The Genesis of Microprogramming

M. V. WILKES

Two papers on microprogramming, one from 1951 and one (written with J. B. Stringer) from 1952 are reprinted, along with a retrospective introduction by the author.

*Categories and Subject Descriptors: B.1 [**Control Structures and Microprogramming**]; C.0 [**General**]—instruction set design; K.2 [**History of Computing**]—hardware*
General Terms: Design
Additional Key Words and Phrases: microprogramming, EDSAC, Cambridge University, Manchester University

Foreword

The ideas underlying the modern stored-program computer were formulated by a small group of mathematicians and engineers who came together at the Moore School of Electrical Engineering in Philadelphia in 1945 and early 1946. These ideas were spread partly by word of mouth, partly by written material, and partly by a course of lectures organized by the Moore School in the late summer of 1946. No one who attended these lectures had any doubt that a stored-program computer could be successfully constructed. However, neither in the lectures nor at any other time was a complete schematic diagram at gate level for such a computer presented. Every group that set out to build a computer had to evolve a design for itself, and it is not surprising that there was much variety in the approaches they took.

The computer with which I was concerned was the EDSAC, built at the Mathematical Laboratory (now the Computer Laboratory) of Cambridge University.

Author's Address: Digital Equipment Corporation, 77 Reed Road, Hudson, MA 01749.
© 1986 AFIPS 0164-1239/86/020116–126$01.00/00

The EDSAC was serial in operation. Its main units were the *memory*, the *arithmetic unit*, the *input unit*, and the *output unit*, together with a unit known as the *main control*. These units were interconnected by a bus along which numbers or instructions could be passed. The memory was capable of reading and writing, and the arithmetic unit could perform various operations such as addition, subtraction, and multiplication. The units were conditioned to perform the operation required of them by waveforms emanating from the main control. These waveforms were static; that is, they retained their state—high or low as the case might be—for the duration of the instruction being executed. In addition, there were nonstatic waveforms responsible for sequencing; some of these were generated in the main control, and some were generated locally in the units concerned.

The EDSAC had an ultrasonic memory consisting of 32 mercury tanks, each tank having an electronic unit associated with it. The electronic units were all identical, and the memory thus had a simple and regular structure. By contrast, the control and arithmetic circuits taken together were complex and irregular. It would obviously be an improvement if the whole computer could be made as simple and regular as the memory.

I had originally taken it for granted that a serial machine would contain a smaller amount of equipment than a parallel one. Largely through contact with Julian Bigelow, who was the designer of a ma-

chine under construction at the Institute for Advanced Study in Princeton, I had come to realize that this was not necessarily the case and, furthermore, that the arithmetic circuits of a parallel machine had the advantage of being very regular. However, the control circuits, while simpler than in a serial machine, were still irregular in structure. I felt there must surely exist some principle on which a control unit with a regular structure could be designed, and I consciously set myself the task of discovering what it was. The problem was uppermost in my mind in the summer of 1950, and I remember worrying over it during the enforced leisure provided by a voyage by sea across the Atlantic in July of that year.

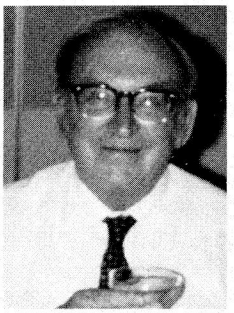
All new developments have their antecedents. There was one slight element of systematic design in the main control of the EDSAC. The digits comprising the operation code of an instruction were decoded by means of a diode tree and were then re-encoded in a different manner to provide the static waveforms supplied to the operational units. The re-encoding was done by means of a diode matrix. It seemed to me that here was a clue. Something similar was needed that could generate sequencing waveforms.

While I was in the United States, I saw the Whirlwind computer then under construction at MIT. Whirlwind was a parallel machine, and the designers had gone a long way toward rationalizing the control function. Each instruction—with the exception of multiplication, which had its own control unit—took place in exactly eight steps. The waveforms used to perform the steps were generated in a central control unit, and their shapes were determined by the positions at which diodes were placed in a control matrix. The elegance of this design appealed to me greatly; it was enhanced by the fact that the diodes were arranged physically, as well as electrically, in the form of a matrix. There was, however, the limitation that the same number of time steps had to be allocated to each instruction and that the sequence of waveforms generated for a given instruction was invariable. The system could not, therefore, be used for operations such as multiplication and division in which some of the steps must be conditional.

Eventually I realized that the objective that I had set myself could only be achieved by giving the control unit the full flexibility of a programmed computer in miniature; this led me to the idea of microprogramming. As soon as the idea crystallized in my mind, I gave a short impromptu presentation to my colleagues in the Mathematical Laboratory. Sometime later I was invited to speak at a conference being organized to mark the inauguration of the Ferranti Mark I computer in Manchester, and I took advantage of the opportunity thus provided to make a public presentation. The paper I gave is reprinted here. It will be noted that the paper deals not only with microprogramming, but more generally with the structure of a computer. It contains the first advocacy of what later became known as bit-slicing. Also reprinted is a follow-up paper I published a little over a year later jointly with J. B. Stringer, who had helped me to work out the details of a sample microprogram.

Although I stated that I had been asked to open a discussion rather than present a paper, in the published proceedings of the conference no such discussion is recorded as having taken place. I certainly cannot remember any comments having been made on the subject of microprogramming. Representatives of some half a dozen British computer projects were among the 169 people listed as having been present at the conference, but it is probably true to say that, for the most part, they were too fully occupied with immediate and pressing problems to be interested in philosophical discussions about how computers should be designed. It must be remembered that in the summer of 1951 only a handful of stored-program computers were actually in operation.

Interest in microprogramming was slow to build up. I think that this was because the engineers of the day found the concept hard to understand and because they did not feel as forcibly as I did the need for simplicity and regularity in the design of the control circuits of a computer. From the practical point of view, a major inhibiting factor was the difficulty of designing an efficient read-only memory using vacuum tubes. The coming of transistors, with their lower



Maurice V. Wilkes was formerly professor of computer technology at Cambridge University, England, and was head of the Computer Laboratory. He is now with the Digital Equipment Corporation in Hudson, Massachusetts. He is a Fellow of the Royal Society, a Distinguished Fellow of the British Computer Society, a Foreign Honorary Member of the American Academy of Arts and Sciences, and a Foreign Associate of both the National Academy of Sciences and the National Academy of Engineering. He delivered the ACM Turing Lecture in 1967. AFIPS presented him the Harry Goode Memorial Award in 1968 and the Eckert-Mauchly Award in 1980.

impedance, made read-only memories much more practicable. A breakthrough occurred when IBM decided to make use of microprogramming in the System/360 announced in 1964. I have always understood that IBM was led to contemplate taking this decision through the influence of W. S. Elliott, who was the founder of the IBM laboratory at Hursley. Elliott and I had been close friends since our student days, and he was in the audience when I made my presentation at the conference in Manchester.

I surveyed the growth of interest in microprogramming in a paper published in the first issue of *Computing Surveys* (1969). The reference to this paper is in the Bibliography, together with references to other early papers of which I was author or part author. There is also a reference to a paper in which I discuss some of Babbage's work and its relation to microprogramming. Chapters 16 and 17 of the last reference contain some material on microprogramming.

BIBLIOGRAPHY

Wilkes, M. V., W. Renwick, and D. J. Wheeler. 1958. The design of the control unit of an electronic digital computer. *Proc. IEE*, Vol. 105B, p. 121.

Wilkes, M. V. 1959. Microprogramming. *Proc. EJCC*, 1958, p. 18.

Wilkes, M. V. 1969. The growth of interest in microprogramming. *Computing Surveys 1*, p. 139.

Wilkes, M. V. April 1981. The design of a control unit—Reflections on reading Babbage's notebooks. *Annals of the History of Computing*, Vol. 3, No. 2, pp. 116–120.

Wilkes, M. V. 1985. *Memoirs of a Computer Pioneer*. Cambridge, MIT Press.

The Best Way to Design an Automatic Calculating Machine[1]

M. V. WILKES

I would like to begin by adding my congratulations to the many others which have been received by Professor Williams, Manchester University and Ferranti Ltd., on the construction of the machine which has just been inaugurated. In the face of this beautifully engineered machine, the title I have chosen for my opening remarks in this discussion may sound a little impertinent. But, as Dr. Kilburn remarked yesterday, the designer of an electronic calculating machine must continually take decisions, and he does not know when he takes them whether they are right or wrong. I might put it by saying that in a mathematical sense the solution to the problem of designing an electronic calculating machine is unstable. Two similar groups of engineers with similar backgrounds and assisted by similar groups of mathematicians will, if working independently, produce quite different machines. Moreover, the machines finally built will depend on the scale on which the projects are conducted, the experience and background of the teams, and the state of technical developments at the time. The last item is important since new developments in electron tubes, or in non-linear devices of the germanium type, might well affect even so fundamental a decision as the choice between the serial or parallel modes of operation for the machine. It is desirable, therefore, to keep under review the considerations which underlie the design of calculating machines and to try to examine them in the light of general principles as well as of current technical developments. I am aware that in doing this one is in danger of saying things which are sufficiently obvious without being said, but I am in the fortunate position of having been asked to open a discussion rather than to give a paper. I shall not, therefore, attempt to present a logical thesis but shall allow myself to raise issues rather than settle them.

I think that most people will agree that the first consideration for a designer at the present time is how he is to achieve the maximum degree of reliability in his machine. Amongst other things the reliability of the machine will depend on the following:

(a) The amount of equipment it contains.
(b) Its complexity.
(c) The degree of repetition of units.

By the complexity of a machine I mean the extent to which cross-connections between the various units obscure their logical inter-relation. A machine is easier to repair if it consists of a number of units connected together in a simple way without cross-connections between them; it is also easier to construct since different people can work on the different units without getting in each other's way.

As regards repetition I think everyone would prefer to have in a particular part of the machine a group of five identical units rather than a group of five different units. Most people would prefer to have six identical

[1] "The Best Way to Design an Automatic Calculating Machine" was presented at the Manchester University Computer Inaugural Conference, July 1951, published by Ferranti Ltd. It is reprinted here with permission.

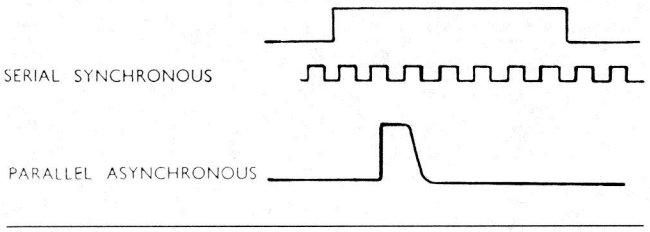


Figure 1

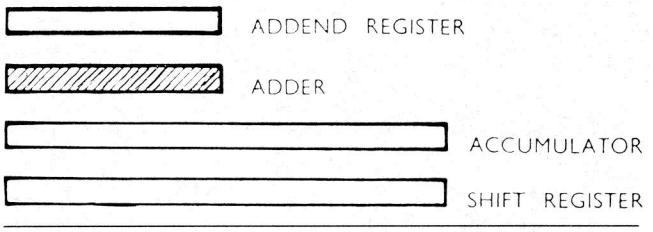


Figure 2

units rather than five different units. How far one ought to be prepared to go in the direction of accepting a greater quantity of equipment in order to achieve repetition is a matter of opinion. The matter may be put as follows. Suppose that it is regarded as being equally desirable to have a particular part of the machine composed of a group of n different units, or composed of a group of kn identical units, all the units being of similar size. What is the value of k? My conjecture is that $k > 2$. I should say that I am thinking of a machine which has about 10 groups of units and that n is approximately equal to 10.

The remarks I have just made are of general application. I will now try to be more specific. If one builds a parallel machine one has a good example, in the arithmetical unit, of a piece of equipment consisting of identical units repeated many times. Such an arithmetical unit is, however, much larger than that in a serial machine. On the other hand I think it is true to say that the control in a parallel machine is simpler than in a serial machine. I am using the word *control* here in a very general sense to include everything that does not appertain to the store proper (i.e., it includes the access circuits) or to the registers and adders in the arithmetical unit. That the control can be simpler in a parallel machine may I think be seen by comparing the waveforms which must be produced in order to effect the transfer of a number from one register to another in a serial synchronous machine and in a parallel asynchronous machine. These are the two extreme cases. In the case of a serial synchronous machine the waveform must rise at some critical moment relative to the clock and must fall at another critical moment, and its edges must be sharp. In a parallel asynchronous machine all that is needed is a single pulse whose time of occurrence, length, and shape are all non-critical (see Fig. 1).

The arithmetical unit of a parallel machine is often shown diagrammatically as in Fig. 2.

At the beginning of a multiplication the multiplier is placed in the right-hand half of the accumulator register. The right-hand half of the shift register may be dispensed with if shifting is done in two stages. Showing the right-hand half of the accumulator as a separate register we then have the diagram of Fig. 3.

We are thus led to think of an arithmetical unit composed of a number of standard units each containing four flip-flops (one belonging to each of four registers) together with an adder. Gates would be provided to make possible the transfer of numbers from one register to another, through the adder when necessary. These transfers would be effected by pulsing one or more of a set of wires emerging from the arithmetical unit.

It is also necessary to have registers in the control of a machine. These, with the names given to them respectively in the Manchester machine and in the E.D.S.A.C., are as follows:

Register for holding the address of the next order due to be executed (control, or sequence control tank).

Register holding order at present being executed (current instruction register, or order tank).

Register for counting the number of steps in a multiplication or shifting operation (not needed with the fast multiplier on the Manchester machine, timing control tank in the E.D.S.A.C.).

In addition the Manchester machine has a number of B registers.

If one B register is considered to be sufficient the parallel machine we are considering can use the same unit (containing four flip-flops and one adder) for the control registers as for arithmetical registers. In this way an extreme degree of repetition can be achieved.

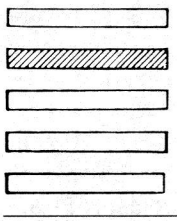


Figure 3

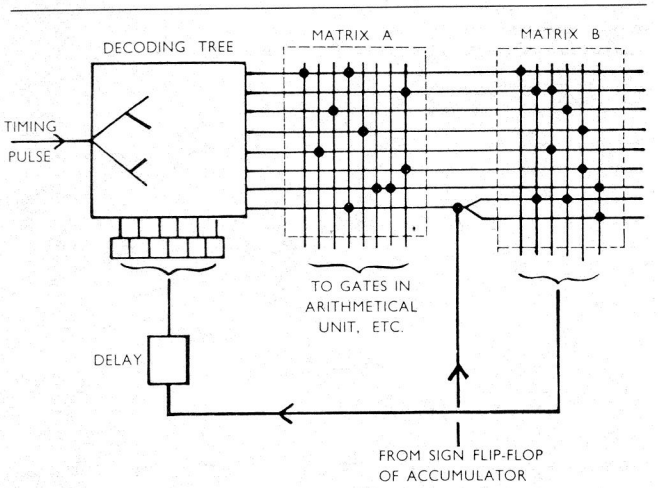


Figure 4

It remains to consider the control proper, that is, the part of the machine which supplies the pulses for operating the gates associated with the arithmetical and control registers. The designer of this part of a machine usually proceeds in an *ad hoc* manner, drawing block diagrams until he sees an arrangement which satisfies his requirements and appears to be reasonably economical. I would like to suggest a way in which the control can be made systematic, and therefore less complex.

Each operation called for by an order in the order code of the machine involves a sequence of steps which may include transfers from the store to control or arithmetical registers, or *vice versa*, and transfers from one register to another. Each of these steps is achieved by pulsing certain of the wires associated with the control and arithmetical registers, and I will refer to it as a "micro-operation." Each true machine operation is thus made up of a sequence or "micro-programme" of micro-operations.

Fig. 4 shows the way in which pulses for performing the micro-operations may be generated. The timing pulse which initiates a micro-operation enters the decoding tree and is routed to one of the outputs according to the number set on the register R. It passes into the rectifier matrix A and gives rise to pulses on certain of the output wires of this matrix according to the arrangement of the rectifiers. These pulses operate the gates associated with the control and arithmetical registers, and cause the correct micro-operation to be performed. The pulse from the decoding tree also passes into matrix B and gives rise to pulses on certain of the output wires of this matrix. These pulses are conducted, via a short delay line, to the register R and cause the number set up on it to be changed. The result is that the next initiating pulse to enter the decoding tree will emerge from a different outlet and will consequently cause a different micro-operation to be performed. It will thus be seen that each row of rectifiers in matrix A corresponds to one of the micro-orders in the sequence required to perform a machine operation.

The system as described would enable a fixed cycle of operations only to be performed. Its utility can be greatly extended by making some of the micro-orders conditional in the sense that they are followed by one of two alternative micro-orders according to the state of the machine. This can be done by making the output of the decoding tree branch before it enters matrix B. The direction the pulse takes at the branch is controlled by the potential on a wire coming from another part of the machine; for example, it might come from the sign flip-flop of the accumulator. The bottom row of matrix A in Fig. 4 corresponds to a conditional micro-order.

The matrix A contains sequences of micro-orders for performing all the basic operations in the order code of the machine. All that is necessary to perform a particular operation is that "micro-control" shall be switched to the first micro-order in the appropriate sequence. This is done by causing the function digits of the order to be set up on the first four or five flip-flops of the register R, zero being set on the others.

A control system designed in this way is certainly very logical in structure but two comments, slightly contradictory in their implications, might be made. In the first place it might be said that there is nothing very new about the arrangement since it makes use of flip-flops, gates, and mixing diodes which are the elements out of which any control is built. With this criticism I would agree. In fact, the controls of various machines now in existence or being constructed could no doubt be drawn in some way closely resembling Fig. 4. The other objection is that the scheme appears to be rather extravagant in equipment. This I think is not true, particularly if some departures from the precise form of Fig. 4 are allowed. I think that by starting with a logical layout one is likely to arrive at a final arrangement which is both logical and economical. Moreover, one is able to see at each stage what one is sacrificing in the way of logical layout in order to achieve economy and *vice versa*.

In order to get some idea of the number of micro-orders required I have constructed a micro-programme for a simple machine with the following orders: add, subtract, multiply (two orders, one for the multiplier, one for the multiplicand), right and left shift (any number of places), transfer from the accumulator to the store, conditional operation depending on the sign

of the number in the accumulator, conditional operation depending on the sign of the number in the *B* register (one *B* register is assumed), transfer from the store to the *B* register, input, and output. The micro-programme also provides for the preliminary extraction of the order from the store (Stage 1 in E.D.S.A.C. terminology). Only 40 micro-orders are required to perform all these operations.

The considerations involved in drawing-up a micro-programme resemble those involved in drawing-up an ordinary programme. The final details of the control are thus settled by a systematic process instead of by the usual *ad hoc* procedures based on the use of block diagrams. Of course, sound engineering would be necessary to produce designs for the decoding tree and the matrices which could be used for any desired micro-programme by arranging the rectifiers suitably in the matrices. One important advantage of this method of designing the control is that the order code need not be decided on finally until a late stage in the construction of the machine; it would even be possible to change it after the machine had been put into operation simply by rewiring the matrices.

If desired some of the micro-orders can be made conditional in their action as well as (or instead of) conditional as regards the switching of micro-control. This can be done by making the output of the decoding tree branch before it enters matrix *A*. I doubt if much economy can be achieved this way and if it is done to any extent the advantage that micro-programming resembles ordinary programming is lost. Other variants of the scheme as I have described it will no doubt occur to you.

The matrices may be regarded as very high-speed stores holding fixed information. If they could be replaced by an erasable store to which information could be transferred from the main store of the machine when required we should have a machine with no fixed order code; the programmer would, in fact, be able to choose his order code to suit his own requirements and to change it during the course of the programme if he considered it desirable. Such a machine would have a number of fascinating possibilities but I doubt whether, in view of the amount of equipment it would doubtless involve, its construction could be justified.

Micro-Programming and the Design of the Control Circuits in an Electronic Digital Computer[2]

M. V. WILKES AND J. B. STRINGER

1. Introduction

Experience has shown that the sections of an electronic digital computer which are easiest to maintain are those which have a simple logical structure. Not only can this structure be readily borne in mind by a maintenance engineer when looking for a fault, but it makes it possible to use fault-locating programmes and to test the equipment without the use of elaborate test gear. It is in the control section of electronic computers that the greatest degree of complexity generally arises. This is particularly so if the machine has a comprehensive order code designed to make it simple and fast in operation. In general, for each different order in the code some special equipment must be provided, and the more complicated the function of the order the more complex this equipment. In the past, fear of complicating unduly the control circuits of the machines has prevented the designers of electronic machines from providing such facilities as orders for floating-point operations, although experience with relay machines and with interpretive subroutines has shown how valuable such orders are. This paper describes a method of designing the control circuits of a machine which is wholly logical and which enables alterations or additions to the order code to be made without *ad hoc* alterations to the circuits. An outline of this method was given by one of us (M. V. W.) at the Conference on Automatic Calculating Machines at the University of Manchester in July 1951 (1).

The operation called for by a single machine order can be broken down into a sequence of more elementary operations; for example, shifting a number in the accumulator one place to the right may involve, first, a transfer of the number to an auxiliary shifting

[2] "Micro-Programming and the Design of the Control Circuits in an Electronic Digital Computer" was published in the Proceedings of the Cambridge Philosophical Society, Volume 49, 1953, pages 230–238. It is reprinted here with permission.

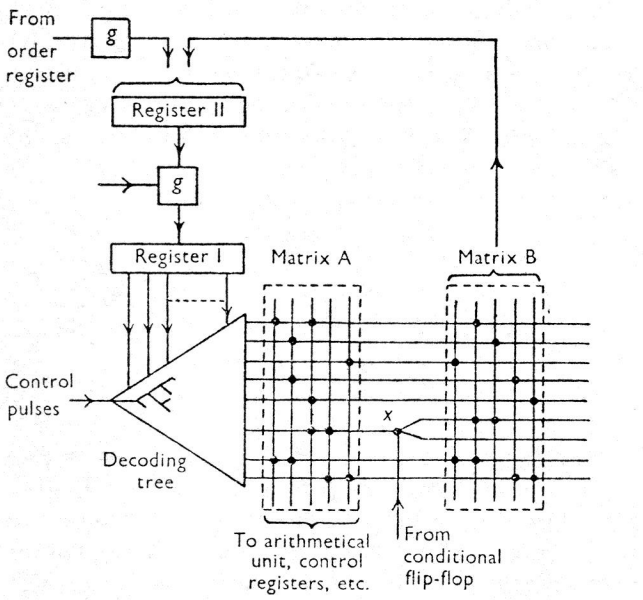


Figure 1. Micro-control unit.

register, and secondly, the transfer of the number back to the accumulator along an oblique path. These elementary operations will be referred to as *micro-operations*. Basic machine operations, such as addition, subtraction, multiplication, etc., are thought of as being made up of a *micro-programme* of micro-operations, each micro-operation being called for by a *micro-order*. The process of writing a micro-programme for a machine order is very similar to that of writing a programme for the whole calculation in terms of machine orders.

For the method to be applicable it is necessary that the machine should contain a suitable permanent rapid-access storage device in which the micro-programme can be held—a diode matrix is proposed in the case of the machine discussed as an example below—and that means should be provided for executing the micro-orders one after the other. It is also necessary that provision should be made for conditional micro-orders which play a role in micro-programming similar to that played by conditional orders in ordinary programming.

Since the only feature of the machine which has to be designed specially for any particular set of machine orders is the configuration of diodes in the matrix, or the corresponding configuration in whatever equivalent device is used, there is no difficulty in making changes to the order code of the machine if experience shows them to be desirable; in fact, the design of the machine in the first place can be carried out completely without a firm decision on the details of the order code being taken, as long as care is taken to provide accommodation for the greatest number of micro-orders that are likely to be required. It would even be possible to have a number of interchangeable matrices providing for different order codes, so that the user could choose the one most suited to his particular requirements.

2. Description of the Proposed System

The system will be described in relation to a parallel machine having an arithmetical unit designed along conventional lines. This will contain a set of registers and an adder together with a switching system which enables the micro-operations in the various machine orders to be performed. Some of the micro-operations will be simple transfers of a number from one register to another with or without shifting of the number one place to the left or the right, while others will also involve the use of the adder. Any particular micro-operation can be performed by applying pulses simultaneously to the appropriate gates of the switching system. In certain cases it may be possible for two or more micro-operations to take place at the same time.

It will be convenient to regard the control system as consisting of two parts. A register is needed to hold the address of the next order due to be executed, and another to hold the current order while it is being executed, or at any rate during part of that time. Some means of counting the number of steps in a shifting operation or a multiplication must also be provided. One method of meeting these requirements is to provide a group of registers and an adder together with a switching system which enables transfers of numbers, with or without addition, to be made. This part of the control system will be called the *control register unit*. In any case the operations which need to be performed on the numbers standing in the control register unit during the execution of an order are, like the operations performed in the arithmetical unit, regarded as being made up of a sequence of micro-operations, each of which is performed by the application of pulses to appropriate gates.

The other part of the control system is concerned with control of the sequence of micro-orders required to carry out each machine order, and with the operation of the gates required for the execution of each micro-order. This will be called the *micro-control unit*; it consists of a decoding tree, two rectifier matrices and two registers (additional to those of the control register unit) connected as indicated in Fig. 1, which shows how the pulses used to operate the gates in the arithmetical unit and control register unit are generated. A series of control pulses from a pulse generator

are applied to the input of the decoding tree. Each pulse is routed to one of the output lines of the tree, according to the number standing in register I. The output lines all pass into a rectifier matrix A and the outputs of this matrix are the pulses which operate the various gates associated with micro-operations. Thus one input line of the matrix corresponds to one micro-order. The *address* of the micro-order is the number which must be placed in register I to cause the control pulse to be routed to the corresponding line. The output lines from the tree also pass into a second matrix B, which has its outputs connected to register II. This matrix has wired on it the address of the micro-order to be performed next in time so that the address of this micro-order is placed in register II. Just before the next control pulse is applied to the input of the tree a connexion is established between register II and register I, and the address of the micro-order due to be executed next is transferred into register I. In this way the decoding tree is prepared to route the next incoming control pulse to the correct output line. Thus application of pulses alternately to the input of the tree and to the gate connecting registers I and II causes a predetermined sequence of micro-orders to be executed.

It is necessary to have means whereby the course of the micro-programme can be made conditional on whether a given digit in one of the registers of the arithmetical unit or control register unit is a 1 or a 0. The means of doing this is shown at X in Fig. 1. A two-way switch, controlled by a special flip-flop called a *conditional flip-flop*, is inserted between matrix A and matrix B. The conditional flip-flop can be set by an earlier micro-order with any digit from any one of the registers. Two separate addresses are wired into matrix B, and the one which passes into register I, and thus becomes the address of the next micro-order, is determined by the setting of the conditional flip-flop.

Conditional micro-orders play the same part in the construction of micro-programmes as conditional orders play in the construction of ordinary programmes; apart from their obvious uses in micro-programmes for such operations as multiplication and division, they enable repetitive loops of micro-orders to be used.

If desired, two branchings may be inserted in the connexions between matrix A and matrix B, so that any one of four alternative addresses for the next micro-order may be selected according to the settings of two conditional flip-flops. Another possibility is to make the output from the decoding tree branch before it enters matrix A so that the nature of the micro-operation that is performed depends on the setting of the conditional flip-flop.

The micro-programme wired on to the matrices contains sections for performing the operations required by each order in the basic order code of the machine. To initiate the operation it is only necessary that control in the micro-programme should be sent to the correct entry point. This is done by placing the function digits of the order in the least significant part of register II, the other digits in this register being made zero. The micro-programme is constructed so that when this number passes into register I, control in the micro-programme is sent to the correct entry point.

The switching system in the arithmetical unit may either be designed to permit a large variety of micro-operations to be performed, or it may be restricted so as to allow only a small number of such operations. In a machine with a comprehensive order code there is much to be said for having the more flexible switching system since this will enable an economy to be made in the number of micro-orders needed in the micro-programme.

A similar remark applies in connexion with the degree of flexibility to be provided when designing the switching system for the control register unit. If the specification of the machine allows the same number of registers to be used in the arithmetical and control sections, the construction of these two sections may be identical except as far as the number of digits is concerned. In a new machine under construction in the Mathematical Laboratory, Cambridge, the registers are being constructed in basic units each containing five registers and an adder-subtractor together with the associated switching system. It is hoped that it will be possible to use identical units in the arithmetical unit and in the control register unit.

3. Example

An example will now be given to show the way in which a micro-programme can be drawn up for a machine with a single-address order code covering the usual operations. It is supposed that the arithmetical unit contains the following registers:

- A multiplicand register,
- B accumulator (least significant half),
- C accumulator (most significant half),
- D shift register.

The registers in the control register unit are as follows:

- E register connected to the access circuits of the store; the address of a storage location to which access is required is placed here,
- F sequence control register; contains address of next order due to be executed,
- G register used for counting.

It was assumed when drawing up the micro-programme that there was an adder-subtractor in the arithmetical unit with one input permanently connected to register D, and a similar adder-subtractor in the control register unit with one input permanently connected to register G. For convenience it was assumed that the switching systems in each case were comprehensive enough to provide any micro-operation required. It was further supposed that the arithmetical unit provided for 20 digits and that the numbers 0, 1 and 18 could be introduced at will into one of the registers or the adder of the control register unit. Two conditional flip-flops are used. All micro-operations including those involving access to the store are supposed to take the same amount of time. Reference will be made to this point in Section 4.

Table 1 gives the order code of the machine, and Table 2 the micro-programme. Each line of Table 2 refers to one micro-order; the first column gives the address of the micro-order, the second column specifies the micro-operations called for in the arithmetical unit of the machine, and the third column specifies the micro-operations called for in the control register unit. The fourth column shows which conditional flip-flop, if any, is to be set and the digit which is to be used to set it; for example, $(1)C_s$ means that flip-flop number 1 is set by the sign digit of the number in register C, while $(2)G_l$ means that flip-flop number 2 is set by the least significant digit of the number in register G. In the case of unconditional micro-orders columns 5 and 7 are blank and column 6 contains the address of the next micro-order to be executed. In the case of conditional micro-orders column 5 shows which flip-flop is used to operate the conditional switch and columns 6 and 7 give the alternative addresses to which control is to be sent when the conditional flip-flop contains a 0 or a 1 respectively.

Micro-orders 0 to 4 are concerned with the extraction of orders from the store. They serve to bring about the transfer of the order from the store to register E and then cause the five most significant digits of the order to be placed in register II with the result that control is transferred to one of the micro-orders 5 to 15, each of which corresponds to a distinct order in the machine order code. In this way the sequence of micro-orders needed to perform the particular operation called for is begun.

The way in which the various operations are performed can be followed from Table 2. In the section dealing with multiplication, it is assumed that numbers lie in the range $-1 \leqslant x < 1$ and that negative numbers are represented in the machine by their complements with respect to 2. It will be noted that the process of drawing up a micro-programme is very similar to that of drawing up an ordinary programme for an automatic computing machine and the problems involved are very much alike.

Table 1

Notation:

Acc	=	accumulator
Acc_1	=	most significant half of accumulator
Acc_2	=	least significant half of accumulator
n	=	storage location n
$C(X)$	=	contents of X (X = register or storage location)

Order	Effect of order
$A\ n$	$C(Acc)+C(n)$ to Acc
$S\ n$	$C(Acc)-C(n)$ to Acc
$H\ n$	$C(n)$ to Acc_2
$V\ n$	$C(Acc_2)\cdot C(n)$ to Acc, where $C(n)\geqslant 0$
$T\ n$	$C(Acc_1)$ to n, 0 to Acc
$U\ n$	$C(Acc_1)$ to n
$R\ n$	$C(Acc)\cdot 2^{-(n+1)}$ to Acc
$L\ n$	$C(Acc)\cdot 2^{n+1}$ to Acc
$G\ n$	If $C(Acc)<0$, transfer control to n; if $C(Acc)\geqslant 0$, ignore (i.e. proceed serially)
$I\ n$	Read next character on input mechanism into n
$O\ n$	Send $C(n)$ to output mechanism

4. The Timing of Micro-Operations

The assumption that all micro-operations take the same length of time to perform is not likely to be borne out in practice. In particular in a parallel machine it may not be possible to design an adder in which the carry propagation time is sufficiently short to enable an addition to be performed in substantially the same length of time as that taken for a simple transfer. It will be necessary, therefore, to arrange that the wave-form generator feeding the decoding tree should, when suitably stimulated by a pulse from one of the outputs from matrix A, supply a somewhat longer pulse than that normally required. Other operations may take many times as long to perform as an ordinary micro-order; for example, access to and from the store (particularly if a delay store is used) and operation of the input and output devices of the machine. The sequence of operations in the micro-programme must therefore be interrupted. One way of doing this is to prevent pulses from the wave-form generator reaching the decoding tree during the waiting period. This method, although quite feasible, appears to involve just the kind of complication which the present system is designed to avoid. A more attractive system is to make the machine wait on a conditional micro-order which transfers control back to itself unless the associated conditional flip-flop is set. Setting of this flip-flop takes place when the

operation is completed; and control then goes to the next micro-order in the sequence. The machine is thus in a condition of "dynamic stop" while waiting for the operation to be completed. This system has the advantage that no complication is introduced into the units supplying the wave-forms to the decoding tree and that the control equipment required is similar to that already provided for other purposes.

Table 2

Notation: *A*, *B*, *C*, . . . stand for the various registers in the arithmetical and control register units (see Section 3 of the text). "*C* to *D*" indicates that the switching circuits connect the output of register *C* to the input of register *D*; "(*D* + *A*) to *C*" indicates that the output of register *A* is connected to the one input of the adding unit (the output of *D* is permanently connected to the other input), and the output of the adder to register *C*.

A numerical symbol *n* in quotes (e.g., "*n*") stands for the source whose output is the number *n* in units of the least significant digit.

		Arithmetical unit	Control register unit	Conditional flip-flop Set	Conditional flip-flop Use	Next micro-order 0	Next micro-order 1
	0		*F* to *G* and *E*			1	
	1		(*G* + "1") to *F*			2	
	2		Store to *G*			3	
	3		*G* to *E*			4	
	4		*E* to decoder			—	
A	5	*C* to *D*				16	
S	6	*C* to *D*				17	
H	7	Store to *B*				0	
V	8	Store to *A*				27	
T	9	*C* to Store				25	
U	10	*C* to Store				0	
R	11	*B* to *D*	*E* to *G*			19	
L	12	*C* to *D*	*E* to *G*			22	
G	13		*E* to *G*	(1)C_s		18	
I	14	Input to Store				0	
O	15	Store to Output				0	
	16	(*D* + Store) to *C*				0	
	17	(*D* − Store) to *C*				0	
	18				1	0	1
	19	*D* to *B* (*R*)*	(*G* − "1") to *E*			20	
	20	*C* to *D*		(1)E_s		21	
	21	*D* to *C* (*R*)			1	11	0
	22	*D* to *C* (*L*)†	(*G* − "1") to *E*			23	
	23	*B* to *D*		(1)E_s		24	
	24	*D* to *B* (*L*)			1	12	0
	25	"0" to *B*				26	
	26	*B* to *C*				0	
	27	"0" to *C*	"18" to *E*			28	
	28	*B* to *D*	*E* to *G*	(1)B_l		29	
	29	*D* to *B* (*R*)	(*G* − "1") to *E*			30	
	30	*C* to *D* (*R*)		(2)E_s	1	31	32
	31	*D* to *C*			2	28	33
	32	(*D* + *A*) *to C*			2	28	33
	33	*B* to *D*		(1)B_l		34	
	34	*D* to *B* (*R*)				35	
	35	*C* to *D* (*R*)			1	36	37
	36	*D* to *C*				0	
	37	(*D* − *A*) to *C*				0	

* Right shift. The switching circuits in the arithmetic unit are arranged so that the least significant digit of register *C* is placed in the most significant place of register *B* during right shift micro-operations, and the most significant digit of register *C* (sign digit) is repeated (thus making the correction for negative numbers).

† Left shift. The switching circuits are similarly arranged to pass the most significant digit of register *B* to the least significant place of register *C* during left shift micro-operations.

5. Discussion

It will be seen that the equipment needed to execute a complicated order in the machine order code is of the same form as that required for a simple one, namely outlets from the decoding tree and diodes in the matrices. Quite complicated orders can, therefore, be built into the machine without difficulty. In particular, arithmetical operations on numbers expressed in floating binary form and other similar operations can be micro-programmed and it is found that they do not involve very large numbers of micro-orders. For example, a micro-programme providing for the floating-point operations of addition, subtraction, and multiplication needs about 70 micro-orders. The switching system in the arithmetical unit must, of course, be designed with these operations in view. The decoding tree and matrices of a parallel machine with 40 digits in the arithmetical unit and provision for 256 micro-orders would only amount to about 15% of the total equipment in the machine, so that it appears that such a machine can well be provided with built-in facilities of considerable complexity.

The number of micro-orders needed in a complicated micro-programme can sometimes be reduced by making use of what might be called *micro-subroutines*. For example, when two numbers have to be added together in a floating binary machine, some shifting of one of them is usually necessary before the addition can take place. By making the micro-orders for this shifting operation serve also when a multiplication is called for, considerable saving is effected.

Four registers is the bare minimum needed in the arithmetical unit in order to enable the basic arithmetical operations to be performed. If any extension or refinement of the facilities provided is required, it may be necessary to increase the number of registers. For example, four registers are not sufficient to enable a succession of products to be accumulated without the transfer of intermediate results to the store, since the accumulator must be clear at the beginning of a multiplication. The addition of one register enables the accumulation of products to be provided for in the micro-programme. If this register is associated with the outlet from the store, it also enables some of the waiting time for storage access to be eliminated. To do this the micro-programme is arranged to call for a number from the store as soon as it is known that the number will be required and to continue with other necessary micro-operations before finally proceeding to use the number. The "dynamic stop" would occur just before the number is required for use. Another way of saving time is to arrange, in the case of those orders which permit it, for the next order to be extracted from the store before the operation currently being performed has been completed.

The minimum number of registers required in the control register unit of the machine for the simplest mode of operation is three. If extra registers are provided facilities similar to those provided by the B-lines in the machine at Manchester University could be included in the micro-programme.

6. Micro-Programming Applied to Serial Machines

All the discussion so far has been with reference to parallel machines because the technique described in this paper is most adapted to that type of machine. It is, however, possible to design a serial machine along the same lines. In a parallel computer with an asynchronous arithmetical unit every gate requires only one kind of wave-form to operate it and the timing of that wave form is not critical. In a serial machine, on the other hand, different gates require different wave-forms and the same gate may require different wave-forms at different times; further, all these wave-forms must be critically timed. These complications may be handled by including in the micro-control unit a third matrix, C, for selecting the appropriate wave-form for each micro-order. The main wave-form, routed by the decoding tree and matrix A, opens a gate which is fed by a wave-form selected by matrix C. This enables a wave-form of correct duration to be applied to any selected gate in the arithmetical or control sections of the machine.

The authors wish to express their thanks to Mr. A. L. Freedman and Mr. W. Renwick for assisting them in clarifying a number of points, and to Prof. D. R. Hartree, F.R.S., for his generous help with the preparation of the paper.

REFERENCE

(1) Wilkes, M. V. *Report of Manchester University computer inaugural conference, July* 1951 (Manchester, 1953).