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.

The corporate members who are supporting the Society are ICL and Vaughan Systems.

Resurrection

The Bulletin of the Computer Conservation Society

ISSN 0958 - 7403

Number 20

Summer 1998

Contents

Editorial

Nicholas Enticknap

The major event since issue 19 of *Resurrection* has unquestionably been the golden jubilee of the first operational stored program computer, which was celebrated with great enthusiasm in Manchester in June. The major events included the "Launch Event" at the Bridgewater Hall on 17 June, the Golden Anniversary Conference which spanned the next two days, and the handover of the rebuilt SSEM to the Museum of Science and Industry on the anniversary itself, 21 June.

These events were attended by just about everybody connected with Manchester computing past and present. Sir Freddie Williams was represented by his widow, Lady Williams, who operated the switch that formally brought the replica SSEM to life, in front of an audience of 1800 people in the Bridgewater Hall. Professor Tom Kilburn is happily still with us — a mere stripling of 76 — and he gave both this audience and the one that attended the Golden Anniversary conference his reminiscences of the days leading up to 21 June 1948.

All of these events are reviewed in more detail in the following pages. We are also particularly pleased to publish an article which provides an original *Resurrection* contribution to the anniversary. It was written by Geoff Tootill, and describes the detective work put in by himself and Professor Kilburn in re-creating the world's first operational program in the exact form in which it ran on that epochal day 50 years ago. Geoff's article includes complete listings of both the earliest surviving form of this program (dating from a month after the first run) and of the original version as reconstructed.

Within 15 years computing had advanced so fast that it was possible for the world's leading computer manufacturer, IBM, to launch a complete range of compatible machines on 7 April 1964 — another key date in the history of computing. In this issue, Peter Titman recalls the thinking and decision-making processes that lay behind this "bet your company" decision.

By then it was clear that the major market for computers lay in data processing rather than computation. But there were other important application areas. Maurice Gribble completes the feature articles in this issue with a look at how Ferranti applied computer technology to industrial process control.

News Round-Up

The Annual General Meeting on 29 May approved a change to the constitution of the Society which provides formal recognition of our close links with the Museum of Science and Industry in Manchester.

- 101010101 -

Society Chairman Brian Oakley highlighted three achievements of the CCS year in his address to the AGM: the reconstruction of the Manchester University Small-Scale Experimental Machine; the progress made on the Bombe project; and the successful move of the Pegasus from the old canteen of the Science Museum to Blythe House.

- 101010101 -

The AGM re-elected all current officers and committee members of the Society.

- 101010101 -

The London Science Museum plans to feature the CCS Pegasus as the highlight of a new exhibition which is scheduled to open in the year 2000.

- 101010101 -

Nortel has made a generous sponsorship donation to the Bombe Project. A large cheque was handed over at a formal ceremony on 12 June.

- 101010101 -

The highlight of this event was the decryption of an original wartime German coded message, sent on 1 November 1944. The message was of course decoded at the time, but no record was kept of the Enigma settings used to do it. Tony Sale used the Bombe simulator to find these settings, and then configured an actual Enigma to decrypt the message. Then, to prove the concept, he successfully decoded a further message sent later the same day.

- 101010101 -

We are grateful to everybody who responded to our appeal in the last issue for volunteers to help with the Bombe Project. This has allowed the team to complete the database construction and to make very substantial progress on the CAD work.

- 101010101 -

Adrian Johnstone has reassumed the chairmanship of the DEC Working Party, and is focussing the team's efforts on a PDP-11/34 used at one time in the policing of the nuclear test ban treaty.

- 101010101 -

Three meetings were held in London during the spring. The first, on 12 March, saw Donald Davies and Roger Scantlebury presenting on "The Origins of Packet Switching and the Early Arpa network", and attracted an audience of 37.

- 101010101 -

This was followed by another seminar on a networking theme on 30 April, where Bob Cooper spoke on "The History of Academic Networking in the UK". He was followed by Peter Kirstein, who gave a talk on "The History of Arpanet and early Internet in the UK".

Finally, following the AGM Brian Shackel addressed members on "Human Factors — the Early Days 1954-70".

- 101010101 -

Details of the North West Group meetings programme for the autumn can be found under Forthcoming Events on page 44. Planning is also well advanced for three meetings in early 1999: on "The Leo Computer" on 26 January; on "Computers on Display" on 23 February; and on "The Transputer" on 23 March.

- 101010101 -

The Committee of the Society has formulated a policy statement concerning procedures for dealing with computers of historical interest that come to the Society's attention. This is published in full below.

- 101010101 -

CCS Collection Policy

1. The Society has no Collection of its own, and no premises in which to house one. There is no intention to change this.
2. When the Society hears of historic equipment which is becoming available for conservation, it will attempt to find a suitable home for it in one of the following major collections:
    - The Bletchley Park Museum Trust
    - The Science Museum, South Kensington
    - The Museum of Science and Industry, Manchester
3. The Society will also alert other collections to the availability of surplus equipment, where the major collections are unable to offer to house it, if it fits the appropriate area of interest. Members who know of such collections are asked to ensure that the Secretary is aware of their location and subject matter.

Computing's Golden Jubilee

Nicholas Enticknap

The computer is now more than 50 years old. The golden jubilee was marked by a series of celebrations in Manchester in the week leading up to the anniversary on 21 June, which fell on a Sunday.

From the Society's point of view, the star of the show was the replica Small-Scale Experimental Machine, or "Baby", created by Chris Burton and his team with 11,000 hours of voluntary effort. The machine was formally inaugurated by Lady Williams, widow of Sir Freddie Williams, during the launch ceremony on 17 June. Later that day Chris was presented with an honorary Master of Science degree in recognition of the achievement of himself and his team.

Achievement it certainly was. The designers of the original were too busy building it and getting it to work to have time to produce drawings or circuit diagrams (lack of documentation has been a problem since the very dawn of the computing age!).

Fortunately, Dai Edwards, who joined the computer team three months after the first program ran, did produce some detailed drawings while the SSEM was in the process of evolving into the first full-scale Manchester computer, the Mark I, and the notebook he used has survived. Chris Burton told the packed audience at the launch ceremony, "I mainly used that, and also Alec Robinson's photographs, which I scanned into a computer to work out the dimensions accurately".

Building the replica was the next problem, as the components used in it have been obsolete for many years. Despite that, Chris found that there are still dealers who maintain stocks of things such as thermionic valves and 1940s-vintage cathode ray tubes.

He added, "The pushbuttons used to key in a program were originally used on Spitfires. They cost 3/11 [just under 20p] in 1953, and I got them for £30 in 1995". The standard Post Office steel racks used to house these components presented more difficulty than anything: two of them eventually came "from the bottom of a Shropshire garden".

The result of this reconstruction process was highly praised by Tom Kilburn, who described the replica as "not only identical in appearance, but identical in every wire and every circuit. It would not be able to be differentiated from the Baby by any computer scientist". Kilburn's only reservation was that the replica was much cleaner than the original!

Professor Kilburn recalled the events of 21 June 1948 in two separate addresses, one during the launch ceremony and the other during the Golden Anniversary conference which followed on 18-19 June.

Most readers will know that the prime focus of the research conducted by Williams and Kilburn at Manchester University during the immediate postwar years was the development of the cathode ray tube as a memory device. By the autumn of 1947 they had got to the point where they could store 1024 bits for a period of several hours. "At this stage", said Kilburn, "we had demonstrated that letters typed in via a keyboard could be stored, but not digits arrived at as a result of computation. We needed a computer to do this."

There was, of course, no such thing available in the shops at the time, so the team had to build its own. That was the reason for building the SSEM.

The story of the SSEM has been told many times: as Kilburn said, "What new can be said about the Baby?". What is perhaps not so well known is the story of the program than ran on 21 June 1948 itself. "The problem was to find the highest factor of an integer. We selected this problem to program because it used all seven instructions, and also combinations of instructions. It was of no mathematical significance, but everybody understood it."

As with the machine itself, there is no contemporary extant documentation of the program. It "was actually written some time before, on the train going to Dewsbury", where Kilburn was living at the time. It was developed before the run on 21 June 1948, and developed further before Geoff Tootill wrote down the listing around a month later. Considerable detective work was needed to reconstruct the version actually used for the first run, and this is described in detail by Tootill elsewhere in this issue.

It was left to Gordon Bell to put the successful construction and operation of the SSEM into perspective. He pointed out that although the computer as understood today was conceived in 1946, "until you actually do it it doesn't exist, it's just another idea". Several teams of researchers were trying to translate the concept into reality: the Manchester team was the one that succeeded and so kicked the computer industry into life.

Bell also paid tribute to the CRT development work as a major necessary precursor to the computer. "It was really the Williams tube that got the machine going".

Why did Manchester get to the winning post first? Two principal themes emerged from talks given by various speakers. One of them was the enormous influence of the Telecommunications Research Establishment (TRE) at Malvern.

The CRT and subsequent SSEM projects had their genesis there, before Williams and Kilburn moved to Manchester. Kilburn revealed that "we had got to the stage of decoding 0 and 1 before we left Malvern, but we had to decide why it worked, and how they could be refined".

Alec Robinson paid tribute not only to "the backlog of ideas" which came from TRE, but also to the "TRE techniques" that were followed at Manchester. TRE influence was also valuable in insulating the design team from the problem of postwar shortages.

Robinson was one of the few Manchester people who had not himself come from TRE (he was previously with English Electric). Geoff Tootill did, and he recalled it as "the pre-eminent electronics R&D establishment during the war", adding that it was "a hothouse of talent". Alumni included, apart from those that went to Manchester, Maurice Wilkes (who went on to develop Edsac) and John Pinkerton (the chief designer of Leo).

The other major factor was the leadership of Freddie Williams, acknowledged by Manchester University Chancellor Lord Flowers in a tribute to his "unique inspiration, skill and enthusiasm" which "was so vital in the early days of computing". Geoff Tootill described him simply as "the boffin's boffin".

Alec Robinson, who joined the Manchester team in 1947, recalled that "The atmosphere was very pleasant, and very informal compared to English Electric. Professor Williams' way of working was to let people get on with it. He didn't leave you alone, though: he provided plenty of encouragement, and gave you a lot of ideas.

"There were very few people involved, so communications were easy. You were allowed to pursue your own ideas in a way that was very exciting."

Readers who would like to see what the very first operational stored program computer looked like can inspect Chris Burton's replica at the Manchester Museum of Science and Industry.

The Original Original Program

Geoff Tootill

There is no contemporary record of the epoch-making program that started computing as we know it today. This article documents an attempt to discover the precise form of that program. The attempt has almost certainly been successful, but there is still one remaining unanswered question.

Tom Kilburn and I have been trying to reconstruct the original version of the little Highest Factor program (or "Table of Instructions") Tom wrote in 1948, to test the Small-Scale Experimental Machine (SSEM) at Manchester University. This was the first program ever to run on a wholly electronic stored program computer.

Until 1996, all that was extant of this program was the "Amended Version" recorded in my lab notebook, which is dated 18 July 1948. By then I had improved the version that ran on 21 June 1948.

When I looked at this matter in 1975, at Simon Lavington's request, I remembered that to produce the version in my notebook, I had altered the original to prevent the starting value for the trial factor being overwritten. I had also determined that I could save one or more instructions to improve Tom's code.

In 1975, I couldn't remember how, and I wasn't able then to work out how to "disimprove" my version. To preserve the starting trial factor, and to avoid having to input both positive and negative versions of it, I would have added probably four instructions: however, I then cut out two of Tom's instructions, making the surviving documented version just the known nett two instructions longer than the original version.

With a fresh look at all the data, including statements made in the contemporary letter by Williams and Kilburn to *Nature*, Tom and I now think we've largely solved the problem.

We discussed it in detail on 22 March 1996. I then dug out a (rather poor) photocopy of my 1948 notebook, and rediscovered some notes we made when the SSEM first ran (see next page). I copied them, made some inferences from them, and sent a copy to Tom on 25 March.

Tom and I had further talks on 27 and 29 March, and agreed various points, which I incorporated in a revised document called "Further Analysis and Comments" (which follows).

SSEM Operating Notes

```
    c(?) 11011 10111 00111                       (Note 1)
2???(b) 00100 01000 11                           (Note 2)
22 ?(b) 00111 10111 01111 1-                     (Note 3)

20 Answer 11101000001                            (Note 4)
```

Answer <u>1046   +1   in</u> 1 minute

a = 4537 (= 13 x 349)   <u>Problem</u>

```
        a= 10011, 10110, 001      4537
line 19 c(a) 11100,01001,11011,11...
    20   b  00011,10110,00100      4536
    22 c(b) 00010,01001,11011,11...              (Note 5)

line 20 Answer −1 should be 348    101110101     (Note 6)
line 24 should be blank                          (Note 7)
CR no. should be 17                              (Note 8)
```

<u>Time 2 mins 25 secs</u>

Additional notes made in 1996: these Operating Notes are taken from a photocopy of the 29th page of my 1948 notebook, which was headed "Numbers used in Tom's Routine, 21/6/48". The first three lines are in my handwriting, and most of the rest in Tom's. "?" denotes illegible in the photocopy.

Note 1: c( ) means twos complement; this one is $-a = 3141$.

Note 2: This one is b1, of value $a - 1 = 3140$

Note 3: and this is $-b1$.

Note 4: This is 1047, ie the highest factor of 3141; I think we worked this out beforehand, so we would know if the answer was right.

Note 5: I infer we had to load b in 20, and $-b$ *in 22.*

Note 6: Tom bracketed this line with the next two and awarded them a tick to indicate that the machine had got the right answer.

Note 7: So line 24 held the remainder.

Note 8: So the "Stop" instruction was in line 17.

Tom, meanwhile, dictated his further thoughts onto a cassette tape on 29 March (this has now been transcribed under the name "The First

Program"). He sent copies of the tape to Chris Burton and myself, accompanied by a manuscript dated 29 March and starting "The most likely first program is...". The program Tom had worked out is exactly the same as the one I had worked out independently, which I sent to him on 31 March. The two documents crossed in the post.

What follows is my attempt at an exhaustive version of the analysis of the available data and synthesis of the original "Table of Instructions".

Further Analysis and Comments

The "Amended Version" dated 18 July 1948 in my notebook consists of:
Initialisation (lines 1-4)
Trial subtraction, Test sign of difference and Jump back (lines 6-8)
Form remainder *r n*, Test it and Stop if Zero (lines 9-13)
Form new divisor *b(n+1)*, Jump back, Load *a* (lines 14-19 and 5)
Fixed data (lines 20-24)
Variable data (lines 25-27)

I don't think we invented initialisation until a week or so after 21 June — the lavish idea of having instructions that would only be obeyed once seemed inefficient to me — but all the other elements must have been present in the Original Version.

The Operating Notes show that, in the Original Version, at the "Stop", line 20 contained one less than the "Answer", so, rather strangely, the "Stop" occurred after "Form new divisor". Therefore after the end of the "Trial subtractions" the sequence must have been: "Form remainder" (correction of the overshoot of the "Trial subtraction" cannot be postponed, because the value of bn needed would not still be available), "Form new divisor", "Test remainder", etc.

Furthermore, the first "Trial subtraction" must have preceded "Form new divisor" in the Table of Instructions, because it seems we worked out $b1 = a - 1$, and $-b1$ by hand and typed them in. If "Form new divisor" had preceded the first "Trial subtraction", we could just have put $b1 = a$, and not bothered to work out $-b1$. So the first lines of the main loop were "Load a", "Trial subtraction", "Test sign of difference", "Jump back".

From the letter to *Nature*, we know that the Table of Instructions

Amended Version of First Program

The 18 July 1948 version of the program is as follows (I wrote the explanatory comments in italics in 1998):

```
 1 -24,  C        load −b1
 2   c,  26       store −b1
 3 -26,  C        load +b1
 4   c,  27       store +b1

 5 -23,  C        load a

 6 Sub 27         Trial subtraction
 7 Test           is difference negative?
 8 Add 20,   Cl   still positive, Jump back two lines

 9 Sub 26         overshot, so add back bn
10   c, 25        store +r n
11 -25,  C        load −rn
12 Test           is remainder zero?

13 Stop           yes

14 -26,  C        no; load +bn
15 Sub 21         form b(n + 1) = bn − 1
16   c,  27       store b(n + 1)
17 -27,  C        load −b(n + 1)
18   c,  26       store it
19  22,  Cl       Jump to line 5
```

	init	current	final
20	−3		
21	1		
22	4		
23	−a		
24	b1		
25	-	r n	r N(=0)
26	-	−bn	−bN
27	-	bn	bN

was 17 in number, ignoring repetitions of the inner loop. From the Operating notes, we know that the "Stop" instruction was in line 17, and not immediately after the "Test" instruction as in the Amended Version. This was very likely intended to show with complete clarity that the original version could be part of a larger program, with further instructions following line 16 and replacing the "Stop".

Tom concurs in his audio tape that this is what he had in mind. Similarly, he agrees that the first line of the program cleared the accumulator simply to illustrate a point of sound practice, even though it was quite unnecessary in this case.

The reconstructed original version that follows respects all these constraints. It also uses the lines that follow the instructions consecutively (except for line 24) for the fixed and variable data indiscriminately, so I can believe that Tom started with consecutive lines but altered the sequence a bit during program development.

There may be other solutions to the problem of disimproving my Amended Version, but I cannot find one that meets all the conditions. For example, storing *$(r\ n -\ bn)$*, then forming $-b(n+1)$, then forming and testing $r\ n$, then forming $+b(n+1)$ could have 17 instructions, but leaves bN rather than *$b(N+\ 1)$* in line 20. I conclude that the result Tom and I arrived at is at the very least a close approximation to the Original Version.

Background Data

I do not have a copy of the Williams/Kilburn letter to *Nature*, but according to Simon Lavington's book it included the following passage:

"The highest proper factor of 2^{18} was found by trying in a single routine every integer from $(2^{18} - 1)$ downwards... Thus about 130,000 numbers were tested, involving some 3.5 million operations. The correct answer was obtained in a 52 minute run. The instruction table in the machine contained 17 entries."

Tom says that "operations" meant store accesses. Furthermore, he agrees that lines 17, 19, 20, 22 and 24 in the reconstructed version must be correct, to conform to the Operating Notes.

Until recently I thought that on 21 June 1948 the digit period of the machine was 8.5 microseconds, and the "bar" (ie the instruction time) was 1.2 milliseconds. These two figures are consistent with each other,

The Original Version

Here is the original "Table of Instructions" as synthesised from this analysis on 31 March 1996 (I wrote the italicised comments in 1998):

1	-18, C	*clear accumulator*
2	-19, C	*load +a*
3	Sub 20	*Trial subtraction*
4	Test	*is difference negative?*
5	Add 21, *Cl*	*still positive, Jump back two lines*
6	Sub 22	*overshot, so add back bn*
7	c, 24	*store +r n*
8	-22, C	*load bn*
9	Sub 23	*form* $b(n+1) = bn - 1$
10	c, 20	*store* $b(n+1)$
11	-20, C	*load* $-b(n+1)$
12	c, 22	*store it*
13	-24, C	*load* $-r\ n$
14	Test	*is remainder zero?*
15	25, *Cl*	*yes, Jump to line 17*
16	23, *Cl*	*no, Jump to line 2 (but see text)*
17	Stop	

	init	current	final
18	0		
19	$-$a		$-$a
20	b1	bn or $b(n+1)$	$b(N+1)$
21	-3		
22	$-b1$	$-bn$ or $-b(n+1)$	$-b(N+1)$
23	1		
24	-	r n	r N
25	16		

given that the bar was four "beats", with each beat being 32 digit periods plus 4 digit periods for the timebase flyback.

However, on the 23rd page of my notebook (which was probably dated 9 June 1948), I find the following: "13.4 [million] operations, approx. Each takes 1.35 mS." On 13 June a similar estimate of time also uses a figure of 1.35 milliseconds. These cannot be measured times for the calculation, because the date is before the computer started to work: the presumption is that they were measured bar times. This implies a digit period of 9.4 microseconds.

Between 10 and 13 July 1948, according to my notebook, I tested the version of Turing's long division program that I had corrected. I noted that 333 divided by 3 took two seconds, and that 1,073,741,823 divided by 34,636,833 took "about $1\frac{1}{2}$ seconds". I can't remember how I measured these times, but I doubt if they would give us an estimate of the digit period at that date with useful accuracy, and I certainly cannot assume that it hadn't been changed since 21 June. We probably changed it as often as we changed our socks.

But on 3 August 1948 there is "...digit place 39..." and "Total time [1,103,872] bars = 27.5 mins ±5%", ie 1.49 milliseconds per bar, and now with 45 digit periods per bar, the implied digit period is still 8.5 microseconds. Was this just another paper estimate, the 5% tolerance arising from the probable error in the assumed digit period, or was it a measured time? I don't know.

Simon Lavington states that by April 1949 the word length was 40 bits and the bar 1.8 milliseconds. So, with a 5-digit period flyback, the digit period was 10 microseconds. I do remember that we altered the digit period to 10 microseconds, but I can't remember whether we noticed what it really was beforehand.

On various dates in 1948 and 1949 we experimented with different ways of digging and filling wells on the face of the main storage tube. It is quite possible that we altered the digit period for particular tests, and did not put it back to its previous value afterwards.

Inferences from the Background Data

The "Further Analysis and Comments" section above seems to provide valid reasoning for the reconstructed version of the original program shown, with one proviso: there is no evidence one way or the other for instruction 16. I said in my letter to Tom of 31 March that one can choose either 15 or 16 instructions in the main loop, by jumping back to line 2 or line 1. What I meant by this was that the main loop written code could have consisted of either 15 or 16 instructions.

This is misleading in the context, because what we are really interested in is the number of instructions obeyed for one iteration of the main loop — that is, in testing one possible factor. For the values of *a* and *b1* in the letter to *Nature*, lines 3 and 4 are executed twice and line 15 not at all. So in the reconstructed program as shown above, 16 instructions are executed per number tested.

If, alternatively, line 16 reads "18, *Cl*", 17 instructions are executed. As it happens, both the constants (0 and 1) needed to jump to either line 1 or line 2 are already available, because they're both used for other purposes as well. So we cannot deduce from that whether it was 16 or 17 bars per number tested, by including or not including the superfluous "Clear accumulator" instruction in each iteration. To choose between these two alternatives we need further data.

From $(2^{18} - 1)$ to 2^{17} is certainly 131,072 numbers, so in a 52 minute run a digit period of 8.5 microseconds implies 19.45 instructions. This is implausible in view of the evidence for either 16 or 17 instructions.

However, we have further timing information on the same program from the Operating Notes, with data that results in a different ratio of outer to inner loop executions. It is not too laborious to work out the number of bars needed for values of *a* for which we know the measured times, and indeed I have verified the figures I quote in Table 1 below by emulation.

There is good agreement between the digit periods calculated for the runs for 4537 and 3141, which took place on 21 June 1948, and the discrepancy between these and those for the 52 minute run quoted in the letter to *Nature* is at least 16%. It therefore seems unlikely that the digit period was in fact the same on these two occasions.

I conjecture that, having seen the logic design of the hardware and software vindicated on 21 June, we did a little preventative maintenance before embarking on a more exhaustive test of the robustness of the hardware, probably a day or more later. During this maintenance, we quite probably altered the digit period.

If indeeed there was some justification for operating at the figure of 9.4 microseconds noted on 7 and 13 June 1948, instead of 8.5 microseconds, then conceivably during the maintenance period we set to that figure. It could be, in fact, that we just reset the three- or four-gang .0005 mfd variable condenser that determined the basic oscillator frequency to a particular pencil mark on the front panel. This hypothesis implies that an accurate observed value of 9.7 microseconds, being nearer to an approximate 9.4 microseconds, is more plausible than an accurate one of 10.3 microseconds. So this point argues for a jump back to line 1 rather than to line 2.

I explain the reversion on 3 August 1948 to the figure of 8.5 microseconds by assuming either that we did in fact correct it, by reviewing the pencil marks on the front panel, or that we knew the machine would work with the shorter digit period, and I therefore used this value for my estimate.

On the other hand, with line 1 in the loop we have 31 main store accesses per iteration, and a total of 4.06 million during the 52 minutes. Without line 1, we have 29 accesses per iteration and 3.80 million in total. Neither figure agrees with the letter to *Nature*, so it seems we should accept Tom's suggestion (on his tape of 29 March) that the figure of 3.5 million was wrong.

Tom points out that we had been struggling to make the machine work for some days before 21 June, which was a Monday, after we had had the weekend off. During these efforts we would certainly not have been prepared to wait as long as say 60 seconds for each trial, and we would therefore have tried a smaller number. We agree on this point.

In fact, we probably varied the smaller number a bit. Tom suggests we used 19 or 31 for the first successful test, since he thinks we got a zero in line 20. That is, we tested a prime number. I can't recall any more detail myself, I'm afraid, but I have included these two numbers in Table 1, to show they are indeed quite plausible.

Conclusion

In the Original Program, the jump back to line 1 gives a digit period a bit more plausible that a jump back to line 2. That is, line 16 should read 18, *Cl* and not 23, *Cl* as I have it. But this is hair splitting, and Tom later concluded that line 2 as shown is correct. It seems unlikely that further palaeographical work would shed further light on this point.

a	2^{18}	4537	3141	31	19
b1	$2^{18}-1$	4536	3140	30	18
bN	2^{17}	349	1047	1	1
Measured run time (minutes/seconds)	52m	2m 25s	1m		
Estimated tolerance on measure	±0.3%	±2%	±5%		
Number of trial subtractions	262,145	14,088	4713	142	77
Number of inner loop instructions	786,435	42,264	14,139	426	231
Jump back to line 1					
Total number of instructions executed	2,228,228	88,333	37,174	757	430
Ratio inner loop/ other instructions	0.5455	0.9174	0.6138	1.2870	1.1608
Therefore bar time (milliseconds)	1.400	1.64	1.61		
and digit period (microseconds)	9.72	11.4	11.2		
Jump back to line 2					
Total number of instructions executed	2,097,147	84,146	35,081	728	413
Ratio inner loop/ other instructions	0.6000	1.0091	0.6752	1.4106	1.2692
Therefore bar time (milliseconds)	1.487	1.72	1.71		
and digit period (microseconds)	10.33	12.0	11.9		

Table 1: Bar Times and Digit Periods

The Thinking Behind the IBM 360

Peter Titman

Why did IBM embark on the risky venture of producing the System 360 series when it was already dominant in the market? The author describes the decision-making process that led to the development of the world's first range of compatible computers.

I joined IBM United Kingdom as what was called an applied science representative, which meant selling scientific machines to universities and engineering installations. That job disappeared when it was decided that we did not need separate marketing organisations for the scientific market and the commercial market.

But when my applied science job ended the 360 wasn't going to be announced for two years, and from a salesman's point of view, something that is not going to be announced for two years doesn't exist. I was spare, so I got assigned to worrying about what was going to happen in two years time.

So my view is quite different from the people who were involved in delivering the 360. I was concerned with questions like: what have they got us in to? And what were we going to do about it?

It is hard for those of us who were with IBM then not to appear boastful and stand up and say, "Gosh, weren't we marvellous". The fact is that we had an enormously powerful marketing position at this time, which was the end of 1961, beginning of 1962. We were selling the 1401 in incredible numbers. You could go to a company that had a punched card installation, say two accounting machines and a calculator, and sell them a 1401 for the same price. They could do very much more with it, and they could do it simply.

Most computer users in those days were doing really simple things — just producing the invoices and getting the money back was a chore they were delighted to give to a machine. There were some people doing more complicated computing, but not in large numbers, so they weren't the main source of revenue. Revenue was coming from really basic things.

We did have incredible control over the computer market. That is in worldwide terms: in the UK, we weren't the largest company, but in almost every other country we were.

By the end of 1964, we had sold 13,000 1401s — orders of magnitude more than the ICT 1301 and the RCA 301 (which ICT also sold). This made a big difference when it came to how much program support you could provide, and in many other ways.

I came from Leo Computers where we had to beg for every transistor. At one time I designed memories: when I had finished that I was at once put to work on the tape channels. We were that short of people. IBM had so much more resource.

So the first message is that we were really strong. In some market segments such as large machines, we were dominant even in the UK. There was a good political reason for our dominance here — the people who bought large machines in those days were from organisations such as the Atomic Energy Authority and the CEGB. These companies were at that time very closely tied to the US nuclear industry, so had a strong incentive to buy the same machines that were being used in the US. That was one reason why we had 7090s in Aldermaston and in CEGB and other 704s dotted around in Risely and similar sites.

So we were very strong in that sector. The place where we were weakest was the scientific area, and particularly in the UK the small scientific area.

I sold 1620s when I first joined IBM. The 1620 was a lovely little desktop machine with a memory of 20,000 decimal digits and a typewriter, and you sat down and used it like a PC. But it cost more than the Elliott 803 which had 8000 words of memory and ran much faster. The 1620 didn't even have an adder: it did arithmetic by doing table lookup, and it did it very slowly. A multiplication took 200 milliseconds or so.

Compared to the 803, the 1620 was an absolute dog. It had the most unbelievable paper tape reader. It was designed by someone who designed magnetic tape readers. Everybody knew that the way to sense timing on the paper tape reader was to sense the sprocket hole. But whoever designed this device knew that the way you drove paper tape was by a cog wheel through the sprocket hole to which you attached the spindle to which you attached a disc with holes in from which you then sensed the timing. This had the enormous advantage that you got the timing out of place and also, for a bonus, it screwed up the sprocket holes as well.

So we weren't well placed in that sector of the market. But in the US we did sell the 1620. We sold it to universities because we were IBM and universities bought IBM machines. The 803 was much, much better and sold, almost uniquely among UK machines, in reasonable quantities, abroad as well as at home.

The success of the 803 was one reason why John Fairclough got the job of developing a small scientific machine at Hursley. That raised a question: companies like Esso bought machines in any part of the world that they chose, but IBM maintained systems only in the countries where we sold them. Esso was very unhappy that we didn't always immediately announce things in the UK that we had announced in the States because the parent company wanted to have the same machines in all subsidiaries. The thought of someone in the UK announcing a better machine and the parent company not being able to have it was absolutely unthinkable.

So John's chances — though he didn't know it at the time, and it is only in hindsight that I recognise this — were not that good because it raised the question of how, if we had something brilliant announced in World Trade, we were not going to announce it immediately in the States. (World Trade was the part of IBM that covered the whole of the rest of the world outside North America. World Trade then had about a third of the sales volume of the US.)

That was one of the issues: should we have a separate UK product line, or at least a World Trade product line? Don't forget that in those days, the disparity in wealth between the States and Europe was much greater. An American cost about twice as much as an Englishman and as three times as much as an Italian. So when you came to do sums about displacing people and so on, you could bear much higher costs in the States. So there was a reason for thinking that the World Trade market was different. But the big decision was that it wasn't different.

Anybody in our strong position would normally continue operating along the same lines. What was remarkable was that IBM decided to do something different. If we were doing so well, why change?

Apart from market dominance, our other enormous strength was technology. People in the UK didn't understand the strength or depth of IBM technology.

As I said earlier, when I finished doing memories at Leo I was put onto looking at tapes. The Decca tape had not been a great success on Leo II: EMI was trying to enter the tape drive market, but the main companies we investigated were Potter and Ampex. Leo III eventually used Ampex drives. But there was no possibility of making our own tape drive.

IBM tape drives, I thought at the time, were terrible, because they used to do dreadful things to the tape. They had pinch roll capstans that used to grind it, the tape ran in contact with the head, and the drives needed quite a lot of maintenance. What I didn't realise, though it seems obvious in hindsight, was that if you wanted to get faster tape, as everybody did, you could either make the tape move faster or pack data more densely. To pack data more densely it was a huge advantage to have your head in contact.

Only IBM did this at the time. Everybody else avoided contact because they felt they couldn't manage the wear on the tape. But IBM had the resources to apply to this problem and solve it. The solution proved to be the development of proprietary heavy duty tape. IBM had the technology right through, covering the materials technology, the tape technology, the drive technology, the controllers, everything. No other company could put that lot together to make a new solution.

In printers, IBM also innovated with the 1401. Everybody else used Analex drum printers. On these, a big drum went round and round and you had to strike it at exactly the right time to produce good quality printing — if your timing was out, the letters went up and down a bit.

The IBM answer was to have a chain that went sideways instead of a drum. You still had to bang it at the right time, but if the alignment was slightly out, the letters were displaced slightly left or right rather than up or down. That didn't look nearly so bad. Psychologically it looks quite different to have a letter slightly to the left rather than up or down.

Furthermore, the mechanism was cheaper. You just had this one chain and that itself was replaceable, so you could have different fonts, and you could have upper and lower case and other capabilities which were impossible on the old Analex.

I can remember the Powers Samas printer on the Leo II. Powers Samas really didn't believe in electricity. If it could run with a belt or a cable, it was done with a belt or a cable. Electricity was dangerously new fangled. The IBM 1403 printer was a totally different machine in terms of its simplicity, reliability and all the other factors.

IBM's disc technology was another plus. The first disc paper I can trace dates from about 1953 and was a theoretical study of floating a head on a disc — the air bearing disc. The first disc subsystem itself, 305 Ramac, was not a success. It had 5Mb capacity accessed by a single arm, which provided access times in the order of seconds. But it gave us valuable experience.

By the time we got to the 360, we had not only realised that discs were important — far better than tape as a subsidiary storage medium — but also what an enormous advantage it was to have your operating system on a disc. Operating systems on tape were continuously chuntering up and down. If you'd performed a compile, and then wanted to link something, you had to access another part of the tape or use another tape drive. There was lot of movement trying to get the flow of jobs through a tape machine. Discs obviated this problem.

We had virtually a monopoly on disc technology. I can remember John Pinkerton of Leo Computers coming to Wigmore Street to explore buying discs from us because there was no alternative source for him. So Leo Computers would have had to resell our discs onto their customers, a tremendous disadvantage. A lot of our other competitors just didn't go into discs at all. But we had a disc technology that was good and reliable, and we understood disc operating systems.

Another technological strength was memory. We had invested a lot of money: the reason we were ahead was not because we were big but because we had invested a lot. For example one major investment was in a plant which automated the process of making a memory stack.

But having made the investment, we could sell memory for about five times its manufacturing cost, or more than that even. The manufacturing cost of memory was nothing. So we made enormous profits on memory.

So why, with these immense strengths, did we embark on this very risky System 360 venture? Why abandon our existing products and produce a new range of machines that was totally incompatible with them?

I was conscious at the time that customers such as ICI were keen to have the same machines in all their offices and in all their computing centres. They were looking for standards. This was an issue for IBM, too: we were having to train customer engineers and support engineers in all sorts of different machines. When I joined I had go through programming courses on the 650, the 7070, the 1620 and the 709. It was all rather expensive.

To compound this problem, there was a huge explosion of programming languages and operating systems. For the 7090 alone we were on to Ibsys version 13 by the time we were talking about the 360. Then there was a Fortran monitor system, and another system called the Share Operating System (SOS), designed in conjunction with our customers, which was a total disaster. Essentially the customer group (called Share) did the design, and we funded it, supported it, and made it happen.

SOS was an example of what you get if you listen to your customers. You have to understand what your customers want — that's important. But your customers are not going to tell you what to build, because they don't know. It is important to listen to your customers but you have to understand what they really need. That was the key thing.

The initial Fortran group was comprised entirely of IBM people, seven of them. Share played a part in its subsequent development, but the initial design — and Fortran I was a most remarkable system because its optimisation was better than anything that came after it — was really good.

In the sixties everybody thought that if you had the right program language, you could do anything. All that you had to do was to find it. We had Fortran, Comtran (the commercial translator for the 7090), Cobol and Algol.

People were starting to ask some quite basic questions about programming, such as "What's it for?" and "Why are we — IBM — writing programs for free?" All the customers did when you gave it to them was moan. When we announced Ibsys, people said "Gosh it's complicated isn't it? Doesn't it take a lot of memory? Do we need it?". Of course they needed it. But they weren't grateful for it in the least, so it was a real concern.

Then there was the question of what we should spend our development money on. A lot of it was going on the 7090 because all the interesting activity was on the big scientific machines. Where the revenue was coming from was from the 1401, but we weren't spending nearly so much on it. No one knew why.

So questions like these were being bounced around group staff in the corporation. As I recollect it, group staff, and a chap called Don Spaulding in particular, had decided that we had to do something about the proliferation of software. Therefore they set up a committee.

But like all the best committees the people who set it up had a very good idea of what they wanted the committee to say beforehand. The terms of reference were something like this: "to produce a complete plan for the total product range, suitable for all countries and for both scientific and commercial computing". There was strong pressure to make a compatible family of machines.

No one knew whether it could really be done, but Bob Evans in particular had strong feelings about compatibility and felt that it was possible.

He was the president of DSD — the lab systems development division — and had been involved in the 7070 which was a mixed commercial-scientific machine.

The committee came back with the required conclusion. They presented us with a report recommending what was known as the New Line Plan.

This plan caused me personally a lot of worry for a long time. It envisaged a staggered announcement of machines, the first of which was to be a small binary version of the model 40 — the machine being developed in Hursley. Immediately there was a problem. What we were aiming at in the long term was a grand strategy. What we were allowing to poke out was one small binary machine which, though an interesting machine, was not enough on its own to get anybody excited.

The other problem was compatibility with previous systems. The report acknowledged the problem, but expected IBM to be selling new applications that depended on users migrating from tape to disc, and further ahead some exciting teleprocessing applications. Accordingly it decided to ignore the compatibility issue. The difficulty was that even people doing new exciting things tend to have done some boring things first which they would like to continue doing on the same machine. So compatibility was a big worry.

We discussed the plan for about 18 months, exploring how we were going to achieve the ambitious targets laid down. But IBM does change it plans, though not easily. A lot of pressure has to build up to effect such a change. The catalyst in this case was the Honeywell 200 with its cheekily named Liberator program, which would take a 1401 program and run it faster and more cheaply — a beautiful thing to sell. Naturally it sold in large numbers, very quickly. This was around about the end of 1963.

Then a very typical IBM thing happened. Tom Watson became involved and ordered a solution to the Honeywell 200 to be found in 90 days. Suddenly we discovered something we knew already, that the way to solve the migration problem was through emulation. We had microprogramming, we had the technology to do it, we just had never somehow reached the conclusion that emulation was an important compatibility technique. But by this stage it was something marketing people knew they had to have.

The other conclusion we reached was not to mess around announcing tiny bits at a time, but to launch the whole lot at once. That allowed us to emphasise the strategy rather than individual machines. Quite suddenly we had the big bang. It was the end of 1963 when the decision was taken that a complete range of machines would be announced on 7 April 1964.

The range comprised six models of CPU and a complete set of new I/O devices. Remarkably, you could attach any I/O device to any CPU: they weren't handcrafted the way they were before. We developed emulators for every existing computer that was important. So suddenly we had a completely new exciting product line to sell.

We sold it in the UK with an enormous full page text advert: we had a full newspaper page of detailed text about the 360 series on the day it was announced. Of course we took our customers to Hursley to show them round as well, but it was the media promotion that really made for an exciting launch. This was not at all the way the committee had planned it in the first place.

Editor's note: this is an edited version of the talk given by the author to the Society at the Science Museum as part of the IBM System 360 seminar on 21 November 1995.

Editorial fax number

Readers wishing to contact the Editor may do so by fax, on 0181-715 0484.

The Argus Computer and Process Control

Maurice Gribble

The first Argus grew out of a small computer that was built to study digital control systems. The circuits of that machine, which were designed about 1954-55, used Mullard OC71 low frequency germanium pnp transistors. They were originally developed for producing shift register sequences, or chain codes, for use in a rotating beacon missile guidance system.

The circuits comprised NOR-gates and flip-flops which were clocked by narrow shift pulses to produce shift registers and counters. Positive logic was assumed, that is, the transistors were turned on — collector at earth potential — for "1" and off — collector negative — for "0". Diode OR-gates were followed by a transistor via a coupling network to produce a NOR-gate. The flipflops consisted of two transistors cross-coupled via RC networks with a pulse steering circuit for shift pulses, which enabled shift registers and synchronous counters to be made. The shift pulses were produced by a crystal controlled blocking oscillator, the clock frequency being 25KHz. The power supply provided +6 and −6 volts.

Experimental Computer

In 1956 a small computer was proposed for studying sampled data systems and the feasibility of digital control of weapon systems. Experimental transistor computers were being built at Manchester University and at AWE Harwell. These machines used point-contact transistors, which at that time had the advantage of speed and the fact that only one was needed to make a latching circuit. However, they were difficult to manufacture and not very reliable, so it seemed sensible to design circuits around junction transistors, in the hope that faster devices would eventually become available. Due to the rapid progress in the semiconductor industry, this happened sooner than expected.

There was a need for additional circuits, such as a 3-entry flipflop, so that registers could be loaded in parallel and shifted in both directions. This was achieved by replicating the pulse steering circuit and ORing them together. There was also need for a shift gate, so that various shift pulse sequences could be produced. This also used a pulse steering circuit to gate shift pulses with the logic levels.

The circuits were built into plug-in units, which were hand wired. A printed circuit version, which was fairly novel in those days, was also produced, but these were not used in this experimental machine, which used serial arithmetic with a 10-bit word length and represented negative numbers in twos complement form. It incorporated a Booth short-cut multiplier, which could handle positive and negative numbers with equal facility. The Newton-Raphson method was used for division and calculating square roots, while Chebyshev polynomials were used for trigonometric functions.

Only a small amount of data storage was supplied in the form of flip-flops, since data was read in from transducers as required, and was output as soon as the calculations were complete. Other inputs were a random number generator and a timer. Output was analogue, either as voltages or in the form of width-modulated current pulses, for operating servos.

The single address order code contained only 19 instructions and the program was stored on a pluggable diode matrix. There were two program counters: one gave 64 program steps for the main program, the other 25 steps for subroutines. Both were ring counters to simplify matrix decoding, and the final OR-diodes were glued into the plugs. This was a source of trouble, as there was differential expansion between the plastic plug and the glass diode, which shattered the glass. The solution was to use a silicone rubber cement instead of Araldite.

A Gray or cyclic progressive (CP) code was used for measuring angular position. This has the merit that, as the shaft with the code disc is turned, only one digit changes at a time, whereas in binary code, several digits often change in going from one number to the next. These changes do not all occur at exactly the same time and can result in false readings.

Conversion from CP code to binary was easily implemented. The data from a disc was selected by suitable logic and transferred in parallel into a register where it was shifted left cyclically via an Exclusive-OR circuit. This was carried out by separate hardware in the computer in order to speed up the process and avoid adding to the complexity of the arithmetic unit.

The analogue voltage output converted the least significant five bits of the number by summing currents into an operational amplifier. The logic took account of the sign of the number and arranged that the amplifier saturated for numbers of larger magnitude.

The width-modulated current pulses for driving hydraulic valves were produced as follows: the servo error was stored in a 5-bit accumulator and, according to the sign of the error, unity was either added to or subtracted from it every word time until it was zero. Until zero was reached, a large current of the appropriate sign was switched into the valve, resulting in a pulse of current whose length, and thus average value, was directly proportional to the error. By making the maximum current twice that needed to produce maximum output from the valve, two such valves could be operated on a time-shared basis.

Royal Demonstration

The computer was completed in October 1957, in time for HRH Prince Philip's visit to the Wythenshawe Labs in November, and used in a demonstration of parallax error correction and servo control.

Two turntables, separated by a metre or so and carrying digitising discs, could be rotated rapidly through about 90 degrees by hydraulic jacks. Each turntable had a optical projector mounted on it, which projected a spot of light onto a screen, one being coloured red and the other green. Care was taken to ensure that the port side was red and the starboard side green! A third digitiser was fitted with a knob and, when it was turned, the angle measured by this digitiser was compared with that of the left-hand table and the error operated a digital servo so as to minimise it. To correct the parallax error, the right-hand table was, at the same time, turned by a similar servo to another angle, given by the trigonometric relationship below, so that the green spot of light fell on top of the red one.

The formula used was:

$$\tan R = \tan L \ / \ [s/d \ (\tan L) - 1]$$

where s is the separation of the turntables, d is the distance of the screen, and L and R are the left and right digitiser angles respectively.

Three term Chebyshev polynomials calculated tan and arctan to sufficient accuracy, but, due to the computation time, there was a noticeable lag in the position of the green spot if the knob was turned too quickly.

The demonstration behaved well when Prince Philip turned the knob. He then looked at a printed circuit version of a flipflop; the name obviously amused him, for as he went out of the door he could be heard saying, "Flipflop, flipflop ...".

Faster Transistors

High frequency transistors, developed for radio, soon became available. They were alloy junction transistors with thinner base regions, and it was necessary to design new circuits to get the fastest operation with them. Alloy transistors have a relatively large base region that can store charge carriers — holes in pnp transistors. In switching circuits, if excess base current is used, in order to turn the transistor on quickly, there is an excess of holes in the base region that has to be removed before the transistor ceases to conduct. Faster turn-off times can be achieved if this saturation is avoided, so the Argus circuits used a diode between the collector and the base coupling network in a negative feedback anti-saturation circuit. An extra power supply of −24 volts was also needed.

There was also a requirement for a power NOR-gate that was capable of a much larger fan-out. It used an npn transistor to speed the removal of charge from the base of the output transistor, whose collector was returned to the 24 volt supply. This pull-down resistor was removed and replaced with a line-terminating resistor when outputs, which were inverse signals, were common-collector ORed onto a bus.

The flipflop circuit was, in effect, two NOR-gates connected back to back plus a trigger circuit. Diode inputs were used everywhere to standardise the circuit loading and, in the case of the trigger circuit, to improve the flipflop recovery time.

The shift gate for producing sequences of shift pulses used a pulse gating arrangement similar to that used in the flipflop trigger circuit. There were two designs of shift gate — one avoided a heavy standing current by using an npn transistor that was transformer coupled to the output transistor.

These circuits were engineered in printed circuit form and used in large numbers in several Ferranti computers at Wythenshawe and Bracknell, and also in Canada for the Ferranti-Packard 6000, which became the basis of the ICL 1900 series.

The improved version of the Bloodhound missile required a complex sequence of pre-launch operations involving calculations and decisions. It was realised that these were more suited to a digital rather than an analogue computer, if one could be built that that was sufficiently fast and reliable. A high degree of accuracy was not required for most of the calculations, so a short word length would suffice. Furthermore, the program was unlikely to be changed when the equipment was in service, so, for the sake of reliability, a fixed store was preferable.

The Argus Computer

A design was undertaken based on these requirements. More storage was needed for both program and data. A core store was used for the latter, while the plug board used for the program was replaced by a patent magnetic store using ferrite rods, which will be described later. This also held constants, used in the calculations.

The word length could be either 12 or 24 bits and the clock frequency was 500KHz. The machine operated in a serial/parallel mode — two bits in parallel. For a 12-bit word, this was a reasonable compromise between speed and cost when store access times were taken into account, and resulted in a 20 microsecond addition time.

The order code was based on that of Pegasus, because there was already a body of experience with that machine, and because the Ferranti London Computer Centre was concerned with promoting the machine for process control. For example, Babcock and Wilcox, a prospective Argus customer, was a keen Pegasus user.

There were eight accumulators, one of which was zero. The instruction was 24 bits long and the instruction format was similar to that of Pegasus. The first 12 bits (the N address), addressed the main store, the next three (the X address), an accumulator, the next six specified a function, the next three (the M address), a modifier, and the last bit (the C bit) indicated whether the number was an instruction or one of two 12-bit constants, selected by the least significant address bit. So only the first 2048 locations in the program store were available for constants.

The main differences between the Pegasus and the Argus instruction set is that the latter had special "30" type orders for input and output and the "40" type orders involved literals, not counters. Also Argus had special jumps for interrupts — "70" type orders.

As in Pegasus, the accumulators could be addressed either as registers, using the N address, or as accumulators, using the X address; there was, however, a difference. Registers 0 to 5 corresponded to accumulators with the same addresses, but accumulator 6 was 24 bits long and consisted of registers 6 and 7. Likewise, accumulator 7 was a double length one, consisting of accumulators 4 and 5. Double length arithmetic was automatically used when either of these long accumulators was addressed. Accumulator 6 was also called P, with accumulators 4 and 5 being called P1 and P2 , respectively; while accumulator 7 was also called Q. Accumulators 0 to 3 could also be used as address modifiers.

The serial/parallel arithmetic used two adder/subtractors with ripple-through carry, so the delay in the carry path was kept as short as possible. The flip-flops were triple-entry types with three different shift pulse inputs to allow left shift, right shift and double right shift for addition and subtraction.

The multiplier was a modification of the Booth multiplier used in the experimental machine. The Booth method examines the least significant bit p_n of the multiplier together with an extra bit p_{n+1}, initially set to 0, and, according to the values of these two bits, performs the operations shown in Table 1.

$\mathbf{p}_n$	$\mathbf{p}_{n+1}$	**Operation**
0	0	Shift R
0	1	Add multiplicand, shift R
1	0	Subtract .. ..
1	1	Shift R

Table 1: Booth short-cut multiplier

In the Argus method, the bits p_{n-1}, p_n and p_{n+1} are examined with results given in Table 2. Use is made of both single and double right shift.

$\mathbf{p}_{n-1}$	$\mathbf{p}_n$	$\mathbf{p}_{n+1}$	**Operation**
0	0	0	Double shift
0	0	1	Add, Double Shift
0	1	0	Add, ...
0	1	1	Shift, Add, Shift
1	0	0	Shift, Subtract, Shift
1	0	1	Subtract, Double Shift
1	1	0	Subtract, ...
1	1	1	Double Shift

Table 2: Argus Multiplier (Modified Short-cut)

Like the Booth method, the Argus method worked with both positive and negative numbers and gave a useful increase in speed as well as a lower spread in multiplication time, which was important in control applications.

Method	**Number of Add & Subtract Operations**

	Average	Maximum
Booth	n/2	n
Argus	n/3	n/2

Table 3: Relative Speed

Argus used a non-restoring divider, which would operate with both positive and negative numbers. The dividend was held in the accumulator P, the divisor in register D and the quotient in register QR.

The bits of divisor and dividend were compared, starting at the MS end. If they were the same, D was subtracted from P and a least significant 1 added to QR. If they were different, D was added to P and a least significant 1 subtracted from QR. After either operation, P and D were shifted one place left. Finally, if the signs of the divisor and dividend were different, 1 was added to the quotient in QR.

Division could be rounded or unrounded and, as in Pegasus, the dividend could be either a double or a single length number. This held when either 12-bit or 24-bit arithmetic was used.

Square root was programmed using either Newton-Raphson or the digit by digit method, while Chebyshev polynomials were used for trigonometric and other transcendental functions.

As previously mentioned, data was held in a core store. This was in modules of 1024 12-bit, plus parity, words and could be increased up to a maximum of 3072 words. It was bit-organised, that is, it used 3-D selection — X, Y and Inhibit. The cycle time, using state-of-the-art cores and transistors, was six microseconds. It was one of the earliest transistor-driven core stores and many problems had to be overcome. The transistors had to be fast enough, they had to handle large currents and they had to stand the relatively large voltage produced by the back emf of the magnetic cores, which were conflicting requirements.

Since there was some doubt about the long-term reliability of the core store under adverse conditions and the program was seldom altered in control applications, program and constants were held in a fixed store consisting of ferrite rods the size of pencil leads. These provided magnetic coupling between a set of drive tracks and an orthogonal set of read tracks, one for each bit in the word, etched on a double-sided printed circuit. Current pulses, selected by a diode matrix, were sent through the drive tracks in turn; and the read tracks only picked up a signal when a ferrite peg was placed in a hole at the intersection and disturbed the symmetry of the magnetic field.

The store was engineered as a set of trays, each holding 64 25-bit words, including parity, together with the decoding diode matrix and transformers. The ferrite pegs had little rubber caps to make them easier to handle and were held in position by a perspex cover — another source of trouble, for it used to get electrified and pull the little pegs out when it was removed. The store consisted of a box of 16 such trays, together with their associated drive and read circuits, making 1024 words total. The 12-bit address could accommodate four such stores, if required.

This store was rather bulky, so another magnetic fixed store was developed for a military application where size was important. It was not used in Argus, as its development came too late, but it is included here for historical interest.

In this store the ferrite pegs were replaced by square coupling loops, which could be cut to prevent induced eddy-currents coupling to the read tracks. The loops were arranged on a thin fibre-glass printed circuit, which was identical in size and shape to a Hollerith punched card, so that the loops could be cut with a modified card punch. Each card contained a 12 by 25 array of 300 loops, representing 12 25-bit words.

The store had 4096 words and 342 cards were required to store them, which somewhat complicated the decoding. The cards were arranged in four shelves of 86 cards, the last one having only four usable words, and were held in close proximity to the drive and read tracks by pressure pads in such a way that that they could be easily replaced. There were 1024 drive tracks, each one driving four cards — one on each shelf.

There were four long flexible printed circuit read tracks and four sets of read amplifiers, one for each shelf. These were selected with two bits of the address, the other 10 bits selecting the drivers. The read tracks were arranged in concertina fashion, and such was their length that the time delay necessitated four different strobe times.

The store was only about a third of the size of an equivalent ferrite peg store and card punching was assisted by a Pegasus program. Only one production model was ever built.

For process control applications requiring more data storage, there was also an optional magnetic drum with a capacity of 50,000 words.

The Argus monitor panel displayed the contents of the accumulators and that of any store location, selected by the rotary switches; it also had hand switches for manual intervention and testing.

Argus had two types of interrupt. Interrupt was a novel concept in 1957, and Argus was possibly the first machine to use it outside the US.

In the previous experimental machine, the sampling time for the digital servos was not constant, as a result of the variable multiplication time of the Booth multiplier, among other things, and this made system analysis difficult. When a program contained conditional jumps, the situation was much worse and some method of getting a constant sampling time was called for.

The method adopted in Argus was to use a timer, which consisted of a register that was decremented every other word time (40 microseconds), independently of other operations. When the timer register became zero, the program was interrupted, the number in the instruction counter stored in a link register and the instruction register set to address 0, which was the start of the interrupt routine. The latter stored the contents of all the registers in a reserved part of the core store and then jumped to a subroutine that performed the required operation. At the end of the subroutine, the interrupt routine restored all the registers, incremented the number in the link register and transferred it to the instruction counter so that the main program could continue where it left off.

Thus it operated in a very similar way to interrupt routines in more modern machines. By using other registers as counters several different interrupt subroutines could be called at times that were multiples of the shortest time. This was a very useful facility for process control.

Another type of interrupt was the core store interrupt (CSI), which is now known as DMA (direct memory access). It was used to transfer blocks of data directly to and from the core store independently of the arithmetic unit when the latter was involved with multiplication, division and other operations not requiring store access. Asynchronous data to and from peripheral equipment was usually handled in this way.

A variety of peripheral equipment could be attached to Argus. Both the military and commercial applications involved input and output data in various forms: angular data from digitisers, analogue voltages, single bit on/off data and display information, to name a few. In process control applications, there was considerable input and output data as well as a requirement to handle paper or magnetic tape and to provide printed output, or output in analogue form for driving chart recorders.

Solid state switching was used to select digital and analogue inputs and outputs, which were to 10-bit accuracy, but thermocouple and other low level inputs were selected by relays. Because of the delay in selecting and digitising inputs, selection and reading in the data were arranged to be separately programmable operations, so that the computer could carry on with other operations and not waste time waiting. Also a "Jump on Busy" order allowed the computer to handle asynchronous outputs, like the printer, without waiting.

Once the transistor failures had been eliminated by switching from a soldered to a welded can, and the printed circuit connectors, whose contacts were brittle and prone to spreading, replaced, the machine achieved a high degree of reliability, as a result of the technology used. This was further improved by a checking program that was periodically run as an interrupt, which, if it detected a failure, could sound an alarm and isolate the computer from the plant.

Argus Process Control Applications

The first application of Argus was the military one for which it was designed — the Bloodhound missile pre-launch calculations. One of the problems in that application is a common radar one, that of smoothing and predicting an aircraft track. The radar information is in polar co-ordinates — range and bearing — and must first be converted to X-Y co-ordinates, or latitude and longitude.

Radar data is usually noisy: it can be smoothed by fitting a straight line or a low order polynomial to it by the method of least squares. The easiest way to do this is by use of orthogonal polynomials, and the noise reduction of the smoothing formulae so obtained is given by the sum of the squares of the coefficients. Taking an odd number of points and estimating the midpoint by means of a smoothing formula will always result in the greatest noise reduction: the greater the number of points and the lower the order of the polynomial the better, in this respect.

Unfortunately, such an estimate of position is out of date; if one wants to obtain an estimate of the latest point, or even one point ahead, to allow for the computation time, a large number of points must be used in the smoothing formula to obtain a useful reduction in noise.

The first process control application of Argus was to control a soda ash plant for ICI at Fleetwood. Following discussions between Ferranti and ICI at Nantwich, Alan Thompson of ICI realised that the use of a computer could be an economical solution if it not only took a supervisory role — printing plant and alarm data and calculating set points — but was also able to replace all the pneumatic 3-term controllers, of which there were just over a hundred.

The plant was a fairly docile one, so if the computer failed, as long as the control valves were isolated, it could coast along without danger or too much loss of efficiency. Three-term controllers take a weighted sum of terms which are proportional to the error and the differential and integral of it. The last two terms deal with stability and drift respectively.

Experiments with sampling servos and programming transfer functions to stabilise them, using the z-transform, had already been conducted with the small computer that preceded Argus. It was a revolutionary way of using a digital computer, because time was a parameter — as it is in a digital filter. Incidentally, a smoothing formula is really a low-pass filter and its frequency response can be calculated by z-transform methods.

The integral term could be obtained by summation instead of integration, although this needed to use double length arithmetic in order to obtain the necessary accuracy. The rate of change of error was found by taking the difference of two consecutive sample values, but noise due to quantisation and other sources necessitated smoothing the data. A smoothing formula like that used for the radar application would have used too much storage when many loops were to be controlled, so the solution was to use a recursive smoothing formula. Put simply, this involved taking a fraction K of the latest difference plus a fraction 1−K of the previous value of the rate as the new value for the rate, giving the following smoothed rate term:

$$y_n = K(x_n - x_{n-1}) + (1 - K)y_{n-1}$$

The degree of smoothing depended on the value of K — the noise was reduced by a factor K/(2−K). For example, with K=1/8, the noise was reduced to 1/15, as with a 15-point moving average filter, yet required the storage of only one extra value. The filter inserted a first order lag in the system, with time constant T/[(−ln(1−K)], T being the sampling time. For T=5 seconds and K=1/8, the time constant was about 37.5 seconds, which was small compared with that of the plant and did not affect loop stability.

The settings of the valves that had to be controlled were calculated sequentially every second for the fast loops — half of them — and every five seconds for the remainder. The valves were operated pneumatically; air was either admitted or exhausted at a controlled rate using solenoid valves. This only required two bits of information, so that the control signals for six valves were packed into one 12-bit word. The positions of the main valves, which took about 10 seconds to stroke, were measured with a potentiometer to 1% accuracy and examined 20 times a second so that they did not overshoot the dead space. This operation used 70% of the computer time, but left sufficient time for the other calculations, which were carried out less frequently.

Isolating the computer from the plant when the checking program found a fault was considered to be too drastic in certain cases, for example where the trouble was an instrumentation failure. An improvement was simply to indicate where the fault lay, so that the affected part of the plant could be controlled either manually or by conventional methods; the isolation procedure then being used only for computer faults, which happened rarely.

The computer was installed at Fleetwood in 1962, but after two years, the plant was shut down and the computer was used to control another similar plant at Winnington in Cheshire. Then in 1980, after 18 years of service, and still in working order, it was removed to the Science Museum in London.

Editor's Note: this article is based on the talk given by the author to the North West Group of the Society at the Manchester Museum of Science and Industry on 10 February 1998.

Obituary: Peter Hunt

Peter Hunt, a pioneer of large-scale software development, died in February aged 70.

Peter became one of the first UK computer users when he was employed, in the early fifties, in the Aerodynamics Department of the de Havilland Aircraft Company. During his three years there he learnt how to program Edsac, the Pilot Ace, Pegasus and the Elliott 401, and developed a variety of aircraft applications.

In 1955 he was recruited by Ferranti, initially to work on aircraft applications for Pegasus. In the hope of stimulating sales of this computer, he developed some of the earliest standardised packages. He was later responsible for software and customer support on Perseus, the first Ferranti commercial computer. In the early 1960s, as Head of the Lily Hill Laboratories in Bracknell, he was responsible for the project management of two of the largest contracts undertaken at that time by the UK computer industry, the Orion 2 system at the Prudential and the Atlas 2 system at the UK Atomic Energy Authority.

Following the takeover of the Ferranti computer division by ICT in 1964, he was made Head of Systems Development Organisation, responsible for developing a portfolio of systems and application software for the 1900 series. Under his inspirational leadership and sound project management more than 1000 programmers produced more than 100 software products, and as a result ICL won a Queen's Award for Industry in 1968 "for technical innovation in the production of software". Peter gave a talk to the Society on this subject at our 1900 seminar at the Science Museum in May 1996.

In 1968 Peter left ICT (which had by then become ICL) to take up a position as managing director of the UK branch of Leasco Software. Over the next 12 years Peter built the company up to become one of the most influential of the early UK software houses. Leasco Software specialised in real time software projects, many of which involved pioneering communications developments. Peter was also chairman for four years of sister company Leasco Response, which operated a timesharing bureau service.

In 1980 he left Leasco to operate as an independent consultant. During his later years he also developed a practice as an expert witness in legal cases concerning computer software in both the UK and the US.

Society Activity

Small-Scale Experimental Machine

Chris Burton

After moving the machine from the University to the Museum of Science and Industry in Manchester at the end of February, we continued to improve reliability and added various authentic details, for example some special stabilised power units and the correct Monitor CRT EHT unit. Other cosmetic improvements such as signwriting took place. The Remote Monitor was improved so that it could be incorporated in the barrier separating the public from the machine. A special comparator unit was designed and wired in to help find transient faults in the three CRT units. Experience helped us to optimise the setting up of the stores so as to be reasonably reliable, giving program runs of half an hour or more between failures.

There was intense activity around us in the museum during the week starting 15 June, as the galleries and supporting material for the SSEM were being prepared. The official "switch-on" took place on 17 June, using a satellite video link from the museum to the Bridgewater Hall, where 1800 people saw the mssage "HELLO WORLD 1948" stored in the CRT tube and displayed on the monitor.

By 19 June, the main CRT store had suddenly badly deteriorated, and would not store correctly. Intensive work over the next two days could not identify what had changed, so that by Sunday 21 June — the actual 50th anniversary — we felt the store was not reliable enough to get through the day's events. I therefore had to declare that for the day we had bypassed the main store. The machine was formally handed over to the safekeeping of the museum, and then Tom Kilburn and Geoff Tootill re-ran the first program at 1115, as they had done half a century before. We thereby met our goal in spirit if not to the letter. In the afternoon of 21 June Sebastian de Ferranti made the awards to the winners of the programming competition. This was the last of the week's events.

The substantial part of our project is now complete. There is a great deal of documentation still to finish, and we have to find out what has happened to the main store and fix it. We also have to work out a way of maintaining the machine and demonstrating it to the public, without the team feeling it has a job for life!

To have got this far is a stunning tribute to project team members Charlie Portman, Ken Turner, Keith Wood, George Roylance, Bill Purvis, Adrian Cornforth and Suzanne Walker, who have collectively put in nearly 11,000 hours of voluntary effort, and who have brought tremendous credit to the CCS. In material and supportive terms, the project has been made possible by the £150,000-worth of sponsorship from ICL, by the University of Manchester and the Museum of Science and Industry, and by numerous individuals.

Pegasus Working Party

Len Hewitt

Since the last Pegasus Working Party report in *Resurrection* issue 18 in autumn 1997, work has continued at fortnightly intervals. We have made remarkable progress, with some setbacks on the way. Pegasus has worked consistently well, and we have cleared a number of intermittent faults which had plagued us for a long time.

In February we started to test the spare packages in their number order in the positions recommended in the maintenance manual. I had to go to the USA in the middle of the month, and at the last meeting I attended before I left we had a major disaster.

We had got up to testing the Type 13 Clock/Reset package in position 10C. All previous testing had been done with HT on when removing and replacing packages. Things had been going so well that I had forgotten the significance of the power valves in the Type 13 package, and the heavy currents dissipated. This problem took a few meetings to resolve, as it was a multiple fault which affected more than the package we were testing. The fact that the problem arose on Friday 13th was just a coincidence!

We have now completed testing the spare packages, and moved on to testing the drum. We have had consistent failures on the drum tests for many years, and were concerned that it was a drum surface problem and that we would have to swap head connections under the drum covers to sort the problem out. In fact by swapping packages and then replacing some of the switching diodes in the read switch packages we were able to establish that the drum surface was not the problem.

We have now run through the complete drum tests several times, experiencing just an occasional failure on track zero. I believe by doing some more work on the drum packages we should be able to get even better reliability.

Our next major task is to get the machine working with the marginal voltages reduced. We were advised at the last CCS Committee Meeting that Pegasus could be in the "Making of the Modern World" gallery which the Science Museum is planning to open in the year 2000 — this was very heartening news.

I would like to thank the members of the Working Party — Derek Milledge, Martin Wingstedt and Peter Holland — for their support, and also Chris Burton, who has always had time to discuss problems even though he has been fully occupied with the Baby rebuild project.

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 the next page .

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**. The pages of information at our Web site include information about the SSEM rebuild project as well as selected papers from *Resurrection.* Full access to the FTP archive is also available for downloading files, including the current and all past issues of *Resurrection* and simulators for historic machines.

The Universal Resource Locators for a few of other organisations concerned with the history of information technology are as follows:

Bletchley Park (contains information on Colossus)
http://www.cranfield.ac.uk/CCC/BPark/
Manchester University (for its early computers)
http://www.computer50.org/
Science Museum
http://www.nmsi.ac.uk/
National Archive for the History of Computing
http://www.man.ac.uk/Science_Engineering/CHSTM/nahc.htm
The Virtual Museum of Computing (a rich source of links to other computer history resources)
http://www.comlab.ox.ac.uk/archive/other/museums/computing.html

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: hamish.carmichael@bcs.org.uk
George Davis: georgedavis@bcs.org.uk
John Harper: bombe@jharper.demon.co.uk
Dan Hayton: Daniel@newcomen.demon.co.uk
Len Hewitt: leonard.hewitt@virgin.net.
Roger Johnson: r.johnson@bcs.org.uk
Adrian Johnstone: adrian@dcs.rhbnc.ac.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

Forthcoming Events

5-6 September 1998, 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

29 September 1998 North West Group meeting

“The Early Days of the NCC”

20 October 1998 North West Group meeting

“From Freelance Programmers to F1 Group plc”

24 November 1998 North West Group meeting

“The Distributed Array Processor”

The North West Group meetings will take place in the Conference room at the Manchester Museum of Science and Industry, starting at 1730.

For information on London meetings, readers should refer to the insert enclosed with this issue.

Queries about London meetings should be addressed to George Davis on 0181 681 7784, and about Manchester meetings to William Gunn on 01663 764997.

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