

Konrad Zuses Rechenmaschinen

Die Erfindung des modernen Computers

Yuan Bi
Technische Universität München
yuan.bi@tum.de

Andreas Schönleben
Technische Universität München
andreas.schoenleben@tum.de

ABSTRACT

Diese Seminararbeit beschäftigt sich mit Konrad Zuse, seinen wichtigsten Beiträgen zur Entwicklung des modernen Computers und wie diese geschichtlich einzuordnen sind.

Keywords

Konrad Zuse; Z1; Z3; Plankalkül

1. BIOGRAPHIE ZUSES

1.1 Jugend und Studium

Konrad Zuse wurde am 22. Juni 1910 in Deutsch-Wilmersdorf, welches heute Teil von Berlin ist, geboren. Er besuchte zunächst das Gymnasium in Braunsberg und später, als sein Vater dort Oberpostmeister wurde, in Hoyerswerda. Neben seiner Begeisterung für das Zeichnen, zeigte er schon früh technisches Interesse. Beispielsweise baute er mit Hilfe seines Stablbaukastens in seinem Zimmer einen großen Greiferkran und andere mechanische Modelle. Ein weiteres Beispiel für sein frühes planerisches Interesse ist das Verkehrssystem, dass er als Schüler für eine Millionenstadt entwarf. Dieses basierte auf dem 60°-System und beinhaltete weitere Konzepte wie etwa Expresszüge, die ständig in Bewegung waren und an die anderen Züge während der Fahrt zum Umsteigen angekoppelt wurde. Seine Skizze ist in Abbildung 1 zu sehen.

Im Anschluss an das Abitur begann er ein Studium des Maschinenbaus. Er fühlte sich durch allerdings schnell durch die Starrheit der vermittelten Inhalte eingeschränkt. Er wechselte deshalb zur Architektur, da er sich hiervon bessere Möglichkeiten versprach, seine Kreativität einzusetzen. Auch dieser Studiengang entsprach aber nicht seinen Erwartungen und er studierte schlussendlich Bauingenieurwesen. Auch während des Studiums experimentierte er z. B. mit Fotografie, baute einen Warenautomaten mit Geldrückgabe und stellte Überlegungen zu möglichen Raumfahrtmissionen an. Kennzeichnend für die vielseitigen Interessen Zuses ist auch, dass er bevor er schließlich sein Bauingenieurdiplom erwarb, ein Jahr als Reklamezeichner arbeitete. Während seiner Studienzeit war Mitglied der studentischen Verbindung "Motiv", deren Mitglieder regelmäßig kleine Theaterstücke aufführten.

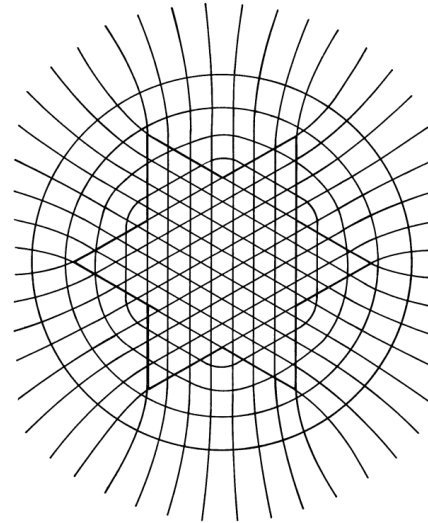


Figure 1: Zuses Entwurf für das Verkehrssystem einer Planmetropole

1.2 Berufseinstieg und Arbeit an Rechenmaschinen

Nach dem Studium begann Zuse bei den Henschel-Flugzeugwerken als Statiker zu arbeiten. Diese Arbeitsstelle kündigte er jedoch bereits kurz darauf, um an Rechenmaschinen zu arbeiten. Hierbei unterstützen ihn seine Eltern, die ihm ein Zimmer ihrer Wohnung zu Verfügung stellten. Auch seine Freunde halfen ihm mit Geld oder beim Bau der Maschinen. Zuses erste Versuche waren ausschließlich mechanisch. Sowohl Rechen- als auch Speicherwerk waren mit mechanischen Schaltgliedern umgesetzt. Seine Freunde waren ihm dabei besonders beim Ausschneiden der Bleche für die mechanischen Schaltglieder mit der Laubsäge behilflich. Sein Freund Helmut Schreyer wies ihn auch auf die Möglichkeit hingewiesen, den Rechner mit Röhren zu bauen. 1938 führten Schreyer und Zuse eine Röhrenversuchschaltung in der Technischen Hochschule vor. Dieser Ansatz wurde aber auf Grund des hohen geschätzten materiellen Aufwands und des hohen Strombedarfs einer solchen Anlage nicht in größerem Ausmaß weiter verfolgt.

Trotz einiger Zweifel wurden Zuses Bemühungen vom Fabrikanten für Spezialrechenmaschinen Dr.-Ing. Kurt Pannke mit ungefähr 7000 Reichsmark unterstützt. Nach

der komplett mechanischen Z1 baute Zuse nun die Z2, die noch einen mechanischen Speicher, aber ein elektromechanisches Rechenwerk besaß. Beim Ausbruch des zweiten Weltkriegs wurde Zuse eingezogen, nahm jedoch nicht an Kampfhandlungen teil. Dr. Pannkes verfasste einen Brief mit der Bitte um Urlaub für Zuse, damit dieser die Z2 ordnungsgemäß übergeben konnte. Dieser wurde allerdings nicht gewährt. Nach gescheiterten Versuchen, eine Freistellung für den Bau an Rechenmaschinen zum Beispiel zur Nachrichtenverschlüsselung zu erwirken, wurde Zuse schließlich von Prof. Herbert Wagner als Statiker für die Sonderabteilung F der Henschel-Flugzeugwerke unabhkömmlich gestellt. Er wirkte dort zum Beispiel an der Gleitbombe Hs 293 mit. Diese wurde per Funk ferngesteuert und war zur Bekämpfung von Erd- und Seezielen gedacht. Ein weiteres Projekt war ein von Flugzeugen abzuwerfender Torpedo, der per Draht ferngesteuert wurde.

In seiner Freizeit arbeitete Zuse weiter an seinen Rechenmaschinen. 1940 stellte er seine meist nur unzuverlässig funktionierende Z2 Vertretern der Deutschen Versuchsanstalt für Luftfahrt vor. Die Demonstration überzeugte die anwesenden Vertreter soweit, dass sie einer Teilfinanzierung der im Bau befindlichen Z3 zustimmten. Diese wurde nun vollständig elektromechanisch konstruiert. Die Tatsache, dass auch die Z3 aus gebrauchten Bauteilen aufgebaut wurde, verkomplizierte das Projekt. Sie wurde 1941 fertig gestellt.

Besonderes Interesse an der Maschine herrschte laut Zuse für die Berechnung der Eigenfrequenzen von Flugzeugflügeln. Hierfür ist die Berechnung einer komplexen Matrix und damit hoher Rechenaufwand notwendig. Die Maschine wurde allerdings nie im Routinebetrieb eingesetzt, da Zuse hierfür nicht „uk“ gestellt wurde. Die Z3 wurde 1944 von Bomben zerstört

Kurz nach der Fertigstellung der Z3 wurde Zuse erneut eingezogen, wurde jedoch kurz darauf erneut „uk“ gestellt. Er erreichte sogar, dass er einen Teil seiner Zeit mit der Rechnerentwicklung verbringen konnte und gründete die „Zuse Ingenieurbüro und Apparatebau, Berlin“. Mit rund 20 Mitarbeitern baute er im Auftrag der Deutsche Versuchsanstalt für Luftfahrt den Rechner Z4. Außerdem konstruierte er den Spezialrechner für die Flügelvermessung S1. Dieser tastete Fertigungsungenauigkeiten der Flügel von Gleitbomben ab, rechnet diese Messergebnisse mit Analog-Digital-Wandlern um und berechnete daraus die nötigen Korrekturen der Ruder.

Zuse begann 1942 an der Z4, dem Nachfolgemodell der Z3 zu arbeiten. Der Arbeitsfortschritt war nun schon stark durch den Bombenkrieg beeinträchtigt. Durch die Tochter seines Buchhalters, die beim Geheimdienst arbeitete erfuhr Zuse von einem Foto einer ähnlichen amerikanischen Maschine. Hierauf beantragten Mitarbeiter von ihm offiziell den Zugang zu etwaigen Informationen. Das Foto des Mark 1, der in Harvard entwickelt wurde, gab aber keinen Aufschluss über die Funktionsweise des Rechners.

Während die Z1, Z2 und Z3 im Bombenkrieg zerstört wurden, wurde weiter an der Z4 gearbeitet. Diese konnte schlussendlich auch mit dem Zug aus Berlin nach Göttingen

gerettet werden. Dort wurde die Z4 fertig gestellt. Zuse floh und ein paar seiner Mitarbeiter flohen anschließend mit der Z4 auf einem Lastwagen mit Anhänger nach Oberjoch in Bayernort aus. Sie fuhr weiter nach Hinterstein, wo sie die Z4 in einem Keller versteckten.

1.3 Nach dem zweiten Weltkrieg

In den Jahren 1945 und 1946 beschäftigte sich Zuse mit dem Plankalkül, welches heute als erste höhere Programmiersprache gilt. Dieses wurde allerdings nie praktisch angewandt und hat deshalb eher historische Bedeutung. Nach dem Umzug nach Hopferau bei Füssen konnte Zuse mit Hilfe seines früheren Mitarbeiters Stuckens die Z4 wieder funktionsfähig machen. Sie gründeten dort das „Zuse-Ingenieurbüro, Hopferau bei Füssen“. Trotz Verhandlungen mit IBM über die Schutzrechte an der Z4 entschied sich Zuse schließlich, die Weiterentwicklung seiner Rechenmaschinen selbst voranzutreiben. Remington-Rand erteilte ihm dann einen ersten Auftrag für ein Rechengerät in mechanischer Schaltgliedtechnik. Die Zusammenarbeit mit Remington-Rand führte auch zu einer ersten Reise Zuses und Stuckens in die USA. Neben ihren Geschäftspartnern trafen sie dort auch Howard Aiken, der persönlich seine Rechner vorstellte. 1949 schloss Zuse einen Vertrag über den Verleih der Z4 an die Eidgenössische Technische Hochschule Zürich. Die Einnahmen aus den Aufträgen von Remington-Rand sowie die Vermietung der Z4 an die ETH bildete den Grundstock für die ebenfalls 1949 in Neukirchen in Hessen mit Harro Stuckens und Alfred Eckhard gegründete Zuse KG.

Durch den Einsatz speziell durch die Firma Alois Zettler entwickelter Relais konnte die sonst übliche mechanische Abnutzung vermieden werden. Außerdem wurde durch die Fertigung von Relais mit genau definierten Ansprech- und Abfallzeiten eine höhere Taktrate möglich. Trotz dem, dass nun verschiedene Universitäten in Deutschland Rechner bauten, herrschte oft Skepsis, wenn es um den praktischen Einsatz von Rechnern ging. Beispielsweise wurde der Vorschlag an KLM, ein Platzbuchungssystem einzuführen mit der Argumentation zurückgewiesen, dass eine Maschine niemals die vielen Einflussfaktoren berücksichtigen könne, die bei der gezielten Überbuchung von Flügen zur Profitmaximierung erforderlich sind. Auch die automatische Erzeugung von Webmusterlochkarten für ein Teppichunternehmen wurde zunächst nicht umgesetzt. Zuse baute den letzten großen Relaisrechner Z5 für die Firma Ernst Leitz, die diesen zur Berechnung von Objektiven nutzte. Auch andere Unternehmen der optischen Industrie setzten in der Folge Geräte der ZUSE KG ein. Ein weiterer wichtiger Anwendungsbereich für die verhältnismäßig langsamen, jedoch zuverlässigen und haltbaren Relaisrechner war die Flurbereinigung. Hierbei werden die im Laufe der Zeit durch Vererbung zerstückelten Flurstücke sowie deren Bodenwert aufgenommen und in zusammenhängenden Stücken gleichen Werts neu verteilt. Für die hierfür erforderlichen Rechnungen lieferte die ZUSE KG die Z11 in Deutschland und Österreich aus.

Mit der Z22 bot die ZUSE KG zum ersten Mal auch eine elektronische Rechenmaschine an. Als die Deutsche Forschungsgemeinschaft dann auch Gelder für die Anschaffung

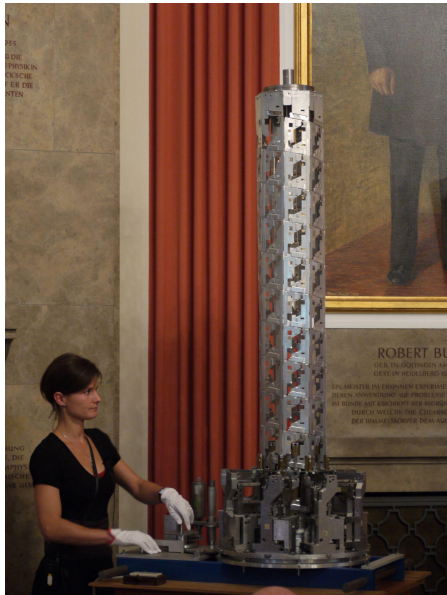


Figure 2: Helixturm

von elektronischen Rechnern bereit stellte, wurde diese ein Erfolg. Auf die Z22 folgte die Z23, die nun erstmals mit Transistortechnik ausgestattet war.

Nach Jahren des schnellen Firmenwachstums wurde der Bau eines neuen Firmengebäudes notwendig. Hierfür mussten Kredite aufgenommen werden. Außerdem sorgte der immer größere Aufwand, der für die Programmierung der Geräte (Softwareproblem) und die Erwartung der Kunden, dass dieser im Kaufpreis eingeschlossen sei, für eine schwierige wirtschaftliche Situation. Nach Schwierigkeiten mit der Z25 wurde die Firma 1964 durch die Brown, Boveri & Cie. AG und 1967 durch die Siemens AG übernommen. Konrad Zuse verließ die Firma im Anschluss.

Im Anschluss an seine unternehmerische Karriere widmete er sich dem Softwareproblem und griff seine Überlegungen zum Plankalkül wieder auf. Er veröffentlichte sein Manuskript hierzu 1972 und widmete sich auch erneute seinen Ideen zum Rechnenden Raum sowie der selbst reproduzierender Systeme. Auf die Verleihung der Ehren doktorwürde durch die TU Berlin folgten zahlreiche weitere Ehrungen, später auch in den USA, sowie die Berufung als Honorarprofessor durch die Universität Göttingen.

Konrad Zuse war auch weiterhin erfinderisch tätig. Ab 1989 entwickelte er den Helixturm. Dieser ist ein aus modularen Bauteilen zusammengesetzter Turm, der automatisch aufgebaut werden konnte. Der Turm wurde nur als Modell umgesetzt, welches heute im Deutschen Museum in München steht. Das Original sollte sich auf eine Höhe von 80-120 Metern ausfahren lassen und hätte zum Beispiel für Windkraftanlagen genutzt werden sollen. Außerdem malte er leidenschaftlich. [4] [1]

2. Z1 UND Z3

Die ersten Rechenmaschinen Konrad Zuses werden als sein wichtigster Beitrag zur Entwicklung des modernen Compu-



Figure 3: Konrad Zuse und Bill Gates beim Treffen auf der Cebit mit dem von Zuse gemalten Porträt von Gates

ters angesehen. Während die Z1 noch rein mechanisch funktioniert, ist die Z3 aus elektromechanischen Bauelementen also Relais aufgebaut. Da die Geräte bezüglich des logischen Aufbaus weitestgehend identisch sind, wird wir im Folgenden hauptsächlich die Z3 beschrieben. [3]

2.1 Hardware-Architektur

Im Folgenden werden die schaltungsmäßigen Grundlagen für das Verständnis der Rechenmaschinen kurz erläutert.

2.1.1 Grundlagen der Schaltalgebra

Abbildung 4 zeigt die Grundelemente der Schaltalgebra. Aus diesen können beliebig komplexe Strukturen aufgebaut werden, so auch die im weiteren beschriebenen Rechner.

2.1.2 Mechanische Schaltgliedtechnik

Die Z1 konstruierte Konrad Zuse rein mechanisch. Die nötigen Schaltglieder sind aus entsprechend ausgeschnittenen Blechen, die entweder fest oder beweglich waren, sowie Stiften, die durch die Bleche bewegt werden, aufgebaut. In Abbildung 5 ist ein einfacher Schalter zu sehen. Blech d ist fest, die Bleche a und b sind in x-Richtung, c in y-Richtung beweglich. Die Stellung von Blech c bestimmt, ob der Stift in der Einkerbung von a einrastet. Ist dies der Fall, so wird bei einer Bewegung von a auch b mitbewegt. c schaltet also die Signalweitergabe an und aus.

2.1.3 Relaistechnik

Bei der Z3 handelte es sich bereits um einen vollständig elektromechanischen Rechner. Sowohl Speicher als auch Rechenwerk waren mit Relais aufgebaut. Abbildung 6 zeigt den Aufbau eines einfachen Relais. Liegt eine Spannung an den Spulenkontakten an, erzeugt der Stromfluss durch die Spule ein Magnetfeld. Dieses magnetisiert den Anker und zieht in an. Der Anker wiederum drückt die Federkontakte aneinander und schließt den Sekundärkreis.

Für den Speicher wird eine Selbsthalteschaltung ähnlich Abbildung 7 verwendet. Diese ermöglicht es jeweils ein Bit mit einem Relais zu speichern, welches seinen eigenen Zustand hält.

Wird Schalter S_2 betätigt, schließt das Relais. Auch wenn Schalter S_2 nicht mehr betätigt wird, fließt weiterhin Strom

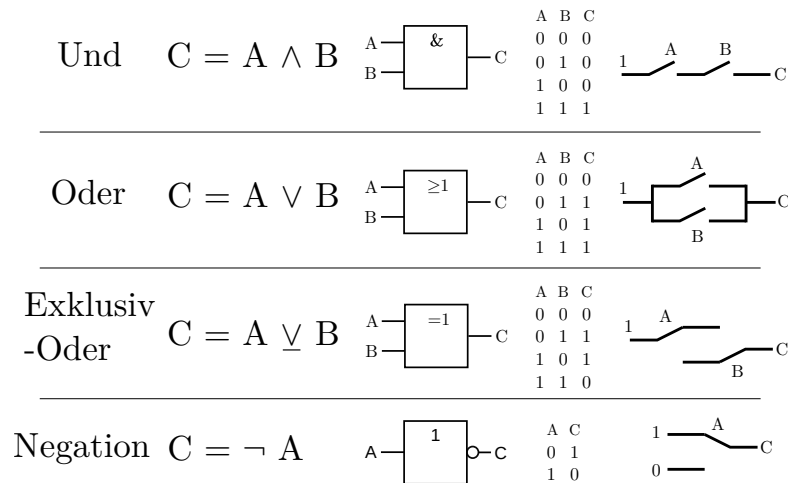


Figure 4: Grundschaltelemente

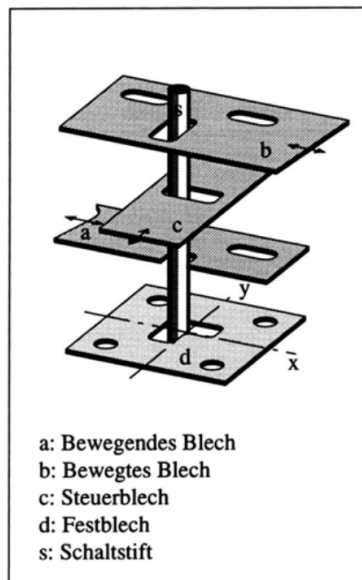


Figure 5: Mechanischer Schalter

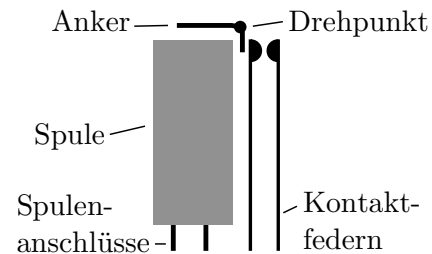


Figure 6: Aufbau eines Relais

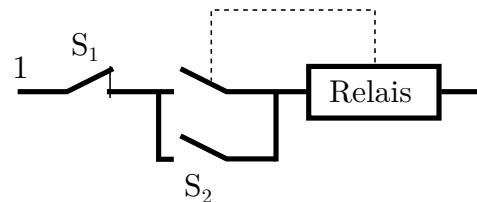


Figure 7: Relaiselbsthalteschaltung

durch die Windung des Relais, welches wiederum den Stromkreis geschlossen hält. Die Schaltung hält den Zustand. Wird der Öffner S_2 betätigt, öffnet das Relais

2.1.4 Grundstruktur

Die Z3 basierte auf dem Binärsystem und wurde in der im vorigen Abschnitt beschriebenen Relais-technik umgesetzt. Dabei trennte Zuse in seinem Entwurf Speicher und Prozessor. Die theoretischen Vorteile dieses Aufbaus wurden 1945 durch den ungarisch-amerikanischen Mathematiker John von Neumann gezeigt, welcher heute auch namensgebend für die Architektur ist. Außerdem fanden im Gegensatz zu vielen zeitgenössischen Rechnern Gleitkommazahlen Verwendung. Die Programmierung erfolgte über Lochkartenstreifen. Hierfür stand ein minimaler Befehlssatz zu Ver-

fügung, welcher neben den Grundrechenarten nur Befehle zur Ein- und Ausgabe bzw. Speicheroperationen beinhaltet.

2.1.5 Haupteinheiten

Die Hauptbestandteile des Rechners bilden der Lochstreifenleser, die Steuereinheit, die Ein- und Ausgabereinheit, der Speicher sowie die arithmetische Einheit. Diese sind durch das Bussystem verbunden. Die durch den Lochstreifenleser eingelesenen Programmbefehle werden durch die Steuereinheit interpretiert. Entsprechend den für den gelesenen Befehl festgelegten Mikrosequenz wird der Rechner gesteuert. Der Rechner führt nun die Befehle nacheinander aus und speichert entsprechend den Lochstreifenbefehlen unter der angegebenen Adresse ab, beziehungsweise lädt diese in ein Register der arithmetischen Einheit. Soll ein Wert von der Tastatur eingelesen werden, wird der Rechner angehalten, bis der Nutzer die Eingabe getätigt und bestätigt hat. Ebenso kann der Wert in Register 1 abgerufen und auf der Lampenmatrix sichtbar gemacht werden.

2.1.6 Zahlendarstellung

Zur Zahlendarstellung wurden Gleitkommazahlen mit 22 Bit Länge verwendet. Hierbei repräsentiert das erste Bit das Vorzeichen. Sieben Bits sind für den Exponenten vorgesehen. Dieser wird als Zweierkomplement gespeichert und deckt damit den Bereich von -64 bis 63 ab. Da die Z3 mit normalisierten Gleitkommazahlen arbeitet und die Zahl vor dem Komma immer eins ist, werden nur die Nachkommaziffern gespeichert. Zahlen mit Exponent -64 werden als null, Zahlen mit Exponent 63 als unendlich interpretiert.

2.1.7 Programmierung

Die Programme der Z3, von Zuse noch Rechenpläne genannt, wurde in Filmstreifen gestanzt. Die Befehle hatten dabei eine Länge von 8 Bit. Neben den Grundrechenarten Addition, Subtraktion, Multiplikation und Division stand auch die Quadratwurzel zu Verfügung. Außerdem konnten, über die Tastatur eingegebene, Werte eingelesen werden. Hierfür musste der Nutzer die vier Ziffern der Mantisse sowie den Exponenten als Dezimalwerte über die Knopfmatrix eingeben. Der Ausgabebefehl ermöglichte die des Werts in Register auf der zur Knopfmatrix analogen Lampenmatrix. Beim Speicher- bzw. Ladebefehl bildeten die letzten sechs Bits die Adresse des entsprechenden Wortes im Speicher.

Abbildung 12 zeigt die Schaltung der Z3, die die 8-Bit-Befehle, die in Abbildung 1 aufgelistet sind, entschlüsselt. $pa_1 - pa_5$ repräsentieren dabei die ersten fünf Bits der Befehle.

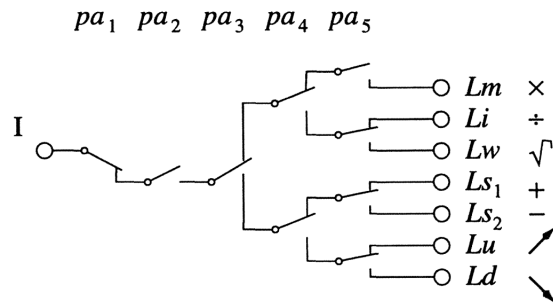


Figure 11: Befehlsentschlüsselung [3]

Befehl	Lochstreifen	Beschreibung
Lu	●●●●○○○○	Lese Wert von Tastatur nach R1
Ps 1	●○○○○○○○	Speichere Wert in R1 in Adresse 1
⋮	⋮	⋮
Lu	○○●●●○○○	Lese Wert von Tastatur nach R1
Ps 6	○○○○○○●●	Speichere Wert in R1 in Adresse 6
Pr 1	●○○○○○○○	Lese Wert von Adresse 1 nach R1
Pr 4	●●○○○○●●	Lese Wert von Adresse 4 nach R2
Lm	○○○○●○○○	Multipliziere Werte in R1 und R2
Ps 7	●○○○○●●○	Speichere Ergebnis in Adresse 7
Pr 2	●●○○○○●●	Lese Wert von Adresse 2 nach R1
Pr 5	●○○○○●○○	Lese Wert von Adresse 5 nach R2
Lm	○○○○●○○○	Multipliziere Werte in R1 und R2
Ps 8	●○○○○●●●	Speichere Ergebnis in Adresse 8
Pr 3	●●○○○○●○	Lese Wert von Adresse 3 nach R1
Pr 6	●○○○○●●●	Lese Wert von Adresse 6 nach R2
Lm	○○○○●○○○	Multipliziere Werte in R1 und R2
Pr 8	●○○○○●●●	Lese Wert von Adresse 8 nach R2
Ls1	○○●○○○○○	Addiere Werte in R1 und R2
Pr 7	●●○○○○●○	Lese Wert von Adresse 7 nach R2
Ls1	○○●○○○○○	Addiere Werte in R1 und R2
Ld	○○●●●○○○	Zeige Ergebnis mit Lampenmatrix an

Figure 12: Programm zur Berechnung des inneren Produkts zweier Vektoren der Länge drei

2.1.8 Arithmetische Einheit

Die Arithmetische Einheit (Abbildung 13) besitzt zwei Register, in die Werte geladen werden können. Dabei wird der Wert beim ersten Speicherabruf bzw. der Eingabe in Register R_1 , bei jeder weiteren in R_2 gespeichert. Das Ergebnis einer Operation wird immer in R_1 abgelegt. Der Mantissen- und der Exponententeil der Register sind im Bild getrennt dargestellt. Af und Ab bezeichnen die Exponenten der Register R_1 und R_2 , Ab und Bb die Mantissen. Es stehen zwei zusätzliche Register zur Speicherung von Zwischenergebnissen zu Verfügung.

Die Einheit besteht aus zwei Teilen, die jeweils die Verarbeitung der Exponenten und der Mantissen durchführen. Beide Teile basieren beide auf jeweils einem Addierer. Der Mantissenteil besitzt außerdem zwei Shifter, die die effiziente Implementierung der Multiplikation mit, bzw. Division durch Zweierpotenzen ermöglichen. Außerdem ermöglicht ein Multiplexer, das Mantissenzwischenergebnis

Table 1: Befehlssatz der Z3 nach [3]

Klasse	Befehl	Beschreibung	Befehlscode
Ein-/Ausgabe	Lu	Lesen von Tastatur	01 110000
	Ld	Ergebnis anzeigen	01 111000
Speicher	Prz	Lesen von Adresse z	11 $z_6 z_5 z_4 z_3 z_2 z_1$
	Psz	Speichern in Adresse z	10 $z_6 z_5 z_4 z_3 z_2 z_1$
Arithmetik	Lm	Multiplikation	01 001000
	Li	Division	01 010000
	Lw	Quadratwurzel	01 011000
	Ls_1	Addition	01 100000
	Ls_2	Subtraktion	01 101000

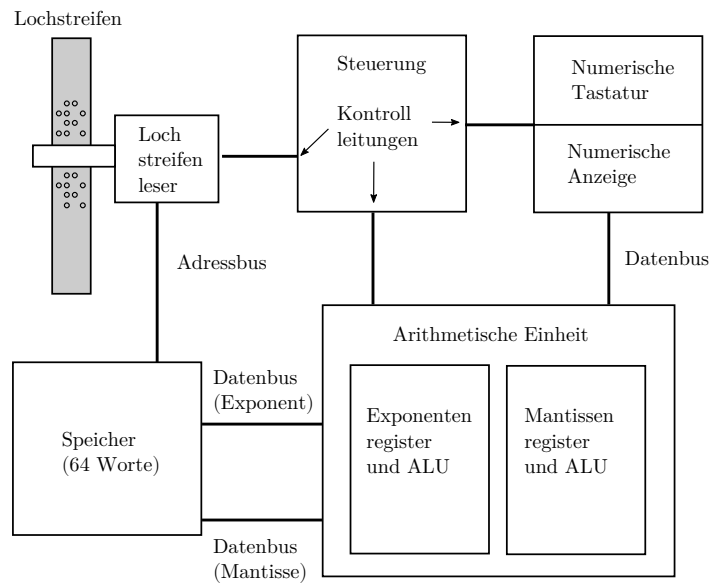


Figure 8: Überblick über das Gesamtsystem nach [3]

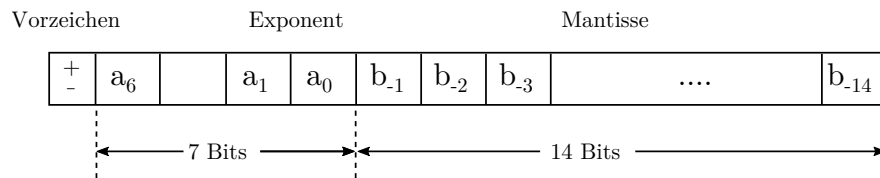


Figure 9: Gleitkommazahlendarstellung nach [3]

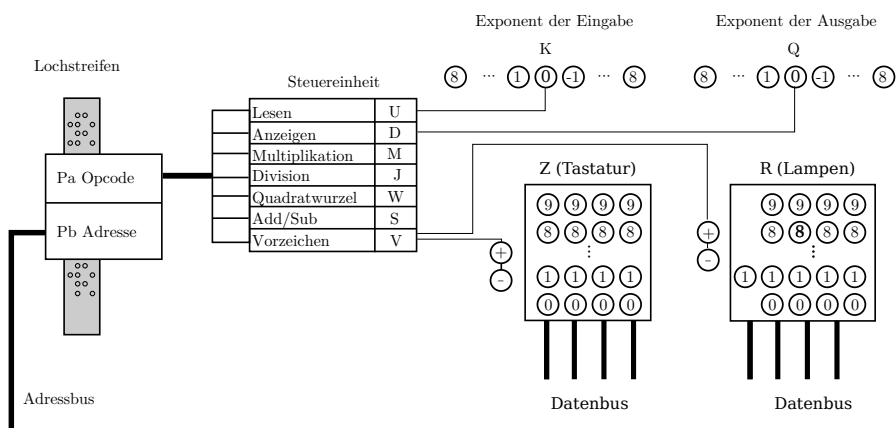


Figure 10: Steuerwerk nach [3]

Be anstelle des Additionsergebnisses auszuwählen. Die im Bild mit Ex bzw. Fx gekennzeichneten Blöcke sind Relais, die den Datenfluß steuern. Wird beispielsweise das Relais Ea aktiviert, so wird der Wert aus Af in Aa geschrieben.

2.1.9 Mikroprogrammsteuerung

Schrittschalter, wie in Abbildung 14 zu sehen, steuern den Ablauf von Multiplikation, Division und Quadratwurzelberechnung. Dabei bewegt sich der Arm schrittweise von einer Position zur nächsten. Dabei werden nacheinander die gekennzeichneten Relais geschaltet.

Im Beispielbild, wird zunächst das Relais Ea geschaltet. Dieses ist Teil der arithmetischen Einheit und bewirkt, dass der in Af gespeicherte Wert in Aa übertragen wird. Danach schaltet der Schrittschalter eine Stellung weiter und die Relais Ec und Fc bewirken den Übertrag der Werte aus Ae in Aa und Be zum linken Shifter des Mantissentheils der arithmetischen Einheit. Dies wird fortgeführt bis den Befehl vollständig abgearbeitet ist.

Die Verwendung dieser Mikroprogramme ermöglicht die Nutzung eines sehr einfachen Rechenwerks und theoretisch auch das Hinzufügen weiterer Befehle.

2.1.10 Pipelining

Üblicherweise werden bei der Ausführung eines Befehls zunächst die Argumente aus dem Speicher geladen, dann die Operation durchgeführt und schließlich das Ergebnis in den Speicher geschrieben. Zuse hat dies, wie in Abbildung 15 zu sehen, so implementiert, dass bereits während das Ergebnis der letzten Operation abgespeichert wird, die Argumente für die nächste Operation geladen werden. Dies führt zu einer deutlichen Zeitersparnis. Die Technik ist heute unter dem Namen Pipelining bekannt.

2.2 Algorithmen

In diesem Abschnitt sollen die numerischen Algorithmen der Z1 und Z3 besprochen werden.

2.2.1 Addition und Subtraktion

Zuse hat die Schaltung des Addierers optimiert. Wenn eine Addition oder Subtraktion ohne diese Optimierung implementiert wird, müssen dann die Übertragbits von einer Position zur nächsten Position weitergeleitet werden. Für die Mantisse sind dann 16 Zyklen notwendig. Mit der Hilfe der Technik des **carry-look-ahead** kann eine Addition oder Subtraktion in einem Zyklus erledigt werden.

Abbildung 16 zeigt die Schaltung des Addierers. Die Argumente für die Addition oder Subtraktion werden in den Registern Ba bzw. Bb gespeichert. ba_i bezeichnet das i -te Bit von Register Ba . Zuerst wird ein Zwischenergebnis berechnet, $bc_i = (ba_i \oplus bb_i)$. Dann wird eine AND-Operation durchgeführt, $ba_i \wedge bb_i$. Die Zwischenergebnisse bd_i werden mit Hilfe der in Abbildung gezeigten Schaltung berechnet. Das Endergebnis ist $be_i = bd_i \oplus bc_i$. Es ist zu bemerken, dass alle Relais gleichzeitig aktiviert werden. Das heißt, alle Stellen werden gleichzeitig berechnet.

Der Algorithmus für Addition und Subtraktion ist mit Hilfe von Zuses Addierer einfach zu realisieren. Zuerst wer-

den die zwei Argumente auf den gleichen Exponenten gebracht. Zuerst wird die Differenz der Exponenten ausgerechnet. Dann wird die Mantisse der kleineren Zahl um so viele Stellen nach rechts verschoben und der Exponent um die gleiche Zahl erhöht. Danach addiert oder subtrahiert die Maschine einfach die Mantissen.

Table 2: Die Mikrosequenzierung der Addition nach [3]

Zyklus	Stufe	Exponent	Mantisse
0	I,II,III		
1	IV,V	$Aa := Af$	
	I,II,III	$Ae := Aa - Ab$	$Be := 0 + Bb$
2	IV,V	wenn $(Ae \geq 0)$ dann $Ab := 0$, $Aa := Af$ sonst $Aa := 0$	wenn $(Ae \geq 0)$ dann $Ba := Bf$, $Bb := Be$ sonst $Ba := Be$, $Bb := Bf$ (Be oder Bf werden Ae Stellen nach rechts verschoben) $Be := Ba + Bb$
	I,II,III	wenn $(Be \geq 2)$ dann $Ae := Aa + Ab + 1$ sonst $Ae := Aa + Ab$	
3	IV,V	wenn $(Be \geq 2)$ $Af := Ae$	dann $Bf := Be / 2$ sonst $Bf := Be$

Die Vorzeichen der zwei Argumente werden zuerst geprüft, um der Art der Berechnung festzulegen. Falls eine Addition gefordert wird und die Vorzeichen gleich sind, wird eine Addition durchgeführt, sind sie ungleich, wird eine Subtraktion durchgeführt. Falls eine Subtraktion gefordert wird und die Vorzeichen gleich sind, wird eine Subtraktion durchgeführt, sind sie ungleich, wird eine Addition durchgeführt.

Tabelle 2 zeigt die Mikrosequenzierung der Addition. Die Argumente werden in den Registerpaaren $\langle Af, Bf \rangle$ und $\langle Ab, Bb \rangle$ gespeichert. Zuerst wird der Unterschied zwischen der zwei Argumente ausgerechnet. Dann wird die Mantisse der kleineren Zahl um so viele Stelle nach rechts verschoben. In Zyklus 2 werden die Mantissen addiert und das Ergebnis wird normalisiert. Im letzten Zyklus wird das Endergebnis im Registerpaar $\langle Af, Bf \rangle$ gespeichert.

Es soll beispielsweise die Addition $25 + 5$ durchgeführt werden. Die richtige Gleitkommazahldarstellung für 25 und 5 sind 1.1001×2^4 bzw. 1.01×2^2 . Zuerst werden die Darstellungen dieser zwei Zahlen auf einen gleichen Exponenten gebracht. In diesem Fall wird der Exponent der kleineren Zahl auf 4 erhöht und die Mantisse um 2 Stellen nach rechts verschoben. Nach der Verschiebung wird die kleinere Zahl als 0.0101×2^4 dargestellt. Danach werden die Mantissen der zwei Zahlen addiert. Schlussendlich erhalten wir als Mantisse des Endergebnisses 1.1110. Der Exponent des Endergebnisses ist einfach der größere Exponent der zwei Argumenten. Hier ist 4. Deshalb ist das Ergebnis 1.1110×2^4 .

Die Tabelle 3 zeigt zeigt die 4 bzw. 5 Zyklen, die für eine Subtraktion gebracht werden. Die ersten beiden Zyklen sind fast identisch mit den ersten beiden Zyklen des Additionsalgorithmus, es werden jetzt lediglich die Mantissen subtrahiert. Zyklus 3 wird nur durchgeführt, wenn die Differenz der Mantissen negativ ist. In Zyklus 3 macht die Maschine die Mantisse des Ergebnisses positiv. Im Zyklus 4 wird das Ergebnis normalisiert und in Zyklus 5 wird es im Registerpaar $\langle Af, Bf \rangle$ gespeichert.

Nach der Berechnung setzt eine spezielle Schaltung das

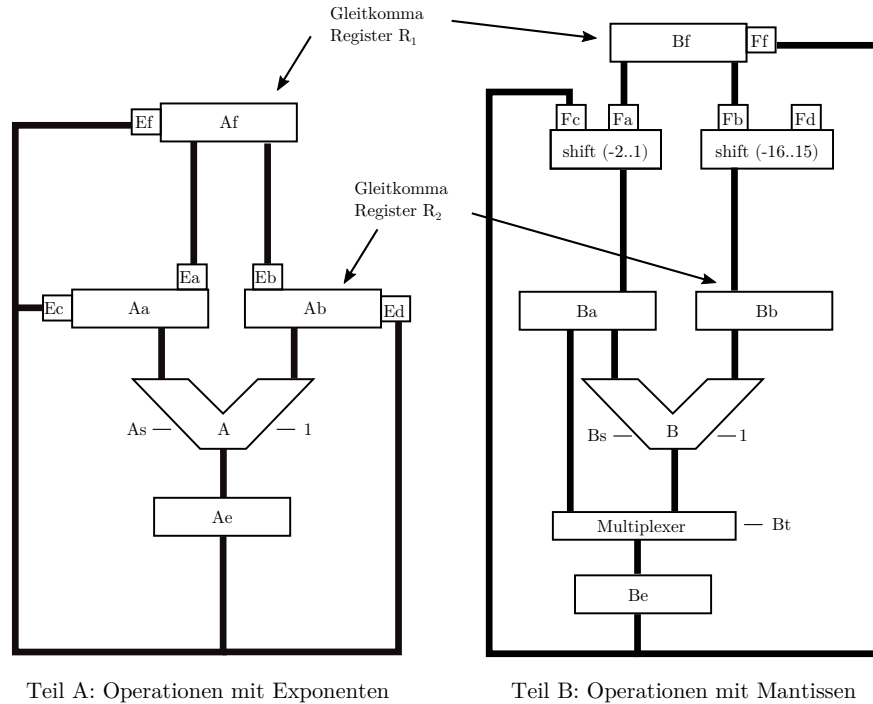


Figure 13: Arithmetische Einheit nach [3]

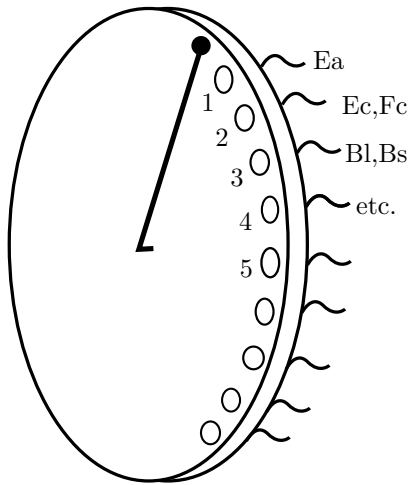


Figure 14: Schrittschalter für die Mikroprogrammsteuerung nach [3]

Vorzeichen für das Endergebnis in Abhängigkeit von den Vorzeichen der Argumente und vom Vorzeichen des partiellen Ergebnisses. Die Addition und Subtraktion werden von einer Relaiskette statt einem Schrittschalter gesteuert, weil die Anzahl der Zyklen klein ist.

2.2.2 Multiplikation

Der Algorithmus für Multiplikation der Z3 ist dem Verfah-

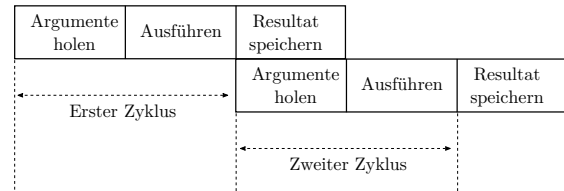


Figure 15: Pipelining nach [3]

ren, welches Schülern gelehrt wird, sehr ähnlich. Der Multiplikand und der Multiplikator werden in Register $\langle Af, Bf \rangle$ bzw. $\langle Ab, Bb \rangle$ gespeichert. Das Teilergebnis wird im Registerpaar $\langle Aa, Ba \rangle$ gespeichert und ist am Anfang Null. Tabelle 4 zeigt die 16 Zyklen, die für eine Multiplikation gebracht werden. Im ersten Zyklus werden die Exponenten der zwei Argumente addiert. Dies ist bereits das Endergebnis für den Exponenten und verbleibt in Teil A. In Teil B bearbeitet die Maschine die Mantissen und wiederholt die folgende Operation:

$$\begin{aligned}
 Ba &:= Be/2 \\
 Be &:= Ba + Bb \times (i - \text{tes Bit von } Bf) \\
 &\text{für } i = -14, \dots, 0.
 \end{aligned}$$

Wenn das i -te Bit des Registers Bf 1 ist, wird der Multiplikator addiert. Ist es Null, wird Null addiert. In jedem Schritt wird das Teilergebnis Be um eine Stelle nach rechts verschoben, so dass gilt $Ba := Be/2$. Diese Verschiebung wird durch den mit Fc verbundenen Shifter realisiert. Im letzten Zyklus wird das Endergebnis normalisiert und im Registerpaar $\langle Af, Bf \rangle$ gespeichert.

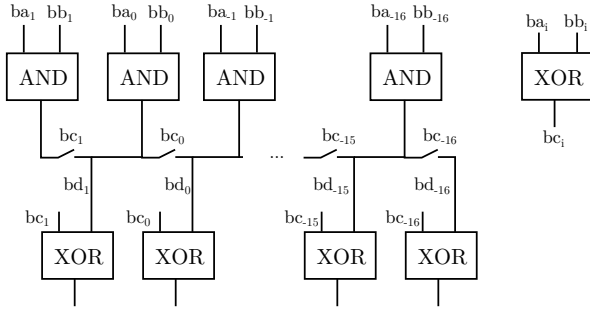


Figure 16: Carry-look-ahead nach [3]

Table 3: Die Mikrosequenzierung der Subtraktion nach [3]

Zyklus	Stufe	Exponent	Mantisse
0	I,II,III		
1	IV,V	Aa:=Af	
	I,II,III	Ae:=Aa-Ab	Be:=0+Bb
2	IV,V	wenn (Ae ≥ 0) dann Ab:=0, Aa:=Af sonst Aa:=0	wenn (Ae ≥ 0) dann Ba:=Af, Bb:=Be (verschoben) sonst Ba:=Be, Bb:=Bf (verschoben) (Be oder Bf werden Ae Stellen nach rechts verschoben)
	I,II,III	Ae:=Aa+Ab	Be:=Ba-Bb
3	IV,V	Aa:=Ae, Ab:=0	Ba:=0, Bb:=Be
	I,II,III	Ae:=Aa+Ab	Be:=Ba-Bb
4	IV,V	Aa:=Ae, Ab:=0 Ab:=Anzahl der Verschiebungen	Bb:=Be (verschoben) (Be wird durch Verschiebung nach links normalisiert)
	I,II,III	Ae:=Aa-Ab	Be:=0+Bb
5	IV,V	Af:=Ae	Bf:=Be

Es soll zum Beispiel 20 mit 5 multipliziert werden. Diese werden als normalisierte Gleitkommazahlen als 1.0100×2^4 bzw. 1.01×2^2 dargestellt. Zuerst addiert die Maschine mit Hilfe des Addierers die Exponenten von Multiplikanden und Multiplikator, $4 + 2 = 6$. Dann wird geprüft, ob die letzte Ziffer des Multiplikators 1 ist. In diesem Fall ist die letzte Ziffer des Multiplikators 1. Deshalb wird der Teilergebnis um den Multiplikanden erhöht.

$$Be := Ba + Bb \times (i - \text{tes Bit von } Bf) \\ = 0 + 1.0100 \times 1 = 1.0100$$

Im Anschluss prüft die Maschine die zweite Ziffer. Diese ist 0.

$$Ba := Be/2 = 1.0100/2 = 0.10100 \\ (\text{eine Stelle nach rechts verschieben})$$

$$Be := Ba + Bb \times (i - \text{tes Bit von } Bf) \\ = 0.10100 + 1.0100 \times 0 = 0.10100$$

Die rekursive Rechnung geht weiter.

$$Ba := Be/2 = 0.10100/2 = 0.010100 \\ (\text{eine Stelle nach rechts verschieben})$$

$$Be := Ba + Bb \times (i - \text{tes Bit von } Bf) \\ = 0.010100 + 1.0100 \times 1 = 1.100100$$

Schließlich erhalten wir das Endergebnis 1.100100×2^6 .

2.2.3 Division

Der Divisionsalgorithmus ist dem Multiplikationsalgorithmus sehr ähnlich, nur werden jetzt iterativ Subtraktionen

Table 4: Die Mikrosequenzierung der Multiplikation nach [3]

Zyklus	Stufe	Exponent	Mantisse
0	I,II,III	Ae:=Aa+Ab	
	IV,V	Aa:=Ae, Ab:=Af	
1	I,II,III	Ae:=Aa+Ab	wenn (Bf[-14]=1) dann Be:=Ba+Bb sonst Be:=Ba
	IV,V	Aa:=Ae, Af:=0, Ab:=0	Ba:=Be/2
2	I,II,III	Ae:=Aa+Ab	wenn (Bf[-13]=1) dann Be:=Ba+Bb sonst Be:=Ba
	IV,V	Aa:=Ae	Ba:=Be/2
3	I,II,III	Ae:=Aa+Ab	wenn (Bf[-12]=1) dann Be:=Ba+Bb sonst Be:=Ba
.	.	.	.
.	.	.	.
.	.	.	.
i	IV,V	Aa:=Ae	Ba:=Be/2
	I,II,III	Ae:=Aa+Ab	wenn (Bf[-15]=1) dann Be:=Ba+Bb sonst Be:=Ba
.	.	.	.
.	.	.	.
.	.	.	.
15	IV,V	Aa:=Ae	Ba:=Be/2
	I,II,III	wenn (Be ≥ 2) dann Ab:=1 Ae:=Aa+Ab	wenn (Bf[0]=1) dann Be:=Ba+Bb sonst Be:=Ba
16	IV,V	Af:=Ae	wenn (Be ≥ 2) dann Bf:=Be/2 sonst Bf:=Be Bb:=0

statt Additionen ausgeführt. Der Dividend wird in Registerpaar $\langle Af, Bf \rangle$ und der Divisor in Registerpaar $\langle Ab, Bb \rangle$ gespeichert. Das Teilergebnis wird im Registerpaar $\langle Aa, Ba \rangle$ gespeichert und wird auf 0 gesetzt. Tabelle 5 zeigt die 18 Zyklen, die für eine Division gebracht werden.

Die Exponenten werden im ersten Zyklus subtrahiert. Dieser Wert bleibt im Weiteren unverändert. In Teil B werden die Mantissen bearbeitet. Nehmen wir an, dass wir x/y für die normalisierten Mantissen x und y berechnen möchten. Die Zahlen sind alle normalisierte Gleitkommazahlen. Deshalb ist die erste Ziffer des Ergebnisses gleich 1, wenn $x \geq y$ ist und gleich 0, wenn $x < y$ ist. Wenn $x \geq y$ ist setzen wir die Ziffer des Ergebnisses auf 1 und berechnen den Rest, der $x - y$ ist. Dann wird der Rest um eine Stelle nach links verschoben, neue Ergebnis an der Stelle [-1] im Register Bf gespeichert. Ist das Ergebnis 0, wird die Division mit x statt $x - y$ fortgeführt.

Die Hauptschleife der Division hat die folgende Form:

$$Ba := 2 \times Be \\ \text{wenn}(Ba - Bb \geq 0) \\ \text{dann } Be := Ba - Bb, Bf[i] := 1 \\ \text{sonst } Be := Ba, Bf[i] := 0 \text{ } f r i = 0, \dots, -14$$

Das Teilergebnis Be wird um eine Stelle nach links verschoben. Dies erzeugt $Ba := 2 \times Be$. Die Operation kann durch die Verwendung des Shifters sehr schnell durchgeführt werden. Im letzten Zyklus wird das Endergebnis normalisiert und im Registerpaar $\langle Af, Bf \rangle$ gespeichert.

Es soll exemplarisch 100 mit Gleitkommadarstellung 1.100100×2^6 durch 5 mit Gleitkommadarstellung 1.01×2^2 geteilt werden. Dabei wird zunächst die Differenz der Exponenten von Dividenten und Divisor gebildet. Hier gilt $6 - 2 = 4$. Im Fall der Mantisse prüft die Maschine zuerst, ob die Mantisse von Dividenten größer als die Mantisse von Divisor ist. Dies ist im Beispiel der Fall.

$$(Ba - Bb \geq 0) \\ \text{dann } Be := Ba - Bb = 1.100100 - 1.01 =$$

Table 5: Die Mikrosequenzierung der Division nach [3]

Zyklus	Stufe	Exponent	Mantisse
0	I,II,III		
1	IV,V	Aa:=Af	Ba:=Bf
	I,II,III	Ae:=Aa-Ab	wenn (Ba-Bb ≥ 0) dann Be:=Ba-Bb, bt:=1 sonst Be:=Ba, bt:=0
2	IV,V	Aa:=Ae Ab:=0	Bf:=0 wenn (bt=1) dann Bf[0]:=1 Ba:=2×Be
	I,II,III	Ae:=Aa+Ab	wenn (Ba-Bb ≥ 0) dann Be:=Ba-Bb, bt:=1 sonst Be:=Ba, bt:=0
3	IV,V	Aa:=Ae	wenn (bt=1) dann Bf[-1]:=1 Ba:=2×Be
	I,II,III	Ae:=Aa+Ab	wenn (Ba-Bb ≥ 0) dann Be:=Ba-Bb, bt:=1 sonst Be:=Ba, bt:=0
.	.	.	.
.	.	.	.
.	.	.	.
i	IV,V	Aa:=Ae	wenn (bt=1) dann Bf[2-i]:=1 Ba:=2×Be
	I,II,III	Ae:=Aa+Ab	wenn (Ba-Bb ≥ 0) dann Be:=Ba-Bb, bt:=1 sonst Be:=Ba, bt:=0
.	.	.	.
.	.	.	.
.	.	.	.
16	IV,V	Aa:=Ae	wenn (bt=1) dann Bf[-14]:=1 Ba:=2×Be
	I,II,III	Ae:=Aa+Ab	wenn (Ba-Bb ≥ 0) dann Be:=Ba-Bb, bt:=1 sonst Be:=Ba, bt:=0
17	IV,V	wenn (Bf[0]=0) dann Ab:=-1	wenn (bt=1) dann Bf[-15]:=1 Bb:=0 wenn (Bf[0]=0) dann Ba:=2×Be sonst Ba:=Be
	I,II,III	Ae:=Aa+Ab	Be:=Ba-Bb
18	IV,V	Af:=Ae	Bf:=Be

0.010100
Bf[0] := 1

Im nächsten Schritt wird zuerst das Teilergebnis Be eine Stelle nach links verschoben. Dann geht die rekursive Rechnung identisch weiter.

Ba := 2 × Be = 0.010100 × 2 = 0.10100
(Ba - Bb < 0)
dann Be := Ba = 0.10100, Bf[-1] := 0

Nächster Schritt:

Ba := 2 × Be = 0.10100 × 2 = 1.0100
(Ba - Bb ≥ 0)
dann Be := Ba - Bb = 0.0, Bf[-2] := 1

Zum Schluss erhalten wir die Mantisse des Endergebnisses 1.01. Der Exponent des Endergebnisses wurde bereits im ersten Schritt ausgerechnet. Somit bekommen wir das Ergebnis, 1.01×2^4 .

2.2.4 Quadratwurzelberechnung

Tabelle 6 zeigt die 20 Zyklen, die für die Berechnung der Quadratwurzel benötigt werden. Das Argument wird im Registerpaar <Af, Bf> gespeichert. Das Teilergebnis wird im Registerpaar <Aa, Ba> gespeichert und auf 0 gesetzt. Der Exponent des Arguments muss gerade sein. Falls der Exponent ungerade ist, wird die Mantisse um eine Stelle nach rechts verschoben und der Exponent um 1 erhöht. Diese Operation wird im ersten Zyklus durchgeführt. Im Zyklus 19 wird der Exponent halbiert. Dann bekommen wir der Exponent des Endergebnisses.

Grundsätzlich wird die Quadratwurzel auf eine Division umgesetzt. Nehmen wir an, dass wir die Wurzel von x berechnen wollen, dann suchen wir eine Zahl Q derart, dass $x/Q = Q$ ist. Um diese Q zu finden, können wir nacheinander das i -te Bit von Q auf 1 setzt und prüft ob $x \geq Q^2$ noch gilt. Falls ja, wird das i -te Bit auf 1 gesetzt. Falls nein, wird das i -te Bit auf 0 gesetzt.

Nehmen wir an, wir haben bereits Bit 0 bis Bit $-i+1$ des Endergebnisses berechnet.

$Q_{-i+1} = Bf[0] \times 2^0 + Bf[-1] \times 2^{-1} + \dots + Bf[-i+1] \times 2^{-i+1}$
Bit $-i$ wird dann auf q_{-i} gesetzt und es muss gelten, dass

$$x \geq Q_{-i}^2 = (Q_{-i+1} + q_{-i}2^{-i})^2$$

Dies ist erfüllt, wenn

$$x - Q_{-i}^2 = (x - Q_{-i+1}^2) - 2^{-i}q_{-i}(2Q_{-i+1} + 2^{-i}q_{-i}) \geq 0$$

Dann wird t_{-i} so definiert

$$2^{-i}t_{-i} = x - Q_{-i}^2 = (x - Q_{-i+1}^2) - 2^{-i}q_{-i}(2Q_{-i+1} + 2^{-i}q_{-i})$$

Die obere Gleichung kann als folgende geschrieben

$$2^{-i}t_{-i} = t_{-i+1}2^{-i+1} - 2^{-i}q_{-i}(2Q_{-i+1} + 2^{-i}q_{-i})$$

Die Gleichung $2^{-i+1}t_{-i+1} = (x - Q_{-i+1}^2)$ wird in die obere Gleichung gesetzt. Nach der Vereinfachung bekommen wir

$$t_{-i} = 2t_{-i+1} - q_{-i}(2Q_{-i+1} + 2^{-i}q_{-i})$$

Für $q_{-i} = 1$ bekommen wir

$$t_{-i} = 2t_{-i+1} - (2Q_{-i+1} + 2^{-i})$$

Falls $t_{-i} \geq 0$, dann wird das Bit $-i$ des Endergebnisses auf 1 gesetzt, d. h. $Bf[-i] := 1$. Falls t_{-i} negativ ist, dann wird das Bit $-i$ des Endergebnisses auf 0 gesetzt. Am Anfang ist $t_0 = x$. Q_{-i+1} wird in jedem Schritt im Register Bf gespeichert. Be enthält in der i -ten Iteration t_{-i+1} . Bit $-i$ wird in jedem Schritt versuchsweise auf 1 gesetzt und das Vorzeichen von t_{-i} wird geprüft. Die Hauptschleife hat folgende Form:

Ba := 2 × Be
Bb := 2 × Bf
Bb[-i] := 1
wenn (Ba - Bb ≥ 0)
dann Be := Ba - Bb, Bf[-i] := 1
sonst Be := Ba, Bf[-i] := 0

Beispielweise rechnen wir die Quadratwurzel von 9. Die normalisierte Darstellung für 9 ist 1.001×2^3 . Der Exponent ist ungrade. Deshalb wird die Mantisse eine Stelle nach rechts verschoben und der Exponent um 1 erhöht. Der Exponent des Endergebnisses ist halb so groß, $4/2 = 2$. Und zur Mantisse folgen wir den rekursiven Algorithmus.

Erste Schritt

$t_0 - 1.0 < 0$
dann Bf[0] = 0

Zweiter Schritt

$t_{-1} = 2t_0 - (2Q_0 + 2^{-1})$
 $= 2 \times 0.1001 - (2 \times 0.0 + 0.1) = 0.101 \geq 0$
dann Bf[-1] = 1

Dritter Schritt

$$\begin{aligned} t_{-2} &= 2t_{-1} - (2Q_{-1} + 2^{-2}) \\ &= 2 \times 0.101 - (2 \times 0.1 + 0.01) = 0.0 \geq 0 \\ \text{dann } Bf[-2] &= 1 \end{aligned}$$

$t_{-2} = 0$, die rekursive Berechnung endet hier. Am Endeffekt bekommen wir das Endergebnis, 0.11×2^2 . Dieses Ergebnis wird dann normalisiert. Das Endergebnis ist 1.1×2^1 , das 3 entspricht.

Table 6: Die Mikrosequenzierung der Quadratwurzelberechnung nach [3]

Zyklus	Stufe	Exponent	Mantisse
0	I,II,III		
1	IV,V		wenn (Af[0]=1) dann Ba:=2×Bf sonst Ba:=Bf Bb[0]:=1
	I,II,III		wenn (Ba-Bb≥0) dann Be:=Ba-Bb, bt:=1 sonst Be:=Ba, bt:=0
2	IV,V		Bf:=0 wenn (bt=1) dann Bf[0]:=1
	I,II,III	Ae:=Aa+Ab	wenn (Ba-Bb≥0) dann Be:=Ba-Bb, bt:=1 sonst Be:=Ba, bt:=0
3	IV,V		wenn (bt=1) dann Bf[-1]:=1 Ba:=2×Be, Bb:=2×Bf, Bb[-2]:=1
	I,II,III		wenn (Ba-Bb≥0) dann Be:=Ba-Bb, bt:=1 sonst Be:=Ba, bt:=0
.	.	.	.
i	IV,V		wenn (bt=1) dann Bf[2-i]:=1 Ba:=2×Be, Bb:=2×Bf, Bb[1-i]:=1
	I,II,III		wenn (Ba-Bb≥0) dann Be:=Ba-Bb, bt:=1 sonst Be:=Ba, bt:=0
.	.	.	.
18	IV,V		wenn (bt=1) dann Bf[-16]:=1 Ba:=2×Be, Bb:=2×Bf
	I,II,III		wenn (Ba-Bb≥0) dann Be:=Ba-Bb, bt:=1 sonst Be:=Ba, bt:=0
19	IV,V	Aa:=Af/2	Ba:=Bf, Bb:=0
	I,II,III	Ae:=Aa+0	Be:=Ba+Bb
20	IV,V	Af:=Ae	Bf:=Be

2.2.5 Ausnahmenbehandlung von Gleitkommazahlen

Wie bereits erwähnt, behandelt die Z3 alle Zahlen mit Exponent -64 als 0 und alle mit Exponent 63 als ∞ . Um diese Sonderfälle zu erkennen, überwacht die Z3 den Wert des Exponenten von jedem arithmetischen Ergebnis und jeder aus dem Speicher gelesenen Zahl. Wenn der Exponent von Register R_1 -64 ist, dann wird ein Relais Nn_1 auf 1 gesetzt. Falls der Exponent von Registerpaar $\langle Ab, Bb \rangle$ -64 ist, dann wird ein anderes Relais Nn_2 auf 1 gesetzt. Falls der Exponent von Registerpaar $\langle Af, Bf \rangle$ und $\langle Ab, Bb \rangle$ 63 ist, dann werden das Relais Ni_1 bzw. Ni_2 aktiviert. So wissen wir jedes Mal, wenn Null oder Unendlich auftritt.

Nach der Detektion der Ausnahmezahle werden die Operation zuerst ganz normal durchgeführt. Dann wird das Ergebnis überschrieben. Z.B. $10/\infty$. Zuerst wird die Berechnung ganz normal gerechnet. Aber am Ausgang setzt die Überwachungsschaltung der Exponent des Ergebnisses auf -64 . Wenn der Exponent auf -64 gesetzt wird, ist der Wert der Mantisse unbedeutend. Die Multiplikation mit Null wird analog durchgeführt. Aber manche Operationen kann Z3 nicht durchführen, wie $0/0$, ∞/∞ , $\infty-\infty$ und $0 \times \infty$. In diesem Fall wird ein Ausnahmelämpchen beleuchtet.

Die Implementierung einer Überwachungsschaltung von Ausnahmezahle ist ein wichtiger Fortschritt. Mit Hilfe dieses Entwurfs brauchen die Programmierer nicht vor und nach jeder Berechnung die numerische Überschreitung zu prüfen.

2.3 Ein- und Ausgabe von Dezimalzahlen

Um die Zahl in die Maschine einzugeben, soll der Operator zuerst die Maschine durch den Befehl *Lu* stoppen und dann kann der Operator eine Dezimalzahl mit vier Ziffer in die Tastatur eintippen. Auch der Dezimalexponent wird per Tasten eingegeben. Nach Bestätigung der Eingabe, wandelt die Maschine die Dezimalzahl in die normalisierte Gleitkomma-Darstellung um. Für die Ausgabe wird die im Registerpaar $\langle Af, Bf \rangle$ gespeicherte Gleitkommazahl in eine Dezimalzahl umgewandelt. Dann werden die entsprechenden Lampen angeschaltet. Die Vorzeichen und der Dezimalexponent werden ebenfalls durch Lampen angezeigt. Abbildung ?? zeigt ein Foto der Bedienoberfläche der Z3.

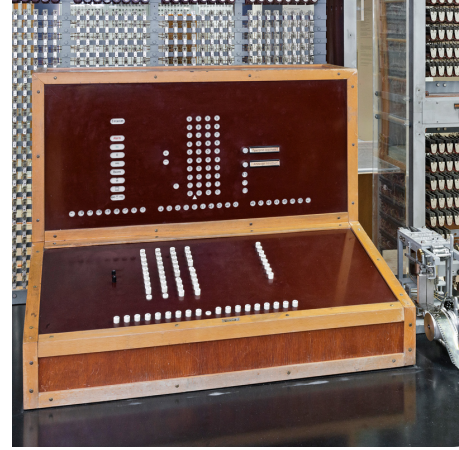


Figure 17: Ein- und Ausgabegeeinheit der Z3

2.3.1 Eingabe von Dezimalzahlen

Die eingegebene Dezimalzahl wird zuerst zifferweise in binäre Zahl umgewandelt. Für jede Ziffer erzeugt die Maschine eine vier-Bit Binärzahl. Die erste vier Bits werden durch Busverbindung Za in $Ba[-10]$, $Ba[-11]$, $Ba[-12]$ und $Ba[-13]$ gespeichert. Die Ziffer wird ab -13 gespeichert, weil die größte Eingabe 9999 ist, die auch als $1,0011100001111x2^{13}$ geschrieben kann und die kleinste Eingabe 0 ist, die auch als $1(0,0000000000001x2^{13})$ geschrieben kann. Dann wird die Zahl im Ba mit 10 multipliziert und mit der Zahl im Bb addiert. Die Multiplikation mit 10 wird durch zwei Shifter, einmal eine Stelle nach links und einmal drei Stellen nach links und eine Addition dieser zwei Werten realisiert. Der Register Bb wird mit 0 initialisiert. Das Teilergebnis wird im Bb gespeichert. Die andere drei Ziffern werden durch Busverbindungen Zb , Zc bzw. Zd in Ba geladen. Nach vier Verarbeitungen wird die Dezimalzahl in eine binäre Zahl umgewandelt. Die gesamte Operation braucht 8 Zyklen, ein Zyklus für Einlesen und ein Zyklus für Multiplikation mit 10. Tabelle 7 zeigt die 8 Zyklen, die für eine Binär-Dezimal Umwandlung gebracht werden. Im letzten Zyklus wird das Ergebnis normalisiert.

Table 7: Die Mikrosequenzierung der Binär-Dezimal Verwandlung nach [3]

Zyklus	Stufe	Exponent	Mantisse
0	I,II,III		
1	IV,V		Ba:= $d_4 \times 2^{-13}$ Z_a aktiv
	I,II,III		Be:=Ba+Bb
2	IV,V	Ba:= $2 \times Be$, Bb:= $8 \times Be$	
	I,II,III		Be:=Ba+Bb
3	IV,V		Ba:= $d_3 \times 2^{-13}$ Z_b aktiv
	I,II,III		Be:=Ba+Bb
4	IV,V	Ba:= $2 \times Be$, Bb:= $8 \times Be$	
	I,II,III		Be:=Ba+Bb
5	IV,V		Ba:= $d_2 \times 2^{-13}$ Z_c aktiv
	I,II,III		Be:=Ba+Bb
6	IV,V	Ba:= $2 \times Be$, Bb:= $8 \times Be$	
	I,II,III		Be:=Ba+Bb
7	IV,V		Ba:= $d_1 \times 2^{-13}$ Z_d aktiv
	I,II,III		Be:=Ba+Bb
8	IV,V	Aa:=13	Ba:= $2 \times Be$, Bb:= $8 \times Be$
	I,II,III	Ab:=z Ae:=Aa-Ab	Be:=Ba+Bb
9	IV,V	Aa:=Ae Ab:=0	Bb:=Be (normalisiert)

Nach der Umwandlung soll der Dezimalexponent berechnet werden. Ist der Exponent positiv, dann wird die Mantisse entsprechend oft mit 10 multipliziert. Falls der Exponent negativ ist, dann wird die Mantisse entsprechend oft mit 0,1 multipliziert. Schließlich wird das Ergebnis normalisiert und in das Registerpaar $\langle Af, Bf \rangle$ bzw. $\langle Ab, Bb \rangle$ gespeichert.

Falls der Dezimalexponent positiv ist, wird die Mantisse mit 10 multipliziert. Die Mantisse wird dann mit 1.01×2^3 multipliziert und der Exponent wird um 3 erhöht. Die Tabelle 8 zeigt die Mikroprozedur dieses Prozesses.

Table 8: Multiplikation mit 10 nach [3]

Zyklus	Stufe	Exponent	Mantisse
i	IV,V	Aa:=Ae, Ab:=-3	Ba:=Be, Bb:=Be/4
	I,II,III	Ae:=Aa-Ab	Be:=Ba+Bb

Falls der Dezimalexponent negativ ist, wird die Mantisse mit 0.1 multipliziert. Diese Multiplikation ist relativ komplexer, weil 0.1 eine periodische Zahl im Binärsystem ist.

$$0.1 \text{ (dezimal)} = 2^{-4} \times 1.1001100110011001 \text{ (binär)}$$

Die Maschine implementiert diese Multiplikation in vier Schritten.

- $x_1 = 1.1 \cdot x_0 = x_0 + x_0/2$,
 - $x_2 = 1.0001 \cdot x_1 = x_1 + x_1/2^4$,
 - $x_3 = 1.00000001 \cdot x_2 = x_2 + x_2/2^8$,
 - $x_4 = 1.0000000000000001 \cdot x_3 = x_3 + x_3/2^{16}$,
- Am Endeffekt bekommen wir $x_4 \approx x_0 \cdot 0.1$. Dann am Ende des vierten Zyklus wird der Exponent um 4 reduziert. Die Tabelle 9 zeigt die Mikroprozedur dieses Prozesses.

Table 9: Multiplikation mit 0.1

Zyklus	Stufe	Exponent	Mantisse
i	IV,V		Ba:=Be, Bb:=Be/2
	I,II,III		Be:=Ba+Bb
i+1	IV,V		Ba:=Be, Bb:=Be/2 ⁴
	I,II,III		Be:=Ba+Bb
i+2	IV,V		Ba:=Be, Bb:=Be/2 ⁸
	I,II,III		Be:=Ba+Bb
i+3	IV,V	Aa:=Ae, Ab:=-4	Ba:=Be, Bb:=Be/2 ⁸
	I,II,III	Ae:=Aa+Ab	Be:=Ba+Bb

2.3.2 Ausgabe von Dezimalzahlen

Wenn die Maschine das ausgerechnete Ergebnis anzeigen soll, wird zuerst der Exponent der auszugebenden Zahl in den Bereich von 0 bis 3 gebracht. Dies passiert durch wiederholte Multiplikation mit 0.1 oder 10. Dann wird der Exponent durch Verschiebung der Mantisse auf 2 gebracht. Danach nehmen wir die ersten vier Bits von Be . Dies bildet eine Dezimalzahl zwischen 0 und 15 und stellen die ersten zwei Ziffern der gewünschten Dezimalzahl dar. Diese können durch die ersten beiden Spalten der Ausgabelampen angezeigt werden. Die vier Bits werden in Register Bf gespeichert (an den Positionen 0 bis -3) und dann in Be gelöscht.

Anschließend wird der Rest der Mantisse mal 10 genommen. Analog zu vorher werden die vier Bits vor dem Komma in Be gespeichert und in Be gelöscht. Nach vier Schleife ist die Zahl vollständig behandelt. Dann kann sie zur Ausgabereinheit übertragen werden, welche die entsprechenden Lampen aktiviert.

Der Pseudocode für die dargestellte Operation (ohne Abrundung) ist folgender:

```

Bf[0..-3] := [Be]
Be := Be - [Be]
Ba := 2 × Be, Bb := 8 × Be
Be := Ba + Bb
Bf[-4..-7] := [Be]
Be := Be - [Be]
Ba := 2 × Be, Bb := 8 × Be
Be := Ba + Bb
Bf[-8..-11] := [Be]
Be := Be - [Be]
Ba := 2 × Be, Bb := 8 × Be
Be := Ba + Bb
Bf[-12..-15] := [Be]
Be := Be - [Be]

```

Nachdem die Dezimalzahl berechnet und ausgegeben wurde, stoppt die Maschine und wartet, bis der Operator die Taste "Automatik" drückt. Die Werte in den Registerpaare $\langle Af, Bf \rangle$ und $\langle Ab, Bb \rangle$ werden dann gelöscht.

2.3.3 Bedingte Sprünge

Prof. Raúl Rojas hat gezeigt, dass die Z3, obwohl beim Bau nicht dafür ausgelegt, in der Lage ist, bedingte Sprünge zu simulieren. Dies kann erreicht werden, indem Teile des Programms deaktiviert werden. Dies wird erreicht, indem unter einer bestimmten Bedingung der Speicherzugriff für das Programm ausgeschaltet wird.

Der Speicherzugriff soll aktiviert sein, wenn die gilt $z = i$. Hierfür wird mit einer kleinen Zahl e

$$d = (z - i) * (z - i)$$

$$g = d / (d - e)$$

berechnet. Für $z = i$ erhalten wir $g = 0$, für $z \neq i$ $g \approx 1$. Mit

$$t = (2^{16} + g) - 2^{16}$$

wird das Ergebnis bereinigt, so dass wir $g = 0$ bzw. $g = 1$ als Endergebnis erhalten.

Soll nun ein Speichervorgang abhängig von $z = i$ aktiviert werden, wird die folgende Rechnung mit a als Result einer Berechnung und x als aktuellem Wert im Speicher vor der Speicheroperation durchgeführt.

$$a * t + (1 - t) * x$$

Dies führt dazu, dass bei $z = i$ der aktuelle Wert im Speicher erneut geschrieben wird, bei $z \neq i$ das Rechenergebnis abgespeichert wird.

Wird ein entsprechender Bedingungsblock vor jedem Codesegment eingefügt, kann abhängig dessen Wirksamkeit von den Ergebnissen der vorherigen Segmente abhängig gemacht werden.[3]

2.4 Die vollständige Architektur der Z3

Abbildung 18 zeigt die vollständige Architektur der Z3, deren Einheiten in den vorhergehenden Kapiteln behandelt wurden. Einzig einige Details fehlen noch zum Verständnis des Gesamtaufbaus.

Die Busverbindungen zwischen der Tastatur und dem Register Ba werden durch die Relaischaltungen Za , Zb , Ze und Zd aktiviert. Dies dienen der Übertragung der eingegebenen Ziffer. In Teil A finden wir neben schon bekannten Relais Eg und Ei . Mit Hilfe dieser zwei Relaischaltungen können man direkt zwei nützliche Zahlen, 13 und -4, in das Register Aa bzw. Register Ab geschrieben werden. Außerdem ist der Shifter Ee für die Quadratwurzelberechnung eingezeichnet. Das Flip-Flop Ah_1 speichert, ob Werte in Register R_1 oder R_2 geladen werden. Auch die Ausnahmebehandlung, hier mit „Null, Unendlich“ gekennzeichnet, ist in der Zeichnung enthalten. Schlussendlich sind mit Fp und Fq sowie Fh bis Fm auch die Steuerleitungen der bereits vorher erwähnten Shifter des A- und B-Teils der arithmetischen Einheit dargestellt. [3]

3. GESCHICHTLICHE EINORDNUNG

In diesem Kapitel werden Zuses Computers, Z1 und Z3, mit anderen historischen Rechnern und mit heutigen Rechnerarchitektur verglichen.

3.1 Vergleich mit anderen historischen Rechnern

Ràul Rojas schreibt in seinem Buch „Die Rechenmaschinen von Konrad Zuse“: „Die Definition einer universellen Rechner ist: Eine Maschine mit ausreichendem Speicher, der sowohl Daten wie Befehle fasst, und mit einem zur Ausführung der Befehle CLR (löschen), INC (inkrementieren), LOAD (lesen), STORE (speichern) und BR (springen falls Null) fähigen Prozessor ist eine universelle Maschine. In diesem Sinne war die Z1 keine universelle Maschine, aber die anderen frühen Rechenmaschinen waren dies ebenso wenig.“ Er bezieht sich dabei auf die ABC von John Vincent Atanasoff, die nur für das Lösen großer linearer Gleichungssysteme gedacht war, den Mark I der ebenfalls keinen bedingte Sprungbefehl unterstützte und der ENIAC,

dessen Bausteine noch händisch verdrahtet werden mussten.

Dennoch zeigt Rojas, wie auch in dieser Arbeit beschrieben, dass die Z3 in der Lage gewesen wäre, bedingte Sprünge zu simulieren. Er weist auch darauf hin, dass Schleifen durch das aneinanderkleben der Enden von Lochkartenstreifen umgesetzt werden können. Auf diese Weise wäre eine Simulation eines universellen Rechners unter großem Aufwand möglich gewesen.

Table 10: Vergleich der architektonischen Eigenschaften I nach [3]

Maschine	Speicher und CPU getrennt?	Bedingte Sprünge?	Soft- oder Hardware-programmierung	Sich selbst modifizierende Programme?	Indirekte Adressierung?
Zuses Z1	✓	x	Software	x	x
Atanasoff	✓	x	Hardware	x	x
Harvard	x	x	Software	x	x
Mark I					
ENIAC	x	teilweise	Hardware	x	x
Manchester	✓	✓	Software	✓	x
Mark I					
EDSAC	✓	✓	Software	✓	x

Table 11: Vergleich der architektonischen Eigenschaften II nach [3]

Maschine	Interne Codierung	Fest- oder Gleitkomma?	Bit-Sequentielle Arithmetik?	Architektur	Technologie
Zuses Z1	Binär	Gleitkomma	nein	sequentiell	Mechanisch
Atanasoff	Binär	Festkomma	ja	vektoriell	Elektronisch
Harvard	Dezimal	Festkomma	nein	parallel	Elektronisch
Mark I					
ENIAC	Dezimal	Festkomma	nein	Datenfluss	Elektronisch
Manchester	Binär	Festkomma	ja	sequentiell	Elektronisch
Mark I					
EDSAC	Binär	Festkomma	nein	sequentiell	Elektronisch

Wie aus der Gegenüberstellung von Zuses Computer und anderen frühen Rechnern in Tabelle 10 und 11 zu entnehmen ist, waren auch die Zeitgenossen der Z1 und Z3 keine Universalrechner. Die deutlich später, nämlich 1948/49 fertig gestellten EDSAC (Electronic Delay Storage Automatic Calculator) und Manchester Mark 1 ermöglichten immerhin bereits bedingte Sprünge. Sie boten außerdem die Möglichkeit, Programme direkt in den Speicher zu laden, statt diese aus Lochkarten einzulesen. Der erste universelle Computer ist die in Manchester zwischen 1946 und 1948 gebaute 'Baby' Mark 1.

3.2 Vergleich mit heutigen Architekturen

Table 12: Vergleich mit heutigen Architekturen

Maschine	Speicher und CPU getrennt?	Bedingte Sprünge?	Soft- oder Hardware-programmierung	Sich selbst modifizierende Programme?	Indirekte Adressierung?
Zuses Z3	✓	x	Software	x	x
Heutige Architektur	✓	✓	Software	✓	✓
Maschine	Interne Codierung	Fest- oder Gleitkomma	Bit-Sequentielle Arithmetik	Architektur	Technologie
Zuses Z3	Binär	Gleitkomma	nein	sequentiell	Elektro-mechanisch
Heutige Architektur	Binär	Gleitkomma	ja	sequentiell	Elektronisch

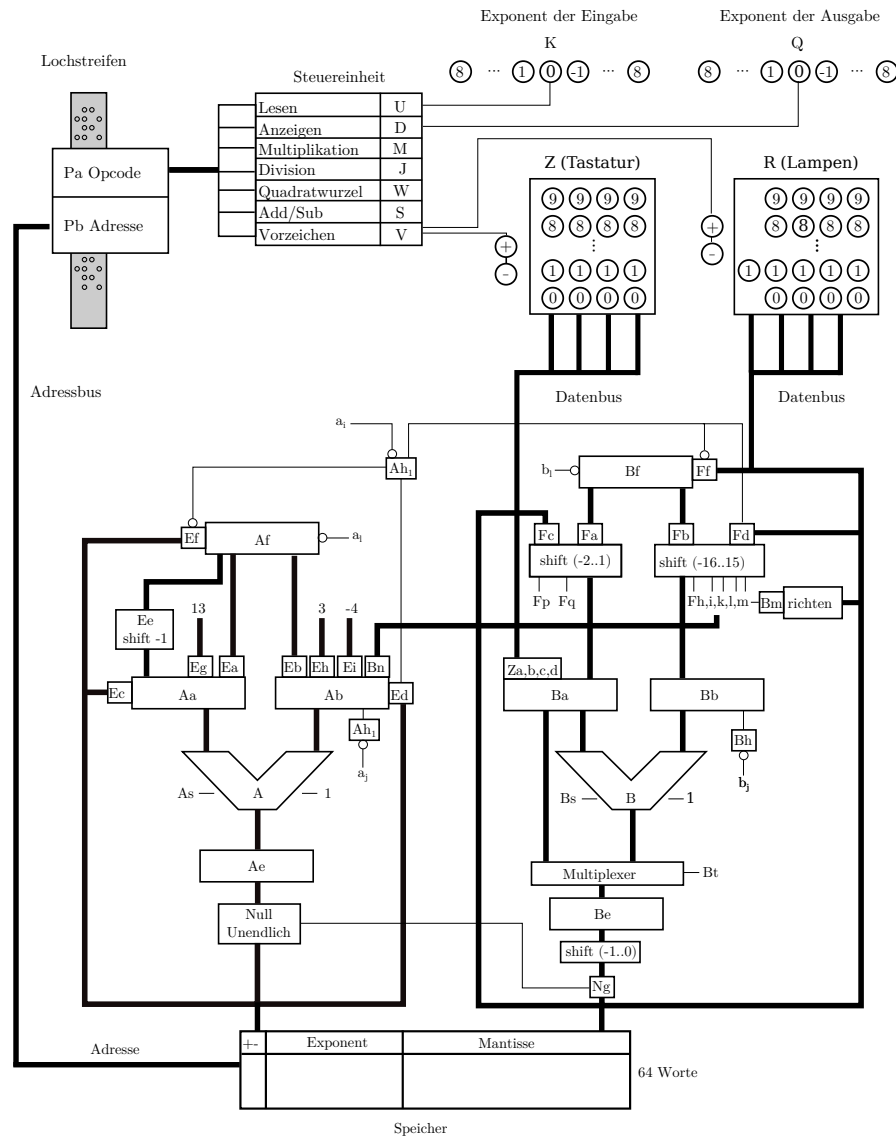


Figure 18: Vollständige Architektur der Z3 nach [3]

Konrad Zuse hat in der Z3 viele Merkmale moderner Rechner realisiert:

- **Verwendung des binären Zahlensystems**
- **Gleitkommazahlberechnung** – Die automatische Gleitkommazahlberechnung wurde bereits 1914 von Torres y Quevedo gefasst, aber er hat diese Rechnung nicht realisiert. Zuse hat den Algorithmus optimiert. Im Zuses Algorithmus werden die Mantisse und der Exponent separat bearbeitet. Zuses Algorithmus ist fast identisch wie den heutigen Prozess für Gleitkommazahlberechnung.
- **Der einschrittige Übertrag**
- **Minimalprinzip des Entwurfs** - Dieses Prinzip eines minimalen Befehlssatzes ist heute in RISC-Architekturen zu finden.

• Trennung von Speicher und Prozessor

4. DER PLANKALKÜL

Konrad Zuses zweiter großer Beitrag zur Computergeschichte ist der Entwurf des Plankalküls. Dieser gilt als die erste hohe Programmiersprache und wurde beispielsweise von Friedrich L. Bauer als Zuses größter Beitrag angesehen.[3] Zu einer Zeit, als es nur zwei funktionierende Computer auf der Welt gab, hat Zuse eine Programmiersprache entworfen, dessen Grundkonzepte sehr modern erscheinen. Die Sprache wurde nie in der Praxis genutzt und besitzt hauptsächlich historische Bedeutung. Im Jahr 2000 wurde an der Freien Universität Berlin ein Teil der Sprache implementiert.

4.1 Grundkonzepte

Der Plankalkül ist eine imperative Programmiersprache, enthält wiederverwendbare Funktionen, Variablen verschiedener Typen, Arrays, Bedingungen und Schleifen. Im Gegensatz zu modernen Programmiersprachen wird der Plankalkül in einem gemischt ein-/zweidimensionalen Format notiert.

4.2 Variablennotation

Es existieren drei Arten von Variablen. Diese werden als V-, Z- und R-Variablen bezeichnet. V-Variablen sind Variablen, die nur gelesen werden können. Z-Variablen können gelesen und geschrieben, R-Variablen nur geschrieben werden. Zusätzlich werden Schleifenvariablen genutzt.

Variablen besitzen einen Typ, der bei ihrer Benutzung stets angegeben wird. Dabei steht „0“ für ein einzelnes Bit. Soll die Variable ein Byte darstellen, so wird der Typ als „8.0“, als eine Struktur aus 8 Bits angegeben. Es ist auch möglich Tupel mit unterschiedlichen Typen zu bilden, welche dann mit Klammern gekennzeichnet werden. „16.(8.0, 16.0)“ beschreibt einen Array mit 16 Elementen, die jeweils eine Variable mit acht Bit und mit 16 Bit beinhalten.

Variablen werden nicht zu Beginn eines Programms deklariert, sondern ihr Typ wird bei jeder Nutzung mit angegeben. Variablen werden vertikal notiert. Die erste Zeile gibt an, ob es sich um eine Eingabevariable V, eine Variable für Zwischenergebnisse Z oder eine Ausgabevariable R handelt. Die Zeile darunter gibt die Nummer der angesprochenen Variablen an. In der dritten Zeile wird die Komponente der Struktur angegeben. Die Indizierung beginnt bei 0. In der letzten Zeile steht der Typ der angesprochenen Variablen. Bild von Variablen mit Zeilenbenennung.

V	Variablentyp: Nur lesbare Variable
1	Variablennummer: Variable 1
0	Element des Feld: Element 0
7.0	Struktur der Variable: Array aus 7 Bits

Figure 19: Variablendarstellung

4.3 Syntax

Neben den arithmetischen Grundoperationen werden in Plankalkül auch Konjunktion, Disjunktion und Verneinung unterstützt. Das folgende Beispiel zeigt eine Addition, wie sie in Plankalkül dargestellt wird. Die Variablen werden wie beschrieben, vertikal dargestellt. Der Pfeil \Rightarrow steht für eine Zuweisung.

V	+	V	\Rightarrow	Z
1		1		3
0		2		
7.0		7.0		7.0

Figure 20: Addition

Die bedingte Ausführung von Code wird in Plankalkül mit „Guarded Commands“ umgesetzt. Ist die logische Bedingung auf der linken Seite des korrekt, wird die Operation auf der rechten Seite ausgeführt. Auch die Vergleichsoperatoren $=$, $<$ und $>$ können in den Bedingungen genutzt werden.

Z	=	Z	\rightarrow	V	+	V	\Rightarrow	Z
1		4		1		1		3
				0		2		
6.0		6.0		7.0		7.0		7.0

Figure 21: Guarded Commands

Ein Anweisungsblock wird durch eckige Klammern gekennzeichnet. Ein solcher Block kann als While-Schleife durchgeführt werden. Diese wird so lange durchgeführt, bis in einem Durchgang alle Bedingungen im Block als Falsch evaluiert wurden. Es ist aber auch möglich, den Block eine gewisse Anzahl von Malen durchzuführen. Dies entspricht dann einer For-Schleife.

$$W \left[\begin{array}{c} C_1 \rightarrow S_1 \\ \vdots \\ C_n \rightarrow S_n \end{array} \right]$$

Figure 22: While-Schleife

Die Deklaration von Funktionen mit Ein- und Ausgangsvariablen erfolgt im „Randauszug“. Die Funktion wird benannt die Parameter werden aufgezählt. Der Aufruf einer Funktion erfolgt mit dem Funktionsnamen gefolgt von den Parametern in Klammern.[2]

P4 Min	(V, V)	\Rightarrow	R
	1 2		3
	7.0 7.0		7.0

Figure 23: Randauszug einer Minimumfunktion

5. FAZIT

Konrad Zuse hat beeindruckende Innovationen sowohl im Bereich der Rechnerentwicklung als auch deren Programmierung hervorgebracht. Er hat sowohl die erste frei programmierbare, in binärer Gleitpunktrechnung arbeitende Rechenmaschine gebaut, als auch die erste hohe Programmiersprache entworfen. Es war in den politischen Verhältnissen der Zeit begründet, dass sein Einfluss auf die Computergeschichte in der zweiten Hälfte des 20. Jahrhunderts geringer ausfiel, als der seiner angloamerikanischen Kollegen.

6. REFERENCES

- [1] N. Eibisch. Der helixturm von konrad zuse.
- [2] G. F. M. K. Ràul Rojas, Cuneyt Göktekin.
Plankal'Technical report.
- [3] R. Rojas, editor. *Die Rechenmaschinen von Konrad Zuse*. Springer, Heidelberg, 1 edition, 1998.
- [4] K. Zuse. *Der Computer – Mein Lebenswerk*. Springer, Heidelberg, 5 edition, 2010.