LINUX VOICE
TUTORIAL

KONRAD ZUSE: (NEARLY) THE GERMAN TURING

JULIET KEMP

Try a programming language designed amid the rubble of post-war Germany before there were any computers on which to run it.

WHY DO THIS?

- Discover an under-appreciated pioneer of computer science.
- Plan your next trip to the technical museums of Germany.

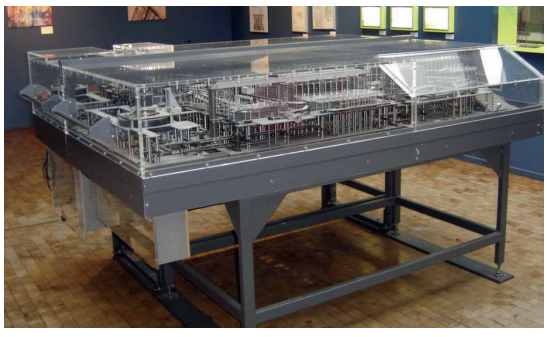
If you have any interest in computer history, and possibly even if you haven't, you'll have heard of two of the early computer pioneers: Alan Turing and John von Neumann, who were involved with the machines being developed during World War II. But there's a fair chance that you haven't heard of Konrad Zuse, in Germany — despite the fact that he was achieving very similar things over four years earlier.

Unlike both Turing and von Neumann, Zuse was working in isolation — he had no similarly able colleagues in Germany, and did not of course have any contact with the leading computer scientists and mathematicians working for the Allies. Nevertheless, in the Z3 he built the world's first fully operational electromagnetic programmable computer, in 1941; and came up with the theory of stored-program computation in 1937, several years before von Neumann proposed it.

Konrad Zuse in 1992 (he died in 1995). Photo: CC-SA, Wolfgang Hunscher, Dortmund.

Z1 and Z2

In 1935, a young Zuse was working as a design engineer at an aircraft factory near Berlin. Much of his time was spent in doing large numbers of calculations by hand, and Zuse, understandably, found this massively tedious. He began to wonder whether he could construct a machine to calculate for him. Working in his parents' flat, he began building the Z1 in 1936, from bits of metal plate and pins. The Z1 wasn't a computer, but a floating-point binary mechanical calculator. It had some programming capacity, and read instructions from holes punched in 35mm film. Zuse filed two patents in 1937, which most importantly included the idea of stored-program computation and what has become referred to as "von Neumann architecture", years before von Neumann himself proposed it. The Z1 was finished in 1938, but it never worked particularly well, as its 30,000 metal parts were not precise enough. It was destroyed in an air raid in 1944, although a replica is now in the German Museum of Technology in Berlin.



Replica of the Z1 in the German Museum of Technology in Berlin. Image: CC-SA, ComputerGeek.

Zuse's next attempt was the Z2, which he built in 1939–40. He had been called into military service, and so had a research subsidy, but initially at least was still working in his parents' flat. The Z2 took up several rooms of the flat when he presented it to the Deutsche Versuchsanstalt für Luftfahrt (DVL, the German Research Institute for Aviation), which rather makes you wonder how big the flat was and how tolerant Zuse's parents were! T

The Z2 was basically an improved version of the Z1, but using 600 telephone relays rather than the metal plates of the Z1. It had a 64-word mechanical memory, and electrical relay circuits for the arithmetic and control logic. It weighed 300kg. It worked better than the Z1, but was still very unreliable — though it worked well for the presentation to the DVL and impressed them enough that they coughed up further funding.

Z3

In 1941, with subsidies from the DVL, Zuse was able to start a company and (finally!) hire a lab to work on his next machine, the Z3. This was a programmable calculator with a memory, which had loops but no conditional jumps (so no if/then logic). Like the Z2, it was relay-based, using 2,000 relays and 22-bit words,

but it was far more reliable. Zuse's co-worker Helmut Schreyer had suggested vacuum tubes to Zuse, as were used in Colossus in 1943, but he dismissed them as a crazy idea. (IBM's Harvard Mark II, built in 1947, used relays, so they were by no means obsolete.) As with the previous machines, the Z3 used punched film for code and data input; it also had a terminal and lamps for input and output. It was Turing-complete (see boxout, right), and as such was the world's first fully operational electromechanical computer. However, Turing-completeness was not of interest to Zuse or his backers the DVL, who were interested only in automating calculations. (It was a similar story with the ENIAC in the US, which was originally intended to calculate artillery firing tables; but the wider possibilities were quickly realised by US mathematicians and scientists. ENIAC wasn't ready until 1945, though, several years after the Z3.)

Like the Atanasoft-Berry Computer in the US (tested in 1942, but not programmable, being designed to solve linear equations), but unlike ENIAC and IBM's early machines (which were decimal), the Z3 was binary. The punched tape system was also ahead of other early computers — Colossus and ENIAC were both programmed with plugs and switches. It was an eminently practical machine, for the time, thanks undoubtedly to Zuse's engineering background. His main aim was to automate engineering calculations, and the Z3 did this admirably. Its primary use at the DVL was analysing wing flutter (vibration in certain flying conditions, which can damage or destroy aircraft). Zuse did ask for funding to replace the relays with electronic switches, but this was considered "not war-important" and denied.

Meanwhile, Zuse was also working on the S1 and S2, which were special-purpose computing machines to calculate corrections to the wings of radio-controlled flying bombs – the precursors to the modern cruise missile.

Z4 and afterwards

The Z3, along with Zuse's workshop, was destroyed in an air raid in 1943, but the successor Z4 (also relay-based) was in a different workshop, and was not affected. It was eventually packed up and moved, half-finished, to Berlin in February 1945, then evacuated to Göttingen where it was completed, after which it was moved again to Bad Hindelang in Bavaria, near the Austrian border, where it was hidden in a shed to avoid its capture by the Allies.

For the next couple of years, Zuse's priority was survival — he sold woodcuts to farmers and US troops to earn money. He began working on the Z4 again in 1948, but electricity was only intermittently available and there was only rarely enough of it to run the Z4. A visit from Prof Stiefel from Zurich led to the Z4 eventually being delivered to the Swiss Federal Institute of Technology in Zurich in July 1950. At the time, it was the only working computer in continental Europe. Zuse formed the company Zuse KG, which went on to build a further 250 computers before being sold to Siemens in 1967.

IBM bought an option on his patents in 1946 (Zuse, it seems, might have preferred to work for them directly, but they weren't interested).
The exact influence of this on IBM's work is unknown, but it is possible that information from Zuse's binary machines were part of IBM's move from decimal and analog to binary and digital computers.

Turing-completeness

A Turing-complete machine is one that can simulate any single-taped Turing machine. In practice this basically means that it can (in theory and approximately) simulate any other general-purpose computer; so it can do anything you expect a "computer" to be able to do. It might, however, take a very long time!

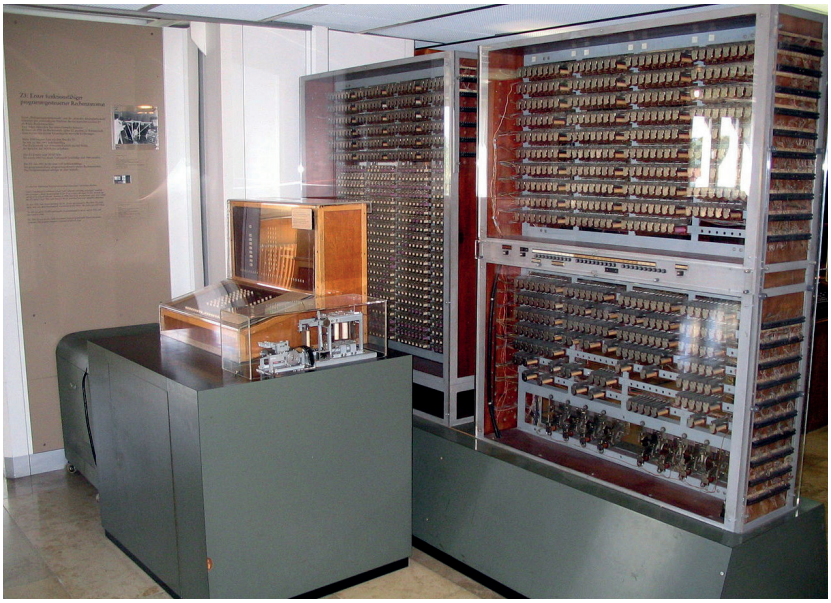
Since the Z3 had no conditional branching (if/then), it is not straightforwardly obvious that it is Turing-complete. In 1998 Raúl Rojas proved that it was, by proposing a program that instead of branching, would compute both sides of every branch. It would therefore calculate all possibilities, and cancel out the unnecessary ones. In an abstract theoretical sense, then, the Z3 was Turing-complete. In practice, this doesn't mean that it was in any real sense the same as a modern computer, or even a 1940s/50s computer with branching capability. However, the Z4 did have conditional branching.

Simulators

There's a really nice Z3 simulator available online (note that the site is in German). It runs there as an in-browser applet, which I wasn't able to get running on my Linux browser. (I could run it on Mac, which is usually pickier about Java, so the applet definitely does work; a different hardware and software setup may be all that's needed.) Alternatively, I was able to run it on Linux by downloading the file **Z3.zip** from the simulation overview page, unzipping it, and running **appletviewer simulation.html** from the resulting folder. I couldn't initially see the film tape part of the main window, but it did reappear after I resized the window, choose Programm > Neu, and hit Ende. (And in fact you can program the simulation without seeing the film tape, although it's nice to see your instructions appear!). If you only have a big purple box in the middle of the top and no 'film' picture, this is the 'Speicherauswahl' box referred to below. Enter your memory locations in here and use the right-hand buttons for operations just as detailed below.

The Z3 is labelled in German. Some of the labels are immediately obvious, but here's a quick translation of some of the others:

German	English
Vorzeichen	**Sign (positive/negative)**
Ziffern	**Numbers**
Komma	**Comma**
Wurzel	**Root**
Einlesen	**Read in**
Ausgeben	**Output**
Eingabe	**Input**
Mantisse	**Mantissa (significand)**
Speicher	**Memory**
Rücksetzen	**Reset**
Fortsetzen	**Resume/continue**



Replica of the Z3 at the German Museum in Munich. Image: CC-SA, Venusianer.

The registers R1 and R2 are the working registers, and the memory (Speicher) has 64 words available. You can set this manually by clicking the circles. I found the mantissa/exponent setup a little confusing but each line has a decimal translation at the end so you can play around until you have the idea.

The mantissa (or significand)/exponent is a way of describing floating point numbers. For example, a significand of 1234 and an exponent of -1 would describe the decimal number 123.4.

You can either enter a calculation directly, using it in effect as a desk-top calculator, or enter a program. (Sadly, you can't save programs.) As with other computers of a similar age, to run a calculation, you first enter two numbers. These will be loaded into the two working registers R1 and R2; the next instruction is then applied to those registers, and the output stored in R1, ready for the next calculation.

Here's an example of manually adding two numbers, 11 and 2:

- Enter 11 with the top set of buttons (Eingabe).
- Hit the Einlesen button, and watch the circuitry change. Notice that the R1 circle on the bottom left will now be lit.
- Enter 2 with the top set of buttons (Eingabe).
- Hit the Einlesen button again. Both R1 and R2 are now lit.
- Hit Addition, then Ausgeben. The output, 13.0, will appear in the very bottom left.

Zuse and Turing

Zuse and Turing may have met briefly after the war, in 1947, at a colloquium in Göttingen which included a few other British and German researchers. ('Colloquium' is a polite way of describing a discussion which has also been described as "an interrogation". The participation of the German scientists was almost certainly not optional.) However, this meeting is only described in Heinz Billing's memoirs, and no details survive. The historical detail is discussed in a paper by Herbert Bruderer. If Zuse and Turing did meet it is likely, due to the secrecy of the war and post-war period, that neither of them was familiar with the achievements of the other, which seems more than a little sad.

You can also use the 'film' to enter a program. When you first load the applet, there's a program provided on the film. To run this, go to the applet's Programm menu and choose Start. To enter your own new program, go to Programm > Neu (new). You'll then get an extra three buttons: Laden (load/read from memory); Speichern (store to memory); and Ende (end). You use the purple Speicherauswahl box to enter a memory location, and the buttons to enter an operation code (such as addition, multiplication, read from storage, etc).

Here's how to enter a program to add two numbers:

- Enter 0 in the purple Speicherauswahl box, and hit Laden. This reads from memory location 0.
- Enter 1 in the Speicherauswahl box, and hit Laden, to read from memory location 1.
- Hit Addition. This will add the last two numbers that were read in.
- Hit Ausgeben. This will output the result.
- Hit Ende to finish the program.
- Go to the Speicher window and enter a number in the 0 location and in the 1 location.
- Choose Start from the Programm menu. Your program will run, and you'll see the result (the sum of your two numbers) at the bottom-left.
- To start again, you'll need to hit Fortsetzen (Reset).
- To store the result in a specific memory location, say location 6, you can replace the Ausgeben instruction with

Speicherauswahl 6, Speichern.

Run this (you'll have to re-enter the whole thing), and keep an eye on the Speicher window. You'll see your result show up in memory location 6.

Here's a program to calculate 4! (4*3*2*1):

- In the Speicher box, enter values 1, 2, 3, 4 in memory locations 0, 1, 2, 3.
- In the main window, start a new program.
- Speicherauswahl 0, Laden.
- Speicherauswahl 1, Laden.
- Multiplikation.
- Speicherauswahl 2, Laden.
- Multiplikation.
- Speicherauswahl 3, Laden.
- Multiplikation.
- Ausgeben.
- Ende.

Run the program to get the output 24. Note that multiplication steps take a while! You'll see here the advantage of having the output of each calculation stored in R1 ready to be used. By rewriting memory addresses it should be possible to construct a loop; have a go and see what you can manage.

If you want more information about the simulation, there is an article by Raúl Rojas which discusses the construction of the simulation and includes the instruction set. There are also instructions for using the simulator (in German, but Google Translate does a reasonable enough job) on the Zuse project webpage.

While building the Z4, Zuse concluded that an alternative was needed to programming in machine

code, to make programming more straightforward. In 1945/6, when he was living in the rural Allgäu and couldn't work on hardware, he designed Plankalkül ("Plan Calculus"), which was the first high-level programming language. However, this only existed in theoretical form during his lifetime; a team finally implemented a compiler in the year 2000, five years after his death. Plankalkül has been compared to APL and relational algebra, but it did not in practice have an impact on future languages, since it wasn't implemented at the time. It is, however, the first theoretical description of high-level programming.

Programming in Plankalkül

Zuse's original notation for Plankalkül was two-dimensional, although a linear notation was devised when implementing it in the 1990s. The full report from the Free University of Berlin team is a fascinating read, but here are a few of the basics:

There are three basic types of variables:

- V variables (V0, V1…), read-only, used to pass parameters into programs.
- Z variables (Z0, Z1…), read/write, used for intermediate results.
- R variables (R0, R1…), write-only, used to pass the final results of a program.

Loop variables are also used, written i0, i1, i2, etc.

Variables have one of the following types:

- One bit, written 0.
- n bits, written n.0.

Tuples of other types, written (n.0, m.0, …). So (3.0, 4.0) would be a tuple with two members, one 3-bit variable and one 4-bit variable. Tuples can have two or more elements.

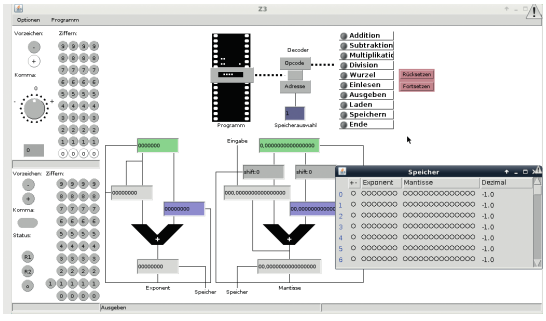
Vectors of a single type: so m.n.0 is a vector (or array) with **m** members each of which has **n** bits. Vectors are used for arrays of the same type, tuples for arrays of different types.

Here's a quick example that adds two numbers:

```
P1 (V0[:8.0], V1[:8.0]) => R0[:8.0]
  V0[:8.0] + V1[:8.0] => R0[:8.0]
END
```

Note that the report would have **R(V0[:8.0]…)** in that first line, but the online compiler at the Zuse Project website doesn't like that.

After Zuse KG was bought, Zuse wrote the book *Calculating Space*, in which he suggested that the universe itself is running on a cellular automaton. (Von Neumann had an interest in cellular automata, too.) There's no physical evidence against this thesis, and other scientists have expanded on it since. After retirement, he spent his time painting; he died in 1995.



The Z3 window and memory window, in the middle of entering a program.

Z4 (the real thing!) on display in the German Museum in Munich. Image: Clemens Pfeiffer, CC-G.

Looking at Zuse's history, it's hard not to make comparisons with Turing, von Neumann, or Hopper, working at the same time in other countries; and to wonder what might have happened if Zuse had been taken more seriously in his own country. Or, more cheerfully, if all of them had been truly able to collaborate in a peaceful world across international boundaries. What would programming languages look like today if Plankalkül had been implemented before COBOL? Would things have moved faster if the Z3 hadn't been destroyed (or if Colossus, in the UK, had been an open project)? Or, on the other hand, did the war drive developments that would otherwise have been much slower? The ethics on all sides are difficult, too; all the pioneers of this time were working on war projects. Zuse, while he was working for the Nazi regime, was never a member of the Nazi Party (unlike many other German scientists of the time). In later life he suggested that scientists and engineers usually have to choose between working for questionable interests (commercial or military), or not working at all.

What is clear is that Zuse was working at the very top of his field, even if he wasn't able to work alongside the others doing the same. His machines were at least two–three years ahead of the teams in the UK and US. Although the Z4, his 'final' version, was finished at roughly the same time as ENIAC and a little after Colossus, it was more programmable than both and genuinely general-purpose. Zuse was an immensely talented scientist whose contribution to computing has gone unfairly unnoticed.

Juliet Kemp is a programming polyglot, and the author of O'Reilly's *Linux System Administration Recipes*.