Published: 01 April 1957

ARTICLE

An Automatic Programming Routine for the Elliott 401

F. YATES, Rothamsted Research, Harpenden, Hertfordshire, U.K.

S. LIPTON, Rothamsted Research, Harpenden, Hertfordshire, U.K.

An Automatic Programming Routine for the Elliott 401*

F. YATES AND S. LIPTON

Rothamsted Experimental Station, England

Introduction

In a recent paper [1] Gordon has outlined a routine for the IBM type 650 calculator, which recodes the addresses of a programme written without considerations of optimum coding, so as to produce a final programme which is reasonably optimal as regards timing. A similar automatic programming routine has been constructed for the Elliott 401 computer. This is a prototype machine which was built by Elliott Brothers under a development contract with the National Research Development Corporation. It is at present operated by the Statistics Department of Rothamsted Experimental Station and is mainly used for statistical work in agricultural and biological research.

The two routines, which were developed independently, use similar methods for recoding the addresses so as to obtain satisfactory timing. The 401, however, has a considerably more complex order code than the 650, and the 401 routine is designed to perform a number of operations which are not required in programming for the 650.

Programming for the 401

The 401[1] has a one-plus-one address system with a word time of 100 μs and a word length of 32 binary digits. The 10 most significant digits of an order specify the address (termed the "A_2" address) of the next order to be obeyed. The ten least significant digits specify the address (the "A_1" address) that gives the store location when there is reference to the store, or the address of the alternative next order in a discrimination order. The address in the store at which the order is located is termed the A_0 address. All these addresses have timing significance. The central 12 digits of an order are the operational digits; they are split into four groups of 3, termed the source (S), function (F), destination (D), and control (K).

The main store is a magnetic disc of 23 tracks numbered 0 to 6 and 7/0 to 7/15, each of 128 words. Orders are read directly from the disc. Switching between tracks 7/0 to 7/15 is by relays, with change from one relay track to another being effected by a relay switching order. Reference to any relay track has its track number coded 7. Switching between tracks 0 to 6 and the particular relay track in circuit at the time is electronic. Writing on track 0 is inhibited, as this track is used for a set of permanent input routines and a division routine. The only immediate access storage consists of five single word registers, of which

*Received August, 1956.

[1]A note on this computer is given in [2].

one (R_1) is used as the accumulator, the second (R_2) is used in double length working, and the remaining three (R_3, R_4 and R_5) are used for temporary storage and modification (B modification) of orders. In multiplication the factors are placed in R_2 and R_3.

Clearly it is essential, if satisfactory operating speeds are to be attained, to make the timing of programmes optimal. Moreover, the coding is further complicated by a number of conditions which must be satisfied by the various addresses of certain types of orders. For example, in a multiplication order A_2 must be exactly 31 word times after A_1, in shift orders A_2–A_1 determines the number of places shifted, while in operations involving R_2 the parity of all three addresses influences the operation. Such conditions, although individually simple, make in the aggregate considerable demands on the programmer and consequently lead to many errors.

Main points of the routine

The routine has been constructed with two main objects in view: Firstly, that the programmer should be freed from having to give attention to the conditions that the addresses have to satisfy, and, secondly, that the addresses should be allocated so that the timing is reasonably optimal. In addition, every effort has been made to construct a routine that is simple to use and that will direct the programmer's attention to any detectable errors while still continuing to operate in the face of such errors.

The following summary lists the main points of the routine and of the simplified coding associated with it.

(1) In so far as orders are to be obeyed in the sequence in which they are written, no indication of the next order to be obeyed is required.

(2) Apart from sets of numbers, such as a set of constants referred to by B modification, the locations of orders and numbers in the store need not be specified.

(3) Conditions that are part of the general rules of programming are imposed by the machine.

(4) Additional items of information not inherent in the order (e.g., number of places of a shift) can be entered in simple form at any time during the writing of the provisional programme.

(5) Provision is made for inserting any real ("final") address (such as an entry point to a sub-routine) into an order, or assigning an order or number to a definite location in the store. It is thus possible to programme part of any routine—such as an inner loop for which the timing is vital—by hand.

(6) The programmer has control over the order in which the routine assigns final addresses to the orders and over the track or tracks on which each sequence of orders is to be placed. Inner loops can thus be dealt with while there are plenty of free locations; this also ensures that addresses which are optimal for the loops are allocated to numbers required in them. The assignment of addresses can be started at the A_1 of the first order of a sequence and/or stopped at the A_1 of the last order.

(7) Printed indication is given of the type and location of any definite or probable errors which can be deduced from internal evidence of the programme under construction.

(8) A final programme is constructed in which the timing is reasonably optimal and all

required conditions are satisfied. This programme is punched out in standard form as an order tape.

(9) A record of occupied addresses is printed in suitable form.

Coding for the routine

The key to the whole procedure is the use of "provisional addresses", which the machine later replaces by final addresses. The standard form in which an order is written is

$$A_0: \quad A_2 \quad \text{SFDK} \quad A_1$$

This form is retained in the simplified coding of the routine, but A_0 represents the provisional address, and A_2 and A_1, if they are written at all, also normally represent provisional addresses. The provisional addresses are obtained by numbering the orders consecutively. This is normally done when the programme is complete. Numbers can be given provisional addresses at the outset or as encountered. Orders or numbers to which final A_0 addresses are assigned before the programme is processed can still be referred to by provisional addresses in A_2 or A_1, thus permitting any necessary organisation of the store to be made after the programme is written without alteration of any A_2 or A_1.

As, in general, orders are written down in the sequence in which they are obeyed it follows that the provisional address that would be in the A_2 position will in most cases be the provisional address of the succeeding order. In this event A_2 is left blank. The few exceptions occur at ends of loops, etc.; here the provisional address of the next order to be obeyed is written in A_2. A_1 is left blank unless the order involves a reference to store or a discrimination.

To carry additional information, such as the final A_0 address (if assigned), number of places in a shift, indication that an A_2 or A_1 address is final, etc., each provisional order has an "auxiliary" associated with it. The auxiliary is written alongside the provisional order and punched with it. In most cases all or most of the auxiliary will be blank. Provision is made to reduce the punching of zeros (representing blanks) to a minimum. The provisional orders and auxiliaries are modified by the routine as the construction of the final programme proceeds.

Other facilities

(a) *Sets of numbers which are referred to by B modification of an A_1 address.* Since a set of numbers of this type must occupy a set of appropriately spaced (usually consecutive) store locations, the locations must be assigned before the programme is processed. This, however, can be done when the programme is otherwise complete, and the set can still be referred to by a single (or if required, more than one) provisional address. The routine checks that all references to such sets are B modified, and that all B modified orders refer to such a set. (This check is particularly necessary in the 401 since the B modification instruction for an

order is contained in the preceding order or orders.) In addition, the range of the set is taken into account when assigning the corresponding A_2.

(b) *Arrangements for setting up programmes on the relay tracks.* In long programmes it is often necessary or advantageous to locate parts of the programme on relay tracks. This involves additional complications since every change of relay track involves a relay switching order, and care must be taken that no reference is made to a relay track not in circuit. Provisions to deal with this are as follows. For each sequence of orders that is to be set up on one or more relay tracks the track number on which setting up is to commence is specified in the sequence control. Whenever a relay switching order is encountered the number of the set up track is automatically changed. In addition, any orders or numbers whose provisional A_0 is coded 0.01 will be placed on a non-relay track whose number is specified in the sequence control. Such orders and numbers can thus be referred to by orders appearing on different relay tracks. If desired the set up of a relay track sequence can commence on this specified non-relay track and so continue until a relay switching order is encountered. In addition, whenever a reference to an order or number already located on a relay track is encountered a test is made that the track number agrees with the relay track number (if any) determined by the current sequence control or last relay switching order of the current sequence.

(c) *Track V programmes.* Certain programmes such as library sub-routines have to be placed on different tracks according to the contexts in which they are used. To allow for this, these programmes have their track numbers punched V (V = 14 and therefore does not in itself correspond to any track). During input, the permanent routine on track 0 replaces the V by the track number designated on the hand switches. Such programmes are constructed by the automatic routine in the same manner as ordinary programmes, setting up on track V being controlled by the sequence controls. In relay track sequences track V acts as the specified non-relay track, and numbers and orders which have the A_0 of their auxiliaries coded 0.01 are always placed on track V.

(d) *Segregation on a single track of store locations in which writing is required.* In the Elliott 401 it is possible to inhibit the writing on any of the tracks 1 to 6, and on the relay tracks as a group. This enables the orders and constants of a programme to be protected from mutilation by overwriting. Orders and numbers which are changed in the course of the programme must of course be placed on a track on which writing is not inhibited. This can be done in programmes constructed by the routine by coding the relevant auxiliaries $A_0 = 0.02$. All such orders and numbers will then be placed on track 6.

(e) *The use of repeated constants.* To speed up a programme, constants which are required in a number of contexts can be repeated in more than one initially assigned store location. The machine will select the most advantageous location for each reference.

Outline of machine operations

The routine is divided into five sections: Input, A, B, C, and Output.

Input. The provisional orders and auxiliaries are read into the store by a special input routine.

Section A. Each order is examined in turn and any additional information which is deducible from the order itself and its relation to other orders is planted in the auxiliary. Various tests for definite or probable errors and inconsistencies are applied at this stage.

Section B. The parity conditions planted in Section A and those (if any) imposed by the programmer are collated, and definite parities are assigned. If any inconsistency arises, an indication of this is given.

Section C. Final addresses are assigned to the A_0, A_2, and A_1 of each order so as to conform with the conditions now contained in the auxiliary, with due regard to the requirements of optimum programming and the positions still vacant in the store.

Output. A register of occupied store positions is printed and an order tape produced in standard form. The latter can be read back to check for punching errors.

General Remarks

The routine now has two forms. The short form can handle programmes requiring up to 371 store locations. The long form can deal with programmes requiring up to 1023 store locations. In either form the orders and numbers can be distributed over any desired tracks by means of the sequence controls, and parts of any track can be reserved for data, sub-routines, etc.

The routine is inevitably of considerable complexity. It occupies about 1800 locations (including constants and working store positions), and contains 245 discrimination orders. In the short form the routine itself is placed on fourteen of the relay tracks, and the provisional orders and auxiliaries are stored on the non-relay tracks. In the long form, the provisional orders and auxiliaries are stored in the relay tracks and the processing is carried out by stages, with the relevant part of the routine on tracks 1 to 6.

In order to avoid parts of a track or tracks being packed too tight early in a programme, in which case the timing of later parts would suffer, provision is made for introducing a spacing parameter λ', which has the effect of passing over the first λ' vacant positions during the allocation of each address in section C. λ' is specified by the programmer at the same time that he indicates the sequences in which the programme is to be processed. It can have a different value for each sequence.

A register of occupied store positions—the analogue of the track diagrams used in hand programming to mark off used locations—is kept for the 22 tracks (excluding track 0) and track V separately. The register is held in 92 consecutive store positions, four words representing one track.

Performance of the routine

The performance of automatic programming routines is sometimes assessed by comparing the speed of programmes constructed by the machine with that of the corresponding programmes written serially (that is, with A_2 of each order equal to $A_0 + 1$). The increase in computing time resulting from the latter procedure depends very much on the type of programme as well as on the machine; for the 401 an increase of five to ten times can easily result. This, however, does not in

our opinion give a meaningful comparison, since no competent programmer would use a non-serial machine in this manner. In any event for many machines, of which the 401 is one, parity conditions prevent strictly serial arrangement. Our aim in writing the automatic programming routine was that it should construct programmes whose timing would be no worse than that which a reasonably competent programmer would produce by hand. This object it is believed has been achieved.

For example, detailed examination of two programmes constructed by the machine—one for analysing the data of randomized block experiments and the other for probit analyses, each occupying three tracks—showed that in all the inner loops the timing was as good as that which hand programming would have produced. In another three-track programme for calculating the means, variances, covariances and correlations of multivariate data, the timing was actually better than a similar programme previously hand-written. In this latter case, the saving in time was mainly due to the insertion of timing delays which had been omitted in the original programme between the A_2 and A_1 of certain types of orders. We have found that even the most experienced programmers make occasional slips of this type, which of course are not revealed by machine tests.

In addition to producing programmes which are adequately speedy in operation it has been found that the routine substantially reduces the amount of machine time required for testing programmes. Even allowing for the machine time consumed in the actual construction of programmes, there is a very substantial net saving.

Conclusion

The short form of the routine was originally written early in 1955 without the special facilities (b), (c) and (d) above. Experience in its use indicated the need for greater capacity, so that programmes of more than three tracks could be dealt with in one operation, and for the above facilities. In its original form the routine has proved very successful and has substantially reduced the labour of writing long programmes. Equally important, it has been found that the finished programmes contain far fewer errors. This relative freedom from error is partly attributable to the elimination of certain types of error entirely (for example, overwriting) and to the indications provided by the routine of other probable errors, but it is also a reflection of the fact that the programmer is freed from having to attend simultaneously to a mass of troublesome detail and can consequently give more adequate attention to matters of real substance.

REFERENCES

1. GORDON, B. An optimizing program for the IBM 650. *Journal of the Association for Computing Machinery* 3, (1956) 3–5.
2. LIPTON, S. A note on the electronic computer at Rothamsted. *Math. Tables and Other Aids to Computation* 9, (1955) 69.