Computer Conservation Society

Aims and objectives

The Computer Conservation Society (CCS) is a co-operative venture between the British Computer Society, the Science Museum of London and the Museum of Science and Industry in Manchester.

The CCS was constituted in September 1989 as a Specialist Group of the British Computer Society (BCS). It is thus covered by the Royal Charter and charitable status of the BCS.

The aims of the CCS are to

- ◇ Promote the conservation of historic computers and to identify existing computers which may need to be archived in the future
- ◇ Develop awareness of the importance of historic computers
- ◇ Encourage research on historic computers and their impact on society

Membership is open to anyone interested in computer conservation and the history of computing.

The CCS is funded and supported by a grant from the BCS, fees from corporate membership, donations, and by the free use of Science Museum facilities. Membership is free but some charges may be made for publications and attendance at seminars and conferences.

There are a number of active Working Parties on specific computer restorations and early computer technologies and software. Younger people are especially encouraged to take part in order to achieve skills transfer.

Resurrection

The Bulletin of the Computer Conservation Society

ISSN 0958 - 7403

Number 23

New Year 2000

Contents

Editorial

Nicholas Enticknap

For virtually everyone with an interest in computer conservation, the main talking point this autumn has been the continuing doubts over the future of Bletchley Park. I had hoped to be able to include positive news about the situation in this issue, but as we went to press no agreement had been reached on either the future of the Park or the Society's plans for museum activities. Regrettably I have to report that the wrangling publicised in the national press is still continuing.

We all must hope that the new millennium brings a speedy outbreak of common sense in all relevant quarters, and that the Bletchley Park site which played such a vital role in the development of the electronic computer is allowed finally to become a permanent showcase for its historic 1940s technology.

There is better news from the Science Museum, where plans to display the Society's Pegasus are now well advanced. It is probable that before our next issue visitors to the Museum will be able to experience the very different world of computing nearly half a century ago at first hand.

The IBM System/360 range, launched 45 years ago last April and still going strong in System 390 guise today, is the subject of two of our features in this issue. Ivor Jones, who still works for the company, describes how the range introduced the concept of a computer architecture and how that architecture evolved. He also highlights the important role of microprogramming in achieving the compatibility that was the novel feature of the range.

Mike Flinders, who also worked at Hursley where the 360/40 was designed, discusses the input/output arrangements, in particular the development of the selector and multiplexer channels which became such a distinctive feature of IBM mainframes.

Our other article takes us a step backwards in time to the early fifties when British Tabulating Machine Co (BTM) was just beginning to recognise the potential of electronics. Lorin Knight's article describes the development of the Calculator range which preceded the early BTM computers described by Dickie Bird in issue 22.

News Round-Up

Congratulations to Society Chairman Brian Oakley, who was made a BCS Honorary Fellow in October. The citation acknowledged his work as Director of the Alvey Programme and his support for progress in quantum computing as well as his role as CCS chairman.

- 101010101 -

An archive of computer documents has been published on the Web, as part of a project organised by the Science Museum in conjunction with the Museum of Science & Industry in Manchester. It is the product of extensive work by the Society's Archivist, Harold Gearing.

The Web site, at <www.sciencemuseum.org/ncclp/ncclp.html>, provides information about the computing collections in both museums. It will be developed to include collections from other museums, including Birmingham and the National Museum of Scotland, and will also include links to other useful sites, including the Archive for the History of Computing at Manchester.

- 101010101 -

Simon Lavington reports that he has received 71 replies so far to his appeal for personal reminiscences of involvement with pre-1970 UK computers. The CCS Preservation Priorities Working Party is grateful to those respondents, who should all have received an acknowledgment from Simon. The CCS is still keen to hear from anyone else who has recollections of the design, production or use of UK-designed computers that ran before the arrival of IBM's System 370 (see pages 30-32 of issue 22 for a complete list of such machines). The appeal appeared in *Computer Weekly*, *Computer Bulletin* and *IEE News* as well as in our last issue.

- 101010101 -

The Society's North West Group has an urgent requirement for space for members to store equipment and spares and to do some restoration work. If any member knows of any available workshop space in the Manchester area, would they please contact the Group's Secretary, Ben Gunn, on 01663 764997. The Group is looking for a room about 24 feet square, with power and light, in a secure building.

- 101010101 -

CAFS aficionados may like to know that a reunion is being planned for the middle of 2000. Anyone who would like to be kept informed should let Hamish Carmichael know by phoning 020 8337 3176.

- 101010101 -

Society member Alan Bosworth spends some of his spare time converting files on obsolete disc media to new formats. He estimates he can cope with an astonishing 600 formats on both single and double density floppies. Members who need help in rescuing old material can contact Alan at Arosfa, Watts Green, Chearsley, Aylesbury, Bucks HP18 0DD, or telephone him on 01844 208380.

- 101010101 -

Readers who have general queries to put to the Society should address them to the Secretary at the address given on the inside back cover.

Members who move house should notify Hamish Carmichael of their new address to ensure that they continue to receive copies of *Resurrection*. This is because the CCS membership is different from the BCS list.

The BTM Calculators

Lorin Knight

I joined BTM in April 1950, six months before the company terminated its tie-up with IBM. At that time I think Doc Keen was still called the Head of Research but, as Dickie Bird has already related[1], Doc would have nothing to do with electronics which he regarded as incapable of providing adequate reliability.

Consequently Cyril Holland-Martin, the Technical Director, who foresaw electronics having some impact on the punched-card world, had set up a small electronics laboratory reporting directly to himself, to launch the company into the electronic era. It was this happy group, consisting of Billy Woods-Hill, Bill Davis, Alec Trussell and Martin Circuit, that I joined. Martin and I were both new boys to the team, Martin having joined shortly before me.

The initial assignment for the group was to produce an electronic black box which could be coupled to a 100 card per minute punch and perform calculations of the type (AxB)±C, taking its input from, and punching the answer back into, the same card. And it was to be capable of handling numbers in decimal or £sd notation.

The available 501 electromagnetic multiplier took so long to perform a multiplication that it reduced the card speed to a pitiful 10 per minute. Whether or not one thought there was any future for electronic computers it was clear that for a multiplier to work at an acceptable speed it would have to use electronic technology. IBM had launched its 603 Electronic Multiplier in 1946 and already replaced it with the much more powerful 604 Electronic Calculator. Both of these were available to BTM but the company took no interest in them because they could not handle £sd.

The Bug-eyed Monster

At the time I arrived the electronics group had produced a rack containing several hundred radio valves called the BEM. This was an acronym for British Electronic Multiplier but it was sometimes irreverently translated as Bug-Eyed Monster.

Not long after I arrived the BEM was fully working and doing all its arithmetic correctly. Although composed of nominally identical flip-

[1] In his article "BTM's First Steps Into Computing" in *Resurrection* issue 22.

flop circuits and nominally identical gate circuits, the multiplier had been coaxed into a working condition by selecting a "better" valve for a few odd positions and changing a resistor value in the odd location which didn't seem to like the standard value.

When the working multiplier was demonstrated to Holland-Martin he made a simple perceptive reply. "Now," he said, "remove all the valves and plug them back into different positions". This was done and the BEM was never persuaded to work correctly again!

Birth of the 541

It was time to go back to the drawing board. This time much more attention was paid to ensuring that the basic circuit elements could tolerate the variations in valve characteristics which would be encountered as well as the variations in loading they would receive. More attention was also paid to making the multiplier more suitable for manufacture in modest quantities.

The circuits of what was to become the 541 Electronic Multiplier were built onto plug-in chassis which held 12 octal-based valves, care being taken to limit the number of different chassis types so far as was practicable. ECC33 double triodes were used for the flip-flops and 6F32 pentodes were used for the gates.

The 6F32 was unusual in that its anode current could be cut off by quite a small negative potential on grid-3 or grid-1 and seemed the obvious choice for the gates. It was not manufactured in any large quantities and, because of this, might not have turned out to be very reliable — but we were lucky!

There were three registers, each capable of holding a 10-digit number in either decimal or £sd notation. The binary code for each digit was held on four flip-flops and automatically switched as necessary to be in scale-of-10, scale-of-12 (for pence) or scale-of-2 (for multiples of 10/-). The 10 digits were handled serially with their binary constituents handled in parallel.

The clock frequency was 12kHz which we thought to be near the limit for achieving reliable operation with all the stray wiring capacitances we had. We never dreamed that by the end of the century digital circuits working with clock frequencies as high as 120MHz would be considered slow. Much of the logic design was based on ideas of Billy Woods-Hill which had been incorporated in the BEM — in particular the arithmetic unit and the "halving and doubling" system of multiplication, which he

claimed had been used by the ancient Egyptians.

The example below shows how this method was used to multiply 17 by 11.

Register A	Register B	Register C	
Multiplier	*Multiplicand*	*Product*	
11	17	17	
5	34	34	
2	68		
1	136	136	
0	272		

		187	*Product*

It will be seen that the *multiplier* is progressively halved (fractions being ignored) and the *multiplicand* is progressively doubled. Each time the *multiplier* is odd a copy of the *multiplicand* is added into the *product* register. Eventually the *multiplier* reaches 0 (taken as an even number) and no more transfers to the *product* register are required.

It is, of course, binary multiplication, disguised by retaining the numbers in decimal notation, and requires little of the arithmetic unit other than the ability to add, halve and double. The method was equally applicable to sterling calculations, where the *multiplicand* and *product* would be in £sd notation.

There was, however, one little problem with this method of multiplication. It required the *multiplier* to be a whole number. If it contained decimal places it had first to be multiplied by 10^n, a suitable power of 10 to make it an integer, and the *multiplicand* divided by 10^n to compensate. With scale-of-10 numbers this was simple because multiplication and division by 10 were achieved just by moving the decimal point, but if the *multiplicand* was in £sd notation it was not quite so simple. A special divide-by-10 facility for £sd quantities had to be added and this would be used n times to obtain division by 10^n.

By means of a plugboard, a limited sequence of additions, subtractions and multiplications could be programmed. Steps in the program could be made conditional on certain parameters, such as whether a given number was positive or negative or whether a certain designation had been punched into the card. A separate plugboard would be set up and plugged in for each job the Multiplier was required to do.

The front of the 541 had a display of around 300 miniature neon lamps which showed the state of every flip-flop and gave a pictorial representation of the contents of the registers and the arithmetic unit, plus the state of all the control lines. Coupled with the facility for replacing the electronically generated clock pulses with single, manually generated pulses, these lamps provided a powerful faultfinding tool.

Not long after I joined, Bill Davis left the multiplier group to work on the "Plastab" (a mysterious materials-handling project which never got off the ground) and Dick Cox joined us. We were a motley crew, coming from quite different backgrounds and providing complementary expertise.

Our lab was tucked away behind the Field Engineering headquarters on the north side of Icknield Way in Letchworth, away from Doc Keen and his boys on the opposite side. We had no telephone. Anyone wishing to contact us had to ring an office in Field Engineering. The girl there then jiggled a piece of string which went through a hole in the wall to our lab and rattled some pieces of scrap aluminium to summon one of us to the phone. Our location, indeed our existence, seemed to be unknown to most of BTM and for over a year we were able to get quietly on designing our multiplier.

Towards the end of 1951 we had a prototype working successfully — and there was then a rush to get it into production as soon as possible. Drawings covering most of the mechanical construction had already been produced for the building of the prototype but no drawings existed for the physical construction of the units which sat in the base of the multiplier and produced the $2\frac{1}{2}$ kW or so of stabilised AC and DC used to power the 800 or so valves.

We had built these power supplies in the lab, improvising metal frames from assorted steel rods and brackets salvaged from the Field Engineering scrap heap just outside our lab. There was no time to tidy up the physical design; the draughtsmen came in and faithfully documented our rather unusual construction method.

The prototype 541 Electronic Multiplier was displayed at the Business Efficiency Exhibition at Birmingham in the spring of 1952, and several production models were installed in customers' offices later that year.

I remember demonstrating the 541 at the BEE to an elderly gentleman who turned out to be one of the top brass from Powers Samas. For most of his working life, it transpired, he had been involved in the development of punched card machines and punched card accountancy techniques and

he was obviously rather proud of the knowledge and expertise he had accumulated. "But", he said, "this is only the starting point for young fellows like you with new ideas and new technologies who are jumping straight in where we left off — and leaving me and my generation way behind". I felt a tinge of sympathy for him — even though he came from the enemy!

In the three years that I had now been with BTM it had become obvious to the top management that electronics was going to have a much greater influence on punched card accountancy than they had originally thought and several significant moves had been taken.

Dickie Bird had been given the job of running a second electronics lab, charged with producing a small computer suitable for use with punched card accountancy machines. John Womersley had been brought in from the National Physical Laboratory to be head of computer research and Doc Keen had been promoted sideways to consulting engineer. Somewhere around this time the calculator lab moved into new premises in Stevenage, where we had a workshop, an office — and a telephone.

On to the 542

With the 541 in production we immediately set about the design of its successor, the 542, which was to give an improved performance and to be the first of a family of more powerful electronic machines. The basic logical design was similar to the 541 but various improvements gave it an enhanced versatility. For example, some buffer input stores were provided so that the calculation was not restricted to whatever could be read into the registers. These stores consisted of banks of 0.1μF capacitors which were charged or discharged by relays on the punch and could be repeatedly interrogated during the calculating period.

The card punch provided with these machines had two reading stations, one ahead of the punching station and one after it. Provided that there was sufficient calculating time available, this meant that a very useful check could be made by repeating the sequence of calculations (preferably using a different method) and comparing the second result with that punched in the card.

The basic building bricks were two-valve plug-in units (about 4x3x2 inches) which we called turrets. These were plugged into metal gates on the main frame, each gate holding around 100 turrets and an electric blower which forced cooling air up through them. Peter Briggs, of the

Works Division, was responsible for the physical design of the turret — this illustrates the close liaison which by now existed between us and the people who were going to be responsible for manufacturing the electronic machines.

Apart from a few miniature thyratrons used in connection with read out to the punch, the only valve type used was the ubiquitous 12AU7 (ECC82) double triode. There were two kinds of gate circuit. The "pulse gate" emitted an output pulse in response to an input pulse provided that the applied control voltage was at its "high" level. This used one triode, the input pulse sitting on top of the control voltage and both needing to be present for an output pulse to occur.

The other type of gate performed logical operations on control lines and would use two or more triodes. Flip-flops required just two triodes. Such germanium and silicon diodes as were available were not suitable for use with the high voltages encountered in valve circuitry but selenium diodes were used in some applications where their high forward resistance and high self-capacitance could be accommodated.

Efforts to improve the tolerance of the circuits to valve deterioration went a little further than with the 541. The target, occasionally not quite met, was that every circuit, in its machine environment, should work correctly with a "standard dud valve". Such a valve could be synthesised from a typical new one by adding 10kΩ in series with the anode and 1.5kΩ in series with the cathode.

The theory was that valves deteriorating toward the "standard dud valve" level would be all be discovered by marginal testing and replaced during scheduled maintenance. Thus it would only be a heater failure that was likely to cause a machine failure.

In an attempt to reduce the frequency of heater failures, the stabilised 6.3v supply which powered them was arranged to bring the voltage up gradually in order to eliminate any significant switch-on shock which might lead to eventual heater fatigue.

These measures certainly seem to have done some good. From memory I believe the proportion of valves causing machine failures was only around 0.25% per 1000 hours. Failures due to contact problems on the turrets and plugboards were more numerous, and often more difficult to find, but experience gained regarding the precise nature of the connector problems and ways of controlling them enabled some significant improvements to be made as time went by.

I think the first delivery of a 542 to a customer was in 1954. During 1955, deliveries rose to around two a week.

Big Brother

The range was quickly augmented by the 542's big brother, the 550 Electronic Calculator. It contained nearly 600 turrets of some 40 different types and dissipated approximately 3kW of heat. The most significant additional features were a fourth register and the ability to perform the following types of division:

1. $A \div B = C$
2. $£A.Bs.Cd. \div D = £E.Fs.Gd.$
3. $£A.Bs.Cd. \div £D.Es.Fd. = G$

A basic scale-of-10 division such as example 1. was done in the same way as we would have done it manually at school. For a division such as 2. the *dividend*, £A.Bs.Cd, was first converted to a scale-of-ten number by dividing it by a suitable power of 10 to bring it down to just pence and decimals of a penny. The resultant *quotient* then had to be multiplied by the same power of 10 to obtain the required £sd answer.

For a division such as 3. the *dividend* and the *divisor* were both divided first of all by the same power of 10 to convert them to pence. The resultant numbers gave the same *quotient* as would have been given by the original *dividend* and *divisor*.

We never dreamed that anyone would want to divide zero by zero — but of course there was someone who did! Cadbury's had a stock control program which regularly reviewed the average value of each item number in stock by dividing the total value by the quantity. It wasn't long before it found a quantity of zero with a total value of zero. This caused the 550 to freeze up and refuse to do anything more. A hastily introduced modification provided an intercept for $0 \div 0$ which cancelled the normal division routine and set the quotient to 0.

Although they only had a clock frequency of 14kHz, within the field for which they were designed the 550 and the 542 were surprisingly fast compared with the general purpose computers of the period. Some samples of the 550's calculating times are given below.

Total available time	170ms
Addition or Subtraction	$850\mu s$
£999.19s.11d. x 999.9 to nearest penny	15ms
$98876543 \div 9999$ to one decimal place	49ms
£99999.19s.11d. $\div$ *9876* to nearest penny	55ms
PAYE calculation of employee's tax payable	50ms

Top of the Range

The 555 Electronic Calculator was at the top of the range and the first delivery to a customer was in 1957. It had the luxury of drum which could hold the equivalent of 105 registers. It had a massive double plugboard which accommodated up to 150 program steps (compared to the 36 program steps available on the 550). Extra calculating time, if required, was automatically provided by some electromechanical wizardry which put a suitably small delay on the arrival of the following card. This meant that somewhat fewer than 100 cards per minute could be processed with a complex sequence of programs.

The outside of the drum was wound with steel wire which gave a robust surface and a high electrical output. The inspiration for this came from experiments with an audio wire recorder which Billy Woods-Hill just happened to have in the lab. A rather nice little circuit arrangement allowed writing onto the drum, or reading off it, one digit at a time. This extended the step-by-step fault-finding facility to the drum.

Martin Circuit and I left the Calculator Lab in 1955 to work on a project in the USA. Deliveries of the 550 had begun by then and a good start had been made on the design of the 555. By now there were quite a few others who had worked in the Lab or who were still working there. Names which come to mind are Steve Hare, Stan Massam, John Boorman, Julian Tempel and Bert Heath.

Quite a large number of BTM machines were installed in India and my last days with the Lab were spent designing variants of the 542 and 550 which would handle rupees, annas and pies. I had barely finished this task when the Indian Government announced its intention to change to decimal currency!

In conclusion I would like to thank Martin Circuit and John Boorman, who have both provided some useful input to this account of those early days before an electronic calculator was a solar-powered device which could

be held in the palm of the hand.

Donations

We are most grateful to all those members who responded so generously to our appeal for donations in *Resurrection* issue 22. But the purse strings are still tight, so we repeat our appeal for the benefit of those members who did not see issue 22.

At the Society's Annual General Meeting in May 1999, it was agreed that the Society should try for another year to subsist without imposing personal subscriptions, although further efforts would be made to attract additional corporate subscriptions. Since the Society's running costs are partly covered by a grant from the British Computer Society, it can be argued that those CCS members who are also members of the BCS are in effect already paying for a share of the work of the Computer Conservation Society through their annual subscriptions to the parent organisation.

Those who are not members of the BCS are therefore invited to consider making voluntary donations to help cover the costs. (These consist chiefly of the costs of publication and postage.) Cheques should be made payable to The Computer Conservation Society, and should be sent to:

The Treasurer
The Computer Conservation Society
31 The High Street
Farnborough Village
Orpington
Kent BR6 7BQ.

IBM 360 Architecture and Microprogramming

Ivor Jones

Architecture was a word that came into computers with System/360. The concept of the specification of a machine which was independent of technology, of whether it was made of transistors or valves or had magnetic core store or semiconductor memory, was introduced into computer science by Fred Brooks.

Computers before System/360

Examination of the attributes of the machines that existed before the IBM System/360 shows that they were a really diverse collection. "Commercial" machines were typically cost-oriented. The IBM 1401 was a very good example of this type of machine. It drove the printer well and it drove the card reader well, but it wasn't very fast. These commercial machines could handle character data, and featured variable length fields with many different schemes for specifying the field length. They also used decimal memory addressing and some had really terrible extension schemes when they ran out of memory addresses, using the zone bits in the address characters.

"Scientific" machines' main selling point was speed. Most were binary, and many had floating point hardware. However, commercial and scientific machines were beginning to acquire some of the same characteristics, especially in programming support. IBM offered ComTran (a commercial translator) on the 7090 (a scientific machine), and I think there was a 1401 Fortran.

There was experience of building a bigger machine compatible with an existing system, but it was always difficult to make improvements. There was never enough scope to add memory, and there were inconsistencies. For example, some customers experimented with unassigned instruction codes, and discovered some very interesting effects. They might be upset if the successor machine did not work in exactly the same way.

The large machines mostly used magnetic tape input and output, and used smaller machines to handle punched cards. In Hursley at one point we had a 7090 with two or three 1401s handling the card-to-tape and tape-to-printer processing. The processors in most scientific machines had one or two accumulators which were used for both fixed point and floating

point operations. They had other registers — index registers — but they were separate from the accumulators and were much shorter.

Finally, dealing with binary data meant dealing with octal — base eight. The 7090 had a set of piano keys right across the console, with keys in groups of three coloured alternately in two shades of grey.

That was the background when the System 360 design process started.

System/360 goals

I don't think I ever saw the Spread Committee charts that laid down the guidelines for the design. We knew that we had to make a wide range of compatible implementations, and that we had to be smarter with input/output — from both engineering and programming points of view, input/output should look the same irrespective of the size of machine. We had to do something about the end user's view of the machine by improving job turnaround time. I cannot recall any specific architecture features for this, but turnaround times did improve because big machines became capable of running both the printers and card readers if necessary.

We had to consider the operating system. There was intended to be a place in the plan for multiple processor machines, although few of those were made. We wanted more powerful floating point capability. Fred Brooks stressed repeatedly that as the hardware became more reliable, people started believing in the machine, and stopped duplicating runs. They trusted the machine and therefore if it did make a mistake they were going to be much more upset. We should build in checking to make sure that everything was correct at every point.

The System/360 architecture was developed in Poughkeepsie, 80 miles north of New York. Contributions came from people all over IBM, including our Hursley development group. I went to the US in 1961 for a trip that was scheduled for two weeks but actually lasted six. I also spent another two or three weeks there later, but for much of the time, communication between Hursley and Poughkeepsie was via telex. There was an architectural review meeting every week. All proposals were sent to us by telex, and we would have a meeting in Hursley to decide our position, and telex back our input to the review meeting. A report was then produced on what had been decided. On many issues there were phone calls, but telex was the way we worked most of the time.

Making an instruction set, or making an architecture, involves many interrelated decisions. When you pick an optimum word length for fixed

point, you have to use that same word size to store instructions, and so on. They all interact.

We set out with a relatively small number of people and worked for about six months. I wrote a chapter of a specification describing a machine with a push-down stack. More and more people were recruited to implement the design; after about six months we tore up that design and started again. After another two or three months, including what was called a design competition, the architecture began to settle down.

Data format decisions

There was a considerable debate whether the basic unit of data should be six or eight bits. The designers of the fastest machine wanted a 48-bit word, with 6-bit characters; they were aiming to get their costs down and their performance up. They had performance goals, and it takes longer to perform a 64-bit multiply than a 48-bit multiply. Their registers were built of transistors, which were expensive.

The designers of the smallest machine, on the other hand, wanted an 8-bit character; they thought that for most purposes 16-bit addressing would be sufficient and two characters gave them that. We in Hursley, designing the next larger machine, were in favour of the 6-bit character, as we thought a 24-bit memory would have given very efficient instructions. Both machines put registers in core storage, where length was less critical.

The two main arguments that took us to 8-bit characters were the larger character set (it was felt that the 6-bit character would restrict text handling), and the fact that two decimal digits per character allowed numbers to be stored in commercial files more efficiently.

Larger character set was a sound point, but commercial file efficiency seems a highly dubious argument. One may question whether two decimal digits per character was such a good idea. It led to the machine having instructions handling both packed (two decimal digits per character) and unpacked (character) data. Almost the only advantage was that decimal arithmetic with the length of a number specified using just four bits allowed up to 31 digits. I suspect that arithmetic on unpacked decimal data would have needed more than four bits and would have caused other problems.

Thus, character sets, floating point register cost, and instruction efficiency were all weighed. The key architects were Gene Amdahl and Gerrit Blaauw. On this issue Gene was in favour of 6-bit while Gerrit was in favour of 8-bit characters. The issue was resolved by Fred Brooks and Bob

Evans, who decided in favour of 8-bit. I think they were right, because of the larger character set possible and the binary arithmetic advantage.

Other decisions included whether to use hex or binary for floating point. Hexadecimal floating point was much easier for the smaller machines to implement, and I think that was the right decision.

The next argument was over two's complement binary arithmetic. Most of IBM's binary machines had used sign and magnitude for binary data, though they used essentially two's complement in indexing and other operations. We had to do some education to convince some of the Americans that two's complement was a complete and consistent form of arithmetic.

Among earlier IBM machines, some had long parallel decimal registers; their engineers thought that decimal arithmetic should be done on fixed length operands. The small machines held out for variable length decimal arithmetic and I am sure that was right.

Previous machines used many ways to delimit variable length data. Word marks — an extra bit on every character — were very successful in the 1401, but we found that they did not lead to a clean instruction set with binary data. Instructions were shorter as delimiters were in the data, but separate instructions were needed to insert the word marks and move them. We chose to specify the length of data in the instruction — eight bits for character data, and four bits for decimal data.

We established terminology for binary data: bytes, half-words (16-bit quantities), words (32-bits) and double-words. It was decided these fixed length entities should be aligned: addresses for half-words, words and double-words had to be a multiple of two, four or eight; this was checked by everyone, but machines with wider memories benefited.

Addressing

There was little argument about binary addressing. We decided to address bytes, or characters. It was clear that we needed a smaller unit than a word. The IBM Stretch machine, a supercomputer of its day, actually had bit addressing, but that was rather expensive and we decided that we could not do that.

One ground rule we arrived at was that we would use the same address for an operand whether it was being used in an operation which processed left to right, or right to left, for example for decimal arithmetic. Some earlier machines used different addresses. Thus was big-endian addressing

adopted. I don't remember very much debate on that.

We needed an addressing mechanism which combined efficient instructions with the ability to generate long addresses which were obviously needed for large memory. Twenty-four bits allowed up to 16 megabytes, which was more memory than anybody dreamt of in those days.

Actual memories were much smaller. Our 360/40 machine was specified originally to attach up to 128 kilobytes, but we built it in such a way that it was easy to expand. We were pleased we had done this because before announcement the requirement went up to 256Kb. The 360/30 machine below us got faster and faster and seemed to be coming very close to us. However, the designers built the machine tightly around 16-bit addresses; it was very difficult to attach more than 64Kb memory to it later. That left quite a large slice of the market for our machine.

Balancing the size of the address versus the space within the instructions we came to a scheme called 'base addressing'. A 12-bit offset would cover most data structures, but on the other hand, there was a base register from which that offset was measured. Base registers were full length and could be manipulated with all the fixed point arithmetic and logic instructions.

Arithmetic and logical operations

Code for single accumulator machines showed a lot of operands were refetched, so multiple registers were much more efficient. Having set out originally with a push-down stack, we changed to explicitly addressed registers which were much more flexible. To really optimise the code for repeated operands or common sub-expressions, a stack needed a way of addressing entries in the stack. The big machines needed transistor registers. Architecturally one would like to have a fairly unlimited stack, but moving data from core memory to transistor registers was not very easy. Explicitly addressed registers were simpler. We chose four double word floating point registers. We came up with the usual sort of register-to-store, register-to-register instructions. These were mostly two address instructions, which are best for this type of working.

Word-length registers were to be used for addresses, fixed point operands, logic operations and so on, and we chose to have 16 of them. On the other hand, decimal data was not processed much, and there was very little benefit from registers and therefore we arranged for decimal arithmetic to be storage-to-storage, with operands up to 16 bytes, 31 digits plus sign. We

also added storage-to-storage operations on character strings of up to 256 bytes.

We made instruction formats that were based on a two-byte unit — a half word. The instruction set contained one, two and three half-word instructions.

Sequencing

Most older machines had instructions such as Branch-on-Accumulator-Plus, but with 16 fixed point registers and four floating point registers, not to mention decimal results in memory, specifying the test and the branch in the same instruction became rather difficult. We adopted the idea of a Condition Register, set by most arithmetic and logic instructions according to their result. The definition of each instruction specified exactly what would happen. A comprehensive set of branches tested the value in the Condition Register. There were also some more specialised instructions for closing loops, such as count and branch.

One of the requirements was to make a supervisor program that could not be inadvertently or deliberately sabotaged. We needed to prevent wild branches into the supervisor. For each type of interrupt, the entry points were prespecified. Supervisor Call was an instruction to get into the supervisor. The program could pass an operand but couldn't specify where to go; the machine went to a prespecified address. There were External Interrupts, one use of which was in multiprocessors, to allow the individual processors to tap each other for various purposes.

Monitor Control

Storage protection was a key requirement for a multiprogramming machine and we came up with a scheme for that. We had an Interval Timer, positioned in the memory, and this created one of the external interrupts.

We had a number of Program Exceptions, for example Storage Protection. We were careful to close off all the unused operation codes. It seemed that a lot of the microcode was checking things like "Has an odd address been used for a fixed word operand?". There were some more useful facilities such as overflow detection. Those were all classed as Program Exceptions.

The key element in the interrupt system was the Program Status Word (PSW), a double word unit which contained almost everything there was

to know about the program — not only the current instruction address, but whether the I/O channels were enabled to interrupt it, its storage protection key, and so on. We provided an instruction length code to allow counting back to the previous instruction, which is difficult when instructions are variable in length. PSW swapping became the interrupt mechanism. The current PSW was stored in one place and a new PSW was loaded from another. The new PSWs were to be set up by the control program.

I have mentioned some of the requirements for the supervisor. We had a supervisor state which was controlled by one of the bits in the PSW. Certain instructions were privileged. Anything that touched I/O devices, anything that dealt with program switching, and the actual instruction to load PSW, were all privileged; for a program whose PSW had the supervisor state bit off, all those instructions would be invalid. Lastly, we had a wait state, where the machine was not using memory cycles. It could sit idly while allowing I/O to carry on, and would crank back into operation when an interrupt would come along.

Microprogramming

In the Scamp machine, and in an earlier smaller machine in Hursley, we had used microprogramming starting from the ideas of Professor Wilkes, and evolved the scheme into the read only store (ROS) approach. There were five teams of engineers trying to implement the System/360 architecture, and all but one of the models used microprogramming; it was a key feature in achieving compatibility. I'll go further: we would not have achieved compatibility in the time frame we did without the microcode.

Another benefit from microcode was in servicing the machines. When a set of compatible machines has implemented an instruction set in different ways, it is difficult to use that instruction set to locate a specific piece of hardware that has failed. Customer engineers would much rather be told to replace a particular card, than have to hunt and diagnose the fault. Therefore, microcode incorporated tests, which were very rigorous, and which gave a good basis for diagnosing faults to specific parts of the machine.

It was also of benefit during development that simulating programs was a well understood technique. We had used it on Scamp. The most difficult part of engineering any hardware-controlled machine was always to get the control system right, and logic simulation was not particularly effective or

much practised at that time.

Emulation of older machines was a late thought, but microprogramming achieved it very well. The emulators consisted of a mixture of microcode and 360 programming: the I/O was always done by System/360 programs. Hardware additions were made to some machines but really very little, and I am not aware of any additions to the model 40 for the 1401 emulator (apart from increasing the size of the read-only-store).

In Wilkes' scheme, control signals for a machine cycle were generated by switching a magnetic core. A wire threaded the cores for all cycles where a signal was required. That developed into a read only store scheme, with a word for each cycle. Instead of taking signals directly to control the data paths, we used decoders for mutually exclusive signals, which was more efficient.

The output from the ROS went into a microinstruction register. Some signals were used early in the cycle and some later, and needed extra registers. Some of the gates related to finding the next address, and the decision where to go in the read only store could depend on bits in the data flow.

Model 40 data flow

The model 40 we built was a machine with four registers which were essentially two-byte registers. The main memory cycle was two and a half microseconds in the end. The cycle time of the machine (and of the read only store) was 625 nanoseconds. We had a local store as well, to which we could read or write in one cycle.

There was a one byte adder and a microinstruction used 16 bits to specify source registers for the two inputs to the adder, what function the adder should use and how to set the carry, whether to use a four-bit shift, and the destination register to receive the output. There was a two-byte data path between the local store and the registers, and we could take something from one of the registers to an R register and then direct it somewhere else. Another set of registers addressed the local store. A microinstruction used five bits to describe the combinations of which one to use and whether to increment it as we went through. A lot of the features we developed, for example the independent carry latches, appeared in very similar form in the other machines which were developed quite independently.

We used the same data flow for I/O operations. The channels used that

data flow for the complicated bits of starting and changing operations, but they tended to have independent hardware for just transferring data. We were very conscious of the number of words that we had available in the read only store, and saving was very important. One had to preserve a balance between making code as compact as you possibly can and making it possible to change if there was a mistake. It was 'all-in' programming.

Following System/360

In the 20 years following the design of System/360 it was extended in many ways. Notably in about 1970 IBM introduced dynamic address translation. The restrictions on aligning fixed and floating point operands on word boundaries were relaxed. Instructions were added to extend the maximum character string length beyond 256; 'compare and swap' is a more powerful form of semaphore-type operation for synchronising processes when multiple cpus were present. Later on, around 1980, IBM introduced the XA architecture with 31-bit addressing and improvements in numbers of I/O channels and device addressing schemes.

As well as introducing the word 'architecture', many of the features in System/360 became standard concepts: hardware compatibility, microprogramming, emulation, channels and a standard Input Output Interface, Program Status Words, Condition Codes, 64-bit words. Practically every machine is now based around eight bytes and the language of dumps has progressed from octal to hex.

Editor's note: this is an edited version of the talk given by the author to the Society during the IBM 360 seminar at the Science Museum in November 1995.

IBM 360 input and output

Michael Flinders

This article describes the design of the input and output subsystem on the System/360, starting with the principles underlying the architecture, looking at the system elements and their role in the transfer of data, and concluding with a discussion of the practical aspects of designing the I/O channels of the 360/40.

It was in 1962 that I began to get involved with the architects of System/360. I feel very fortunate to have been able to work with such an outstanding group of people as Fred Brooks, Gene Amdahl and Gerrit Blaauw.

Our design of the model 40 progressed faster than all the other machines. When the architects had a problem and wanted to bounce it off the hardware designers, it was to us in Hursley that they came. So we tended to have more interaction with the architects than the people designing the other systems. And when those engineers hit a problem they could be fairly confident that we'd already solved it. So we also had more interaction with the designers of the other systems, who were based at Endicott and Poughkeepsie.

A consequence of this was that I worked very closely with the I/O architect, whose name was Andris Padegs. Andris was a Latvian who had emigrated to America in his teens: he had a thick Latvian accent and so was not an easy person to understand. But he had an incredible ability to write concisely and precisely. I commend to you the 360 Principles of Operation for the quality of the technical English, in terms of the way it expresses complex concepts very tightly.

The first principle on which the IBM System/360 I/O architecture was founded was that the central processing unit and the channels should be able to operate completely independently, so that I/O operations could be in progress moving data from devices into memory at the same time as the cpu was executing its instruction stream. Another principle was that multiple I/O operations should be able to take place concurrently. The third requirement was that every I/O device should attach via the same type of plug and socket and logical connection, which became known as the standard interface.

The two basic components of the I/O system were the channel — the

box of logic which talked to the devices — and the standard interface — the bunch of wires which connected the I/O devices to the channel. As for the peripherals themselves, there were two broad classes: those which were single devices like a printer, where the printer mechanism was driven by a control unit that talked to the standard interface; and multiple device I/O subsystems like tape units, where you might have a bank of tape units all connecting to a single control unit which connected to the interface. Communications subsystems were another example of a multiple device.

The sequence of an I/O operation was that the central processing unit would be proceeding through its instruction stream when it came to a "start I/O" instruction. It would then pass to the channel the number of the device that it wanted to start up, and the channel would then go out across the standard interface and issue the relevant command to that device. The cpu would then be released to continue processing its instruction stream.

Sometimes the device would be busy, or there might be some condition which prevented the operation happening, such as a paper jam on a printer or a broken tape. In that case the channel would tell the cpu that the I/O couldn't be started and the cpu would return to its instruction stream.

The standard interface was, I think with hindsight, very conservatively designed. The channel had a group of control lines which went out to which all of the devices listened: these line were called Address Out, Command Out, and Service Out. There was also a data bus which went out from the channel: one example of the conservatism of the design is that there was a parity bit carried along with the eight data bits on the bus.

The control lines — address, command and service — stated what type of data the channel was putting out onto its bus. The control units on the interface also had lines into the channel — again address, status and service — as well as a bus into the channel: again the control units qualified the type of information on the bus by activating the appropriate control line.

There was one other pair of lines — select out and select in — which formed the polling mechanism on the bus. Once a selection sequence was started by the channel putting the address on the bus, every device would wait for the select out signal. Each one would, when it saw it, check the address on the bus, and if it didn't recognise it would pass the select out signal to the next device. That was how tie breaking was achieved.

The interface was designed to work in two modes. With slow devices

it could interleave transfers from different devices, or with high speed peripherals such as disc and tape the interface could be dedicated to a single device. The standard interface had 31 coaxial cables, so it was quite a big physical object.

One of the consequences of having a standard interface was that it opened the door to other people who wanted to make input/output devices to attach to IBM systems. Indirectly it gave rise to the plug-compatible industry.

The interface was a totally asynchronous interface — all transfers involved handshaking. So a device that wanted service would raise "address in", and then would wait until it saw "command out" from the channel before proceeding. When that happened, the device would drop "address in", and the channel would then drop "command out". At this stage there were only zeros on the bus, which was the sign for the device to proceed. It would then raise "service in" (saying it was requesting a byte of data), and the channel would raise "service out" when it had put that byte of data on the bus. So it was a completely asynchronous interface.

The channels were capable of executing a rudimentary channel program, consisting of a chain of channel control words (CCWs). To start, an I/O instruction specified the address of the device that was to be the subject of the I/O operation. The address of the first channel command word was placed in location 72 in storage. (This was a fixed assignment in storage, decided by the architects.) That was the pointer to the channel program, the string of control words.

When an I/O operation terminated, the status of that I/O operation was dumped by the channel into location 64 — eight bytes which gave the address of the last command word that had been executed, any residual count if all the data hadn't been transferred in the CCW, and various flags which defined the status of that operation.

Those two memory locations, 64 and 72, were quite hard worked because they served for all the channels in the system. I think they subsequently turned out to be something of a bottleneck in the architecture.

The fact that there were basically two broad classes of I/O devices — slow byte-by-byte units and high speed peripherals — really dictated the development of two types of channel. One was the selector channel which on the large machines was an independent, stand-alone box: it would talk to one disc or one tape at a time. The other was the multiplexer channel, which multiplexed data transfers between many devices.

The information that the channel stored about the I/O activity taking place on its interface was held in what was called the sub-channel information, and consisted of: the channel control word address for the control word currently being executed, or maybe the next one to be executed; the address in memory where data was being moved to or from; the type of operation (input or output); and the count of the number of bytes that had been transferred during that operation. The channel was continually updating this information during an I/O operation.

The selector channel only needed one sub-channel, because it only executed one I/O operation at a time. On the multiplexer channel, you had up to 256 operations concurrently. We found that the most economical way of storing the sub-channel information on the model 40 was in the "bump-on" memory. The "bump" literally was a bump: it was just some extra cores threaded on the side of the memory planes with some control logic to make sure that nobody else but the channel could get to it.

Now for some comments on the design of the channels. On the large systems they didn't have the same sort of cost constraints as we had on the 360/40. The selector channels were stand-alone boxes and were shared among the large systems: the same channel design was used on all the machines from the 360/65 through 360/90.

The challenge on the small systems was to make a selector channel which would run I/O operations up to the full bandwidth of memory (which was around 400kb, I think, on the model 40) without using an inordinate amount of hardware. We actually had three goes at this before we got a satisfactory solution.

The first attempt was to put all hardware registers and hardware control in the channel. That proved to be just too expensive, so we abandoned that idea. The second attempt was to use the local store as a buffer and use microcode for controlling the loading of the buffers and for analysing the end conditions.

The architecture was very strict about the channel returning exactly the number of bytes that had been transferred. If it had not, there were two possibilities: either the device had requested more than the CCW said, or it hadn't requested as many. Whichever condition applied, it had to be detected by the channel, and that became very difficult on a microcode control channel using buffering in the local store. So we gave up that idea too.

The third attempt worked very nicely. We kept hardware registers,

we kept hardware addressing and we put a five byte shift register on the interface. The microcode used to load two bytes at a time into that five byte register while the device was taking bytes out of the end of it. When the count went to zero we had a flag which travelled down the buffer with the last byte of data so that we knew whether we had given exactly the right number of bytes or not. That was very manageable.

With the multiplexer channels, the large systems again used a single stand-alone hardware box. The model 30, the model 40 and the model 50 all used the cpu data flow as their channel and the multiplexer channel actually had no data flow at all of its own. We added about 50 logic gates over and above what was needed for the cpu for the multiplexer channel: that was just logic to detect the tags on the interface.

The operation of the multiplexer channel on the model 40 was that when the "address in" line was detected on the interface, that triggered a branch in the microcode at the end of the current memory cycle. (We were never allowed to interrupt during a memory cycle, we had to wait until the end.)

Addressing then forced the branch in the microcode to a routine which took seven cycles to store in a local store all of the cpu registers. Then there was about 26 cycles of microcode where we fetched the CCW information out of the bump for the device that was requesting the service, updated it, put a byte on the interface and then at the end of that sequence there was another seven cycles where we restored the data flow back to its previous state. The cpu instruction that had been interrupted then carried on working.

So the cpu in the model 40 was continually changing hats. I think we could have up to 128 sub channels, so you could theoretically have about 128 I/O operations and the cpu instruction stream sharing the same data flow.

The time on the multiplexer channel was around 4.5 microseconds to dump the data flow, another 4.5 to restore it and about 16 microseconds to process a byte.

On the multiplexer channel we tended to think in terms of the interference of I/O activities with the cpu operation. If you had 10 kilobytes per second I/O activity on the multiplexer channel, that equated to about 25% interference with the cpu, meaning that the cpu was effectively running at about 75% of its speed.

Editor's note: This article is based on a talk given by the author to the

Society at the IBM seminar held at the Science Museum on 21 November 1995. Michael Flinders joined the IBM development project at Hursley in 1960 and worked there as a logic designer and microprogrammer.

Letters to the Editor

Dear Mr Enticknap,

This is a rather long footnote to Raymond 'Dickie' Bird's splendid article in *Resurrection* issue 22. It is always fascinating to see what life was like on the other side of the competitive fence.

In the autumn of 1958 I spent six weeks at the Indian Statistical Institute preparing a quotation for a Pegasus 2 system to do the Institute's work. This was at the request of the Commonwealth Relations Office, who were fed up with the dominance of the Russians in Calcutta at that time. The Sputnik was up and Russian prestige was very high. Central planning of economies was all the rage and the MacDonalds Hotel in Calcutta was said to be full of Russians.

The Institute's main task was to assemble economic statistics from a network of sample villages all over India, thus avoiding the massive paperwork involved in an exhaustive census. There were about 400 graduates at the Institute.

The Institute was founded by PC Mahalanobis FRS, the great Indian statistician and adviser to Nehru's government on economic planning. He had married a daughter of the poet Sir Rabindranath Tagore, and based the Institute in Tagore's palatial house near Dum Dum (Calcutta Airport). The house was at one end, there was a lake in the middle and a new block with air conditioning at the other end. This used water from the lake for cooling.

Also there was a Russian Ural computer, which was definitely not working. It had laid for a long time in zinc-lined cases in Calcutta Docks. Eventually they got it working. However, disaster struck during the monsoon when the inlet pipe from the lake to the air conditioning got blocked with weeds and the computer was soaked in fog. The engineers were there for months replacing all the germanium diodes.

The Ural was a very strange machine. Words were stored in quarter words at 90° intervals, so there was no way you could read a word in less than three quarters of a revolution. It had perforated cinematograph film for input and output. The order code was available and I succeeded in teaching it to a visiting statistician.

The team of about a dozen Russian engineers — one with his wife and family — were resident in the building. We all messed together in the

same dining room with the Russians on two sides and the rest of us on the remaining side. Furthest from us was the interpreter, generally thought of as a KGB man. The Russians were charming and very English in temperament. At half time the interpreter was changed from an English speaker to one speaking Hindi.

In the background to all this was a sturdy HEC 2M which really did very well. I got to the point of running some simple code on it. It was certainly the only working computer at the Institute for a long time. I have a silver salver on our hall table given to me when I left the Institute after six weeks. It is dated 15 November 1958, and I am sure the machine was working then. I do not know the HEC 2M's ultimnate fate.

Yours sincerely,

Conway Berners-Lee
London SW14
28 August 1999

Dear Nicholas,

The article "BTM's First Steps Into Computing" mentioned briefly the HEC 4 (1201) specifically designed for commercial work. One of the first of these was delivered to the Ministry of Supply at Chessington, Surrey, for payroll work on 29 August 1957. (To the best of my recollection the very first deliveries were to Morgan Crucible Co earlier that year and to the Ministry of Agriculture, Food and Fisheries at Guildford, Surrey, just before our own, for agriculture subsidy payouts and payroll).

The first live Ministry of Supply payroll was produced in June 1958, and Chessington has been turning them out continuously ever since — but not on 1201s!

Yours sincerely,

Cecil Marks
Banstead, Surrey
15 September 1999

Editorial contact details

Readers wishing to contact the Editor may do so by fax to 020 8715 0484 or by e-mail to NEnticknap@compuserve.com.

Computing anniversaries in 2000

Once the millennium hangovers have cleared, it will be time to start planning the rest of the year's festivities. Here are some of the landmark events in computer history that people may be celebrating during the course of 2000.

70th anniversary The beginnings of the computer age could be vaguely discerned when Vannevar Bush built the first differential analyser in 1930.

50th IBM had seen the light by 1950 when it announced its first computer, the Defense Calculator (later renamed the 701).

45th BTM became the first of the British office automation companies to enter the computer age with the delivery of its first computer, the HEC 2M, in 1955.

40th Business computing moved into the mainstream as Cobol first came into use in 1960.

35th The computer industry moved a step forward as IBM and ICL both delivered the first models in their first computer ranges, the 360 and 1900 respectively, in 1965.

30th Two of the most important computer launches in history came from Digital Equipment with the PDP-11 and IBM with the System 370 in 1970.

30th IBM became the first company to use modern solid-state technology for main memory with the 370/145 later in 1970.

25th The first step towards the modern computer age was taken with the formation of Microsoft in 1975.

15th The growing importance of standards and open interfaces was symbolised by the formation of the X/Open Group in 1985.

10th EMC launched Symmetrix, the first Raid disc subsystem, in 1990.

5th Hitachi announced Skyline, the last mainframe range to feature traditional ECL technology.

Bombe Rebuild Project

John Harper

The major addition to the rebuild since my last report has been the gate, or Jack Frame as BTM called it. This is a large square item over 4 feet square which is mounted on hinges and can be swung open 90° for servicing purposes. The inside carries 104 sense relays, 40 control relays and various rectifiers, resistors and capacitors. On the outside or rear of the machine are to be mounted the 228 Jacks into which the WRNS would have plugged up the Menu.

Various other fittings have been made and attached to the gate, the most noticeable being the six sense relay shelves. As before, everything was assembled in AutoCad "on paper" before metal was cut. Minor queries arose and were rectified during this process. As a result, the gate fitted perfectly first time. As a spin off from this exercise there is now a full size drawing of the Jack Area on display at Bletchley Park in the Bombe Hut.

All of our iron castings are now cast and the first, the large main gearbox housing, has gone for machining to the Society of Model and Experimental Engineers in South London. We are very grateful for this help.

We have made a start on metal sheet construction, making such items as fixed Oil Trays, the End Cover and the Switch Panel. The last two items have been test fitted, and have now gone to be painted in crackle black. The whole exterior of the machine will be painted likewise, so we will soon have an idea of the overall finish of the Bombe.

Perhaps a little prematurely, we have started to produce cableforms. As we have 12 miles of wire to convert into cables this will be a long and labour intensive activity. Our first cable has taught us a lot about lacing and routing. We have also learnt that it is wise to use a length of rope to thread through the machine and check lengths and bends before making the actual cableform. In this instance CAD cannot help us as much as it has in other areas because curved cables are not easily described in the three axes. Nortel Harlow has kindly found a basement area in which their volunteers can lay out cables on pin boards.

I am very grateful to Chris Burton for arranging a visit to a Manchester telephone exchange and for helping me to strip out the equipment we needed. We now have enough Jack and Plug contacts to complete the

Bombe.

Our appeal for help with turning in *Resurrection* 22 produced a very good response, and the number of people helping us is now approaching the number we can most efficiently use. We now have four volunteers making turned parts, and progress is very satisfying.

I do however have a new area to tackle which I do not have the skills to approach myself. This is hardening and grinding. A high proportion of the turned parts require such treatment. If anyone can provide expert advice, or knows where we can have hardening and grinding carried out, preferably at below commercial rates, we would be very pleased to hear from them.

We now need to produce a fairly large number of square section spacers. The operation involved is to cut steel bars to length and then drill and tap threads as appropriate. There is little need for specialist tools but accuracy is important and in some cases a drilling jig would be well worth making beforehand. Volunteer effort would be most welcome.

Readers who can help in any way can contact me by phone or mail (see inside back cover) or by email at <bombe@jharper.demon.co.uk>.

Software Preservation
David Holdsworth

So far we only have material for the preservation of George 3. There are offers of other material on 7-track magnetic tape, but we have not yet identified a means of reading this medium.

The state of the George 3 project is described on our Web site in some detail: see <www.personal.leeds.ac.uk/~ecldh/ccs/preserve.html>. The important point is that enough of the system works to give us a clear proof of concept.

The filestore we use was created by deleting all the company specific material from a live store that was being shut down by CAP Gemini. This filestore still contains several compilers, certainly Fortran, Algol 60 and Cobol, but not the seminal Algol 68R.

It is a pleasure to record the co-operation of CAP Gemini (especially Dave Higgins) in the exercise, and also the readiness with which ICL granted the necessary intellectual property rights.

The hope is that we shall be able to preserve other systems, so that the historians (or just a curious public) of the future will be able to understand

the computer scene of the sixties and seventies, which otherwise is in danger of being lost for ever.

Pegasus Working Party

Len Hewitt

Over the last few months Pegasus has worked well generally from first being switched on. Various investigations of some of the added features on this machine have taken place. The capability of having the first track of the drum placed into 16 336-bit long lines has been looked at. This allows the first 16 blocks of the drum to be written and read at delay line speeds rather than the much slower drum transfer speed. About half of the long lines are working but we have not attempted to replace the faulty ones yet. The 6-bit character handling instructions have also been made to work.

The good news is that Pegasus is to go into the Science Museum computing gallery very soon. The plans are being made now and some time next year there will be a fully working Pegasus computer on display. I think this will be the first working 40-year-old computer in any museum.

Simulators

Simulators for a variety of historic computers, including Edsac, Elliott 903, Pegasus, the Manchester University Small-Scale Experimental Machine and Zebra, can be found at our FTP site. Access details are on page 36.

Forthcoming Events

Every Tuesday at 1200 and 1400 Demonstrations of the replica Small-Scale Experimental Machine at Manchester Museum of Science and Industry

8-9 January 2000, and fortnightly thereafter Guided tours and exhibition at Bletchley Park, price £3.00, or £2.00 for concessions

Exhibition of wartime code-breaking equipment and procedures, including the replica Colossus, plus 90 minute tours of the wartime buildings

25 January 2000 North West Group meeting on "Punched Card Machines — their history and applications"

Speakers John Bennett and Hamish Carmichael

25 February 2000 North West Group meeting on "Early Line Printers"

Speakers T Wix and WA Gunn

4 April 2000 North West Group meeting on "The Early Days of Medical Computing"

Speaker Dr B Richards

The North West Group meetings will take place in the Conference room at the Manchester Museum of Science and Industry, starting at 1730; tea is served from 1700.

Queries about London meetings should be addressed to George Davis on 020 8681 7784, and about Manchester meetings to William Gunn on 01663 764997.

FTP, Web and E-mail Addresses

The Society has its own World Wide Web (WWW) site: it is located at **http://www.cs.man.ac.uk/CCS/**. This is in addition to the FTP site at **ftp.cs.man.ac.uk/pub/CCS-Archive** (please note that these URLs are case-sensitive). Our Web site includes information about the SSEM project as well as selected papers from *Resurrection*. Readers can download files, including the current and all past issues of *Resurrection* and simulators for historic machines.

Readers of *Resurrection* who wish to contact committee members via electronic mail may do so using the following addresses.

Chris Burton: chris@envex.demon.co.uk
Martin Campbell-Kelly: mck@dcs.warwick.ac.uk
Hamish Carmichael: hamishc@globalnet.co.uk
George Davis: georgedavis@bcs.org.uk
Nicholas Enticknap: NEnticknap@compuserve.com
John Harper: bombe@jharper.demon.co.uk
Dan Hayton: Daniel@newcomen.demon.co.uk
Len Hewitt: leonard.hewitt@virgin.net.
Dave Holdsworth: D.Holdsworth@leeds.ac.uk
Roger Johnson: r.johnson@bcs.org.uk
Adrian Johnstone: adrian@dcs.rhbnc.ac.uk
Simon Lavington: lavis@essex.ac.uk
Brian Oakley: brian.oakley@ukonline.co.uk
Tony Sale: t.sale@qufaro.demon.co.uk
Robin Shirley: r.shirley@surrey.ac.uk
John Sinclair: john.eurocom@dial.pipex.com
John Southall: jsouthall@bcs.org.uk
Doron Swade: d.swade@ic.ac.uk

Resurrection is the bulletin of the Computer Conservation Society and is distributed free to members. Additional copies are £3.00 each, or £10.00 for a subscription covering four issues.

Editor – Nicholas Enticknap
Typesetting – Nicholas Enticknap
Typesetting design – Adrian Johnstone
Cover design – Tony Sale
Printed by the British Computer Society

Committee of the Society

Chairman **Brian Oakley CBE FBCS**, 120 Reigate Road, Ewell, Epsom, Surrey KT17 3BX. Tel: 020 8393 4096.
Vice-Chairman **Tony Sale FBCS**, 15 Northampton Road, Bromham, Beds MK43 8QB. Tel: 01234 822788.
Secretary **Hamish Carmichael FBCS**, 63 Collingwood Avenue, Tolworth, Surbiton, Surrey KT5 9PU. Tel: 020 8337 3176.
Treasurer **Dan Hayton**, 31 The High Street, Farnborough Village, Orpington, Kent BR6 7BQ. Tel: 01689 852186.
Science Museum representative **Doron Swade CEng MBCS**, Assistant Director, The Science Museum, Exhibition Road, London SW7 2DD. Tel: 020 7938 8106.
Chairman, Elliott 803 Working Party **John Sinclair**, 9 Plummers Lane, Haynes, Bedford MK45 3PL. Tel: 01234 381 403.
Chairman, Elliott 401 Working Party **Chris Burton CEng FIEE FBCS**, Wern Ddu Fach, Llansilin, Oswestry, Shropshire SY10 9BN. Tel: 01691 791274.
Chairman, Pegasus Working Party **Len Hewitt MBCS**, 5 Birch Grove, Kingswood, Surrey KT20 6QU. Tel: 01737 832355.
Chairman, DEC Working Party **Dr Adrian Johnstone CEng MIEE MBCS**, Royal Holloway and Bedford New College, Egham, Surrey TW20 0EX. Tel: 01784 443425.
Chairman, S100 bus Working Party **Robin Shirley**, 41 Guildford Park Avenue, Guildford, Surrey GU2 5NL. Tel: 01483 565220.
Chairman, Turing Bombe Project **John Harper CEng MIEE MBCS**, 7 Cedar Avenue, Ickleford, Hitchin, Herts SG5 3XU. Tel: 01462 451970.
Chairman, North West Group **Professor Frank Sumner FBCS**, Department of Computer Science, University of Manchester, M13 9PL. Tel: 0161 275 6196.
Meetings Secretary **George Davis CEng FBCS**, 4 Digby Place, Croydon CR0 5QR. Tel: 020 8681 7784.
Editor, Resurrection **Nicholas Enticknap**, 4 Thornton Court, Grand Drive, Raynes Park SW20 9HJ. Tel: 020 8540 5952. Fax: 020 8715 0484.
Archivist **Harold Gearing FBCS**, 14 Craft Way, Steeple Morden, Royston, Herts SG8 0PF. Tel: 01763 852567.

Dr Martin Campbell-Kelly, Department of Computer Science, University of Warwick, Coventry CV4 7AL. Tel: 01203 523196.
Professor Sandy Douglas CBE FBCS, c/o AMM Douglas, 7 Sevenoaks Drive, Bournemouth, Dorset BH7 7JH.
Dr Dave Holdsworth MBCS CEng, University Computing Service, University of Leeds, Leeds LS2 9JT. Tel: 0113 233 5402.
Dr Roger Johnson FBCS, 9 Stanhope Way, Riverhead, Sevenoaks, Kent TN13 2DZ. Tel: 020 7631 6709.
Professor Simon Lavington FBCS FIEE CEng, Department of Computer Science, University of Essex, Colchester CO4 3SQ. Tel: 01206 872677.
Graham Morris FBCS, 43 Pewley Hill, Guildford GU1 3SW. Tel: 01483 566933.
John Southall FBCS, 8 Nursery Gardens, Purley-on-Thames, Reading RG8 8AS. Tel: 0118 984 2259.