Turing's Zeitgeist

Brian E Carpenter
brian@cs.auckland.ac.nz

Robert W Doran
r.doran@auckland.ac.nz

Department of Computer Science
The University of Auckland

Abstract: **This chapter reviews the history of Alan Turing's ACE design proposal and how he came to write it in 1945, and takes a fresh look at the numerous formative ideas it included. All of these ideas resurfaced in the young computing industry over the following fifteen years. We cannot tell to what extent Turing's unpublished foresights were passed on to other pioneers, or to what extent they were rediscovered independently as their time came. In any case, they all became part of the *Zeitgeist* of the computing industry.**

Introduction

At some universities, such as ours in New Zealand, the main computer in 1975 was a Burroughs B6700, a 'stack' machine. In this kind of machine, data, including items like the return address for a subroutine, are stored on top of one another so that the last one in becomes the first one out. In effect, each new item on the stack buries the previous one. Apart from the old English Electric KDF9, and the then recently introduced Digital Equipment Corporation PDP-11, stack machines were unusual. Where had this idea come from? It just seemed to be part of computing's *Zeitgeist*, the intellectual climate of the discipline, and it remains so to this day.

Computer history was largely American in the 1970s – the computer was called the von Neumann machine and everybody knew about the early American machines such as ENIAC and EDVAC. Early British computers were viewed as a footnote; the fact that the first stored program in history ran in Manchester was largely overlooked (which is probably why the word 'program' is usually spelt the American way). There was a tendency to assume that all the main ideas in computing, such as the idea of a stack, had originated in the USA.

At that time, Alan Turing was known as a theoretician and for his work on artificial intelligence. The world didn't know he was a cryptanalyst, didn't know he tinkered with electronics, didn't know he designed a computer, and didn't know he was gay. He was hardly mentioned in the history of practical computing.

There were clues. The first paper published on the ACE did say 'based on an earlier design by A M Turing'.[1] A trade press article by Rex Malik[2] described Turing as 'a four in the morning system kicker' –

that didn't sound like a theoretician. There were growing rumours of secret stuff happening during World War II, and there were Brian Randell's 1972 paper[3] and 1973 book.[4] In 1972, NPL issued a reprint of Turing's 1945/46 ACE proposal as a technical report,[5] and the Ultra secret was finally blown in 1974.[6]

The ACE proposal

The present authors read the ACE proposal in 1975 and were enchanted by its style and originality, and its contrast with the conventional wisdom about the state of the art in 1945, i.e. John von Neumann's EDVAC report.[7] Subsequently, we analysed the original ACE design in some detail and wrote *The Other Turing Machine*, published in the Computer Journal in 1977.[8] Of course Turing had read the EDVAC report, but apart from his very different and lucid writing style (he couldn't match von Neumann's ponderous 'octroyed temporal sequence'), he provided a complete and quite detailed design for ACE. In this chapter, we use the word 'ACE' specifically to refer to the late 1945 design described in Turing's original report, unless otherwise noted.

The report was not detailed enough to act as an engineering blueprint, and it was not the design that was actually used, either for the Pilot ACE or for the Full Scale ACE. However, compared to the EDVAC report, much less was left as an exercise for the reader. Many of the ideas in the ACE design were common knowledge by the early 1960s, yet Turing was unknown as a computer designer.

In 48 typed pages, Turing described the concepts of a stored-program universal computer, a floating-point library, artificial intelligence, details such as a hardware bootstrap loader, and more, down to the level of detailed circuit diagrams and sample programs. Of course not all these ideas were his, but as an act of synthesis, the proposal is remarkable. How was a theoretician able to write such a report in 1945? To a large extent, that is a trick question: Turing *wasn't* just a theoretician by 1945. Even his most famous theoretical work, *On Computable Numbers*, published in 1936, was a thought experiment invoking a memory tape and logic circuits. He had made two serious attempts to build mathematical machines before World War II. During the war, he designed information processing machines and witnessed large scale data processing at Bletchley Park, and personally built electronic devices at Hanslope Park. Even in 1936, the basic components to build an electronic universal Turing machine existed: magnetic wire recording dated back to 1898, and the flip-flop (multivibrator) to 1919. Electronic AND gates (coincidence circuits) and binary counters came along in 1930. By 1945, Turing, well aware of Colossus, was ideally positioned to design a fully-fledged computer.

Credit should be given to the National Physical Laboratory for making this possible. The NPL Mathematics Division was approved in late 1944, supported by the Ministry of Supply, Commander Edward Travis of GCCS, Douglas Hartree, and Leslie J. Comrie, the New Zealander who founded Scientific Computing Service Ltd. in 1938. (The company was still registered at Companies House at the time of writing.) The Mathematics Division's job was to provide and coordinate national facilities for automated computation, including military applications. The first head of the Division was John R. Womersley, better known today for his later work on fluid dynamics. He'd worked with a differential

analyser and read *On Computable Numbers* before the war, and he was sent to the U.S.A. in early 1945 to learn about ENIAC and plans for EDVAC, after being appointed to NPL but just before taking up the job. Womersley understood the potential of universal automatic computers and was willing to foster unconventional ideas. He showed Turing the EDVAC report and hired him as a one-man section of his Division to study the design of an Automatic Computing Engine. By the end of 1945, Turing's *Proposed Electronic Calculator*, also known as *Proposals for Development in the Mathematics Division of an Automatic Computing Engine (ACE)*, was finished. It was presented to the NPL Executive Committee in March 1946, supported by Womersley and Hartree. The ACE project (but not the detailed design) was approved by the committee, chaired by NPL Director Sir Charles Darwin, grandson of *the* Charles Darwin.

Turing's proposal gave an outline of the principles of stored program computers, binary representation, and floating point arithmetic; a detailed architecture and instruction set; detailed logic diagrams; electronic circuits for various logic elements; and example programs. It also gave a budget estimate of £11,200 (twenty times Turing's annual salary at NPL), starting a long tradition of hopelessly underestimating costs in the IT industry. This version of ACE was to be a serial machine operating with a 1 MHz clock and 32-bit words. Turing was an early adopter, if not the originator, of the word 'word' in this usage. Completely unlike EDVAC, ACE was to have 32 registers in the central processing unit, known as TS1 through TS32, where 'TS' meant 'temporary storage', actually a short mercury delay line. The instruction set was register-to-register, whereas EDVAC was an accumulator machine. There were only eleven instructions, closely reflecting the hardware design, giving Turing a fair claim of being the first RISC (reduced instruction set computer) designer.[9]

The proposed applications ranged from numerical analysis, as expected by NPL, to counting butchers, solving jigsaws, and playing chess. The latter was certainly not expected, but it was a topic that Turing had probably discussed with Claude Shannon during the war. Turing foresaw relocatable code and something very like assembly language, called 'popular' form. He also foresaw a subroutine library, including the floating point routines, and presented examples, including BURY and UNBURY which implemented a stack for nested subroutine calls.

Formative ideas

In our 1975 paper, we identified the following set of formative technical ideas clearly found in the 1945 ACE proposal, only a few of which are also to be found in the EDVAC report. We will explain most of these concepts in more detail below.

Formative idea	Present in EDVAC report	Present in ACE proposal
Stored program	✓	✓
Binary implementation using standardised electronic logic elements	✓	✓
Complete notation for combinational and sequential circuits	✓	✓
Four architecture units (Memory, Control, Arithmetic, Input/Output)	✓	✓
Conditional branch instructions (although clumsy)	✓	✓
Address mapping (in a simple form)		✓
Instruction counter and instruction register		✓
Multiple fast registers, for data and addressing		✓
Microcode (in a simple form); hierarchical architecture		✓
Whole-card input/output operations (similar to direct memory access)		✓
Complete set of arithmetic, logical and rotate instructions		✓
Built in error detection and margin tests		✓
Floating point arithmetic		✓
Hardware bootstrap loader		✓
Subroutine stack		✓
Modular programming; subroutine library		✓
Documentation standards		✓
Programs treated as data; link editor; symbolic addresses		✓
Run time systems (input/output conversions; hints of macro expansion)		✓
Nonnumerical applications		✓
Artificial intelligence		✓

The stored program concept – that a computer could contain its own program in its own memory – derived ultimately from Turing's own paper *On Computable Numbers*, and Konrad Zuse also developed it in Germany, in the form of his *Plankalkül* language, without having read *On Computable Numbers*. The next four ideas were also to be found in the EDVAC report that Turing had recently read, suggesting that he was partly influenced by various American sources, perhaps via John von Neumann. All the other ideas at first appeared to be completely new, without published antecedents. One should not overlook that they preceded the influential Moore School lectures by several months (these lectures, held at the University of Pennsylvania from July 8th to August 31st 1946, exposed the American computer design ideas to the world's experts.)

Today, we know about Turing's discussions with Shannon during the war, about his experience at Bletchley Park, and his knowledge of Colossus and of the work of Jacquard, Babbage and Lovelace in the

previous century. Very likely, some of Turing's ideas were not brand new in the ACE report, but it remains quite startling to find them all in one place at such an early date. Other pioneers took a few more years to reach a similar point.[10]

Another interesting aspect of the proposal is the format of the logic diagrams. They use a notation derived via von Neumann from the famous American pioneers of computational neurophysiology, Warren McCulloch and Walter Pitts,[11] who in turn cited Turing's *On Computable Numbers*.

What happened to Turing's ideas?

Turing's ACE design was much modified before the Pilot ACE of 1950, the English Electric DEUCE and its other successors and derivatives were built, as described elsewhere in this book. Here, we will consider in turn the fate of the formative ideas apparently unique to Turing in 1945. All of them resurfaced over the following fifteen years or so; the question is how much of that was rediscovery, and how much was unacknowledged re-use. His descriptions were not in the open literature; the 1945 ACE report, mimeographed in a limited number of copies, was out of stock by 1948 and vanished from view until 1972. The Pilot ACE was well known in itself, but with little mention of Turing's contribution.

The Cambridge computer design team (see Chapter [Campbell-Kelly]), especially Maurice Wilkes, never admitted to being much influenced by Turing, although Stanley Gill, for one, worked alternately on Pilot ACE and the Cambridge EDSAC, but in both cases after the main design choices were fixed. In practice, the University of Manchester team mainly followed the EDVAC line. Yet all the ACE ideas showed up in later designs. Without going into technical detail and using modern jargon, it is hard to convey the breadth and depth of Turing's originality in the ACE design. The following paragraphs aim to give a flavour of his insights and why they were so prophetic in 1946.

Several of Turing's ideas were bound up with the notions of words in memory, their addresses, and registers. Computer memory is divided up into small pieces, known as 'words' and in the case of ACE consisting of 32 bits each, a word size still commonly used today. The position of a given word in memory is simply a number (zero for the first word, one for the next word, and so on) and this is called its 'address.' When the content of a word in memory must be processed by a computer instruction, it is often copied into a temporary storage device called a 'register' which is part of the computer's central processing unit. Like words in memory, ACE registers each held 32 bits.

A common technique in all modern computers is that memory addresses are 'mapped' for the convenience of the programmer; in this way the programmer can assume that programs and data are always in the same place, even if they are actually in different parts of the memory during different runs of the program. ACE presaged address mapping with a memory interleaving trick. Although this was not carried through to the Pilot ACE, address mapping later resurfaced in Manchester, most famously in the Atlas computer of 1962.

Turing made it clear that a special register was needed to contain the instruction currently being obeyed by the machine, and another one had to contain the address of the following instruction. These two registers, usually called the instruction register and the instruction counter, became universal in computer designs, perhaps because there's really no other way to do it, but Turing wrote it down first.

Turing planned multiple registers, whereas EDVAC had very few. Apart from DEUCE, the first production machine with multiple registers was the Ferranti Pegasus in 1956. Multiple fast registers, used to contain data or addresses, were widely adopted in the 1960s, for example in the IBM System/360. A particular kind of register that Turing foreshadowed in ACE is the 'index register' which is used as a pointer to a block of data. Such a register (known as a 'B-line') was implemented by 1949 at the University of Manchester.

Many modern computers use what is called a 'register-to-register' design, in which instructions use registers as the source and destination for arithmetic or logical operations. This was first proposed in the 1945 ACE design and was kept in Pilot ACE and DEUCE. In 1956, the Ferranti Pegasus also had this design. The idea famously reappeared in 1970, embedded in Digital Equipment Corporation's PDP-11/20, designed by Gordon Bell, who was a DEUCE user while he was a Fulbright scholar in Australia.

ACE had a rather hierarchical design, with a simple basic arrangement and simple basic instructions. However, some instructions could be modified to perform quite complex operations by setting extra bits on or off in their binary code. Today we recognise this technique as 'microcode', a concept also presaged in the MIT Whirlwind (1947), which reappeared most famously in Cambridge (EDSAC2, 1956).

Another important modern technique is 'direct memory access' whereby data are transferred to or from an external device automatically, without needing a complex sequence of machine instructions. Input or output of a whole punched card in one go was a feature of the ACE design. This clearly presaged direct memory access, which, despite Turing's precedence, is generally credited to the US National Bureau of Standards DYSEAC (1954), or to a technique called 'channels' first used in the IBM 709 (1957).

ACE had a complete set of arithmetic instructions, as was to be expected, but it also had logic instructions (AND, OR etc.), the latter probably suggested by cryptanalysis requirements. Logical instructions re-emerged in the Manchester Mark I (1949) and the IBM 701 (1952).

Turing recommended built-in error detection (involving extra hardware to detect errors) and running margin tests (operating the device under stress to see if it is close to failure.) He presumably knew about the need for these from Bletchley Park experience or directly from Tommy Flowers, the designer of Colossus, although Colossus itself did not include margin tests. Other builders of thermionic valve computers had to learn about this the hard way.

ACE was to have floating point software. Floating point arithmetic effectively means that a machine stores a certain number of significant digits (e.g., 3142) and a decimal multiplier (e.g., 0.001) separately instead of storing the single value 3.142. The multiplier is economically stored as an exponent (e.g., -3

for 10^{-3}). This allows a machine to store and process a very wide range of numbers. The technique had been known conceptually since 1914 and was found in electromechanical machines (Zuse Z1, 1938; Harvard Mark II, 1944; Stibitz Model V, 1945). Implementation of floating point arithmetic in electronic hardware, as opposed to using software packages, appeared in the Manchester MEG, the prototype of the Ferranti Mercury, in 1954, and in the IBM 704 in the same year.

When computers were first invented, they normally did nothing when first switched on, and a program had to be laboriously inserted by hand using switches on the front panel. Today, computers always start up an elementary program – they pull themselves up with their own bootstraps – and the small built-in program that does this is called a bootstrap loader. Amazingly, ACE was designed in 1945 to have a form of bootstrap loader; conventionally this idea is credited to the IBM 701 (1952).

Turing also clearly described what we would now call modular programming and a subroutine library. He recognised that large programs needed to be built up out of smaller ones (called subprograms or subroutines), and that many of the smaller ones could be kept and re-used later on, thus becoming a library. These ideas were reinvented at least twice, by Grace Hopper in the USA (1951-2) and by Maurice Wilkes, David Wheeler and Stanley Gill in Cambridge (1951). Software documentation standards (conventions to be followed by programmers to make their work easier to understand and re-use) are usually credited to Grace Hopper around 1952, but Turing recognised the need for them in 1945, three years before any stored program machine was built.

A very important concept in computer science is recursion, in which a subroutine calls itself. Technically, that is a bit tricky because the computer has to keep track of where it is. This is done by stacking data items on top of each other, and unstacking them in reverse. Turing described exactly that in the ACE report, calling the stacking and unstacking processes BURY and UNBURY, as mentioned earlier. A recursive stack appeared in the LISP programming language in 1958 and in the Algol language by 1960 (Mike Woodger, Turing's colleague at NPL, was an author of the original Algol report, but he has told us that the explicit notion of recursion in Algol was approved by Edsger Dijkstra; however, there were multiple contributors to this aspect of Algol.[12]) Recursion then appeared in hardware in the English Electric KDF9 (1960), the Burroughs B5000 (1961), and the Manchester University and Ferranti Atlas (1962).

Turing also anticipated several basic types of software development tool. In modern terms, the first of these was the equivalent of a link editor, a program that takes several pieces of machine code and combines them into a single program. The notion of a program that manipulates another program was truly spectacular in 1945. Secondly, he suggested programming using symbolic addresses: instead of writing numerical memory addresses, the programmer could use comprehensible names (such as 'total' or 'tally'). Thirdly, he suggested writing programs not in numerical machine code but in a readable ('popular') form of machine code very similar to a modern 'assembly language' (a symbolic language close to the machine instructions allowing the computer to be programmed at a detailed level). Along these lines, EDSAC had alphabetic instruction mnemonics by 1949, which EDSAC itself translated into machine code.

Turing described a simple run-time system for input-output conversions etc (that is, keeping frequently used subroutines permanently in the computer's memory, such as for converting numbers from decimal on punched cards to binary). Such systems soon became universal, but constituted another remarkable Turingesque foresight in 1945.

He discussed non-numerical applications (such as solving a jig-saw puzzle), presumably inspired by the use of electronic devices for cryptanalysis, which could not then be mentioned in writing under threat of the Official Secrets Act. He also discussed artificial intelligence in the ACE proposal; pioneering work for which he would eventually be credited, along with Claude Shannon, with whom he had discussed it during the war.

Finally, Turing showed remarkable sociological foresight, and we can do no better than to quote his own words. At Bletchley Park, he saw what was needed to manage a data-intensive operation. In 1945, he foresaw the profession of computer programmer:

'Instruction tables will have to be made up by mathematicians with computing experience and perhaps a certain puzzle-solving ability...

'This process of constructing instruction tables should be very fascinating. There need to be no real danger of it ever becoming a drudge, for any processes that are quite mechanical may be turned over to the machine itself.'[5]But he could see that developing programs was going to be a formidable task. Not much later, in 1947, he wrote:

'One of our difficulties will be the maintenance of an appropriate discipline, so that we do not lose track of what we are doing. We shall need a number of efficient librarian types to keep us in order... I have already mentioned that ACE will do the work of about 10,000 [human] computers.'[13]

He also foresaw what systems programmers would be like, with their mystique and gibberish:

'The masters [programmers] are liable to get replaced because as soon as any technique becomes at all stereotyped it becomes possible to devise a system of instruction tables which will enable the electronic computer to do it for itself. It may happen however that the masters will refuse to do this. They may be unwilling to let their jobs be stolen from them in this way. In that case they would surround the whole of their work with mystery and make excuses, couched in well chosen gibberish, whenever any dangerous suggestions were made.'[13]

Conclusion

How much credit should Turing get? It's clear that he was the first to work seriously on a general-purpose computer design in the UK, in late 1945, and that he showed remarkable foresight and inventiveness. He wrote down clearly and coherently in a few pages most of the important concepts needed to construct and use computers. The community of computer pioneers in the UK and the USA was relatively small until

well after 1954 when Turing died; we can assume that word of mouth had a significant effect, and that ideas were not always properly credited. We know that apart from NPL, Turing had some direct input to the Manchester computer developments but little influence on the EDSAC at Cambridge. In the end, it is impossible to tell, today, to what extent Turing's amazing foresights were passed on directly or indirectly to other pioneers, or to what extent they were simply rediscovered as their time came. Regardless of that, they all formed part of the computing *Zeitgeist* for years to come.

Acknowledgements

This chapter was significantly improved by multiple comments made by participants at a meeting of the BCS Computer Conservation Society in London in May 2012, and by comments from Jack Copeland . In addition to the specific references cited in the text, we are indebted to Andrew Hodges's *Alan Turing: The Enigma* and to numerous web sites.

Brian Carpenter was a visitor at the Computer Laboratory, University of Cambridge, during part of the preparation of this chapter.

References

1. J.H. Wilkinson, *The Pilot Ace*, in *Automatic Digital Computation*, Proceedings of a Symposium held at NPL March 25-29 1953, 1954. (Reprinted in Bell and Newell, *Computer Structures*.)

2. Rex Malik, *In the beginning - early days with ACE*, Data Systems, March 1969, 56-59 & 82.

3. B. Randell, *On Alan Turing and the Origins of Digital Computers*, Machine Intelligence 7, 1972.

4. B. Randell (ed.), *The Origins of Digital Computers - Selected Papers,* Springer Verlag, 1973.

5. Alan M. Turing, *Proposed Electronic Calculator* [also: *Proposals for Development in the Mathematics Division of an Automatic Computing Engine (ACE)*], NPL internal report E882, 1946. Reprinted as NPL Technical Report *Com Sci 57*, April 1972. Reprinted in Brian E. Carpenter and Robert W. Doran (editors), *A.M.Turing's ACE Report of 1946 and other papers*, Vol. 10 in the Charles Babbage Institute Reprint Series, MIT Press, 1986.

6. F.W. Winterbotham, *The Ultra Secret*, Weidenfeld & Nicolson, 1974.

7. J. von Neumann, *First draft of a report on the EDVAC (June 30, 1945)*, Contract No. W-670-ORD-4926. Moore School of Electrical Engineering, University of Pennsylvania, 1945 (extracts included in Randell[4]).

8. Brian E. Carpenter and Robert W. Doran , *The other Turing machine*, Computer J. 20, 269-279, 1977.

9. Robert W. Doran, *Computer Architecture and the ACE Computers*, in B. Jack Copeland, *Alan Turing's Electronic Brain: The Struggle to Build the ACE, the World's Fastest Computer*, Oxford University Press, 2012.

10. M.V. Wilkes, *The Best Way to Design An Automatic Calculating Machine*, Manchester University Computer Inaugural Conference, July 1951.

11. W.L. McCulloch and W. Pitts, *A Logical Calculus of the Ideas Immanent in Nervous Activity*, Bull. of Math. Biophysics, 5 , 115-133, 1943.

12. Edgar G. Daylight, *Dijkstra's Rallying Cry for Generalization: The Advent of the Recursive Procedure, late 1950s — early 1960s*, Computer J. 54(11) 1756-1772, 2011.

13. Alan M. Turing, Lecture to the London Mathematical Society, 20 February 1947. Reprinted in Brian E. Carpenter and Robert W. Doran (editors), *A.M.Turing's ACE Report of 1946 and other papers*, Vol. 10 in the Charles Babbage Institute Reprint Series, MIT Press, 1986.