The Stored-Program Universal Computer: Did Zuse Anticipate Turing and von Neumann?

B. Jack Copeland and Giovanni Sommaruga

Abstract This chapter sets out the early history of the stored-program concept. The several distinct 'onion skins' making up the concept emerged slowly over a ten-year period, giving rise to a number of different programming paradigms. A notation is developed for describing different aspects of the stored-program concept. Theoretical contributions by Turing, Zuse, Eckert, Mauchly, and von Neumann are analysed, followed by a comparative study of the first practical implementations of stored-programming, at the Aberdeen Ballistic Research Laboratory in the US and the University Manchester in the UK. Turing's concept of universality is also examined, and an assessment is provided of claims that various historic computers—including Babbage's Analytical Engine, Flowers' Colossus and Zuse's Z3—were universal. The chapter begins with a discussion of the work of the great German pioneer of computing, Konrad Zuse.

Keywords ACE • Alan turing • Analytical engine • Automatic sequence controlled calculator • BINAC • Charles Babbage • Colossus • EDVAC • ENIAC • F.C. Williams • Henschel AG • History of computing • History of hardware • History of software • Howard Aiken • J. Presper Eckert • John Mauchly • John V. Atanasoff • John von Neumann • Konrad Zuse • M.H.A. Newman • Manchester Baby • Plankalkül • Richard Clippinger • Stored–program concept • Thomas H. Flowers • Thomas Haigh • Tom Kilburn • Universal Turing machine • Zuse KG • Z1 • Z2 • Z3 • Z4 • Z5 • Z11 • Z22 • Z23

The original version of this chapter was revised. A correction to this chapter can be found at https://doi.org/10.1007/978-3-319-22156-4_14

B.J. Copeland (✉)
Department of Philosophy, University of Canterbury, New Zealand

The Turing Centre, ETH Zurich, Zurich, Switzerland

School of Historical and Philosophical Inquiry, University of Queensland, Australia
e-mail: jack.copeland@canterbury.ac.nz

G. Sommaruga
The Turing Centre, ETH Zurich, Zurich, Switzerland

Department of Humanities, Social and Political Sciences, ETH Zurich, Zurich, Switzerland
e-mail: sommarug@ethz.ch

G. Sommaruga, T. Strahm (eds.), *Turing's Revolution*,
DOI 10.1007/978-3-319-22156-4_3

1 Introduction

To Konrad Zuse belongs the honour of having built the first working program-controlled general-purpose digital computer. This machine, later called Z3, was functioning in 1941.[1] Zuse was also the first to hire out a computer on a commercial basis: as Sect. 2 explains, Zuse's Z4 was rented by the Swiss Federal Institute of Technology (ETH Zurich) for five years, and provided the first scientific computing service in Continental Europe.

Neither Z3 nor Z4 were electronic computers. These machines were splendid examples of pre-electronic relay-based computing hardware. Electromechanical relays were used by a number of other early pioneers of computing, for example Howard Aiken and George Stibitz in the United States, and Alan Turing at Bletchley Park in the United Kingdom. Bletchley's relay-based 'Bombe' was a parallel, special-purpose electromechanical computing machine for codebreaking (though some later-generation Bombes were electronic).[2] Aiken's giant relay-based Automatic Sequence Controlled Calculator, built by IBM in New York and subsequently installed at Harvard University (known variously as the IBM ASCC and the Harvard Mark I) had much in common with the earlier Z3.

Alan Turing, 1912–1954.
Credit: King's College Library, Cambridge

From an engineering point of view, the chief differences between the electromagnetic relay and electronic components such as vacuum tubes stem from the fact that, while the vacuum tube contains no moving parts save a beam of electrons, the relay contains mechanical components that move under the control of an

[1] Zuse, K. 'Some Remarks on the History of Computing in Germany', in Metropolis, N., Howlett, J., Rota, G. C. (eds) *A History of Computing in the Twentieth Century* (New York: Academic Press, 1980).

[2] For additional information about the Bombe, including Gordon Welchman's contributions, and the earlier Polish Bomba, see Copeland, B. J., Valentine, J., Caughey, C. 'Bombes', in Copeland, B. J., Bowen, J., Sprevak, M., Wilson, R., et al., *The Turing Guide* (Oxford University Press), forthcoming in 2016.

electromagnet and a spring, in order to make and break an electrical circuit. Vacuum tubes achieve very much faster digital switching rates than relays can manage. Tubes are also inherently more reliable, since relays are prone to mechanical wear (although tubes are more fragile). A small-scale electronic digital computer, containing approximately 300 tubes, was constructed in Iowa during 1939–42 by John V. Atanasoff, though Atanasoff's machine never functioned satisfactorily. The first large-scale electronic digital computer, Colossus, containing about 1600 tubes, was designed and built by British engineer Thomas H. Flowers during 1943, and was installed at Bletchley Park in January 1944, where it operated 24/7 from February of that year.[3]

Zuse's Z3 and Z4, like Aiken's ASCC, and other relay-based computers built just prior to, or just after, the revolutionary developments in digital electronics that made the first electronic computers possible, were a final luxuriant flowering of this soon-to-be-outdated computing technology (though for purposes other than computing, relays remained in widespread use for several more decades, e.g. in telephone exchanges and totalisators). Outmatched by the first-generation electronic machines, Zuse's computer in Zurich and Aiken's at Harvard nevertheless provided sterling service until well into the 1950s. While relay-based computers were slower than their electronic rivals, the technology still offered superhuman speed. Electromechanical computers carried out in minutes or hours calculations that would take human clerks weeks or months.

It was not just the absence of digital electronics that made Z3, Z4 and ASCC pre-modern rather than modern computers. None incorporated the *stored-program* concept, widely regarded as the sine qua non of the modern computer. Instructions were fed into the ASCC on punched tape. This programming method echoed Charles Babbage's nineteenth-century scheme for programming his Analytical Engine, where instructions were to be fed into the Engine on punched cards connected together with ribbon so as to form a continuous strip—a system that Babbage had based on the punched-card control of the Jacquard weaving loom. If the calculations that the ASCC was carrying out required the repetition of a block of instructions, this was clumsily achieved by feeding the same instructions repeatedly through the ASCC's tape-reader, either by punching multiple copies of the relevant block of instructions onto the tape or, if the calculation permitted it, by gluing the tape ends together to form a loop.[4] Zuse's Z3 and Z4 also had punched tape programming (Zuse preferred cine film to paper).[5] In a stored-program computer, on the other hand, the same instructions can be selected repeatedly and fed from

[3]Copeland, B. J. et al. *Colossus: The Secrets of Bletchley Park's Codebreaking Computers* (Oxford: Oxford University Press, 2006, 2010).

[4]Campbell, R. V. D. 'Aiken's First Machine: The IBM ASCC/Harvard Mark I', in Cohen, I. B., Welch, G. W. (eds) *Makin' Numbers: Howard Aiken and the Computer* (Cambridge, Mass.: MIT Press, 1999), pp. 50–51; Bloch, R. 'Programming Mark I', in Cohen and Welch, *Makin' Numbers*, p. 89.

[5]Zuse, 'Some Remarks on the History of Computing in Germany', p. 615; see also the photograph of a segment of Zuse's cine film tape in the *Konrad Zuse Internet Archive*, http://zuse.zib.de/

memory, providing an elegant solution to the problem of how to loop through a subroutine a number of times.

Konrad Zuse, 1910–1995.
Credit: ETH Zurich

Although Z3 and Z4 (like their predecessors Z1 and Z2) used punched-tape program control, there have always been rumours in the secondary literature that Zuse independently invented the stored-program concept, perhaps even prior to Turing's classic 1936 exposition of the concept and the extensive further development of it by both Turing and John von Neumann in 1945. Nicolas Jequier, for example, described Z3 as the first computer to have an '[i]nternally stored program'.[6] Jürgen Schmidhuber recently wrote in *Science*:

> By 1941, Zuse had physically built the first working universal digital machine, years ahead of anybody else. Thus, unlike Turing, he not only had a theoretical model but actual working hardware.[7]

In *Nature* Schmidhuber wrote:

> Zuse's 1936 patent application (Z23139/GMD Nr. 005/021) also described what is commonly called a 'von Neumann architecture' (re-invented in 1945), with program and data in modifiable storage.[8]

item/8OeSo6XPtIV2X44R (thanks to Matthias Röschner, vice-director of the Deutsches Museum Archiv, for information).

[6]Jequier, N. 'Computer Industry Gaps', *Science and Technology*, vol. 93 (Sept. 1969), pp. 30–35 (p. 34).

[7]Schmidhuber, J. 'Turing in Context', *Science*, vol. 336 (29 June 2012), pp. 1638–1639. sciencescape.org/paper/22745399.

[8]Schmidhuber, J. Comments on a review by John Gilbey (*Nature*, vol. 468, 9 December 2010, pp. 760–761), at www.nature.com/nature/journal/v468/n7325/abs/468760a.html.

Computer historians Brian Carpenter and Robert Doran are more cautious, saying only that

> The stored program concept—that a computer could contain its program in its own memory—derived ultimately from Turing's 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*.[9]

John von Neumann, 1903–1957. *Credit: Photographer unknown. From the Shelby White and Leon Levy Archives Center, Institute for Advanced Study, Princeton, NJ, USA*

Zuse described his sophisticated *Plankalkül* programming language (discussed in Sects. 2 and 7) as embodying 'the notation and results of the propositional and the predicate calculus' and he called it 'the first programming language'.[10]

Fascinated by these persistent rumours about Zuse, we investigated his unpublished writings from the period 1936–1945, in order to discover what he had actually said about universality and the stored-program concept. We studied especially his April 1936 patent application Z23139, his December 1936 patent application Z23624, entries from June 1938 in his workbook, his 1941 patent application Z391 (setting out the design of Z3[11]), and his 1945 manuscript 'Der Plankalkül'.[12]

[9]Carpenter, B. E., Doran, R. W. 'Turing's Zeitgeist', in Copeland, Bowen, Sprevak, Wilson et al., *The Turing Guide*.

[10]Zuse, 'Some Remarks on the History of Computing in Germany', p. 621; Zuse interviewed by Christopher Evans in 1975 ('The Pioneers of Computing: An Oral History of Computing', London: Science Museum; © Board of Trustees of the Science Museum).

[11]Konrad Zuse interviewed by Uta Merzbach in 1968 (Computer Oral History Collection, Archives Centre, National Museum of American History, Washington D.C.).

[12]Zuse, K. Patent Application Z23139, 'Verfahren zur selbsttätigen Durchführung von Rechnungen mit Hilfe von Rechenmaschinen' [Procedure for the automatic execution of calculations with the aid of calculating machines], 9 April 1936, Deutsches Museum Archiv, document reference NL 207/00659; Zuse, K. Patent Application Z23624, 'Rechenmaschine' [Calculating machine], 21 December 1936, Deutsches Museum Archiv, NL 207/0991; Zuse, K. Patent Application Z391, 'Rechenvorrichtung' [Calculating device], 1941, in the *Konrad Zuse Internet Archive*, http://zuse.

The fact that Zuse wrote in German has always presented an obstacle to the dissemination of his achievements among Anglophone historians. Indeed, much of Zuse's unpublished work is hand written, in a form of old-fashioned German shorthand. We present English translations of key passages from Zuse's documents (so far as we know, for the first time).

Our conclusion will be that the truth lies somewhere between Schmidhuber's statements and the more cautious statement by Carpenter and Doran. Their cautious statement is true, but there is more to be said. We cannot, however, endorse Schmidhuber's or Jequier's claims.

The structure of this chapter is as follows. After a short overview of Zuse's life and work in Sect. 2, based largely on Zuse's own accounts of events in tape-recorded interviews given in 1968 and 1975, Sect. 3 goes on to provide a comparative account of Turing's and von Neumann's contributions to the stored-program concept. Both men made fundamental and far-reaching contributions to the development of this keystone concept. In the voluminous secondary literature, however, von Neumann's contributions are generally exaggerated relative to Turing's, even to the extent that many accounts describe von Neumann as the inventor of the stored-program concept, failing altogether to mention Turing. Section 3 explains why this von Neumann-centric view is at odds with the historical record, and describes in detail the respective contributions made by Turing and von Neumann to the development of the concept during the key decade 1936–1946. Section 3 also discusses aspects of the work of the many others who contributed, in one way or another, to the development of this concept, including Eckert, Mauchly, Clippinger, Williams, and Kilburn.

Section 4 offers a fresh look at the stored-program concept itself. Six programming paradigms, that existed side by side during the decade 1936–1946, are distinguished: these are termed **P**1–**P**6. **P**3–**P**6 form four 'onion skins' of the stored-program concept and are of special interest. Equipped with this logical analysis, and also with the historical contextualization provided in Sect. 3, Sects. 5 and 6 turn to a detailed examination of unpublished work by Zuse, from the period 1936–1941. Finally, Sect. 7 summarizes our conclusions concerning the multifaceted origins of the stored-program concept.

zib.de/item/axy7eq6AntFRwuJv; Zuse, K. 'Der Plankalkül', manuscript, no date, in the *Konrad Zuse Internet Archive*, http://zuse.zib.de/file/1rUAfKDkirW803gT/a3/1c/07/c3-af26-4522-a2e5-6cdd96f40568/0/original/aa32656396be2df124647a815ee85a61.pdf; Zuse, K. 'Der Plankalkül', typescript, no date, Deutsches Museum Archiv, NL 207/0235.

2 Zuse: A Brief Biography

'It was a foregone conclusion for me, even in childhood, that I was to become an engineer', Zuse said.[13] Born in Berlin on 22 June 1910, he grew up in East Prussia and then Silesia (now lying mostly within the borders of Poland).[14] His father was a civil servant in the German Post Office. Young Konrad initially studied mechanical engineering at the Technical University in Berlin-Charlottenburg, but switched to architecture and then again to civil engineering.[15] Graduating from the Technical University in 1935, with a diploma in civil engineering, he obtained a job as a structural engineer at *Henschel-Flugzeugwerke AG* (Henschel Aircraft Company) in Schonefeld, near Berlin.

A determined young man with a clear vision of his future, Zuse left Henschel after about a year, in order to pursue his ambition of building an automatic digital binary calculating machine.[16] As a student, Zuse had become painfully aware that engineers must perform what he called 'big and awful calculations'.[17] 'That is really not right for a man', he said.[18] 'It's beneath a man. That should be accomplished with machines.' He started to rough out designs for a calculating machine in 1934, while still a student, and with his departure from Henschel set up a workshop in the living room of his parents' Berlin apartment.[19] There Zuse began constructing his first calculator, in 1936.[20] His 'parents at first were not very delighted', Zuse said drily.[21] Nevertheless they and his sister helped finance the project, and Kurt Pannke, the proprietor of a business manufacturing analogue calculating machines, helped out as well with small amounts of money.[22] Some of Zuse's student friends chipped in, too, and they also contributed manpower—half a dozen or more pairs of hands assisted with the construction of Zuse's first machine.[23]

[13]Zuse interviewed by Merzbach.

[14]Zuse interviewed by Merzbach.

[15]Zuse, K. *Der Computer – Mein Lebenswerk* [The computer—my life's work] (Berlin: Springer, 4th edn, 2007), p. 13.

[16]Zuse interviewed by Merzbach; Zuse, 'Some Remarks on the History of Computing in Germany', p. 612.

[17]Zuse, 'Some Remarks on the History of Computing in Germany', p. 611.

[18]Zuse interviewed by Merzbach.

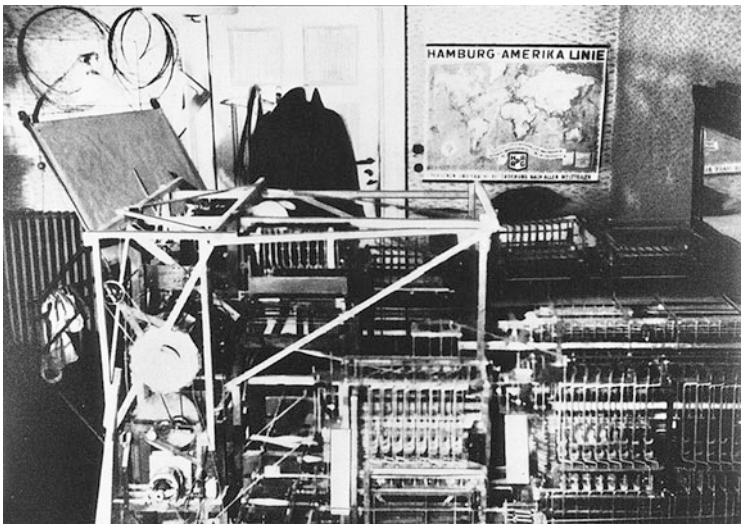
[19]Zuse interviewed by Merzbach; Zuse interviewed by Evans; Zuse, 'Some Remarks on the History of Computing in Germany', p. 612.

[20]Zuse, 'Some Remarks on the History of Computing in Germany', pp. 612–613.

[21]Zuse interviewed by Evans.

[22]Zuse interviewed by Merzbach; Zuse interviewed by Evans.

[23]Zuse interviewed by Merzbach.



Zuse's Z1 computer, in a Berlin apartment belonging to Zuse's parents. *Credit: ETH Zurich*

Later named Z1, his first calculator was completed in 1938, but never worked properly.[24] Z1 was purely mechanical.[25] Zuse said that its storage unit functioned successfully, and although the calculating unit could, Zuse recollected, multiply binary numbers and do floating-point arithmetic, it was prone to errors.[26] A significant problem was that the punched-tape program control was defective and Z1's various units never functioned together as a whole.[27]

Zuse believed initially that a mechanical calculator would be more compact than a relay-based machine.[28] Nevertheless, he set out detailed ideas concerning an electromechanical computer as early as 1936, as Sect. 6 describes. By 1938, the difficulties with Z1's calculating unit had convinced him of the need to follow an electromechanical path, and he built Z2, a transitional machine.[29] While the storage unit remained mechanical, the calculating unit was constructed from relays.[30] According to his son, Horst, Zuse used 800 telephone relays in the calculating unit.[31]

[24]Zuse interviewed by Merzbach. See also Rojas, R. 'Konrad Zuse's Legacy: The Architecture of the Z1 and Z3', *IEEE Annals of the History of Computing*, vol. 19 (1997), pp. 5–16.

[25]Zuse, 'Some Remarks on the History of Computing in Germany', pp. 613, 615.

[26]Zuse interviewed by Merzbach; Zuse interviewed by Evans.

[27]Zuse interviewed by Merzbach; Zuse interviewed by Evans.

[28]Zuse interviewed by Evans.

[29]Zuse interviewed by Merzbach; Zuse, 'Some Remarks on the History of Computing in Germany', p. 613.

[30]Zuse, 'Some Remarks on the History of Computing in Germany', p. 615.

[31]Zuse, H. 'Konrad Zuse Biographie', www.horst-zuse.homepage.t-pnline.de/kz-bio.html, p. 1.

Z2 was completed in 1939, the same year that Aiken and IBM produced the first detailed circuit drawings for the ASCC.[32] Z2 was binary, controlled by punched tape, and offered fixed-point arithmetic. Little more than an experiment in relay technology, the tiny computer had only 16 binary digits of storage.[33] It 'didn't work properly', Zuse said.[34] The problem was the relays. For economy's sake, he had bought used ones which he attempted to refurbish, but he set the contact pressure too low.[35]

When war came, in 1939, Zuse was drafted into the army.[36] But he saw no fighting, and in fact spent less than a year as a soldier.[37] Herbert Wagner, head of a department at Henschel that was developing flying bombs, urgently needed a statistician, and managed to arrange for Zuse to be released from the military.[38] Back in Berlin, Zuse was able to start work again on his calculating machine. At first this was only 'at night, Saturday afternoons and Sundays', he said, but then Henschel's aviation research and development group became interested.[39] Suddenly Zuse was given additional resources.

By 1941 he was able to set up a business of his own, while continuing to work part-time as a statistician. *K. Zuse Ingenieurbüro und Apparatebau* (K. Zuse Engineering and Machine-Making Firm) had a workshop in Berlin and ultimately a staff of about twenty.[40] The workshop had to be moved three or four times, as buildings succumbed to the bombing.[41] According to Zuse, his *Ingenieurbüro* was the only company in wartime Germany licensed to develop calculators.[42] Various armament factories financed his work, as well as the *Deutsche Versuchsanstalt für Luftfahrt* (German Institute for Aeronautical Research, DVL). According to a 2010 article in *Der Spiegel*, the DVL provided over 250,000 Reichsmarks for Zuse's calculator research (approximately 2 million US dollars in today's terms).[43] *Der Spiegel* wrote: 'The civil engineer was more deeply involved in the NS [National Socialist] arms industry than was believed hitherto. His calculating machines were

[32]Campbell, 'Aiken's First Machine', p. 34. Zuse wrote that Z2 was completed in 1939 (in 'Some Remarks on the History of Computing in Germany', p. 615); Rojas, however, gave 1940 as the completion date (Rojas, R. 'Zuse, Konrad', p. 3, zuse.zib.de/item/RuvnRJScXfvdt7BA).

[33]Zuse interviewed by Merzbach; Zuse interviewed by Evans.

[34]Zuse interviewed by Merzbach.

[35]Zuse interviewed by Merzbach; Zuse interviewed by Evans.

[36]Zuse interviewed by Merzbach.

[37]Zuse, *Der Computer – Mein Lebenswerk*, pp. 50, 57.

[38]Zuse interviewed by Merzbach; Zuse, 'Some Remarks on the History of Computing in Germany', p. 612.

[39]Zuse interviewed by Merzbach.

[40]Zuse interviewed by Merzbach.

[41]Zuse interviewed by Merzbach.

[42]Zuse, *Der Computer – Mein Lebenswerk*, p. 68.

[43]Schmundt, H. 'Rassenforschung am Rechner', *Der Spiegel*, Nr. 24 (2010) (14 June 2010), pp. 118–119.

considered important for the "final victory"'.[44] German historian Hartmut Petzold previously gave a lower figure, saying the DVL provided 50,000 Reichsmarks for Zuse's work.[45]

At Henschel, Zuse was involved with the Hs 293 rocket-propelled missile. He designed two special-purpose calculating machines, named S1 and S2, to assist with the manufacture of these weapons.[46] Air-launched and radio-controlled, the missiles were built on an assembly line (at a rate of one every ten minutes, Zuse estimated), and then during a final stage of production, a complicated system of sensors monitored the wings, while their geometry was fine-tuned.[47] Several hundred sensors were used to achieve the aerodynamic accuracy required for guiding the missile by radio. Initially, calculations based on the sensor data were done by hand, using a dozen Mercedes calculating machines and working day and night.[48] Each individual calculation 'took hours and hours', Zuse remembered.[49] He built S1 to automate these calculations. S1 was a relay-based binary calculator with a wired program, set by means of rotary switches.[50] He recollected completing the prototype, containing about 800 relays, in 1942.[51] Eventually there were three S1s, 'running day and night for several years', Zuse said.[52] Operators still had to enter the sensor data by hand, using a keyboard. The later S2 was designed to eliminate this data-entry stage, by connecting the sensors' outputs directly to the calculator, via a form of analog-to-digital converter.[53] Zuse explained that S2 was completed in 1944, but never became operational, because the factory (in Sudetenland) was dismantled just as the computer became ready.[54]

Zuse began building his fully electromechanical Z3 in 1940, again in his parents' living room, and completed it in 1941.[55] Z3's speed was on average one operation per second, Zuse recollected, and the memory unit had 64 storage cells.[56] The

[44]Schmundt, 'Rassenforschung am Rechner', p. 119.

[45]Petzold, H. *Moderne Rechenkünstler: Die Industrialisierung der Rechentechnik in Deutschland* (Munich: C. H. Beck, 1992), pp. 193–4, 201 ff.

[46]Zuse, *Der Computer – Mein Lebenswerk*, p. 54.

[47]Zuse interviewed by Merzbach; Zuse interviewed by Evans; Zuse, 'Some Remarks on the History of Computing in Germany', p. 619.

[48]Zuse interviewed by Merzbach; Zuse interviewed by Evans.

[49]Zuse interviewed by Merzbach.

[50]Zuse interviewed by Merzbach; Zuse, 'Some Remarks on the History of Computing in Germany', p. 615.

[51]Zuse interviewed by Evans.

[52]Zuse, 'Some Remarks on the History of Computing in Germany', p. 615; Zuse interviewed by Merzbach.

[53]Zuse interviewed by Merzbach.

[54]Zuse interviewed by Evans.

[55]Zuse, 'Some Remarks on the History of Computing in Germany', pp. 613, 615; Zuse interviewed by Merzbach; Zuse interviewed by Evans.

[56]Zuse interviewed by Evans.

time of large-scale electromechanical computing machines had come. By 1941 Turing's Bombes were turning Bletchley Park into a codebreaking factory, while at IBM progress was running slightly slower: Aiken's ASCC was partially working in 1942, and the computer solved its first practical problem on 1 January 1943.[57] Z3 contained some 2000 relays, 1400 in the storage unit and 600 in the calculating unit (which was capable of floating-point operations).[58] By comparison, Turing's Bombe contained 111 electromechanical 26-point rotary switches and approximately 150 relays; and the ASCC contained 3300 relays, as well as 2200 electromechanical 10-point rotary switches for storing decimal numbers.[59]

Zuse recollected that Z3 cost about 20,000 Reichsmarks to build, in an era when top-of-the-range calculating machines of the type used by engineers cost at most around 3000 Reichsmarks.[60] The DVL financed Z3 (and according to Zuse also provided about 10,000 Reichsmarks for his wing-geometry calculators).[61] Z3 was never used on a day-to-day basis, Zuse said, but did carry out a number of 'small test calculations': 'things that were interesting for aerodynamics, for airplane construction'.[62] In 1943, bombs destroyed Z3 as it stood in Zuse's workshop.[63] 'Z3 was total loss', he said.[64] All documentation was also destroyed in the bombing; only his 1941 patent application remained as a record of the machine.[65]

Z3, used only for 'program testing', was a step on the way to a larger and better computer, Z4, which Zuse had begun working on in 1942.[66] He returned to the idea of a mechanical store, but retained relays for the calculating unit.[67] Curiously, while building Z4 he heard from the German Secret Service about Aiken's ASCC; after the war, the two met when Aiken visited Zurich, and then again when Zuse visited New York in 1950 (at the invitation of Remington Rand) and made a detour to Boston to pay a call on Aiken.[68]

---

[57] Campbell, 'Aiken's First Machine', p. 55; Bashe, C. 'Constructing the IBM ASCC (Harvard Mark I)', in Cohen and Welch, *Makin' Numbers*, p. 74.

[58] Zuse, *Der Computer – Mein Lebenswerk*, p. 55; Zuse, 'Some Remarks on the History of Computing in Germany', p. 615.

[59] 'Operations of the 6812th Signal Security Detachment, ETOUSA', 1 October 1944 (US National Archives and Records Administration, College Park, Maryland, RG 457, Entry 9032, Historic Cryptographic Collection, Pre–World War I Through World War II, Box 970, Nr. 2943), pp. 82–84; Campbell, R. V. D., Strong, P. 'Specifications of Aiken's Four Machines', in Cohen and Welch, *Makin' Numbers*, p. 258.

[60] Zuse interviewed by Merzbach.

[61] Zuse interviewed by Merzbach.

[62] Zuse interviewed by Merzbach.

[63] Zuse interviewed by Merzbach.

[64] Zuse interviewed by Merzbach.

[65] Zuse interviewed by Merzbach.

[66] Zuse interviewed by Evans.

[67] Zuse interviewed by Merzbach.

[68] Zuse interviewed by Merzbach.

Z4 was almost complete when, as Germany teetered on the brink of collapse, Berlin became too perilous for Zuse to remain.[69] With Z4 strapped to an army truck he fled to Gottingen. Zuse put the machine back together again in the DVL's Gottingen laboratory, and according to Fritz Bauer it was in Gottingen that Z4 was put into operation for the first time.[70]

As Soviet troops drew ever closer to Gottingen, the Air Ministry ordered Zuse to move his computer into the vast subterranean tunnels housing the factories for V1 and V2 flying bombs. He visited the tunnels and found 'horrible conditions', where 'twenty thousand people who had been inmates of concentration camps' worked as slaves.[71] 'Anywhere at all, only not here', he said.[72] Then he was offered a place on a special convoy taking rocket scientist Wernher von Braun to Bavaria.[73] After only six weeks in Gottingen, Zuse loaded his computer onto a truck again, and travelling with about a dozen of his staff was transported south through Munich and Ettal, heading for the relative safety of the mountains by 'a very adventurous route', he said.[74]

Zuse's first hiding place was the tiny mountain village of Hinterstein, lying at the head of a remote valley in the Bavarian Alps, and only 5 km from the alpine border with Austria. He concealed the computer in a barn covered with hay.[75] There was 'no possibility to continue the work with hardware', he said, and with time on his hands Zuse decided to turn to developing his *Plankalkül*. He had begun thinking about a logical programming calculus during the war, producing sheaves of rough handwritten notes, partly in shorthand and partly in programming notation of his own devising.[76] It was in Hinterstein, during 1945, that he 'put together' his 'theoretical ideas ... and made a real calculus of it'.[77] Zuse produced a manuscript, titled 'Der Plankalkül', of some 250 handwritten folios divided into five chapters. This was subsequently typed, on paper headed 'K. Zuse Ingenieurbüro und Apparatebau, Berlin'. Some extracts from his 1945 manuscript are translated in Sect. 7.

In 1946, once American troops occupied the mountainous area, it seemed safe to shift the computer to the larger village of Hopferau, some 25 km away, where Zuse

[69]Zuse interviewed by Merzbach.

[70]Zuse interviewed by Merzbach; Bauer, F. L. 'Between Zuse and Rutishauser—The Early Development of Digital Computing in Central Europe', in Metropolis, Howlett and Rota, *A History of Computing in the Twentieth Century*, p. 505.

[71]Zuse interviewed by Merzbach; Zuse interviewed by Evans.

[72]Zuse interviewed by Merzbach.

[73]Zuse interviewed by Merzbach; Zuse interviewed by Evans.

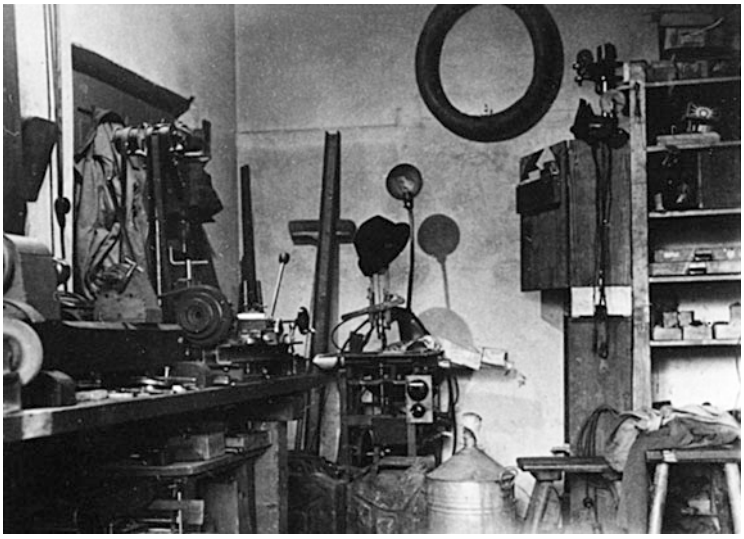
[74]Zuse interviewed by Merzbach; Zuse interviewed by Evans.

[75]Horst Zuse in conversation with Copeland.

[76]Zuse, K. Rough notes on the Plankalkül, no date, probably 1944 or the early months of 1945, Deutsches Museum Archiv, NL 207/0783 and NL 207/0887.

[77]Zuse interviewed by Evans.

remained until 1949.[78] An option with the German branch of Hollerith helped tide him over, he said—a couple in Hopferau had told an acquaintance at Hollerith of the 'strange inventor' in the village.[79] 1946 saw the start of his *Zuse-Ingenieurbüro* in Hopferau.[80] Another early contract was with Remington Rand Switzerland. He explained that the money enabled him to enlarge his company and to employ two or three men. His small factory produced a program-controlled relay calculator for Remington Rand. Named the M9, this was attached to punched card equipment. According to Zuse about thirty were delivered and Remington's clients used them in Switzerland, Germany and Italy.[81]



Zuse's computer workshop in the alpine village of Hopferau. *Credit: ETH Zurich*

It was while Zuse was in Hopferau that a 'gentleman from Zurich' visited him, starting the chain of events that led to Z4's delivery to ETH.[82] The visitor was Eduard Stiefel, founder of ETH's Institute for Applied Mathematics. Stiefel wanted a computer for his Institute and heard about Z4.[83] Both Stiefel and his assistant Heinz Rutishauser had recently visited the US and were familiar with Aiken's

[78]Zuse interviewed by Merzbach; Zuse, *Der Computer – Mein Lebenswerk*, p. 96.

[79]Zuse interviewed by Merzbach.

[80]Zuse, 'Konrad Zuse Biographie', p. 2.

[81]Zuse interviewed by Merzbach.

[82]Zuse interviewed by Merzbach.

[83]Bruderer, H. *Konrad Zuse und die Schweiz. Wer hat den Computer erfunden?* [Konrad Zuse and Switzerland. Who invented the computer?] (Munich: Oldenbourg, 2012), p. 5.

work.[84] They knew the worth of the large German electromechanical computer which the vagaries of war had delivered almost to their doorstep. ETH offered Zuse a rental agreement.

Shortly after Stiefel's visit, in 1949, Zuse moved to the small town of Neukirchen, about 50 km north of Dusseldorf, and there founded *Zuse Kommanditgesellschaft* (Zuse KG), increasing his staff to five.[85] Zuse KG would supply Europe with small, relatively cheap computers. Zuse's first task at Neukirchen was to restore and enlarge Z4 for ETH. He related that a second tape reader (or 'scanner') was attached, enabling numbers as well as programs to be fed in on punched tape, and circuitry was added for conditional branching (none of Z1–Z3 had been equipped with conditional branching).[86] He said the storage unit was enlarged from 16 cells to 64.[87] Rented by ETH from July 1950 until April 1955, Z4 was the first large-scale computer to go into regular operation in Continental Europe; and Stiefel's Institute for Applied Mathematics became a leading centre for scientific and industrial calculation. Despite assorted problems with the relays, Z4 was reliable enough to 'let it work through the night unattended', Zuse remembered.[88]

Now that Z4 had a home, Zuse moved on to Z5.[89] The German company Leitz, manufacturer of Leica cameras, needed a computer for optical calculations, and commissioned Z5. According to Petzold, the computer cost 200,000 Deutschmarks (about US$650,000 in today's terms) and was six times faster than Z4.[90] Next came Z11, a small relay-based wired-program computer that Zuse developed for the Géodésie company, again used mainly for optical calculations.[91] About 50 Z11s were built.[92] Applications included surveying and pension calculations.[93]

In 1957, Zuse moved his growing company to Bad Hersfeld, 50 km south of Kassel, and the following year embarked on Z22, his first vacuum tube computer.[94] Zuse had come round to tubes just as they were becoming outmoded for computer use—MIT's TX-0 transistorized computer first worked in 1956. Nevertheless,

[84]Zuse interviewed by Merzbach.

[85]Zuse interviewed by Merzbach.

[86]Zuse interviewed by Merzbach; Zuse, 'Some Remarks on the History of Computing in Germany', p. 616.

[87]Zuse interviewed by Merzbach.

[88]Zuse, 'Some Remarks on the History of Computing in Germany', p. 619. Urs Hochstrasser, one of the leading users of Z4 at ETH, gave an account of the problems associated with Z4's relays; see Bruderer, *Konrad Zuse und die Schweiz. Wer hat den Computer erfunden?*, pp. 19–27.

[89]Zuse interviewed by Merzbach.

[90]Petzold, *Moderne Rechenkünstler*, p. 216.

[91]Zuse interviewed by Merzbach.

[92]Zuse interviewed by Merzbach.

[93]Petzold, *Moderne Rechenkünstler*, pp. 216–217.

[94]Zuse, 'Konrad Zuse Biographie', p. 2.

Zuse's Z4 computer at ETH in Zurich. *Credit: ETH Zurich*

Zuse KG's Bad Hersfeld factory turned out more than 50 of the low-priced Z22 computers.[95] A transistorized version, Z23, went on the market in 1961.[96] Other electronic computers followed, the Z25 and Z64.[97] Oddly, Petzold says that with 'the step to electronic technology, Zuse KG also made the step to modifiable stored programs and thus to the von Neumann concept'.[98] As we explain, this concept is hardly von Neumann's, and in any case Zuse himself wrote of storing programs as early as 1936.

According to Horst Zuse, Zuse KG produced a total of 250 computers, with a value of more than 100 million Deutschmarks.[99] In 1964, however, Zuse and his wife relinquished ownership of the company.[100] By that time, despite Zuse KG's rapid growth, the company was overburdened by debt, and the Zuses put their shares on the market. German engineering company Brown Boveri purchased

[95] 'Zuse Computers', Computer History Museum, www.computerhistory.org/revolution/early-computer-companies/5/108.

[96] Zuse, *Der Computer – Mein Lebenswerk*, p. 125; Bruderer, *Konrad Zuse und die Schweiz. Wer hat den Computer erfunden?*, p. 63.

[97] Zuse, *Der Computer – Mein Lebenswerk*, pp. 126, 131–132.

[98] Petzold, *Moderne Rechenkünstler*, p. 217.

[99] Zuse, 'Konrad Zuse Biographie', p. 2.

[100] Zuse, *Der Computer – Mein Lebenswerk*, p. 137.

Zuse KG, with Zuse staying on as a consultant. Another sale in 1967 saw Siemens AG owning Brown Boveri.[101] Following further sales, a distant successor of Zuse's former company still exists today on Bad Hersfeld's Konrad-Zuse-Strasse, ElectronicNetwork GmbH, a contract electronics manufacturer.

As Sect. 1 mentioned, Zuse applied for a number of patents on his early computer designs (his most important patent applications, discussed in detail in Sects. 5 and 6, were in 1936 and 1941). However, the German authorities never granted Zuse a patent. During the war, he said, 'nothing much' happened regarding his patent, and then in the postwar years 'nothing whatever happened': his application 'lay around, gathering dust in a drawer of the patent office for years'.[102] When things finally did get moving, his efforts to patent his inventions came to the attention of IBM. Zuse explained that IBM worked through another company, Triumph Corporation, 'who lodged the protest'.[103] A 'serious legal battle' followed, Zuse said, and things dragged on until 1967, when the German federal patent court finally and irrevocably declined a patent.[104] The problem, according to the judge, was the patent's lack of *Erfindungshöhe*, literally 'invention height'. As Zuse explained matters, the judge stated that 'the requisite invention value has not been attained'.[105]

Konrad Zuse died in Huhnfeld on 18 December 1995.

3 Turing, von Neumann, and the Universal Electronic Computer

This section outlines the early history of the stored-program concept in the UK and the US, and compares and contrasts Turing's and John von Neumann's contributions to the development of the concept.[106] Although von Neumann is routinely said to be the originator of the stored-program concept, we find no evidence in favour of this common view. Turing described fundamental aspects of the concept in his 1936 article 'On Computable Numbers', which von Neumann had read before the war. When von Neumann arrived at the University of Pennsylvania's Moore

[101] Zuse, H. 'Historical Zuse-Computer Z23', 1999, www.computerhistory.org/projects/zuse_z23/index.shtml.

[102] Zuse interviewed by Merzbach.

[103] Zuse interviewed by Merzbach.

[104] Zuse interviewed by Merzbach; Zuse, *Der Computer – Mein Lebenswerk*, pp. 97–100. See also Petzold, H. *Die Ermittlung des 'Standes der Technik' und der 'Erfindungshöhe' beim Patentverfahren Z391. Dokumentation nach den Zuse-Papieren* [Establishing the 'state of the technological art' and 'inventiveness' in patent application Z391. Documentation from the Zuse papers] (Bonn: Selbstverlag, 1981).

[105] Zuse interviewed by Merzbach.

[106] Von Neumann was an alumnus of ETH Zurich, graduating as a chemical engineer in October 1926.

School of Electrical Engineering, in 1944, he recognized the potential of applying Turing's concept to practical computing. His own principal original contribution was devising practical coding schemes for stored programming. We also discuss the view, surprisingly popular, that what Turing termed a 'machine' in 1936 was a mathematical abstraction—essentially a set of quintuples—and that he made no connection between his abstract 'machine' and real computers.

At Cambridge, during the first three months of 1935, the young Alan Turing attended a course of advanced lectures on the Foundations of Mathematics, given by Max Newman, a Fellow of St John's College.[107] It was in these lectures that Turing heard of David Hilbert's *Entscheidungsproblem*, or *decision* problem. Yorick Smythies attended the lectures in 1934 and took detailed notes. Newman covered the Hilbert programme, propositional and predicate calculus, cardinals, theory of types and the axiom of reducibility, Peano arithmetic, Hilbert on proving consistency, and Gödel's first and second incompleteness theorems; and he mentioned that the *Entscheidungsproblem* had been settled only in the special case of monadic expressions.[108]

As stated by Turing, Hilbert's *Entscheidungsproblem* for the functional calculus (first-order predicate calculus) is this: *Is there a general (mechanical) process for determining whether a given formula A of the functional calculus* ***K*** *is provable*.[109] As everyone knows, Turing took on the *Entscheidungsproblem* and showed it to be unsolvable, along the way inventing the universal Turing machine. In a single, brilliant paper Turing ushered in both the modern computer and the mathematical study of unsolvability.

Newman's own contribution was not limited to bringing the *Entscheidungsproblem* to Turing's notice. In his lectures, Newman defined a *constructive* process as one that a *machine* can carry out. He explained in an interview:

> And this of course led [Turing] to the next challenge, what sort of machine, and this inspired him to try and say what one would mean by a perfectly general computing machine.[110]

Turing's 1936 paper 'On Computable Numbers' is the birthplace of the fundamental logical principles of the modern computer, and in particular the two closely

[107]Cambridge University Reporter, 18 April 1935, p. 826.

[108]Notes taken by Yorick Smythies during Newman's Foundations of Mathematics lectures in 1934 (St John's College Library, Cambridge).

[109]Turing, A. M. 'On Computable Numbers, with an Application to the Entscheidungsproblem', in Copeland, B. J. (ed.) *The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life* (Oxford: Oxford University Press, 2004), p. 84. 'On Computable Numbers' was published in 1936 but in the secondary literature the date of publication is often given as 1937, e.g. by Andrew Hodges in his biography *Alan Turing: The Enigma* (London: Vintage, 1992). The circumstances of publication of 'On Computable Numbers' are described on p. 5 of *The Essential Turing*.

[110]Newman interviewed by Christopher Evans ('The Pioneers of Computing: An Oral History of Computing', London: Science Museum); quoted in *The Essential Turing*, p. 206 (transcription by Copeland).

related logical ideas on which modern computing is based. We call these the 'twin pillars'. They are the concepts of (1): a *universal* computing machine, that operates by means of (2): a program of instructions *stored in the computer's memory* in the same form as data.[111] If different programs are placed on the memory-tape of the universal Turing machine, the machine will carry out different computations. Turing proved that the universal machine could obey any and every 'table of instructions'—any and every program expressed in the programming code introduced in his 1936 paper. His machine was universal in the sense that it could carry out *every* mechanical (or 'effective') procedure, if appropriately programmed.

The stored-program universal Turing machine led ultimately to today's archetypical electronic digital computer: the single slab of hardware, of fixed structure, that makes use of internally stored instructions in order to become a word-processor, or desk calculator, or chess opponent, or photo editor—or any other machine that we have the skill to create in the form of a program. Since these electronic machines necessarily have limited memories (unlike the universal Turing machine, with its indefinitely extendible tape), each is what Turing called 'a universal machine with a given storage capacity'.[112]

Turing's universal machine has changed the world. Yet nowadays, when nearly everyone owns a physical realization of one, his idea of a universal computer is apt to seem as obvious as the wheel and the arch. Nevertheless, in 1936, when engineers thought in terms of building different machines for different purposes, Turing's vision of a universal machine was revolutionary.

Zuse also briefly outlined a computing machine that would make use of programs stored in memory, in a few handwritten pages and a sequence of diagrams contained in a 1938 notebook, two years after Turing gave his extensive and detailed treatment of the idea. There is, however, no evidence that Zuse also formulated the concept of

[111]The first historians to insist that the stored-program concept originated in Turing's 1936 paper were (so far as is known) Brian Carpenter and Bob Doran, in a classic article that is one of New Zealand's earliest and greatest contributions to the history of computing: Carpenter, B. E., Doran, R. W. 'The Other Turing Machine', *The Computer Journal*, vol. 20 (1977), pp. 269–279. They said: 'It is reasonable to view the universal Turing machine as being programmed by the description of the machine it simulates; since this description is written on the memory tape of the universal machine, the latter is an abstract stored program computer' (p. 270). In the United States, Martin Davis has been advocating powerfully for the same claim since 1987; see Davis, M. D. 'Mathematical Logic and the Origin of Modern Computers', in Herken, R. (ed.) *The Universal Turing Machine: A Half-Century Survey* (Oxford: Oxford University Press, 1988); and Davis, M. D. *Engines of Logic: Mathematicians and the Origin of the Computer* (New York: Norton, 2000). However, the proposition that the stored-program concept originated in 'On Computable Numbers' is far from being a historians' reconstruction: as the present chapter explains, this was common knowledge among Turing's post-war colleagues at the National Physical Laboratory, and it was obvious to members of Max Newman's wartime group at Bletchley Park that digital electronics could be used to implement practical forms of Turing's universal machine of 1936.

[112]Turing, A. M. 'Intelligent Machinery', in *The Essential Turing*, p. 422.

a universal machine, as distinct from a general-purpose computer, as we explain in Sect. 5.

As early as the 1830s Charles Babbage—one of the first to appreciate the vast potential of computing machinery—had also envisioned a general-purpose computer, his Analytical Engine. Babbage said that the 'conditions which enable a finite machine to make calculations of unlimited extent are fulfilled in the Analytical Engine'.[113] Zuse first learned of Babbage's work (probably in 1938) from the US Patent Office which, on the basis of a comparison with Babbage's plans, declined Zuse's application for a patent.[114] In 1950, Turing stated that Babbage's Analytical Engine was universal, and the same has recently been proved of a modified form of Zuse's Z3 by Raul Rojas (see Sect. 5).[115] Such judgments are possible only from the vantage point of 'On Computable Numbers'—Babbage himself did not have the concept of a universal machine, a machine that is able to carry out all effective procedures.[116]

Nor did Babbage have the stored-program concept. As Sect. 1 mentioned, the Analytical Engine's program resided on punched cards and, as each card entered the Engine, the instruction marked on that card would be obeyed. True, the cards, strung together with ribbon, bore some resemblance to the universal Turing machine's tape; but in the Analytical Engine there was a distinction of kind between program and data, and this is dispensed with in the universal Turing machine. As Turing's friend and colleague the mathematician Robin Gandy put the point, in the universal Turing machine 'the mechanisms used in reading a program are of the same kind as those used in executing it'.[117]

Newman reported that Turing was interested 'right from the start' in building a universal computing machine.[118] However, Turing knew of no suitable technology.

[113]Babbage, C. *Passages from the Life of a Philosopher*, vol. 11 of Campbell-Kelly, M. (ed.) *The Works of Charles Babbage* (London: William Pickering, 1989), p. 97.

[114]Zuse interviewed by Merzbach.

[115]Rojas, R. 'How to Make Zuse's Z3 a Universal Computer', *IEEE Annals of the History of Computing*, vol. 20 (1998), pp. 51–54.

[116]Historian Thomas Haigh, in his impassioned outburst 'Actually, Turing Did Not Invent the Computer' (*Communications of the ACM*, vol. 57 (2014), pp. 36–41), confuses the logical distinction between, on the one hand, the universal machine concept and, on the other, the concept 'of a single machine that could do different jobs when fed different instructions'. Talking about this second concept, Haigh objects that it was not Turing but Babbage who 'had that idea long before' (pp. 40–41). Babbage did indeed have that idea; the point, however, is that although Babbage had the concept of a general-purpose computing machine, the universal machine concept originated with Turing. (All this is explained in Copeland's 'Turing and Babbage' in *The Essential Turing*, pp. 27–30.)

[117]Gandy, R. 'The Confluence of Ideas in 1936', in Herken, R. (ed.) *The Universal Turing Machine: A Half-Century Survey* (Oxford: Oxford University Press, 1998), p. 90. Emphasis added.

[118]Newman interviewed by Evans; Newman, M. H. A. 'Dr. A. M. Turing', *The Times*, 16 June 1954, p. 10.

Relays, he thought, would not be adequate.[119] So, for the next few years, Turing's revolutionary ideas existed only on paper. A crucial moment came in 1944, when he set eyes on Flowers' racks of high-speed electronic code-cracking equipment, at Bletchley Park. Colossus was neither stored-program nor general-purpose, but it was clear to Turing (and to Newman) that the technology Flowers was pioneering, large-scale digital electronics, was the way to build a miraculously fast universal computer, the task to which Turing turned in 1945. Meanwhile, Zuse had pressed ahead with relays and had built a general-purpose computer, but neither knew of the work of the other at that time.

Did Zuse and Turing meet postwar? Probably not. Zuse said (in 1992) that he had no knowledge of Turing's 'On Computable Numbers' until 1948.[120] This recollection of Zuse's, if correct, makes it unlikely that he and Turing met the previous year at a colloquium in Gottingen, as German computer pioneer Heinz Billing reported in his memoirs.[121] A more likely connection is Turing's colleague from the National Physical Laboratory (NPL) Donald Davies, who interrogated Zuse in England.[122] Zuse was invited to London in 1948 and placed in a large house in Hampstead, where a number of British computer experts arrived to question him.[123] Zuse remembered it as a 'very nice trip'.[124] Quite likely Davies—who, along with Turing's other colleagues in NPL's ACE section, saw 'On Computable Numbers' as containing the 'key idea on which every stored-program machine was based'—would have mentioned Turing's paper to Zuse.[125] Davies recollected that the interview did not go particularly well: Zuse eventually 'got pretty cross', and things 'degenerated into a glowering match'. Zuse was 'quite convinced', Davies said, that he could make a smallish relay machine 'which would be the equal of any of the electronic calculators we were developing'.

Although Turing completed his design for an electronic stored-program computer in 1945, another four years elapsed before the first universal Turing machine in electronic hardware ran the first stored program, on Monday 21 June 1948. It was the first day of the modern computer age. Based on Turing's ideas, and almost

[119]Robin Gandy interviewed by Copeland, October 1995.

[120]Zuse in conversation with Brian Carpenter at CERN on 17 June 1992; Copeland is grateful to Carpenter for sending him some brief notes on the conversation that Carpenter made at the time. See also Carpenter, B. E. *Network Geeks: How They Built the Internet* (London: Springer, 2013), p. 22.

[121]Jänike, J., Genser, F. (eds) *Ein Leben zwischen Forschung und Praxis—Heinz Billing* [A Life Between Research and Practice—Heinz Billing] (Dusseldorf: Selbstverlag Friedrich Genser, 1997), p. 84; Bruderer, *Konrad Zuse und die Schweiz. Wer hat den Computer erfunden?*, pp. 64–66.

[122]Davies interviewed by Christopher Evans in 1975 ('The Pioneers of Computing: An Oral History of Computing', London: Science Museum; © Board of Trustees of the Science Museum).

[123]Zuse, *Der Computer – Mein Lebenswerk*, p. 101; Zuse interviewed by Evans; Davies interviewed by Evans.

[124]Zuse interviewed by Evans.

[125]Davies interviewed by Evans.

big enough to fill a room, this distant ancestor of our mainframes, laptops, tablets and phones was called 'Baby'.[126] Baby was built by radar engineers F. C. Williams and Tom Kilburn, in Newman's Computing Machine Laboratory at the University of Manchester, in the north of England.[127]

However, historians of the computer have often found Turing's contributions hard to place, and many histories of computing written during the six decades since his death sadly do not so much as mention him. Even today there is still no real consensus on Turing's place in computing history. In 2013, an opinion piece by the editor of the Association for Computing Machinery's flagship journal objected to the claim that Turing invented the stored-program concept. The article's author, Moshe Vardi, dismissed the claim as 'simply ahistorical'.[128] Vardi emphasized that it was not Turing but the Hungarian-American mathematician John von Neumann who, in 1945, 'offered the first explicit exposition of the stored-program computer'. This is true, but the point does not support Vardi's charge of historical inaccuracy. Although von Neumann did write the first paper explaining how to convert Turing's ideas into electronic form, the fundamental conception of the stored-program universal computer was nevertheless Turing's.

Von Neumann was close to the centre of the American effort to build an electronic stored-program universal computer. He had read Turing's 'On Computable Numbers' before the war,[129] and when he became acquainted with the U.S. Army's ENIAC project in 1944, he discovered that the stored-program concept could be applied to electronic computation.[130] ENIAC was designed by Presper Eckert and John Mauchly at the Moore School of Electrical Engineering (part of the University of Pennsylvania), in order to calculate the complicated tables needed by gunners to aim artillery, and the computer first ran in 1945. Like Colossus before it, ENIAC was programmed by means of re-routing cables and setting switches, a process that could take as long as three weeks.[131] Viewed from the modern stored-program world, this conception of programming seems unbearably primitive. In the taxonomy of programming paradigms developed in Sect. 4, this method of programming is **P**1, the most rudimentary level in the taxonomy.

[126]Tootill, G. C. 'Digital Computer—Notes on Design & Operation', 1948–9 (National Archive for the History of Computing, University of Manchester).

[127]For more about Baby, see Copeland, B. J. 'The Manchester Computer: A Revised History', *IEEE Annals of the History of Computing*, vol. 33 (2011), pp. 4–37; Copeland, B. J. *Turing, Pioneer of the Information Age* (Oxford: Oxford University Press, 2012, 2015), ch. 9.

[128]Vardi, M. Y. 'Who Begat Computing?', *Communications of the ACM*, vol. 56 (Jan. 2013), p. 5.

[129]Stanislaw Ulam interviewed by Christopher Evans in 1976 ('The Pioneers of Computing: an Oral History of Computing', Science Museum: London).

[130]Goldstine, H. *The Computer from Pascal to von Neumann* (Princeton: Princeton University Press, 1972), p. 182.

[131]Campbell-Kelly, M. 'The ACE and the Shaping of British Computing', in Copeland, B. J. et al. *Alan Turing's Electronic Brain: The Struggle to Build the ACE, the World's Fastest Computer* (Oxford: Oxford University Press, 2012; a revised and retitled paperback edition of the 2005 hardback *Alan Turing's Automatic Computing Engine*), p. 151.

Conscious of the need for a better method of programming, the brilliant engineer Eckert had the idea of storing instructions in the form of numbers as early as 1944, inventing a high-speed recirculating memory.[132] This was based on apparatus he had previously used for echo cancellation in radar, the mercury delay line. Instructions and data could be stored uniformly in the mercury-filled tube, in the form of pulses—binary digits—that were 'remembered' for as long as was necessary. This provided the means to engineer the stored-program concept, and mercury delay lines were widely employed as the memory medium of early computers—although in fact the first functioning electronic stored-program computer used not delay lines but an alternative form of memory, the Williams tube. Based on the cathode ray tube, the Williams tube was invented by Williams and further developed by Kilburn.

Along with Zuse, Eckert has a strong claim to be regarded as a co-originator of the stored-program paradigm that in Sect. 4 is denoted '**P**3'. Eckert said that the stored-program concept was his 'best computer idea'—although, of course, he arrived at the idea approximately eight years after the publication of Turing's 'On Computable Numbers'.[133] Endorsing Eckert's claim, Mauchly commented that they were discussing 'storing programs in the same storage used for computer data' several months before von Neumann first visited their ENIAC group.[134] Art Burks, a leading member of the ENIAC group and later one of von Neumann's principal collaborators at the Princeton computer project, also explained that—long before von Neumann first visited them—Eckert and Mauchly were 'saying that they would build a mercury memory large enough to store the program for a problem as well as the arithmetic data'.[135]

Maurice Wilkes, who visited the Moore School group in 1946 and who went on to build the Cambridge EDSAC delay-line computer (see the timeline in Fig. 3), gave this first hand account of the roles of Eckert, Mauchly and von Neumann:

> Eckert and Mauchly appreciated that the main problem was one of storage, and they proposed ... ultrasonic delay lines. Instructions and numbers would be mixed in the same memory. ... Von Neumann ... appreciated at once ... the potentialities implicit in the stored program principle. That von Neumann should bring his great prestige and influence to bear was important, since the new ideas were too revolutionary for some, and powerful voices were being raised to say that ... to mix instructions and numbers in the same memory was going against nature.[136]

Burks presented a similar picture:

> The second revolution [at the Moore School] was the stored-program computer. ... There were two main steps. Pres [Eckert] and John [Mauchly] invented the circulating mercury

[132]Eckert, J. P. 'The ENIAC', in Metropolis, Howlett and Rota, *A History of Computing in the Twentieth Century*, p. 531.

[133]Eckert, 'The ENIAC', p. 531.

[134]Mauchly, J., commenting in Eckert, 'The ENIAC', pp. 531–532.

[135]Letter from Burks to Copeland, 16 August 2003.

[136]Wilkes, M. V. *1967 ACM Turing Lecture*: 'Computers Then and Now', *Journal of the Association for Computing Machinery*, vol. 15 (1968), pp. 1–7 (p. 2).

> delay line store, with enough capacity to store program information as well as data. Von Neumann created the first modern order code and worked out the logical design of an electronic computer to execute it.[137]

Von Neumann, then, did not originate the stored-program concept, but contributed significantly to its development, both by championing it in the face of conservative criticism, and, even more importantly, by designing an appropriate instruction code for stored programming. As Tom Kilburn said, 'You can't start building until you have got an instruction code'.[138]

During the winter of 1944 and spring of 1945, von Neumann, Eckert and Mauchly held a series of weekly meetings, working out the details of how to design a stored-program electronic computer.[139] Their proposed computer was called the EDVAC. In effect they designed a universal Turing machine in hardware, with instructions stored in the form of numbers, and common processes reading the data and reading and executing the instructions. While there are no diary entries to prove the point beyond any shadow of doubt, nor statements in surviving letters written by von Neumann at this precise time, his appreciation of the great potentialities inherent in the stored-program concept could hardly fail to have been influenced by his knowledge of Turing's 'On Computable Numbers', nor by his intimate knowledge of Kurt Gödel's 1931 demonstration that logical and arithmetical sentences can be expressed as numbers.[140] Von Neumann went on to inform electronic engineers at large about the stored-program concept.

In his 1945 document titled 'First Draft of a Report on the EDVAC', von Neumann set out, in rather general terms, the design of an electronic stored-program computer.[141] However, shortly after this appeared in mid 1945, his collaboration with Eckert and Mauchly came to an abrupt end, with the result that the ill-fated EDVAC was not completed until 1952.[142] Trouble arose because von Neumann's colleague Herman Goldstine had circulated a draft of the report before Eckert's and

[137] Burks, A. W. 'From ENIAC to the Stored-Program Computer: Two Revolutions in Computers', in Metropolis, Howlett, and Rota, *A History of Computing in the Twentieth Century*, p. 312.

[138] Kilburn interviewed by Copeland, July 1997.

[139] Von Neumann, J., Deposition before a public notary, New Jersey, 8 May 1947; Warren, S. R. 'Notes on the Preparation of "First Draft of a Report on the EDVAC" by John von Neumann', 2 April 1947. Copeland is grateful to Harry Huskey for supplying him with copies of these documents.

[140] Gödel, K, 'Uber formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I.' [On formally undecidable propositions of Principia Mathematica and related systems I], *Monatshefte für Mathematik und Physik*, vol. 38 (1931), pp. 173–198.

[141] Von Neumann, J. 'First Draft of a Report on the EDVAC', Moore School of Electrical Engineering, University of Pennsylvania, 1945; reprinted in full in Stern, N. *From ENIAC to UNIVAC: An Appraisal of the Eckert-Mauchly Computers* (Bedford, Mass.: Digital Press, 1981).

[142] Huskey, H. D. 'The Development of Automatic Computing', in *Proceedings of the First USA-JAPAN Computer Conference*, Tokyo, 1972, p. 702.

Mauchly's names were added to the title page.[143] Bearing von Neumann's name alone, the report was soon widely read. Eckert and Mauchly were furious but von Neumann was unrepentant.

'My personal opinion', von Neumann said defiantly in 1947, 'was at all times, and is now, that this [the distribution of the report] was perfectly proper and in the best interests of the United States'.[144] Widespread dissemination of the report had, he said, furthered 'the development of the art of building high speed computers'. Perhaps he was hinting that Eckert and Mauchly would have opposed widespread distribution of the report. It would be perfectly understandable if they had, since the report's entering the public domain effectively prevented them from patenting their ideas. Eckert later wrote bitterly of 'von Neumann's way of taking credit for the work of others'.[145] Jean Jennings, one of ENIAC's programmers and a member of the ENIAC group from early 1945, noted that von Neumann 'ever afterward accepted credit—falsely—for the work of the Moore School group. ... [He] never made an effort to dispel the general acclaim in the years that followed'.[146]

After a dispute with the University of Pennsylvania about intellectual property rights, Eckert and Mauchly formed their own Electronic Control Company, and began work on their EDVAC-like BINAC. Meanwhile, von Neumann drew together a group of engineers at the Institute for Advanced Study in Princeton. He primed them by giving them Turing's 'On Computable Numbers' to read.[147] Julian Bigelow, von Neumann's chief engineer, was well aware of the influence that 'On Computable Numbers' had had on von Neumann. The reason that von Neumann was the 'person who really ... pushed the whole field ahead', Bigelow explained, was because 'he understood a good deal of the mathematical logic which was implied by the [stored program] idea, due to the work of A. M. Turing ... in 1936'.[148] 'Turing's machine does not sound much like a modern computer today, but nevertheless it was', Bigelow said—'It was the germinal idea'. The physical embodiment of Turing's universal computing machine that von Neumann's engineers built at Princeton began working in 1951. Known simply as the 'Princeton

[143] See e.g. Stern, N. 'John von Neumann's Influence on Electronic Digital Computing, 1944–1946', *Annals of the History of Computing*, vol. 2 (1980), pp. 349–362.

[144] Von Neumann, Deposition, 8 May 1947.

[145] Eckert, 'The ENIAC', p. 534.

[146] Jennings Bartik, J. *Pioneer Programmer: Jean Jennings Bartik and the Computer that Changed the World* (Kirksville, Missouri: Truman State University Press: 2013), pp. 16, 18.

[147] Letter from Julian Bigelow to Copeland, 12 April 2002; see also Aspray, W. *John von Neumann and the Origins of Modern Computing* (Cambridge, Mass.: MIT Press, 1990), p. 178.

[148] Bigelow in a tape-recorded interview made in 1971 by the Smithsonian Institution and released in 2002; Copeland is grateful to Bigelow for previously sending him a transcript of excerpts from the interview.

computer', it was not the first of the new stored-program electronic computers, but it was the most influential.[149]

Although Turing is not mentioned explicitly in von Neumann's papers developing the design for the Princeton computer, von Neumann's collaborator Burks told Copeland that his, von Neumann's, and Goldstine's key 1946 design paper did contain a reference to Turing's 1936 work.[150] Von Neumann and his co-authors emphasized that 'formal-logical' work—by which they meant in particular Turing's 1936 investigation—had shown 'in abstracto' that stored programs can 'control and cause the execution' of any sequence (no matter how complex) of mechanical operations that is 'conceivable by the problem planner'.[151]

Meanwhile, in 1945, Turing joined London's National Physical Laboratory, to design an electronic universal stored-program computer. John Womersley, head of NPL's newly formed Mathematics Division, was responsible for recruiting him. Womersley had read 'On Computable Numbers' shortly after it was published, and at the time had considered building a relay-based version of Turing's universal computing machine. As early as 1944 Womersley was advocating the potential of electronic computing.[152] He named NPL's projected electronic computer the Automatic Computing Engine, or ACE—a deliberate echo of Babbage.

Turing studied 'First Draft of a Report on the EDVAC', but favoured a radically different type of design. He sacrificed everything to speed, launching a 1940s version of what is today called RISC (Reduced Instruction Set Computing).[153] In order to maximise the speed of the machine, Turing opted for a decentralised architecture, whereas von Neumann described a centralised design that foreshadowed the modern central processing unit (cpu).[154] Turing associated different arithmetical and logical functions with different delay lines in the ACE's Eckert-type mercury memory, rather than following von Neumann's model of a single central unit in

[149]The Princeton computer is described in Bigelow, J. 'Computer Development at the Institute for Advanced Study', in Metropolis, Howlett, Rota, *A History of Computing in the Twentieth Century*.

[150]Letter from Arthur Burks to Copeland, 22 April 1998.

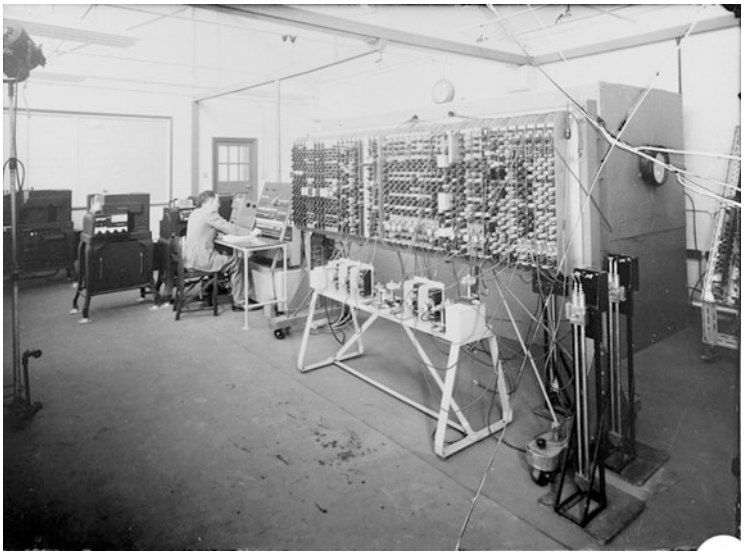
[151]Burks, A. W., Goldstine, H. H., von Neumann, J. 'Preliminary Discussion of the Logical Design of an Electronic Computing Instrument', Institute for Advanced Study, 28 June 1946, in vol. 5 of Taub, A. H. ed. *Collected Works of John von Neumann* (Oxford: Pergamon Press, 1961), section 3.1 (p. 37).

[152]See Copeland, B. J. 'The Origins and Development of the ACE Project', in Copeland et al., *Alan Turing's Electronic Brain*.

[153]Doran, R. W. 'Computer Architecture and the ACE Computers', in Copeland et al., *Alan Turing's Electronic Brain*.

[154]The terms 'decentralised' and its opposite 'centralised' are due to Jack Good, who used them in a letter to Newman about computer architecture on 8 August 1948; the letter is in Good, I. J. 'Early Notes on Electronic Computers' (unpublished, compiled in 1972 and 1976; a copy is in the National Archive for the History of Computing, University of Manchester, MUC/Series 2/a4), pp. 63–4.

which all the arithmetical and logical operations take place.[155] Turing was (as his colleague James Wilkinson observed[156]) 'obsessed' with making the computations run as fast as possible, and once a pilot version of the ACE was operational, it could multiply at roughly 20 times the speed of its closest competitor.[157]



The pilot model of Turing's Automatic Computing Engine, in the Mathematics Division of London's National Physical Laboratory. *Credit: National Physical Laboratory © Crown copyright*

Turing described his design in a report titled 'Proposed Electronic Calculator', completed by the end of 1945.[158] The proposals in the report were in fact much more concrete than those contained in von Neumann's rather abstract treatment in the 'First Draft'. The 'First Draft' hardly mentioned electronics, and Harry Huskey, the engineer whose job it was to draw up the first hardware designs for the EDVAC, said he found the 'First Draft' to be of 'no help'.[159] Turing, on the other hand,

[155]For additional detail concerning the differences between Turing's decentralized architecture and the centralized architecture favoured by von Neumann, see Copeland et al., *Alan Turing's Electronic Brain*; and Copeland, 'The Manchester Computer: A Revised History'.

[156]Wilkinson interviewed by Christopher Evans in 1976 ('The Pioneers of Computing: An Oral History of Computing', London: Science Museum).

[157]See the table by Martin Campbell-Kelly on p. 161 of Copeland et al., *Alan Turing's Electronic Brain*.

[158]Turing, A. M. 'Proposed Electronic Calculator', ch. 20 of Copeland et al., *Alan Turing's Electronic Brain*.

[159]Letter from Huskey to Copeland, 4 February 2002.

gave detailed specifications of the various hardware units, and even included sample programs in machine code.

Turing was content to borrow some of the elementary design ideas in von Neumann's report (and also the notation, due originally to McCulloch and Pitts, that von Neumann used to represent logic gates—a notation that Turing considerably extended in 'Proposed Electronic Calculator'). One example of a borrowing is Turing's diagram of an adder, essentially the same as von Neumann's diagram.[160] This borrowing of relatively pedestrian details is probably what Turing was referring to when he told a newspaper reporter in 1946 that he gave 'credit for the donkey work on the A.C.E. to Americans'.[161] Yet, the similarities between Turing's design and the von Neumann-Eckert-Mauchly proposals are relatively minor in comparison to the striking differences.

In their 1945 documents 'Proposed Electronic Calculator' and 'First Draft of a Report on the EDVAC', Turing and von Neumann both considerably fleshed out the stored-program concept, turning it from the bare-bones logical idea of Turing's 1936 paper into a fully articulated, electronically implementable design concept. The 1945 stored-program concept included:

- dividing stored information into 'words' (the term is used by Eckert and Mauchly in a September 1945 progress report on the EDVAC[162])
- using binary numbers as addresses of sources and destinations in memory
- building arbitrarily complex stored programs from a small stock of primitive expressions (as in 'On Computable Numbers').

Each document set out the basis for a very different practical version of the universal Turing machine (and the von Neumann design was indebted to extensive input from Eckert and Mauchly). Each document also replaced Turing's pioneering programming code of 1936 with a form of code more appropriate for high-speed computing. Again, each presented a very different species of code, von Neumann favouring instructions composed of operation codes followed by addresses, while Turing did not use operation codes: the operations to be performed were implied by the source and destination addresses.

Turing pursued the implications of the stored-program idea much further than von Neumann did at that time. As has often been remarked, in the 'First Draft' von Neumann blocked the wholesale modification of instructions by prefixing them

[160]Compare Fig. 10 of Turing's 'Proposed Electronic Calculator' (on p. 431 of Copeland et al., *Alan Turing's Electronic Brain*) with Fig. 3 of von Neumann's report (on p. 198 of Stern, *From ENIAC to UNIVAC*).

[161]*Evening News*, 23 December 1946. The cutting is among a number kept by Sara Turing and now in the Modern Archive Centre, King's College, Cambridge (catalogue reference K 5).

[162]Eckert, J. P., Mauchly, J. W. 'Automatic High Speed Computing: A Progress Report on the EDVAC', Moore School of Electrical Engineering, Sept. 1945. http://archive.computerhistory.org/resources/text/Knuth_Don_X4100/PDF_index/k-8-pdf/k-8-u2736-Report-EDVAC.pdf. Copeland is grateful to Bob Doran for pointing out this early occurrence of the term 'word' (in correspondence).

with a special tag. Only the address bits could be modified. Carpenter and Doran pointed out in their classic 1977 paper that, because von Neumann 'gave each word a nonoverrideable tag, he could not manipulate instructions', and they emphasized that it was Turing, and not von Neumann, who introduced 'what we now regard as one of the fundamental characteristics of the von Neumann machine'.[163]

The manipulation of instructions as if they were numbers was fundamental to the computer design that Turing put forward in 'Proposed Electronic Calculator'. He described program storage in editable memory as giving 'the machine the possibility of constructing its own orders'.[164] His treatment of conditional branching involved performing arithmetical operations on instructions considered as numbers (e.g. multiplying an instruction by a given digit).[165] Carpenter and Doran emphasized, 'Von Neumann does not take this step' (the step of manipulating instructions as if they were numbers) in the 'First Draft'.[166] The idea that instructions and data are common coin was taken for granted by ACE's programmers at the NPL. Sometimes instructions were even used as numerical constants, if an instruction considered as a number happened to equate to the value required.[167]

Furthermore, Turing recognized in 'Proposed Electronic Calculator' that a program could manipulate other programs.[168] As Carpenter and Doran again say, 'The notion of a program that manipulates another program was truly spectacular in 1945'.[169] Zuse had similar ideas in 1945, envisaging what he called a '*Planfertigungsgerät*' [plan producing machine], a 'special computer to make the program for a numerical sequence controlled computer'.[170] He added: 'This device was intended to do about the same as sophisticated compilers do today'.[171] Zuse discussed automated programming in his 1945 manuscript 'Der Plankalkül', describing this process as 'calculating calculating plans'.[172] Turing even envisaged programs that are able to rewrite their own instructions in response to experience. 'One can imagine', he said in a lecture on the ACE, 'that after the machine had been operating for some time, the instructions would have altered out of all recognition'.[173]

[163]Carpenter and Doran, 'The Other Turing Machine', p. 270; see also Carpenter and Doran, 'Turing's Zeitgeist'.

[164]Turing, 'Proposed Electronic Calculator', p. 382.

[165]Turing, 'Proposed Electronic Calculator', pp. 382–383.

[166]Carpenter and Doran, 'The Other Turing Machine', p. 270.

[167]Vickers, T. 'Applications of the Pilot ACE and the DEUCE', in Copeland et al., *Alan Turing's Electronic Brain*, p. 277.

[168]Turing, 'Proposed Electronic Calculator', p. 386.

[169]Carpenter and Doran, 'Turing's Zeitgeist'.

[170]Zuse, 'Some Remarks on the History of Computing in Germany', pp. 616–617.

[171]Zuse, 'Some Remarks on the History of Computing in Germany', p. 617.

[172]Zuse, 'Der Plankalkül' (manuscript), pp. 30–31.

[173]Turing, A. M. 'Lecture on the Automatic Computing Engine', in *The Essential Turing*, p. 393.

The two 1945 documents by Turing and von Neumann each had a very different philosophy. Essentially, von Neumann's 'First Draft' presented a novel form of numerical *calculator*. Right at the beginning of 'First Draft', in the course of what he called 'some general explanatory remarks', he offered this 'definition' of his subject matter: 'An *automatic computing system* is a (usually highly composite) device, which can carry out instructions to perform calculations of a considerable order of complexity'.[174] The EDVAC, he explained, would be a 'very high speed' automatic digital calculator. Turing, on the other hand, was envisaging a different kind of beast. For instance, he listed in 'Proposed Electronic Calculator' an assortment of *non*-numerical problems suitable for the ACE. These included solving a jig-saw, a problem that he described as 'typical of a very large class of non-numerical problems that can be treated', adding: 'Some of these have great military importance, and others are of immense interest to mathematicians.'[175] By this time Turing already had significant experience with non-numerical computation: his Bombe was designed to solve a specific type of non-numerical problem.[176] Turing also mentioned chess in 'Proposed Electronic Calculator', making his famous remark that 'There are indications ... that it is possible to make the machine display intelligence at the risk of its making occasional serious mistakes'.[177] Despite its modest title, 'Proposed Electronic Calculator' offered far more than a numerical calculator.

In January 1947 Turing travelled to the United States, to attend the Harvard Symposium on Large-Scale Digital Calculating Machinery. Organized by Aiken at his Harvard Computation Laboratory, this was the world's second sizable computing conference, with more than 300 delegates attending (a smaller conference with around 85 delegates was held at MIT in fall 1945).[178] With the birth of the stored-program electronic computer expected imminently, the time was ripe for a collection of visionary lectures; but Aiken, the leading champion in the US of electromechanical program-controlled computation, did not seize the moment. With the exception of Mauchly, Goldstine, and Jay Forrester—who at MIT was planning the Whirlwind I computer, one of the first stored-program machines to run (see the timeline in Fig. 3)—none of the leading pioneers of the new stored-program technology lectured at the symposium, not even the physically present Turing. His contributions were confined to astute comments during the brief post-lecture

[174] Von Neumann, 'First Draft of a Report on the EDVAC', p. 181 in Stern, *From ENIAC to UNIVAC*.

[175] Turing, 'Proposed Electronic Calculator', pp. 388–9.

[176] Turing, A. M. 'Bombe and Spider', in *The Essential Turing*.

[177] Turing, 'Proposed Electronic Calculator', p. 389.

[178] 'Members of the Symposium', *Proceedings of a Symposium on Large-Scale Digital Calculating Machinery. Jointly Sponsored by The Navy Department Bureau of Ordnance and Harvard University at The Computation Laboratory 7–10 January 1947*. Vol. 16 of *The Annals of the Computation Laboratory of Harvard University* (Cambridge, MA: Harvard University Press, 1948), pp. xvii–xxix.

discussions, including the following succinct expression of what we now see as the central feature of universal computers:

> We [at the National Physical Laboratory] are trying to make greater use of the facilities available in the machine to do all kinds of different things simply by programming ... This is an application of the general principle that any particular operation of physical apparatus can be reproduced ... simply by putting in more programming.[179]

To sum up, Turing's vision transcended the numerical calculator of von Neumann's 'First Draft'. Turing was planning an entirely new kind of machine, one capable of rewriting its own programs, of reproducing the behaviour of a wide range of different forms of physical apparatus, and even of exhibiting intelligence. As philosopher Teresa Numerico put it, 'In Turing's project, but not von Neumann's, we are confronted by a machine different from all previous machines.'[180]

In the end, it was von Neumann's simple, centralized design rather than Turing's quirky decentralized design that went on to become the industry standard, the ubiquitous 'von Neumann machine'. Von Neumann, though, was actually very clear—both in private and in public—in attributing the twin logical pillars to Turing. It is unfortunate that his statements are not more widely known. He explained in a letter in November 1946 that Turing's 'great positive contribution' was to show that 'one, definite mechanism can be "universal"'[181]; and in a 1949 lecture he emphasized the crucial importance of Turing's research, which lay, he said, in Turing's 1936 demonstration that a single appropriately designed machine 'can, when given suitable instructions, do anything that can be done by automata at all'.[182] Von Neumann's friend and scientific colleague Stanley Frankel recollected that von Neumann 'firmly emphasized to me, and to others I am sure, that the fundamental conception is owing to Turing'.[183] Frankel added, 'In my view von Neumann's essential role was in making the world aware of these fundamental concepts introduced by Turing ...'. IBM's Cuthbert Hurd, who also worked closely with von Neumann, emphasized 'I never heard him make the claim that he invented stored programming'.[184]

---

[179] 'Sheppard Discussion', *Proceedings of a Symposium on Large-Scale Digital Calculating Machinery*, p. 273.

[180] Numerico, T. 'From Turing Machine to "Electronic Brain"', in Copeland et al., *Alan Turing's Electronic Brain*, p. 182.

[181] Letter from von Neumann to Norbert Wiener, 29 November 1946; in the von Neumann Archive at the Library of Congress, Washington, D.C. (quoted on p. 209 of *The Essential Turing*).

[182] 'Rigorous Theories of Control and Information', in von Neumann, J. *Theory of Self-Reproducing Automata* (Urbana: University of Illinois Press, 1966; ed. Burks A. W.), p. 50.

[183] Letter from Frankel to Brian Randell, 1972 (first published in Randell, B. 'On Alan Turing and the Origins of Digital Computers', in Meltzer, B., Michie, D. (eds) *Machine Intelligence 7* (Edinburgh: Edinburgh University Press, 1972)). Copeland is grateful to Randell for giving him a copy of this letter.

[184] Hurd, C., Comments on Eckert, 'The ENIAC', in Metropolis, Howlett, and Rota, *A History of Computing in the Twentieth Century*, p. 536.

Returning to Moshe Vardi's efforts to refute the claim that Turing originated the stored-program concept, Vardi states—defending von Neumann's corner—that 'we should not confuse a mathematical idea with an engineering design'. So at best Turing deserves the credit for an abstract mathematical idea? Not so fast. Vardi is ignoring the fact that some inventions do belong equally to the realms of mathematics and engineering. The universal Turing machine of 1936 was one such, and this is part of its brilliance.

What Turing described in 1936 was not an abstract mathematical notion but a solid three-dimensional machine (containing, as he said, wheels, levers, and paper tape[185]); and the cardinal problem in electronic computing's pioneering years, taken on by both 'Proposed Electronic Calculator' and the 'First Draft', was just this: How best to build a practical electronic form of the universal Turing machine?

The claim that in 1936 Turing came up merely with an abstract mathematical idea, and moreover without perceiving any connection between it and potential real computing machinery, is a persistent one. Notoriously, 'Proposed Electronic Calculator' did not so much as mention the universal machine of 1936, leading some commentators to wonder whether even in Turing's mind there was any connection between the ACE and his earlier abstract machine (a doubt forcefully expressed by George Davis, a pioneer of computing from the pilot ACE era).[186] Computer historian Martin Campbell-Kelly also doubted that the universal Turing machine was a 'direct ancestor of the ACE', pointing out that the memory arrangements of the 1936 machine and of the ACE were very different, with the ACE's 'addressable memory of fixed-length binary numbers' having 'no equivalent in the Turing Machine'.[187]

However, some fragments of an early draft of 'Proposed Electronic Calculator' cast much new light on this issue.[188] The fragments survive only because Turing used the typed sheets as scrap paper, covering the reverse sides with rough notes on circuit design; his assistant, Mike Woodger, happened to keep the rough notes. In these fragments, Turing explicitly related the ACE to the universal Turing machine, explaining why the memory arrangement described in his 1936 paper required modification when creating a practical design for a computer. He wrote:

> In 'Computable numbers' it was assumed that all the stored material was arranged linearly, so that in effect the accessibility time was directly proportional to the amount of material stored, being essentially the digit time multiplied by the number of digits stored. This was

[185]Turing, A. M., draft précis (in French) of 'On Computable Numbers' (undated, 2 pp.; in the Turing Papers, Modern Archive Centre, King's College Library, Cambridge, catalogue reference K 4).

[186]George Davis, verbal comments at the ACE 2000 Conference, National Physical Laboratory, Teddington, 2000; and also at a seminar on Turing organised by the British Computer Conservation Society, Science Museum, London, 2005.

[187]Campbell-Kelly, 'The ACE and the Shaping of British Computing', pp. 156–157.

[188]These fragments were published for the first time as Turing, A. M. 'Notes on Memory', in Copeland et al., *Alan Turing's Automatic Computing Engine*, Oxford: Oxford University Press, 2005).

> the essential reason why the arrangement in 'Computable numbers' could not be taken over as it stood to give a practical form of machine. Actually we can make the digits much more accessible than this, but there are two limiting considerations to the accessibility which is possible, of which we describe one in this paragraph. If we have N digits stored then we shall need about log2N digits to describe the place in which a particular digit is stored. This will mean to say that the time required to put in a request for a particular digit will be essentially log2N × digit time. This may be reduced by using several wires for the transmission of a request, but this might perhaps be considered as effectively decreasing the digit time.[189]

Arguments that the ACE cannot have been inspired by the universal machine of 1936, since Turing did not mention his 1936 machine in 'Proposed Electronic Calculator', are plainly non-starters. It must also be remembered that the NPL hired Turing for the ACE project precisely because Womersley was familiar with, and had been inspired by, 'On Computable Numbers'.

Historian Thomas Haigh, who, like Vardi, is fighting in von Neumann's corner, weighs in on the side of the sceptics, attempting to raise doubt about whether 'Turing was interested in building an actual computer in 1936'.[190] He tries to undermine Max Newman's testimony on this point (almost as though he were von Neumann's lawyer), writing that the information 'is sourced not to any diary entry or letter from the 1930s but to the recollections of one of Turing's former lecturers made long after real computers had been built'.[191] Haigh fails to inform his readers that this former lecturer was none other than Newman, who (as explained above) played a key role in the genesis of the universal Turing machine, and for that matter also in the development of the first universal Turing machine in electronic hardware (to the point of securing the transfer, from Bletchley Park to the Manchester Computing Laboratory, of a truckload of electronic and mechanical components from dismantled Colossi).[192] Even in the midst of the attack on the German codes, Newman was thinking about the universal Turing machine: when Flowers was designing Colossus in 1943, Newman showed him Turing's 1936 paper, with its key idea of storing symbolically-encoded instructions in memory.[193] Donald Michie, a member of Newman's wartime section at Bletchley Park, the 'Newmanry'—home to nine Colossi by 1945—recollected that, in 1944–45, the Newmanry's mathematicians were 'fully aware of the prospects for implementing physical embodiments of the UTM [universal Turing machine] using vacuum-tube technology'.[194]

In fact, Newman's testimony about the foregoing point is rather detailed. He explained in a tape-recorded interview that when he learned of Turing's universal

[189]Turing, 'Notes on Memory', p. 456.

[190]Haigh, 'Actually, Turing Did Not Invent the Computer', p. 39.

[191]Haigh, 'Actually, Turing Did Not Invent the Computer', p. 39.

[192]Copeland et al., *Colossus*, p. 172; for a detailed account of Newman's role in the Manchester computer project, see Copeland, 'The Manchester Computer: A Revised History'.

[193]Flowers in interview with Copeland, July 1996.

[194]Letter from Michie to Copeland, 14 July 1995.

computing machine, early in 1936, he developed an interest in computing machinery that he described as being, at this time, 'rather theoretical'. Whereas, Newman continued, 'Turing himself, right from the start, said it would be interesting to try to *make* such a machine'.[195] Newman emphasized this same point in his obituary of Turing in *The Times*:

> The description that he then [1936] gave of a 'universal' computing machine was entirely theoretical in purpose, but Turing's strong interest in all kinds of practical experiment made him even then interested in the possibility of actually constructing a machine on these lines.[196]

Donald Davies, in 1947 a young member of Turing's ACE group at NPL, emphasized in a 1975 interview that the stored-program concept originated in 'On Computable Numbers'.[197] It seems to have been common knowledge among those involved with Turing at the NPL that the fundamental idea of the ACE derived from 'On Computable Numbers'. Sir Charles Darwin, Director of the NPL, wrote in a 1946 note titled 'Automatic Computing Engine (ACE)': 'The possibility of the new machine started from a paper by Dr. A. M. Turing some years ago'.[198] In 1947 Turing himself gave a clear statement of the connection, as he saw it, between the universal computing machine of 1936 and the electronic stored-program universal digital computer:

> Some years ago I was researching on what might now be described as an investigation of the theoretical possibilities and limitations of digital computing machines. I considered a type of machine which had a central mechanism, and an infinite memory which was contained on an infinite tape. ... [D]igital computing machines ... are in fact practical versions of the universal machine. There is a certain central pool of electronic equipment, and a large memory, [and] the appropriate instructions for the computing process involved are stored in the memory.[199]

4 A Hierarchy of Programming Paradigms

In a recent critique of the stored-program concept ('"Stored Program Concept" Considered Harmful'), Thomas Haigh maintains that 'discussion of the "stored program concept" has outlived the purpose for which it was created and provides

[195]Newman interviewed by Evans.

[196]Newman, M. H. A. 'Dr. A. M. Turing', *The Times*, 16 June 1954, p. 10.

[197]Davies interviewed by Evans.

[198]Darwin, C. 'Automatic Computing Engine (ACE)', National Physical Laboratory, 17 April 1946 (National Archives, document reference DSIR 10/385); published in Copeland et al., *Alan Turing's Automatic Computing Engine*, pp. 53–57.

[199]Turing, 'Lecture on the Automatic Computing Engine', pp. 378, 383.

a shortcut to confusion'.[200] He observes that the phrase 'stored program' rarely appeared in early documents, saying that this 'fairly obscure term' originated in an IBM internal memo in 1949.[201] Haigh continues:

> In the late 1970s and 1980s ... the idea of the 'stored program computer' was borrowed from technical discourse, developing from a fairly obscure term into a central concept in heated debates over what should be considered the first true computer and why.[202]

This 'resurgence of the stored program concept ... as a concept for historical discussion' was 'harmful', Haigh says, and '[t]he time has come to replace it'.[203] His readers are told that there is 'endemic confusion surrounding the stored-program concept'; and that the term 'stored program' is 'hopelessly overloaded with contradictory meanings' and is 'unhelpfully imprecise'.[204] However, Haigh offers no substantial arguments to support these views. Indeed, the main purpose of his diatribe against the stored-program concept appears to be to pave the way for his proposal to replace traditional terminology by a suite of neologisms that are designed to position von Neumann centre stage. If Haigh got his way we would all be speaking of 'the von Neumann architectural paradigm' and the 'modern code paradigm'—the latter a term explicitly introduced in order to 'describe the program-related elements of the 1945 First Draft'.[205]

Haigh's claim that the stored-program concept was by and large a construction of 1970s and 1980s historians does not withstand scrutiny. Far from the concept's being 'fairly obscure' before historians latched onto it, the concept in fact played a central role in numerous key documents from the early years of electronic computing. The reason that the *phrase* 'stored program' generally did not appear in these documents is simply that the founding fathers tended not to use the word 'program'. Von Neumann preferred 'orders', as did Zuse (*Befehle*).[206] Zuse also used the term 'calculating plan' (*Rechenplan*). Turing, in 1936, introduced the term 'instruction table' for what we would now call a program, and in 'Proposed Electronic Calculator' he continued to use this natural term, speaking of storing an instruction table or simply of storing instructions. In the introduction to the first of his Adelphi Lectures (the British counterpart of the Moore School Lectures,

[200]Haigh, T. '"Stored Program Concept" Considered Harmful: History and Historiography', in Bonizzoni, P., Brattka, V., Löwe, B. (eds) *The Nature of Computation. Logic, Algorithms, Applications* (Berlin: Springer, 2013), p. 247. See also Haigh, T., Priestley, M., Rope, C. 'Reconsidering the Stored-Program Concept', *IEEE Annals of the History of Computing*, vol. 36 (2014), pp. 4–17.

[201]Haigh, '"Stored Program Concept" Considered Harmful', pp. 243–244.

[202]Haigh, '"Stored Program Concept" Considered Harmful', p. 244.

[203]Haigh, '"Stored Program Concept" Considered Harmful', pp. 245, 247; Haigh, Priestley and Rope, 'Reconsidering the Stored-Program Concept', p. 12.

[204]Haigh, Priestley and Rope, 'Reconsidering the Stored-Program Concept', pp. 4, 14, 15.

[205]Haigh, '"Stored Program Concept" Considered Harmful', pp. 247, 249; Haigh, Priestley and Rope, 'Reconsidering the Stored-Program Concept', p. 12.

[206]Von Neumann, 'First Draft of a Report on the EDVAC', Sections 14–15 (pp. 236 ff in Stern, *From ENIAC to UNIVAC*).

although on a much smaller scale), Turing explained that 'the machine will incorporate a large "Memory" for the storage of both data and instructions'.[207] In an early homage to the joys of stored programming, he remarked enthusiastically that the 'process of constructing instruction tables should be very fascinating', continuing: 'There need 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'.[208]

Others followed Turing's usage. In their famous 1948 letter to *Nature*, announcing the birth of the Manchester Baby, Williams and Kilburn explained that the 'instruction table' was held in the computer's 'store'.[209] In the same letter they also used the term 'programme of instructions', and emphasized that 'the programme can be changed without any mechanical or electro-mechanical circuit changes'. Their selection of the terms 'store' and 'programme' proved to be a way of speaking that many others would also find natural, and by 1953, usage was sufficiently settled for Willis Ware (in a discussion of ENIAC and von Neumann's subsequent Princeton computer project) to be able to write simply: 'what we now know as the "stored program machine"'.[210] As for the centrality of the stored-program concept, in their writings from the period Williams and Kilburn repeatedly highlighted the concept's key position. For example, Kilburn said in 1949: 'When a new instruction is required from the table of instructions stored in the main storage tube, *S*, a "prepulse" initiates the standard sequence of events'.[211]

Similar examples can be multiplied endlessly from books and articles published on both sides of the Atlantic. That electronic computers could usefully edit their own stored programs was basic common knowledge. The 1959 textbook *Electronic Digital Computers* said:

> [I]t is at once apparent that instructions may be operated upon by circuitry of the same character as that used in processing numerical information. Thus, as the computation progresses, the machine may be caused to modify certain instructions in the code that it is following.[212]

We believe that the useful and well-known term 'stored program' is reasonably clear and precise. Like many terms, however, it will certainly benefit from some careful logical analysis, and this we now offer. Looking back over the early years of the computer's history, as outlined in Sect. 3, at least six different *programming*

[207] 'The Turing-Wilkinson Lecture Series (1946–7)', in Copeland et al., *Alan Turing's Electronic Brain*, p. 465.

[208] Turing, 'Proposed Electronic Calculator', p. 392.

[209] Williams, F. C., Kilburn, T. 'Electronic Digital Computers', *Nature*, vol. 162, no. 4117 (1948), p. 487.

[210] Ware, W. H. 'The History and Development of the Electronic Computer Project at the Institute for Advanced Study', RAND Corporation report P-377, Santa Monica, 10 March 1953, p. 5.

[211] Kilburn, T. 'The Manchester University Digital Computing Machine', in Williams, M. R., Campbell-Kelly, M. eds *The Early British Computer Conferences* (Los Angeles: Tomash, 1989), p. 138.

[212] Smith, C. V. L. *Electronic Digital Computers* (New York: McGraw Hill, 1959), p. 31.

paradigms can be distinguished. We denote these paradigms **P**1, **P**2, **P**3, **P**4, **P**5 and **P**6. To borrow Turing's onion-skin metaphor, the stored-program concept, like an onion, consists of a number of layers or levels. **P**3, **P**4, **P**5 and **P**6 are four such layers.

4.1 Paradigm P1

Programming consists of physically rewiring the machine, e.g. by means of plugs and switches. Colossus is a leading exemplar of this paradigm, and also ENIAC as originally designed. Since a signal can travel much faster along a wire than through, say, a vacuum tube circuit, this form of programming can lead to an inherently very fast machine—as in the case of Turing's Bombe, for example. **P**1's leading disadvantage is setup time. By 1945, the Colossus group were considering the use both of tape and punched cards to ease setup difficulties.[213] Eckert underlined the impoverished nature of **P**1 in reflections on the ENIAC: 'It was obvious that computer instructions could be conveyed in a numerical code, and that whatever machines we might build after the ENIAC would want to avoid the setup problems that our hastily built first try ENIAC made evident'.[214]

4.2 Paradigm P2

The main advantage of **P**2 over **P**1 is the ease of setting up for a new job and the corresponding reduction in setup time. Instructions are expressed in the form of numbers, or other symbols, and these numbers or symbols are stored in a memory medium such as tape or punched cards. The processes used in writing and reading the instructions are not of the same kind as those used in executing them (an echo of Gandy's statement, quoted above). Exemplars of this paradigm are Babbage's Analytical Engine, whose instructions were pre-punched into cards, and Aiken's ASCC and Zuse's Z1, Z2, Z3 and Z4, whose instructions were pre-punched into tape. This form of programming is read-only and the computer does not edit the instructions as the program runs.

We shall call machines programmed in accordance with **P**1 or **P**2 'program-controlled' machines, in order to set off **P**1 and **P**2 from **P**3–**P**6.

[213]Good, I. J., Michie, D., Timms, G. *General Report on Tunny*, Bletchley Park, 1945 (National Archives/Public Record Office, Kew; document reference HW 25/4 (vol. 1), HW 25/5 (vol. 2)), p. 331. A digital facsimile of *General Report on Tunny* is available in *The Turing Archive for the History of Computing* <http://www.AlanTuring.net/tunny_report>.

[214]Eckert, 'The ENIAC', p. 531.

4.3 Paradigm P3

Instructions are expressed in the form of numbers, or other symbols, and these numbers or symbols are stored in a (relatively fast) read-only memory. As with **P**2, the operations used in executing the instructions are not available to edit the instructions, since the memory is read-only. Writing the instructions into the memory may be done by hand—e.g. by means of a keyboard or hand-operated setting switches. The main advantage of **P**3 over **P**2 is that with this form of memory, unlike tape or cards, instructions or blocks of instructions can be read out repeatedly, without any need to create multiple tokens of the instructions in the memory (so avoiding copying blocks of cards or punching blocks of instructions over and again on the tape).

An exemplar of this paradigm is the modified form of ENIAC here called 'ENIAC-1948'. In order to simplify the setup process, ENIAC was operated from 1948 with instructions stored in a 'function table', a large rack of switches mounted on a trolley.[215] (Eckert recorded that this hand-switched, read-only storage system, essentially a resistance network, was based on ideas he learned from RCA's Jan Rajchman.[216]) The switch trolleys offered a slow but workable read-only memory for coded instructions. Richard Clippinger from Aberdeen Ballistic Research Laboratory (where ENIAC was transferred at the end of 1946) was responsible for ENIAC's transition to this new mode of operation. Clippinger explained:

> I discovered a new way to program the ENIAC which would make it a lot more convenient. ... I became aware of the fact that one could get a pulse out of the function table, and put it on the program trays, and use it to stimulate an action. This led me to invent a way of storing instructions in the function table.[217]

It seems, though, that Clippinger had reinvented the wheel. Mauchly stated that Eckert and he had previously worked out this idea.[218] Referring to Clippinger's rediscovery of the idea, Mauchly said: '[P]eople have subsequently claimed that the idea of such stored programs were [sic] quite novel to others who were at Aberdeen'. Eckert emphasized the same point: 'In Aberdeen, Dr. Clippinger later "rediscovered" these uses of the function tables'.[219] However, it can at least be said that Clippinger reduced the idea to practice, with the assistance of Nick Metropolis,

[215] Jennings said that ENIAC ran in this mode from April 1948, but Goldstine reported a later date: 'on 16 September 1948 the new system ran on the ENIAC'. Jennings Bartik, *Pioneer Programmer*, p. 120; Goldstine, *The Computer from Pascal to von Neumann*, p. 233.

[216] Presper Eckert interviewed by Christopher Evans in 1975 ('The Pioneers of Computing: an Oral History of Computing', Science Museum: London).

[217] Richard Clippinger interviewed by Richard R. Mertz in 1970 (Computer Oral History Collection, Archives Center, National Museum of American History, Smithsonian Institution, Washington, D.C.), p. I-I-11.

[218] John Mauchly interviewed by Christopher Evans in 1976 ('The Pioneers of Computing: an Oral History of Computing', Science Museum: London).

[219] Eckert, 'The Eniac', p. 529.

Betty Jean Jennings, Adele Goldstine, and Klari von Neumann (von Neumann's wife).[220]

In the secondary literature, this idea of using the switch trolleys to store instructions is usually credited to von Neumann himself (e.g. by Haigh in his recent von Neumann-boosting work).[221] But Clippinger related that 'When Adele Goldstine noted that I had evolved a new way to program the ENIAC, she quickly passed the word along to von Neumann'.[222] According to Clippinger and Jennings, what von Neumann contributed, in the course of discussions with Clippinger and others, was a more efficient format for the stored instructions, a one-address code that allowed two instructions to be stored per line in the function table (replacing Clippinger's previous idea of using a three-address code).[223]

There is a strong tradition in the literature for calling this paradigm 'stored program', and we follow that tradition here. Nick Metropolis and Jack Worlton said that ENIAC-1948 was 'the first computer to operate with a read-only stored program'.[224] Mauchly also referred to the switch trolley arrangement as involving 'stored programs' (in the above quotation). In passing, we note that we would not object *very* strongly to the suggestions that **P**3 be termed 'stored-program *in the weak sense*' or 'stored-program in the *minimal* sense', or even as transitional between **P**2 and genuine stored programming. However, the important point is that the major difference between **P**3 and **P**4–**P**6 should be marked somehow; and so long as the distinction is clearly enough drawn, it hardly matters in the end which words are used. Later in this section, a systematic notation is developed that brings out the minimal and somewhat anomalous status of **P**3.

4.4 Paradigm P4

Instructions in the form of numbers or other symbols are stored in a memory medium, in such a way that the processes used in reading and writing these instructions are of the same kind as those used in executing them. In other words, the processes used in executing the instructions can potentially 'get at' the instructions. The pre-eminent exemplar of this paradigm is the universal Turing machine as Turing described it in 1936.

---

[220]Clippinger interviewed by Mertz, p. I-I-14; Metropolis, N., Worlton, J. 'A Trilogy on Errors in the History of Computing', *Annals of the History of Computing*, vol. 2 (1980), pp. 49–59 (p. 54).

[221]Haigh, '"Stored Program Concept" Considered Harmful', p. 242. Also Goldstine, *The Computer from Pascal to von Neumann*, p. 233.

[222]Clippinger interviewed by Mertz, p. I-I-12. See also Metropolis and Worlton, 'A Trilogy on Errors in the History of Computing', p. 54.

[223]Clippinger interviewed by Mertz, pp. I-I-12, I-I-13; Jennings Bartik, *Pioneer Programmer*, pp. 11–12, 113.

[224]Metropolis and Worlton, 'A Trilogy on Errors in the History of Computing', p. 54.

In the context of electronic machines, use of **P**4 enables the computer to access its instructions at electronic speed, since the access operations are themselves electronic—so avoiding the 'instruction bottleneck' arising when instructions are supplied from some slower-than-electronic medium, such as punched tape.

4.5 Paradigm P5

P4 contains the potential for instruction editing. In **P**5 and **P**6 that potential is realized. The instruction editing made possible by **P**4 conforms to one or other of two logical types: instruction editing without the creation of new instructions, and editing that does produce new instructions. In **P**5 only the first occurs, whereas in **P**6 the second occurs. In **P**5, the processes used in executing instructions are also used to edit the instructions themselves, by adding, manipulating or deleting symbols, in order to *mark* or unmark portions of an instruction.

Again the pre-eminent exemplar is Turing's universal machine of 1936. Turing introduced a number of subroutines for inserting marker symbols into instructions, and for operating on marked up segments of an instruction in various ways; subsequently one of the subroutines would delete the markers, leaving the instructions exactly as they were originally found. For example, Turing's routines *kom* and *kmp* place markers showing the next instruction to be obeyed. His routine *sim* marks up the next instruction with various other symbols, and *inst* copies marked portions of the instruction to other locations on the tape, finally deleting the marker symbols from the instruction.[225]

4.6 Paradigm P6

P6 is but a very short step away from **P**5. In **P**5, the editing of instructions is limited to the insertion, manipulation and deletion of symbols that function as markers, while in **P**6 the editing processes from time to time delete and insert symbols in such a way as to produce a *different instruction*. **P**6 first appeared in the historical record in 1945. 'Proposed Electronic Calculator' and 'First Draft of a Report on the EDVAC' both describe this programming paradigm (although, as noted above, von Neumann initially protected symbols other than address bits from editing, only lifting this restriction in later publications).

In **P**6, the instructions that are edited may be those of the program that is actually running, as with address modification on the fly, or the modifications may be made to the instructions of some other program stored in memory. These different cases are here referred to as 'reflexive' and 'non-reflexive' editing respectively. Section 3

[225]Turing, 'On Computable Numbers', pp. 71–72 in *The Essential Turing*.

noted that in 1945 both Turing and Zuse farsightedly mentioned the idea of one program editing another. The importance of non-reflexive editing of stored programs was in fact widely recognized from computing's earliest days (further disproof of Haigh's claim that the stored program concept had only a 'fairly obscure' existence until the late 1970s). In one of computing's earliest textbooks, published in 1953, Alec Glennie wrote:

> It has been found possible to use the Manchester machine to convert programmes written in a notation very similar to simple algebraic notation into its own code. . . . [I]nstructions take the form of words which, when interpreted by the machine, cause the correct instructions in the machine's code to be synthesized. . . .[226]

In his 1959 textbook *Electronic Digital Computers*, Charles Smith wrote:

> This ability of the machine to modify a set of instructions to refer to various sets of operands makes it possible so to compress the instruction codes of many problems that they can be held, along with partial results and required data, in the limited internal memory of computers that have no relatively fast secondary or external memory.[227]

The key differences between **P**3, **P**4, **P**5 and **P**6 can be summarized as follows:

P3. This paradigm is: *read-only stored instructions*. In a notation designed to make the differences between **P**3, **P**4, **P**5 and **P**6 transparent ('*S*-notation'), **P**3 is *S0*, a step beneath the unmarked case *S*.

P4. **P**4 is simply *S*: *stored instructions* potentially accessible to editing by the processes used to execute the instructions. Actually making use of this potential leads on to **P**5 and **P**6.

P5. This paradigm is: *stored instructions/editing/no instruction change*. **P**5 is *S/E/=* ('=' signifying no change of instruction). **P**5 includes editing markers in a way that involves applying numerical operations, such as addition and subtraction, to the markers. For example, having marked an instruction with the marker '1010', the mechanism may repeatedly read out that instruction, each time subtracting 1 from the marker, stopping this process when the marker becomes 0000.

P6. This paradigm is: *stored instructions/editing/instruction change*. **P**6 is *S/E/*$\triangle I$ ('$\triangle I$' signifying instruction change). Where it is helpful to distinguish between reflexive and non-reflexive editing, *S/E/*$\triangle$*I/SELF* is written for the former and *S/E/*$\triangle$*I/OTHER* for the latter. *S/E* is sometimes used to refer to any or all of *S/E/=*, *S/E/*$\triangle$*I/SELF* and *S/E/*$\triangle$*I/OTHER*. For ease of reference, the *S*-notation just presented is summarized in Figure 1.[228]

Important logical features of **P**2, **P**3, **P**4, **P**5, and **P**6, as well as an important forensic principle, can be brought out by means of a short discussion of a memory model we call 'Eckert disk memory'. Eckert considered this form of internal

[226]Glennie, A. E. 'Programming for High-Speed Digital Calculating Machines', ch. 5 of Bowden, B. V. ed. *Faster Than Thought* (London: Sir Isaac Pitman & Sons, 1953), pp. 112–113.

[227]Smith, *Electronic Digital Computers*, p. 31.

[228]Copeland developed the *S*-notation in discussion with Diane Proudfoot.

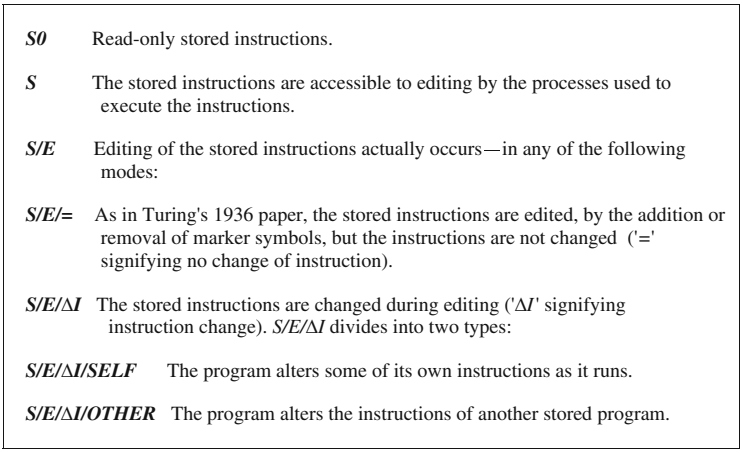


Fig. 1 The *S*-notation

memory in 1944, but soon abandoned the idea in favour of his mercury delay line, a technology he had had more experience with, and which, moreover, he thought would offer faster access times.[229] Eckert described the disk memory in a typewritten note dated January 1944.[230] He said 'The concept of general internal storage started in this memo'—but in fact that concept appeared in Zuse's writings in 1936, as Sect. 6 explains.[231]

Eckert disk memory consists of a single memory unit containing a number of disks, mounted on a common electrically-driven rotating shaft (called the time shaft). As Eckert described it, the memory unit formed part of a design for a desk calculating machine, an improved and partly electronic form of 'an ordinary mechanical calculating machine', he said.[232] Some of the disks would have their edges engraved with programming information expressed in a binary code. As the disk rotated, the engravings would induce pulses in a coil mounted near the disk. These pulses would initiate and control the operations required in the calculation (addition, subtraction, multiplication, division). Eckert described this arrangement as 'similar to the tone generating mechanism used in some electric organs'.[233]

The engraved disks offered permanent storage. Other disks, made of magnetic alloy, offered volatile storage. The edges of these disks were to be 'capable of being magnetized and demagnetized repeatedly and at high speed'.[234] Pulses would

[229]Eckert interviewed by Evans.

[230]The note is included in Eckert, 'The ENIAC', pp. 537–539.

[231]Eckert, 'The ENIAC', p. 531.

[232]Eckert, 'The ENIAC', p. 537.

[233]Eckert, 'The ENIAC', p. 537.

[234]Eckert, 'The ENIAC', p. 537.

be written to the disk edges and subsequently read. (This idea of storing pulses magnetically on disks originated with Perry Crawford.[235]) Eckert explained that the magnetic disks were to be used to store not only numbers but also function tables, such as sine tables and multiplication tables, and the numerical combinations required for carrying out binary-decimal-binary conversion.

A 1945 report written by Eckert and Mauchly, titled 'Automatic High Speed Computing: A Progress Report on the EDVAC', mentioned this 1944 disk memory. Eckert and Mauchly said: 'An important feature of this device was that operating instructions and function tables would be stored in exactly the same sort of memory device as that used for numbers'.[236] This statement is true, but requires some careful unpacking.

The storage of programming information on the engraved disks is an exemplification of paradigm **P**2. As with punched tape, the processes used in writing the engraved instructions (etching) and reading them (induction) are manifestly of a different kind from the processes used in executing the instructions. The processes used in execution are incapable of editing the engraved instructions. It is true that instructions are stored *in the same memory device* as numbers—i.e. the disk memory unit itself—but this device has subcomponents and, as we shall see, the subcomponents conform to different programming paradigms. Conceivably Eckert's intention was to control the rotation of the engraved disk in such a way that instructions or blocks of instructions could be read out repeatedly, in which case the engraved disk would conform to paradigm **P**3 rather than **P**2; however, Eckert did not mention this possibility in the memo.

Eckert also considered using a magnetic alloy disk for instruction storage, saying 'programming may be of the temporary type set up on alloy discs or of the permanent type on etched discs'.[237] Storing the programming information on an alloy disk exemplifies paradigm **P**4, since the same read and write operations that are used to fetch and store numbers during execution are also used to write the binary instructions onto the disk and to read them for execution.

When the instructions are stored on one of these magnetic alloy disks, the possibility arises of editing the instructions. For example, spaces could be left between instructions where temporary markers could be written to indicate the next instruction (**P**5); or, indeed, the instructions themselves might be edited, to produce different instructions (**P**6). However, Eckert mentioned none of this; the idea of editing instructions was simply absent from his memo. Of course, it is a short step conceptually from simply storing the instructions on one of these alloy disks to editing them. Nevertheless, it would be a gross mistake to say that Eckert's memo entertains either $S/E/=$ or $S/E/\triangle I$, since there is nothing in the document to indicate that editing of instructions was envisaged. Eckert might have been aware

[235]Eckert interviewed by Evans.

[236]Eckert and Mauchly, 'Automatic High Speed Computing: A Progress Report on the EDVAC', p. 2.

[237]Eckert, 'The ENIAC', p. 538.

of these possibilities at the time, or he might not. When interpreting historical documents, there is an obvious forensic principle that should be in play, to the effect that ideas not actually mentioned in the document must not be projected into it by the interpreter. Evidence internal to Eckert's 1944 memo enables us to state that he proposed *S* at this time, but there is no evidence in the document that his thinking extended to *S/E/=* or *S/E/△I*. Similarly, although Turing's 'On Computable Numbers' used *S/E/=*, and *S/E/△I* is but a very short step from *S/E/=*, it would be a mistake to ascribe *S/E/△I* to Turing's 1936 document. Turing might very well have realized in 1935–1936 that his machine's instructions could be edited to produce new instructions, simply by applying the editing processes not only to symbols marking the instructions but also to the symbols of the instructions themselves; however, there is no evidence of *S/E/△I* to be found in the actual document. Eckert's 1944 memo describes *S*, nothing more; and Turing's 1936 paper describes *S/E/=*, nothing more.

Armed with this forensic principle and these various distinctions, let us now turn to a consideration of Zuse's work.

5 Zuse and the Concept of the Universal Machine

In 1948, in a lecture to the Royal Society of London, Max Newman defined *general purpose* computers as 'machines able without modification to carry out any of a wide variety of computing jobs'.[238] Zuse had undoubtedly conceived the idea of a digital, binary, program-controlled general-purpose computer by 1936. In a patent application dating from April of that year he wrote:

> The present invention serves the purpose of automatically carrying out frequently recurring calculations, of arbitrary length and arbitrary construction, by means of calculating machinery, these calculations being composed of elementary calculating operations. ... A prerequisite for each kind of calculation that is to be done is the preparation of a calculating plan ... The calculating plan is recorded in a form that is suitable for the control of the individual devices, e.g. on a punched paper tape. The calculating plan is scanned by the machine, section by section, and provides the following details for each calculating operation: the identifying numbers of the storage cells containing the operands; the basic type of calculation; the identifying numbers of the cell storing the result. The calculating plan's fine detail [*Angaben*] automatically triggers the necessary operations.[239]

Concerning his use of the binary system, Zuse said:

> [O]ne can ignore human habits and choose the simplest number system. Leibniz previously recognized ... the system with base 2 to be the simplest system. This obviously holds for the case of the calculating machine as well.[240]

[238]Newman, M. H. A. 'A Discussion on Computing Machines', *Proceedings of the Royal Society of London*, Series A, vol. 195 (1948), pp. 265–287 (p. 265).

[239]Zuse, Patent Application Z23139, April 1936, pp. 1–2.

[240]Zuse, Patent Application Z23139, April 1936, p. 8.

Is there any evidence that Zuse went further than this in his thinking, to reach the point of formulating the concept of a *universal* computer, independently of Turing? A patent application dating from 1941 contained a passage that might be taken to suggest an affirmative answer. Our view, though, is that the correct answer is negative. In the 1941 application, Zuse first explained that

> New [to the art] is the combination of elements in such a way that orders are given to the whole system from a scanner ... The calculating unit A is connected with the storage unit C so that the calculating unit's results can be transferred to any arbitrary cell of the storage unit, and also stored numbers can be transferred to the individual organs [*Organe*] of the calculating unit.[241] P is the plan unit together with the scanner. It is from here that the calculating unit's operating keys are controlled, as well as the selection unit, Pb, which connects up the requisite storage cells with the calculating unit.[242]

About the arrangement just described Zuse claimed:

> By means of such a combination it is possible, in contrast to currently existing devices, to calculate every arbitrary formula [*Formel*] composed from elementary operations.[243]

This statement might be thought to parallel the Church-Turing thesis that every *effective* calculation can be done by the universal machine, and so to embody an independent notion of universality.[244]

We argue that this is not so. Zuse's 1936 and 1941 patent applications described an automatic numerical calculator—a very advanced form of relay-based desk calculator, programmable and capable of floating-point calculations, and yet compact enough and cheap enough to be stationed permanently beside the user, probably an engineer. Zuse's machine was, in a sense, a personal computer. Rojas said: 'As early as 1935 or so, Zuse started thinking about programmable mechanical calculators specially designed for engineers. His vision at the time, and in the ensuing years, was not of a large and bulky supercomputer but of a desktop calculating machine.'[245] Zuse's desktop calculator added, subtracted, multiplied, and divided. There is no trace in Zuse's 1936 and 1941 descriptions of his machine of Turing's grand vision of a universal machine able not only to calculate, but also to solve non-numerical problems, to learn, and to reproduce the behaviour of a wide range of different forms of physical apparatus. Not until his work on the *Plankalkül* did Zuse consider 'steps in the direction of symbolic calculations, general programs for relations, or graphs, as we call it today, chess playing, and so on'.[246]

---

[241] Von Neumann also spoke of 'specialized organs' for addition, multiplication, and so on; von Neumann, 'First Draft of a Report on the EDVAC', p. 182 in Stern, *From ENIAC to UNIVAC*.

[242] Zuse, Patent Application Z391, 1941, p. 4.

[243] Zuse, Patent Application Z391, 1941, p. 4.

[244] Copeland, B. J. 'The Church-Turing Thesis', in Zalta, E. (ed.) *The Stanford Encyclopedia of Philosophy*, http://plato.stanford.edu/entries/church-turing/.

[245] Rojas, R., Darius, F., Göktekin, C., Heyne, G. 'The Reconstruction of Konrad Zuse's Z3', *IEEE Annals of the History of Computing*, vol. 27 (2005), pp. 23–32 (p. 23).

[246] Zuse, 'Some Remarks on the History of Computing in Germany', p. 625.

The 'five operations: addition, subtraction, multiplication, division and finding the square root, as well as translating from decimal to binary and back' are the basis of Zuse's numerical calculator.[247] In his patent application dated December 1936, he gave a detailed account of the implementation of the operations in his *Relaistechnik* (relay technology).[248] These are the 'elementary operations' [*elementaren Rechenoperationen*], each of which, he explained, in fact reduces to the first, addition, with subtraction reducing to addition of the complement.[249] When Zuse asserted that his machine could 'calculate every arbitrary formula that is composed from elementary operations', there is no reason to think he was envisioning a machine capable of carrying out any and every conceivable algorithm (if given enough memory); and in fact every reason to think he was claiming simply that his machine could, in principle, carry out any arbitrary combination of the *elementaren Rechenoperationen*.

In later life, Zuse was given to saying that his early calculating machines were universal computers. For example, talking about Z1, Z2 and Z3 in a 1968 interview, he stated 'Machines up to the Z3 are universal'; and in 1980 he wrote that Z1–Z4 'were universal computers'.[250] However, we encountered nothing in the documents that we examined from the period 1936–1945 to make us think that Zuse arrived independently at Turing's concept of a universal machine. Indeed, so far as we can tell, Zuse's later pronouncements about universality seem in fact to concern the idea of a general-purpose machine rather than a universal machine. The first statement just quoted was made in connection with contrasting Z1–Z3 with various 'specialized machines', the several versions of a 'specialized calculator' that he built for the German aircraft industry during the war (see Sect. 2).[251] In the second quoted statement, Zuse immediately added 'for numerical calculations'. The phrase '*general-purpose* computers for numerical calculations' makes perfect sense.

In 1997, Rojas argued that Z3 is universal, since Z3 can be programmed to simulate the working of the universal Turing machine, and so with Z3 'one can, in principle, do any computation that any other computer with a bounded memory can perform'.[252] Of course, this does not show that Zuse himself had the concept of universality (and nor did Rojas suggest that it does). The crucial step in Rojas's proof was to establish that the universal Turing machine can be simulated by a program occupying a single (finite) loop of punched tape and containing only Zuse's *elementare Rechenoperationen* of addition, subtraction, multiplication, division, and square root. Rojas concluded: 'Zuse's Z3 is, therefore, at least in principle,

[247]Zuse, Patent Application Z391, 1941, p. 9.

[248]Zuse, Patent Application Z23624, December 1936.

[249]Zuse, Patent Application Z23139, April 1936, p. 12. Square rooting is not mentioned in Z23139 but is dealt with in Z391.

[250]Zuse interviewed by Merzbach; Zuse, 'Some Remarks on the History of Computing in Germany', p. 614; and Zuse makes the same claim in his interview with Evans.

[251]Zuse interviewed by Merzbach.

[252]Rojas, 'How to Make Zuse's Z3 a Universal Computer', p. 53.

as universal as today's computers that have a bounded addressing space'.[253] He admitted, though, that his programming approach 'greatly slows down the computations' of Z3 and that 'the large loop of punched tape ... would pose extraordinary and most likely unsolvable mechanical difficulties'.[254] In fact, it is something of an understatement to speak of a 'large' loop of tape.

Rojas's universality proof for Z3 was the first of a genre: there are now a number of logical results showing that ancient computers, never designed to be universal, are so in principle. For example, Benjamin Wells argued in 2003 that a cluster of ten Colossi can implement a universal Turing machine—and there *were* ten Colossi in 1945, nine in the Newmanry at Bletchley Park and another in Flowers' factory.[255] In a later paper Wells conjectured that 'a single late Colossus Mark 2 endowed with an appropriate tape punch and controller' can implement a universal Turing machine—specifically Alastair Hewitt's 2-state, 3-symbol machine, proposed in 2007.[256] Another proof in the Rojas genre, this time concerning Babbage's Analytical Engine, was offered by Leif Harcke. Harcke described 'a program for the Engine that simulates a small [4-state, 6-symbol] universal Turing machine'.[257] This appears to vindicate Turing's claim, doubted by some, that the 'Analytical Engine was a universal digital computer'.[258]

Universality, while not ubiquitous, turns out to be more widespread than might have been expected. It is even possible that Colossus's largely electromechanical predecessor at Bletchley Park (a machine known simply as 'Heath Robinson', after William Heath Robinson, the Rube Goldberg of British cartoon artists) was universal: Wells conjectures that the construction he used to prove Colossus to be universal could apply to Robinson also, because of its functional similarity to Colossus (although, given the current lack of complete knowledge of the Robinson's hardware, this lies beyond the bounds of formal proof). Indeed Marvin Minsky showed long ago that in the context of a suitable architecture, simply an ability to multiply and divide by 2 and 3 leads to universality (his proof concerns program machines with registers that are capable of holding arbitrarily large numbers).[259] In

[253]Rojas, 'How to Make Zuse's Z3 a Universal Computer', p. 51.

[254]Rojas, 'How to Make Zuse's Z3 a Universal Computer', p. 53.

[255]Wells, B. 'A Universal Turing Machine Can Run on a Cluster of Colossi', *American Mathematical Society Abstracts*, vol. 25 (2004), p. 441.

[256]Wells, B. 'Unwinding Performance and Power on Colossus, an Unconventional Computer', *Natural Computing*, vol. 10 (2011), pp. 1383–1405 (p. 1402); Hewitt, A. 'Universal Computation With Only 6 Rules', http://forum.wolframscience.com/showthread.php?threadid=1432.

[257]Harcke, L. J. 'Number Cards and the Analytical Engine', manuscript (Copeland is grateful to Wells for sending him a copy of this unpublished proof). Wells found a lacuna in Harcke's proof but he believes this to be harmless; Wells says 'A memory management limitation can be overcome by seamlessly incorporating virtual memory, as Harcke agrees'.

[258]Turing, A. M. 'Computing Machinery and Intelligence', p. 455 in *The Essential Turing*.

[259]Minsky, M. L. *Computation: Finite and Infinite Machines* (Englewood Cliffs: Prentice-Hall, 1967), p. 258.

the light of Minsky's theorem, it would have been rather curious had Z3 turned out *not* to be universal.

It is an undeniable feature of all these universality proofs for early machines that the proofs tell us nothing at all about these ancient computers as they actually existed and were actually used. Despite Wells' proof that the Colossi were universal, Flowers' actual machines were very narrowly task-specific. Jack Good related that Colossus could not even be coaxed to carry out long multiplication. This extreme narrowness was no defect of Colossus: long multiplication was simply not needed for the cryptanalytical processing that Colossus was designed to do. Wells' result, then, teaches us a general lesson: even the seemingly most unlikely devices can sometimes be proved to be universal, notwithstanding the actual historical limitations of the devices.

Similar remarks apply to Rojas's result about Zuse's machine. His proof tells us nothing at all about the machine as it was used, and viewed, within its own historical context, and nothing at all about its scope and limits as a practical computer. Nevertheless, these results are certainly not without interest. As Wells put it:

> Colossus *was* the first functioning electronic universal machine. The Analytical Engine, Colossus, and Z3 were all universal. This has nothing to do with the intentions or writings of Babbage, Flowers, or Zuse—it is an objective property of their creations.

6 Zuse and the Stored-Program Concept

Zuse presented instructions in two different formats, a high-level human-friendly format, and the low-level form punched into the programming tape. The high-level format described in his April 1936 patent application was a three-address code, with each instruction consisting of four components: (1) the address of the cell of the store containing the first operand; (2) the address in the store of the second operand; (3) the operation code, which Zuse wrote as 'Add.', 'Subt.', 'Mult.', and so on; and (4) the address to which the result was to be stored.[260] In his 1941 patent application, this three-address code was replaced by a single-address code. The instructions were of the following forms: Read from cell n; Store in cell n; $+$; $-$; $\times$; Output result.[261] The numerical output was displayed on banks of lights.[262]

Zuse's machine-level code corresponded to the punch holes in the program tape.[263] Each of six positions falling on a straight line across the width of the tape could be punched or blank. Every instruction was represented by eight of these lines of punch-holes; eight *Felder*, in Zuse's terminology. The two-dimensional array of holes making up a *Feld* operated the appropriate relays directly. The first two

[260] Zuse, Patent Application Z23139, April 1936, p. 4.

[261] Zuse, Patent Application Z391, 1941, p. 5.

[262] Zuse, 'Some Remarks on the History of Computing in Germany', p. 618.

[263] Zuse, Patent Application Z391, 1941, p. 40.

Felder indicated whether the instruction was a read-from-store, write-to-store, or an order to the calculating unit. Subsequent *Felder* were used to specify addresses, and also a calculating operation in the case of an order to the calculating unit. As well as punching instructions on the tape, Zuse also made provision for punching numbers, such as $\sqrt{2}$, π, g, and other frequently-used constants. A special tag indicated whether the next group of *Felder* contained an instruction or number, and in the latter case the tag's punch-pattern switched the scanner temporarily to number mode.

As Zuse described the fundamental working of the machine in 1936:

> For each operation, the machine's work routines consist of 4 phases [*Takten*], in accordance with the calculating plan: 1.) and 2.) transfer of the operands into the calculating unit, 3.) calculating operation, 4.) transfer of the result into the storage unit.[264]

In the 1941 design, groups of four machine-code instructions implemented this fetch-operate-store cycle.

Zuse's separation of the addressable store and the calculating unit is possibly what Schmidhuber was referring to when he said that, in the April 1936 patent application, Zuse 'described what is commonly called a "von Neumann architecture" (re-invented in 1945)'. However, this idea in fact goes back to Babbage's Analytical Engine, with its separate Store and Mill.[265] In the 1941 patent application, Zuse himself emphasized that the coupling of an individual storage unit and counting unit was 'well known', citing as prior art 'counting units attached to a drum'.[266]

The machine that Zuse described in detail in his 1936 patent applications falls fairly and squarely into programming paradigm **P**2. The punching and scanning operations used to implant the instructions on the tape and to read them into the machine are not of the same kind as the operations used to execute the instructions, viz store, read from store, add, subtract, and multiply.

However, after describing his machine's four-phase operation, Zuse gave a list of potential refinements and extensions, reproduced here in full:

> If a number remains the same for the next operation in the calculating device, the phases 'store the result' and 'bring over the 1st operand for the next calculation' can be omitted. In this way redundant phases are avoided.
>
> By installing two connections between the storage unit and calculating device, so enabling numbers to be transferred back and forth, phases can be nested.
>
> Several calculating units, storage units, distributors, scanners, punches etc. can be installed and thus several operations carried out at the same time.
>
> Frequently used numbers, such as $\sqrt{2}$, π, and g, can be made permanently available in fixed number storages.
>
> The scanner and punch for the initial values and result can be replaced by setting and reading devices.

[264] Zuse, Patent Application Z23139, April 1936, p. 5.

[265] Bromley, A. 'Charles Babbage's Analytical Engine, 1838', *Annals of the History of Computing*, vol. 4 (1982), pp. 196–217.

[266] Zuse, Patent Application Z391, 1941, p. 3.

> The calculating plan can be stored too, whereby the orders are fed to the control devices in synchrony with the calculation.
> Correspondingly, calculating plans can be stored in a fixed form if the machine is to carry out the same calculation often.[267]

Beyond those two brief sentences, noting the possibility of storing the calculating plan, Zuse said nothing more about the matter in his patent application. In particular, he did not say where or how the plan would be stored. If storage was to be in some unit logically equivalent to Eckert's engraved disks, then the device that Zuse was describing still conforms to paradigm **P**2. If, however, he meant that the calculating plan, expressed in binary code, was to be placed in the addressable relay store, together with the initial values, and whatever numbers were transferred to the store from the calculating unit as the calculation progressed—and it seems reasonable enough to interpret him in this way, since he mentions no other kind of storage apart from the addressable store and the punched tape—then Zuse can reasonably be taken to be suggesting *S0* programming, albeit very briefly. The 1938 documents examined later in this section tend to support this interpretation. If Zuse was indeed thinking of *S0* programming, however, there is little in the 1936 document to indicate that his thinking went beyond *S0* at this time (we discuss his 'two connections between the storage unit and calculating unit' below). In particular there is no evidence to suggest that the further steps involved in *S/E* were in his mind.

Schmidhuber's claim that in 1936, Zuse described 'a "von Neumann architecture" … with program and data in modifiable storage' is immensely misleading. What Zuse described in 1936, in great detail, was a **P**2 architecture. In two brief sentences he mentioned in passing the possibility of storing the program, but gave no architectural detail whatsoever. Moreover, far from offering further development in his 1941 patent application of this idea of storing the program, it is not even mentioned there. 'The calculating plan has the form of punched paper tape', Zuse stated in 1941.[268]

It is not so surprising that in his 1941 design Zuse did not pursue his idea of placing binary coded instructions in the relay store, nor implement the idea in Z3 or Z4. Any speed differential between the relay-based calculating unit and the tape mechanism was not so great as to create an instruction bottleneck, and the internal storage of instructions would use up precious cells of the relay store.

Nevertheless, Zuse did return to the program storage idea: half a dozen handwritten pages in a 1938 workbook extended his cryptic suggestion of 1936. The entries are dated 4 and 6 June and are a mixture of labeled diagrams and notes in the Stolze-Schrey shorthand system. Zuse's shorthand was transcribed into ordinary German by the *Gesellschaft für Mathematik und Datenverarbeitung* (Society for Mathematics and Data Processing) during 1977–1979.[269] In these pages, Zuse

[267] Zuse, Patent Application Z23139, April 1936, pp. 6–7.

[268] Zuse, Patent Application Z391, 1941, p. 40.

[269] Both workbook and transcription are in the Deutsches Museum Archiv, NL 207/01949. We are grateful to Matthias Röschner of the Deutsches Museum for information.

introduced a special 'plan storage unit' [*Planspeicherwerk*]. This was of the same type as the main relay store. The plan storage unit was coupled with a 'read-out unit' [*Abfühlwerk*] and these two units are shown in his diagrams as replacing the punched tape for supplying instructions to the calculating unit. As the notes progressed, Zuse dropped the distinction between the plan storage unit and the main relay store, using the main store for plan storage.

Zuse's principal focus in these notes was to develop a 'simpler way' of dealing with 'plans with iterating or periodical parts' (as mentioned previously, if a block of instructions punched on tape needs to be iterated, the instructions are punched over and over again, a clumsy arrangement). In addition to the plan storage unit, Zuse introduced a number of supplementary units to assist with program management and control. He named programs by a 'plan identification number' [*Plan-Nummer*], and to the plan storage unit he added a 'setting unit' [*Einstellwerk*] and a 'plan selection unit' [*Planwählwerk*]. These three units together with the read-out unit made up the 'plan unit' [*Planwerk*]. When a plan identification number was written into the setting unit, the plan selection unit would progressively select the lines of the store containing the identified plan, allowing the read-out unit to deliver the plan's instructions one by one.

Next Zuse introduced the idea of a numbered 'subplan' and of a 'self-controlling' calculating unit. He distinguished between what he called 'outer orders' [*äussere Befehle*] and 'inner orders' [*innere Befehle*]. He wrote: 'Outer orders control the work unit [*Arbeitswerk*], inner orders control the order unit [*Befehlswerk*]'. The work unit consists of the selection unit, the storage unit, and the operation unit (calculating unit). Unfortunately Zuse did not explain the term 'order unit' [*Befehlswerk*], which occurs only once in Zuse's notes and is presumably different from the plan unit [*Planwerk*].

In the default case, the outer orders are delivered sequentially from the plan storage unit. Inner orders, however, can trigger the operation of a 'counting unit' [*Zählwerk*]. There are two of these counting units, E_0 for the main plan and E_1 for the subplans. Zuse's diagram shows the plan storage unit supplying inner orders to the counting units. These inner orders appear to be essentially parameters. The arrival of a parameter from the plan storage unit at one of the counting units toggles the unit into action. Zuse wrote: 'Inner orders trigger changes in the relevant counting unit, whereby E_1 (subplan unit) carries on counting from the desired Nr and E_0 carries on counting from where it stopped last.' While E_0 is toggled on, the plan selection unit causes the plan storage unit to stream outer orders from the main plan to the work unit; and while E_1 is toggled on, the plan storage unit streams outer orders from the subplan. The subplan is selected by writing the subplan identification number into the setting unit.

Finally, Zuse described an arrangement whereby the plan storage unit was able to place the return address (the number of the instruction that will be read out of the plan storage unit once a subplan has been completed) into a special storage unit for instruction numbers. Zuse also explained that, by using additional counting units, all of them controlled by parameters from the plan storage unit, multiple nesting of subplans [*Mehrfachverschachtelung*] could be achieved.

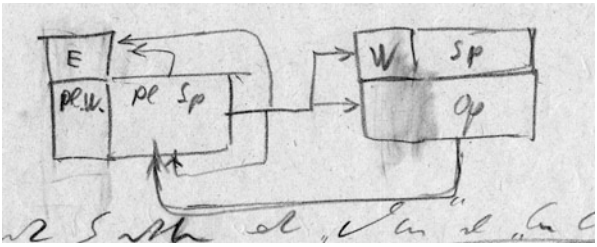


Fig. 2 Transferring information [*Angaben*] from the 'work unit' to the 'plan unit'. *E* is the setting unit, *Pl.W.* is the plan selection unit, *Pl Sp* is the plan storage unit, *W* is the selection unit, *Sp* is the storage unit, and *Op* is the calculating unit. *Credit: Deutsches Museum (Nachlass Konrad Zuse, Bestellnr. NL 207/0717)*

These arrangements of Zuse's are reminiscent of ENIAC-1948—and, like ENIAC-1948, exemplify **P**3. Instruction storage is read-only and Zuse is describing *S0*, not *S*. In Zuse's metaphor, the difference between **P**3 and **P**4 is the existence, in **P**4, not only of 'a controlling line going from left to right, but also from right to left'.[270] The effects of the arrangements described so far, Zuse said, can be achieved equally with punched tape. He emphasized that the whole setup described up to this point in the notes 'only serves the purpose of setting out rigid plans with iterating or periodical parts in a simpler way and it can be replaced in every case by a punched paper tape'.

Then, right at the end of the entry for June 4, Zuse adopted a new tack, in a brief section headed 'Living Plans' [*Lebende Pläne*]. He distinguished what he called living plans from 'rigid plans' [*starre Pläne*]. In two sketch diagrams and a mere 14 words of German he expounded the idea of allowing the work unit to write to the plan unit. Zuse heavily underlined the shorthand for 'Rück-Koppelung'—*feedback* between the work unit and the plan unit. This was a development of his pithy 1936 remark, quoted above, that by 'installing two connections between the storage unit and calculating device, so enabling numbers to be transferred back and forth, phases can be nested'. In this brief section of his 1938 notes, Zuse added a control line leading from 'right to left'. His diagram, showing an arrowed line leading back from the work unit to the plan unit, is captioned 'Transferring information [*Angaben*] from the "work unit" to the "plan unit"' (see Fig. 2).

So Zuse appears to have made the jump from **P**3 to **P**4, from *S0* to *S*. In his 1936 patent application, the two-way connection was suggested only as a means of permitting the nesting of phases [*Takten*], but in the 1938 notes, with his talk of 'living plans', Zuse seems to have had more in mind than nesting *Takten*.

In a final diagram, he swept away the distinction between the plan storage unit and the main store altogether, amalgamating the two. This showed two-way traffic between the work unit and the general-purpose store, with one arrowed line leading

[270]Zuse, 'Some Remarks on the History of Computing in Germany', p. 616.

from the store to the work unit, and a second arrowed line leading back from the work unit to the store.

How was the *potential* implicit in *S*—implicit in the bi-directional connection—to be used? About this Zuse wrote tantalizingly little. All he said, in his June 6 entry in the workbook, headed 'Dependent and independent feedback', was this:

> Independent feedback = independent from the initial details (note: initial details [*Ausgangsangaben*] = input values [*Eingabewerte*]) serves only to enable the plan to be represented in a more compact form, and for this to unfold as it runs. Plans with independent feedback are still said to be rigid.
> Dependent feedback = actual living plans. Effect [*Einfluss*] of the details [*Angaben*] calculated [*errechneten*], thus also of the initial details [*Ausgangsangaben*] on the sequence of events [*Ablauf*] in the computation [*Rechnung*].

These comments are altogether too cryptic for an interpreter to be certain what Zuse meant. We offer the following speculative example of a 'living plan', which causes the machine to select a subroutine for itself, on the basis of calculations from input values, and also to calculate from the input how many times to run the subroutine. We select this particular example in order to make the idea of a living plan vivid. The example derives from Turing's work on note-playing subroutines, as described in his *Programmers' Handbook for Manchester Electronic Computer Mark II*.[271] The example conforms well to Zuse's 1976 description of his early ideas: 'instructions stored independently and special units for the handling of addresses and subroutines'.[272]

Consider a simple piece of software for playing musical notes. The program takes as input values (a) a number *n*, and (b) the name of a musical note. In the case we will describe, the note is C_4, middle C (the subscript indicating the octave). These input values cause the computer to play the note of C_4, the number *n* functioning to tell the computer how long to hold the note. The computer plays the note by outputting a stream of suitable pulses (a stream of 1s separated by appropriate numbers of 0s) to an attached amplifier and loudspeaker. The actual details of how the sound is produced are not relevant to the example. The key point is that the note is generated by a subroutine consisting of a loop of instructions; running the loop continuously sends the correct stream of pulses to the loudspeaker. Each note-playing subroutine has a subplan identification number. *n* is the timing number, causing the computer to hold the specified note for *n* seconds.

Inputting the note name C_4 (in binary, of course) causes the calculating unit to calculate a subplan identification number from the binary input, by means of a fixed formula. The calculating unit then feeds this identification number back to the plan storage unit, and thence the number is transferred to the plan selection unit (by some

[271]Turing, A. M. *Programmers' Handbook for Manchester Electronic Computer Mark II* Computing Machine Laboratory, University of Manchester, no date, circa 1950; a digital facsimile is in *The Turing Archive for the History of Computing* at www.AlanTuring.net/programmers_handbook. See also Copeland, B. J., Long, J. 'Electronic Archaeology: Turing and the History of Computer Music', in Floyd, J., Bokulich, A. (eds) *Philosophical Explorations of the Legacy of Alan Turing* (New York: Springer, 2016).

[272]Zuse, 'Some Remarks on the History of Computing in Germany', p. 616.

mechanism that Zuse left unspecified). Next, the calculating unit divides *n* by the time taken to obey a single outer instruction (assumed to be a constant), and feeds this parameter *m* back to E_1, the counting unit for subplans. The parameter's arrival has the effect of toggling E_1 on, with the result that the subplan whose identification number is in the plan selection unit starts to run. After *m* steps, control passes back to the main program, at which point the loudspeaker has played C_4 for *n* seconds.

This example of **P**4 programming, which involves no editing of instructions, appears to us to illustrate the principles described by Zuse in his 1938 notes. While it cannot be ruled out that Zuse might have had in mind editing an inner instruction in the course of the unspecified steps leading to the delivery of the subplan identification number and the parameter *m* to their respective units, he certainly did not say so. Zuse in fact gave only the vaguest description of the inner order to E_1, saying merely that these orders were of the form: '"continue E_1" Nr ...'. More importantly, we find no trace of evidence in these notes that Zuse was thinking of *S/E* (either *S/E/*$\triangle I$ or *S/E/*=) in the case of outer orders.

7 Concluding Remarks

To conclude, we reprise the main events in the stored-program concept's early history. We also present translations of key remarks from Zuse's 1945 manuscript 'Der Plankalkül', which, unlike his 1936 and 1938 documents, did describe instruction editing.

The stored-program story began in 1936 (see the timeline in Fig. 3), when in 'On Computable Numbers' Turing published a description of *S/E/*=: storage of instructions, with editing, but no creation of different instructions. *S/E/*$\triangle I$, editing that does lead to new instructions, is very obviously available in the setup Turing described in 'On Computable Numbers'; but he did not mention *S/E/*$\triangle I$ in 1936. In the absence of evidence, it is impossible to say whether or not he noticed this intriguing possibility at the time.

Also in 1936, in an unpublished and subsequently withdrawn application for a patent, Zuse gave a bare mention of *S0*, the bottom rung of the hierarchy of levels comprising the stored-program concept. He also introduced, very briefly, a proto-version of his idea of two-way traffic between the storage unit and the calculating unit, limiting the scope of the idea to nesting *Takten*. In 1938, in a few sheets of handwritten personal notes, Zuse developed his brief 1936 suggestion of placing instructions in his addressable relay store. In these notes he initially described a sequence of architectures involving *S0*, and then, by adding *Rück-Koppelung*—feedback from the work unit to the plan unit—he apparently described a calculating machine that implements *S*. Zuse therefore seems to have been the first to sketch (in barest outline) a practical architecture in which *S* could be achieved. However, he did little at this time to tap the potential of *S*. Not even his 'living plans' can be said to involve *S/E/*= or *S/E/*$\triangle I$: any such claim would go well beyond the evidence present in the document.

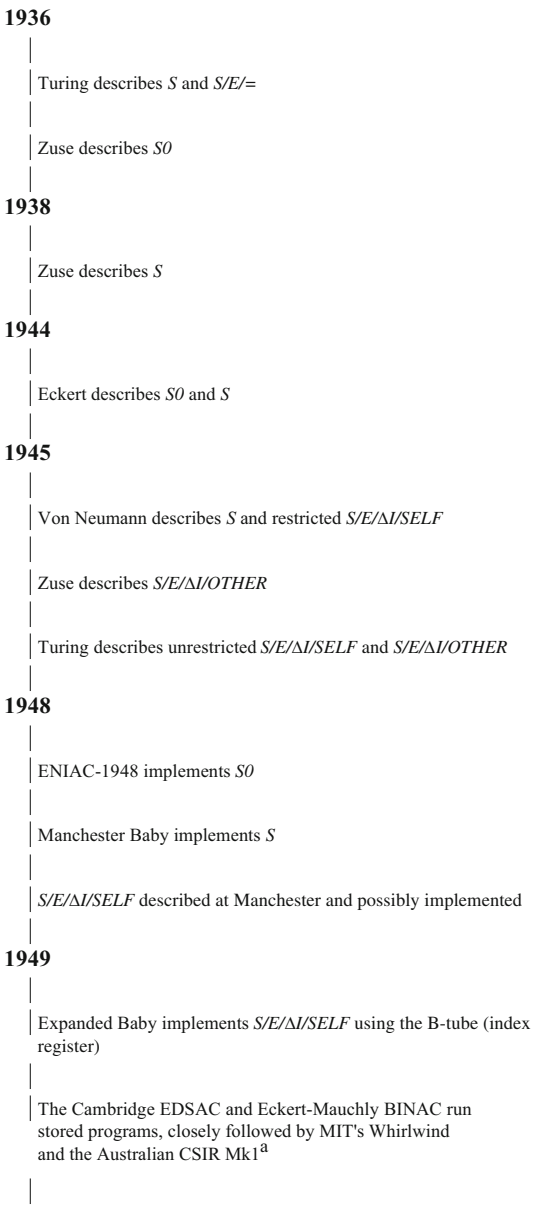


Fig. 3 Timeline: early history of the stored-program concept[a]

[a]Wilkes, M. V. *Memoirs of a Computer Pioneer* (Cambridge, Mass.: MIT, 1985), p. 142; Lukoff, H. *From Dits to Bits: A Personal History of the Electronic Computer* (Portland, Oregon: Robotics Press, 1979), p. 84; Woodger, M. 'Stored-program Electronic Digital Computers' (handwritten

Zuse then turned his back on the stored-program idea. His 1941 Z3 and 1945 Z4 used punched tape to control the computation and are exemplars of the **P**2 programming paradigm. Much later, reflecting on his early work, Zuse said he had felt that implementing the *Rück-Koppelung* of his 1938 notes 'could mean making a contract with the devil'.[273] 'When you make feedback from the calculating unit to the program, nobody is able to foresee what will happen', he explained.[274] Zuse emphasized that Z1–Z4 had 'no feedback in the program unit'.[275]

When Zuse began developing his *Plankalkül* language, however, he did return to the stored-program concept. In 1945, the *annus mirabilis* of the stored-program concept, Zuse wrote his five-chapter manuscript 'Der Plankalkül'. Unpublished and little-known, this document is the shadowy third member of a momentous trilogy of reports from 1945, each developing the stored-program concept in its own way—the others being, of course, von Neumann's 'First Draft of a Report on the EDVAC' and Turing's 'Proposed Electronic Calculator'.

In a section of 'Der Plankalkül' titled 'Repetition plans', Zuse said:

> [T]he orders contained in the repetition part [of the plan] are subject to continuous changes that result from the repetitions themselves.[276]

A 'variation instruction' [*Variationsvorschrift*] produces 'variations of the plan' [*die Variationen des Planes*].[277] There is, however, no 'contract with the devil': Zuse's examples of repetition plans seem, as with his 1938 notes, to involve *S* but no editing of instructions within the meaning of the act.

'Der Plankalkül' does clearly set out the concept of *S/E/△I/OTHER*, describing the automatic 'calculating of calculating plans'. Zuse wrote:

> The modifying of calculating plans is a function of the plan variables, which can consist of variable operation symbols, plan symbols [introduced so that 'subplans occurring inside a calculating plan can be modified'], structure symbols ['calculating plans of the same structure can adopt different meanings by modification of the algebraic dimension'], or other values. . . .
>
> The use of such variable structure symbols plays an important role when the task is to represent calculating plans for storage in calculating machines in as compact a form as possible, in order to capture far-reaching variation with little extra expenditure. . . .
>
> In the cases discussed, calculating plans are modified by the simple insertion of variable symbols. However, modifications of an essentially more complicated nature are possible. One can move from this kind of calculating plan to free calculating plans . . .[278]

---

note in the Woodger Papers, Science Museum, Kensington, London); McCann, D., Thorne, P. *The Last of the First. CSIRAC: Australia's First Computer* (Melbourne University Press, 2000), p. 2.

[273]Zuse, 'Some Remarks on the History of Computing in Germany', p. 616.

[274]Zuse interviewed by Evans.

[275]Zuse, 'Some Remarks on the History of Computing in Germany', p. 616.

[276]Zuse, 'Der Plankalkül' (manuscript), p. 32.

[277]Zuse, 'Der Plankalkül' (manuscript), p. 32.

[278]Zuse, 'Der Plankalkül' (manuscript), pp. 23, 24, 25.

Free calculating plans, seemingly the same as or at any rate similar to the 'living plans' of Zuse's 1938 workbook, are those in which 'the actual variables [*eigentlichen Variablen*] have an effect on the course of the calculation'.[279] In 'Der Plankalkül', Zuse described the automatic calculation of free calculating plans, noting that this process might determine 'only parts' of the plan, leaving other details to be filled in by the machine as the plan actually runs.[280] The *Plankalkül* and Zuse's early ideas about compilation are topics for a more detailed discussion in a further article.

Turing also described *S/E/△I/OTHER* in 1945 and, of course, Turing and von Neumann both described *S/E/△I/SELF* in that year (in their respective reports on the ACE and the EDVAC). Von Neumann restricted the scope of editing to the instruction's address bits, whereas Turing described unrestricted editing of instructions. It was not until his later papers, from 1946 onwards, that unrestricted *S/E/△I/SELF* appeared in von Neumann's work.

Jumping back a year in order to mention developments at the Moore School: in 1944 Eckert described a disk memory device involving both *S0* and *S*, but there is no evidence that he was considering *S/E/△I* or even *S/E/=* in connection with this device. Around the same time, he and Mauchly conceived the idea of storing instructions in the ENIAC's function tables, an idea later reduced to practice at the Ballistic Research Laboratory, by Clippinger and others, in 1948. Storing instructions in ENIAC-1948s read-only function tables was an example of the **P**3 programming paradigm, achieving *S0* but not *S*.

S and *S/E/△I/SELF* were first implemented at Manchester, in Max Newman's Computing Machine Laboratory. *S* was achieved in June 1948, but initially the Baby was used without instruction editing, as a surviving laboratory notebook shows.[281] During the summer, though, Kilburn added a kind of halfway house, a relative control transfer, achieved by adding a number from the store to the address of the next instruction (or subtracting the number from the address), this address being stored in a control tube.[282] Soon the editing of instructions was considered useful enough for Williams and Kilburn to add special hardware for it. The new hardware was known simply as the *B*-tube, the accumulator already being named the *A*-tube and the control tube the *C*-tube. In modern terms, the *B*-tube was an index register. Kilburn explained:

> Instructions can, of course, be modified by the normal processes ... in the same way as numbers, but this is often inconvenient and wasteful of time and storage space. Therefore each instruction ... is preceded by a single digit called the *b* digit. If $b = 0$, the content of

[279] Zuse, 'Der Plankalkül' (manuscript), p. 25.

[280] Zuse, 'Der Plankalkül' (manuscript), p. 31.

[281] Tootill, 'Digital Computer—Notes on Design & Operation'.

[282] Kilburn interviewed by Copeland, July 1997.

line *B0* of *B* (normally zero) is added into the present instruction ... before this instruction is used. If $b = 1$, the content of line *B1* of *B* is used in the same manner.[283]

Baby. Tom Kilburn is on the left, Freddie Williams on the right. *Credit: University of Manchester School of Computer Science*

In the Manchester laboratory notebook previously mentioned (compiled by engineer Geoff Tootill), the earliest dated appearance of the *B*-tube idea was in an entry for 13 October 1948, giving coded instructions for the operations that Kilburn described in the above quotation.[284] The original idea had emerged some weeks earlier, during a discussion between Newman, Williams, Kilburn and Tootill.[285] It arose initially (Williams and Kilburn explained) as 'a convenient means of shifting the effect of a whole block of instructions by a constant amount, while leaving others

[283]Kilburn, T. 'The University of Manchester Universal High-Speed Digital Computing Machine', *Nature*, vol. 164, no. 4173 (1949), pp. 684–7 (p. 687).

[284]Tootill, 'Digital Computer—Notes on Design & Operation', list of the machine's instructions dated 13/10/48. There are also two undated references to instructions involving the B-tube a few pages earlier. Kilburn included the same B-tube instructions in his 'Code for Charge Storage Computer', a list of the machine's instructions that is dated 30 November 1948. (Copeland is grateful to Simon Lavington for sending him a copy of Kilburn's document, seemingly a lecture handout. Lavington included a retype of the document in his 'Computer Development at Manchester University', in Metropolis, Howlett and Rota, *A History of Computing in the Twentieth Century*, p. 439.)

[285]Kilburn interviewed by Copeland, July 1997.

that were not B-modified unaffected'.[286] A young assistant, Dai Edwards, was given the job of developing the new piece of hardware.[287]

The B-tube was probably being used to run engineering test programs by March 1949 and was ready for routine use in April 1949.[288] It was part of an extensive upgrade, completed in April, that transformed the computer from a small demonstration model into a usable machine: the upgrade also included a magnetic drum memory, improved CRT storage, a hardware multiplier, and an increase in word length to 40 bits.[289] Manchester was well ahead of the field, with a number of other pioneering computer projects in the UK, US and Australia succeeding in running stored programs later in 1949 (see the timeline in Fig. 3).

Was the less convenient method of instruction modification that Kilburn mentioned—not involving the B-tube—ever actually implemented during the period June 1948–April 1949? This is a tantalizing and important question, since the first implementation of instruction editing was a key moment in the history of computing. The method was important enough for Turing to discuss it in detail in his *Programmers' Handbook*: he described 'the formation of an instruction in the accumulator by addition, the copying of this instruction into the list of instructions, and the subsequent obeying of it' (adding, 'This is a rather clumsy process').[290] Turing gave several examples of small programs using this method of instruction editing, in sections of the *Handbook* describing what he called the 'reduced machine'. The 'reduced machine' is by and large the Manchester computer as it existed before April 1949.

Thus it is certainly possible that Manchester's first use of instruction editing preceded the arrival of the B-tube. But if so, no record appears to have survived. Therefore the question of precisely when instruction editing was first implemented—one of computer science's most historic dates—remains open.

Acknowledgments Copeland is grateful to the following institutions for supporting this research: University of Canterbury, New Zealand; University of Queensland, Australia; Federal Institute of Technology (ETH), Zurich, Switzerland; and Det Informationsvidenskabelige Akademi,

[286]Williams, F. C., Kilburn, T. 'The University of Manchester Computing Machine', in Bowden, *Faster Than Thought*, p. 122.

[287]Lavington, S. H. *A History of Manchester Computers* (Manchester: NCC Publications 1975), p. 12.

[288]Tootill, 'Digital Computer—Notes on Design & Operation', note dated 27/3/49 inserted within an older note dated 15/7/48; Dai Edwards in interview with Simon Lavington, 6 May 2015. (Copeland is grateful to Lavington for making the transcript of this interview available.)

[289]Williams and Kilburn, 'The University of Manchester Computing Machine', pp. 121–122; Edwards in interview with Lavington, 6 May 2015; Kilburn, T. Tootill, G. C., Edwards, D. B. G., Pollard, B. W. 'Digital Computers at Manchester University', *Proceedings of the Institution of Electrical Engineers*, vol. 77 (1953), pp. 487–500.

[290]Turing, *Programmers' Handbook for Manchester Electronic Computer Mark II*, pp. 19–20. A typo has been corrected: in Turing's original, the second occurrence of 'instruction' was typed 'instructions'.

Copenhagen University, Denmark. Copeland and Sommaruga are grateful to Herbert Bruderer, Brian Carpenter, Martin Davis, Bob Doran, Simon Lavington, Teresa Numerico, Diane Proudfoot, and Benjamin Wells for helpful comments on a draft of this chapter, and to the Deutsches Museum for providing digital facsimiles of various original documents by Zuse.