Computer Conservation Society

Aims and objectives

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

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

The aims of the CCS are to

◇ Promote the conservation of historic computers

◇ Develop awareness of the importance of historic computers

◇ Encourage research on historic computers

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 6

Summer 1993

Contents

Editorial

Nicholas Enticknap, Editor

Welcome to issue 6 of *Resurrection.* As it is a relatively short time since the last issue there is less news to report, but the issue is still a large one - only four pages less than last time's record size. We hope that there is something to grab the attention of most members somewhere in the 36 pages.

As last time, we have included four feature articles, three of which are based on talks given to the Society at the Science Museum.

One, covering Laurence Clarke's recollections of his experiences with the Elliott 400 series, is especially appropriate, as April saw the 40th anniversary of the first public running of the 401, the machine currently under restoration. The anniversary was marked by a commemorative function at the Science Museum.

Over 30 guests attended, including many of those who were involved with the computer (which was the only one of its type made) during its development and operational life. Laurence Clarke himself was there, along with Harry Carpenter, John Coales, William Elliott, John Gower, Peter Holland, John Mullett, Gavin Ross, Andrew St Johnston and Maurice Wilkes. A report on the occasion is on page 6.

Tony Sale's article on Bletchley Park is also timely, as negotiations continue for the acquisition of the Park with a view to establishing a new museum of computing and cryptology on the site. Our Secretary's article describes the activities at the Park during World War II.

Of our two other articles, one, by Peter King,is for software buffs, covering the evolution of the database management system over three decades, while the other is contributed by Society member Peter Excell, and describes his schoolboy experiences of one of the first educational computer systems.

As we promised last time, we have included a report on the archiving work being carried out under the leadership of Harold Gearing, work which complements the physical restoration projects described in earlier issues and which is of no less importance.

Our Guest Editorial is supplied this time by Society Chairman Graham Morris, and reflects on the progress of the CCS so far. It also outlines the areas where help from members is most needed.

Guest Editorial

Graham Morris, Chairman

In the last issue of *Resurrection* Maurice Wilkes emphasised in his Guest Opinion the responsibilities of all of us to ensure that important documents and records and physical items were properly archived and preserved. He also urged us to involve younger people in CCS activities.

I am very conscious of the Computer Conservation Society's responsibilities and opportunities in these matters. In a thoroughly paradoxical way we, of all the specialist groups of the BCS, have the most assured future. Current technologies will inevitably fade and be replaced by even more exciting developments. So, too, will many of the specialist groups, even if only in name.

But in our case, these very changes will provide the vital raw material for our work and ensure our future, provided we remain enthusiastic and responsive. And for the CCS, unlike the other specialist groups, every member of the BCS is a potential recruit, because our field is the whole story and history of computing.

In our relatively short existence — we have yet to celebrate our fourth birthday — our Society has achieved a remarkable amount. Those of you who have attended our In Steam days will have seen the physical achievements in conserving early machines, achievements which we hope will soon find a permanent place in the Science Museum's galleries rather than the "engine shed" round the back. Our latest endeavour, on which substantial progress has already been made under the leadership of Chris Burton, is the restoration of *the* Elliott 401.

Our lectures and seminars too have provided fascinating insights — and hindsights — of a whole variety of early developments, and without the intrusion of competitive selling! My only disappointment has been in the attendance levels at our events. They haven't been bad, but the material and speakers have merited much larger audiences.

A recent talk by Vic Maller, for example, dealt with the origins of ICL's Content Addressable File Store (CAFS) and traced its development. CAFS? you may say — surely that's still with us and hardly history? It was that fact that made it such a fascinating occasion. In our field 10 years old is already historic! In May we look forward to a seminar on NPL and ACE. Perhaps next year we might have a seminar on the 360 or the 1900, or even the 2900: "promise and achievement".

None of this will happen without your positive support and involvement, and I urge all our members to play a more active part in our affairs, and to spread the word to others.

There is one particular area where we urgently need help. We already have a large amount of archive material and Harold Gearing, our archivist, working with Susan Julian-Ottie of the Science Museum, faces a daunting task. Recognising that there must be mountain ranges of documentary material building up in our field, we urgently need help in dealing with what we already have.

It would be of special value to have volunteers who were active in the seventies and can thus complement Harold's experience. He describes his work in this issue in the article that follows. Please consider volunteering to help with this important work and get in touch with Tony Sale.

Finally, your hard-working committee welcomes suggestions for future events, especially when they include offers of practical help. Why not turn up at our next meeting and talk things over with us?

The Society's Archives

Harold Gearing, Archivist

As a founder member of the BCS and first joint editor of the *Computer Journal*, who had retired from active 'computing' some years ago, I was honoured in May 1991 to be invited by the CCS Committee to become its Archivist. This article is a preliminary report on our work which is now making good progress. In a later article, I plan to describe in more detail how the material has been indexed, where it is filed, and how access may be obtained to it by researchers once the Index is available.

By the end of 1992 the Society had received over 2000 documents, mostly relating to the period up to 1975. This includes original papers as well as published manuals on the early machines. It relates to all machines in use since the end of the Second World War, and includes surveys, research papers, and accounts of the personal experiences of those who developed both the hardware and particularly the operating systems and other early software. Material continues to arrive and will be added to the Index as soon as possible.

In October 1992 the Science Museum Registrar recruited an additional assistant on a temporary contract, initially to index archive material relating to the machines restored by the Society and intended for future display in the gallery under Curator control. To assist her the PC terminal in the CCS project office was linked to the Prime computer system dedicated to Science Museum records, under control of a manager covering those wider responsibilities. This has made available to the Society a fuller system than we had first contemplated when working with the PC in isolation.

Indexing of all material relating to the two systems already restored, the Pegasus and the Elliott 803, and also of the limited material so far received on the Elliott 401 which is currently being restored, was well advanced by the end of February. The material is being stored in boxes of Public Record Office standard. Eventually the Museum will take control of the storage, as the Curators have a public duty to preserve records relating to historical scientific objects and machines that are on display.

The importance of our Archives will be appreciated by all who want to preserve the record of how we progressed from the mechanical systems of 1939 to the integrated computer systems available just 50 years later. This historical detail can be preserved for reference by future research students only if we are able to complete, with the help of Museum staff, the indexing of *all* the material donated to the Society by members and others.

As a user myself, one who witnessed the progression of business data handling from pen-and-ink loose-leaf ledgers to the comprehensive computer systems of today, I hope very much that we shall be able, in our next report to *Resurrection* readers, to recount that the objectives of our exercise are being achieved.

Meanwhile, may we appeal to anyone closing down an office, or moving house, and wishing to discard documents that are no longer required, please to pass on to us anything of historic interest covering the development of computers over the past 50 years. Anyone who can help should contact the Secretary, Tony Sale, on 071-938 8196.

I would like to take this opportunity to thank personally all those who have donated material to us to date.

Editor's Note: voluntary assistance for Harold and the Museum staff with their ever-growing workload would be greatly appreciated. In particular, they would welcome help with documents relating to the post-1975 period. Any member interested in helping out should contact the Secretary on the number given in the text.

40 Years of the Elliott 401

The Society held an evening reception in the Fellows Room of the Science Museum on 22 April. It marked the 40th anniversary of the first public running of the Elliott 401 at the Physical Society Exhibition in 1953.

Various parts of the machine, now restored to pristine condition, were on display. They included parts of the processor cabinet and one of the plug-in boards (an and-gate), but it was the drum, installed in around 1955 to replace the original disc, that attracted the most attention.

A cursory glance immediately showed how different things were in the fifties. The drum surface is exposed to the open air, which would be unthinkable today. A Heath Robinson-ish touch was the clinical thermometer located in a holder beside the drum: this was needed because variations in temperature had a significant effect on head clearances.

The formal proceedings were opened by the Director of the Science Museum, Neil Cossons. He thanked the Society's Working Party for their efforts in restoring the machine, noting "It would be quite impossible for us to do what we are doing without the sort of help you can provide". Dr Cossons also paid tribute to Douglas Rees, the manager of the machine throughout its time at Rothamsted, who was unable to be present at the reception.

William Elliott ran over the history of the 401, from the design process via its appearance at the Physical Society Exhibition "where it worked reliably for a week" to its operational life at first Cambridge and then Rothamsted and finally its retirement to the Science Museum in 1965, after "playing its own funeral march at the retirement ceremony".

Gavin Ross represented Rothamsted, where the 401 spent most of its working life. Rothamsted, he revealed, is the oldest agricultural research station in the world, being well into its second century by the time the 401 arrived, and distinguished as "the birthplace of modern mathematical statistics".

Tony Sale, whose many responsibilities include the role of the Science Museum's Project Manager for the 401 restoration project, observed that the work was "an exercise in information gathering and collecting as much as on the hardware side". He paid tribute to Working Party chairman Chris Burton and also to the Science Museum staff responsible for the conservation work, Micky Box and Helen Kingsley. Roger Johnson concluded the formalities on behalf of the British Computer Society.

The Enigma of Bletchley Park

Tony Sale

Bletchley Park is situated in what is now Milton Keynes by the town of Bletchley. Its major claim to fame is the role it played in cracking the Enigma code.

Enigma was a machine designed and built in the early 1920s. It was first developed for bank communications, but was rapidly seized on as a potential source of enciphered communication by the German military.

With fast moving tank blitzkrieg operations you need communications: because it's moving it has to be radio communications, and since radio communications can be intercepted the communications had to be enciphered. So Enigma was adopted as a method of passing information in a secure manner by radio.

How Enigma worked

The Enigma machine had a typewriter keyboard for entering messages. When you press one of the keys it makes a contact, and an electrical circuit is formed through a rotor system — a set of numbered rotor wheels. Each rotor wheel is wired from front to back. There's a set of contacts on the front of the rotor, and there's a spring-loaded set of small plates on the back. The wiring between the front and back is scrambled.

Each rotor is cross-wired differently. But every rotor with a given number on it has to be the same, otherwise you can't use them in different machines. So the Germans included three rotors (later increased to five).

The battery-powered electrical current goes in at the entry plate, through the cross-connections in each rotor and then to a reflector plate. It comes out of another contact on that plate, back through the rotors and then lights a lamp.

So when you press a key, a lamp lights. If you press the same key again, then a different lamp lights. This is because pressing the key also activates a mechanical connection which turns the first wheel round one position. After every 26 positions the next rotor is turned round one, and after 26 turns of that wheel the final rotor is turned round once. So every time you press a key the actual electrical circuitry is changing.

The number of combinations you can get by just that method with the rotors is quite high, but not astronomically high. The Germans felt insecure with only that number of combinations, so they introduced another variant called the Steckerboard.

This device consists of a series of jack plugs with two pins on each end: they make connections in the base which transpose a key. If you stecker R to A, for example, then every time you press R it is as if you had pressed A; and when the current emerges that would have lit the R light it lights the A light. So the Steckerboard transposes both the keyboard and the lamp.

Initially there were 10, later 16 stecker wires which you could plug in. That raised the number of combinations for any one key depression to 10^{20}.

A further complication was that the Germans used a double encipherment of the key for a message.

They started off with the sheet of the day, which was labelled by day and month down the side, and told the operator which of the numbered wheels to put into the machine on the day in question.

He picked one out of a box of wheels and placed it into position. Before he did that he had to do the "ring setting" within the actual wheel.

It was possible to turn the central barrel which contained all the cross wiring in relation to the output wheel on which the letters were marked, and also more importantly, where the carry was set. The outer wheel had an indent on it which was picked up by the mechanism and transferred to the next one. So by altering the position (which you did by pushing out a little finger on a catch) you altered the position at which it is transferred the carry to the next wheel. That you had to set on a numbered basis.

So you set your wheels to the sequence of numbers, you put the wheel in the machine, and then you started doing the Steckerboards according to a table which gave you the connections for each particular day. You had to chose one of two steckers depending on the time of day, before 1500 or before 2300. Having done all that you then had to know which radio net you were on, and that gave you the starting wheel positions.

So if the radio net list gave the positions for the day as K-I-M, the operator used a finger or pencil to rotate the wheels until the letters showing in the window at the top of the machine read K-I-M. Everybody on that net would put their machines to the same starting position.

But now the operator selected a three letter key of his own for the particular message he was going to send. Supposing he chose A-S-D, he would type the letters A, S and D in turn, and his oppo would note down which lamps lit as he did so. (A German command post housed a pair of operators, one of them pressing the keys and the other one calling out and noting the lamps. There was also a radio operator ready to send the morse out when the message has been completed.)

So our operator typed A-S-D, and then, because there might be interference problems in the transmission, he did it again. The oppo once more noted which lamps were lit. Having sent the second sequence the operator reset the wheels to A-S-D and then typed out his message.

First attempts at codebreaking

The double encipherment of the message key actually proved to be the Achilles Heel of the German system.

The Poles started working at intercepting German traffic and breaking their ciphers earlier than anyone else, in the mid-1920s. They were very successful until they started receiving Enigma transmissions — that is, ciphered transmissions in morse which were obviously coming from a machine. They were considerably helped by information on key settings supplied by an agent to the French and passed on to them.

They quickly realised that there was a pattern of three letters repeated. This drastically prunes the number of possible combinations. They were able to work out a method which enabled them, after intercepting about 80 messages on the same net, to extract the starting position and thence the message key. By 1935 the Poles were breaking into a large number of German transmissions and were way ahead of anybody else.

When war became imminent in 1939 the Poles realised they had to give the secret away. By then they had built replicas of Enigma. This was a really phenomenal feat of intellect; they had actually deduced the internal wiring of the wheels from the breaking of the messages.

They built three or four models in all, and before they were invaded they gave one to Britain, one to France and took one with them into France. One was brought back to Bletchley Park, and it suddenly spurred everything into top gear there.

The Poles also used a device which they called a bombe. The reason it was so called is obscure: one theory is that they had the idea while eating ice-cream. The bombe was a semi-mechanical device for working through the combinations of the wheels in the Enigma machine.

Bombes

When news of the Polish work reached the UK, Alan Turing, Willy Knox and others at Bletchley Park started to devise new ways of breaking the codes. One of these was the British bombe. This performed a straightforward exhaustive search, but of course it was not possible to do a complete search of 10^{20} combinations; with a mechanical machine the war would have been long over before it even got through the first part of it.

The Germans, though, being human, did a lot of silly things. For example, when they selected their individual message key they used combinations like A-S-D or Q-A-Z or W-S-X which are next to each other on the keyboard, or they used the first letters of their girlfriends' names.

The codebreakers in the Park were able to pick up this sort of repetition, and that enabled them to reduce the search effort considerably. The other way they got into breaking the code was through cribs.

The Germans used very stultified forms of communication. They had to have their full rank, name, number at the top, and they had to sign off in standard way at the bottom.

The codebreakers' favourite German was a poor officer out in the sticks in the middle of Germany who was running a base stores. He had to make a return every week on a certain net and a certain frequency, and most of the time he made a nil return — nothing to report.

So they had, coming through regularly on the same frequency, a message of the same length, and were able to deduce that it was their friend transmitting his usual message. Now having got a crib you can reduce the search tree, because you know that the ciphered text is probably going to transcribe into that crib. By selecting a number of cribs you can create what they called a *menu*, and that is what went into the bombe.

By putting in what they thought was the starting position, they could then go through a cyclic process to see whether there was a combination in that menu that produced the expected German text. If that came out the machine stopped, and when it stopped the girls got a *drop*.

They would then pass the possible key settings (it still wasn't proven at this stage that they were the actual key settings) into the machine rooms in the huts, and there they would use a type X machine. Its operator would set up the key settings and start keying in the ciphered text and see if plain German came out. More often than not it didn't, but she was then able to try a few settings either side, and very often that would result in a break.

The bombes were dispersed all over the country. Wrens were brought in and trained on the machines in the Park and then sent out to places like Eastcote and Stanmore. They were able to use parallel processing; they could put the same menu out to a large number of different bombe sites if they were particularly keen to break it, and so work in parallel.

The volume of traffic intercepted was quite phenomenal. Early in the war they were breaking something like 900 messages a week, and by 1943 it was 84,000 Enigma messages a month.

One of the most impenetrable German ciphers was the naval cipher. This Enigma machine started off with three wheels, rapidly went to four and eventually five wheels because they were worried about interception. These machines were carried on the U-boats, which used them to send their position back to base, and to receive information in enciphered text on the positions of the refuelling ships.

So it was important to get into this and although eventually there was a breakthrough it wasn't particularly deep, but it did begin to stop the losses in the Atlantic. Then the U-boats changed over to the five wheel machine — the Park called that the Triton code — and that was the one which Turing broke.

Overall operation

Here is an overview of how the whole operation fitted together. The German messages were intercepted in a Y station in morse code. The message was then sent to Bletchley Park either by teleprinter or by motorbike. There were 30 bikes an hour coming into the Park from all over the country. Also there was an enormous teleprinter installation, with 1200 teleprinters. In one room known as the Hell Hole there were 326 teleprinters and they had to get BBC sound engineers to soundproof the room to make it possible for the girls to work there.

When the message reached Bletchley Park it went to the registry. The booking in had to be meticulous, because often the smallest clue could cause the break into a key. Then the message went to one of two pairs of huts, numbered 3 and 6, and 4 and 8. The former were for Army and Air Force messages, while 4 and 8 were the Naval ones.

The huts worked in pairs. For example, the initial finding of the keys would be done in hut 6, and then the message would be passed to hut

3, where it would be unscrambled from the enciphered text into German. Then the German would be translated, and it was the job of hut 3 to garnish the information and if necessary to use the index to find out any supporting facts, or other ways of finding out what the message meant.

At that stage the message was ready to be acted upon. Welchman and Winterbottom set up a system to control the dissemination of the deciphered information: they were aware that if the Germans suspected that the Enigma key had been broken, they would have changed the system. So they went to great lengths to protect *Ultra* — the word used for the intelligence coming from Bletchley.

One of the ways they did this was to use the SLUs (security liaison units). The SLUs worked alongside the field commanders, and had direct access to them, much to disgust of some of the junior ranks who couldn't understand why this very lowly captain could go and see Montgomery at dinner and take him away and give him a message.

They had to make certain that the information went only to the commander, and that he destroyed it after reading it. Also, the SLUs had to paraphrase the information so that it could not be attributed to Ultra.

Some of the lengths they went to conceal the source of Ultra were quite remarkable. In the Mediterranean campaign particularly they had great success in reading the signals which said which convoys were leaving to supply Rommel, at what time, and what was on each ship. To protect the source, there was an absolute rule that a convoy could not be attacked until a spotter aircraft had flown over it which could have seen it.

In one case where there was a very thick fog, so that it could not be seen, they were still allowed to attack — they sent a message to a fictitious agent in an Italian port, thanking him for his information about the convoy. This was sent in a code they knew the Germans could break.

Those were the sort of lengths taken to protect the source of Ultra, and it worked. The Germans had their suspicions — they had three audits during the war but each time they concluded it was unbroken and unbreakable. They believed that because it was a machine it couldn't be broken, and they were right; if it had been used correctly it was unbreakable.

Heath Robinsons

That is the story of Enigma, but Bletchley Park was about a lot more than just Enigma. Another German machine used an automatic method of sending cipher signals based on Baudot code and the 5-hole code on tape. It was automatic in that the operator pressed the keyboard and an enciphered punched paper tape came out, which was then put into a high speed morse transmitter. It was called the Geheimschreiber.

This traffic was mainly on landlines, and there wasn't much chance of intercepting them. An exception was in Norway where one landline went through a place where the Norwegian intelligence managed to get a tap onto it, and they got an enormous amount of text out of that.

That was useful for Bletchley Park to start working on. Initially they like the Norwegians used manual deciphering methods, but soon people like Max Newman started devising electromechanical methods.

Newman first designed a method of producing a key tape which simulated the action of the wheels in the original machine. Then he produced what were called the Heath Robinsons — machines with about 80-85 valves which compared the key tape and the tape with the encipherment on. It was a hit and miss affair, because you had to get the tapes exactly in synchronism and keep them there. It wasn't very successful, but it did demonstrate that this was a possible line of progress.

Colossus

There was a group of Turing, Max Newman, Donald Michie and Good who set about producing the designs for Colossus.

The great advance with Colossus was that it used a single tape. Whereas the Heath Robinsons had to have two tapes synchronised, Colossus just ran the intercepted ciphered text in 5-hole code continuously round, and the sprocket readings from that synchronised the whole machine.

The tape was read at 2500 characters a second. It's a great pity that this wasn't known about because it took ordinary computers a long time to get back up to that speed again. It was done with continuous reading; there was great difficulty in making it into a loop - there were all sorts of problems with sticky tape. If anything went wrong you got tape all over the room, rather like a mag tape in an uncovered computer system.

Colossus Mark I had 1500 valves. It was a logical computing engine, not a computer as we know it today — it had no stored program and no memory. But it was the first large valve device in the world, and it was working in 1943. It was a pity that it was kept secret for so long because the circuitry could have been divulged with the algorithms being kept secure. There's nothing magic about the circuitry — it's very good but it's not magic.

Colossus Mark 2 was much more sophisticated, and contained 2500 valves. It had conditional expressions in it, and was fully interactive. We're not allowed to say too much about exactly how it worked yet, but it worked interactively with the output between the person at the console and girls; he was changing various settings on the machine interactively in order to converge on a solution.

I do not know the exact figures of the number of Geheimschreiber breaks but it was very significant. What was even more important was that it was high level traffic. The Germans, considering the Geheimschreiber to be absolutely unbreakable, used it for high level information directly between Hitler and his generals (whereas Enigma mostly was fairly rapid day-to-day battle line information).

The Geheimschreiber strategic information was invaluable, particularly after D-day. That was why there was a tremendous rush to get the Mark 2 Colossus installed and running in the Park by D-day in 1944.

That's the story of the Bletchley Park code breakers. It was a tremendous operation. It is almost unbelievable that there were 12,000 people there towards the end of the war. About 8,500 were genuine code breakers, the rest being support and ancillary staff. What is so incredible is that despite that large number its secret was kept until 1975.

This article is an edited and abridged version of the talk given by the author to the Society at the Science Museum on 29 October 1992.

Recollections of the Elliott 400 series

Laurence Clarke

The Elliott 400 series played a pivotal role in the evolution of the computer in the UK from an academic research project to a commercial product. The author was deeply involved concerned throughout, starting with the one-off experimental 401 and going right through to the relatively mass production 405. Here he describes his memories of those pioneering days.

In 1951 I became one of the first raft of new graduates ever taken into the computer industry in this country. It was a very exciting environment, and I don't apologise for being a bit boastful — not on my own behalf but on Elliott's — about the many firsts we introduced.

I was part of the Elliott 401 design team formed by Andrew St Johnston. Others included Chris Philips, our mechanical genius who incidentally had made fire control equipment for the Navy from about 1914; and Derek Stallworthy, who worked on circuits which had first been developed by Peter Atkinson.

John Halsey developed the nickel delay line memories for us, after Andrew did the design in his famous red book. Hugh Devonald, who later went to Pegasus, did software, and I was looking after logic.

In 1951-52 there were three new technologies which allowed Andrew to design the 401, and do it as a potentially low priced viable commercial computer. They were magnetic disc (later turned into a drum), the miniature twin triodes and the nickel delay lines.

My first task was to take the existing Owen circuits and re-engineer them using the new and much cheaper 12AT7 valves. These were twin triodes, although they may have been a little bit less reliable than the previous valves.

The Owen circuits were based on a delay circuit which was very determined. The input was clocked: this caused a valve to make an inductance, which then charged up a condenser. At the end of the next clock period a reset pulse sucked the charge out very rapidly, and that left you with a totally determined pulse.

What this enabled us to do was to put it together in very standard forms with a number of other devices — AND-gate, OR-gate, digit delay, coincidence gated converter. These were combined to produce a gated delay — the standard circuit which you could then turn into what in other computers of the day was a flipflop.

I think this activity was the beginning of the end of suck-it-and-see electronics. When I first joined Elliotts, there was an awful lot of "Oh that resistor doesn't work, it isn't quite right, we'll put in a bigger one." The quick prototyping of software today is perhaps a little like that. You don't actually analyse the thing completely and then build the circuit.

At Elliotts we started to work in a more methodical fashion. We analysed worst-worst cases for the logic loading rules. The inverter had a resistor chain down and that was analysed very carefully. I know that because I did it myself.

A little time later there was a paper by Dummer of RRE describing this work, but we had done it earlier. It led incidentally to rubber stamps; these blocks were so reliable you didn't have to worry about the circuitry. Just as today people use integrated circuits or whole computers as blocks on a diagram, we used these packages as blocks, as did the Pegasus team later.

When NRDC made a big visitation we had to have something to show them. I had made something which was called the snarc — the short nickel line accumulator calculator. This was I think the first time most of the NRDC people had actually seen something that perhaps they could believe would turn into a computer in the long run.

We can boast another first here. Elliotts were heavily involved in radar, and had a radar research lab. I think possibly the first digital instrument may have been the use of a snarc in calibrating the quartz fibres in torque vane wattmeters. They needed a very accurate timing of the swing of a pendulum on a quartz fibre, and with a photocell we did that exceedingly accurately, probably even more accurately than with the snarc.

On the question of tolerancing and quality, before we made the 401 at all we built one cabinet of equipment which we called the *interference test set of use.* This was able to do some very elementary test cycles.

We were then able to go around with Tesla coils hammers and so on to try and introduce interference. This worked pretty well and in fact we had the thing working for a three week period at one stage with no errors at all. I don't think it worked that well thereafter. When the arithmetic unit eventually got put together with the control, everything was going very fast.

Incidentally, Andrew St Johnston said in April 1952 that the pilot model was supposed to be running by September. I think that was quite remarkable. It wasn't running by September, but was first demonstrated in April 1953 at the Physical Society. We were working it very fast and we got the arithmetic logic together before the disc was available.

So we then used the interference test set to check out that logic and it was one of those times, working very late into the night with Hugh Devonald, that we both learnt that to blame the machine is not the right thing.

We were testing the multiplier. We only had binary switches to set up numbers, so looked for an easy number to set up. We chose one-third, which was 101010101. If you square that you get $\frac{1}{9}$, 111000111000111000. We pressed the button and there was the answer. Then we looked again: in the middle there were four zeroes, not three.

We checked everything. At midnight we had got nowhere and went home, but when we arrived in the morning we both said "Got it!". If you have 101010101 curtailed to 32 bits and you multiply it, you shift the left side half of your result by one place to the right.

Andrew has said the 401 was the first computer to be exhibited, and I believe that to be true. After that exhibition when we were all exceedingly tired, we were entertained by the directors of Elliott Bros. I only realised much later how prophetic those directors were.

I remember well Harry Carpenter and myself being addressed by Dr Ross, the technical director, who was a Hungarian and not easy to understand. I, as a young chap, was tired and half cut on gin, but this man was babbling on at me about "Zee cat crackers" and how computers were going to control catalytic crackers in oil refineries and the like. Remember we're talking about the early 1950s when we weren't doing anything like that.

Later on in the early sixties Elliott Automation was to be the first and I suspect the only UK firm to export computers to the USA for on-line process control, very much through his efforts.

Leon Bagrit, the managing director at that time, foresaw nearly all the impact of computers brought about by microelectronics today in his Reith lecture series in 1964. In fact he expected some of them even earlier than they happened. The old man had a vision that could have served the UK well if only they'd had the commercial and financial stamina to carry it through.

Back to the 401. The machine only had 1000 words: 4K bytes of store in all, with 128 words on each track. You had 13 millisecond maximum access, and therefore as Andrew said you had to have optimum programming. This was considered difficult by some and would be totally unacceptable today, but it did cause real interesting challenges for the kind of people who were programmers then.

The initial 401 didn't have signed multiplication. Just as we were getting it to what we thought was completion, NRDC produced Christopher Strachey who was a real inspiration to us. He very rapidly said "We can do that", and redesigned the multiplier area so we had a signed multiplier. Better still, he gave us a way of doing order multiplication.

These were the two real changes made to the 401 before it went to Cambridge for a short time and then to its final home at Rothamstead Agricultural Experimental Station. Under the devoted care of Doug Rees it worked valiantly until its retirement on July 30th 1965. So it had good long life of very useful work.

In parallel with the development of the 401, Norman Hill, Ed Hursom and Bruce Banbrough in Theory Lab were developing what I think was the first machine — Nicholas — to be made specifically for a computing service. As I remember they'd received a computation contract from the Ministry and they decided that the best way to implement that contract would be by making a computer.

They used the Owen circuits, and found that if they got bits of paxolin and components and took them home, they could wire them up there and bring them back in the morning.

This machine had 1024 words of store, like the 401 but in the form of 16-word nickel delay lines. So they had a minimum access time problem as well, but they believed that by having a single address instruction and taking the next instruction when it became available, they could get two orders per word and thus get more useful programs on the store that they'd got.

The machine did the computations that it was required to do. In fact it ran for many years and was used for lots of other tasks as well as the original contract.

The actual prototype 153 boards were taken away after they started to build the real ones and used by us to make the prototype 402 computer. This all happened at the height of popularity of the Goon Show. So this machine was called Eccles, for Elliott Computer Costing Less.

The only new technologies that we used here, under the guidance of

Hunter Mitchell, was that by being able to use a single package with the line running round the handle, two receive coils and one transmit coil, we could save a lot of space in the machine. We were able to have 16 immediate access registers which enabled us to increase the speed enormously and also to increase the number of B-lines. We went over from disc to a drum, allowing us to have up to 5000 words of store.

I think it's sad that the first 402 delivered wasn't an electronic computer at all; it was a 'calculatrice electronique' installed in the Centre Nationale de Recherche Scientifique in Paris. I rather think it was the first scientific computer ever in France.

The only major enhancement that we made to the hardware was the floating point unit, and that was made for a German firm, Hans Leitz.

In July 1957 we implemented one of the first real-time uses of a proprietary computer. This was used for recovery of fighter aircraft, and was demonstrated to the Central Fighter Establishment. This guided a lot of Elliott philosophy, which was to use general purpose computers in a whole range of process control and military applications. This spirit still exists in many parts of GEC today.

Nine 402 machines were made, which is not a lot. Almost all were still in service in 1969. During that period there were two other much larger machines made. The first was the 403 which was made for the Long Range Weapon Establishment in Australia. It was for the analysis of missile range status. This machine had pipelining in 1955!

It worked this way. We had four word nickel delay lines as the main store so we were getting somewhere near a good size immediate access store — 512 words — but it wasn't fast enough for what we needed. So orders were extracted in the single word lines (assuming no conditional transfer) and started to be decoded while the previous operation was being carried out.

It had of course a much larger magnetic disc, like the 153, and also magnetic tape units. These were manufactured by Pye to a Cambridge University design developed by Donald Wilkes.

The output was offline because there was a tremendous amount of it. The magnetic tape was taken away and fed to Bull line printer, and also to a series of plotters.

I suspect that these were the first plotters to be used for output from a digital computer. They were very crude. They used Mufax weather report receivers, where there was a rotating helix which moved across the paper. An electrical signal was pulsed, and chemically sensitised paper made a mark where there was a pulse and didn't where there wasn't. With a binary disc we knew the position of the helix so we were able to get fairly accurate plotting.

The 403 machine dissipated about 15 Kw. My first task on arriving in Australia to finish the recommissioning was to switch it off, and tell the superintendent that it was staying off until the air conditioning was in proper working order. It seemed likely that the machine would be irrevocably damaged if it carried on in the Australian temperatures that were prevalent at that time.

Colin Reeves of ICI (and later Leeds University) came across a fault in their 402 multiplier. Just occasionally, he said, it was giving the wrong answer.

That had to wait till I got back from Australia seven weeks later before I could do anything about it. We didn't have faxes and phoning was practically impossible. Letters took a long time, and anyway I don't think I could have done it by post.

The problem was purely and simply that he was a very good programmer and he had a shift optimally programmed at the end of an optimally programmed multiplication. He was shifting up to a significant place a digit that should have fallen off the very bottom of the double length accumulator. So when it was a one he came up and looked at it and when it wasn't he didn't, so he got a totally unpredictable result.

Following close on the heels of the 403, the 405 was designed to meet the requirements of the commercial market, which had been explored experimentally using the 402. Andrew and I had the temerity to present a paper in 1954, before the first production of the 402, talking about how we had studied the use of the 402 for the payroll of Mars Co. We did have the grace to say at the end that this was not really a feasible proposition until there was satisfactory magnetic file storage.

The 405 had another first — autonomous data transfer from peripherals (although the term hadn't been coined then). The disc, the film, and the line printer each had assigned blocks of store with dual inputs, and the access from the computer to that store was free unless a transfer had been initiated and was not yet completed at the specific address that was being accessed. Then each device had a completely separate and independent controller.

The film caused all sorts of controversy. People wondered why on earth Elliotts used film rather than magnetic tape. Well, we had quarter inch magnetic tape and it was very unreliable. The reason, we believed, was that the base material in those days was toffee paper. It wasn't an optical medium at all; if we used film, then the coating would go down better and it would be more reliable. So we used magnetic film.

I was asked this question at a Cambridge colloquium at which I talked about the 405 (incidentally, these colloquia should be given credit for bringing together what really was the whole computer community outside Manchester). I was floored by the question "Why do you use film?". Fairthorn of RAE, who was one of the earliest pioneers, adjusted his hearing aid and said in a loud voice, "Because it has the non-trivial advantage of existence." I have since used this phrase often.

Elliotts of course started in the commercial field. Andrew wasn't really steeped in commercial know-how, and neither was I. Yet we and a few of us by then were doing the selling.

So we recruited some people from the commercial world. Later, at a user association meeting, one of the users got up and said to Andrew, "Mr St Johnston, we find that we have to test every film before we use it." Andrew was floored by that observation, but his commercial oppo Ronnie Michaelson (who'd come from Hollerith) immediately got up and said, "Well how many failures have you had?". "Oh, I haven't had any failures yet." So questions can be answered in several different ways.

The first 405 was sold to Norwich City Council for dealing with the rates, a brave step by the City Treasurer, Mr Barnes. There were 30 sold in all, including one to the National Gas Turbine Establishment for trials recording and analysis. This was the first of many Elliott online process systems.

However it soon became apparent that Elliotts (by then Elliott Automation) did not have the financial or manpower stamina to market the product. An agreement was made with the National Cash Register Company that they sold the 405 worldwide outside North America.

It was a very exciting time. The team involved in detail, of only six to 10 people, had developed three very different machines in a 12 month period. This was made possible by the ease of use of the Owen circuits, rubber stamps and all.

This article is based on a talk given by the author as part of the Elliott/Pegasus all-day seminar at the Science Museum on 21 May 1992.

Thirty Years of Databases

Peter King

I shall break the period down into three decades, which fits nicely. 1962 to 1972 I'll call the 1960s, 1972 to 1982 the seventies, and 1982 to 1992 the eighties.

We have to remember the rapidly evolving hardware scene. Processors have ever rising mips ratings: we have raw hardware power tripling every three years roughly. Discs have gone from one or two megabytes to one or two gigabytes and more.

We've seen a change in the way hardware is used. In 1962 we had printed reports, cards going in, files going in and coming out, and data on disc which was referred to and modified. Now we have an enormous number of different ways of using hardware, we've got the person at the workstation with the mouse, and we've got networks.

Some things remain constant. For example, the vision is always ahead of the technology of the time. It's easy to have ideas; you can lie in your bath and think how things are going to run and say you'll have software that does this. Then you try it, it goes like swimming through porridge, and then you get bogged down in matters of optimisation.

We used to think "If only I had some more memory, 512 words instead of 256, everything would be solved". Now with all the windows it's "If only I could have a 24 inch screen instead of a 19 inch one, and two mice." So you're always bumping up against the technology.

The other thing that's always true is that, even when the technology catches up, it's about 10 to 20 years from prophets to profits — from when you start doing the work till you actually have a product that is used and makes money.

This was true of paging systems: it took from 1957 to about 1979. It was true of high level, third generation programming languages. People started trying to write compilers in 1959-60, but I can remember conferences in the early seventies where IBM people would say "Should you use Cobol or Assembler to really get productivity?".

The sixties

I'll start the story with Charles Bachman and Susan Brewer, who produced IDS (Integrated Data Store). They did that because they wanted interrelated files held on disc, for manufacturing applications.

Then there was APL — Associative Programming Language — which was based on PL/I but produced independently by Dodd. That did the same kind of thing as IDS. I suppose it was that that led to the formation of the Codasyl List Processing Task Force.

Memory is essentially a collection of cells linked together with pointers. You can have pointer chains, and follow them around. The argument in the sixties was: wouldn't it be useful in data processing to have the same kind of linking techniques, not between cells in memory but between records held on disc. So let's have a task force to look at it and see whether this can be done usefully. It was called the List Processing Task Force, properly the Codasyl List Processing Task Force - Codasyl was the body at that time responsible for developing Cobol.

In 1967 it turned itself into the Data Base Task Group, but essentially continued with the same work and produced its famous Codasyl Data Base Task Group report, which came out in July 1971.

That was one of the early themes. There was another which I call 'from GUAM and RATS to IMS'. GUAM was Generalised Update and Access Method. The thinking was: Cobol records only allow you a repeating group at the end of the record, and that's pretty limiting. Why can't we have very large Cobol records with unrestricted repeating groups — repeating groups within repeating groups — and hold them on disc? Then let's be able to look at subsets of them. That was a requirement in relation to managing the data connected with the Apollo moon landing project.

RATS — Remote Access Terminal System — addressed what is now an everyday occurrence, getting access to data by a terminal. GUAM and RATS were combined around 1967-8. In 1968 they were frozen with the launch of the moon rocket — even then it was realised you couldn't have people changing the software while you were in the middle of launching a moon rocket.

Then in 1969 IBM released it as a product. This is the one exception to the rule that it's 10-20 years from prophets to profits (although whether IBM made any profit out of the early versions of IMS I know not). It has been improved ever since and is still in existence today.

Another theme of the sixties I call 'Total and Adabas'. Total came first and in my view is the more interesting of the two. It was produced by Cincom, with first release around 1968. The interesting thing is not only the rise of the independent software vendors. Total in particular emphasised data independence.

Then we have a relational model of data for large shared data banks which was an IBM internal research report in 1969, and was published as a Comm. ACM paper in 1970 with a few revisions.

During 1970-72 we had the great debate. The Codasyl Data Base Task Group report put forward the case for a certain type of product for linking records, based on the so-called network model.

IBM had IMS which they didn't understand. This was because there was a confusion between hierarchies which are the result of access. If you were describing IMS now you'd say it enables you to store complex objects with a hierarchical structure, but we didn't see things that way then. So it was a mystery that IMS was marvellously successful in some contexts, and a disastrous failure in others.

It was during this period that Codd's relational model emerged, which claimed to do all sorts of things. It threw the baby out with the bath water, in that it produced such a simple data structure that there were large numbers of things you couldn't thereafter do except by acres of programming. Nonetheless, it was enthusiastically taken up by academics.

So much for the facts: what about the underlying motivations?

The motivation of the people who did APL was that they wanted better data structuring facilities, simple serial files and repeating groups within records. They wanted data structures which would assist analysts and designers to structure files in a more useful way.

You could go in and retrieve a record for a particular customer from the customer file; then automatically pick it up from the order file all the orders from that customer. That was one motivation, which led to programming language features that enabled you to handle them.

There was also the wish for data to be shared, although that to some extent came later. In my view, though, the driving motivation was the idea of logical data independence.

It was characteristic of large organisations in the late 1960s that had been early into computing to have data processing systems with large numbers of programs, say 120 different Cobol programs. All of these were variously accessing 15-20 or more different files.

If you wanted to modify a program, and as a result needed to modify the record structure of a file, you first made the change to the file. You then modified the program, and recompiled it. You then had to recompile all the other programs that accessed the file. That was a tremendous maintenance effort — and then the whole system would crash six months later because you'd forgotten some obscure thing.

I think that was the real driving force for database management software. The object was to interpose the DBMS between the data as it was stored and the data as the program wished to view it. Once you did that you could produce what was essentially a virtual file of some kind, which could remain stable providing you modified the tables which caused it to be materialised from the data that was actually stored.

There were also a number of sub-themes that I call 'dream' sub-themes, for example that data should be accessible to the casual user at terminals — someone, say, who asks "How many red headed under 35 policy holders do we have that haven't had a claim in the last 5 years?".

The seventies

We then move into the 1970s. The Codasyl systems of 1972-75 gradually came into use as did IMS, Total and Adabas. These systems were steadily improved throughout the 1970s and well into the 1980s. They did go some way to solving, or at least addressing, the two main problems of logical data independence and structuring. But they did not solve the problem of schema modification to a running system.

I know of IMS systems in the City that are still running, where everybody knows they've got the wrong schema: if only they could change the schema things would be much better. But they can't actually take them down or reorganise the data.

As a result IBM devoted significant resources to trying to make the relational model actually work.

What the relational model had actually done was throw away data structuring in the belief that thereby you could actually raise the level of programming. There's no doubt that the relational model was good at generating research into locking the problem of sharing data, the problem of recovery, because you have a simple model actually to work with. SQL was developed and is now with us for better or worse; I had a modest little hand in that, how to make the first order predicate calculus understandable to Cobol programmers.

The other fascinating event in the seventies was the charge of the unemployed mathematicians. Relational database theory became a fashionable topic of research: we had normalisations from third normal form, fourth normal form, universal relation, fifth normal form, Armstrong's axioms, lossy joins, lossless joins, tableaus, and so on.

Normalisation theory followed the 80-20 rule: 80% of the benefit came from the first 20% of the ideas. The first 20% of the ideas were essentially second normal form and third normal form, which had become standard and readily understood.

By the middle of the 1970s we had three schema architectures, and models for the conceptual schema. This is interesting because the professional database community was trying to extend its sphere of influence into what previously had been the province of systems analysis and design.

When I designed my first Cobol files back in 1959-60, and my machine code files before that, I never had any doubt that each record represented a real world entity, and that the fields in the record were storing the extensions of functions from that real world entity. It seemed to me such an obvious way to store data that you took that as read.

I thought of it that way probably because by origin I'm a mathematician but other systems analysts do that intuitively. That was the province of systems analysis and design and what we were doing in the 1960s was providing programming tools which would in some way be useful to the systems analysts and designers.

But once the normalisation people got going they extended the province of database into semantic data modelling, the idea of modelling the real world. This systems analysts had been doing for a long time, albeit intuitively.

The eighties

That's where we were through the seventies. Then we come on to the 1980s. The products of the seventies continue: we've still got IMS systems and IDMS systems running. Total systems are dying, while Adabas has developed its own peculiar flavour with its language Natural. We've got Cobol itself now at last in decline.

At the same time we've seen the arrival of the relational systems, principally Oracle, DB2, SQL/DS, Ingres and others. These are developing 4GLs and front ends. In my view they're trying to develop with their 4GLs more than their substructure can bear, because the underlying model is inadequate.

What some of us are doing now is to try to make happen ideas which first emerged a long time ago.

In 1972 there was a paper by Brackey from Milan saying you don't want entity relations. Binary relations are adequate if you represent abstract objects explicitly internally but not externally. Cher Nyson was saying the same thing.

Indeed in 1976 I and Cher Nyson were the two invited speakers at the annual conference of the Italian Computer Conference in Milan. We had dinner with Brackey, and at that time we were talking about having DBMSs based on binary relational models.

A slide I used at a lecture in 1984 says: if we use a binary relational model we can integrate data and metadata. We can have a schema that doesn't distinguish between instances and types, but simply regards them both as objects of the same kind. If we want more elaborate data models we can map them to binary relational graphs, and we can create even more elaborate fourth generation constructs.

All these ideas are very old. Binary relational graphs to represent data were being talked about in 1972 by Brackey, in 1974 by Abrialle, and in 1976 by Cher Nyson and myself.

Mapping them to triple stores was being talked about by Norman Winterbottom in 1978, and Peter Tipman and others earlier, in the 1960s. So there is nothing new in current research; what it is about is taking lots of good ideas that we didn't have the hardware power for 20 years ago, and making them actually happen now. But they haven't happened yet.

There is another theme which is currently confusing matters. About 10 years ago Xerox PARC had a research project which produced the notion of object-oriented programming. I remember vividly John Florentine rushing into my room in 1981, explaining to me that object-oriented programming was the greatest thing since sliced bread.

But if you look back at the database literature of the seventies describing conceptual modelling you'll find the words 'object' and 'entity' are used more or less synonymously. Some of the papers that you think are about entity modelling actually use the word 'object'. The Smith papers on aggregation use the word 'object' throughout.

So 'object' and 'entity' have been used synonymously in many areas, but not in programming. Programming language theorists like talking about objects; they mean bit streams, binary trees, and double length representations of real numbers. So the word 'object' has been used in programming theory as a generic term for a long time, and when it came into object-oriented programming, naturally what they were talking about was a programming construct.

An object in Smalltalk is a piece of store with a number of slots. It's like a Cobol record — you can have pointers and you can have variables. In other words it's a very elaborate type of programming language. What object-oriented programming is about is defining these elaborate data types and defining chunks of code that work on them.

The term 'complex object' is also used now in the database research literature. It's important to understand what people don't understand in the database area. For example you read a lot of papers at the moment arguing that the relational model is very good for business and commercial applications but is no good for engineering applications.

What they need is complex objects like IMS records. They never know that when you try and use the relational model in the commercial context, modelling is really quite difficult. We've got the other source of confusion: 'oriented' and 'orientation 'are ordinary English words with a commonly understood meaning, whereas in object-oriented programming 'object-oriented' is a technical term which means what the object- oriented programming message defines it to mean.

We now have 'object-oriented databases'. What can this mean?

During the 1980s there was one important idea which came into the database area, the idea of persistence (as in persistent programming). The importance of the idea is that you're effectively operating as though your database were entirely in memory. It isn't, of course; your database sits out there on disc and is paged in as needed, but that is entirely hidden from you.

So an object-oriented database is something like Gemstone which provides you with persistent Smalltalk. Any of you who have used Smalltalk will know it's a fine programming language, but use it on your PC and immediately you run out of memory it goes bang, because it's an in-memory system. There's no way of storing any data. So you've got database management software now, so called, things like Gemstone which provides persistent Smalltalk and like Ontos which provides persistent C++.

This article is an edited version of the talk given by the author to the Society at the Science Museum on 16 December 1992.

A Pioneer Initiative in School Computing

Peter Excell

Hatfield School was one of the UK pioneers in introducing computing to the curriculum. The author was one of the first pupils there to benefit from the innovation, and this article documents his experiences.

Hatfield School in Hertfordshire was set up in the late 1950s as a state school with a deliberate bias towards science and technology. The school shared the same campus as Hatfield Technical College, which allowed some sharing of facilities.

An innovative example of this arose when the college installed its first computer, an Elliott 803, in 1963. The head of the school mathematics department, Bill Tagg, negotiated an arrangement whereby pupils had some access to the computer, around two hours per week.

Elliott Autocode

The only programming language available to start with was Elliott Autocode. This had a reasonable instruction set but was hampered by its restriction to one operation per line. All code was entered on five-hole paper tapes using either Westrex teleprinters or 'blind punches', which had a keyboard but no printout. The pupils rapidly became adept at reading the five-hole tape directly.

A major hazard with five-hole tape was the use of 'figure shift' and 'letter shift' characters which modified the meaning of subsequent characters. If these shift characters became corrupted, the meaning of the following characters would be changed into nonsense.

For my first program, I attempted a rather ambitious task: 'cubing the sphere', a three-dimensional equivalent of 'squaring the circle', in which the input is the radius of a sphere and the output is the side length of a cube having the same volume. With beginner's luck my program worked first time and I immediately became a computing enthusiast.

After about a year, the school acquired its own blind punch, a huge boost to morale. These punches had a heavyweight electromechanical action in which the whole tape output device moved up and down with a solid clunk as each character was punched. Later the school devised a scheme to purchase a teleprinter with money raised by parents.

Pupils were permitted to enter the computer room to attempt a run only after the tape had been punched and the printout checked by a teacher. Assuming the system was running properly, we would be invited in to witness the Autocode compiler being loaded via a solidly-built optical tape reader. One could not fail to be impressed with the speed at which the tape was streamed through, especially since errors rarely seemed to occur in this reader.

Then we might be invited to place our program tape in the reader. A button was pressed on the console and the tape was read, pausing at the end of each line to the accompaniment of impressive noises from the loudspeaker on the console.

There was an indefinite delay when an error was detected. It would be marked on the tape with a pencil, and then another button was pressed to continue the check.

Successful input was followed by a 'thinking' phase, after which the program was ready to accept input data tapes. Output tapes would spew out of a tape punch housed in a partially soundproofed box.

The loudspeaker in the console was one of the most memorable features. The basic sound was a pulse sent to the loudspeaker every time a 'jump' instruction was obeyed. We rapidly became acquainted with normal and abnormal sounds.

In particular, a genuinely repetitive sound indicated a program stuck in a loop. However, an iterative routine might involve a loop with data that varied only slightly.

The observant rapidly became intrigued by the row of buttons on the console. These were to remain an enigma until the opportunity arose to delve into the secrets of machine code programming.

Algol

Eventually the memory was upgraded, from 4K to 8K 39-bit words. This allowed the introduction of Algol.

The Algol compiler was on two large reels of paper tape which frequently became rather untidy on their spools. (Tearing one of these reels was one of the gravest sins a user could commit.) Not surprisingly, it took a relatively long time to load, and thus any corruption of the loaded version was a cause of deep frustration.

Algol was a satisfying language to use, having a rational structure and very little machine-dependent dialect. We spent many happy hours studying McCracken's text book[1] and many of us produced Algol programs. The largest (not one of mine) was a collaboratively-written fortune-telling program which took a very long time to compile and run.

Machine Code

Another intriguing mystery was the meaning of the compiler tapes. As we had developed a facility for reading source code tapes by eye, we could immediately see that compiler tapes defied all the rules of correct tape punching we had been taught. Not only were the characters in an apparently random sequence but, more noticeably, they were manifestly lacking any regular punctuation with 'shift' and 'carriage-return-line-feed' characters that were the norm in source code.

Most intriguing of all, the tapes always started with a string of single holes down one side. I and a fellow pupil determined to work out what this meant.

By talking to a friendly computer officer we found out about the bootstrap routine wired into the bottom four words of memory, and we were also shown a machine code instruction set. From this it was possible to deduce that the bootstrap routine acted as a very basic input program, but that the norm on compiler tapes was to use this to load a more advanced input routine some 30 words long, located at the top of the memory.

Armed with this knowledge and with a copy of the machine code instruction set, we proceeded to devise our own machine code program, to be loaded into the top 30 words of store in the belief that this space was unused between tape loading operations. We then created some programs to operate on this principle.

Since we were not offered use of the official machine code assembler (which would in any case have overwritten the high-level language compiler needed by other users) we wrote the programs in binary, then converted them into a list of characters to be typed on the blind punch to produce the ultimate binary tape that was required.

The largest program written in this way was a simple machine code translator capable of reading in sequences of teleprinter-readable decimal numbers, corresponding to the separate parts of an instruction word, and

[1]McCracken, DD: 'A Guide to Algol Programming', New York, Wiley, 1962.

converting them to binary form for output by the tape punch.

The input data thus consisted of a string of 13 decimal digits which were interpreted in five separate groups, as follows:

1. Two digits for the first instruction (00 to 77, in octal)
2. Four digits for the address of the data for the first instruction (0000 to 8191)
3. One digit for the B-digit (0 or 1)
4. Two digits for the second instruction (00 to 77, in octal)
5. Four digits for the address (subject to modification by the B- digit) of the data for the second instruction (0000 to 8191).

This program worked. It could be loaded in the same way as a compiler, by transferring control to word number 0 at the start of the bootstrap routine through appropriate use of the buttons on the console. It produced the expected output from test inputs.

However, it had to be used within a queue of other pupils' jobs, mostly Algol programs. A problem rapidly arose in that the Algol compiler tended to fail after the 'home made' assembler had been used.

Evidently the assumption that the top 30 words contained an input routine that would not be needed during compilation was incorrect. In retrospect, with only 8K words available it is not surprising that the Algol compiler authors re-used this space for essential compiler functions after the bulk of the inputting had been completed.

The B-digit (a name derived from early work at Manchester University) was a neat artifice which added the contents of the address of the first instruction to the absolute address specified for the second instruction in the word, thus enabling relative addressing to be used. The first instruction would normally be null (00) in this case.

In this investigation of machine code programming we had become 'hackers' of an early type, in that we were attempting to get to the heart of the system by an unorthodox route. There was, naturally, no nefarious objective: we took pains to avoid overwriting areas of code needed by other users. We were merely intrigued by the possibility of finding out how the machine worked at its deepest level and how to drive it at this level.

Our teachers, however, were a little alarmed at our activities, and were unsure of the desirability of what we were doing. A Computer Officer from the college was brought in to give a lecture on the principles of machine code programming, but we were not offered use of the Elliott assembler.

Benefits

With hindsight, it appears likely that many of the decisions taken were coloured by the prevalent view that computing was a part of mathematics. Thus machine code programming was not seen as an important activity because (arguably) it was a distraction from the more mathematical activities of high-level language programming. Similarly, I did not give any serious thought to a career in computing because maths was not my strongest subject.

Nonetheless to this day I regard it as one of my most pleasurable and beneficial experiences of computing. Particularly in getting down to the lowest machine code level, I obtained a very clear view of the von Neumann architectural philosophy and of the non-standard design decisions made with this particular computer: the B-digit; the two-instructions-per-word format; and the hardwired bootstrap routine.

I saw the key features of the architectural philosophy as being the use of a single memory for instructions and data, and the use of a single accumulator for interim operand storage. The machine code instruction set introduced me to bit-wise operations, such as left shift, right shift and collate. The timings in the machine code manual clearly illustrated the time penalties with some operations, notably multiply and divide.

I feel the learning of Algol gave me an excellent training in the approach to construction of sound, readable algorithms, and that the experience of machine code gave me a very sound introduction to the basics of computer technology, which has been useful in my limited encounters with microprocessors.

Dr Excell wishes to express his gratitude to Dr Bill Tagg, who made this experience possible and who provided information to supplement the author's own recollections. Dr Excell also acknowledges the collaboration of Richard Driscoll, the pupil who shared the investigation of machine code.

The author is Reader in Applied Electromagnetics, Department of Electronic and Electrical Engineering at the University of Bradford.

Letters to the Editor

Dear Mr Enticknap,

Towards the end of the article on Andrew Booth's APE(X) and MAC 1 machines in issue 5 of *Resurrection*, Andrew Collin wonders what was achieved in terms of practical computing.

If I remember correctly, an engineered version of the APE(X) design became the HEC 4 (Hollerith Electronic Computer) produced by the British Tabulating Machine Co Ltd, one of the predecessor companies of today's ICL. This was one of the earliest machines to tackle office tasks, typically payroll, with the first being installed at Morgan Crucibles early in 1957 and the first Government system in the Ministry of Supply at Chessington in August of that year. The designation HEC4 was soon changed to 1201, the first in the 1200 range which were successful "data processing" machines of their day.

The choice for DP work, between say Pegasus/Elliott on the one hand and English Electric Deuce/Hollerith on the other, was not then solely a question of internal specifications.

Early office applications were usually developed from punch card installations with pre-computer machines of some sophistication, so that it was a natural progression to go for computers which had card input/output rather than paper tape.

Moreover in those days many subsidiary operations which involved large volumes of data had to be done "outside" the computer, eg sorting, merging, interpreting and printing (and machinery for such operations was well established, whereas for tape it was non-existent).

Also, although this was never openly admitted — indeed strictly forbidden — it was easy to rectify an error made when punching up a program in reverse binary by carefully replacing the tiny rectangle in its hole in the card with the aid of some saliva and the back of the thumb-nail!

Yours sincerely,

Cecil Marks
Banstead, Surrey
2 March 1993

Dear Mr Enticknap,

I have just read with interest my first copy of *Resurrection* (issue 5). It was a thrill to see my name and address published in your bulletin.

I would like to add some points about the CP/M+ ProGroup, which stands for CP/M Plus Programmers Group. It publishes a 60-page monthly called *Journal of CP/M Plus Research.* Eight issues have been released so far.

Issue 0 tried to collect all the older texts I have about CP/M. Issue 1 was devoted to Universal Turing Machines, and included the original 1936 Turing paper.

Issue 2 contained the 'First Draft of a report on the EDVAC' by John von Neumann (1945), the classical IBM 360 text "Linkers and Loaders", and an article on programming language translation techniques. Issue 3 contains 15 miscellaneous short articles about every aspect of programming, and is the most popular so far (after issue 0).

Issue 4 contains the first comprehensive article about CP/M Plus, an article about a dozen small useful BASIC subroutines, an article on two areas of CP/M Plus called 'Page zero' and 'System Control Block' and one the best articles I have seen so far about RS-232C.

Issue 5 is devoted to a listing of the CCP (Console Command Processor) of CP/M Plus, with a very interesting article about a little known subject - computer failure and corrosion due to salt water. It is four pages long, and would maybe interest you, as you live on an island...

Issue 6 is devoted to a listing of the 111 volumes of the CP/M Software Library collected by the CP/M User Group (UK). More than 550 discs of 240Kb are available (over 120Mb) from its Diskcopying Service. Issue 7 is devoted to a little-known programming language called Mouse, detailing its implementation using Pascal.

As you can see, only the first three issues were concerned with the historical aspects of microcomputers. The Journal deals mainly with the software aspect of programming CP/M Plus, and the next issues will be more and more filled with assembly language listings.

I can provide copies of the journal for £5 sterling.

Yours sincerely,

Emmanuel Roche
Troyes, France
27 March 1993

Forthcoming events

20 May 1993 Seminar on Early NPL and English Electric Computers, from Turing to DEUCE

Free to members, but tickets must be obtained in advance from the Secretary. Starts at 11.00 am.

2 June 1993 In steam day

17 June 1993 Evening meeting

24 June 1993 Seminar on conservation and restoration of historic computers

Intended for other curators and Museum staff: price of admission around £40; members welcome.

23 September 1993 Evening meeting

28 October 1993 Evening meeting

18 November 1993 Society Open Day

In Steam Days start at 10 am and finish at 5 pm. Members are requested to let the secretary know before coming, particularly if bringing visitors. Contact him on 071-938 8196. All evening meetings take place in the Science Museum Lecture Theatre and start at 5.30 pm.

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