# Engineering Credibility

The maintenance engineers on the London Atlas were a pretty bright bunch. They had to be. But they didn't take kindly to junior programmers who had failed to find bugs in their programs accusing Atlas of having a hardware fault – something that, albeit not a frequent event did occur from time to time.

Well it was a Tuesday morning. My 'phone rang and one of the young ladies from Data Control was asking me to take a look at one of the overnight batch runs which had failed. Having nothing more urgent to do, I wandered down to their domain to take a look. "This one always runs" I was told. "I don't understand it." I was handed the output. Now it has to be said that diagnostic output was not one of Atlas's strong points. You got the failing instruction plus the instruction on either side, and the value of all the non-zero B-Registers. That was it. In this case the problem was an illegal instruction – a floating point zero. The location was word 0 of store block 272 - 420000 in octal (I'm making this address up – 40 years on I simply can't remember, but it'll do to illustrate the problem). The next and previous words were the same so it seemed likely that the program has simply jumped into a previously unused store block which would have contained 512 floating point zeros. Beyond that observation I hadn't got very far. The program was written in Atlas Commercial Language (ACL) and I wasn't familiar with ACL's use of B-Registers.

Then I was approached by the doyenne of Data Control. Could I also have a look at this other failing program please? Well this second program was completely different but had failed almost identically. Look for the common factor then. Well, it turned out that both programs had employed COMPILER LOAD. What that? You might ask.

The Atlas Supervisor was designed as a compile and go system. For the most part programs were not run repeatedly without modification. But when they were it was necessary to recompile from source. And if you were doing that once a week for several years, that amounted to a serious waste of machine time. So somebody (I know not who) had hit on the idea of allowing users to save a store image of their program on magnetic tape with some red tape data to define how the program should be reloaded. But Atlas had no loader. Nor was there any obvious place to put one in the supervisor. So it was that "COMPILER LOAD" was invented. The user could call for it as if for a compiler, but instead of reading source code and converting it to an object code image in store, it read the program down from the specified tape into store. Not actually a compiler, but doing something similar and fitting into the philosophy of the Supervisor reasonably well.

Given that hardly any processor time had been expended in either of the failing jobs and given that I wasn't getting anywhere by other means, it seemed reasonable to look at the LOAD compiler to see whether the fault lay there. I found a listing in a dusty cupboard and took it down. Fortunately LOAD was written in Atlas's Assembler, ABL so it was possible to find out what was going on and happily LOAD wasn't a particularly big program. After half an hour or so I found an instruction which set one of the general B-registers to the value shown in the diagnostic output. The instruction was located at word 509 of store block 287 – 437775 in octal. The next 2 instructions would not have caused a jump and the one after that should have been word 0 of block 288 – 440000 in octal – 20000 away from where the failure had occurred. This seemed a bit of a coincidence. Suppose that

the carry had failed when adding 1 to the control register (B127). Well that fitted the observed behaviour and after a lot more checking everything else did too. Almost lunchtime now but I walked into the machine room and collared one of the engineers.

“Your machine has a fault”, I said. “No it hasn’t. Look!” came the very reasonable reply. And indeed, tapes were merrily spinning, lines being printed, cards read and paper tapes flying through the air with what we then called “gay abandon”. I persisted. “What do you thing is wrong with it then?” “Ah, the control register has a half adder associated with it doesn’t it?” I was bluffing a bit but it seemed a reasonable assumption. “Yeess” came the reply with a degree of suspicion attached to it. “I’m suggesting that the carry between bit 16 and bit 17 isn’t working.” “Engineers’ bits or programmers’ bits?” “Engineers.” I said with rather more confidence. What the engineers didn’t know is that I was a dropout from the Manchester University Computer Science course and I had learned a little about how the hardware worked, not least that engineers numbered their bits the wrong way round. I offered to write a little ABL program to demonstrate the fault. It did. “OK I’ll check”.

And so it was that 20 minutes later the machine was brought to a halt, a board was changed and Atlas was restarted. My new friend came back looking embarrassed but the two errant jobs were re-run without incident. For 24 hours I assumed heroic status in Data Control.

I became the engineers’ pet programmer. Every so often I’d get a request to write a little diagnostic program. It taught me one thing. Always take the trouble to talk to the engineers. Learn a little about what they do. They know things you don’t. A lesson which stood me in good stead for years afterwards