A routine although used twice is called by the master programme only once, the arcos A routine calling it the second time. This second use is facilitated by the use of a $1A$ function in the arcos routine which has its position previously placed in location 32 by its initial $1S$ function.

The last entry of the master programme, which is written as it is to be punched onto tape, calls the computer to transfer control to the head of the master programme, and to start the computation, and to commence the reading of the tape containing the values of x in decimal punch code.

This master programme consists of only 20 commands and controls the passage to and from the detail programme of about 120 commands. Most of the operations of the master programme are associated with the sequence register. This is as it should be with a programme intended to supply the highest order of control to a calculation.

XII. The Construction of Programme and Data Tapes

The computer is capable of accepting data only from punched paper tape and of punching results onto similar tape. Punched cards are used as an auxiliary medium. The two media are associated through the use of special editing equipment used for coding programmes.

Table 8

MASTER PROGRAMME

Location Entered	Serial No.	Commands and Control Designations				Action
158	0	$6S$	$29 \rightarrow 0_t$			Stores 158 in 37; causes line feed
159	1		$30 \rightarrow 0_t$			Causes carriage return
160	2		$(S) \rightarrow D_{15}$			Link-plant command
161	3	$2A$	$0 \rightarrow S$		[(33)]	Enters as $38 \rightarrow S$; shifts to "dec. input"
162	4		$(A) \rightarrow D_2$			Stores x in D_2
163	5		$(S) \rightarrow D_{15}$			Link-plant command
164	6	$5A$	$0 \rightarrow S$		[(36)]	Enters as $125 \rightarrow S$; shifts to print values of x
165	7		$(D_2) \rightarrow A$			Restores x to A
166	8		$(S) \rightarrow D_{15}$			Link-plant command
167	9	$1A$	$2 \rightarrow S$		[(34)]	Enters as $94 \rightarrow S$; shifts to compute sin $\pi(A)/2$
168	10	$3A$	$0 \rightarrow S$		[(34)]	Enters as $60 \rightarrow S$; shifts to compute exp $[-(A)]$
169	11		$(S) \rightarrow D_{14}$			Link-plant command (2nd order)
170	12	$4A$	$0 \rightarrow S$		[(38)]	Enters as $73 \rightarrow S$; shifts to compute $2/\pi$. $\cos^{-1}(A)$
171	13	$6A$	$18 \rightarrow C$			Places multiplicand from 177 in C
172	14		$c(A) \overset{\times}{\rightarrow} B$			Forms product to reduce (A) to (degrees of arc) $\times 10^3 \times 2^{-19}$ and rounds off
173	15		$(R) \overset{+}{\rightarrow} A$			
174	16		$(S) \rightarrow D_{15}$			Link-plant command
175	17	$5A$	$0 \rightarrow S$		[(36)]	Enters as $125 \rightarrow S$; shifts to print result as an integer
176	18	$6A$	$0 \rightarrow S$		[(37)]	Enters as $158 \rightarrow S$; returns control to head of master programmes
177	19		$<0{\cdot}1716614>$			Constant used to convert result to degrees of arc, equal to $9 \times 10^{-3} \times 2^{-19}$
		$6A$	$0 \rightarrow S$	D		Enters as $158 \rightarrow S$ and obeyed immediately

The library of routines is stored on punched cards of the usual Hollerith or I.B.M. type. Each routine or sub-routine occupies one or more cards, and commands and binary data are punched in a columnar fashion in groups of 10 digits, the address group being X-punched. Control designations are included. Punched columns on each card are terminated by a single column punched with an X and Y only. This punch, when read by a card reader, causes the card to be ejected and the next in the deck fed into the reading position. The card reader consists of an electric duplicating key punch in which the punches have been replaced by reading brushes.

The advantage of the use of cards is that they are easily stored in a small space and that many copies of each routine or sub-routine may be stored and replenished when required. Damage sustained in use by such copies is of little importance.

After the design of the detail and master programmes has been completed the necessary routine cards are selected from the library and placed into the card reader in the order in which they are to be transferred to tape. Blank tape is fed into the tape punch. The routines are copied from the cards directly onto the tape column for row except for the final XY of each card. Data such as control designations and parameters are supplied via the keyboard between successive routines. Finally, the master programme is punched via the keyboard together with open routines which it may incorporate. The latter are reproduced from cards if standard. Before a tape is used a second tape is prepared in the same way and the two are compared by a special tape comparator. This ceases comparing when any corresponding non-identically punched rows are detected. Punching errors, but not programming errors, can then be eliminated. Tapes may also be duplicated by connecting a tape reader to the editing tape punch in place of the card reader. This also allows correct portions of tapes to be copied and errors to be corrected from the keyboard.

Cards are also used to provide decimal data. This provision is due to the great distances between populated areas in Australia. A distant user may punch decimal data onto cards in the conventional manner. These data are transferred for input directly onto tape without translation of code.

Results may be punched onto tape in the same code as cards are decimally punched. After the output tape has been edited the results are transferred to cards through a tape reader and a special card punch. The cards may then be sorted, reproduced, and listed in any desired manner. This procedure is useful for the production of printed tables. The form of the page may be chosen and the cards may then be listed in a suitable manner for reproduction as a volume. The card punch is used also for recording routines onto cards for the library from the keyboard.

XIII. Checking

The computer possesses no independent or automatic checking devices. All responsibility for adequate checking of results is left to the programmer. There are, of course, two main aims in checking, firstly to ensure that the programmes are correctly designed and, secondly, to ensure the absence of

instrumental faults. Special devices are applicable to one or other of these aims, although in some circumstances the same device may be used for the detection of either type of error.

Much source of error in the construction of programmes is diminished by the use of library routines whose properties are known precisely. However, although the correct routines may be chosen for a particular calculation, errors in the assembly of data and in the master programme may exist and editing checks must always be applied before placing the tape into the machine. Coding errors may most easily arise from false keying on the keyboard. These are detected by comparing a pair of independently punched tapes. Similarly, before placing a new routine in the library two sets of cards are punched independently and are compared before reproduction in bulk.

Each routine or sub-routine is tested on the machine before being placed into the library. In such tests registers N_1 and N_2 are useful for setting data to be used by the routine being tested. The manual control made possible by these registers allows of the rapid detection of logical faults in both routines under test and in full-scale programmes.

Although it is possible to go through each new programme being tested by performing each command one by one (depression of a " single operation " key on the control panel allows one command to be obeyed) and inspecting the register content, such a procedure is very tedious. Programmes are usually broken into distinct parts during testing, each part being separated by a stop command. These commands are placed in positions at which the content of the registers may be of particular importance.

The special stop commands may, when a programme is known to be correct, be replaced by " record " commands which transfer the content of one register to another to be used for inspection, e.g. a spare position in register D or the store. The contents of registers A, B, C, and H are shown on cathode-ray oscillograph faces as also is the entire content of D and any group of 16 consecutive locations of the store. The state of a calculation may be seen by a glance at the tube faces.

Special routines are sometimes used in tracing logical faults in programmes. These are less highly developed than are checking routines used in the EDSAC mainly because their use has not been found essential in this machine. This is not because such faults have not occurred but that the more elementary devices of viewing results, using stop and record commands together with the use of the hand-set registers N_1 and N_2, have been adequate.

Of the few checking routines used, one will cause the calculation to proceed through the programme tested and will print out the locations of the programme at which a shift of control occurs. The route taken through the programme may then be traced. Another routine will print the content of registers A, B, and C over a chosen range of the programme.

Programmes, once known to be correct in the store, are page printed by use of a special routine. A record of the entire programme in its final form is obtained in case repunching should be necessary. The programme prints the addresses in a letter code as close to the written source and destination symbols as possible.

The sub-address is printed in the scale of 32 for simplicity in repunching. A programme of similar type can be used to punch a tape of the final programme, as it lies in the store. This is particularly useful in cases of programmes required in identical form at a later date.

To check that no electrical faults develop during running of a calculation, programmes are designed to include whatever mathematical checks may be applied to the calculation, and special calculations whose results are known are repeated at intervals. In calculations which have short programmes applied to large amounts of data, selected samples of the data are submitted to calculation before proceeding to the main bulk of the work. The results for the selected data must be reproduced in the main calculation.

Before daily use all units of the machine are subjected to standard tests. The early tests are manually controlled. Later stages, designed to test the arithmetical registers and function gates, involve specially designed programmes.

XIV. Use of the Low Speed Store

The low speed store comprises four independent groups of 1024 locations with an access time of 10 msec. It is used for storing incidental results, matrices of coefficients, tables of functions, approximations to solutions to partial differential equations, etc.

One of these stores is used to hold one or more programmes which may be in use. Thus if a programme is to be used for a number of days it is convenient to hold it in one of these stores, preferably in locations corresponding to those in which it lies in the high speed store when in operation. When required the programme is transferred to the high speed store. This takes only a few seconds.

Although the access time of this store is 10 msec, use of it during calculation does not seriously reduce the rate of operation. This is due partly to its infrequent use even if referred to for much arithmetical work. The speed of operation is controlled primarily by the access time to commands and not to calculation data, to which relatively few commands refer. This is so since the major part of a programme is concerned with its own control, a function which does not require the low speed store.

Programmes which are too extensive to be held entirely within the high speed store may be adopted for action from the low speed store as required. This may be done in two ways, either by taking the commands one by one from the low speed store and obeying them with the aid of a special routine or by transferring portions of the programme from the low speed store to the high speed store, as and when required for immediate use. The latter method is that usually adopted, for greater operating speed can be attained if each routine is used for relatively long periods before being over-written by the next routine. The proportion of time spent in transferring routines is then small.

Programmes may be placed in the low speed store directly from the programme tape. When they are transferred to the high speed store for use the space normally occupied by the primary and control routines and the parameters may be occupied by the programme placed in the low speed store from location zero onward. This saves up to 40 locations in the high speed store.

When programmes are withdrawn from the low speed store in blocks, each block is assembled from the tape into the high speed store and then transferred to the low speed store by a special routine before proceeding to reading in the next block of commands.

XV. Manual Controls

The manual controls include those for starting and stopping the computer at will, for clearing registers, and for setting certain 20-digit data. Others, intended to help the operator, allow the computer to operate in special ways.

Three rows of 20 switches each provide for the adjustment of the content of registers N_1 and N_2 and of the input register. Register N_1 possesses an added facility which allows its content to be accepted as a command irrespective of the content of the sequence register. This allows the operator to change the content of any arithmetical register or of the stores at will, and so allows of manual adjustment of programmes when corrections to them are needed.

Besides the independent application of individual commands via N_1, a succession of commands differing only by sequential sub-addresses may be accepted. The sub-address in each case is the current content of the sequence register. By use of this facility data from the low speed store may be transferred to the high speed store or vice versa item by item, and the whole of the content of a store may be transferred within a few seconds without the use of any additional programme. This facility is also useful in maintenance tests.

A further device called a " trigger unit ", useful for maintenance tests and in programme tests, provides a cathode-ray oscillograph trigger pulse whenever the content of the sequence register coincides with that set on a special group of switches or will stop the computer so that the content of the register may be inspected.

XVI. References

Pearcey, T., and Hill, G. W. (1953).—*Aust. J. Phys.* **6**: 316.

Wilkes, M. V., Wheeler, D. J., and Gill, S. (1951).—" The Preparation of Programmes for an Electronic Digital Computer." p. 10. (Addison-Wesley Press Inc.: Cambridge, Mass.)

Appendix I

PRIMARY ROUTINE AND CONTROL ROUTINES USED DURING THE ENTRY OF DATA INTO THE COMPUTER

Primary routine: locations 0–19; control routine: locations 20–31

Position of Command		Command			Action
0		$(I) \rightarrow C$			Reads column of 10 digits, X, Y to register C
1		$p_{2_0} \overset{+}{\rightarrow} C$			Alters most significant digit (Y)
2		$p_{20} \cdot (C) \overset{c}{\rightarrow} S$			Tests most significant digit
3		$20 \rightarrow S$		[20]	Shifts to 20 if Y on current column
4		$p_{20} \cdot (D_0) \overset{c}{\rightarrow} S$			If no Y punch, tests most significant digit of D_0

APPENDIX I (*Continued*)

Position of Command	Command		Action
5	$12 \rightarrow S$	[12]	If no p_{20} in D_0, shifts to 12
6	$[c(A) \rightarrow 20]$		" Transfer command "—plants A and clears A ; called if previous column X-punched and current column no X or Y
7	$(6) \rightarrow A$		Increases store address of transfer command by unity
8	$p_{11} \overset{+}{\rightarrow} A$		
9	$c(A) \rightarrow 6$		
10	$p_{20} \cdot (D_0) \overset{c}{\rightarrow} S$		Tests most significant digit of D_0 again, p_{20} if previous column X-punched and current column no X.
11	$0 \rightarrow S$	[0]	Returns to read if no X on previous column
12	$p_{19} \overset{-}{\rightarrow} C$		
13	$p_{20} \cdot (C) \rightarrow D_0$		If current column X-punched, p_{20} in C is read to D_0 ; if no X, D_0 cleared
14	$(C) \rightarrow Hl$		Clears out p_{19} and p_{20} digits from C and leaves 10 digits in C and H
15	$(Hl) \rightarrow C$		
16	$p_{20} \cdot (D_0) \overset{c}{\rightarrow} S$		Tests for X on current column
17	$(H_u) \rightarrow C$		If no X, shift column digits to p_{11}–p_{20} groups
18	$(C) \overset{+}{\rightarrow} A$		If no X, add 10 digits p_{11}–p_{20} to A
19	$0 \rightarrow S$	[0]	Return to " read "
20	$(D_0) \overset{-}{\rightarrow} D_0$		Clears D_0 for fresh assemblage to follow
21	$(6) \rightarrow H_u$		Stores address of transfer command sent to H register
22	$(C) \overset{c}{\rightarrow} S$		Digits associated with Y punch counted to sequence register
23	$(31) \overset{+}{\rightarrow} A$		If $(A)=n$, then $(A)'=c(A) \rightarrow n$
24	$9 \rightarrow S$	[9]	Causes transfer command to be replaced by (A) ; $(6)'=\underline{(A)}$
25	$(30) \overset{+}{\rightarrow} A$		This and following cause $H_u \rightarrow 31+\underline{(A)}$ to be placed in 28
26	$(23) \overset{+}{\rightarrow} A$		Causes $(31) \overset{+}{\rightarrow} A+\underline{(A)}$ to be placed in 28
27	$c(A) \rightarrow 28$		Assembled (A) planted for use
28	$p_1 \rightarrow T$		Command assembled placed here and performed
29	$0 \rightarrow S$	[0]	Return to continue reading data
30	$<(0) \rightarrow T>$		Constant when added to (23) forms $(H_u) \rightarrow 31$
31	$<c(A) \rightarrow M>$		Constant used by command 23