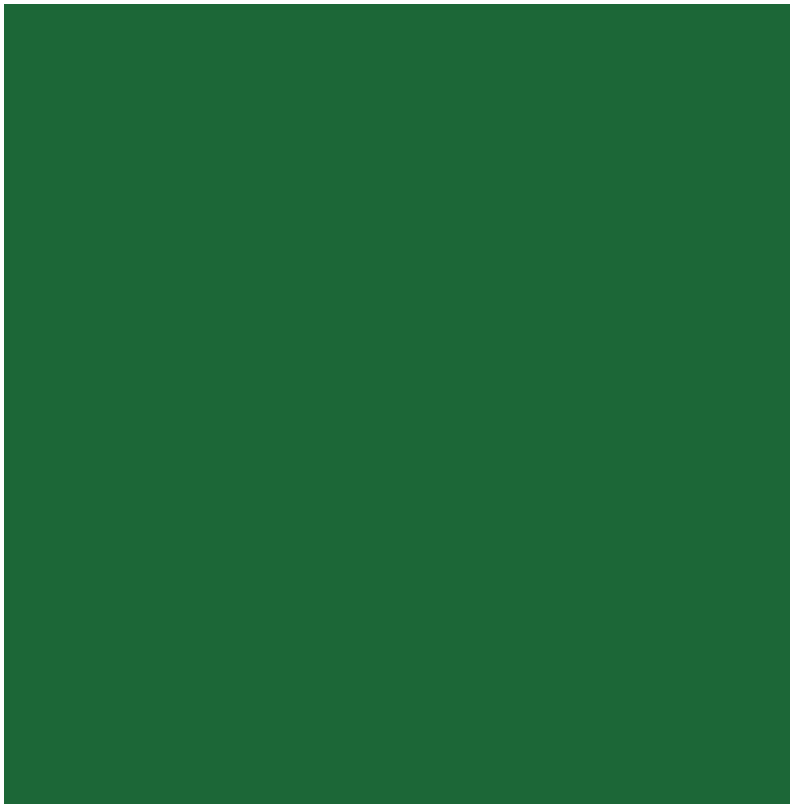
Editor: **Gerard J. Holzmann**
NASA/JPL
gholzmann@acm.org

Dead Programs

Gerard J. Holzmann



COMPUTER SCIENCE IS young enough as a discipline that we can still talk to some of its pioneers. It's astonishing to realize how recently some of the key developments really took place.

As an example, I'll look at the mechanism that's used pretty much universally today to implement function calls with a stack, using "push" and "pop" instructions to modify its contents (see Figure 1). The stack stores parameters for the call and, most important, the return address, so that you know where the execution should resume after the function completes. A stack might seem the obvious data structure to handle this—how else could you support nested calls? But somebody had to be the first to figure this out.

The Prehistory

When I say that computer science is young, I mean that the first practical general-purpose computers were constructed less than a lifetime ago. The construction of more specialized devices for performing specific types of calculation has, of course, a much longer history. The oldest known example is probably the Antikythera mechanism, which could predict the timing of solar eclipses. It originated in the second century BCE.

The first serious attempt to construct a general-purpose computer was by Charles Babbage. He first described the design for his analytical engine in 1837, but the challenges of actually creating a working machine with the then-available technology were overwhelming.[2]

Already at the time of Babbage, a

 0740-7459/17/$33.00 © 2017 IEEE

better understanding of electricity and magnetism started emerging, especially after Hans Christian Ørsted discovered the link between the two in 1820.[3] But these insights took another century to find their way into working computers. That development was clearly in the air in the 1930s, when Alan Turing developed the theory of general-purpose computing machines.[4] No working computers existed yet, but that soon changed.

The First Relay Machines

In the 1930s, in Germany, Konrad Zuse designed and built one of the first electromechanical computers. Around the same time, in the US, Bell Labs researcher George Stibitz built a series of computing devices using electromechanical relays. Relays were used in all telephone switches at the time, so they weren't hard to get. Similarly, vacuum tubes for amplifying signals were readily available. In 1944, in the Netherlands, 18-year-old Willem van der Poel, about whom you'll hear more shortly, also started designing relay computers. A few years later, as a student at Delft University, he submitted the design for a complete electromechanical computer to a contest. He had to settle for an honorable mention.

A key step in the development came in 1945 with the construction of EDVAC (Electronic Discrete Variable Automatic Computer). EDVAC introduced the now familiar von Neumann architecture, in which programs were stored like data in the main memory of a machine, rather than being hard-wired on plugboards.[5] George Dyson's book *Turing's Cathedral* offers a fascinating description of these early developments.[6]

The Testudo and PTERA

With the news of EDVAC spreading, van der Poel got a chance to help build the first Dutch computer in 1947 to help the Delft University optics group automate its calculations. The machine became the subject of his master's thesis three years later. The machine, nicknamed Testudo (Latin for tortoise), took 30 seconds for an addition and 45 seconds for a multiplication, but it was used for many years because it could work tirelessly through the night to perform its calculations.

After graduating, van der Poel joined PTT (the Dutch telephone service provider at the time), which was of course an excellent place to get access to large numbers of telephone relays. He worked there with Leendert Kosten on designing and constructing a series of computers that were based directly on the von Neumann architecture. One of the first of those machines was PTERA (PTT Elektronische Rekenautomaat, or PTT Electronic Computation Automaton), which went into operation in 1953. Both van der Poel and Kosten later joined Delft University as professors in the math department, where I later became their student.

I studied at Delft University for the better part of the '70s. Curiously, given that the Delft faculty included some of the pioneers of computer science, the school didn't yet offer a computer science degree. So, I picked something that seemed close enough. I took courses from Kosten and van der Poel, and later did my PhD with van der Poel as my advisor. Remarkably, I didn't know his or Kosten's role in the early history of computing in the Netherlands until much later. The topic just never came up, and, yeah, there was no Internet to google those things yet.

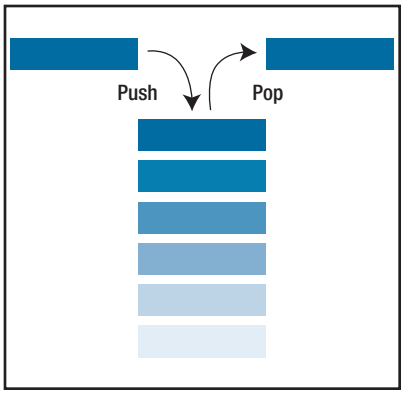


FIGURE. 1 A call stack with push and pop instructions.[1] A stack might seem the obvious data structure to handle function calls, but somebody had to be the first to figure this out.

Who Invented the Stack?

Just how important some of the early contributions of van der Poel were I discovered only recently when I came across a casual note posted in an Internet forum about using stacks for subroutine calls. The note said in part,

> *Subroutines were invented in 1947 on EDSAC [Electronic Delay Storage Automatic Calculator] by David Wheeler, and they were called "Wheeler Jumps." Subroutine calling via a LIFO [last-in, first-out] "stack" is a very old idea, invented by Willem Louis van der Poel in 1952.*[7]

The Wheeler Jump consisted of writing the return address for a subroutine call as a jump instruction at the end of the code of the subroutine just before the call.[8] That assumes, of course, that the subroutine code lives in regular RAM, as it would in a von Neumann-type machine.

The idea is elegant but has at least

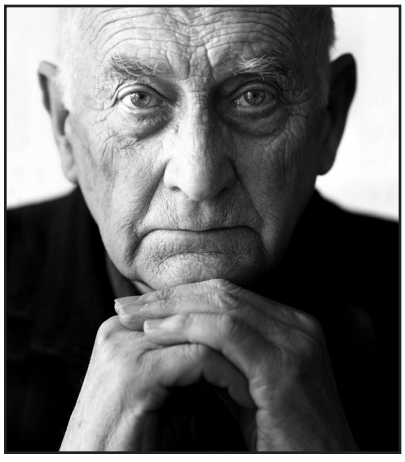


FIGURE 2. Willem van der Poel in 2010. (Source: Sam Rentmeester Fotografie; used with permission.)

two shortcomings, besides that part about using self-modifying code, of course. First, the Wheeler Jump makes it impossible to use either direct or indirect recursion for subroutine calls. This problem was likely considered less important at the time because no general consensus yet existed that recursive subroutine calls were even desirable.

Second, if program code is placed in write-protected memory, the return address can't be inserted on the fly. A way around this is to store the return address in some other designated place in writable memory. Edsger Dijkstra described this method as late as in 1959 in his PhD thesis, in which he talked about programming the X1 computer.[9] The X1 had been designed at the Mathematical Center in Amsterdam and built by the Dutch company Electrologica. It reserved eight places in writable memory for storing subroutine return addresses, but the programmer had to keep track of which one to use.

Dead Programs

To overcome some of the flaws of the Wheeler Jump, van der Poel proposed implementing a push-down stack in writable memory for subroutine calls. For PTERA, "regular memory" meant magnetic drum memory (for a look at that memory, visit www-set.win.tue.nl/UnsungHeroes/machines/pterad672.html?ssc=1,3). Van der Poel described this method in December 1952 in an article with the mysterious title "Dead Programmes for a Magnetic Drum Automatic Computer."[10] "Dead programmes" were programs stored in write-only memory. Van der Poel wrote that to protect program code, PTERA let users block write access to parts of the drum, using a small plugboard. The term "dead" came back also in references to the plugboard itself as providing "dead registers" that the program couldn't modify.

Although the function call stack seems to us like the obvious way to implement function calls, it wasn't immediately picked up as the right method. This is clear from Dijkstra's PhD thesis, which appeared seven years after van der Poel's article was published but still described the older method.

Happily, some of the pioneers are still among us, but as the discipline matures, their numbers are dwindling. I consider myself lucky that I could learn my trade from one of them, even though I didn't realize it at the time. Willem van der Poel (see Figure 2) is now 90 years old and continues to work as he always has. It's just that the computers he uses are a bit faster than the ones he started working on almost three-quarters of a century ago. By about 11 orders of magnitude, that is.

References

1. "Stack (Data Structure)," *Wikipedia*, 2016; simple.wikipedia.org/wiki/Stack_(data_structure).
2. A.G. Bromle, "Charles Babbage's Analytical Engine, 1838," *IEEE Annals of the History of Computing*, vol. 4, no. 3, 1982, pp. 197–217.
3. H.C. Ørsted, *Selected Scientific Works of Hans Christian Ørsted*, K. Jelved, A.D. Jackson, and O. Knudsen, translators, Princeton Univ. Press, 1998, pp. 421–445.
4. A.M. Turing, "On Computable Numbers, with an Application to the Entscheidungsproblem," *Proc. London Math. Soc.*, series 2, vol. 42, 1937, pp. 230–265.
5. J. von Neumann, *First Draft of a Report on the EDVAC*, Moore School of Electrical Eng., Univ. of Pennsylvania, 30 June 1945; www.wiley.com/legacy/wileychi/wang_archi/supp/appendix_a.pdf.
6. G. Dyson, *Turing's Cathedral*, Vintage Books, 2012.
7. C.C. Stacy, "Re: Push and Pop First Used When?," email, 3 Mar. 2003; neil.franklin.ch/Usenet/alt.folklore.computers/20030302_Push_and_pop_first_used_when.
8. D.J. Wheeler, "The Use of Subroutines in Programmes," *Proc. 1952 ACM Nat'l Meeting*, 1952, p. 235.
9. E.W. Dijkstra, "Communication with an Automatic Computer," PhD thesis, Univ. of Amsterdam, 28 Oct. 1959, p. 49.
10. W.L. van der Poel, "Dead Programmes for a Magnetic Drum Automatic Computer," *Applied Scientific Research, Section A*, vol. 3, no. 1, 1954, pp. 190–198.

GERARD J. HOLZMANN works on developing stronger methods for the design and analysis of safety-critical software as a consultant and researcher at Nimble Research. Contact him at gholzmann@acm.org.