The Architecture of Konrad Zuse's Z4 Computer

Raúl Rojas
Freie Universität Berlin
Berlin, Germany
rojas@inf.fu-berlin.de

2021 7th IEEE History of Electrotechnology Conference (HISTELCON) | 978-1-6654-7847-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/HISTELCON52394.2021.9787324

***Abstract*—This paper describes the programming architecture of Konrad Zuse's Z4 computer. The machine's logic was implemented with telephone relays while the memory was a mechanical attachment. It was the successor to the Z3 machine completed in 1941. The Z4 was assembled over the next four years, until German capitulation. In its first embodiment, the machine featured 64 words of mechanical memory, two CPU registers, one punched tape reader and one tape puncher. The keyboard accepted decimal input, but the internal numerical representation was fully binary, based on a particular floating-point format. The computer had a relatively large instruction set of arithmetical operations. In 1949, the Swiss Federal Institute of Technology in Zürich decided to lease-buy the Z4 from Zuse's fledgling computer company. An additional punched tape unit for reading auxiliary programs or number tables was added, and the instructions necessary for calling subprograms, as well as the conditional jump, were included in the instruction set.**

Keywords—Konrad Zuse, Versuchsmaschine 4, computer architecture

I. Introduction

The Z4 was a computer designed and built by Konrad Zuse between 1941 and 1945 (the original name was "Versuchsmaschine 4", or V4). It represents a milestone in the history of computing in Germany. This machine was the culmination of the chain of innovations launched by the German inventor with the Z1, a mechanical computer completed in 1938 in Berlin. The further embodiment of the same general architecture was the Z3, a machine built from telephone relays and demonstrated in 1941. The completed Z4 was the commercial computer Zuse had been struggling to build for so many years. The Z1, Z3 and Z4 shared a few fundamental principles that we recognize today as constitutive of modern computers: all ofthem had a processor distinct from memory, their design was fully binary with decimal input, computations were performed using floating-point hardware, registers were available in the CPU, and all operations were microprogrammed. However, the programs were held in an external medium, i.e., a punched tape. This control tape could be made to loop just by gluing its two ends. The Z4 was Konrad Zuse's vision finally materialized: it took him nine years to arrive at this concluding iteration. He had started working long before the war, in 1936, but the bulk of the development effort for the Z4 was done while WWII was raging in Europe [1].

The story of the Z1 and Z3 machines has been well documented [2, 3, 4]. The Z1 was built between 1936 and 1938 using mechanical logic gates. They consisted of rods and plates whose movement or non-movement could be interpreted as a binary 1 or a 0 [5, 6]. But the mechanical components of the Z1 were not reliable enough. Therefore, Zuse decided to use telephone relays for the Z3 instead [7]. He built a small "proof of concept" prototype of the processor, called the Z2, a machine which was never meant to be a complete computer. After this experiment, he started building the Z3, which was like the Z1, but worked with electricity. In this paper, I refer the reader to the studies cited above, which explain in detail the floating-point circuits used by Zuse, in part also for the Z4. Here, I will describe mainly the architecture of the Z4 from the point of view of the programmer. This is what is sometimes called the "functional" or "programming" architecture of a computer. Fig. 1 shows the Z4 as it stands today in the history of computing hall of Deutsches Museum in Munich. The console is in the front and the circuits built from relays are in the back.

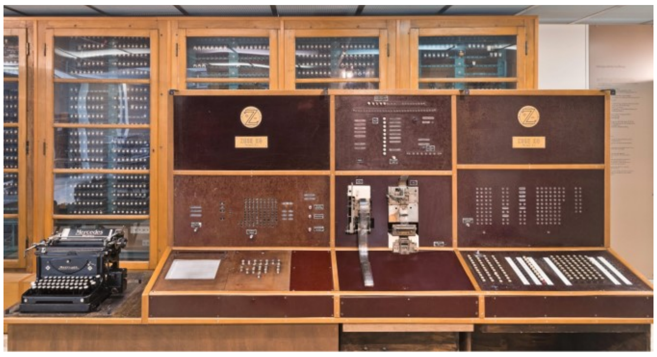
Fig. 1 The Z4 at Deutsches Museum. The console is visible in the foreground. In the background, we can see the racks of telephone relays used for the logic components. The two tape readers are visible in the center of the console. The right side of the console is used for entering instructions and addresses. The left side is used for entering decimal numbers, which are displayed in a lamp array similar to the keyboard of a vintage cash register. Results could be printed with the electric typewriter on the left.

Konrad Zuse's relatives and a Berlin instrument maker financed the construction of the Z1. The Z3 was built while Zuse was working part time for the Henschel Werke developing airplanes and flying bombs during WWII. Zuse dedicated the bulk of his working time to his own company, which was classified as necessary for the war effort. After the successful demonstration of the Z3, Zuse obtained a contract from an aerodynamics institute (Deutsche Versuchsanstalt für Luftfahrt) for the development of a more ambitious machine, that is, the Z4, which was assembled in Zuse's workshop [2]. A few weeks before the Russian Army occupied Berlin, the Z4 was transported to Bavaria, where it remained until 1949.

II. Block Architecture

It is easier to describe the architecture of the Z4 from the point of view of a programmer by referring to a block diagram containing the essential components (Fig. 2, based on [8]).

Programs for the Z4 were encoded in punched tapes. The main reader, At0, could read one instruction at a time and advance the tape. The control unit transformed each instruction into a sequence of microinstructions for the central processing unit. The processor contained two registers (OR-I and OR-II, also called register x and register y, respectively). Data read from memory was loaded to these registers and then operations requiring two arguments, such as addition or multiplication, were executed with their contents. The result was always rewritten into the first register (OR-I). There was an instruction for storing the contents of OR-I to a specified memory address. The machine had 64 memory words, with addresses 0 to 63.

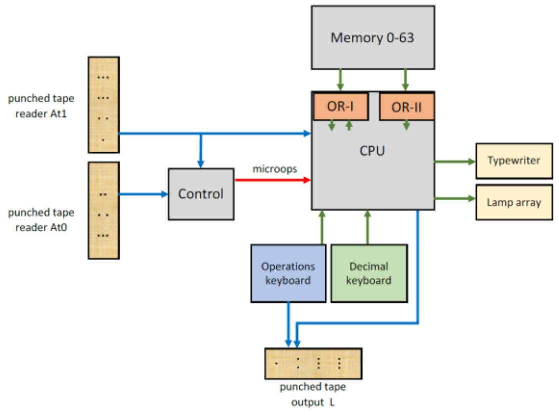

Fig. 2 The block architecture of the Z4 with its main components, as explained in the text. The diagram is based on [8].

The processor of the Z4 computed all arithmetical operations using floating-point. The format in memory was similar to alternatives used today. The Z4 used seven bits of each memory word for encoding the exponent (in two's complement representation), 23 bits for encoding the mantissa, and one bit for encoding the sign of the number [8]. An additional bit was used to flag "special values", such as infinity and "indefinite", or "not a number", as we would say today (NaN). Therefore, each word of memory consisted of 32 bits.

The Z4 could be used as a kind of manually triggered calculator: the operator could enter decimal numbers through the decimal keyboard, these were transformed into the floatingpoint representation of the Z4, and were loaded to the CPU registers, first to OR-I, then to ORII. Then, it was possible to start an operation using the "operations keyboard" (an addition, for example). The result was held in OR-I and the user could continue loading numbers and computing. The result in OR-I could be made visible in decimal notation by transferring it to a decimal lamp array (at the push of a button). It could also be printed using an electric typewriter.

The operator could also use the instructions keyboard to punch a program directly to a tape. Electronics in the console translated keypunches into the appropriate binary code for each instruction. This procedure semi-automated the creation of new programs, which for the Z3 still had to be manually encoded by the programmer in binary notation. The CPU could also control the tape puncher directly. It was then possible to store the binary representation of tables of numbers in punched tapes. The table could be reused later, using the secondary tape reader. It was even possible to compute the binary code for program instructions and punch them in a tape (the program doing this was then a "superprogram"; today it would be a kind of compiler).

The secondary tape reader, At1, could be used to execute a subprogram. While At0 was reading instructions from the main program, control could be transferred to At1 by the control unit. The operator would have previously loaded a tape from a library of subprograms in At1. When control returned to At0, a new tape could be loaded in At1, in case a new subprogram would be needed afterwards. At1 could also be loaded with a table of numbers produced by the tape puncher, and when one of those numbers was needed during execution, the appropriate instruction could load the next number available at At1 to one of the registers.

The Z4 could only use absolute addresses (there was no relative addressing). Therefore, even when it wa feasible to simulate a programming a loop through the simple expedient of gluing the ends of a tape, it was not possible to make an indexing variable point sequentially to a range of addresses, with the purpose of reading them one by one. But in that case, the tape reader At1 could be used as a kind of substitute, since the tape advanced each time a number was read from it.

If I had to summarize the Z4 (of 1950) for an audience of modern computer practitioners in just a few words, I would say that the Z4 was a programmable machine featuring 64 words of memory. It had a floating-point CPU with two registers, and used two punched tape readers to read programming code, one of which could be also used as an external numerical memory or for library tapes. The bulk of the instruction set was dedicated to arithmetical operations, but subroutines could be called (one level deep) and there was a conditional jump instruction. The output of the machine could be visualized in the console using lamps, or printed.

III. Architectural Details

There are a few idiosyncrasies of the Z4 that it is convenient to explain at this point. The first surprise is that while the control part of the Z4 was designed using telephone relays, the memory was a mechanical apparatus. Zuse was a master of mechanical design and his first computer was built using entirely mechanical components. For the Z4, Zuse returned to a mechanical memory because telephone relays were expensive and bulky at that time. He reckoned that he could build the mechanical components necessary for storing memory bits more economically and needing much less volume using his so-called "mechanical relays". His original intention was to build a mechanical memory of up to one thousand words in a small volume. The first iteration of the Z4 contained only one bank of memory of 64 words. But it required much less space than the equivalent memory built with relays (as had been done for the Z3). We don't know how much calculating speed was lost by reading from a mechanical instead of a relay-memory. However, such loss was lessened by making the memory work in parallel with the processor. It must be said, though, that the mechanical memory proved to be reliable enough during the many years the Swiss Federal Institute of Technology (ETH) operated the machine in Zürich, although it had been the main concern when the machine was leased [9]. However, assiduous users of the machine complained about occasional problems when parts of the mechanical memory jammed [10].

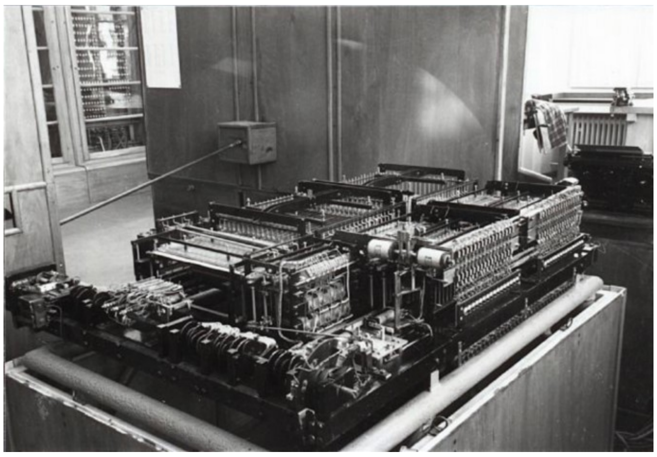

Fig. 3 Picture of the mechanical memory of the Z4 at the ETH in Zurich

As explained before, the floating-point format used by Zuse reserved 7 bits for the exponent, in two's complement representation, so that the range of possible binary exponents ran from -64 to +63. The mantissa was stored using normalized floating-point, where the leading bit before the binary point is always one. In total, 23 bits were needed to store the mantissa in memory, but the representation was expanded to 24 bits in the processor (supplying the leading one). One bit was used for the sign of the number, and another to signal the presence of a special value, so that a total of 32 bits were necessary to store a number in memory. There was a special coding for zero (which cannot be represented as a normalized floating-point number), and also for infinity and not a number (NaN) which was called an "indefinite value" and was represented symbolically by Zuse as "?". Dividing zero by zero, for example, could produce a "?". The special coding for zero, infinity and NaN is not documented in the programming manual. We only know about the additional bit that was used to differentiate between normal numbers and special values [8].

Entering a decimal number through the decimal keyboard required pushing one of ten buttons for each specific decimal digit, one by one, and then the exponent. The result was visualized with an array of nine columns of lamps. Every column had a lamp for the digits zero to nine. The specific decimal digit at each decimal position, for every one of the nine columns, was lit and the operator could write down the result produced by the machine. That lamp array can be seen on the upper left of the console in Fig. 1. The nine full columns for decimal digits are evident. The position of the decimal point in the mantissa was indicated with an additional row of lamps under every column (only one "decimal point" lamp would be switched on). The exponent of the result was shown with additional lamps. There were also lamps for the sign, and to indicate underflow, overflow (infinity) or not a number, indicated by a lamp with a question mark.

IV. THE ARITHMETICAL INSTRUCTION SET

The instructions for the Z4 were implemented using microoperations, as in the Z3. For every operation there was a control sequencer, which was just a circular mechanical stepper advancing from one position to the next, like a clock. At every microstep, different circuits of the processor were activated and this produced the information flow in the Z4. Such a microcoded architecture was first developed for the Z1, and rotary microsteppers were used for the Z3 [7] and Z4. Microcoding made Zuse's machines very flexible. New operations could be created by simply installing a new stepper.

Every program (punched tape) for the Z4 started with the instruction "St". The end of the code was signaled with the instruction "Fin". The Z4 advanced a new tape until the first "St" appeared.

The Z4 performed arithmetical operations with one or two arguments. In what follows, the first register (OR-I) is abbreviated as "x" and the second as "y". The operations with a single argument included nine multiplications by constants (the divisions were transformed into multiplications). The binary representation of the constants was hardwired in the machine:

$-x$	$2x$	$10x$	πx	$x/2$	$x/3$	$x/5$	$x/7$	x/π

Additional one single argument operations were:

$x2$	$\sqrt{x}$	$\|x\|$	$\mathrm{sgn}(x)$	$\max(0, x)$

Two argument operations were:

$x + y$	$x - y$	$y - x$	$x \times y$	x/y	$\max(x, y)$	$\min(x, y)$

The instructions for reading and writing to memory were "A n" and "S n", where n represents the memory address. The first A-instruction would load register 1, while the next A-instruction would load register 2. It was also possible to read a number from the tape reader using the command $\nearrow m$, where m is a number which has been stored in the punched tape immediately after the arrow-up command.

When the Z4 was running, it could request decimal input from the operator. The instruction to request manual input was $\nearrow$. The result contained in register x could be shown activating the console decimal lamps using the instruction $\searrow$, or it could be printed using the instruction D, immediately after $\searrow$.

There were a few additional special commands used for formatting the output, or for making possible certain combinations of otherwise prohibited instruction sequences. For example, due to the slow mechanical memory, the processor would be too fast. It was not possible to store a result to an address and read the address immediately. The programmer had to be careful and wait a certain number of instructions for the memory to be addressable again. This confirms that access to memory was partly asynchronous, relative to the processor, in order not to slow the latter down. Prohibited sequences of operations, in terms of timing, made the machine stop during execution [9]. An expert programmer reviewed the code of the users to make sure that forbidden combinations of instructions were not present.

And that's it. This was the instruction set used by the Z4 until 1945 [11]. It was effectively a superset of the instruction set of the Z3 and the main missing ingredient is, of course, conditional branching and being able to call subroutines. In fact, the tape reader At1 in Fig. 1 was included after 1945, and the instruction set was extended to deal with conditionals.

Fig. 4 shows a diagram produced by Zuse's company before the Z4 was leased to the ETH in Zurich. The mechanical memory has the label 10. There is only one tape reader (8) and the tape puncher (7). The processor logic was housed in the relay casings 1 to 4. There is no electrical typewriter. A planar array of lamps (5b) organized in rows and columns could be used to light up a specific lamp under a sheet of paper, signaling to the operator the name of the variable that had to be entered if the machine stopped for manual input. The programmer had to take care of switching on the correct lamp, which was reached by going down in the rows and to the right, using special instructions embedded in the code stream. This was the state of development of the Z4 until 1945 and before the ETH became interested in the machine.

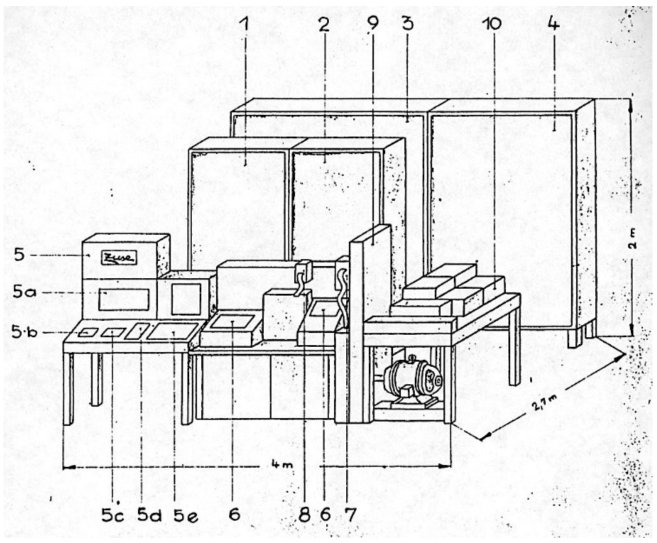

Fig. 4 The Z4 after 1945 und until 1949 [12].

V. Conditionals and control transfer

Konrad Zuse explains in his memoirs that the Z4 was transported out of Berlin a few days before the downfall of the city. The machine spent several years in a cellar in Bavaria. The Swiss mathematician Stifel heard about the computer, visited Zuse in 1949, and was able to see the Z4 working properly. His university, the ETH, decided to lease the machine, with an option to buy it at the end of five years, but important modifications had to be made beforehand, the main one being the inclusion of conditional branching. Zuse complied, and in 1950 the Z4 was delivered to the ETH [13, 10]. It was the first commercial computer rented or sold by any company in continental Europe.

The conditional jump promised to Stifel was implemented by Zuse using the new instruction "Sp" (for "Sprung", in German). If the contents of register x is +1, then the punched tape is rolled down until a new start instruction is found in the tape (that is, "St"). Execution continues normally from that point on.

Before a conditional instruction could be executed, it was necessary to compute a logical result in register x. For this purpose, there were five arithmetical test operations. The operations prove if a condition is fulfilled. The test "x =? ", for example, verifies whether the result of the previous operation was a NaN or not. A successful test fills the register x with +1, otherwise with −1. The logical-test operations were:

$x = 0$	$x \geq 0$	$\lvert x\rvert = \infty$	$x =?$	$\lvert x\rvert >= 1$

The instruction "Up" was used for transferring control to subroutines. Execution continued at the current position of the punched tape in At1 and control was returned to At0 when the instruction "Fin" was reached in the subprogram. There was a conditional variation of Up and Fin which is less relevant to the discussion here (see [14] for a full discussion of subroutine transfer and conditional jumps in the Z4).

Fig. 3 shows the lamps in the console for the complete instruction set.

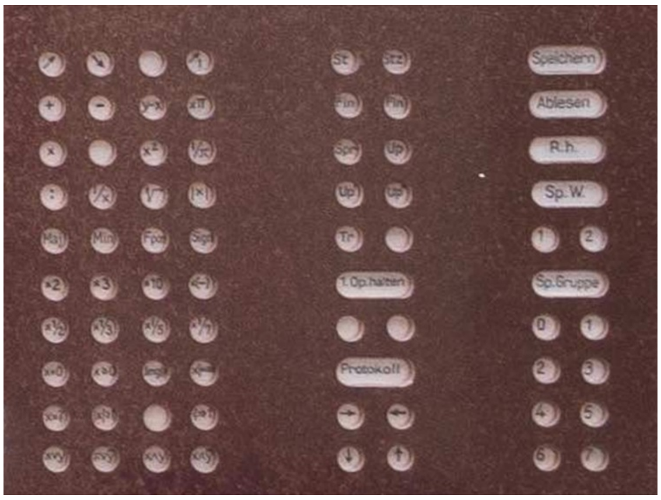

Fig. 5 The complete instruction set in the console lamps of the Z4. The last row of logical operations to the left is not covered in the programming handbook.

After the new conditional and control transfer instructions were introduced, it was possible to call subroutines. The additional tape reader also allowed the programmer to use the secondary tape for reading numbers punched by another program.

However, there is a problem with this implementation of the conditional jump. The program can only jump down in the code. But programming loops require reusing previous code, so that the jump has to be taken upwards in the instruction sequence. The solution in the Z4 was to glue the punched tape, making the code execute in a cycle. The programming manual of the Z4 explains how to glue the tape and also that a minimum length of 50 cm is required so that the tape does not jam. If the loop was shorter, several copies of the code had to be punched one after the other until the minimum tape length was attained. This is called "loop unrolling" and it was used in the Z4 so that the punched tape could meet the minimum specified length.

Zuse's way of implementing the conditional jump is unsatisfactory, because either the programmer resigns to having a single loop in the code, or several possible tricks have to be applied [15]. One trick would be to have the body of the loop as external code in At1 that can be called a fixed number of times in the main program. The other would be to assign a sequential number to all loops in a program and arrange them into a single loop of tape. Then, during execution, we would only enter in the loop guarded by a conditional comparison placed before the loop body. This is cumbersome and it would be interesting if actual code of the Z4 users could be found to see how they solved this problem. Notice that Charles Babbage, who used punched cards strung together, designed the conditional jump to go up into the stream of cards precisely so that loops could be implemented easily [16].

Since the Z4 did not have indirect addressing capabilities, notice that loops that need to address memory sequentially are difficult to implement. Adding 20 numbers in memory, for example, would require specifying the 20 addressing instructions with the consecutive absolute addresses. The alternative would be to use the tape reader At1 so that the numbers can be read one by one from that reader without having to specify absolute addresses. The sequencing of the data is then done automatically when the reader advances. Charles Babbage had the same difficulty with the Analytical Engine, but he envisioned making the stream of "number cards" bidirectionally steerable, so that interesting combinations of data input could be achieved when running loops [17].

VI. Conclusions

From a contemporary point of view, the architecture of the Z4 can be readily explained using modern terminology. And that is actually the biggest surprise when thinking about this machine and comparing it with the computers designed up to 1945.

The first comparison that comes to mind is with the Analytical Engine. Zuse's Z1 was actually something like Babbage's dream materialized, in the sense that all important arithmetical operations were implemented using only mechanical means. The AE went further, though, since it included conditional instructions, so important for universal computation, at least on paper. Curiously, Zuse did not include the conditional jump as a programming instruction in the Z1 or Z3, or in the Z4 (until 1945). Zuse referred to the programs that could be written in this way as "rigid" because he was fully aware that conditional instructions could make programming "flexible". In fact, the microcode in all his machines was based on conditional execution. Not including it for the high-level programming was the main shortcoming of the Z1, Z3 and Z4, which could not be fully avoided using the glued-tape approach (although there are ways of programming a universal computer with a single loop, [15]).

The Z4 was a floating-point machine. Neither the Harvard Mark I, nor the ENIAC used floating point. Both of them used a fixed-point representation. Actually, neither the Mark I nor the ENIAC was fully binary. The internal representation for numbers was decimal, using gears in the case of the Mark I, and arrays of vacuum tubes in the case of the ENIAC. The Z1 was already completely binary when it was finished in 1938.

The separation of memory and processor is also complete and pervasive in Zuse's machines. In the Mark I and the ENIAC, memory and processor are still intermixed, since memory words are used as accumulators for arithmetical operations. Even the Analytical Engine is superior in that respect, since the storage was completely separated from the mill, and they even ran independently, each one using its own set of punched cards.

The ENIAC did not have a conditional jump until a trick was found so that data cables could be used to provide start pulses for accumulators. This required hardwiring the machine appropriately. Curiously, the ENIAC did not execute code. The code was embedded in the way the machine was hardwired, and the connections had to be rearranged for each new problem.

As we can see, all of these machines brought something new in terms of the computing architectures that would become possible in later years. At some point, all of them have been called the "first computer". However, I think that a comparison of their architectures confirms that we can only talk about the "first computers", in plural, since the dawn of the third industrial revolution was an endeavor that went beyond national boundaries. The start of the computer age was a collective enterprise whose first creative spark actually flashed during the heyday of the first industrial revolution with the inception of the Analytical Engine.

References

[1] K. Zuse, Der Computer - Mein Lebenswerk, Springer-Verlag, Berlin, 1984.G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*

[2] H. Petzold, Moderne Rechenkünstler, C.H. Beck Verlag, 1992.

[3] R. Rojas, "How to make Konrad Zuse's Z3 a universal computer", IEEE Annals of the History of Computing, Vol. 20, N. 3, pp. 51-54, 1998.

[4] R. Rojas, J. Röder, H. Nguyen, "Die Prozessorarchitektur der Rechenmaschine Z1", Informatik-Spektrum, Vol. 37, N. 4, Springer-Verlag, June 2014.

[5] R. Rojas, "The Z1: Architecture and Algorithms of Konrad Zuse's First Computer", arXiv 1406.1886, June 2014.

[6] R. Rojas, "The Design Principles of Konrad Zuse's Mechanical Computers", arXiv, March 2016.

[7] R. Rojas, "Konrad Zuse's legacy: the architecture of the Z1 and Z3", Annals of the History of Computing, Vol. 19, N. 2, 1997, pp. 5-16.

[8] K. Zuse, „Bedienungsanweisung für Zuse Z4", ETH Zurich, 1952.

[9] A. Speiser, "Über Episoden aus den Anfängen der Informatik an der ETH", Informatik Spektrum, V. 31, N. 6, 2008.

[10] H. Bruderer, Konrad Zuse und die Schweiz, Technical Report, ETH Zurich, permanent link: https://doi.org/10.3929/ethz-a-006517565.

[11] K. Zuse, "Rechenpläne für das Rechengerät V4", Zuse Papers, Deutsches Museum NL 207/0230, 1945.

[12] K. Zuse, „Zuse Calculators", Zuse Papers 010/017, 1946.

[13] A. Speiser, "Konrad Zuse's Z4: Architecture, Programming, and Modifications at the ETH Zurich", in: R. Rojas, U. Hashagen (eds.), The First Computers - History and Architectures, MIT Press, Cambridge, Mass., 2000.

[14] R. Rojas, "Konrad Zuse und der bedingte Sprung", Informatik-Spektrum, Springer-Verlag, September 2013.

[15] R. Rojas (ed.), Die Rechenmaschinen von Konrad Zuse, Springer-Verlag, Berlin, 1998.

[16] Allan Bromley, "Babbage's Analytical Engine Plans 28 and 28a—The Programmer's Interface", in: Annals of the History of Computing, V. 22, N. 4, Oct. 2000.

[17] R. Rojas, "The Computer programs of Charles Babbage", accepted by IEEE Annals of the History of Computing, accepted for publication.