EDSAC 2

MAURICE V. WILKES

EDSAC 2, which came into operation early in 1958, was designed by the team that had successfully built and operated EDSAC 1, and embodied the experience obtained with that machine. EDSAC 2 was the first computer to have a microprogrammed control unit, and it established beyond doubt the viability of microprogramming as a basis for computer design — this in spite of the fact that vacuum tubes were far from ideal for the purpose.

At the mechanical level of organization, EDSAC 2 was packaged in a bit-sliced manner, with interchangeable plug-in units. This method of packaging was well matched to the vacuum tube technology of the period, and its expected advantages — arising from the replication of units — were fully realized.

The article gives a description of the principal hardware features of EDSAC 2.

The year 1950 saw the beginning of much discussion in the Mathematical Laboratory (later renamed the Computer Laboratory) of the University of Cambridge as to what the future constructional plans of the laboratory were to be. Experience with EDSAC 1 on real problems had demonstrated, beyond doubt, the essential soundness of the stored-program concept and the ability of available technology to implement it. There was some concern on the score of reliability, but no one questioned the conclusion that what was wanted was a better stored-program computer, not a computer of some different kind. This was also the experience of other groups in the US and the UK where early stored-program computers had been built.

This article describes the principal hardware features of EDSAC 2 (Figure 1), which came into operation early in 1958 and was eventually closed down on November 1, 1965. Background information on the project and the way it developed will be found in my *Memoirs*.[1]

Origin of the design concepts

In July 1951 I was invited to speak at a conference held in Manchester to mark the inauguration of the Ferranti Mark I computer. I took advantage of the invitation to set forth some of the conclusions I had come to as a result of the experienced gained from EDSAC 1. I said, "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."[2] I went on to say that, among other things, the reliability of a machine will depend on the amount of equipment it contains, its complexity, and the degree of repetition of units. This led me to advocate a parallel arithmetic unit organized on what later came to be called the bit-sliced principle. The arithmetic circuits were to be divided into physical units or chassis, each of which contained one stage of the adder and one flip-flop from each of the various registers. I pointed out that the control registers, along with the adder associated with them, could be organized in a similar manner. I then went on to introduce the concept to which I gave the name "microprogramming."

Around the end of 1951 or early in 1952, William Renwick and I, with the general agreement of our colleagues, decided to investigate the possibility of building a computer along the above lines. Renwick's views were important since it was on him, as chief engineer of the laboratory, that the main load of detailed design would fall if, as we hoped, the project went ahead. David Wheeler left the laboratory at this juncture to take up a temporary faculty appointment at the University of Illinois and did not return until the late summer of 1953. Some of the main decisions about the register structure of the machine were, therefore, taken in his absence. However, he returned in time to be fully responsible for the design of the instruction set, including the floating-point format and the programming environment.

Technical problems

We felt that we had enough experience to design a parallel arithmetic unit that would be both reliable and fast, although there were a number of engineering design problems which caused concern at the period in question. One was that of avoiding pattern sensitivity in the circuits. An adder, for example, would be found to work perfectly with most input streams, but would give trouble with streams corresponding to certain bit patterns. Pattern sensitivity was a result of memory in the circuits arising, in particular, from the use of capacitors for coupling or other purposes. An important advantage of a parallel design was that the registers could be DC coupled, so that there would be no coupling capacitors that might introduce pattern sensitivity. I

1058-6180/92/1000-0049$03.00 © 1992 IEEE

Figure 1. A program-test session with EDSAC 2, with users standing in line to run their programs.

was personally almost paranoid on that issue and would not even contemplate putting small speedup capacitors across the resistors in the flip-flops. In this I was much influenced by Julian Bigelow, the designer of the computer built at the Institute for Advanced Study in Princeton.

Imperfect switching by gates, leading to what was known as *break-through*, gave rise to further problems for the designer, who had always to be worrying about how to keep the baseline of the waveforms clean.

The problem of designing and maintaining early computers was much aggravated by the lack of stability in the circuit parameters. This was partly, but only partly, due to a steady loss of emission in the vacuum tubes. Equally important were changes with time in the values of the resistors which were, at that period, made of a molded composite material. The value of a resistor would decrease with time, particularly if it were fully loaded. It was necessary to design the circuits with large operating margins and to pay special attention to the stabilization of power supplies.

The problem of the memory

Although the design of a fast parallel arithmetic unit presented no serious problem, there was by contrast grave difficulty in seeing how it could be matched with a memory of compatible speed. In fact, in early 1952 there were only two forms of high-speed memory that could be said to be in any sense out of the research stage. One was the mercury (ultrasonic) memory that we had used in EDSAC 1 and the other was the Williams tube memory. The latter, in a parallel form, was capable of sufficiently high-speed operation. However, there were difficulties both in acquiring high-quality cathode-ray tubes and in designing analog deflection circuits that would operate efficiently and reliably. We had no experience with cathode-ray tube storage and, since we felt that this form of storage would eventually be displaced by something better, we were not anxious to become involved. On the other hand, we thoroughly understood and had confidence in the mercury memory. We felt that there was no reason why a parallel mercury memory — with one mercury tank to correspond to each digital position in the word — should not be capable of operation in a rugged and reliable manner. However, even if we used shorter tanks and a higher pulse rate than in EDSAC 1, the speed of the memory would be far below that of the arithmetic unit. Since we were stressing reliability rather than speed, we decided to proceed with development work on a mercury memory, but with the hope that something better would soon become available. Fortunately, core memory came at the right time.

I was present at MIT in August 1953 when the first core memory was fitted, with dramatic success, to the Whirlwind computer. When I returned to the UK, we took the decision to abandon further work on a mercury memory and work on a core memory instead. By good fortune, Mullard Limited, who were already manufacturers of ceramic magnetic materials, were interested in working with us on the development and testing of cores with a square hysteresis loop. By the beginning of 1953, we saw our way clearly ahead, and the design and construction of EDSAC 2 as eventually built may be said to have begun at that time.

Core memory

The core memory for EDSAC 2 was designed by Renwick on a word-organized basis, using cores manufactured by Mullard in the UK.[1] In a word-organized memory, a separate wire is threaded through each of the cores — 40 in number in the case of EDSAC 2 — used to store the digits of each word in the memory. There are thus as many wires as there are words in the memory. When one of the wires is pulsed, the digits of the corresponding word are read out in parallel. The incoming address digits must be decoded and an appropriate waveform generated to drive the selected word wire. In EDSAC 2, these functions were performed by a switch matrix of 32 × 32 ferrite cores, these cores being of a larger size than memory cores. Further information on word-organized memories will be found in my 1956 book.[3]

I would personally have preferred to adopt the system of memory organization developed at MIT and which became, in due course, generally adopted. However, Renwick was anxious to try his hand at a word-organized system, and it certainly worked very well for a memory of 1,024 words, which was what was required for EDSAC 2.

In addition to the main memory, EDSAC 2 had a read-only memory in which information was permanently wired. This was known as the *reserved store*, and was implemented by means of extra core planes driven by an extension to the switch matrix. These were, in fact, normal core planes, except that they had extra wires — which I will refer to as *priming* wires — threaded through some of the cores. When a current was passed through one of the priming wires, the cores through which that wire passed were magnetized in the direction corresponding to a 1. In a read operation, all cores in the word to be read were first set to 0. The priming operation was then performed and followed by a normal read operation. The primed cores delivered 1s and the unprimed cores delivered 0s. There were four priming wires; which of them was used in a particular read operation was determined by the two high-order bits in the address of the word being read. One plane of the read-only memory had, therefore, the same capacity as four planes of normal memory.

The reserved store contained 768 read-only words plus 64 normal words that could be both written and read. In the context of the times, the reserved store was an architectural innovation of greater importance than may be immediately apparent to the modern reader. It contributed significantly to the speed of a machine by eliminating the loading time for frequently used library subroutines, included input and output subroutines. Moreover, a programmer coming to EDSAC 2 from EDSAC 1 would find his memory management problems greatly eased, since these subroutines were permanently available and he did not need to allocate space for them in memory.

Microprogrammed control

The microprogrammed control was the most strikingly original feature of EDSAC 2, and demonstrated, beyond doubt, the practicability of this way of building the control unit of a major machine. However, it was something of a tour de force to implement a sufficiently fast read-only memory for the microprogram based on vacuum tubes. This was because none of the storage elements available for use in a read-only memory — namely, diodes, capacitors, and magnetic cores — were well matched to the high output impedance of a vacuum tube. Later, the coming of transistors was to transform the problem and make microprogramming much more attractive from both the technical and the economic points of view.

After evaluating the various alternatives, we decided to use magnetic cores for the microprogram memory. The cores were arranged in the form of a matrix and switched on the coincident current principle. The matrix contained 1,024 cores and each corresponded to a micro-operation in the microprogram.

The microprogrammed control was the most strikingly original feature of EDSAC 2, and demonstrated, beyond doubt, the practicability of this way of building the control unit of a major machine.

The cores used were 8 mm in diameter. Each carried two drive windings of 40 turns each, and a bias winding of 28 turns through which a steady current was passed. The matrix was driven by powerful vacuum tubes capable of passing a current of 150 mA through the drive windings. In the case of the selected core, the combined effect of the current passing through the two drive windings was sufficient to overcome the effect of the steady current passing through the bias winding and to reverse the direction of magnetization in the core.

There were about 80 sets of gates in the various parts of the machine that were controlled by the microprogram. Corresponding to each set of gates was a wire which will be referred to as a *gate wire*. If a particular set of gates needed to be operated when a certain microinstruction was executed, the corresponding gate wire was threaded (three times) through the core corresponding to that microinstruction. When the core was switched, an electromotive force of about 9 volts was induced in the wire and this was sufficient, without voltage amplification, to drive the gates and so cause the microinstruction to be executed. In addition to the gate wires, a further set of wires were threaded through the cores and their outputs used to determine the next core to be switched.

The need for conditional microinstructions was met by locating two or four cores at some of the nodes of the matrix instead of only one. At any time, all but one of these cores were biased off, according to the settings of two conditional flip-flops. By setting these flip-flops in advance, the microprogrammer could control which of two (or four) alternative microinstructions would be executed when the node was selected.

The above statement, namely that more than one core was located at some nodes of the matrix, must be interpreted in a logical sense. Physically, all cores were located at the nodes of a 32 × 32 matrix. The wiring was so arranged that, logically, there were nodes with 0, 1, 2, or 4 cores. Details of the way in which this was contrived, along with further details of the microprogram unit of EDSAC 2, will be found in a 1958 paper.[4]

The voltage induced in a wire which threaded a half-switched core — that is, a core on the same row or column as a selected core — was small but not zero. As a result, there was a danger that the voltage induced in a gate wire which did not thread the switched core, but which threaded a number of half-switched cores, would be sufficient to cause spurious operation of the gate or gates to which the wire was connected. Advantage was taken of the fact that micro-operations could be allocated to individual cores in an arbitrary fashion to reduce this danger. It was laid down that a gate wire should never need to be threaded through more than four cores situated on any one row or any one column of the matrix. The total number of contributions from half-switched cores was therefore limited to eight. The allocation of microinstructions to cores in such a way as to meet this requirement was done by means of a program run on EDSAC 1. This must have been one of the earliest examples of design automation in which one computer is used to assist with the design of another in a nontrivial way. More straightforward was the use of EDSAC 1 to generate the wiring schedules for EDSAC 2.

EDSAC 2 was first checked out using a small microprogram matrix with only 48 cores. In this form, the machine known as EDSAC 1½ was capable of real work and was used by Joyce Wheeler in the spring of 1957 for part of her thesis research on stellar structure.

A microprogramming system implemented in the vacuum tube technology just described represented a large investment in equipment, and it was desirable to make full use of that investment. It was accordingly a principle of the design that every gate in the machine that could be driven from the microprogram matrix should be so driven. This goal was achieved to an extraordinary extent. Neither the input and output mechanisms nor the magnetic tape decks had any local sequencing control, but were controlled directly by the microprogram. Even in the memory, the reading of a word and its subsequent rewriting were controlled by sequences of microinstructions.

Arithmetic unit and instruction set

The word length of EDSAC 2 was 40 bits, with 20-bit instructions. Seven of the bits in an instruction constituted the operation code and 11 the address. There were two 11-bit index registers. Two bits in an instruction gave the programmer the option of modifying the address by adding the contents of one or other of those registers, of modifying it by adding the contents of the program counter (thus providing for relative addressing), or of not modifying it at all. We did not follow the EDSAC 1 design and make it possible to address half registers in arithmetic instructions.

Both the instruction set and the initial input routine were designed by David Wheeler. The instruction set may be described as straightforward, symmetrical, and comprehensive, without being overelaborate. Wheeler provided instructions which caused the next instruction in the program to be modified exactly as it would have been if the modifying quantity had been loaded into an index register. Since it is common for an index register to be loaded and used once only, these instructions did much to mitigate any inconvenience the programmer might have felt from the fact that the hardware provided two index registers only. There were special instructions for facilitating the construction of simple loops in a program.

Wheeler provided a single logical operation; this was an *or* operation instead of the more obvious *and* operation that was provided in EDSAC 1. Given the register structure of EDSAC 2, the *or* operation was both simpler and faster. Wheeler argued that many operations for which an *and* instruction might seem appropriate could be better performed by shifting, an example being the building up of instructions during the reading of an input tape. Nevertheless, the decision to provide only an *or* instruction proved to be unpopular and later, when a number of enhancements and improvements were being made, an *and* operation was added.

EDSAC 2, like other second-generation vacuum tube computers, was designed to have floating-point operations. The complexity of such operations had deterred the designers of the very earliest machines from attempting to implement them. The advantages of microprogramming were especially evident in this context, since the complexity was a problem for the designer of the microprogram, not one for the designer of the hardware. In EDSAC 2 a substantial part of the microprogram consisted of microinstructions for implementing the individual steps of floating addition, multiplication, and division.

Although EDSAC 2 was provided with floating-point operations implemented in the microprogram, we took great care over the design of the fixed-point instructions. Built in (that is, microprogrammed) division was provided and was designed in such a way that it was possible to divide a double-length number in the accumulator by a single-length number from memory. Experience with division subroutines for EDSAC 1 had shown that serious scaling problems present themselves if it is only possible to divide a single-length number by another single-length number and obtain a single-length result. Similarly, the ability to accumulate double-length products in the accumulator had been found to ease scaling problems, as well as enabling a scalar product to be evaluated without incurring intermediate rounding-off errors. We provided enough storage on flip-flops in the arithmetic unit to enable these double-length operations to be microprogrammed without requiring intermediate accesses to core memory. This was a more important decision than it may seem, because the provision of an extra register cost a lot in vacuum tubes.

The regular instruction set was supplemented by a set of instructions that activated subroutines held in the reserved store. These instructions all had 59 for their operation codes; what in regular instructions would be the address part indicated the particular subroutine to be called. For example, the instruction "59 f 11" would cause the floating-point number in the accumulator to be replaced by its square root. Analogous instructions would enable trigonometric functions to be computed. Other instructions were concerned with input, output (including page layout control), solution of ordinary differential equations by the Runge-Kutta-Gill method, and control of the magnetic tape system.

Hardware maintenance

If the microprogramming system is excluded, by far the greater part of EDSAC 2 consisted of circuits packaged in a bit-sliced manner. There were 40 arithmetic slices and 11 control slices, each slice being contained in a single plug-in unit. One of the claims made for the use of a bit-sliced organization was that it would facilitate maintenance. Experience with EDSAC 2 bore this out.

When a fault in the computer was suspected, a test program enabled the control and arithmetic functions to be exercised. If there were a fault in one of the above-mentioned plug-in units, the unit concerned would be identified by the program. It could then be replaced by a spare one, and the fault in the original unit could be diagnosed and corrected at leisure using off-line equipment (Figure 2).

This system was so successful that the off-line testing facility was gradually enhanced to enable many of the other units to be similarly tested off line. This extension to units that were not replicated, or not replicated many times, involved a significant investment in equipment, but it was one that experience showed to be amply justified.

A facility known as *marginal checking* was provided to help locate incipient faults ahead of the time when they would impair operation. This enabled a small AC voltage at mains frequency to be injected in series with the signal at critical points throughout the machine. A similar system, but using DC voltages, had been successfully applied to EDSAC 1.[5]

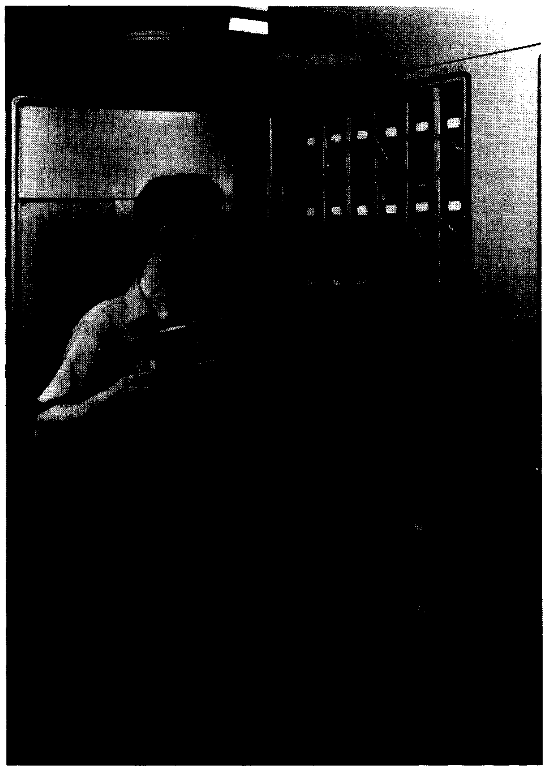


Figure 2. Sidney A. Barton (later chief engineer of the laboratory) inserting a bit-sliced chassis into EDSAC 2. The chassis slid into a slot and was finally driven home by a screw. The screw passed through the chassis and terminated in a handle at the front. The advantage of this system was that it eliminated any danger of the contacts being damaged when the chassis was forced home.

Speed of operation

A sufficient indication of the speed of the machine can be given by saying that an add or subtract instruction took 17 to 42 microseconds (fixed point) and 100 to 170 microseconds (floating point), while an add-product instruction took 270 to 330 microseconds (fixed point) and 210 to 340 microseconds (floating point).

When first put into service in 1958, EDSAC 2 ran somewhat more slowly. This was because the microprogram had been written on the assumption that EDSAC 2 would run at the same clock speed as EDSAC 1½, in spite of the fact that it had a very much larger microprogram matrix. In fact, the microprogram ran more slowly, with the result that some of the nonfunctional microinstructions, which had been included in the microprogram for the sole purpose of giving the adder time to settle down, could be removed.

Input of instructions

EDSAC 1 used five-hole punched paper tape for input and output, and after some discussion we decided that EDSAC 2 should do the same. The main merit of paper tape was that programs could be read from it into the memory at a very high speed — at least 100 instructions per second. The IBM 704, first announced in May 1954, was a machine of similar speed to EDSAC 2 but used punched cards for input. With a card reader operating at 150 cards per minute and one instruction punched on a card, this gave an instruction input rate of only 2.5 instructions per second. As will be seen, this influenced the approach made to the design of the initial input routine.

The initial input routine of EDSAC 2 was the natural development of that written for EDSAC 1. It contained a number of additional features, in particular labels, and the automatic listing of constants that occurred in addresses. I had referred to labels under the title floating addresses at the Eastern Joint Computer Conference held in Philadelphia in 1951. I described them in greater detail at an ACM meeting held in Toronto in September 1952[6] and in a further paper published in the *Proceedings of the Cambridge Philosophical Society* in 1953.[7] Until they were implemented for EDSAC 2, however, no practical use had been made in Cambridge of the techniques described.

The EDSAC 2 initial input routine provided a set of assembly-time parameters denoted by p1 through p99. When an input tape was in the process of being read, p1 denoted the storage location into which the next instruction or number read from the input tape would go; p2 similarly represented the location into which the next constant would go. The remaining parameters could be used at the programmer's discretion. They could be assigned values explicitly by means of directives included on the program tape. Alternatively, they could be set implicitly by being used as labels. For example, if 27 were punched after an instruction or a number on the input tape, when the tape was read, p27 would be set equal to the address of the storage location in which that instruction or number was placed. The programmer could write a parameter in the address part of an instruction, either by itself, or in combination with constants or with other parameters. For example, 2p27 would represent the value of p27 incremented by 2, and p27p28 would represent the sum of the values of p27 and p28.

The primary consideration in the design of the initial input routine was speed of input of orders. The (five-hole) paper-tape code was designed so that all the characters used in instructions were on the same shift of the keyboard, so that the length of the program tape was not unnecessarily increased by shift characters. The high speed of the tape reader used on EDSAC 2 limited the time available for processing between the reading of rows of holes. This and the desire to accommodate the initial-input routine in the reserved store made it imperative to keep that routine short. This led to the acceptance of certain restrictions on the use of parameters ahead of the place in the program at which values would become assigned to them.[8] It also led Wheeler to use decimal digits instead of mnemonic letters to represent operation codes. I personally felt that this was a retrograde step; Wheeler, on the other hand, took the view that if the numerical codes were allocated on a systematic plan, they would in practice be as easy to remember as two- or three-letter alphabetic codes would have been. In the context of EDSAC 2, with only 120 instructions in the instruction set, he was probably right.

The distinctly parsimonious attitude to assembly-time features adopted at Cambridge is in marked contrast to the attitude adopted for the IBM 704. One of the first tasks of the organization that became SHARE was the standardization of the input system for the 704. The program eventually chosen was called the Symbolic Assembly Program, or SAP. Because of the slowness of the card reader and the absence of a reserved store, the same considerations did not apply as at Cambridge, and SAP provided many of the convenient features of modern assembly programs.[9]

Program diagnostic aids

When the machine was initialized for a new program to be run, every bit in the memory was set to a 1, rather than to a 0. Since, according to the format used for floating numbers, no floating number could consist entirely of 1s, any attempt to read a floating-point number from a part of the memory that had not been written into caused the machine to stop. Many a programmer must have been grateful to Wheeler for his foresight in making this happen.

When any unscheduled stop occurred — for example, in the circumstances just described, or when an accumulator overflow occurred — what actually happened was that control was transferred to a routine in the reserved store which caused selected information, such as the place in the program where the stop occurred and the contents of the accumulator, to be printed out before the machine came to a stop. This was known as a *report stop*, and the information printed was useful both for program debugging and for diagnosing machine malfunction.

Experience with EDSAC 1 had demonstrated the value for debugging of being able to keep a record of jumps taken during the running of the program. The EDSAC 1 trace routine was perforce interpretive and ran very slowly. The trace routine in EDSAC 2 was implemented partly in the microprogram and partly in the reserved store. Switching it in slowed the machine down by a factor of no more than about 10. A utility program was provided for analyzing the data accumulated by the trace program and printing the result of the analysis in a compact form, showing cycles within cycles up to a depth of three.

Tape readers and punches

Early in its life, EDSAC 1 was equipped with a photoelectric tape reader in which the tape was advanced one row of holes at a time by a ratchet-driven sprocket wheel, the ratchet being activated by an electromagnet. This would read tape at a speed of about 50 rows of holes per second. B. Pollard, at Ferranti Ltd., was responsible for the development of a roller-driven photoelectric tape reader, using electromagnetic clutches. This had no sprocket wheel, but relied on photoelectric sensing of the sprocket holes. When a row of holes had been read, the tape reader would, unless the next row of holes was immediately called for, bring the tape to rest with that row in the reading position.

We developed, for EDSAC 2, a tape reader in which the tape was driven by a continuously rotating capstan against which it could be pressed by a solenoid-operated pinch roller. It was necessary to have a brake to stop the tape quickly, and this brake took the simple form of a flat soft iron armature held in a resilient mounting in such a position that it rode on the upper surface of the tape. Below the tape was a flat plate in which a solenoid was incorporated. When the solenoid was energized, the magnetic field passed through the tape and attracted the armature. The effect was to bring the tape rapidly to rest by squeezing it between the armature and the plate.

The tape reader could read tape at speeds up to 1,000 rows of holes per second and stop instantly, if required, with the next row in position for reading. The secret was to make the brake sufficiently powerful to arrest the tape even though the pinch roller was still pressing it against the capstan. Rapid stopping could then be achieved even if the pinch roller mechanism was relatively slow in operating. The tape reader was very successful. It was taken up by Elliott Brothers and remained in production for many years.

EDSAC 2 used five-hole punched paper tape for output as well as for input. Punches running at various speeds were used, the fastest being an experimental punch supplied by Creed Ltd. and capable of punching tape at the rate of 300 rows per second. This was much used on account of its high speed, although it was not as reliable as could be desired. The paper-tape code was chosen so that each of the decimal digits from 0 through 9 was represented by two holes and three blanks. This gave good protection against punching errors, since two compensating errors were required to convert a decimal digit into another decimal digit. The same system had been used, with great success, on EDSAC 1 during its later years.

The emphasis placed on paper-tape readers and punches that ran at a high speed, and could be started and stopped rapidly, may seem surprising, since the total volume of input and output was not large. The reason was that input and output time could not be overlapped to any significant extent with computing time. At a later period, input and output would be buffered. In the 1950s, this would have been impracticable, since it would have meant the use, on a large scale, of vacuum tube flip-flops. It must be remembered that techniques by means of which main memory could be used for buffering had not yet been developed. Even if they had been, the main memory would not have been large enough to permit their effective use.

Magnetic tape

The magnetic tape decks attached to EDSAC 2 were bought from Decca Radar, although the basic research and development for them had been done in the Mathematical Laboratory by D.W. Willis. Willis had in due course joined Decca, taking the technology with him and using it to design a product.

A program of work on magnetic-tape systems had been started in the Mathematical Laboratory in the early part of 1952. At that time no work on magnetic tape was being done in the United Kingdom, and little design information was available. We had, therefore, to evolve our own designs for the tape transport mechanism and for the tape servos, as well as for the magnetic heads themselves. We had to learn how to make stampings for the heads from sheet metal, and then how to assemble and wind them. For driving the tape, we developed pneumatic capstans capable of driving the tape at 100 inches per second in either direction.[10] I had much enjoyed working with Willis on magnetic-tape development, and I felt a strong pull to design and have built in the laboratory the tape decks for EDSAC 2. I am sure, however, that we were right to buy the Decca units, which served us well.

In IBM systems, information was written onto magnetic tape in a continuous stream of arbitrary length and terminated by an end-of-record marker. EDSAC 2 followed the alternative plan, in which the tapes were marked out in advance into addressable blocks. Information could be written into a specified block, and at a later time could if desired be overwritten by other information. An interrupt system, based on one that we had experimented with on EDSAC 1, enabled the tape to be positioned ready for a given block to be written or read while other computation was in progress.[11] These facilities enabled the tape system to be used as an addressable auxiliary store. Since there was no drum or disk, this was important.

The EDSAC 2 magnetic tape system was peculiar in that the tape was run out of contact with the heads. We adopted this system because the magnetic tape then available was much troubled with lumps in the magnetic oxide coating that caused the tape, if run in contact, to jump away from the head. Running out of contact also had the advantage of eliminating the wear on the heads, a serious problem in those days of soft heads and very abrasive tape. A disadvantage was that the signal produced was less than with in-contact running, although the noise level was much the same. The idea of running the tape out of contact with the heads was my own and was certainly successful in the context of EDSAC 2. The motivation for it, however, went away as improved tape and heads became available, and I was wrong in thinking that it might become generally adopted.

The EDSAC 1 trace routine was perforce interpretive and ran very slowly. The trace routine in EDSAC 2 was implemented partly in the microprogram and partly in the reserved store.

A larger high-speed memory

By the late 1950s, the 1,024-word high-speed memory began to look woefully small. Much larger memories were becoming commercially available at affordable cost, and we longed to be able to connect one to our machine. The problem was, of course, that there were not enough address bits.

This situation was highly frustrating and, at first, we did not see that we could do anything about it. However, as time went on, there arose in my mind the conviction that some solution must be found, imperfect and inelegant though it might be. Possibly it could be based on a form of indirection. Even if two accesses to the extended memory were necessary in order to reach a given target word, this would be no disaster, since the new memory would have half the cycle time of the existing core memory.

I eventually arrived at a proposal that I thought might be workable. It involved using the relative addressing bit in an instruction in a modified way, and providing a new index register for use exclusively with the new memory. I put this to Wheeler who, after some study, arrived at an improved proposal that we eventually implemented.

Wheeler was able to leave unchanged the meanings of the bits in an arithmetic instruction, except that the range of relative addressing was restricted to the range –256 to +255. In practice, this was no restriction, since users had always been taught to use relative addressing for very local references only and to use labels otherwise. It did, however, make it possible for Wheeler to implement a system in which the first 256 words in the new memory could be addressed directly, and the remainder could be addressed indirectly via those 256 words. (In fact, 512 words could be addressed directly, but only 256 of them could be used for indirect addressing.) Instructions were still executed out of the old memory. Blocks of instructions could be held in the new memory, but it was necessary to transfer them to the old memory for execution.

The new memory was supplied by Ampex, who I believe had acquired the rights from Telemeter, Inc. It had a capacity of 16K words. Implementation of the interface went surprisingly smoothly, and the new system came into use early in 1962. 16K words may not seem much to modern readers, but at the time users of EDSAC 2 found it almost unbelievably large. Wheeler enhanced the initial orders to accommodate the main memory, and it was generally agreed that the overall scheme was not only highly effective, but had a certain elegance after all.

The principal historical importance of EDSAC 2 is that it established beyond doubt the technical viability of microprogramming as the basis for a practical computer design. The attention of IBM was drawn to EDSAC 2 by W.S. Elliott, then head of IBM's Hursley Laboratory, and as a result of this contact, IBM decided to make microprogramming an essential element in their System/360 implementation. My office diary records that on October 2, 1961, Cuthbert Hurd, vice president of IBM, visited the laboratory. Hurd did not disclose the object of his visit, but he has since told me that it was an important factor in leading IBM to take their decision. By the time that System/360 was announced in 1964, transistors had come into general use, and microprogramming was economically, as well as technically, viable.

I regard the bit-sliced packaging used in EDSAC 2 as technically very successful, but in this case, the coming of transistors with their different packaging requirements robbed it of any immediate future it might otherwise have had. Bit-slicing was not heard of again until it was reinvented much later in the context of microcomputers based on the LSI technology of the mid-1970s. ■

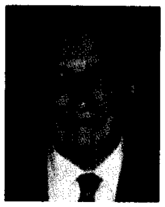
Acknowledgments

The construction of EDSAC 2 was made possible by a generous grant from the Nuffield Foundation. Apart from myself, the principal members of the project team were W. Renwick (chief engineer), D.J. Wheeler (responsible for the design of the instruction set and programming environment), and E.N. Mutch (project manager).

In writing this account, I have drawn on laboratory archives, on my own recollections, and on the recollections of surviving members of the team. I am particularly grateful to K.O. Cox, now chief engineer of the Cambridge University Computer Laboratory, for information about the off-line test equipment.

References

1. M.V. Wilkes, *Memoirs of a Computer Pioneer*, MIT Press, Cambridge, Mass., 1985.
2. M.V. Wilkes, "The Best Way to Design an Automatic Computing Machine," *Manchester University Computer, Inaugural Conference*, July 1951, pp. 16-18; reprinted in *Annals of the History of Computing*, Vol. 8, 1986, pp. 118-121, and in *The Early British Computer Conferences*, M.R. Williams and M. Campbell-Kelly, eds., Charles Babbage Inst. Reprint Series for the History of Computing, Vol. 14, MIT Press, Cambridge, Mass., and Tomash Publishers, Los Angeles, 1989, pp. 182-184.
3. M.V. Wilkes, *Automatic Digital Computers*, Methuen, London, 1956.
4. M.V. Wilkes, W. Renwick, and D.J. Wheeler, "The Design of the Control Unit of an Electronic Digital Computer," *Proc. IEE*, Vol. 105B, 1958, pp. 121-128.
5. M.V. Wilkes, M. Phister, and S.A. Barton, "Experience with Marginal Checking and Automatic Routining of the EDSAC," *IRE Convention Record*, Part 7, 1953, p. 66. Also in *Int'l Symp. Automatic Digital Computation*, National Physical Laboratory, Mar. 1953, pp. 16-18; reprinted in *The Early British Computer Conferences*, M.R. Williams and M. Campbell-Kelly, eds., Charles Babbage Inst. Reprint Series for the History of Computing, Vol. 14, MIT Press, Cambridge, Mass., and Tomash Publishers, Los Angeles, 1986, pp. 446-452.
6. M.V. Wilkes, "Pure and Applied Programming," *Proc. ACM Nat'l Conf.* (Toronto, Sept. 1952), ACM, New York, 1952, pp. 121-124.
7. M.V. Wilkes, "The Use of a 'Floating Address' System for Orders in an Automatic Digital Computer," *Proc. Cambridge Philosophical Society*, Vol. 49, 1953, pp. 84-89.
8. University Mathematical Laboratory, *Programming for EDSAC 2*, Cambridge, UK, 1958, 2nd edition 1959, pp. 56-57.
9. C.J. Bashe et al., *IBM's Early Computers*, MIT Press, Cambridge, Mass., 1986, pp. 349-354.
10. M.V. Wilkes and D.J. Wheeler, "Auxiliary Storage on Magnetic Tape in EDSAC 2," *Congresso Internacional de Automatica* (Madrid, 1958), 1961, p. 185.
11. M.V. Wilkes and D.W. Willis, "A Magnetic Tape Storage System for the EDSAC," *Proc. IEE*, Vol. 103B, Supplement No. 2, 1956, pp. 337-345.



Maurice V. Wilkes received a PhD from Cambridge University in 1936 for a thesis on the propagation of very long radio waves in the ionosphere. In 1937, he was appointed to a junior faculty position at Cambridge in connection with the establishment of a computing laboratory. He left for war service on the outbreak of war in 1939 and worked in radar and operational research. When he returned to Cambridge in September 1945, he was appointed head of the laboratory. He was responsible for the construction of the EDSAC, which was running in June 1949, and later for the construction of EDSAC 2. He left the Computer Laboratory in 1980, and joined the central engineering staff of Digital Equipment Corp. in Maynard, Mass. Returning to England in 1986, he became Member for Research Strategy on the Olivetti Research Board. He is now a consultant to Olivetti.

Wilkes can be reached at Olivetti Research Ltd., 24A Trumpington Street, Cambridge CB2 1QA, UK.