Atlas 2 at Cambridge Mathematical Laboratory
(And Aldermaston and CAD Centre)

Barry Landy, November 2012

How it came about

The Cambridge Mathematical Laboratory ("Maths Lab") had a long history of designing its own machines. EDSAC (1949-1958) was the world's first fully functional stored program (Von Neuman) computer on which it offered a full computing service for users (the world's first). EDSAC 2 (1958-1965), the first micro-programmed computer, carried on the innovative tradition.
.
However, by 1961 it was time to buy or design a new machine, but this time Maurice Wilkes didn't want another home-made one. He had £250,000 offered by the UK government available to buy a replacement for EDSAC 2. This could have bought a KDF9 but that was only about the power of EDSAC 2. The only viable machines were an IBM 7090 or an Atlas, neither of which could be afforded. The price tag for the Atlas was £2,000,000.

Peter Hall (Ferranti's Computer Manager at the time) suggested they sell Cambridge the processing unit of Atlas at "works cost", with Cambridge and Ferranti designing a simpler memory and peripheral arrangement, thereby providing a machine for Cambridge, and a cheaper version of Atlas for Ferranti to market. David Wheeler was design authority for the joint team, responsible for design of the memory access and peripheral coordinator and any consequential modifications to the basic hardware. The wiring diagrams were designed on EDSAC 2 by Roger Needham. While the hardware was built at West Gorton it was agreed that commissioning would happen at Cambridge and two engineers from West Gorton were seconded to Cambridge for that purpose. The Eagle pub welcomed them with open arms!

The machine was officially christened Titan, in line with the Ferranti tradition of choosing names from classical mythology, but was later re-named Atlas 2 by the marketing people. In Cambridge the name Titan stuck as the prototype of Atlas 2.

Hardware – what was different

Atlas had a full one-level store with page address registers and a high-speed drum, and these were removed for Atlas 2. Instead user programs were located dynamically by hardware (more details later) thus saving huge hardware costs.

The Atlas instruction had a 10 bit function part; if the first bit was zero the remainder defined a full set of hardware implemented instructions. If the top bit was one the remaining bits defined (up to) 512 "extracodes" which called supervisor functions (like accessing magnetic tape). On ATLAS these were implemented in fixed memory which could be modified by the operating system design team (brushes); to save cost, this was changed for TITAN so that the extracodes were implemented entirely in Main Memory. This was much slower for instruction access but considerably simpler to maintain.

Simplifications were also made to the peripheral control logic.
Atlas 2 was thus of a similar speed to Atlas but without the flexibility and power of the one-level store and with several other cost saving decisions, including slower main memory (the Atlas 2 design permitted 2 microsecond or 5 microsecond memory; naturally we had the slower). It was therefore much cheaper!

In common with Atlas the memory consisted of words 48 bits long, which could be divided into 8 separately addressable 6 bit characters or 2 24 bit half words. TITAN initially had 32K words of store, soon increased to 64K and eventually to 128K (so 1Mb of 6 bit bytes).

Hardware arrived in Cambridge in 1963; software design had already started (on paper); development could start as soon as enough of the hardware was commissioned.

The memory management system comprised a single base and limit pair of registers with two other registers to lock peripheral transfers implemented as 4 registers, R1-R4.

R1 specified the starting point of a program; the user address was ORed with R1 to provide the absolute hardware address being accessed. The OR feature made allocating main memory quite complicated.

R2 specified the size of the program region.

R4 specified the lower end and R3 the upper end of a locked region within the R1,R2 limits; the program was not allowed to access this sub region. This feature permitted IO transfers (especially intended for magnetic tape).

This design of memory control allowed full user protection, so that, even though multiple programs might be active, a program could only read or write the area specified in the control registers. The major drawback was that the memory allocated to one program had to be contiguous in real memory.

Software – why the Atlas Supervisor couldn't be used

It had always been Cambridge's intention to write a multiprogramming supervisor similar to Atlas comprising a job shop system (that is, jobs input from paper tape or punched cards taken from a physical queue, with the results posted back in a similar queue) with efficient I/O buffering to maximise throughput. The Atlas supervisor, which was already being written, achieved this aim by managing and exploiting the one-level store. For Atlas 2, having no one level store, we had to design and implement a different system, although we learnt from the Manchester effort and used the same methodology in terms of Interrupt processes (IRs), Supervisor Extracode Routines (SERs) and so on.

The Atlas 2 Supervisor was a joint project of Ferranti and Cambridge. In the initial design stages the software team was small, comprising David Barron and David Hartley from Cambridge and Chris Spooner and others from Ferranti (which became ICT in the course of the development). In 1963 Barry Landy joined the team, and his first task was to write a diagnostic system to be used on an IBM

selectric typewriter to enable the supervisor to be debugged. (Although he had first to write a program to discover the character code!)

Meanwhile David Hartley was writing the nucleus, which while based on the Atlas work, had to be different from the Atlas nucleus because of the lack of a one level store. Roger Needham joined the software team when he had finished helping David Wheeler with the hardware developments.

The main external distinction between the two systems was that Atlas used a drum for page and IO buffering, and Atlas 2 did not have drums, and so was designed to make use of magnetic-tape for I/O buffering, known as the "swinging well". This in practice never operated at Cambridge but did in the second Atlas 2 at Aldermaston (see below). While it worked at Aldermaston in their "one job at a time" mode of working I personally doubt it would have worked well for a multiprogramming system. Luckily as it happened we did not have to find out.

Magnetic tape was also used for paging-in parts of the supervisor which were not permanently resident in main memory. The entire Supervisor eventually comprised some 40K words of which about 15K was permanently resident. The remainder was supervisor pages, written in blocks of 512 words, which could be transferred to a free block of memory in one IO operation.

The ICT team eventually grew to 21 people, and the integration of all these pieces of software written by many hands of various standards of ability was done in Cambridge, where by 1964 a day shift for users was operating (as happened on Atlas 1) under a temporary supervisor and a temporary compiler developed by Peter Swinnerton-Dyer. As a result the testing and integration of new pieces of the Main Supervisor had to be done in the evening and night (under the control of Barry Landy).

Eventually the team divided into two; the Cambridge team to finish the system as planned, and the Ferranti/ICT team to modify it to suit Aldermaston

Cambridge University service – how time sharing rescued the reputation

In 1965, during the Cambridge development, Wilkes discovered Time Sharing at MIT, and demonstrated it at Cambridge using a 10 character per second teleprinter (!), on a telex line across the Atlantic. He insisted (rightly) that the supervisor should be modified in mid flight to permit on-line interaction with terminals. Since we had insisted on maintaining multi processing job control we were fortunate that the basics of the operating system were already suitable (or seemed to be).

However there were plainly many things lacking in the hardware.

As originally planned TITAN was fine for a job queue system with no permanent storage (the way that EDSAC 2 worked). A terminal interaction system required a file store and some basic swapping. In the original system design, memory allocation to jobs (ie, selection of which jobs to run) was rudimentary. Jobs were selected from the

(tape) queue based on whether they would fit into the available memory on a first come first served basis. This would not work for a multi user system and so more sophisticated control was needed. Finally there had to be some way of interfacing the terminals (10 cps teletypes) to the hardware.

These problems were variously solved.

A Data Products 16M word disc was acquired for the filestore in 1965 (actually as a gift from Basil de Ferranti). TITAN eventually had two such discs. Each disc had independent heads for each platter, and most importantly a fixed head region which in effect we used as a drum.

The disc was also used for swapping supervisor pages and for buffering the data streams of active programs instead of using magnetic tape.

David Wheeler altered the design of the memory location registers described above by adding an extra control register R5.

R5 specified in the bottom 18 bits the address of a piece of store which could be 64, 512, or 4096 words long; of the top 6 bits 2 control bits were used to specify the size, and the remaining 4 bits specified that one or more of four particular pages of the region (2048 to 4096 relative) were locked for transfers.

This enabled a shared program (carefully written to be re-entrant so that it could be used simultaneously by many different processes) whose memory region was controlled by the first pair (R1,R2) to also have a small dedicated piece of data controlled by R5 (addressed by giving the top octal of the address the value 2; J2). Small though this piece of workspace was, it was fundamental to the success of the system, as the basic elements of an interactive process could be very quickly swapped in and out, using the fixed head region of the disc.

In order to keep as much memory free as possible to enable as many users as possible at one time, the limit register controlling one program was used dynamically so that a program only had the contiguous region it actually used (as opposed to what it declared).

Finally a terminal multiplexor was designed and manufactured to enable 64 devices to connect to the system. The multiplexor built up the character stream from each device in 64 buffers each holding one character, and signaled the system when a character was ready by posting an interrupt. In 1967 the first modems appeared and we connected one to a multiplexor line and demonstrated the working system live in October 1967 from Southampton (Datafair).

The supervisor was modified to use the disc hardware (by Roger Needham and Barry Landy) primarily by changing the "swinging well" to operate on the disc, and by providing a user file store on the disc, and also to use the improved memory control (R5) for shared processes (Barry Landy).

Mike Guy programmed the multiplexor interface.

There were of course other software needs as we quickly discovered. In a traditional system running jobs, all the needs of the job are declared in the Job Description; the input and output streams; the programs that will be run to use them, the memory size required, and so on.

In an interactive system these are all discovered "on the fly" and extracodes had to be provided to implement these needs, including creating an input stream from a disc file or from a previous output stream; creating an output stream; and calling a named program with a specified amount of memory.

There followed a steep learning curve as we discovered the special needs of a shared interactive system. In-house testing of the system showed us the need to deal with rapid response to things like the context editor, and as a result the basic sharing logic which decided which job would be active at a given time had to be adapted to allow for the need for quick response for interactive tasks (jobs on a queue have no such demands). The "Master Space Scheduler" was born which in effect controlled how many tasks may be active, and the CPU scheduler had to be very sensitive to the needs of the running programs, and in particular if they were CPU intensive or interaction intensive.
David Hartley wrote a logging on process, David Barton a command program, Steve Bourne an editor; Sandy Fraser joined the team and designed an access control system, and a tape based backup and archive system for the file store (a complete necessity bearing in mind the lack of robustness of the disc store).

The main supervisor was live for users early in 1966, and the time sharing system was available for staff in October 1966. By 22 March 1967 we were confident enough to have a public opening for users, after which it was available whenever TITAN was running.

Interactive graphics techniques were also being pioneered by Charles Lang.

Of course timesharing placed heavy demands on the hardware and it was important to be as efficient as possible in all aspects of the supervisor and to use any hardware aids. Two tunnel diode stores were developed at Cambridge; one, which worked very well, speeded up the fetching of operands, the other was intended to speed up the fetching of instructions. The idea was that most instructions are obeyed in sequence, so when an instruction was fetched that word was placed in the slave store in the location given by the fetch address modulo 32; the remaining bits of the fetch address were also stored. If the wanted word was in the slave it was read from there instead of main memory. This would give a major speedup to instruction loops up to 32 instructions long, and reduced effect for loops up to 64 words.

Unfortunately the store was too unreliable and gave a parity fault about every five minutes. Had the store automatically flushed itself on these errors all would have been well, but instead it crashed the system and so this slave could not be used.

Eventually Barry Landy had a great idea. He wrote code to trap the parity fault and (in a short loop) rewrite the whole of memory thereby flushing the slave store cache and allowing the system to continue as though nothing had

happened. Measurements showed that even with errors as frequent as every five minutes there was a significant gain in system efficiency.

Among other similar ideas, it sometimes happened that the allocation of memory became jammed up (basically as a result of attempting to put several quarts into a pint pot). When this was detected (by a fairly sophisticated piece of software) all the running programs were swapped out, freeing lots of memory and allowing the system to unknot itself. Users would probably never notice.

What happened next?

From the software engineering point of view the major improvements were provided by exploiting the benefits of interactive working. The diagnostic system was re-written to work interactively against a system dump, thus saving a lot of dedicated debugging time. An intermediate language was designed (IAL) so that in a way that became very familiar later, modules could be compiled into IAL and later linked together. The supervisor assembly system was recast to run online. An integrated system (TSAS) was provide to take the current canonical sources, and a set of edits, to apply the edits, to assemble into IAL, and then to link the supervisor from that.

From the users point of view there were continual improvements to the control language and editor, and provision of a number of programming languages; as well as Autocode, FORTRAN and BCPL were available.

The online system (Cambridge Multiple-Access System) was a great success which I think the original swinging well system would not have been. So successful that it caused great difficulty with the inadequacies of the next purchased system – but that's another story.

TITAN continued running until 7 October 1973.

Others who contributed to the initial system were MJT Guy, PR Radford and JC Viner.

Aldermaston – the machine they didn't want

At the time AWRE at Aldermaston were bidding for an IBM Stretch, but the government insisted that they buy British. Ferranti/ICL sold them an Atlas 2 on the strength of which Ferranti built two production models.

The Ferranti team swelled to 21, but their design aims diverged from Cambridge since Aldermaston were not prepared to accept multi-programming (more than one program in memory at a time) on security grounds. A restriction like that is of course easy to build in to a multiprogramming system so we continued to work together for some time. But Time Sharing would have been a complete No No.

CADCentre – the machine ICL couldn't sell

For some time, the second production machine remained unsold. When it was seen that the Cambridge machine was providing a successful service thanks to the time sharing facilities, HMG searched for ways to have it procured to provide a service to industry. Hence they founded the CADCentre in Cambridge, located close to the University and exploiting the Cambridge Supervisor. Strictly, CADCentre acquired the Hardware of Atlas 2 from ICL, and the supervisor from Cambridge University. From the hardware point of view the basic difference between this machine and TITAN was that it had page translation tables so that the memory allocated to a program did not need to be contiguous in absolute memory addresses. To allow for that in the TITAN supervisor only required a small change.

I am grateful to David Hartley for assistance in preparing this reminiscence.

References

Barron DW, Fraser AG, Hartley DF, Landy B, Needham RM (1967). File handling at Cambridge University (Proceedings AFIPS Spring Joint Computer Conference, 1967)

Landy, B and Whitby-Strevens (1968). TSAS - the time shared supervisor assembly system, Computer Journal vol 12

Hartley DF, Landy B, Needham RM (1968). The structure of a multiprogramming supervisor, Computer Journal vol 11.

Landy, B and Needham, RM (1971). Software Engineering Techniques used on the Development of the Cambridge Multiple-Access System; Software Practice and Experience, I 167.

Landy, B. (1971) Development of Scheduling Strategies in the TITAN Operating System, Software Practice and Experience, I 279

Hartley, DF, (1967) TITAN multi-access reference manual

Landy, B. (1968). TSAS - The Time-Shared supervisor Assembly System Reference Manual