**~~History / Background~~**

* ~~“Zuse started developing program-controlled binary calculators in 1936. In 1941, he completed the first fully operational digital computer [the Z1].” (Giloi 11)~~
* ~~“The mechanical ALU of the Z1 had some problems: under certain conditions it would get jammed and had to be reset. While struggling to make his mechanical design work, Zuse started in 1938 to design a second machine, the Z2. The ALU of the Z2 employed telephone relays instead of mechanical switches, while the memory still consisted of mechanical flip-flops.” (Giloi 12)~~
* ~~“The Z3, the successor of the Z2 built by Zuse from 1939 to 1941, was completely a relay machine. It was actually the first fully operational, programmable digital computer in history.” (Giloi 12)~~
* ~~“In 1941 Zuse applied for a patent with 51 separate claims for all important aspects of the relay-based computer Z3.” (Giloi 14)~~
* ~~Because of the secrecy surrounding the war effort, Zuse’s application did not become public until 1951.” (Giloi 15).~~
* ~~“Zuse built electromechanical computers in Hitler’s Berlin, notably the Z1, the world’s first operational binary computer. Berlin was thriving just before and in the early years of the war, when Zuse first worked on his machines. His primary problem was that he could not make the case for financial and other support to the Nazi government, which apparently thought the war would end quickly and favorably and did not want to expend resources on Zuse’s longer-term ideas. He started building computers where he could, including his parents’ living room . . . .” (Goodman 22)~~
* ~~“However, things got really tough for Zuse when the Allies started winning, and bombing destroyed his Z3 program-controlled relay computer in 1944.” (Goodman 22)~~
* ~~“A German civil engineer, Zuse had created the world’s first functional program-controlled Turing-complete computer, the Z3, in 1941.” (Hyman 21)~~
* ~~“However, Zuse is popularly recognized in Germany as the ‘father of the computer,’ and his Z1, a programmable automaton built from 1936 to 1938, has been called the world’s ‘first programmable calculating machine.’” (Rojas, “Simulating Konrad Zuse’s Computers” 64)~~
* ~~“In 1938, Zuse started building the Z3, a machine consisting purely of electromechanical relays but with the same logical structure as the Z1. It was ready and operational in 1941, four years before the ENIAC.” (Rojas, “Simulating Konrad Zuse’s Computers” 64)~~
* ~~“From 1936 to 1938, Konrad Zuse constructed his Z1, an automaton he had designed in order to perform the time-consuming calculations necessary in his occupation as a civil engineer.” (Tarnoff 81)~~
* ~~“It [the Z1] incorporated a number of important features still used in modern computing including a floating-point, binary numbering system, an addressable memory to store data, distinct decimal input and output units, an arithmetic unit with a carry look-ahead adder, and a control unit that used a two-stage execution pipeline.” (Tarnoff 81)~~
* ~~“The machine itself [the Z1] was mechanical although Zuse later developed machines with electromechanical relays.” (Tarnoff 81)~~
* ~~“His third machine, the Z3, was identical in architecture to the Z1 except that it replaced the unreliable mechanical elements with electromechanical relays and added a square root function.” (Tarnoff 81)~~
* ~~“The Z1 used a Harvard-style architecture executing programs directly from a punch tape reader (35mm standard movie film) while it stored data in a separate memory. In the case of the Z1, storing the programs in memory was avoided because of the expense of the memory.” (Tarnoff 81)~~
* ~~“In direct contrast to these three machines [The ENIAC, Harvard Mark I, and the ABC], the Z1 was more flexible and was designed to execute a long and modifiable sequence of instructions contained on a punched tape. Zuse’s machines, the Z3 and the Z4, were not electronic and were of reduced size. Since the Z3 was completed and was successfully working prior to the Mark I, it has been called the first~~ *~~programmable~~* ~~calculating machine in the world.” (Rojas, “Konrad Zuse’s Legacy” 5)~~
* ~~“The Berlin Polytechnic student Zuse started thinking about computing machines in the 1930s. He realized that he could construct an automaton capable of executing a sequence of arithmetical operations like those needed to compute mathematical tables. Coming from a civil engineering background, he had no formal training in electronics and was not acquainted with the technology used in conventional mechanical calculators. This nominal deficit worked to his advantage, however, because he had to rethink the whole problem of arithmetic computation and thus hit on new and original solutions.” (Rojas, “Konrad Zuse’s Legacy” 5)~~
* ~~“Zuse decided to build his first experimental calculating machine exploiting two main ideas: the machine would work with binary numbers; the computing and control unit would be separated from the storage. Years before John von Neumann explained the advantages of a computer architecture in which the processor is separated from the memory, Zuse had already arrived at the same solution.” (Rojas, “Konrad Zuse’s Legacy” 5)~~
* ~~“In 1936, Zuse completed the memory of the machine he had planned. . . . It was a mechanical device but not of the usual type. Instead of using gears (as Babbage had done in the previous century), Zuse implemented logical and arithmetical operations using sliding metallic rods. The rods could move in only one of two directions (forward or backward) and therefore were appropriate for a binary machine.” (Rojas, “Konrad Zuse’s Legacy” 5-6)~~
* ~~“The processor of the Z1 was completed a few months after the storage unit, using the same kind of technology. It worked in concert with the memory but was never very reliable. The main problem was the precise synchronization that was needed in order to avoid applying excessive mechanical stress on the moving parts. It is interesting to point out that in the same year that Zuse completed the memory of the Z1, Alan Turing wrote his ground-breaking paper on computable numbers, in which he formalize the intuitive concept of computability” ((Rojas, “Konrad Zuse’s Legacy” 6)~~
* ~~“Following the advice of his friend Helmut Schreyer, he [Zuse] considered using electromechanical relays. Zuse built an ‘intermediate’ simpler model (the Z2) using a hybrid approach (a processor built out of relays and a mechanical memory). In 1938, Zuse started building the Z3, a machine consisting purely of relays but with the same logical structure as the Z1. It was ready and operational in 1941, four years before the ENIAC.” (Rojas, “Konrad Zuse’s Legacy” 6)~~
* ~~“The main architectural difference between the Z1 and Z3 was the fact that the square root operation was left out of the Z1. There were also minor differences in the number of bits used for arithmetical operations in the processor (the Z1 used one fewer bit for the mantissa of floating-point numbers) and the number of cycles needed for each instruction. With this minor caveat and taking only the architectural features into account, one can speak of the Z1 and Z3 as nearly equivalent machines.” (Rojas, “Konrad Zuse’s Legacy” 6)~~
* ~~“However, Zuse states in his memoirs that the basic circuits of the Z1 and Z3 were equivalent, and he confirmed this aspect of his work in a private interview.” (Rojas, “Konrad Zuse’s Legacy” 6)~~

**~~Memory Specifications~~**

* ~~“The Z3 consists of a binary memory unit (capable of storing 64 floating-point numbers) . . . .” (Rojas, “Simulating Konrad Zuse’s Computers” 64-66)~~
* ~~“Memory operations encode the address of a word in the lower 6 bits.” (Rojas, “Simulating Konrad Zuse’s Computers” 66)~~
* ~~“From the point of view of the software, the Z3 consists of 64 memory words that can be loaded into two floating-point registers, which I [Rojas] simply call ‘R1 and R2.’” (Rojas, “Simulating Konrad Zuse’s Computers” 66)~~
* ~~“The important point to remember is the following: The first load operation in a program transfers the contents of an address to R1. Any other subsequent load operation transfers a word from memory to R2. R2 is cleared after an arithmetical instruction, whereas the result is stored in R1.” (Rojas, “Simulating Konrad Zuse’s Computers” 66)~~
* ~~“The Z3 consists of a binary memory unit (capable of storing 64 floating-point numbers) . . . .” (Rojas, “Konrad Zuse’s Legacy” 6)~~
* ~~“Memory operations encode the address of a word in the lower six bits, that is, the addressing space has a maximum size of 64 words, as mentioned above.” (Rojas, “Konrad Zuse’s Legacy” 7)~~
* ~~“From the point of view of the software, the Z3 consists of 64 memory words that can be loaded into two floating-point registers, which I [Rojas] simply call R1 and R2.” (Rojas, “Konrad Zuse’s Legacy” 8)~~
* ~~“The Z3 processor employs just 600 relays; the memory needed three times as much. By having to optimize the design and by having to save hardware everywhere, Zuse was~~ *~~forced~~* ~~to think and rethink the logical structure of his machine.” (Rojas, “Konrad Zuse’s Legacy” 15)~~

**~~Registers~~**

How many? Names? Uses? Sizes?

* ~~“From the point of view of the software, the Z3 consists of 64 memory words that can be loaded into two floating-point registers, which I [Rojas] simply call ‘R1 and R2.’ These two registers contain the arguments of arithmetical operations.” (Rojas, “Simulating Konrad Zuse’s Computers” 66)~~
* ~~“The important point to remember is the following: The first load operation in a program transfers the contents of an address to R1. Any other subsequent load operation transfers a word from memory to R2. R2 is cleared after an arithmetical instruction, whereas the result is stored in R1.” (Rojas, “Simulating Konrad Zuse’s Computers” 66)~~
* ~~“There are two parts [of the arithmetical unit]: The left side is used for operations with the exponents of the floating-point numbers, the right side is used for operations with the mantissas.~~ *~~Af~~* ~~and~~ *~~Bf~~* ~~are registers used to store the exponent and mantissa of what, from the programmer’s point of view, is register R1. I [Rojas] refer to R1 as the register pair~~ *~~[Af:Bf]~~*~~. The register pair~~ *~~[Ab:Bb]~~* ~~stores the exponent and mantissa of R2. The pair~~ *~~[Aa:Ba]~~* ~~contains the exponent and the mantissa of a third temporal floating-point register invisible to programmers.” (Rojas, “Simulating Konrad Zuse’s Computers” 66)~~
* ~~“The two arithmetic logical units (ALUs),~~ *~~A~~* ~~and~~ *~~B~~*~~, are used to add or subtract exponents and mantissas, respectively. The result of the operation in the exponent part is put into~~ *~~Ae~~*~~. In the mantissa part, the result of the operation is put into~~ *~~Be~~*~~. The pair~~ *~~[Ae:Be]~~* ~~can be considered an internal register invisible to programmers.” (Rojas, “Simulating Konrad Zuse’s Computers” 66)~~
* ~~“The remaining instructions performed mathematical operations on the arithmetic unit’s two floating-point registers, R1 and R2. Addition, subtraction, multiplication, and division combined R1 and R2 placing the result in R1 and clearing R2.” (Tarnoff 81)~~
* ~~“From the point of view of the software, the Z3 consists of 64 memory words that can be loaded into two floating-point registers, which I [Rojas] simply call R1 and R2. These two registers contain the two arguments of arithmetical operations requiring them. The programmer can write any sequence of instructions, but has to keep in mind the state of the machine’s registers.” (Rojas, “Konrad Zuse’s Legacy” 8)~~
* ~~“The important point to remember is the following: The first load operation in a program (Pr z) transfers the contents of address z to R1. Any other subsequent load operation transfers a word from memory to R2. A read keyboard instruction loads the numerical input into R1 and~~ *~~clears~~* ~~R2, which is used to hold temporary values during the transformation of the decimal input to a binary representation.” (Rojas, “Konrad Zuse’s Legacy” 8)~~
* ~~“There are two parts [of the arithmetical unit]: The left side is used for operations with the exponents of the floating-point numbers, the right side for operations with the significands. Af and Bf are registers used to store the exponent and significand of what, from the programmer’s point of view, is R1. I [Rojas] will refer to R1 as the register pair <Af, Bf>. The register pair <Ab, Bb> stores the exponent and significand of R2. The pair <Aa, Ba> contains the exponent and significand of a third temporary floating-point register invisible to the programmer.” (Rojas, “Konrad Zuse’s Legacy” 8)~~
* ~~“The two arithmetic logic units (ALUs) A and B are used to add or subtract exponents and significands, respectively. The result of the operation in the exponent part is put into Ae. In the significand part, the result of the operation is put into Be. The pair <Ae,Be> can be considered an internal register invisible to the programmer.” (Rojas, “Konrad Zuse’s Legacy” 8)~~

|  |  |  |  |
| --- | --- | --- | --- |
| Af | 7 bits | Bf | 17 bits |
| Aa | 8 | Ba | 19 |
| Ab | 8 | Bb | 18 |
| Ae | 8 | Be | 18 |

(Rojas, “Konrad Zuse’s Legacy” 9)

* ~~“As can be seen from this list, Ae uses one extra bit to handle the addition of the exponents of the arguments. Part B of the processor uses two extra bits for the significands . . . and makes explicit b~~~~0~~~~, which is not stored in memory. The extra bits at positions -15 and -16 are included to increase the precision of the computations. Therefore, the total number of bits needed to store the result of an arithmetical operation in Bf is 17 bits. Registers Ba and Bb require more extra bits . . . to handle intermediate results of some of the numerical algorithms. In particular, the square root algorithm can lead to partial computations in Ba requiring three bits to the left of the decimal point.” (Rojas, “Konrad Zuse’s Legacy” 9)~~

**~~Control Unit and ALU~~**

* ~~“Its [the Z1’s] arithmetic unit worked with a ‘semi-logarithmic’ number representation invented by Zuse—years later this became known as~~ *~~floating-point numbers~~*~~.” (Giloi 11)~~
* ~~“The Z3 is a floating-point machine. Whereas other early computing automatons worked with fixed-point numbers, Zuse decided early on to adopt what he called ‘semilogarithmic’ notation, which corresponds to the modern floating-point representation.” (Rojas, “Simulating Konrad Zuse’s Computers” 64)~~
* ~~“The Z3 consists of a binary memory unit (capable of storing 64 floating-point numbers), a binary floating-point processor, a control unit, and I/O devices.” (Rojas, “Simulating Konrad Zuse’s Computers” 64-66)~~
* ~~“The memory and the arithmetical unit are connected through a data bus, which transmits the exponent and mantissa of the floating-point representation.” (Rojas, “Simulating Konrad Zuse’s Computers” 66)~~
* ~~“The control unit contains the microsequencers needed for each instruction. Control lines going from the control unit to the processor, the memory, and the I/O devices enforce the correct synchronization of all units.” (Rojas, “Simulating Konrad Zuse’s Computers” 66)~~
* ~~“The tape reader provides the opcode of each instruction as well as the address for memory accesses.” (Rojas, “Simulating Konrad Zuse’s Computers” 66)~~
* ~~“There are two parts [of the arithmetical unit]: The left side is used for operations with the exponents of the floating-point numbers, the right side is used for operations with the mantissas.~~ *~~Af~~* ~~and~~ *~~Bf~~* ~~are registers used to store the exponent and mantissa of what, from the programmer’s point of view, is register R1. I [Rojas] refer to R1 as the register pair~~ *~~[Af:Bf]~~*~~. The register pair~~ *~~[Ab:Bb]~~* ~~stores the exponent and mantissa of R2. The pair~~ *~~[Aa:Ba]~~* ~~contains the exponent and the mantissa of a third temporal floating-point register invisible to programmers.” (Rojas, “Simulating Konrad Zuse’s Computers” 66)~~
* ~~“The two arithmetic logical units (ALUs),~~ *~~A~~* ~~and~~ *~~B~~*~~, are used to add or subtract exponents and mantissas, respectively. The result of the operation in the exponent part is put into~~ *~~Ae~~*~~. In the mantissa part, the result of the operation is put into~~ *~~Be~~*~~. The pair~~ *~~[Ae:Be]~~* ~~can be considered an internal register invisible to programmers. In part~~ *~~B~~*~~, a multiplexer allows selection of~~ *~~Ba~~* ~~or the output of the ALU as the result of the operation. The multiplexer is controlled by a relay~~ *~~Bt~~* ~~(it~~ *~~Bt~~* ~~is equal to zero, then~~ *~~Be~~* ~~is set equal to~~ *~~Ba~~*~~).” (Rojas, “Simulating Konrad Zuse’s Computers” 66)~~
* ~~“The small boxes labeled~~ *~~Ea, Eb, Ec, Ed, Ef, Fa, Fb, Fc, Fd,~~* ~~and~~ *~~Ff~~* ~~are switches that open or close the data bus. The structure of part~~ *~~B~~* ~~of the arithmetical unit is similar, but in addition to the multiplexer controlled relay~~ *~~Bt~~*~~, there is also a shifter between~~ *~~Bf~~* ~~and~~*~~Ba~~* ~~and a shifter between~~ *~~Bf~~* ~~and~~ *~~Bb~~*~~. The first shifter can displace the mantissa up to two positions to the right and one position to the left. This amounts to a division of~~ *~~Bf~~* ~~by 4 or a multiplication by 2. The second shifter can displace the mantissa in~~ *~~Af~~* ~~from 1 to 16 positions to the right and from 1 to 15 positions to the left. These shifts are needed for addition and subtraction of floating-point numbers. Multiplication and division with powers of two can therefore be performed when the operands for the next arithmetical operation are fetched and, in this sense, do not consume time.” (Rojas, “Simulating Konrad Zuse’s Computers” 67)~~
* ~~“The basic primitive operation of the datapath is the addition or subtraction of exponents or mantissas. When the relay~~ *~~As~~* ~~(~~*~~Bs~~*~~) is set, the negation of the second argument~~ *~~Ab~~* ~~(~~*~~Bb~~*~~) is fed into the ALU. Therefore, if the relay~~ *~~As~~* ~~is set to 1, the ALU in part~~ *~~A~~* ~~subtracts its arguments; otherwise, they are added. The same is true for part~~ *~~B~~* ~~and the relay~~ *~~Bs~~*~~. The constant 1 is needed to build the two’s complement of a number.” (Rojas, “Simulating Konrad Zuse’s Computers” 67)~~
* ~~“The Z3 is a floating-point machine. Whereas other early computing automatons like the Mark I, the ABC, and the ENIAC worked with fixed-point numbers, Zuse decided very early on to adopt what he called ‘semilogarithmic’ notation, which corresponds to the modern floating-point representation.” (Rojas, “Konrad Zuse’s Legacy” 6)~~
* ~~“The Z3 consists of . . . a binary floating-point processor, a control unit, and I/O devices.” (Rojas, “Konrad Zuse’s Legacy” 6)~~
* ~~Memory and the arithmetical unit are connected through a data bus, which transmits the exponent and significand of the floating-point representation. The control unit contains the microsequencers needed for each instruction. Control lines going from the control unit to the processor, the memory, and the I/O devices enforce the correct synchronization of all units.” (Rojas, “Konrad Zuse’s Legacy” 6)~~
* ~~“Operations involving zero and infinity are treated as exceptions, and special hardware monitors the numbers loaded in the processor in order to set the exception flags.” (Rojas, “Konrad Zuse’s Legacy” 7)~~
* ~~“There are two parts [of the arithmetical unit]: The left side is used for operations with the exponents of the floating-point numbers, the right side for operations with the significands. Af and Bf are registers used to store the exponent and significand of what, from the programmer’s point of view, is R1. I [Rojas] will refer to R1 as the register pair <Af, Bf>. The register pair <Ab, Bb> stores the exponent and significand of R2. The pair <Aa, Ba> contains the exponent and significand of a third temporary floating-point register invisible to the programmer.” (Rojas, “Konrad Zuse’s Legacy” 8)~~
* ~~“The two arithmetic logic units (ALUs) A and B are used to add or subtract exponents and significands, respectively. The result of the operation in the exponent part is put into Ae. In the significand part, the result of the operation is put into Be. The pair <Ae,Be> can be considered an internal register invisible to the programmer. In Part B, a multiplexer allows selection of Ba or the output of the ALU as the result of the operation. The multiplexer is controlled by a relay Bt (if Bt = 0, the Be is set equal to Ba).” (Rojas, “Konrad Zuse’s Legacy” 8)~~
* ~~“The small boxes labeled Ea, Eb, Ec, Ed, Ef, Fa, Fb, Fc, Fd, and Ff are switches that open or close the data bus. If the contents of register Af are to be transferred to Aa, for example, the box of relays Ea is set to one and the result is Aa:=Af. . . . The structure of Part B of the arithmetical unit is very similar, but in addition to the multiplexer controlled by the relay Bt, there is also a shifter between Bf and Ba and a shifter between Bf and Bb. The first shifter can displace the significand up to two positions to the right and one position to the left. This amounts to a division of Bf by four or a multiplication with the constant two. The second shifter can displace the significand in Af from one to 16 positions to the right and from one to 15 positions to the left. These shifts are needed for addition and subtraction of floating-point numbers. Multiplication and division with powers of two can therefore be performed when the operands for the next arithmetical operation are fetched and, in this sense, do not consume time.” (Rojas, “Konrad Zuse’s Legacy” 8-9)~~
* ~~“The basic primitive operation of the data path is the addition or subtraction of exponents or significands. When the relay As or Bs is set, the negation of the second argument (Ab or Bb) is fed into the ALU. Therefore, if the relay As is set to one, the ALU in Part A subtracts its arguments, otherwise they are added. The same is true for Part B and the relay Bs. The constant of one is needed to build the two’s complement of a number.” (Rojas, “Konrad Zuse’s Legacy” 9)~~
* ~~“The computer architect has to provide the correct sequence of activations of the relay boxes in order to get the desired operation. This is done in the Z3 using a technique very similar to microprogramming.” (Rojas, “Konrad Zuse’s Legacy” 9)~~
* ~~“The circuit Pa decodes the opcode of the instruction read from the punched tape. If it is a memory instruction, circuit Pb sets the address bus to the value of the lower six bits of the opcode. The control unit determines the correct microsequencing of the instructions. There are special circuits for each of the operations in the instruction set.” (Rojas, “Konrad Zuse’s Legacy” 9)~~
* ~~“The decimal input must be transformed into a binary number. This requires a chain of multiplications, which is longer according to the absolute magnitude of the exponent. If the exponent is zero, the whole transformation requires nine cycles, but if it is -8, the operation requires 9 + 4 x 8 = 41 cycles.” (Rojas, “Konrad Zuse’s Legacy” 9)~~
* ~~“The heart of the control unit is made up of its microsequencers.” (Rojas, “Konrad Zuse’s Legacy” 9)~~
* ~~“Each cycle of the Z3 is divided into five stages. Stages IV and V are used to move information around in the machine. During Stages I, II, and III, an addition/subtraction is computed in Part A and another in Part B of the Z3. I [Rojas] call this the ‘execute’ phase of an instruction. A typical instruction fetches its arguments, executes, and writes back the result. Zuse took great care to save execution time by overlapping the fetch stage of the next instruction with the write-back stage of the current one. One can think of an execution cycle as consisting of just two stages . . . .” (Rojas, “Konrad Zuse’s Legacy” 9)~~
* ~~“The microsequencing is done by special control wheels. There is one for the multiplication algorithm, another to control division, and yet another for the square root instruction.” (Rojas, “Konrad Zuse’s Legacy” 9)~~
* ~~“The moving arm . . . starts moving clockwise as soon as the control unit decodes the corresponding instruction. In each cycle, the arm moves from one position to the next. The arm conducts electricity and activates the circuits with which it comes into contact.” (Rojas, “Konrad Zuse’s Legacy” 9-10)~~
* ~~“As one can see, such control wheels provide a comfortable platform for modifying the exact sequence of events during an operation. They correspond to the microsequencers used today in modern microprocessors. I [Rojas] stop short of calling them a form of microprogramming, because in this case the microsequence has been hardwired, but it is obvious that microsequencing and microprogramming are closely related.” (Rojas, “Konrad Zuse’s Legacy” 10)~~
* ~~“Extensive use of microsequencing allowed Zuse to simplify the Z3. Once the basic circuits had been laid out, it was just a matter of refining the control until optimal sequences of events could be found. There are a lot of details that the engineer designing the ‘microprogram’ must keep in mind, otherwise short circuits can destroy the hardware.” (Rojas, “Konrad Zuse’s Legacy” 10)~~
* ~~“An important feature of the Z3 is the design of the adders, which compute additions and subtractions using a method called~~ *~~carry look-ahead~~*~~. If binary addition is implemented in a straightforward way, carries have to be passed from one bit position to the next. In the case of the significand, one would need 16 cycles just for the transmission of the carry bits. The adders Zuse designed are much faster than that—they perform an addition or subtraction in Stages I, II, and III of a single cycle. Subtraction is computed by complementing the second argument and adding an extra digit one at the lowest bit position.” (Rojas, “Konrad Zuse’s Legacy” 10)~~
* ~~“The problem with floating-point notation is that special conventions have to be used to deal with the number zero. The Z3 solves this problem and deals with other exceptions (overflow and underflow) by monitoring the value of the exponent after any arithmetical operation or a load from memory. A special circuit looks at the state of the bus Ae and captures exceptions.” (Rojas, “Konrad Zuse’s Legacy” 10)~~
* ~~“Any number with exponent -64 is flagged as zero: A relay denoted Nn~~~~1~~ ~~is set to one if the number is stored in the register pair <Af, Bf>. If the number is stored in the register pair <Ab,Bb>, the relay Nn~~~~2~~ ~~is set to one. In this way, we always know if one or both of the arguments for an arithmetical operation are zero. Something similar is done for any exponent of value 63 (an infinite number, according to the convention). In this case, the relays Ni~~~~1~~ ~~or Ni~~~~2~~ ~~are set to one, according to the register pair in which the number is stored.” (Rojas, “Konrad Zuse’s Legacy” 10-11)~~
* ~~“Operations involving ‘exceptional’ numbers (zero or infinity) are performed as usual, but the result is overridden by the snooping circuit.” (Rojas, “Konrad Zuse’s Legacy” 11)~~
* ~~“The Z3 can detect undefined operations such as 0/0, ∞ - ∞, ∞/∞, and 0 x ∞. In all these cases, the corresponding exception lamp lights on the output panel, and the machine is stopped. The Z3 always produces the correct result when one of the arguments is zero or ∞ and the other is a number within bounds.” (Rojas, “Konrad Zuse’s Legacy” 11)~~
* ~~“An additional circuit looks at the exponent of the result at the output of the exponent’s adder. If the exponent is greater than or equal to 63, overflow has occurred and the result must be set to ∞. If the exponent is lower than -64, underflow has occurred and the result must be set to zero. To do this, the appropriate relay (Nn~~~~1~~ ~~or Ni~~~~1~~~~) is set to one.” (Rojas, “Konrad Zuse’s Legacy” 11)~~
* ~~“Zuse managed to implement exception handling using just a few relays. This feature of the Z3 is one of the most elegant in the whole design.” (Rojas, “Konrad Zuse’s Legacy” 11)~~
* ~~“In order to add or subtract two floating-point numbers~~ *~~x~~* ~~and~~ *~~y~~*~~, their representation must be reduced to the same exponent. After this has been done, only the significands have to be added or subtracted. If the exponents are different, the significand of the smaller number is shifted to the right as many places as necessary (and its exponent is incremented correspondingly to keep the number unchanged) until both exponents are equal. It can, of course, happen that the smaller number becomes zero after 17 shifts to the right.” (Rojas, “Konrad Zuse’s Legacy” 11)~~
* ~~“The signs of the two numbers are compared before deciding on the type of operation to be executed. If an addition has been requested and the signs are the same, the addition is performed. If the signs are different, a subtraction is executed. If a subtraction has been requested and the signs are different, an addition is executed. If the signs are the same, the subtraction is executed. A special circuit sets the sign of the final result according to the signs of the arguments and the sign of the partial result.” (Rojas, “Konrad Zuse’s Legacy” 11)~~
* ~~“Addition and subtraction are controlled by a chain of relays (not by a control wheel), since the maximum number of cycles needed is low. . . . Initially, the arguments for the addition in the register pairs <Af,Bf> and <Ab,Bb>. In the first cycle, the exponents are subtracted. In Cycle 2, the significand with the larger exponent is loaded into register Ba, and the significand with the smaller exponent is loaded into register Bb. The significand in register Bb is shifted as many places to the right as the absolute value of the difference of the exponents (exception handling takes care of the case in which the smaller number becomes zero after the shift). In Stages I, II, and III of Cycle 2, the significands are added, and finally the processor tests if the result is greater than two. If this is the case, the significand is shifted one position to the right and the exponent is incremented by one. Note that the test “if (Be ≥ 2)” in Part A of the arithmetical unit is done~~ *~~after~~* ~~Be has already been computed in Part B during Stages I, II, and III of Cycle 2.” (Rojas, “Konrad Zuse’s Legacy” 11)~~
* ~~“In the case of subtraction, four or five cycles are needed. . . . The first two cycles are almost identical to the first two cycles of the addition algorithm, but now the significands are subtracted. Cycle 3 is executed only when the difference of the significands is negative. The effect of Cycle 3 is just to make the significand of the result positive. Cycle 4 is very important: The difference of two normalized significands can have many zeros in the first bit positions to the left. The result is normalized by shifting Be to the left as many places as necessary (this is done with the shifter between the relay box Fd and register Bb). The number of one-bit shifts is subtracted from the exponent in Part A of the processor. In Cycle 5, the result is stored in the register pair <Af,Bf>.”(Rojas, “Konrad Zuse’s Legacy” 11)~~
* ~~“The multiplication algorithm of the Z3 is like the one used for decimal multiplication by hand, that is, it is based on repeated additions of the multiplicator according to the individual digits of the multiplicand.” (Rojas, “Konrad Zuse’s Legacy” 11)~~
* ~~“The [multiplication] algorithm takes 16 cycles to run. Note that only the bits of the multiplicand from position -14 to position zero are used. The exponents are added in the first cycle and the result just loops afterward in Part A of the arithmetical unit. The significands are handled in Part B of the unit.” (Rojas, “Konrad Zuse’s Legacy” 12)~~
* ~~“The result of the multiplication is a number 1 ≤~~ *~~r~~* ~~< 4 (for arguments within bounds). In the last cycle, there is a check to see if~~ *~~r~~* ~~≥ 2. If this is the case, the result is shifted one position to the right and a one is added to the exponent of the result.” (Rojas, “Konrad Zuse’s Legacy” 12)~~
* ~~“The division algorithm is similar to the multiplication algorithm, but subtraction is used repetitively instead of addition.” (Rojas, “Konrad Zuse’s Legacy” 12)~~
* ~~“The [division] algorithm takes 18 cycles to run.” (Rojas, “Konrad Zuse’s Legacy” 12)~~
* ~~“The main idea of the [division] algorithm is very simple. The exponent of the result is obtained by subtracting the exponents of dividend and divisor. Now for the significand: Assume that we want to compute~~ *~~x~~*~~/~~*~~y~~* ~~for the significands~~ *~~x~~* ~~and~~ *~~y~~*~~. Since we are dealing with normalized numbers, the first digit of the result is one if~~ *~~x~~* ~~≥~~ *~~y~~* ~~and zero if~~ *~~x~~* ~~<~~ *~~y~~*~~. In the first case, we set the first digit of the result to one and compute the remainder, which is~~ *~~x~~* ~~–~~ *~~y~~*~~. The remainder is divided recursively by~~ *~~y~~*~~. To do this, it is shifted one position to the left, and the new result bit is stored at position [-1] of register Bf (in this way nullifying the effect of the shift). If the result bit is zero, the remainder is just~~ *~~x~~*~~, and the recursive division is continued as in the first case.” (Rojas, “Konrad Zuse’s Legacy” 12)~~
* ~~“The result of the division of significands is a number ½ <~~ *~~r~~* ~~< 2. This condition is tested in Cycles 17 and 18. If~~ *~~r~~* ~~< 1, a one is subtracted from the exponent, and the result is shifted one position to the left in order to get a normalized number.” (Rojas, “Konrad Zuse’s Legacy” 12)~~
* ~~“The square root algorithm is the jewel in the crown of the Z3.” (Rojas, “Konrad Zuse’s Legacy” 12)~~
* ~~“ . . . 20 cycles needed to compute the square root of a number.” (Rojas, “Konrad Zuse’s Legacy” 12-13)~~
* ~~“The [square root] algorithm computes the square root of numbers with an even exponent. If the exponent is an odd number, the significand is shifted one place to the left, and the exponent is decremented by one. The final exponent (computed in Cycle 19) is half this initial exponent.” (Rojas, “Konrad Zuse’s Legacy” 13)~~
* ~~“The main idea of the classical [square root] algorithm is to reduce the square root operation to a division.” (Rojas, “Konrad Zuse’s Legacy” 13)~~
* ~~“All bits of register Bf are used for the computation of the square root. If the original number lies within bounds, the result is also within bounds.” (Rojas, “Konrad Zuse’s Legacy” 13)~~
* ~~“The box labeled ‘zero, infinite’ below Ae represents the circuits for exception handling. They snoop permanently on the data bus (results of operations and data from memory) and raise the corresponding exception flags when needed. The shifter below Be is used to displace the significand one bit to the right. This provide the normalization needed for the significand whenever Be ≥ 2.” (Rojas, “Konrad Zuse’s Legacy” 15)~~
* ~~“The Z3 processor employs just 600 relays; the memory needed three times as much. By having to optimize the design and by having to save hardware everywhere, Zuse was~~ *~~forced~~* ~~to think and rethink the logical structure of his machine.” (Rojas, “Konrad Zuse’s Legacy” 15)~~

**~~Data Types~~**

* ~~“The floating-point representation of the Z3 used 1 bit for the sign, 7 bits for the exponent (in two’s complement coding), and 14bits for the normalized mantissa. It was, in fact, somewhat similar to today’s IEEE 754 standard.” (Rojas, “Simulating Konrad Zuse’s Computers” 66)~~
* ~~“The problem with normalized floating-point notation is that special conventions have to be used to deal with the number zero. The minimal exponent was used to code zero, the maximal to code infinite numbers.” (Rojas, “Simulating Konrad Zuse’s Computers” 66)~~
* ~~“Zuse’s floating-point data format is remarkably similar to the IEEE Standard for Floating-Point Arithmetic (IEEE 754) in common use today. The most significant bit is a sign bit for the value. The next seven bits contain the base-two exponent coded in two’s complement representation. The last fourteen bits are the significant without the ‘hidden bit’, i.e., the most significant bit of the significand, which is always a one for non-zero values. Zero was represented with an exponent of -64.” (Tarnoff 81)~~
* ~~“The first bit is used to store the sign of the number, the following seven bits for the exponent, and the last 14 bits for the significand (only the 14 places to the right of the decimal point). . . . The exponent is coded as a two’s complement number. The range of possible values therefore runs from -64 to 64. The significand is stored in~~ *~~normalized~~* ~~form, that is the first digit before the decimal point . . . must always be a one (Donald Knuth attributes the invention of~~ *~~normalized~~* ~~floating-point numbers to Zuse.). This digit does not need to be stored . . . so that the effective range of the numbers in the memory unit is equivalent to a significand of 15 bits.” (Rojas, “Konrad Zuse’s Legacy” 6-7)~~
* ~~“However, there is a problem with the number zero, which cannot be expressed using a normalized significand. The Z3 uses the convention that any significand with exponent -64 is to be considered equal to zero. Any number with exponent 63 is considered infinitely large.” (Rojas, “Konrad Zuse’s Legacy” 7)~~
* ~~“With this convention, the smallest number representable in the memory of the Z3 is 2~~~~-63~~ ~~= 1.08 x 10~~~~-19~~~~, and the largest is 1.999 x 2~~~~62~~ ~~= 9.2 x 10~~~~18~~~~.” (Rojas, “Konrad Zuse’s Legacy” 7)~~

**~~Instruction Set~~**~~,~~ **~~Instruction Format, and Addressing~~**

*ISA Differentiation*

Bits per instruction

Stack-based or register-based

Number of explicit operands per instruction

Operand location (register and/or memory)

Types of operands (e.g., Arithmetic, Branching)

Type and size of operands

*ISA Measurement*

Main memory space occupied by a program

Instruction complexity

Instruction length (in bits)

Total number of instructions in instruction set

*Instruction Format*

Instruction length (short, long, or variable size)

Number of operands (one, two, three, etc.)

Number of addressable registers

Memory organization (byte-addressable or word-addressable; endianness)

* ~~“The program for the Z3 is stored on punched tape. One instruction is coded using 8 bits for each row of the tape. The instruction set of the Z3 consists of the nine instructions in Table 1. Memory operations encode the address of a word in the lower 6 bits. The operating frequency of the Z3 was about 5 Hz.” (Rojas, “Simulating Konrad Zuse’s Computers” 66)~~
* ~~“You can write any sequence of instructions, but you have to keep track of the state of the machine’s registers.” (Rojas, “Simulating Konrad Zuse’s Computers” 66)~~
* ~~“The important point to remember is the following: The first load operation in a program transfers the contents of an address to R1. Any other subsequent load operation transfers a word from memory to R2. R2 is cleared after an arithmetical instruction, whereas the result is stored in R1.” (Rojas, “Simulating Konrad Zuse’s Computers” 66)~~
* ~~“The main defect of the Z3 was the absence of a conditional branch in the instruction set. Nevertheless, it can be proved that a machine capable of executing a single loop and the basic arithmetic operations is equivalent to any computer with a limited addressing space.” (Rojas, “Simulating Konrad Zuse’s Computers” 68-69)~~
* ~~“The Z1 instruction set consisted of eight 8-bit instructions including two for memory transfers and one each for input and output. The remaining instructions performed mathematical operations on the arithmetic unit’s two floating-point registers, R1 and R2. Addition, subtraction, multiplication, and division combined R1 and R2 placing the result in R1 and clearing R2.” (Tarnoff 81)~~
* ~~“The memory load instruction’s operand field held the address from which data was to be loaded, but did not identify the destination register. This was because the first load would load R1 while subsequent loads loaded R2. The memory store instruction’s operand field also did not identify a register. All stores would store R1 to memory and clear R1. A store would reset the loading process to load R1 next.” (Tarnoff 81)~~
* ~~“What the Z1 lacked was program flow control. There were no conditional branches. Loops, therefore, could only be implemented by connecting the paper tape’s ends to form a physical loop in the code. There was also no way to implement decision structures.” (Tarnoff 82)~~
* ~~“The tape reader provides the opcode of each instruction as well as the address for memory accesses.” (Rojas, “Konrad Zuse’s Legacy” 6)~~
* ~~“The program for the Z3 is stored on punched tape. One instruction is coded using eight bits for each row of the tape.” (Rojas, “Konrad Zuse’s Legacy” 7)~~
* ~~“There are three types of instructions: I/O, memory, and arithmetical operators.” (Rojas, “Konrad Zuse’s Legacy” 7)~~
* “~~The opcode has a variable length of two or five bits. Memory operations encode the address of a word in the lower six bits, that is, the addressing space has a maximum size of 64 words, as mentioned above.” (Rojas, “Konrad Zuse’s Legacy” 7)~~
* ~~“The instructions on the punched tape can be arranged in any order. The instructions Lu and Ld (read from keyboard and display result, respectively) halt the machine, so that the operator has enough time to input a number or write down a result. The machine is then restarted and continues processing the program.” (Rojas, “Konrad Zuse’s Legacy” 7)~~
* ~~“The instruction most conspicuously absent from the instruction set of the Z3 is conditional branching. Loops can be implemented by the simple expedient of bringing together the two ends of the punched tape, but there is no way to implement conditional sequences of instructions. The Z3 is therefore not a universal computer in the sense of Turing.” (Rojas, “Konrad Zuse’s Legacy” 7)~~

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Type | Instruction | Description | Opcode | Cycles |
| I/O | Lu  Ld | read keyboard  display result | 01 110000  01 111000 | 9 to 41\*  9 to 41\* |
| memory | Pr z  Ps z | load address z  store address z | 11 z6z5z4z3z2z1  10 z6z5z4z3z2z1 | 1  0 or 1 |
| arithmetic | Lm  Li  Lw  Ls1  Ls2 | multiplication  division  square root  addition  subtraction | 01 001000  01 010000  01 011000  01 100000  01 101000 | 16  18  20  3  4 or 5† |

\* Depending on the exponent

† Depending on the result

(Rojas, “Konrad Zuse’s Legacy” 7)

* ~~“According to Zuse, the time required for a multiplication was three seconds. Considering that a multiplication operation needs 16 cycles, one can estimate that the operating frequency of the Z3 was 16/3 ≈ 5.33 Hz.” (Rojas, “Konrad Zuse’s Legacy” 7)~~
* ~~“The number of cycles needed for the~~ *~~read~~* ~~and~~ *~~display~~* ~~instructions is variable, because it depends on the exponent of the arguments. Since the input has to be converted from decimal to binary representation, the number of multiplications needed with the factor 10 or 0.1 is dictated by the decimal exponent.” (Rojas, “Konrad Zuse’s Legacy” 7)~~
* ~~“Addition and subtraction require more than one cycle because, in the case of floating-point numbers, care has to be taken to set the size of the exponent of both arguments to the same value. This requires some extra comparisons and shifting.” (Rojas, “Konrad Zuse’s Legacy” 7)~~
* ~~“A number can be stored in memory in~~ *~~zero~~* ~~cycles when the result of the last arithmetical operation can be redirected to the desired memory address. In this case, the cycle needed for the store instruction overlaps the last cycle of the arithmetical operation.” (Rojas, “Konrad Zuse’s Legacy” 7)~~
* ~~“The important point to remember is the following: The first load operation in a program (Pr z) transfers the contents of address z to R1. Any other subsequent load operation transfers a word from memory to R2. A read keyboard instruction loads the numerical input into R1 and~~ *~~clears~~* ~~R2, which is used to hold temporary values during the transformation of the decimal input to a binary representation.” (Rojas, “Konrad Zuse’s Legacy” 8)~~
* ~~“Arithmetical operations do not specify their arguments in the opcode. Their implicit semantics are the following:~~

~~Multiplication: R1:=R1 x R2~~

~~Division: R1:= R1/R2~~

~~Addition: R1:= R1+R2~~

~~Subtraction: R1:= R1-R2~~

~~Square Root: R1:= sqrt(R1)” (Rojas, “Konrad Zuse’s Legacy” 8)~~

* ~~“R2 is set to zero after an arithmetical instruction, whereas the result is stored in R1. Subsequent load operations refer to R2. The store and display instructions always refer to R1, which also contains the result of the previous arithmetical operation. After a store or a display operation, R1 is set to zero (by disconnecting its relays, which then become ready to accept a new value). The next load operation refers to R1.” (Rojas, “Konrad Zuse’s Legacy” 8)~~
* ~~“An example is better than many additional remarks to clarify the programming model of the Z3. Assume that we want to compute a polynomial using Horner’s method: x(a~~~~2~~ ~~+ x(a~~~~3~~ ~~+ xa~~~~4~~~~)) + a~~~~1~~~~. Assume further that we have stored the constants a~~~~4~~~~, a~~~~3~~~~, a~~~~2~~~~, a~~~~1~~~~, in the address four, three, two, and one of the memory unit. The value z [actually x] is stored in address five. The program that performs the desired computation is the following:~~

~~Pr 4 load a~~~~4~~ ~~in R1~~

~~Pr 5 load x in R2~~

~~Lm multiply R1 and R2, result in R1~~

~~Pr 3 load a~~~~3~~ ~~in R2~~

~~Ls~~~~1~~ ~~add R1 and R2, result in R1~~

~~Pr 5 load x in R2~~

~~Lm multiply R1 and R2, result in R1~~

~~Pr 2 load a~~~~2~~ ~~in R2~~

~~Ls~~~~1~~ ~~add R1 and R2, result in R1~~

~~Pr 5 load x in R2~~

~~Lm multiply R1 and R2, result in R1~~

~~Pr 1 load a­~~~~1~~ ~~in R2~~

~~Ls~~~~1~~ ~~add R1 and R2, result in R1~~

~~Ld display result” (Rojas, “Konrad Zuse’s Legacy” 8)~~

* ~~“After the last instruction [Ld] has been executed, the processor is reset to its initial state. A new program sequence can then be started.” (Rojas, “Konrad Zuse’s Legacy” 8)~~

**~~Input and Output Methods~~**

* ~~“The input is done through a decimal keyboard; a result is shown in a decimal array of lamps.” (Rojas, “Simulating Konrad Zuse’s Computers” 66)~~
* ~~“The Z3 could be used as a desktop calculator. The buttons to the upper right show the operations that can be performed with the machine. After entering two numbers (“Einlesen” or Input), for example, they can be added or multiplied. The result can then be shown on the display (“Ausgeben” or Output).” (Rojas, “Simulating Konrad Zuse’s Computers” 67)~~
* ~~“Finally, all of these systems [particularly the Z1] had to be halted in order to perform user I/O, which can be used as a starting point to discuss polled I/O and then interrupts.” (Tarnoff 86)~~
* ~~“The I/O devices are connected through a data bus to the computing unit.” (Rojas, “Konrad Zuse’s Legacy” 6)~~
* ~~“The arguments for computations can be entered as decimal numbers on the keyboard of the Z3 (four digits). The exponent of the decimal representation is entered by pushing the appropriate button in a row of buttons labeled -8, -7, …, 7, 8. The original Z3 could accept input only between 1 x 10~~~~-8~~ ~~and 9,999 x 10~~~~8~~~~.” (Rojas, “Konrad Zuse’s Legacy” 7)~~
* ~~“However, the Z3 does no print the numerical results the program produces. A single number is displayed on an array of lamps representing the digits from zero to nine. The largest number that can be displayed is 19,999. The smallest is 00001. The largest exponent that can be displayed is +8, the smallest -8.” (Rojas, “Konrad Zuse’s Legacy” 7)~~
* ~~“Circuit Z represents the panel of buttons used to enter a decimal number in the machine. Only one button in each of the four columns can be activated. The exponent is set by pressing one of the buttons labeled -8 to 8 in circuit K. The output display is very similar to the input panel, but here lamps illuminate the appropriate decimal digits, the exponent of the number (circuit Q), as well as its sign. Note that there is a fifth digit for the output (which can be only one or zero).” (Rojas, “Konrad Zuse’s Legacy” 9)~~
* ~~“The two most complex instructions of the Z3 are those related to the input and output of decimal numbers. A decimal number of four digits entered through the keyboard is first converted into a binary integer. This is done by reading each digit sequentially, transforming it into a binary number, and storing it in the bits Ba[-10], Ba[-11], Ba[-12], and Ba[-13] of register Ba. The number in register Ba is multiplied by 10, and the procedure is repeated for the other digits. After four iterations, the decimal input has been transformed to a binary number (the exponent of the binary representation is formed indirectly via shifts resulting from multiplication by 10).” (Rojas, “Konrad Zuse’s Legacy” 13)~~
* ~~“The difficult part is handling the exponent. If the exponent~~ *~~e~~* ~~is positive, the significand has to be multiplied~~ *~~e~~* ~~times with 10. If it is negative, it must be multiplied |~~ *~~e~~* ~~| times with 0.1. Multiplying by 10 is relatively easy: The significand in Be can be shifted one bit to the left and then stored in Ba (i.e., Ba := 2 x Be). At the same time, Be can be shifted three places to the left and can be stored in Bb (i.e., Bb := 8 x Be).” (Rojas, “Konrad Zuse’s Legacy” 13)~~
* ~~“The addition of Ba and Bb then provides the desired result: the multiplication of the original number in Be with the constant 10. The process takes four cycles for each multiplication, that is, 32 cycles for the decimal exponent +8. Since a read operation needs a minimum of 9 cycles, this means that a decimal number with exponent +8 is read in 41 cycles.” (Rojas, “Konrad Zuse’s Legacy” 13-14).~~
* ~~“In the case of negative exponents, multiplication with the constant 0.1 is performed using the shifters and the adders as well. This multiplication is somewhat more complex, because 0.1 is a periodic number in the binary system.” (Rojas, “Konrad Zuse’s Legacy” 14)~~
* ~~“The display instruction works by multiplying or dividing iteratively by 10. If the binary exponent of the number in register R1 is positive, the number is multiplied with 0.1 as many times as needed to make the binary exponent equal to two and until the first left four bits of register Bf contain a number between zero and nine (0000 and 1001). This is the decimal digit that can be displayed in the next column of the output panel. The number is subtracted from the significand in Bf, and the process continues for the following digits. If the binary exponent of the number in register R1 is negative, the process is similar, but multiplications with constant 10 are used.” (Rojas, “Konrad Zuse’s Legacy” 14)~~

**~~Unusual Features~~**

* ~~“In each cycle, the machine adds/subtracts the contents of its registers and discards the result if it is not needed. The machine was therefore loud.” (Rojas, “Simulating Konrad Zuse’s Computers” 67)~~
* ~~“The main defect of the Z3 was the absence of a conditional branch in the instruction set. Nevertheless, it can be proved that a machine capable of executing a single loop and the basic arithmetic operations is equivalent to any computer with a limited addressing space.” (Rojas, “Simulating Konrad Zuse’s Computers” 68-69)~~
* ~~“The program loop can be obtained in the Z3 by just gluing together both ends of the punched tape. The loop will be performed repetitively until a halting condition is reached.” (Rojas, “Simulating Konrad Zuse’s Computers” 69)~~
* ~~“Because the results of arithmetical operations can be used to set the values of the variables, all kinds of conditional branches can be executed. It can be proved that a Turing Machine with a tape of limited size can be simulated by the Z3 using this approach. . . . Thus, the Z3 can in fact simulate any other computer.” (Rojas, “Simulating Konrad Zuse’s Computers” 69)~~
* ~~“Only one problem remains: Since the program loop is being executed repetitively, how do we stop the machine? This can [be] done easily in the Z3 by causing an arithmetical exception. . . . Had Zuse not included arithmetical exceptions in his Z3, we would not be able to stop the loop, and this whole approach would not work.” (Rojas, “Simulating Konrad Zuse’s Computers” 69)~~
* ~~“The Z1 used a Harvard-style architecture executing programs directly from a punch tape reader (35mm standard movie film) while it stored data in a separate memory. In the case of the Z1, storing the programs in memory was avoided because of the expense of the memory.” (Tarnoff 81)~~
* ~~“The tape reader provides the opcode of each instruction as well as the address for memory accesses.” (Rojas, “Konrad Zuse’s Legacy” 6)~~

**~~Uses / Applications~~**

* ~~“From 1936 to 1938, Konrad Zuse constructed his Z1, an automaton he had designed in order to perform the time-consuming calculations necessary in his occupation as a civil engineer.” (Tarnoff 81)~~
* ~~“The Berlin Polytechnic student Zuse started thinking about computing machines in the 1930s. He realized that he could construct an automaton capable of executing a sequence of arithmetical operations like those needed to compute mathematical tables. Coming from a civil engineering background, he had no formal training in electronics and was not acquainted with the technology used in conventional mechanical calculators. This nominal deficit worked to his advantage, however, because he had to rethink the whole problem of arithmetic computation and thus hit on new and original solutions.” (Rojas, “Konrad Zuse’s Legacy” 5)~~
* ~~“The Z3, representing Zuse’s priority as the inventor and builder of the first free programmable automatic calculator, working in 1941, and anticipating an essential part of John von Neumann’s classical computer concepts from 1945, is a reconstruction, built by Zuse himself from 1961-1962 (the original machine had been destroyed by the bombs of World War II).” (Petzold 47-48).~~

**~~Contributions to the computer architecture landscape~~**

* ~~“Through Brian Randell’s book, an English translation of Zuse’s first patent application of 1936 has become more widely known. The application proves that Zuse had already developed all major concepts of the digital computer years before Burks, Goldstine, and von Neumann wrote their famous report~~ *~~Preliminary Discussion of the Logical Design of an Electronic Computing Instrument~~*~~. Indeed, Zuse’s vision of digital machines and their potential capabilities went beyond the purely sequential computer that is known to this day under the name ‘von Neumann machine.’ For example, the possibility of array processing and even of parallel processing is already mentioned in his patent application of 1936.” (Giloi 13)~~
* ~~“The [patent] claims in particular referred to a program-guided computing engine with an addressable memory, an instruction set (Rechenwerk) for floating-point arithmetic, and a program interpretation unit (Programwerk) to guide the computations. These are exactly the components of the von Neumann computer, described in the paper by Burks, Goldstine, and von Neumann.” (Giloi 15)~~
* ~~“Zuse was not happy with the term ‘von Neumann’ computer. He was of the opinion that such a computer was more appropriately called the Babbage computer or, even better, the Zuse computer. In fact, it was Zuse and not Babbage who first thought of the idea of addressable memory with binary addresses, the key feature of the von Neumann computer (Babbage realized memory by assigning each memory cell a particular position in his punch cards).” (Giloi 15)~~
* ~~“However, unlike Babbage’s concept, Zuse’s first computer did not allow for program jumps (and thus iteration): the Z2 and Z3 read instructions from a punched tape, and they could be read only in a particular order. In contrast, the von Neumann machine stores instructions in memory and thus can randomly access them. Zuse emphasized later that he thought from the beginning of storing both data and instructions. But since he initially did not understand the ramifications of incorporating goto’s, he preferred to leave them out. We suspect that part of the reason was that he did not want to waste precious memory for storing instructions, in particular since the numerical calculations he was familiar with did not (by and large) require loops. Thus, it was not until the Z4 that Zuse’s computers were ‘von Neumann computers’ in all regards.” (Giloi 15-16)~~
* ~~“Zuse’s highest profile claims to technical firsts are for the binary electromechanical computers built in Berlin by the early 1940s. His work was so confined with so little impact that Allied intelligence did not seem to notice it during the war, nor become interested afterward. Zuse and his central European supporters had to campaign later to get recognition for his originality.” (Goodman 24)~~
* ~~“The Z3, representing Zuse’s priority as the inventor and builder of the first free programmable automatic calculator, working in 1941, and anticipating an essential part of John von Neumann’s classical computer concepts from 1945, is a reconstruction, built by Zuse himself from 1961-1962 (the original machine had been destroyed by the bombs of World War II).” (Petzold 47-48).~~
* ~~“We can therefore say that, from an abstract theoretical perspective, the computing model of the Z3 is equivalent to the computing model of today’s computers. From a practical perspective, and in the way the Z3 was really programmed, it was not equivalent to modern computers. That’s why I [Rojas] prefer to speak not of the ‘first computer’ but of the ‘first computers’ of the world, in plural, referring by this to the American, British, and German machines, which were all built almost simultaneously at the dawn of the computer age.” (Rojas, “Simulating Konrad Zuse’s Computers” 69)~~
* ~~“Zuse decided to build his first experimental calculating machine exploiting two main ideas: the machine would work with binary numbers; the computing and control unit would be separated from the storage. Years before John von Neumann explained the advantages of a computer architecture in which the processor is separated from the memory, Zuse had already arrived at the same solution.” (Rojas, “Konrad Zuse’s Legacy” 5)~~
* ~~“Donald Knuth attributes the invention of~~ *~~normalized~~* ~~floating-point numbers to Zuse.” (Rojas, “Konrad Zuse’s Legacy” 6)~~
* ~~“Look again at the diagram of the Z3. Everything makes sense now and looks as conventional as any modern small floating-point processor. It is indeed amazing how Zuse was able to find the adequate architecture right from the beginning. The Z3 processor employs just 600 relays; the memory needed three times as much. By having to optimize the design and by having to save hardware everywhere, Zuse was~~ *~~forced~~* ~~to think and rethink the logical structure of his machine. He was not allowed the luxury of the almost unlimited funding allocated by the U.S. military for the development of the ENIAC or by IBM for the Mark I. He was all alone. While this may have worked to his advantage from the conceptual side, it may also worked to his disadvantage, considering the negligible impact that the Z1 and Z3 had on the emerging U.S. computer industry after World War II.” (Rojas, “Konrad Zuse’s Legacy” 15)~~
* ~~“The main defect of the Z3 was the absence of a conditional branch in the instruction set. When the program is stored on punched tape, a possible fix is to include multiple tapes and a mechanism to switch between them (as was done with the Harvard Mark I). Another possibility is having a ‘program counter,’ so that the tape can be advanced or rewound on demand.” (Rojas, “Konrad Zuse’s Legacy” 15)~~
* ~~“Sometimes the dividing line between calculating machines and universal computers is drawn by differentiating between machines with externally or internally stored programs. I [Rojas] have argued elsewhere that this is not a valid criterion. An external program can work as an interpreter of numerical data. The external program becomes a fixed part of the processor, and the data become the program, much in the same way as a universal Turing machine works as an interpreter. I [Rojas] have argued that what is needed for universal computation is a minimal instruction set and indirect addressing. Indirect addressing can be simulated by writing self-modifying programs, so that the instruction set becomes the defining criterion. A machine with enough addressable memory and an accumulator and that is capable of executing the instructions CLR (clear), INC (increment), LOAD, STORE , and BZ (branch if zero) is a universal computer. In this sense, the Z1 was~~ *~~not~~* ~~a fully fledged computer, but neither were any of the other early machines.” (Rojas, “Konrad Zuse’s Legacy” 15)~~

Comparison of Architectural Features

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Machine | Memory and CPU separated? | Conditional Branching? | Soft or hard programming | Self-modifying programs? | Indirect addressing? |
| Zuse’s Z1  Atanasoff’s  H-Mark I  ENIAC  M-Mark 1 | yes  yes  no  no  yes | no  no  no  partially  yes | soft  hard  soft  hard  soft | no  no  no  no  yes | no  no  no  no  no |

(Rojas, “Konrad Zuse’s Legacy” 15)

Some Additional Architectural Features

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Machine | Internal Coding | Fixed-point or floating-point? | Bit-sequential arithmetic? | architecture | technology |
| Zuse’s Z1  Atanasoff’s  H-Mark I  ENIAC  M-Mark I | binary  binary  decimal  decimal  binary | floating  fixed-point  fixed-point  fixed-point  fixed-point | no  yes  no  no  yes | sequential  vectorized  parallel  dataflow  sequential | mechanical  electronic  electromechanical  electronic  electronic |

(Rojas, “Konrad Zuse’s Legacy” 15)