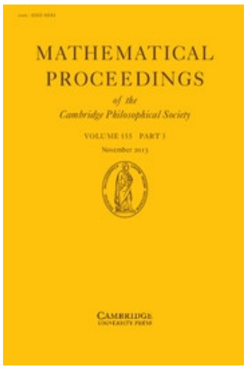

The use of a 'floating address' system for orders in an automatic digital computer

M. V. Wilkes

Mathematical Proceedings of the Cambridge Philosophical Society / Volume null / Issue 01 / January 1953, pp 84 - 89
DOI: 10.1017/S0305004100028073, Published online: 24 October 2008

THE USE OF A 'FLOATING ADDRESS' SYSTEM FOR ORDERS IN AN AUTOMATIC DIGITAL COMPUTER

By M. V. WILKES

Received 24 June 1952

This note is concerned with automatic digital computing machines in which orders are expressed in coded numerical form and held in the same store as numbers. It is possible in such machines to modify orders by performing arithmetical operations on them. The storage locations are numbered serially, and their contents can be referred to when necessary by these numbers.

Wheeler (1) has shown, with special reference to the Edsac, that the work of drawing up a programme can be lightened by the use of a system of relative numbering of storage locations. If this is done, the storage locations used in any programme are divided into a number of separate sequences, each numbered from 0 upwards. The address specified in each order is marked to show the sequence to which it belongs by having a distinctive symbol punched after it on the input tape. When the orders are assembled into a programme, the programmer specifies the location in the store into which the first member of each sequence is to go, and when the orders are read from the input tape the addresses are modified by the addition of the appropriate constants (different for each sequence). This modification of addresses is carried out by a set of 'initial orders' which control the reading of the tape, and the effect is to provide the programmer with facilities which are additional to those built into the machine by the designer. One way of expressing this is to say that the programmer has the benefit of the use of an order code which is better adapted to his needs than the basic order code of the machine. In particular, a relative numbering system greatly facilitates the use of subroutines. If he makes use of an 'assembly subroutine' of the kind devised by Wilkes, the programmer may even be relieved of the task of deciding the precise locations in the store into which the first orders of the various sequences are to go ((2), p. 29).

In this paper a further step is proposed which enables the programmer to be set free to a still greater extent from the limitations of the basic numbering system of the machine. If a programme is examined it will usually be found that only a small fraction of the orders contained in it ever need to be referred to by the serial numbers of the storage locations in which they are placed. A system by which every order is numbered is therefore unnecessarily rigid and carries with it the disadvantage that the programmer must be prepared to undertake extensive renumbering whenever extra orders are inserted into the middle of a programme either to modify its action or to correct an error. In order to avoid this he will normally attempt to postpone numbering until he considers that his programme is in its final form. In the system of 'floating addresses' described here it is proposed that only those orders which are referred to at other points in the programme should be numbered and that the numbers should be allotted freely by the programmer, not necessarily in numerical order. The programmer is then completely relieved of the necessity of renumbering when he makes alterations and he can write the addresses specified in the orders in their correct form from the beginning. The use of floating addresses was referred to briefly at the Joint AIEE-IRE Computer Conference held in Philadelphia in December 1951 (3).

The construction of a set of initial orders providing for floating addresses would be very simple if the floating addresses were restricted so that they could refer only to orders punched earlier on the tape. The initial orders would then make a record of the addresses of the storage locations into which the numbered orders were placed, and would insert these addresses, during input, into orders which required it. However, the restriction referred to would be such as to render the system of little use, and it is therefore necessary for the process of incorporating the correct addresses in orders to be carried out after the whole programme has been placed in the store. A record must be made of the orders whose addresses have to be inserted in this way, either by marking them as they are placed in the store or by forming a list elsewhere in the store.

In order to enable the ideas described above to be tried out a subroutine was constructed which, working in conjunction with the existing initial orders, would provide floating address facilities for the Edsac. The subroutine is given in detail below together with an example of its use. It is assumed that the reader is familiar with the order code and the initial orders used in the Edsac. Details of these are given by Wheeler (1) and by Wilkes, Wheeler and Gill (2).

Description of the floating address subroutine. The action of the floating address subroutine can best be understood by examining the example given below which consists of a master routine and a library subroutine. The floating address system is used only for the master routine, the library subroutine being copied on to the programme tape from the library tape in the usual way. In the master routine those orders which need to be numbered have punched in front of them the symbols G 1 K, G 2 K, G 3 K, etc. These symbols need not be in numerical order. Addresses in orders which refer to locations specified in this numbering system are terminated by θ. These particular symbols were chosen because the existing initial orders could then, if certain modifications were made, be readily adapted to the purposes of the floating address subroutine. Addresses may also be terminated by F, D, or any of the other code letters (except H or N) ordinarily used in Edsac programmes.

When the initial orders are used in their standard form the control combination GnK sends control to order 32 of the initial orders with GnF in the accumulator. When the floating address subroutine is being read from the input tape a jump order is planted in 32 to replace the order normally there, so that the combination GnK now sends control to the floating address subroutine. This subroutine causes the address in the transfer order to be recorded in storage location $l+n$, where l is a constant equal to the number of the storage location immediately following the subroutine. A 'directory' is thus built up from which the storage location in which each numbered order has been placed may be ascertained.

Each order in which the address is terminated by θ is marked, before being placed in the store, by having 2^{-5} added to it. This makes the sixth binary digit (to which no significance is normally attached in Edsac orders) a 1 instead of a 0. The addition is done automatically by the initial orders, the constant 2^{-5} (the pseudo-order $P\,1024\,F$) being placed beforehand in storage location 42. When all the orders of the master routine have been read into the store, control is sent by means of a control combination punched on the tape to a special entry point in the floating address subroutine, and the subroutine proceeds to examine each order in turn to see whether it contains a 1 in the sixth binary position. If it does, the number in the address section of the order is replaced by the correct address taken from the directory; otherwise, the order is left unaltered and the subroutine proceeds to examine the next order. When all the orders have been processed in this way, the floating address subroutine restores the initial orders to their original form and sends control back to them so that further orders and control combinations are read from the tape according to the standard procedure.

The orders of the floating address subroutine are given in the next section, followed by a programme for forming and printing $\sum_{0}^{99} a_n^2$, where $a_n = C(200+n)$. It is assumed that the sequence of numbers a_n has been placed in advance in the store. This example is somewhat artificial, but serves to illustrate the use of the floating address system without being unduly long.

Floating address subroutine. The following control combinations and pseudo-orders are punched on the tape in front of the floating address subroutine (a and b stand respectively for the storage locations into which the first order of the master routine and the first order of the floating address subroutine are to be placed):

```
T a  K
G    K
T 45 K
P b  F   goes into 45 (H parameter)
```

The floating address subroutine is as follows:

```
       P     θ   goes into 46 as PaF (N parameter)
       E  25 K \ cause transfer order to be replaced by TH ≡ T b F
       T     H /
       G     K
   0   A  30 θ \ restore order in 32 to normal
   1   T  32 F /
→  2  (H     N)  selects order to be processed (becomes H a+r F, where a is location of
                   first order to be processed)
   3   A   2 θ \
   4   A   2 F  > increase r (note C(2) = P 1 F)
   5   U   2 θ /
   6   S  31 θ \ form and plant T a+r F
   7   U  21 θ /
   8   S  32 θ \ test and jump when processing is complete
   9   E  36 θ /
  10   T     F   clears accumulator
  11   C  42 F \
  12   S  42 F  > jump if marking digit is 1 (note C(42) = 2^-5)
  13   E  16 θ /
  14   T     F   clears accumulator
  15   E   2 θ
→ 16   C  33 θ \
  17   A  34 θ  > form and plant A 39+α θ where α is address in order being processed
  18   T  19 θ /
```

19	$(Z\quad F)$	becomes $A\ 39+\alpha\ \theta$
20	$C\ 35\ \theta$	forms order with corrected address
21	$(Z\quad F)$	replaces order in programme
22	$E\ 2\ \theta$	jump to process next order
→23	$A\ 38\ \theta$	control comes here after $G\ n\ K$ with $G\ n\ K$ in accumulator. Form and plant $T\ 39+n\ \theta$.
24	$T\ 28\ \theta$	
25	$A\ 22\ F$	place copy of transfer order in $32\,\theta$
26	$U\ 32\ \theta$	
27	$S\ 8\ F$	leaves address in transfer order in accumulator (note $C(8) = T\ F$)
28	$(Z\quad F)$	becomes $T\ 39+n\ \theta$ and places address in directory
29	$E\ 34\ F$	returns control to initial orders
30	$A\ 22\ F$	
31	$K\ 4097\ F$	equivalent to $(H-T)\ 1\ F$
32	$(P\quad F)$	becomes copy of transfer order
33	$P\ 1023\ F$	equivalent to $P\ 2^{10}-1\ F$
34	$A\ 39\ \theta$	
35	$V\quad D$	
→36	$H\ 8\ F$	restore multiplier and return control to initial orders
37	$E\ 34\ F$	
38	$J\ 39\ \theta$	
	$T\ 32\ K$	plant jump order in initial orders during reading of tape
	$G\ 23\ H$	
	$T\ 42\ K$	place 2^{-5} in 42
	$P\ 1024\ F$	
	$E\ 25\ K$	cause transfer order to be replaced by $TN \equiv T\ a\ F$
	$T\quad N$	

The floating address subroutine is followed on the tape by the master routine and any other parts of the programme which are written in terms of floating addresses. These are followed by the control combinations

$$\begin{array}{l} G\quad K \\ E\ 25\ K \\ E\quad H \\ P\quad F \end{array}$$

Example. The master routine is

	$T\ 300\ D$	clears $300\ D$
	$S\ 5\ \theta$	part of counting sequence
→$G\ 3\ K$	$T\ 4\ \theta$	
$G\ 1\ K$	$H\ 200\ F$	form a_n^2 (initially $n = 0$)
$G\ 2\ K$	$V\ 200\ F$	
	$A\ 300\ D$	add to partial sum in 300
	$T\ 300\ D$	
	$A\ 1\ \theta$	modify H and V orders so that a_{n+1}^2 is formed next time (note $C(2) = P\ 1\ F$)
	$A\ 2\ F$	
	$T\ 1\ \theta$	
	$A\ 2\ \theta$	
	$A\ 2\ F$	
	$T\ 2\ \theta$	
	$A\ 2\ F$	part of counting sequence
	$A\ 4\ \theta$	
	$G\ 3\ \theta$	
	$H\ 300\ D$	place sum in multiplier register ready for printing
$G\ 6\ K$	$A\ 6\ \theta$	call in print subroutine
	$G\ 100\ F$	
	$Z\quad F$	stop
$G\ 4\ K$	$(P\quad F)$	counting number
$G\ 5\ K$	$P\ 100\ F$	number of repetitions

It will be observed that floating addresses are used for pseudo-orders as well as for orders.

In this example the first order of the master routine goes into 50 and that of the floating address subroutine into 100; the print subroutine (*P* 15 from the library) is later written over the floating address subroutine. The make-up of the tape is given below and may be compared with that given on p. 47 of (2), which illustrates the normal method of making up tapes for the Edsac.

```
      P     K
      T  50 K
      G     K
      T  45 K
      P 100 F
 ┌────────────────┐
 │Floating address│
 │   subroutine   │
 └────────────────┘
  ┌──────────────┐
  │Master routine│
  └──────────────┘
      E  25 K
      E     H
      P     F
       space
      P     K
      T 100 K
      ┌─────┐
      │P 15 │
      └─────┘
      E  50 K
      P     F
```

Alternative methods. A number of variants on the method used in the subroutine described above will suggest themselves. As an alternative to the system of marking orders which have later to be processed it would be possible to build up a second directory in which their addresses were recorded. More space in the store would be used, but this would be a secondary consideration, since the storage space used by the floating address subroutine is only required temporarily during input.

Synthetic orders. The material of this section is only generally related to the rest of the paper, but it illustrates a type of programme technique which becomes attractive when floating addresses are used. In any machine there will be a number of simple operations which are not covered by the machine order code and which therefore need a group of orders to perform them. A *synthetic order* is defined as a tape entry which has a form similar to that of a machine order but which calls for the performance of one of the 'non-existent' operations. During input a synthetic order is expanded into the requisite sequence of orders, either by a special subroutine, or by the initial orders themselves. Synthetic orders are most useful for dealing with operations which require no more than 5 or 6 orders, and which could not, therefore, be economically performed by means of closed subroutines. The principle was tried out on the Edsac by means of a subroutine which included provision for the synthetic orders set out below. Each synthetic order is distinguished by being terminated by the code letter *S* and is followed by a pseudo-order.

A n S *P m F*	increase address of order in storage location *n* by *m*
R n S *X m F*	replace order in storage location *n* by *X m F* where *X* stands for any function letter
E n S *P m F*	transfer control to storage location *n* if $C(Acc) > m.2^{-15}$
C n S *P m F*	transfer control to storage location *n* on the first *m* occasions this order is encountered and proceed serially on the *m* + 1th occasion

When speed of operation rather than the amount of storage space occupied is the important consideration, synthetic orders might also have a useful application in connexion with double-length or complex arithmetic. They would provide the same degree of convenience for the programmer as would an interpretive subroutine, but the time needed to operate the subroutine would be saved.

The use of synthetic orders would be rather troublesome for the programmer if a floating address system were not available, since he would have to remember to leave the correct number of storage locations vacant to allow for the expansion of the synthetic orders.

As an example of the use of synthetic orders the master routine given above is rewritten below using synthetic orders from the above table. It is assumed that, in addition to indicating a synthetic order, the code letter *S* has the same effect as θ:

	T 300 *D*	
G 1 *K*	*H* 200 *F*	
G 2 *K*	*V* 200 *F*	form a_n^2 and add to partial sum
	A 300 *D*	
	T 300 *D*	
	A 1 *S*	
	P 1 *F*	modify *H* and *V* orders
	A 2 *S*	
	P 1 *F*	
	C 1 *S*	count
	P 100 *F*	
	H 300 *D*	
G 3 *K*	*A* 3 θ	print
	G 100 *F*	
	Z *F*	stop

REFERENCES

(1) Wheeler, D. J. *Proc. roy. Soc.* A, 202 (1950), 573.

(2) Wilkes, M. V., Wheeler, D. J. and Gill, S. *The preparation of programmes for an electronic digital computer* (Cambridge, Mass., 1951).

(3) *Review of electronic digital computers.* Joint AIEE-IRE Computer Conference, Dec. 1951 (p. 114), AIEE, New York.

The Mathematical Laboratory
Cambridge