Some Remarks on the History of Computing in Germany*

Konrad Zuse

Studying the history of computing, we can realize that there is a history of the philosophy of this history, too. From my own subjective point of view I can say that today I watch the computer development in another way than 10 or 20 years ago. Elaborating my lecture I finally realized that there are different philosophies behind the different developments, especially in comparison with my own.

First I should like to mention that in Germany the development of calculating machines began in 1623.

Contrary to the general opinion that Pascal and Leibniz were the first in this field, recent historical research has shown that the German Schickard was their forerunner by some thirty years. We have to thank Professor v. Freytag-Löringhoff who revealed the nearly forgotten work of Schickard in Tübingen and who reconstructed his machine.

We are able to welcome him here on this conference. This evening he will give us some details about the work of Schickard.

In this lecture I want to concentrate mainly on the computer development in Germany connected with my own work.

I have to thank Mr. Bauer that he gave you already on Friday a report on this matter. So I want to speak more about the principal viewpoints and the philosophy behind it.

Pr. Dr. Ing. Konrad Zuse. — 1936-45: developing and constructing of first computer Z3 (1941). — 1949-1966: building up the company Zuse K. G. — 1956: Hon. Professor at Göttingen University. — Since 1966: Research. — Publications: Der Computer mein Lebenswerk, Rechnender Raum, *etc.*

* Lecture at the International Research Conference on the History of Computing, University of California, Los Alamos, 10-15, 1976.

Nevertheless, I have to apologize for some repetitions.

Today we know that the development of the program controlled computer already began during the last century with Babbage. But he was so far ahead of his time that his machine was nearly completely forgotten. So in Germany, when I started in 1934, nearly nobody knew him and his computer development.

I was a student in civil engineering in Berlin. Berlin is a nice town and there were man opportunities for a student to spend his time in an agreable manner, for instance with the nice girls of Berlin.

But instead of that, we had to perform big and awful calculations.

Also later on as an engineer in the aircraft industry I became aware of the tremendous number of monotonous calculations necessary for the design of static and aerodynamic structures. Therefore, I decided to design and construct calculating machines suited to solve these problems automatically. This work proceeded almost parallel to, but quite independently of the developments in the United States by Stibitz, Aiken, Eckert, Mauchly, and others. It is interesting that during the pioneer days the computer development was represented by engineers and scientists who were not specialists in the field of calculating machines. At that time nobody knew the difference between hardware and software. We concentrated ourselves on purely technological matters as well as on logic design and programming.

So I was unprejudiced and free to try new concepts.

In order to illustrate the opinion of the manufacturers of calculating machines at that time, I would like to mention a telephone conversation which I had in 1937 with one of those manufacturers. He told me that it was, indeed, wonderful that I as a young man had dedicated some time and efforts to the development of new ideas, and that he wished me all the best for possible other inventions, but stated that in the techniques of calculating machines all feasible solutions were already exhausted. Therefore, it would be absolutely hopeless to come up with any new ideas. In addition, he asked me whether my machine was based on the « sequential addition principle » or on the « one times one table ».

To this I replied that for my machine this was of no importance whatsoever Here you should know that at that time the specialists of calculating machines were divided in two schools of thought, each applying either principle. According to the opinion prevailing at that time, only a lunatic could make a statement that this difference was irrelevant for his design. Nevertheless, the manufacturer mentioned came to my workshop and I was finally able to convince him that in a machine operating on the binary principle, this was, indeed, irrelevant.

Now some diagrams may show you the most relevant features of this development.

Light 1. First you see a time scale with the most important periods. In 1934, as a student. I started to form my first ideas and designs on paper. In 1936, I began with the hardware and constructed some models Z1, Z2, and Z3. Z1 and Z2 were only test models. They already had all the features of the later computer but did not work satisfactorily.

In 1939, due to the perfectly private state of my workshop and due to the lack of official sponsering I got a soldier with the beginning of the war.

The manufacturer, who assisted me, wrote a letter to my Major requesting a leave to be able to complete my work on an important invention. He wrote that I was working on a machine useful for the calculations and designs in the aircraft industry.

My Major looked at this letter and said, « I don't understand that. The German aircraft is the best of the world. I don't see what to calculate further on ».

Half a year later, I was freed from military service; but not for the development of computers but as an engineer in the aircraft industry.

The Z3 was completed in 1941 and was the first fully operating model.

The year 1945, with the end of worldwar II, cut off the hardware development in Germany. We were able to save only the model Z4, which we transported from Berlin in an adventurous Odyssey to Bavaria, where it was hidden in a small village in the Alpes.

Because of the unfavourable post-war conditions the hardware development was interrupted for some years and could not be continued before 1948.

In the course of this hardware development several technologies were tested and used.

At that time calculating machines normally were small units to be put on a desk. So I was psychologically prejudiced and started with mechanical constructions.

But I made a step from the traditional decimal calculating machines to real binary switching elements. This was, I think, the only attempt to make a mechanical machine based on a two-positional principle. But this technology did not work well with the exception of the storage unit and I decided to change to the electromechanical technology with its well proven relays.

Two additional lines of the hardware development may be mentioned. A model for process control and the electronic calculating devices of Schreyer. I will discuss both later on.

In the lower part of the diagram you see the parallel development of theory and software.

Right from the beginning I tried to base the whole development on a new and solid theoretical foundation. At first, the analogies between switching circuits and the calculus of propositions were discovered and a switching algebra was set up. General considerations concerning the relations between calculating and thinking followed. I realized that there is no boarder-line between these two aspects and by 1938 it was already perfectly clear to me that the development would progress in the direction of the artificial brain.

At that time I knew scarcely anything about the working method of the human brain. Even today we do not know exactly how it works. But I did not see the problem from the technological point of view, but more by analyzing the information process connected with « thinking ».

I took these ideas very seriously and this may have influenced my whole philosophy of the further development. At that time there was practically nobody to discuss with me the consequences of the possible innovations following this line.

Even 10 years later when – after the war – I became acquainted with the pioneer work on the other side of the Atlantic I sometimes had the impression that they were playing with computers like children play with matches without overlooking the whole scope of the new field. But these ideas were elaborated on paper only.

The interruption of the hardware development in 1945 allowed me to concentrate all my attention on theoretical considerations, and to develop a universal algorithmic language, which I called « Plankalkuel ». The background and the situation of that time were the reason for the special philosophy behind it.

Later on, this led to some differences and perhaps to some misunderstandings with my colleagues, for instance Bauer and Rutishauser.
But I was enjoyed to hear from the lecture of Bauer, that they studied seriously the Plankalkuel.

Light 2 gives a table of the models built in Germany during the time from 1935 to 1945. There is the main-line, beginning with the models Z1 to Z4. These were universal computers for numerical calculations. They all operated in the binary system and with the exception of Z2, with floating point arithmetic. The program was read from punched tape. I used an eight-channel input code and one address instruction code. S1 and S2 were special models for process control.

Schreyer built some test models in electronic technology.

The logical computr L1 was a test model in relay technology for programs with bits as operands.

Light 3 shows the way from the propositional calculus to the switching-algebra. I used an abstract representation for switching diagrams, which could be transferred into an arbitrary hardware. We applied it to mechanical elements with metal sheets and slots, connected by pins, to electromagnetic relays and to electronic circuits. The idea to use pneumatic and hydraulic switching elements was only pursued on paper.

Unfortunately, I never published my ideas concerning this matter. Later on I learned that there were some papers, two in German language by Hansi Piesch and Eder, and one in English language by Shannon. But I missed there the consequent confrontation with the calculus of propositions. For us the terms « And », « Or », « Not » belonged to our daily language. We really worked with them and made the step to apply the mathematical logic to the computer design. I translated the logical rules systematically into switching algebra. For instance, the principle of duality gave new insights in the working of switching diagrams.

For propositional formulas it means: change all « and » into « or » and inverse and negate all elementary propositions to get the negated proposition. The analogue rule for contact-circuits reads: change all serial connections into parallel and inverse and change all « on » contacts into « off » contacts and inverse to get the switching diagram representing the inverse of the given circuit.

As a result of this practice the logic represented by the hardware was very sophisticated, using contrived micro-programming depending on complicated conditions, conditional branching, and so on.

So, switching algebra was consequently applied in all the computers we constructed. When Schreyer changed to the electronic technology he, first, had only to design the switching elements corresponding to the three propositional operations: conjunction, disjunction, and negation. After that he was able to translate one to one the already proven diagrams for the electromechanical machines.

Light 4 shows some general aspects of the computer-architecture as we call it today.

The machine of Babbage already had the combination of the Arithmetic Unit and the Storage. Both were directly controlled by a set of punched cards, providing a special place for a hole for each storage-cell.

The machines Z1 to Z4 correspond to this concept, but, contrary to Babbage, they used coded addresses and a Selecting Unit.

Both, the computer of Babbage and my machines, had no conditional orders and feedback in the Program Unit.

The first computers in Germany were exclusively designed for numerical calculations and the limited financial basis and short time available for the construction did not allow any special features. Besides that, the users of the computer did not see the necessity for a more sophisticated logical design in these days.

But on paper there was no limit for further ideas, even during the war.

The idea of general calculating or information processing, as we say today, induced me to consider that the program, too, is information and can be processed by itself or by another program. This general concept was elaborated in all consequences in the Plankalkuel.

In hardware it means that we not only have a controlling line going from left to right, but also from right to left. I had the feeling that this line could influence the whole computer development in a very efficient but also very dangeours way. Setting up this connection could mean making a contract with the devil. Therefore, I hesitated to do so, being unable to overlook all the consequences, the good as well as the bad.

So first I concentrated on theory. This led to the Plankalkuel. It is interesting to follow the further development. My colleagues on the other side had no scruples about the problem I just mentioned. John v. Neumann and others constructed a machine with a storage for all kinds of information including the program.

This idea may have been trivial, as soon as the programs were binary coded and there existed storage units for storing any binary coded information. This requirement was already fulfilled by the machines Z1 to Z4 and others. Besides this, the idea of storing the program was already mentioned for instance in one of my patent applications in 1936. Other pioneers may have had the same idea rather early. I think it was the special organisation of the machine of John v. Neumann which opened the door for universal calculating. He gave the signal « all clear » for the scientists but for the devil, too. This concept was adapted to the situation around 1945 and was very efficient, especially for numerical calculations.

My own designs for future machines on paper were more structured with instructions stored independently and special units for the handling of addresses and subroutines nested in several levels.

I believe that other pioneers, too, have been in a similar situation. The theoretical work on paper, published or not, mostly is going far beyond the machines really constructed.

Now let's return to the situation in Germany during the war. In our situation the only realistic way to process a program by itself was to build a separate computer for this purpose. Thus, the construction of the computers for numerical calculations could be continued without drastic modifications. We called this type of machine « Planfertigungsgerät », that means a special computer to make the program for a numerical sequence controlled computer. This device was intended to do about the same sophisticated compilers do today. But in 1945 we had to stop this interesting development.

I intended to proceed in the following steps:

1) Converting algebraic formulars written in traditional form into a sequence of orders for the computer.
2) Inserting subroutines in a mainprogram in several levels including the changing of corresponding addresses.
3) Development of programs for determinants, matrices etc. of different order and arrangement of non-zero-elements. This means mainly the processing of addresses.
4) Analysing whole technical systems like frame works for constructional engineering and others and setting up the program for the numerical calculations for such a system with varying parameters like measures, forces and so on.

This led, to rather complicated and sophisticated evaluations, using the calculus of relations, predicates etc.

After this general review I want to discuss some details.

Light 5 shows the block diagram of the models Z1 to Z4.

We have a punch, operated by hand to make the program-tape. The program unit is tape controlled and gives orders to the Computing unit and the addresses for the Selecting unit of the storage. Input and output-units are directly connected to the arithmetic unit.

Light 6 shows the details of the input and output of the model Z3. On the lower part you have the arrangement of the keys. There are the keys for 4 decimal digits and the sign +, —. Relative to the decimal digits the point can be set by pressing one of 20 keys. So the number is represented in the same way as you write it, i. e. without separate data for the exponent.

On the right side you have the keys for the operations. Besides the normal 4 arithmetic operations we have some special operations, like square-root and others.

In the output on the upper side the numbers are represented by lamps arranged corresponding to the input-keys.

Light 7 shows some details of the floating point arithmetic and the storage of the model Z3.

The sign, the exponent and the mantissa are handled in separate units.

On *Light 8* we have a survey of the programs, which were run on the computer Z3.

Apart from some general mathematical programs, there was a program for the calculation of a determinant with complex elements and with variables p, q, r which was of special interest for the engineers in the field of aerodynamics. (Calculations of vibrations, Flatterrechnung).

Light 9. I mentioned earlier the special purpose computers S1 and S2. In an aircraft factory guided missiles were being manufactured on an assembly line. These missiles had to fly very precisely in order to be remotely controllable. Therefore, every missile had to pass a special measuring station, where the deviations from the aerodynamic symetry were measured at about 100 points with measuring clocks. These data were the input for a computer, programmed by rotary switches, which calculated the necessary corrections of the positions of the wings. A sequence of some hundred additions, multiplications etc. was executed automatically. This computer was in operation around the clock for two years during the war.

Light 10. In a second version the measuring devices were read automatically via rotary switches, which transferred their positions into the computer. Today we speak of analog to digital conversion and process control.

Light 11. Now let's take a lock at the work of Schreyer. I already mentioned that the switching algebra allowed us to design a computer in abstract diagrams, which could be transferred into a special hardware-technology, for instance electromechanical relay circuits.

Following this idea, Schreyer first designed and constructed the circuits corresponding to the operations of the propositional calculus. Today it is commonplace to speak of Nor- and And-circuits etc. But please remember that the first electronic calculator, the ENIAC, built some years later, worked by simulating decimal gears. Schreyer could not use the semi-conductor-technology at that time (1937), as we do today.

He used a special type of tube with two parallel grids with the same characteristic.

It was interesting for me to learn from the lecture of Randell, that in the « Collossus » similar ideas were applied. There were used circuits corresponding to the propositional operations, too.

Light 12 shows a computing device built by Schreyer during the war. It is a ten digit parallel binary calculator specialized on transferring a 3-digit decimal number into a ten digit binary number. The model was ready for tests in 1944.

During the war we submitted the concept of an electronic computer with 2,000 tubes to the German Government Research Authorities, but their reaction was negative. We would never have attempted to construct a computer with 18,000 tubes and I admire the heroism shown by. Mr. Eckert and Mauchly.

Most of the machines we constructed in Geımany until 1945 were destroyed by airraids. Only the model Z4 could be saved. In 1950, after some improvements, it was leased to the Eidgenössische Technische Hochschule in Zürich, Switzerland. It was so reliable that it was customary to let it work through the night unattendedly. I remember the good cooperation with Stiefel, Speiser, and Rutishauser.

Not before 1950 could we continue the development of computers after an interruption of some years. Together with two friends I started the ZUSE KG near Bad Hersfeld, Hessen. A series of new models followed, but I think this is less interesting for this conference.

At first, the Optical Industry were our customers, then the Authorities for Land-Surveying and the Univeısities. Besides of the last there was no sponsering on assistance by the government.

Perhaps I may mention the development of an automatic plotter with high accuracy about 1958. Today the name of ZUSE KG is cancelled and my former factor is owned by Siemens AG.

Independant of this development and without any knowledge of each other Dr. Dirks during worldwar II constructed a computing device for commercial purposes using a rotating magnetic storage. This was a forerunner of the later developed magnetic drums and discs.

Concerning these two developments it would be interesting to study the priorities in the field of rotating magnetic storage-devices, especially in comparison with paralell constructions at other places. About 1947 another German pioneer, Dr. Billing, constructed a magnetic drum for the use in a computer.

Professor Bauer already gave us further information on the other interesting developments after 1950 in Germany.

Now I will give you some details on the algorithmic language, called Plankalkuel. I told you already that with the end of the war we had to stop our hardware-development and so I concentrated on theoretical investigations. Of course, I only can give you a general survey in this short lecture. Those who are more interested in this matter may study the new edition of the Plankalkuel which has recently been published in English.

The first principle of the Plankalkuel is

Data processing begins with the bit.

I am sorry but even today I have difficulties with some of my colleagues to justify this assertion. Since about 20 to 30 years most of the computers only slowly and gradually they overcame the priority of the numerical calculations. Thus, in our conventional computers the bit is only tolerated as a boolean object for controlling conditional branching and so on.

Contrary to this aspect, the Plankalkuel is fundamentally based on the bit.

Light 13 shows how arbitrary structures may be defined by composing bits, strings of bits and so on.

From mathematical logic I took the instruments of the calculus of propositions, of relations and the predicate-calculus.

Light 14 shows two examples for the application of relations to practical engineering. The structure of a frame and the measurement of a girder both are represented by lists of pairs which may be objects for structural calculations.

Light 15 shows how such pairlists can be split into components (lower figure).

Above you see the special kind of representation of the objects. All data attached to an object are put together, but in separate lines. The first line says only « Variable » or something like that.

The second line contains the indices, completing the name of the object. The third line gives the identification of the component, you wish to select from the given object. This selection can be done in several levels, so that any part of the whole data structure up to the last bit may be handled separately. The last line contains the information about the structure of the selected object or component respectively.

For this purpose we have a systematic code for every structure, for instance So for the bit, and so on.

In other languages this information is given separately beforehand in a declaration.

Light 16 shows the main syntactic features of the Plankalkuel.

Every program is a module in itself. There is a « Randauszug », meaning an « Input-Output-Specification » which gives the relations of the program-module to the environment.

Subroutines are defined in the following way:

Every result of an arbitrary program can be used as a function of some variables. So this language gives the exact logical content of the program but not additional information concerning the details of the implementation like

« Call by value »
« Call by reference » and so on.

The objects of a program may be input-values (variables) local values, results, constants and some auxiliary values for controlling iterative processes or bounded variables for the operators of the predicate calculus.

We then have some special symbols for statements, conditional orders and iterative cycles.

There is also an End-symbol, FIN, which corresponds in a limited sense to the GO TO of other languages. But according to the modular organisation of the Plankalkuel there is no danger of applying it in a confusing manner.

It was interesting for me to test the efficiency and the general scope of the Plankalkuel by applying it to Chessproblems.

I learned playing Chess especially for programming Chess problems. This field seemed to me suited for the formulation of rather sophisticated data structures, nested conditions, and general calculations.

Light 17 shows some special types of data structures defined for this purpose.

Please let us take a look, for instance, at the « field occupation ». For the description we need a list of 64 specifications of the type of occupation of any point and so on.

On the following pages I only will give you a general impression of what programs written in this language look like.

The program is called PΔ160 and evaluates whether the White King is in checkmate or stalemate.

The Input is a « field occupation » and the output are two bits, one for each of the wanted predicates. Then follows a kind of comment, which is not a part of the proper program.

Lights 18a and *18b* show some preliminary comments on the meaning of the used objects.

Light 19 shows the proper program. You see there some operators of the predicate calculus like « this one », or « those which », and some conditional orders.

Looking at the arrangement of the formulas you have the impression of a two-dimensional language. But this is not really so. This form facilitates the reading of the program for the user. For implementation it can be stretched into a linear representation without changing the structure of the program.

Behind the Plankalkuel there is a special philosophy basing on my early conviction, that there is a steady way from simple numerical calculations to high-level thinking processes. In order to test the universality of this language I applied it for several extraordinary fields. Thus, for instance, I made some steps in the direction of symbolic calculations, general programs for relations, or graphs, as we call it today, chess playing and so on.

Here you will miss the normal numerical calculations like linear equations etc. Some general considerations showed me that these are rather trivial in comparison to the other fields selected by me for the further investigations. This later on led to some misunderstandings, when 10 years later just these numerical calculations became popular. The Plankalkuel was critized as a special logical language going too far ahead of the problems then to be solved.

So my concept may have been too advanced at that time. But looking at the present situation I come to the conclusion that it would have been better to base the hardware and software development of the computer on the philosophy of the Plankalkuel from the beginning. Surely, we all assembled here can be proud of the achievements basing on our pioneer activity. Nevertheless, data processing is not yet fully emancipated. There is some confusion and trouble in the field. I think some problems could not yet be solved satisfactorily.

On one side there are sometimes too many mathematicians influencing the computer science in a worldly innocent manner. On the other side, relatively primitive methods and programming languages are still applied in practice.

At the end of my lecture would you please allow me to say something about my more recent ideas.

In the years from about 1948 to 1964 I was occupied with organizing the development of the ZUSE KG. Unfortunately, as a manager I had hardly the time for profound theoretical considerations.

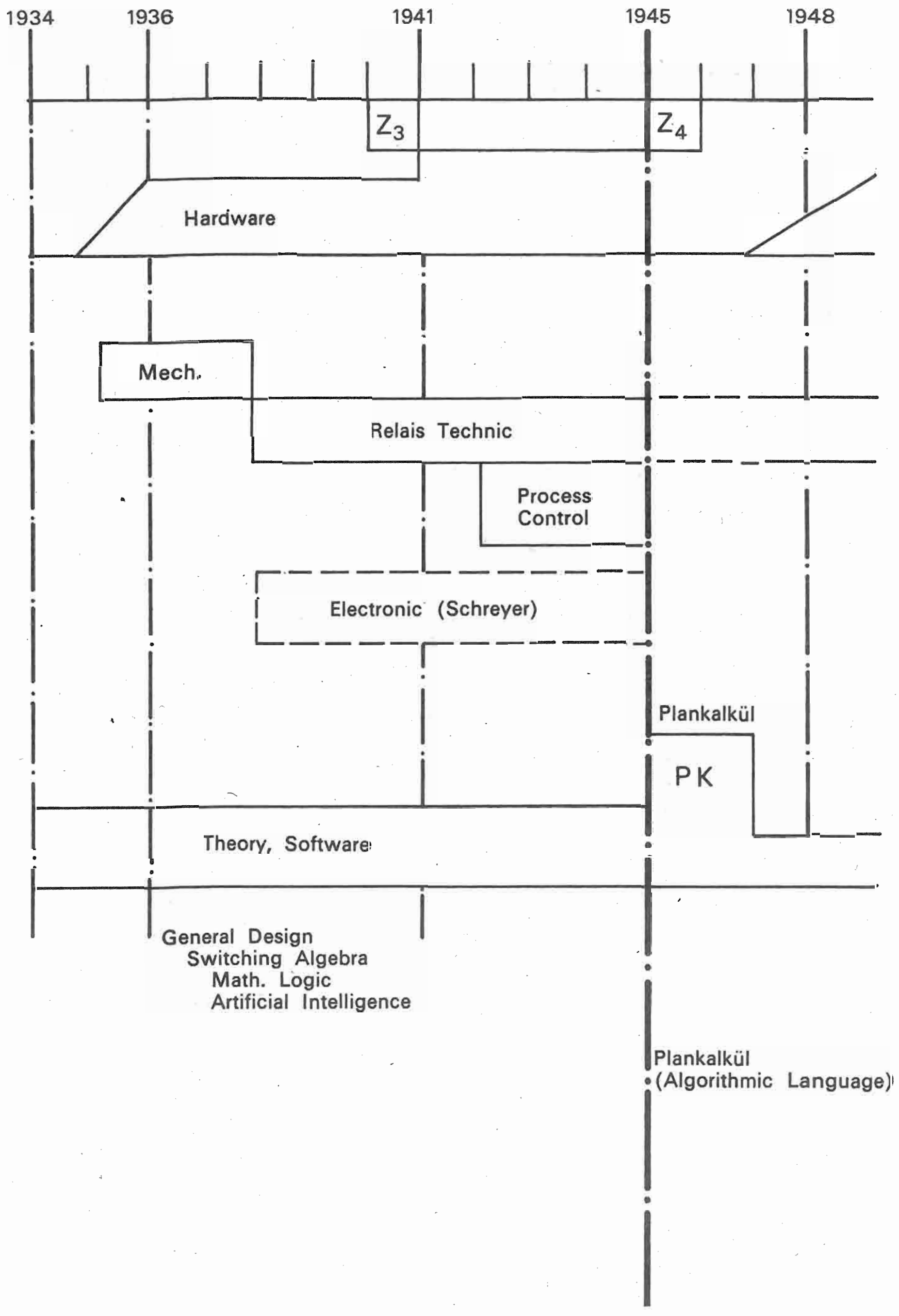
But in the last 10 years I have been able to continue my life as a scientist again. The main objectives are new investigations on the Plankalkuel in comparison with other algorithmic languages. According to my subjective opinion the Plankalkuel is not only interesting from a historical point of view but is also of great significance for solving present day problems. The situation in the field of algorithmic languages is rather confusing. Structured programming is only a partial solution. Some features of the Plankalkuel may help to solve the problems.

Another field of my research are « Self-reproducing Systems ». But I see the problem not from the mathematical point of view, like for instance John v. Neumann, but as an engineer. It may be better that there is nearly no support for the propulsion of such ideas. Perhaps the devil is behind it, too. But speaking about this would go far beyond the frame of this conference.

Almost from the beginning of my work in the field of computing I had the idea of paralell information processing, for instance with cellular automata. This induced me to apply this idea to theoretical physics. I developed some ideas concerning the « Rechnender Raum » meaning something like « Calculating Cosmos ».

This general idea is of increasing interest in other places, too, for instance in the United States. I am convinced that such investigations will gain broader attention in the future also from physicists.

But most of these ideas are as crazy today as the idea of the computer was 30 to 40 years ago. Therefore, it is my fate to perform these investigations on a very limited scale. Nevertheless, I feel happy to be a pioneer until the end of my days.


1934
1936
1941
1945
1948
Z_3
Z_4
Hardware
Mech.
Relais Technic
Process Control
Electronic (Schreyer)
Plankalkül
PK
Theory, Software
General Design
Switching Algebra
Math. Logic
Artificial Intelligence
Plankalkül
(Algorithmic Language)

Light 1

Model constructed by Zuse until 1945

Model	Year	Technology [1]			Point	Programm
		Arithm. Unit		Binary System	Fixed Floating	
Z1	1938	M.	M.	+	float.	Punched Tape
Z2	1939	R	M	+	fixed	Punched Tape
Z3	1941	R	R	+	float.	Punched Tape
Z4	1945	R	M	+	float.	Punched Tape
S1	1942	R	R	+	fixed	Rotary Switches
S2 [2]	1942	R	R	+	fixed	Rotary Switches
Schreyer	1938	E				
Schreyer	1944	E		+	fixed	
Log. 1	1944	R	R	Logical Computer		Punched Tape

[1] M Mechanical Technology
R Electromechanical Relay-Technology
E Electronic Technology
[2] Process Control

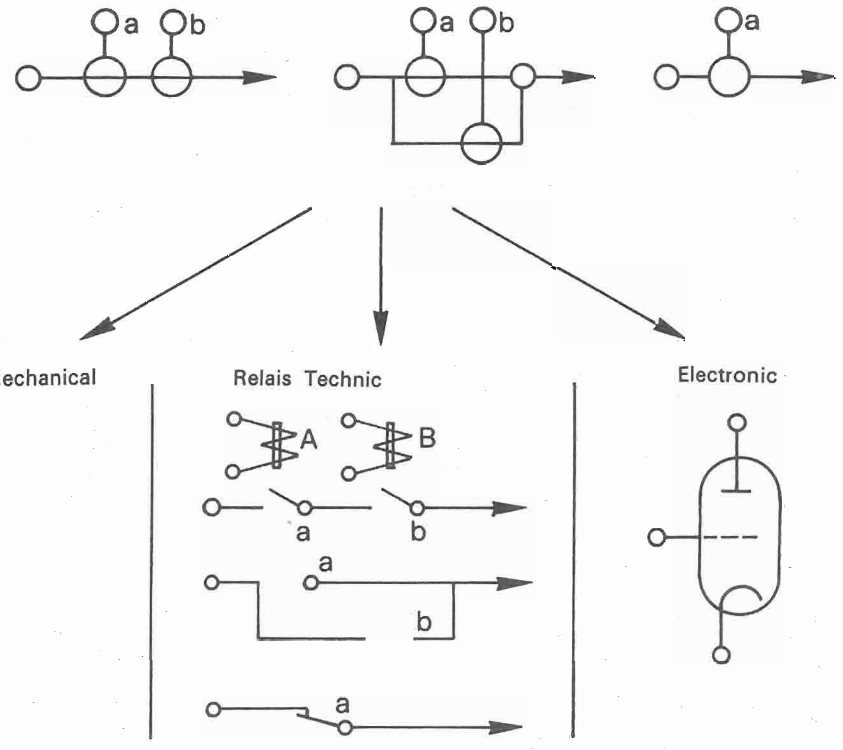
Light 2

Propositional Calculus

a ∧ b, a ∨ b, a

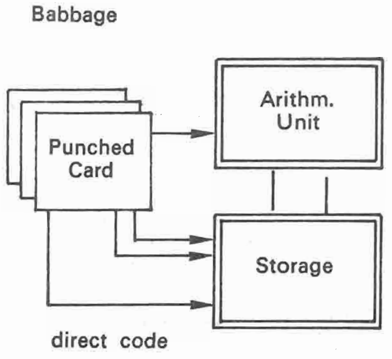
Switching Algebra

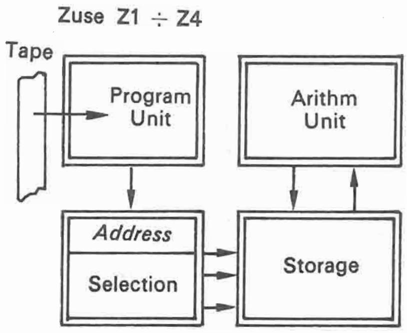
Abstract representation

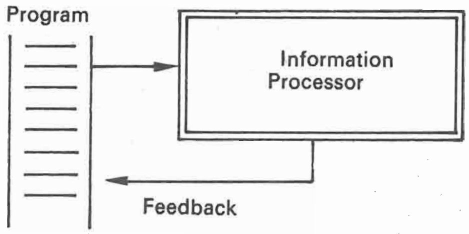


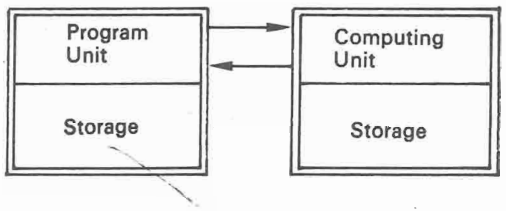
Universal Information Processing

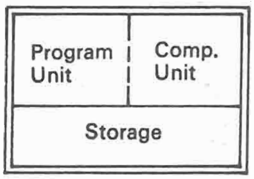
Light 3









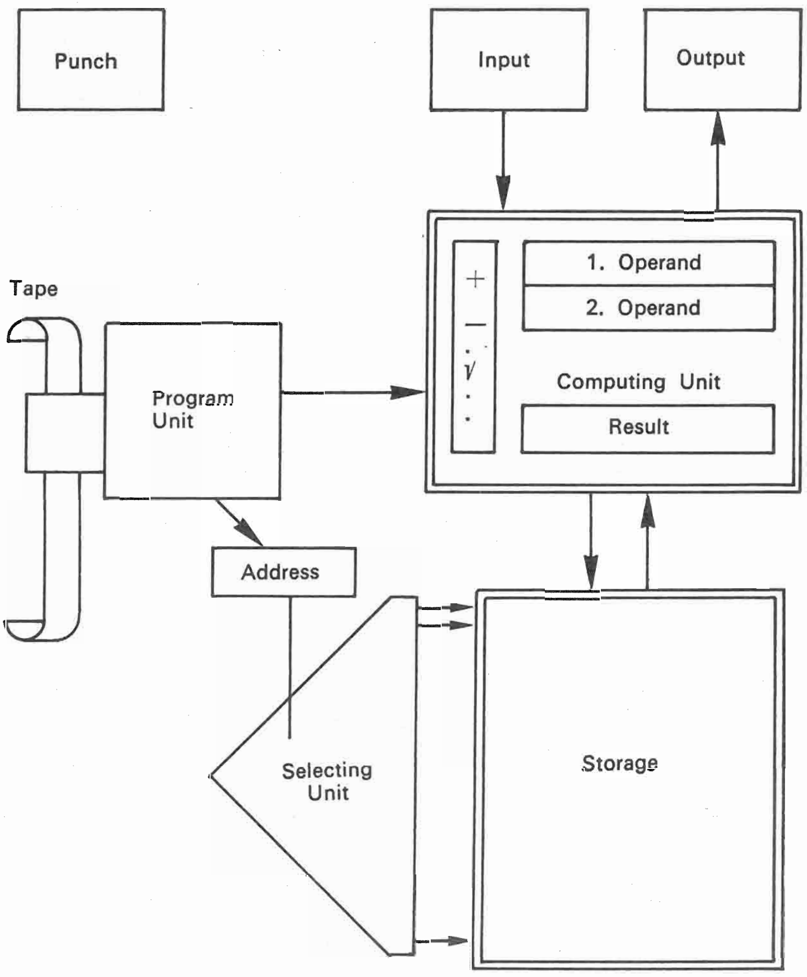


Logistic Computer
General Information Processing
Program-Generator } Planfertigungs-Geräte
Compiler
Associative Memory

1945

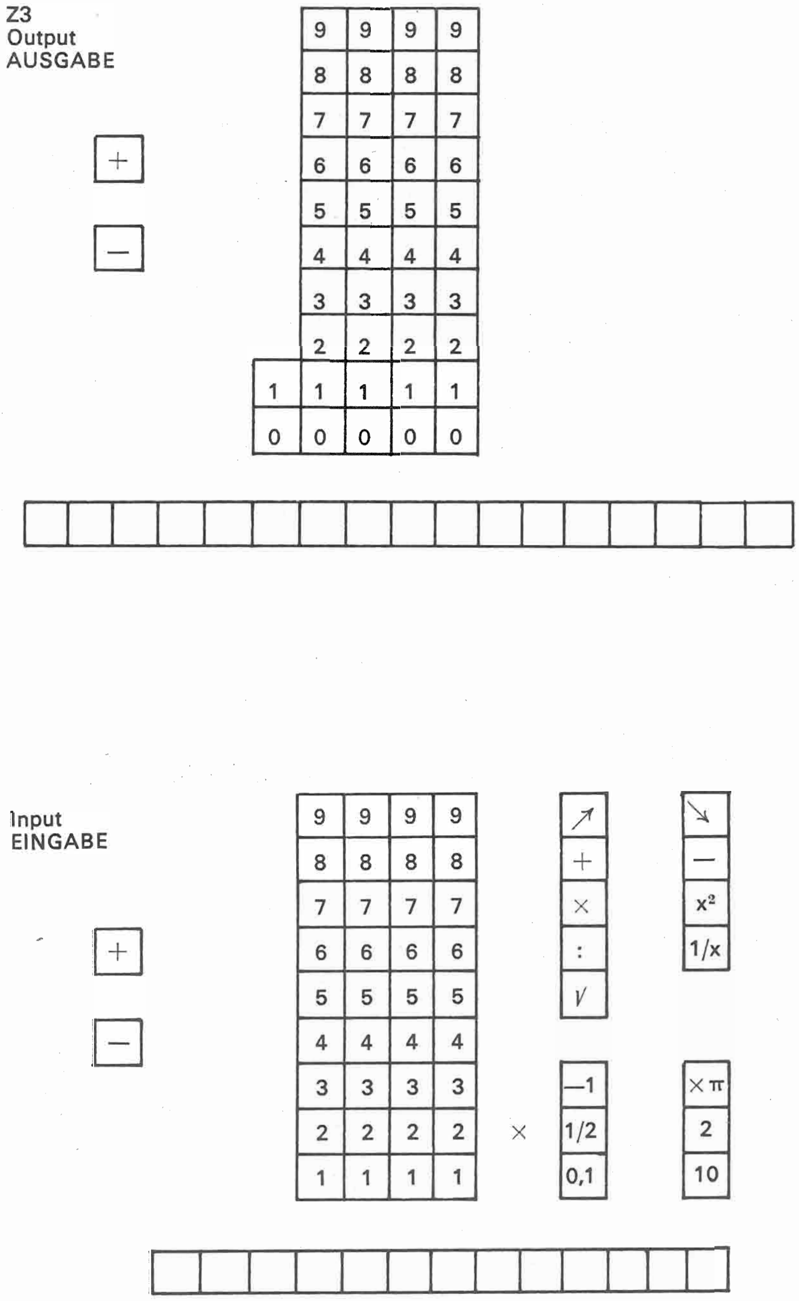
Universal Algorithmic Language
Plankalkül

Light 4

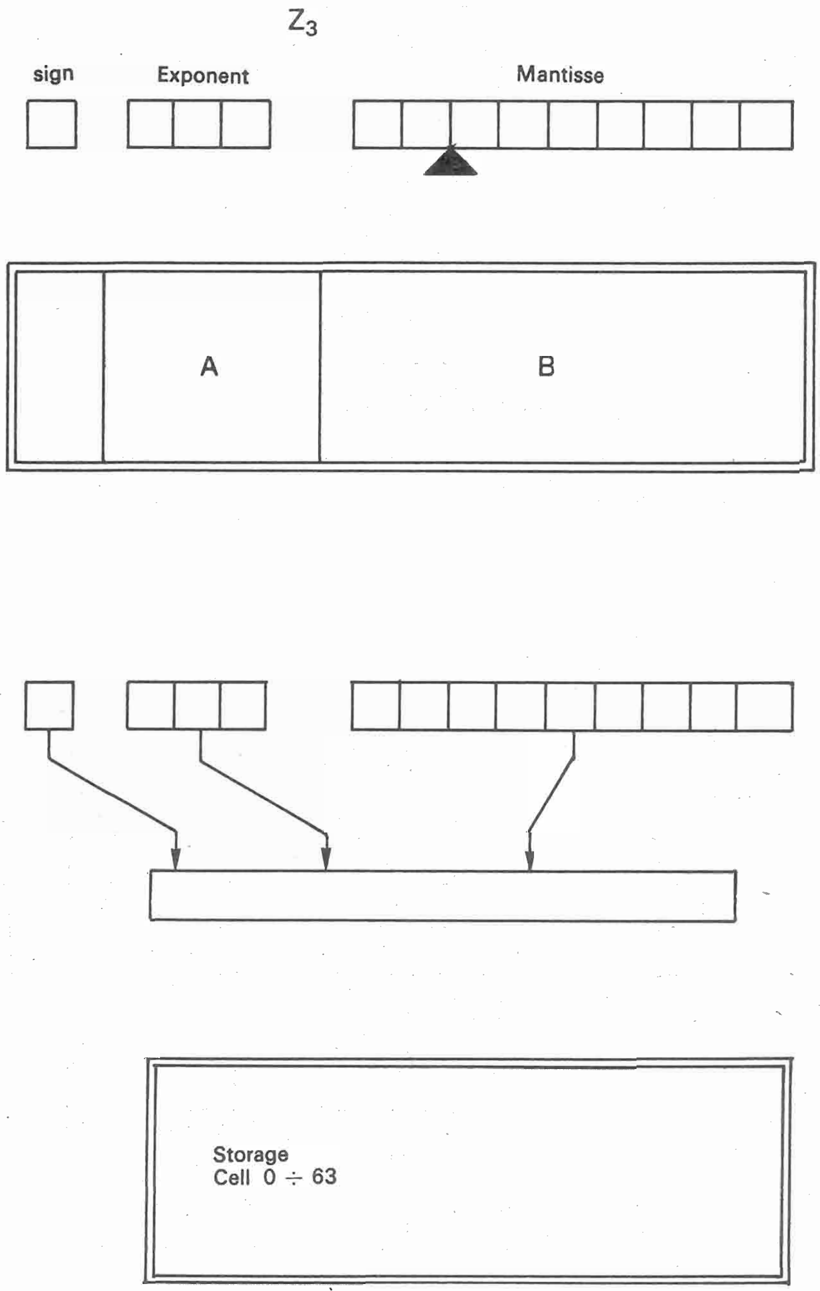


Model Z1 ÷ Z4

Light 5


Z3
Output
AUSGABE
+
−
9 9 9 9
8 8 8 8
7 7 7 7
6 6 6 6
5 5 5 5
4 4 4 4
3 3 3 3
2 2 2 2
1 1 1 1 1
0 0 0 0 0
Input
EINGABE
+
−
9 9 9 9
8 8 8 8
7 7 7 7
6 6 6 6
5 5 5 5
4 4 4 4
3 3 3 3
2 2 2 2
1 1 1 1
×
↗
+
×
:
√
−1
1/2
0,1
↘
−
x²
1/x
×π
2
10

Light 6



Light 7

Programs, calculated with the Computer Z3
(1941-42)

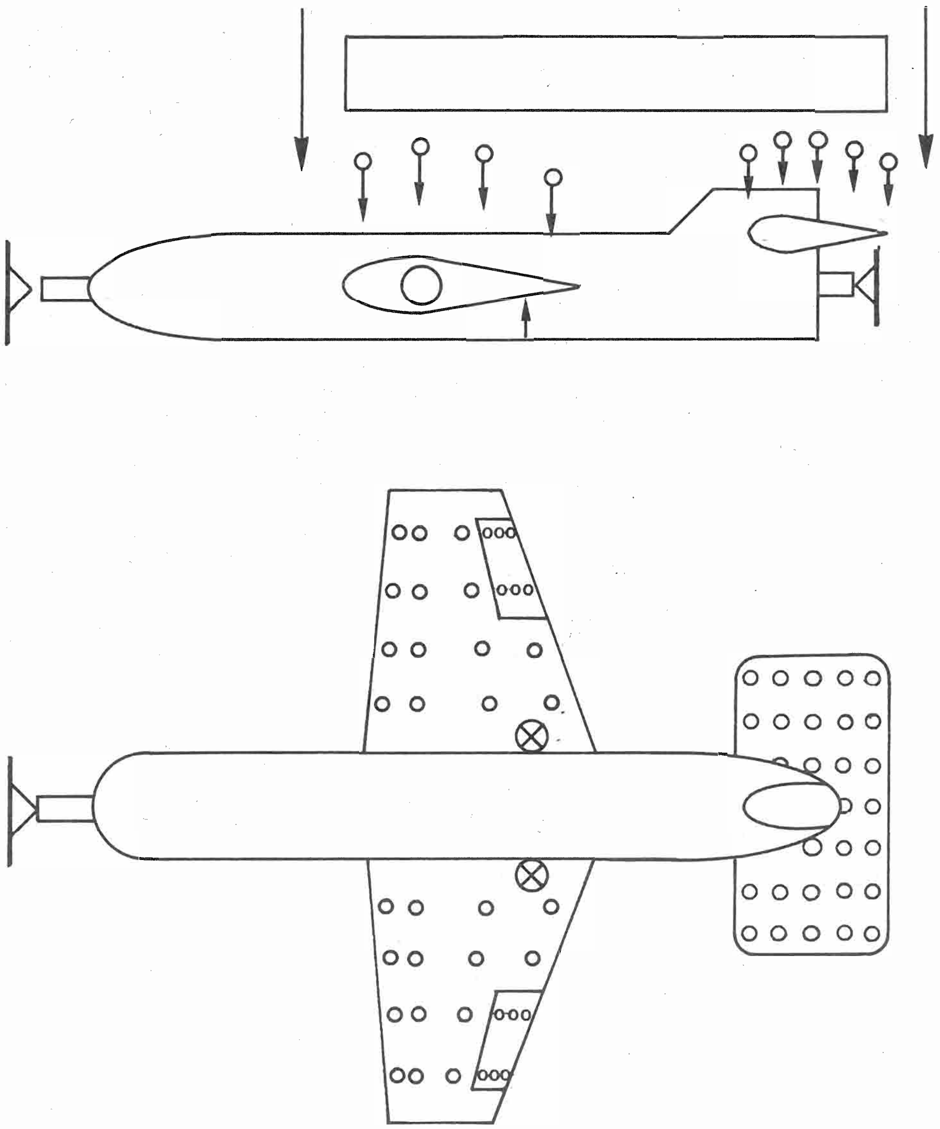
1) General Test-Programs (Testing of Arithmetic Operations and Storage-Cells)
2) Linear Equations (To the Order 3)
3) Quadratic Equations
4) Küssner-Determinant

$$\Delta = \begin{vmatrix} a + ia' + \frac{P}{V^2} & b + ib' & c + ic' \\ c + ic' & d + id' + \frac{q}{V^2} & f + if' \\ g + ig' & h + ih' & k + ik' + \frac{r}{V^2} \end{vmatrix}$$

V is a function of ω (reduced frequence)

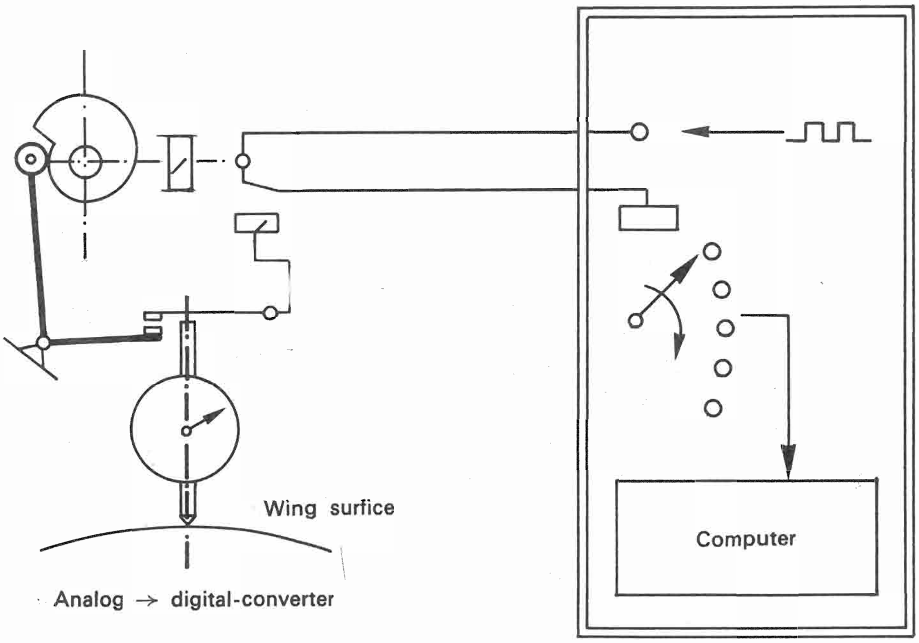
5) Miscellaneous

Light 8

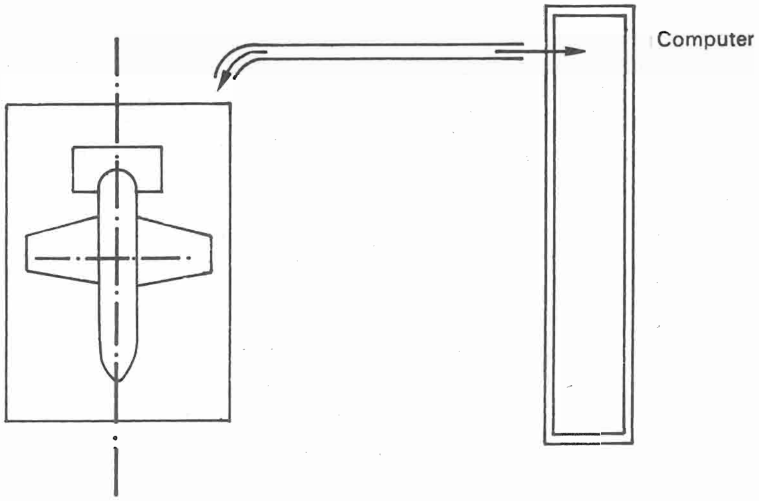


Process Control with Zuse S2

Light 9

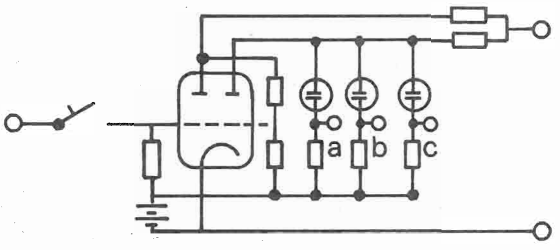


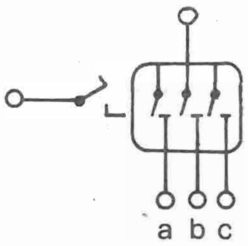
Analog → digital-converter

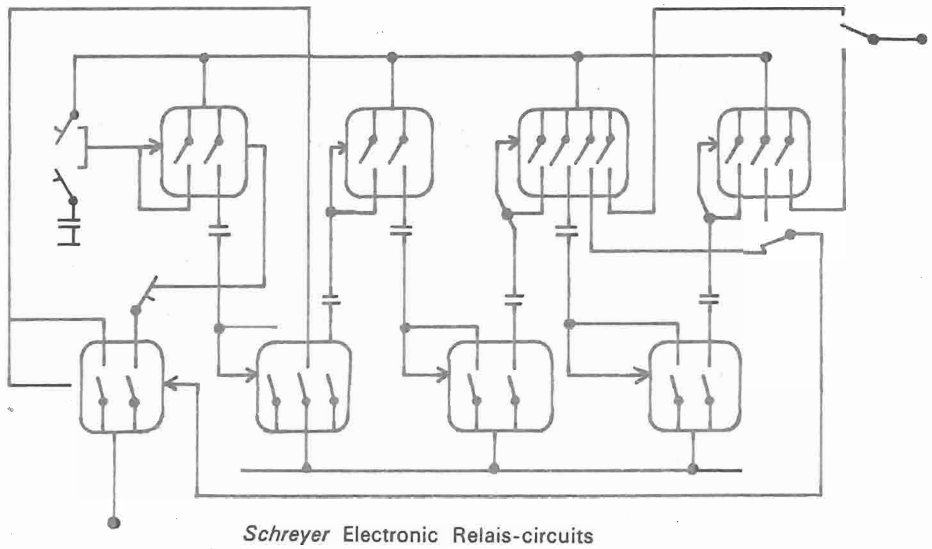


Process-Control with
Model Zuse S2

Light 10







Schreyer Electronic Relais-circuits

Light 11

Schreyer

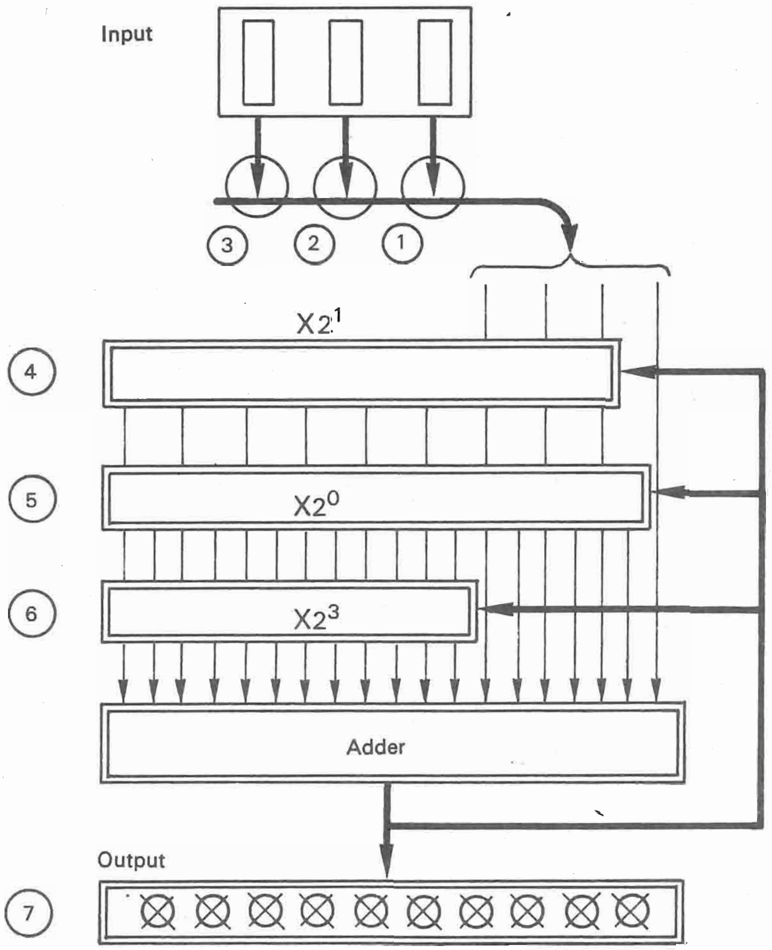
Electronic

Computer

Decimal to
Binary conversion

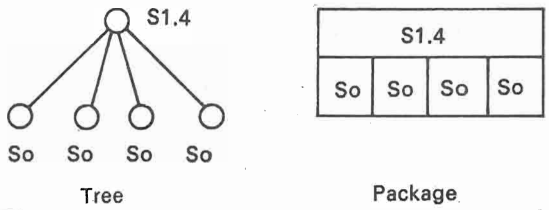
769

Step				
1	1		7	LLL
2	4	6	× 10	LLLo LLLooo
3	5	2	70 + 6	LoooLLo LLo
4	4	6	76 × 10	LooLLoo LooLLooo LooLLooooo
5	5	3	760 + 9	LoLLLLLooo LooL
6	7		769	LLooooooooL

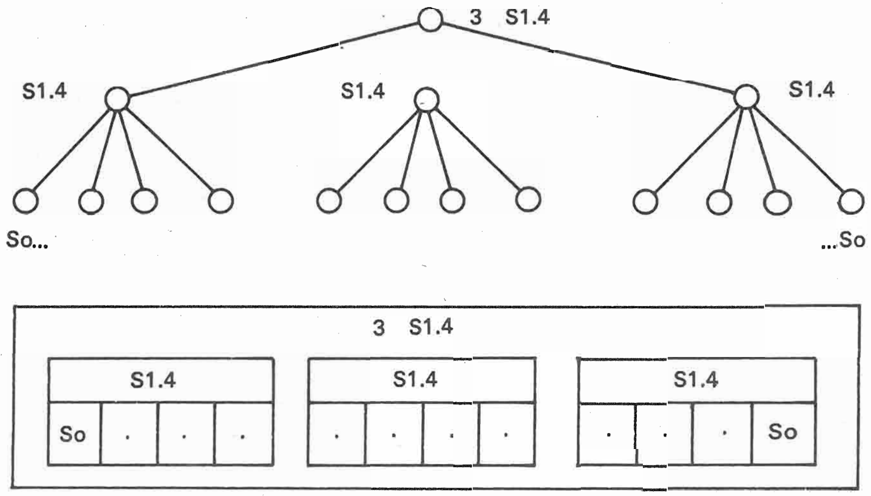


Light 12

S1.n = n × So

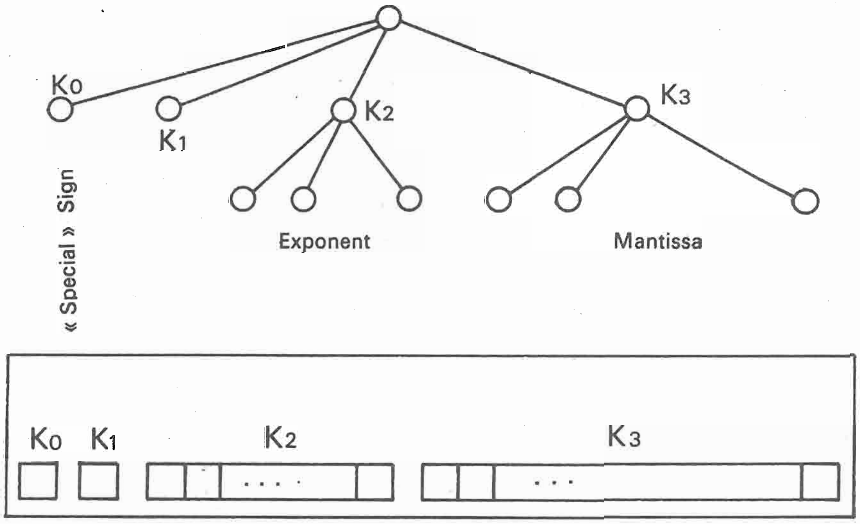


Decimal number with 3 digits



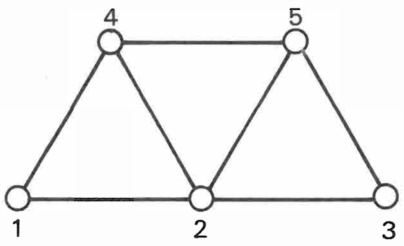
Floating-point binary number

A14.0 = (So, So, S1.8, S1.20)

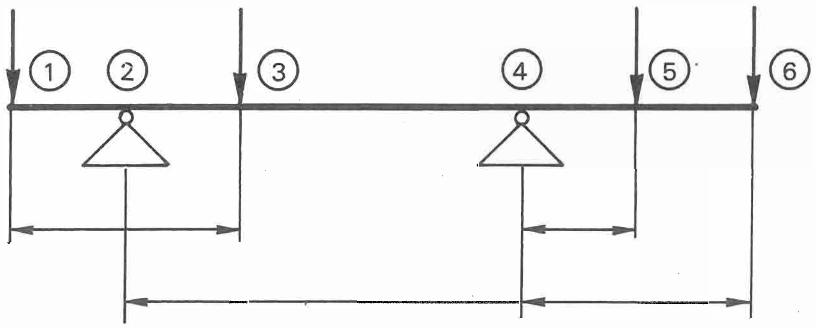


Light 13

List of pairs [Graph]

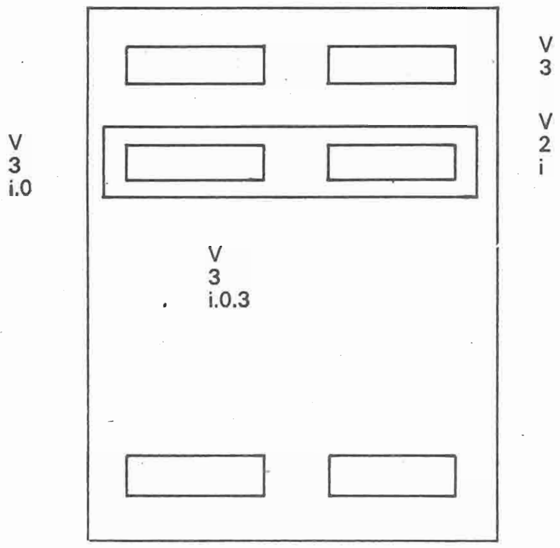


1-2
2-3
1-4
2-4
2-5
3-5
4-5



1-2
2-3
2-4
4-5
4-6

Light 14



Light 15

Input-Output Specification (Randauszug)

$$\begin{array}{l|l} V & R(V_o, V_1) \Rightarrow (R_o, R_1, R_2) \\ S & \quad m\times\sigma \;\; n\times\sigma \qquad (m+n)\times\sigma \;\; 1.n \;\; 0 \end{array}$$

Input: V_o list with m elements V_1 list with n elements

Output: R_o list with m + n elements R_1 binary number R_2 predicat

If the program P3.7 has this input-output specification, then

$$\begin{array}{l|l} V & R3.7(Z_9, Z_{11}) \Rightarrow Z_{12} \\ S & 1.n \quad m\times\sigma \quad n\times\sigma \qquad 1.n \end{array}$$

means: the result R_1 of the program P3.7 applied to the objects Z_9 and Z_{11}

is assigned to Z_{12}. (Z_9, Z_{11} and Z_{12} are local objects).

Objects of a program

V	input values
Z	local values (Zwischenwerte)
R	results
C	Constants (Reference level o)
i, j,	local values for the variation of indices and so on
x, y	bounded variable used in the predicate calculus:

$\Rightarrow$	corresponds to $Z+1 \Rightarrow Z$	$:=$
$\rightarrow$	corresponds to for instance $V_o > V_1 \rightarrow V_o \Rightarrow R_o$	IF... THEN...
FIN	End symbol	
$Z_o \rightarrow$ FIN	conditional end symbol	
W( )	iterative program section	

Light 16

Data-Types (Daternarten) for Chessprograms (extract)

AΔ1	= S1.3	Coordinate
AΔ2	= 2 × S1.3	Point
AΔ3	= S1.4	Specification of occupation
AΔ4	= (AΔ2, AΔ3)	Occupation of a special point
AΔ5	= 64 × AΔ3	Field occupation
	CΔ5	Field occupation at the start
AΔ6	= 64 × AΔ4	Field occupation as AΔ5, but supplemented by the point coordinates
AΔ9	= (AΔ5, So, S1.4, AΔ2)	Actual situation of the game
	Ko AΔ5	Field occupation
	K1 So	« White has the move »
	K2 S1.4	Conditions for Castling
	K3 AΔ2	Aiming point of the last move
CΔ9		Situation at the start
AΔ11	= (AΔ2, AΔ2, So)	Specification of a move

Light 17

PΔ160 *Conditions for checkmate or stalemate*

Input-Output

$$\begin{array}{c|cccc} & R(V) & \Rightarrow R, & R) \\ V & o & o & 1 \\ A & \Delta 5 & o & o \end{array}$$

$\begin{array}{c\|c} & V \\ V & o \\ A & \Delta 6 \end{array}$	Field occupation
$\begin{array}{c\|c} & R \\ V & o \\ S & o \end{array}$	« The white King is checkmate »
$\begin{array}{c\|c} & R \\ V & 1 \\ S & o \end{array}$	« The white King is stalemate »
$\begin{array}{c\|c} & Z \\ V & o \\ A & \Delta 4 \end{array}$	Point occupation
$\begin{array}{c\|c} & Z \\ V & 1 \\ A & \square \times \Delta 4 \end{array}$	List of the pieces attacking the white King
$\begin{array}{c\|c} & Z \\ V & 2 \\ A & 9 \end{array}$	Number of the pieces attacking the white King (A9 = natural number)
$\begin{array}{c\|c} & Z \\ V & 3 \\ S & o \end{array}$	« The white King is checkmate or stalemate »
$\begin{array}{c\|c} & Z \\ V & 4 \\ S & o \end{array}$	« No white piece, except the King, is able to move »

P 160

(1) Evaluation of the point, occupied by the white King (Z_o).

(2) Formation of the list of the pieces attacking the white King (RΔ129, Z_1)

(3) The number of the attacking pieces influences the further evaluations.

(4) If the King is not attacked ($Z_2 = 0$) or the King is able to make an evading move (RΔ148), then the program is finished (FIN)

Light 18 a

(5) If the King is attacked by more than one piece ($Z_2 > 1$), then we have checkmate or stalemate (Z_3). (The case of an evading move is already taken into account by (4)).

(6) If the white King is attacked by one piece ($Z_2 = 1$), then we have checkmate or stalemate if

(7) the attacking piece cannot be captured by White without discovering check (RΔ142)

(8) and in the case of free points between the attacking piece and the King (RΔ19)

(9) there is no white piece being able to move to one of these points (RΔ152)

(10) Investigation, if there are white pieces, except the King, being able to move (RΔ150). If this is not the case, then Z_4 is positive.

(11) Condition for checkmate (R_o).

(12) Condition for stalemate (R_1).

Light 18 b

```
PΔ160

       (1)
  |    x     x∈V  ∧ x   ==——+—    ⇒ Z
V |            o                    o
K |                1
A |    Δ4      Δ6  Δ3               Δ4

       (2)
  |    x     RΔ129(V , x , Z ) ∧ x     ⇒ Z
V |                o       o              1
K |                    o   o   1.3
A |    Δ4          Δ6  Δ2  Δ2  o         □ × Δ4

       (3)
  |    N(Z         ) ⇒ Z
V |      1             2
K |
A |      □ × Δ4        9

       (4)
  |    Z = 0 ∨ RΔ148(V ) →  (—,—) ⇒ (R, R)  |  FIN
V |    2             o                o  1  |
K |                                         |
A |    9      o      Δ6                 o  o  |

       (5)
  |    Z > 1 →  + ⇒ Z
V |    2            3
K |
A |    9            0

       (6)             (7)
  |    Z = 1 →            ¬ RΔ142(V , Z  )
V |    2                          o   1
K |                                   o.o
A |    9                          Δ6  Δ2

                       (8)               (9)
  |                    ∧ + RΔ19(Z , Z ) → RΔ152(V , Z , Z  )  ⇒ Z
V |                             0   1            o   o   1       3
K |                                 o.o              o   o.o     o
A |                            Δ2  Δ2            Δ6  Δ2  Δ2

       (10)                      (11)                 (12)
  |    RΔ150(V ) ⇒ Z        |      Z ∧ ¬Z ⇒ R    |      Z ∧ Z ⇒ R
V |          o     4        |      3     4   o    |      3   4   1
K |                         |                     |
A |          Δ6    o        |      o     o   o    |      o   o   o
```

Light 19