- Medicine: Rensselaer Polytechnic Institute.
- Science: Pittsburgh Supercomputing Center.
- Eduction and Academia: CAST (Center for Applied Special Technology).
- Environment, Energy, and Agriculture: Environmental Resources Information Network (Australia).
- Transportation: Baystate Shippers, Inc.
- Business and Related Services: McKesson Drug Company.
- Finance, Insurance, and Real Estate: Johnson & Higgins.
- Government and Nonprofit Organizations: Los Angeles County Department of Public Social Services.
- Manufacturing: United Technologies Corp., Sikorsky Aircraft.

Interested readers would do well to obtain a copy of the program to learn of the achievements of the winners and those of the nominees and finalists. The program and additional textual and visual information are available on a specially created CD ROM. A limited number of program books and CD ROMs are available from the *Computerworld* Smithsonian Awards Program: (617) 349-3704.

We hope that this awards program can be continued though the years, and that the sponsors and the chairperson's committee can be encouraged to extend their involvement in stimulating the innovative application of information technology to our worldwide community. We would hope also that, through the association with the Smithsonian Institution, further stimulus can be applied to recognize and understand the contributions of our pioneers.

J.A.N. Lee

Anecdotes

JAMES E. TOMAYKO, EDITOR

The Anecdotes department is an opportunity for participants in the history of computing to contribute reminiscences of salient events. These stories can vary in scale from the origins of a term to first-person accounts of critical turning points.

Since the material in this column often represents personal views tempered or sometimes weakened by memory, the editor invites other opinions and evidence.

Editor's note: In this anecdote, Paul Thomas recounts the design and construction of an early solid-state computer used for pedagogy at the University of Glasgow. Thomas was also formerly at Brock University, St. Catharines, Ontario, Canada.

Solidac: An Early Minicomputer for Teaching Purposes

This is a short note concerning the development of a small computer which was started in 1958, with construction being completed in 1963. The design was fully described in my PhD thesis[1] but was unpublished, which was the norm for theses at that time, the thesis itself being considered a publication in its own right. However, there were a number of historically novel features that I thought might be of interest to the readers of this journal. Due to the passage of time, and as I later became distanced from the original project (to be explained later), some of the information and dates given are only approximate. For this reason, I have written this note in a more informal manner, consisting of three parts: A prologue describes what led to the project, the section "Computer description" reviews the important features of the computer design, and an epilogue describes the outcome of the project, somewhat different from the original project purpose.

Prologue

In 1956, while teaching in the Electrical Engineering Department at the University of Glasgow, Scotland, I became interested in electronic computers and started teaching some of the basics of both analog and digital computers as part of a course in electronics. In April 1956 I attended a convention on digital computer techniques (sponsored, I believe, by the Institution of Electrical Engineers), at which the following statement was made in one of the many discussion sessions: "It was suggested that one of the problems was to instil an understanding of digital computers into the engineering world, starting right back at the students in the Universities and Technical Colleges." With these thoughts in mind, I decided that it would be desirable to have a small analog and a small digital computer purely for teaching the basic operations of each of the two types of computer, as opposed to using the large university computer, which was used for regular computing purposes and which was certainly unsuitable for the purposes that I had in mind. Obviously, the large machine would not be suitable for teaching analog computing or the internal operations of a digital computer; nor was it really satisfactory even for teaching fundamental programming at the machine-language level. Hence, in 1956, I went ahead and proposed that the University of Glasgow provide funds to develop these two computers. This was granted and I proceeded by developing the analog computer first, believing that this would be the easier and quicker one to develop, which indeed turned out to be true. Furthermore, it also turned out to be fortuitous, as I explain later.

When the preliminary design for the digital computer was drawn up in 1956, it was intended to use vacuum tubes for the active elements, as the cost of transistors was still very high. The main memory was to be a 1,024-word magnetic-core memory. It was furthermore decided to use a straight binary format with a word length in the range of 16 to 32 bits. By the time (1958) that the analog computer had been built and I returned to the design of the digital com-

Group	OPC	Purpose
0	0	Stop instructions
	1-9	Operations performed on a B-register (index register)
1	12-15	B-register test operations
		As the above instructions used the B-register contents, these instructions could not be modified by the contents of another B-register.
2	16, 17	Special modifier instructions (discussed in the text)
3	21-25	Accumulator test operations
4	26, 27	Program counter initialization (discussed in the text)
5	10, 20, 28, 29, 60, 61	I/O control
6	31-34	L-register operations
7	35-39	Accumulator shift operations
8	40-49	M-register operations
		The "double-length" accumulator (A) consisted of two portions: M (most significant half) and L (least significant half).
9	51-55, 59	D-register operations
		The D-register was used to hold the multiplier in the multiplication operation or quotient of the division operation.
10	56, 58	Multiplication and division

Figure 1. Groups of instructions in the instruction set.

puter, the price of transistors had dropped quite considerably and I decided to use them instead of the vacuum tubes, as this would considerably reduce the physical size and power requirements of the computer. As a result, the name of the computer (as it was the trend in those days to give computers names) was Solidac (solid-state automatic computer).

A second fortuitous effect brought about by the two years' delay concerned the actual construction of the computer. The original plan was to have the computer constructed by the department, which could have been slowed up due to other commitments; nothing could have been done to avoid this, as it was the only method available to me. However, at this time a local company, Barr and Stroud, a long-established firm making optical instruments including binoculars, submarine periscopes, and optical range finders, decided that it was time for them to move forward into developing electronic devices, specifically electronic range finders. One of their staff members, T.H. O'Beirne, approached Professor D.C. Gilles of the Department of Computer Science at the university to see if they had some need for special computing equipment that Barr and Stroud could develop for them, so that they could gain expertise in this field. As Gilles knew of my plans, he suggested to O'Beirne that he discuss the matter with me with the hopes that something could come out of this meeting to our mutual benefit. As a result, it was agreed that I would supply the components and Barr and Stroud would essentially carry out the construction of the computer. The final development would be a joint effort between us, based on my original design, with improvements being carried out by joint agreement. The design, which incorporated a number of features to gain the maximum capability for the cost, was finally completed in 1963.

Computer description

Having made the decision to use all solid-state devices, the next considerations to be made were the word length, memory size, and instruction set, bearing in mind the purpose of the computer and cost restriction. It was generally considered to be good practice to use a word length of a power of two (commonly used lengths being 16 or 32 bits), though this was not always adhered to for economic reasons. Sixteen bits was not unreasonable for the instruction length but not really enough for the data length, for which 32 bits would be more reasonable. This problem was commonly overcome by either using a 16-bit word for the instruction and two words for data or using a 32-bit word for the data and packing two instructions into a word (such as in the IAS or von Neumann design, which packed two 20-bit instructions into a 40-bit word).

Neither of these solutions seemed ideal for a teaching machine; either would cause complications for a student learning machine-language programming. As a compromise, it was finally decided to use a word length in between, namely, 20 bits, which seemed to be a reasonable word length for both purposes. The fact that it was not a "standard" length was not serious, as the memory unit was going to be constructed by us from planes of 32 words each, being hand wound using individual magnetic cores obtained from Mullard, Ltd., a company that had become deeply involved in the manufacture of such devices. The breakdown of the bits used in the instruction format required a certain amount of consideration based on reasonable future expansion capability together with a reasonable instruction set for teaching purposes. The final arrangement was

OPC	XR	Address	
6	3	11	bits

This allowed for 64 operation codes; seven address modification index registers (0 being a nonexistent register corresponding to "no modification"), initially only three being provided; and 2,048 memory locations. Again, for cost reasons, initially only 1,024 memory locations were provided. This figure today must seem unrealistic, but in fact quite a lot of computing was possible due to the relatively large instruction set. The instruction set was divided into nine basic groups of instructions (with a few exceptions), as shown in Figure 1.

Operational speed. One of the problems involved in using low-cost transistors was that the basic pulse speed was low, about 50 kHz. This estimate was based on a number of preliminary tests carried out on a sample of the transistors, type GET103 (manufactured, I believe, by General Electric). In fact, when the computer was finally completed, the speed came out to be only about 30 kHz. The list of the instruction codes shows that all instructions did not have to have the same execution time due to the different operand lengths; most immediately obvious are those handling data (20 bits) and those involving address modification (11 bits). The idea of minimum operation timing was used. Here the two phases of operation were optimized, resulting in the approximate relative timings shown in Figure 2.

General construction. For portability, the complete computer was designed to fit into a standard desk console, the various units being placed in the various drawers, as shown in Figure 3. So the students could have access to the circuits, using an oscilloscope to monitor the various waveforms in the different circuits, a novel construction was used in which the circuits, made up on individual small boards, were mounted on larger boards, "leaves," which were pivoted so that they were normally in the vertical position but could be rotated through an angle of 45 degrees in either direction. Thus, adjacent leaves could have an angle of 90 degrees between them, giving good accessibility to the components, as shown in Figure 4 and close up in Figure 5. For normal use, a panel display was provided, with display lamps that allowed the user to observe the contents of any register, particularly under single-step operation.

Novel circuit features. Having given an overview of the computer, the remainder of this section is devoted to indicating some of the novel features introduced into the computer design: special instruction modifiers, ROM bootstrap, and complex arithmetic operations.

Special instruction modifiers. The normal method of instruction modification was to use an index register, known at the time as a B-register or B-box. I believe this term was first used by the Manchester University group, and I presume that this was to differentiate it from the A-register (accumulator). The length of these registers was equal to the address-field length of the instruction (11 bits in this case), and one of these registers was used to modify the address portion of an instruction. The contents of any one of these registers could be changed or tested using one of the instruction OPCs 1 to 15.

Obviously, we could not use a B-register to modify the address portion of such an instruction, as the XR field was specifying the B-register whose content was being modified or tested. To provide address modification for these instructions, a special modifier instruction (no. 16) was provided, which "added the value of the address field of this modify instruction to the address field of the next instruction." This could, of course, also be used to modify the address of any instruction. A further modifier instruction was provided (no. 17), which allowed the content of the entire next word (as data) to be added to the entire next instruction (after the "data" word). This let the programmer modify any part or the whole of an instruction, specifically the OPC.

Instruction retrieval phase

Short (7 cts)	Where there was no instruction modification required
Half (18 cts)	Where there was instruction modification by a B-register
Full (28 cts)	Where there was instruction modification by function no. 17 and possibly a B-register as well.

Instruction execution phase

Short (7 cts)	Where only parallel transfers were involved
Half (18 cts)	B-register operations
Normal	20-bit data operations (28 cts)
Double length (47 cts)	Operations using double-length data, accumulator
Variable length	Operations such as multiplication, which depended upon the actual data

Figure 2. Approximate relative timings (cts is basic clock times).

Maurice V. Wilkes, who was my PhD external examiner, gave me great encouragement at the oral examination (probably unknowingly to him!) when, early on in the examination, he said, "I wish that we had had this type of instruction on the EDSAC." He probably has forgotten this by now, but I remember it to this day.

ROM bootstrap. A major consideration in the design was the question of the initial starting procedure of the computer when first switched on — or bootstrapping. At the time, the

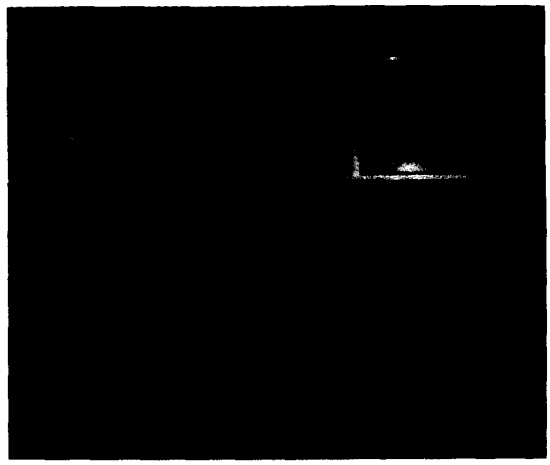

Figure 3. Front view of Solidac.

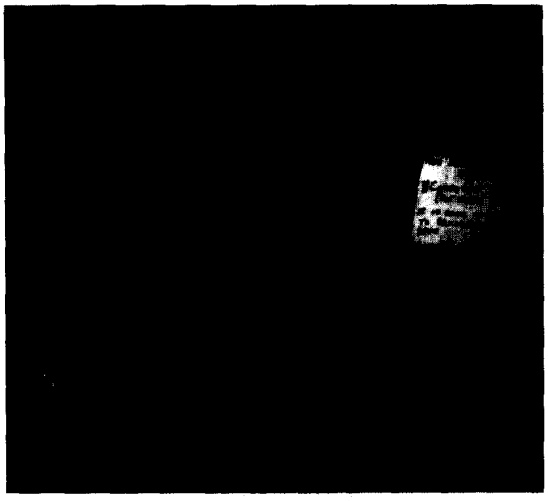

Figure 4. Side view of Solidac showing one drawer open.

general method used was to manually load a few instructions through a set of switches on the front panel, basically enough to activate a paper (or possibly a card) reader. This could then read in a more sophisticated bootstrap program, which could then read in programs as required. As an example, the Digital Equipment Corp. PDP 8 (which appeared later and with which I became very familiar in 1963) had an initial hand-loaded bootstrap program of 15 instructions, which then read in a longer bootstrap program.

In the case of Solidac, it was decided to place the initial bootstrap in a small memory which could only be read (a ROM), thereby not requiring the normal manual loading — this, of course, being common today. Having decided to do this, it made good sense to combine the two bootstrap programs into one, so the entire "program" was about 90 instructions long. As each main memory plane was to be 32 words long, the same was used for the ROM, thus requiring three planes. In fact, the ROM was constructed by using the same form as the main memory, simply replacing the 0's by small plastic rings, the same size as the magnetic cores, which were of course used for the 1's. Thus, when a word was read, the cores were then easily automatically rewritten back by passing the appropriate current pulse through the entire word, writing 1's back into the word cores and, of course, leaving the 0's (plastic) unaltered.

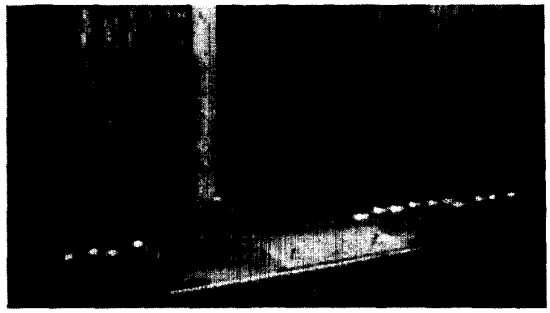

Figure 5. Top view of open drawer showing "leaves": component boards on left, wiring on right.

Having made this decision, the next issue was the placing of this ROM in the total memory address space. Conveniently, it should be placed to start at absolute address "zero" so that at start up it was necessary only to initialize the program counter to zero, a trivial electrical operation. On the other hand, it is also convenient for the user if programs could start at memory location zero. This problem was solved by having the hardware initially select the ROM and then automatically switch over to the normal memory after the bootstrap program was completed.

It was quickly realized that certain simple functions in the ROM could be used by user programs. Hence it was arranged so that the selection of the memory being used could be carried out under user-program control using a special branch instruction (no. 27). This changed the store, letting the user specify the required memory location from which the next instruction was to be extracted from the selected memory unit.

Complex arithmetic operations. At the time that this computer was designed, it was not uncommon to perform the more complex arithmetic operations (that is, those other than addition and subtraction) by means of subroutines, though these were relatively slow, particularly with a serial computer. For this reason it was decided to provide hardware to perform these operations — specifically, the operations of multiplication, division, and normalization of floating-point numbers. Both the multiplier and divider handled signed numbers, the divider using the standard nonrestoring method.

The multiplier was unique in that it was a modified Booth algorithm multiplier that used 3 test bits instead of his 2-bit test. Although the method was new at the time, I have seen an article describing the method in much more recent times. Basically, in this method, instead of comparing the least significant bit of the multiplier and an additional least significant bit (as in the Booth method), the two least significant bits of the multiplier are compared with the additional bit. Thus, as two bits of the multiplier are used at a time, this brings about a doubling of the speed of operation compared with the original Booth method.

The normalizing operation was simply a modified shifting operation which was relatively easily implemented and therefore considered worth incorporating into the hardware. As all integers were in the fractional form, the normalized form used was such that the most significant bit of the fraction was always opposite to the sign bit (positive numbers of the form 0.1... and negative numbers of the form 1.0...). Thus, the normalizing operation simply required comparing these two most significant bits and shifting the fractional part to the left until this condition was met. The only problem with this was when the operation resulted in a legitimate exception which could be rectified — for example, the addition of two positive numbers, which could cause a temporary overflow condition. This was handled by having an additional more significant bit attached to the accumula-

tor. This "overflow" bit was normally equal to the sign bit. However, in this overflow case, the sign bit would finish up opposite in value to the overflow bit, necessitating a single right shift of the fractional part. In all cases, the amount of shift was subtracted from the exponent part of the floating-point number (stored in another memory location). Again, this function was provided as a convenience to the programmer and also to speed up floating-point operations.

Epilogue

In 1961, staff changes took place in the Electrical Engineering Department at the University of Glasgow, bringing about a change of research and teaching orientation. That, coupled with some personal reasons, caused me to move to Canada in 1962. As pointed out in the prologue, the computer was not completed until 1963, after I had left. So what I am reporting now is essentially second hand, partly from what I gained from my periodic trips back to Scotland and partly from personal correspondence. Because of this, some of the details and dates are now a bit hazy, but I will try to be as accurate as possible.

By the time that the computer was ready to be shipped to the university from Barr and Stroud, the department stated that it had no real need for the computer. (How many university projects have been disbanded due to a change of emphasis within a department!) It was agreed that it should remain with Barr and Stroud, at least for the time being. O'Beirne said that he had a personal interest in using it to carry out some research into using a small digital computer for music composition, not so much as "to take the part of the composer, but [we] here are concerned with computers in the role of the executant."[2] In fact, during the next few years, he used Solidac to compose interactively many pieces of music (including Mozart and Haydn dice music) and, because of his Scottish background, finally music for the Highland bagpipe.[3] He also produced an interactive program called Orpheus, about which he said[4]

> Previous considerations are important in programming work, but understanding of them is not vital to musicians who may be interested in making use of the computer. This applies particularly to the ORPHEUS programs which allow the computer to play music after entry of a data tape which is as near to a straight transcription of a conventional score as can be secured within the conventions of five-hole teletype input. A special feature of these programs is the fact that modifications to pitch and speed — independently of each other — are readily made by simple operations at the computer console.

It should perhaps be pointed out that input to Solidac was normally by means of a five-hole teletype paper-tape reader using the Ferranti code (as used in the Pegasus and Sirius computers).

In 1969 the computer was turned over to the Department of Computer Science at the university, which decided that it could be used for its original purpose after some necessary repairs. I understand that Solidac was in fact used by the department for some years to teach machine-language programming before being disbanded in the early 1980s. It was then planned to place it in the university museum or some other museum, but the last time I saw it — about eight years ago — it was still in a basement storage space of one of the university buildings! Maybe someone else reading this article can fill in information on its final demise.

From a personal point of view, it is worth mentioning that when I went to the Computer Science Department at Brock University in St. Catharines, Ontario, Canada, in 1975, among other courses I had to teach an introductory course on machine-language programming. At the time, the only computer available for computing was a Burroughs B5700, but this did not have a machine language available in the normal sense and a simulator program for the IBM System 360 was being used. It happened that it had some bugs in it, which were a little disconcerting for anyone using it, so I looked at possible alternatives. There was another machine simulator available (whose name escapes me now), but this was for a decimal machine, which I was not too happy to use. I therefore decided to write my own simulator, but for what machine?

After some thought I decided to use Solidac as a basis, as I was familiar with its operation and it seemed to cover all the operations required and was reasonably easy to program. Thus I created a simulator for the computer, Brosim (Brock simulator), which could be programmed at the machine-language level, and also an assembler, Brassy (Brock assembly), for students to write and execute symbolic-language programs on the simulator. This worked out extremely well for the intended purpose and was used until the university obtained a DEC VAX 780 computer some years later, and the programming course was run on this machine using its own assembler program.

Paul A.V. Thomas
#503, 500 Talbot Street
London, Ontario
Canada N6A 2S3

Acknowledgments

I would like to acknowledge the help given to me on the project by D.C. Gilles (now retired) and J. Leech (recently deceased), formerly of the Department of Computer Science, University of Glasgow, Scotland, and T.H. O'Beirne (since deceased) of Messrs. Barr and Stroud, Glasgow, Scotland.

References

1. P.A.V. Thomas, *The Design Philosophy of a Small Electronic Automatic Digital Computer*, PhD thesis, Glasgow University, Scotland, 1961.
2. T.H. O'Beirne, "Music, Numbers and Computers," *Bull. Inst. Math. Appl.*, Vol. 3, 1967, pp. 57-66.
3. T.H. O'Beirne, "From Mozart to the Bagpipe with a Small Computer," *Bull. Inst. Math. Appl.*, Vol. 7, No. 1, 1971, pp. 3-8.
4. T.H. O'Beirne, "Computer Programs Which Play Music with Microtones," *Computer J.*, Vol. 13, No. 4, 1970, pp. 344-349.