An attempt to simplify coding for the Manchester electronic computer

1955 Br. J. Appl. Phys. 6 307

(http://iopscience.iop.org/0508-3443/6/9/302)

An attempt to simplify coding for the Manchester electronic computer

By R. A. BROOKER, M.A., A.R.C.S., Department of Electrical Engineering, University of Manchester

[*Paper received* 24 *March*, 1954]

This paper outlines some of the difficulties encountered in attempting to simplify the task of preparing programmes for a large scale digital computer. In general this is most simply achieved at the price of increasing the machine time consumed by the resulting programme. Such a compromise is nevertheless worth while for many "small" problems and a means is described for treating such problems on the Manchester University Computer (Ferranti MK I computer). The method is also applicable to any of the Ferranti range of machines, several of which are now available in this country.

The task of preparing programmes for large scale digital computers has always been recognized as a bottleneck to the use of these machines by persons whose main interest lies only in the results and not in the techniques of coding. For this reason several proposals have been made for converting into precise machine instructions, programmes originally drawn up in a more convenient and general form.[1,2] In all these proposals the conversion is done by the machine itself in the following way. The ordered aggregate of symbols which constitute the original programme are presented to the machine on the input medium. It is then scanned by the machine under the operation of an input-conversion routine which translates the symbols on the input medium into a corresponding programme of words which are recorded in the store. These words may be the final instructions which are subsequently obeyed directly by the machine, or they may be words which serve as "instructions" for a further interpretive routine. The purpose of this report is to describe a means for writing programmes which the use of conversion and interpretive techniques has made possible on the Manchester University Electronic Computer MK I. This is a typical two-level storage machine and has the following characteristics:

working store (electrostatic): 512 words each of 20 bits;
auxiliary store (magnetic drum): 256 tracks each of 128 words;
fixed binary point arithmetic on 40 bit words: addition time 1·2 ms, multiplication time 2 ms;
time to transfer contents of any track to working store: 40 ms, reverse operation: 90 ms;
input-output medium: 5-hole teleprint tape, page printer.

STRATEGICAL CONSIDERATIONS

In coding directly for such a machine two main difficulties are encountered which should be taken into account in any proposals for simplifying the coder's task.

The first is the scale factor difficulty of arranging the calculation so that all the quantities involved are represented to the required accuracy and yet remain within the capacity of the registers. This difficulty can be alleviated to some extent by working throughout with double length arithmetic keeping, say, 40 binary digits for the fractional part of a number, and 40 digits for the integral part. Another solution to the problem is to associate every number occurring in the calculation with its own adjustable scale factor: an interpretive scheme for doing this has been described by Brooker and Wheeler.[3] This so-called floating-point technique requires the use of routines for carrying out operations of the form $a.2^p \pm b.2^q$, etc., and for this reason the time of execution of a floating-point instruction is very long (addition 50 ms, multiplication 40 ms, division 100 ms) compared to that of the corresponding fixed point machine instructions. This is partly offset by the fact that, in general, fewer of the floating variety are needed. It is this second solution which has been adopted in the scheme developed for the Manchester machine.

The second difficulty relates to the physical nature of the storage facilities. The coder's task would be made easier if, instead of two levels of storage with their corresponding accessibility features, there was a single storage medium of indefinitely large capacity and rapid accessibility. Any proposals for simplifying coding ought therefore to include some means for converting a programme written for a hypothetical one-level store into a programme suitable for a two-level store. The real problem is to do this efficiently, so that the computation time spent on the "fast" level approximately matches the time spent in gaining access to material (numbers and instructions) on the "slow" level. As in the case of the scale factor problem an optimum solution really requires an insight into the calculation being programmed. Thus, for example, when allocating magnetic storage to the elements of a vector one would try to keep them as far as possible on the same track so that the entire vector could then be transferred to the working store in a single operation; in other words, the experienced programmer tries to take advantage of any natural ordering that exists in the numerical material. When a machine has been educated to do this it will have gone some way towards being able to think! Nothing so ambitious has been attempted in the scheme adopted, mainly because the use of floating arithmetic does not make it worth while to improve on the method described below for gaining access to numerical material.

The strategy adopted for using the two levels of storage is as follows. After input, the programme of converted "instructions" is recorded on consecutive lines and tracks of the magnetic store. The capacity of the working store is limited to four tracks of the auxiliary store. At any stage, therefore, in the execution of the programme there is only room for one trackful of "instructions" and one trackful of numbers, the remaining space (one-half) being taken up by the interpretive routine. The "instructions" are normally used consecutively and when the trackful has been exhausted arrangements are made to replace it by the contents of the next track, or in the case of a jump "instruction," by whichever new track is involved. This track changing business is

also a feature of the conventional programming technique used with the Manchester machine and has been described by Hume[4]: supplemented by suitable facilities in the input conversion stage it enables the instructions to be written out originally as if for a one-level instruction store. This means of access to "instructions" is efficient because the time taken to transfer each trackful of "instructions" is negligible compared with the time taken to execute them: the ratio is still quite reasonable even where ordinary (machine) instructions are concerned.

The situation is not so favourable, however, in the case of numbers because the order in which access to them is required depends on the nature of the calculation. The conversion problem is simplified if the operands are selected directly from the auxiliary drum store—the capacity is virtually infinite—by the interpretive routine. To do this the routine has first to determine the track involved, then transfer the content of the track to the working store, and finally select the particular line required. The access time for any operand is then at least 40 ms, the time for a magnetic reading transfer. Similarly the corresponding recording operation would take at least 90 ms. These figures about match the times of floating operations so that in this case not much advantage is to be gained by improving the access time.

The position would be very different, however, if the arithmetical operations took only a few milliseconds or less, for in that case most of the time would be spent in selecting and recording numbers on the drum. To achieve any sort of balance in these circumstances it is essential to take advantage of any natural ordering of the numbers. Now, however crudely the original programme is written, any such ordering would almost certainly be reflected in the allocation of storage in the hypothetical one-level store, and the correspondence between this store and the magnetic drum store means that in many cases successive operands will be found on the same track. Unnecessary magnetic transfers can therefore be eliminated by testing whether the magnetic address of each operand involves the same track as the previous operand. This can be done either by the interpretive routine as each operand is required, or once and for all before the programme is executed. The former device has been incorporated in the Manchester scheme because it is simpler and the rewards are just worth while. The second and ultimately more rewarding method would involve examining the context of each "instruction," which, in the case of an "instruction" immediately following a "control" junction, means searching through all possible alternative "control" paths to find the previous operand. This can only be done after input when the whole programme—or at least the important parts, the inner loops—are available for scanning. This stage will be called *secondary conversion*; such conversion as it is convenient to do during input proper will be called *primary conversion.* Primary conversion will include all those techniques usually associated with compiler routines, i.e. use of algebraic symbols, floating addresses,[5] specification of subroutines by a symbol, etc.

[With most machines the floating address facility usually involves secondary conversion, or its equivalent, the "two-pass" technique. Applied to control transfers, however, this is unnecessary with the Manchester machine because of the particular way in which control transfer instructions work. They involve a double reference: instead of giving directly the address of the next instruction (*the control number*) it is the address of this address which is given. Thus for each control transfer instruction a spare location must be found to hold the control number, the address of the next instruction. The associated locations, which are arbitrary, can be allocated as each transfer instruction is read from the tape and the corresponding control numbers filled in as the instructions to which control is transferred are read from the tape.]

CONCLUSION

It can thus be seen that with the aid of routines for the calculation of the elementary functions, and input and output of numerical data, the above techniques enable the machine to be transformed, at the price of considerable loss of speed, into a programmer's conception of an "ideal machine." The total material involved in the transformation of the Manchester machine occupies about twenty tracks of the magnetic store. It is usually run into the machine when required by means of the conventional input routine which is kept permanently on the magnetic drum.

It remains to consider the "ideal machine." In designing the code described below the following points were considered. The price paid in the terms of speed rules out the possibility of using the "machine" for problems involving $0(n^3)$ or more operations where n is large, e.g. partial differential equations. Its use will therefore be confined to *ad hoc* problems and problems which might otherwise have been considered as too small for a large scale computer; a typical example is given later on. Extending the class of problems that can be treated by the machine in this way may mean extending the class of user. The occasional user with a small problem will not, naturally, be very keen to learn the conventional programming techniques, more especially if this means digesting the contents of a hundred-page manual. For this reason the principal consideration in the choice of the "ideal machine" has been economy of description: what the author aimed at was two sides of a foolscap sheet with possibly a third side to describe an example. The psychological advantages which such an economy of description brings to the prospective reader makes the attempt worth while, even at the expense of considerable compression of the material. How far this has been achieved can be judged from the following standard account which is similar to one we hand out to prospective occasional users.

The "machine" used at present differs from that described here in certain minor respects (e.g. the accuracy is limited to nine significant decimals) but will shortly be modified to conform with the following account. Further details can be obtained on application to this laboratory.

THE STANDARD ACCOUNT OF THE "SIMPLE MACHINE"

By the means described above the electronic computer can be made to accept programmes written in the simplified form described below. In this form a programme of calculation consists of an ordered sequence of *instructions* arranged in a

List of permissible symbols employed

FS	*FS* (Blank)	*K*	–	*V*	6
A	1	*L*	*v*	*W*	/
B	2	*M*	*LF*	*X*	⊗
C	*	*N*	*SP*	*Y*	9
D	4	*O*	,	*Z*	+
E	(	*P*	0	*LS*	*LS*
F	)	*Q*	>		
G	7	*R*	⩾	?	*n*
H	8	*S*	3	£	*CR*
I	≠	*T*	*j*	■	■ (Erase)
J	=	*U*	5		

single column. These instructions are chosen from a permissible set and employ only the symbols given in the table. Ultimately the programme is presented to the machine in the form of a length of perforated paper tape which is scanned by a photoelectric tape reader, the input unit of the machine. This programme tape is prepared on a keyboard perforator in a manner to be described later. The machine gives its results on a page printer.

The instructions mostly take the form of equations giving the new value of a computed quantity in terms of one or two previously calculated quantities or parameters. Instructions are also necessary, however, for selecting between alternative courses of action, for printing results, and for the input of further data. The instructions involve the following kinds of quantities.

(*a*) *Variables*. The quantities or intermediate quantities which it is necessary to compute. These are denoted by $v1$, $v2$, $v3$, etc. The number of variables which can be introduced is for practical purposes unlimited: it is in fact about 12000. The range of magnitudes of a variable is virtually infinite, $10^{\pm 10^5}$: the precision is limited to eleven significant decimal figures.

(*b*) *Constants*, or variables of which the value is known in advance. For these quantities the permissible range and precision is limited in order that numerical values may be written in the form:

integral part, decimal point, fractional part.

A certain lack of preciseness is inevitable here, e.g. to six significant figures π may be written 3·14159, 03·14159, 03·141590, +03·141590. All these and similar variations are accepted by the machine, provided the integral part does not exceed 10^{11}: in any case, as explained above, not more than eleven significant figures are recorded inside the machine. Negative numbers must be preceded by sign.

(*c*) *Indices*. These correspond to the subscripts of conventional mathematical notation. In order to take advantage of the repetitive nature of calculation it is desirable to have some means of specifying any of the elements of a sequence, e.g. the components of a vector. For this purpose the notation $vn1$, $vn2$, $vn3$, etc., is introduced where $n1$, $n2$, $n3$, etc., are indices restricted to integral values but otherwise computable like variables. Thus, for example, if $n1 = 5$, then $vn1$ is identified as $v5$. The number of indices which can be introduced is limited to 64. The possible range of values of an index quantity is $-2^{-39} \leqslant x < 2^{39}$, which, of course, far exceeds its usefulness as an index proper. In writing down the numerical value of an index the decimal point and fractional part can be omitted.

Those instructions which take the form of equations are given below as (i), (ii) and (iii). Permissible instructions are obtained from these basic forms by replacing x, y and z by the group of symbols denoting a variable (v), or, except in the case of (iii), an index (n), or a combination (vn). In particular x and y may also be replaced by constants.

(i) $z = x$
(ii) $z = x\theta y$ θ is replaced by one of the symbols $+ \; - \; \otimes \; /$
(iii) $z = Ax$ A is replaced by one of the letters

A ($\equiv \arctan x$)
C ($\equiv \cos 2\pi x$)
E ($\equiv \exp x$)
L ($\equiv \log_e x$)
S ($\equiv \sqrt{x}$)

Instructions are normally obeyed in the order in which they are written down until a jump instruction is encountered. This may be conditional or unconditional and takes the form (iv) or (v) as follows.

(iv) jm means "jump to instruction labelled m"
(v) $jm, x\phi y$ means "jump to instruction labelled m if $x\phi y$," where ϕ denotes one of the symbols $\geqslant \; > \; = \; \neq$.

The *label* is a positive whole number which is written immediately before (i.e. to the left of) the instruction concerned. The remaining instructions are as follows.

(vi) The letter P inserted before any of the instructions (i), (ii) or (iii) causes the computed value of z to be printed in the style described earlier, namely, integral part, decimal point, fractional part. The number of decimal places is determined manually by setting a row of hand switches on the console. Insignificant zeros are suppressed. The machine prints 0·0 if $z < 10^{-11}$; and * if $z > 10^{11}$.

(vii) The following single letter instructions can be inserted at any point in the programme to control the layout of results.

X carriage return
Y line feed
Z space.

(viii) The letter H is a stop instruction: the machine halts. It can be made to resume operation by pressing a key on the console.

As a simple exercise in coding write down instructions for evaluating the sum of squares of $v1, v2, \ldots, v100$. A suitable sequence is:

$$\begin{aligned} n1 &= 1 \\ v101 &= 0 \\ 2\,v102 &= vn1 \otimes vn1 \\ v101 &= v101 + v102 \\ n1 &= n1 + 1 \\ j2,\ 100 &\geqslant n1 \end{aligned}$$

Such a group of instructions may form part of a larger programme involving the variables in question.

A complete programme of instructions and numbers is normally recorded *inside* the machine before being executed: it is therefore necessary to explain the *input* procedure.

The programme tape is prepared on a keyboard perforator on which are engraved the standard symbols. The material is "punched" in the conventional fashion, namely from left to right and down the column. Each instruction is followed by the symbols *CR* (carriage return) and *LF* (line feed). The keyboard is normally on "figures" which means that capital letters have to be preceded by *LS* (letter shift) and followed by *FS* (figure shift). Blank tape corresponds to blank paper and any number of blanks may separate the symbols provided their order is preserved. About six inches of blank should be left at the head of the tape. Associated with the keyboard is a printer which gives a printed copy of whatever is punched; this should agree with the original manuscript.

As the programme tape is scanned the instructions are normally recorded in the store where ultimately they will be obeyed. The rate of scanning is approximately one instruction per second and the rate of execution about eight per second. In the case of instructions included between brackets each instruction is executed immediately after it has been read and is not recorded in the programme proper. This facility is used to start the programme simply by including an unconditional jump instruction between brackets, e.g. ($j1$) which means "stop reading the tape and start obeying the programme at the instruction labelled 1."

Having got the programme started it may be necessary to call on the input medium for further numerical data. This

may be done in two ways for which it is necessary to introduce two further instructions. These are:

(ix) $z = I$, which means "replace z (which has the same significance as before) by the number formed by the next group of symbols on the tape";

(x) the letter T which causes the machine to start reading further instructions from the tape adding them to those already recorded in the store.

The former instruction may be used to read a tape bearing numbers only.

The T instruction, combined with the "bracket" facility, allows data to be input in the form (z = constant). This is a convenient method of altering parameters in between different runs. Thus, for example, if the supplementary instructions take the form

$$\begin{aligned}(&v23 = 0{\cdot}012\\ &v24 = 0{\cdot}965\\ &n3 = 12\\ &j1)\end{aligned}$$

then the effect will be to repeat the run with these new values of the quantities $v23$, $v24$ and $n3$.

To illustrate the coding scheme described above, the following calculation is programmed.

It is required to compute.

$$t = \frac{1-(p_2/p_1)^{2/7}}{1-(p_2/p_3)^{2/7}} \text{ where } p_2 = 30,\ p_3 = 40$$

$$r_e = 1{\cdot}3 + 2\left[1 - \frac{t^{3/2}(p_2/p_3)^{2/7}}{(p_2/p_1)^{2/7}}\right]$$

$$r_m = 1{\cdot}3 + 2\left[1 - \frac{t^{1/2}(p_2/p_3)^{2/7}}{(p_2/p_1)^{2/7}}\right]$$

for values of p_1 of the form $40 - (10n/156{\cdot}4)$, where $n = 0(1)156$.

The programme which required about 10 min to draw up is given below. For each value of p the time of execution is approximately $2\frac{1}{2}$ s plus the time to print the results—about a further $2\frac{1}{2}$ s. If the calculation is treated conventionally the time of execution could be reduced almost to the printing time above but the coding time might well run into several hours.

$2v1 = 40$	$(p_1 = 40)$
$v2 = 30$	$(p_2 = 30)$
$v3 = 40$	$(p_3 = 40)$
$v4 = 10/156{\cdot}4$	
$v5 = v2/v3$	(p_2/p_3)
$v6 = Lv5$	
$v7 = 2/7$	
$v8 = v7 \otimes v6$	
$v9 = Ev8$	$(p_2/p_3)^{2/7}$
$1v10 = v2/v1$	
$v11 = Lv10$	
$v12 = v7 \otimes v11$	
$v13 = Ev12$	$(p_2/p_1)^{2/7}$
$v14 = 1 - v13$	
$v15 = 1 - v9$	
$XYPv16 = v14/v15$	forms and prints t on a new line
$v17 = Sv16$	
$v18 = v17 \otimes v9$	
$v19 = v18/v13$	
$v20 = v/16 \otimes v19$	
$v21 = 1 - v20$	
$v22 = 2 \otimes v21$	
$ZPv23 = 1{\cdot}3 + v22$	forms and prints r_e
$v24 = 1 - v19$	
$v25 = 2 \otimes v24$	
$ZPv26 = 1{\cdot}3 + v25$	forms and prints r_m
$v1 = v1 - v4$	adjusts p_1
$j1, v1 > 29$	tests for last cycle
$(j2)$	starts programme

The following notes are of interest.

(1) The maximum number of instructions allowed cannot be given very precisely but a safe estimate is 500, which, in view of their comprehensive nature, should be adequate for the class of problems envisaged.

(2) No attempt has been made to economize on the use of variables. Thus, for example, instead of $v9 = Ev8$ we could write $v8 = Ev8$ since the argument is no longer needed. However, nothing is gained by so doing since space is virtually infinite for problems of this kind.

(3) A further example of laziness is the means adopted for evaluating 2/7. Instead of bothering to evaluate the fraction we have simply included an instruction for doing so, namely $v7 = 2/7$.

(4) The value of any intermediate quantity can be printed simply by inserting a P before the appropriate instruction. This may be useful in locating mistakes: the P can afterwards be erased by turning it into a ■.

(5) Rounding off. In general the "machine" rounds off only where necessary; in particular the following rules are observed.

(a) During input. Constants, which appear in decimal form on the input medium, are converted to the scale of two inside the machine. Whole numbers are converted exactly but terminating decimal fractions, which may or may not have terminating binary equivalents, are replaced by an approximately equivalent binary fraction. (It is, of course, possible to arrange that those fractions which, like 0·125, do have an exact binary equivalent are converted exactly.)

(b) During the subsequent calculations. In the instructions (ii) rounding off takes place only where the result extends beyond 39 significant binary digits. In the instructions (iii) the accuracy obtainable depends on the nature of the function and argument. For example, in forming the cosine of an angle of about 1000 rev, three significant decimals are unavoidably lost. The jump instruction (v) involves taking the difference $x - y$, an operation which is identical with the corresponding form of instruction (ii).

(c) During output. Numbers are first converted from floating binary to fixed binary form, digits beyond 2^{-40} being discarded. The resulting number can be converted to decimal form exactly since a terminating binary fraction has a corresponding decimal equivalent; but in fact conversion is carried only as far as the number of decimal places required—set on the hand-switches.

This completes the standard account. The "simple machine" is not necessarily the ideal machine for all purposes. For example, the coding of repetitive work could be made more economical in instructions by introducing variables whose index is a general linear function of one or more index quantities. However, this facility would not be appreciated in the kind of problems to which the "simple machine" is restricted. Moreover the inclusion of such facilities would

tend to destroy the main advantage of the "simple machine," namely its simplicity.

REFERENCES

(1) MUTCH, E. N., and GILL, S. *Proceedings of a Symposium on Automatic Digital Computation*, Paper No. 11, National Physical Laboratory, March, 1953 (London: H.M. Stationery Office, 1954).

(2) *Symposium on Automatic Programming for Digital Computers*. Sponsored by the Navy Mathematical Computing Advisory Panel at Washington, D.C., May, 1954.

(3) BROOKER, R. A., and WHEELER, D. J. Floating operations on the Edsac, *Math. Tables Aids Comput.*, **7**, p. 37 (1953).

(4) HUME, J. N. P. Input and Organisation of Subroutines for FERUT, *Math. Tables Aids Comput.*, **8**, p. 30 (1954).

(5) WILKES, M. V. *Proc. Cambridge Phil. Soc.*, **49**, p. 84 (1953).

Instability of photomultipliers

By L. P. DE VALENCÉ, M.Sc., A.Inst.P., Institute of Cancer Research, Royal Cancer Hospital, London, S.W.3

[*Paper first received* 1 *April, and in final form* 1 *June*, 1955]

A phenomenon of dark current instability in specified types of electrostatically focused photomultiplier tubes is described. Particular attention is drawn to the variation with glass envelope potential of dark current fluctuation under fixed circuit constant conditions. The degree of instability does not appear to be related to the relative photometric sensitivity on the tube. A possible explanation of the phenomenon on the basis of electron defocusing is suggested, and a satisfactory method of eliminating the disturbance is described.

This paper describes an investigation into a phenomenon of photomultiplier dark current instability, under conditions obtaining in many experimental procedures where the tube is housed in an earthed metallic casing and operated at room temperature with its anode at approximately earth potential and its cathode at a potential of -1000 V. Under such conditions, it has been found that should the glass envelope either touch, or come into close proximity with, the earthed metallic housing, both the level and the fluctuation in the dark current increase, in some cases by a very large factor. The fluctuation is here defined as the maximum variation in dark current under fixed circuit constant conditions.

An investigation into the effect of glass envelope potential on the dark current *level* has been reported by Taylor,(1) and Morton and Mitchell,(2) whose results are confirmed in the present work. These authors, however, do not deal with the influence on dark current *fluctuation*, which effect is stressed here.

This phenomenon of instability is clearly of importance in experiments concerned with the detection of small changes in light flux, using sensitive scintillation dosemeters of the type described by Belcher.(3) The effect was first observed in optical experiments designed to study the behaviour of light pipes in clinical scintillation dosemeters. In these experiments the photomultiplier and its dynode resistor chain were mounted in a light-tight brass housing of about twice the diameter of the photomultiplier, provision being made for a lateral positional adjustment of the photomultiplier for centring purposes on an optical bench. It was observed that in an extreme setting of the photomultiplier, when the glass envelope either touched or was very close to the metallic housing, the dark current increased considerably and, even of greater importance (since the dark current itself could be "backed off" electronically) there was a considerable increase in the dark current fluctuation. This markedly reduced the accuracy with which small changes of light flux could be detected above the noise level.

EXPERIMENTAL

In the experiments described below, measurement of dark current was made using a valve voltmeter of very high input impedance, based on a circuit suggested by Scroggie.(4) This instrument measures the potential developed by the photomultiplier output current, across a high stability resistor of $10^8\ \Omega$. In all the investigations that follow the time constant of the associated electrical circuits was 10^{-1} s.

In a preliminary investigation, the photomultiplier was supported horizontally, free from any earthed screens, in a large light-tight box. The dark current measured under these conditions is hereafter referred to as "normal." A single turn of naked copper wire was then loosely suspended around the glass envelope and the dark current was again found to be normal, provided that the wire was electrically floating. If, however, the wire was earthed the dark current and dark current fluctuation increased noticeably.

The same phenomenon was observed when the single turn of wire was replaced by a turn of insulated wire. In this case, however, a capacitive effect was apparent in that immediately the photomultiplier e.h.t. was switched on, the dark current and fluctuation was excessively high, and then gradually decreased to an equilibrium value which was higher than normal. The time taken to reach the equilibrium value could be increased by increasing the number of turns of wire. Further, the actual position of the wire on the glass