Sechzig Jahre Computergeschichte – Die Architektur der Rechenmaschinen Z1 und Z3 *

Raúl Rojas †

24. Oktober 1996

Zusammenfassung

Dieser Artikel bietet die *erste* detaillierte Beschreibung der Architektur der Rechenmaschinen Z1 und Z3, die zwischen 1936 und 1941 von Konrad Zuse entworfen wurden. Die notwendigen Informationen wurden durch eine umfassende Auswertung der von Zuse 1941 eingereichten Patentanmeldung erhalten. Zusätzliche Erkenntnisse brachte eine Softwareemulation der Z3. Die Z1 wurde ausschließlich aus mechanischen Komponenten gebaut, die Z3 benutzte elektromagnetische Relais. Beide Maschinen hatten jedoch eine gemeinsame logische Struktur und das Programmiermodell war identisch. Es wird gezeigt, daß die Z1 und die Z3 Eigenschaften besaßen, die in heutigen Computern selbstverständlich sind: Speicher und Prozessor waren getrennte Einheiten, der Prozessor konnte Gleitkommazahlen bearbeiten und beherrschte die vier arithmetischen Grundrechenarten ebenso wie die Quadratwurzelberechnung. Das Programm wurde auf einem Lochstreifen gespeichert und sequentiell gelesen. Der letzte Abschnitt des Artikels setzt die Architektur der Z1 und der Z3 durch einen Vergleich mit anderen Rechenmaschinen in historischen Kontext.

Schlüsselwörter: Konrad Zuse, Rechenmaschinen Z1 und Z3, Geschichte des Computers.

This paper provides the *first* detailed description of the architecture of the computing machines Z1 and Z3 designed by Konrad Zuse in Berlin between 1936 to 1941. The necessary basic information was obtained from a careful evaluation of the patent application filed by Zuse in 1941. Additional insight was gained from a software simulation of the machine's logic. The Z1 was built using purely mechanical components, the Z3 using electromechanical relays. However, both machines shared a common logical structure and the programming model was the same. We argue that both the Z1 and the Z3 possessed features akin to those of modern computers: memory and processor were separate units, the processor could handle floating-point numbers and compute the four basic arithmetical operations as well as the square root of a number. The program was stored on punched tape and was read sequentially. In the last section of this paper we bring the architecture of the Z1 and Z3 into historical perspective by offering a comparison with computing machines built in other countries.

Keywords: Konrad Zuse, computing machines Z1 and Z3, history of computing.

Computing Reviews Classification: K.2, B.2.0.

1 Frühe Rechenmaschinen

Konrad Zuse wird in Deutschland gemeinhin als der Vater des Computers angesehen und seine Z1, ein zwischen 1936 und 1938 gebauter programmierbarer Automat, wird hierzulande als der erste Computer der Welt bezeichnet. Andere Nationen reklamieren dieses Privileg für einen ihrer eigenen Wissenschaftler und es hat eine lange und oft bittere Debatte über den "wahren" Erfinder des

*erscheint im *Informatik Spektrum, Zeitschrift der GI*
†email: rojas@informatik.uni-halle.de

Computers gegeben. Manchmal wird die Diskussion vorab mit Beschlag belegt durch die genaue Spezifizierung von technischen Eigenschaften einer speziellen Maschine. Der ENIAC (Akronym für *Electronic Numerical Integrator and Computer*) ist zum Beispiel als der erste *allgemeine, großformatige, elektronische* Computer der Welt bezeichnet worden [2]. Diese Maschine wurde an der *Moore School of Electrical Engineering* an der Universität von Pennsylvania von Mai 1943 bis 1945 gebaut. Der ENIAC löste sein erstes Problem im Dezember 1945 und wurde im Februar 1946 offiziell eingeweiht.

Ein weiterer Anwärter auf den Titel des ersten Computers ist der Mark I, der von Howard Aiken an der Harvard Universität zwischen 1939 und 1944 gebaut wurde. Der Mark I war eine elektromechanische Maschine, d.h. er war nicht total mechanisch wie frühere Rechenmaschinen, verwendete aber auch nicht die damals bereits verfügbaren elektronischen Komponenten [1]. Die Maschine von John Atanasoff (die später ABC genannt wurde), die zwischen 1938 und 1942 am Iowa State College gebaut wurde, benutzte Vakuumröhren, konnte aber nur Vektoren addieren und subtrahieren und hatte nicht die für allgemeine Berechnungen notwendige minimale Struktur [3]. Im direkten Gegensatz zu diesen drei Maschinen waren Z1 und Z3 weit flexibler; sie konnten eine lange Folge von Befehlen ausführen, die auf einem Lochstreifen gestanzt war. Die Maschinen von Zuse waren mechanisch oder elektromechanisch und von kleineren Abmessungen. Weil die Z1 vor dem Mark I fertiggestellt wurde, wird sie als die erste *programmierbare vollautomatische* Rechenmaschine der Welt bezeichnet. Natürlich kann die alte Debatte mit einem einizigen Artikel nicht beendet werden, es wird jedoch in den nächsten Abschnitten gezeigt, wie fortschrittlich die Maschinen von Zuse aus dem Blickwinkel der modernen Computerarchitektur waren und wie gut sie im direkten Vergleich mit anderen Entwürfen ihrer Zeit abschneiden.

In den dreißiger Jahren fing Konrad Zuse noch als Student an, über Rechenmaschinen nachzudenken. Er erkannte, daß er in der Lage war einen Automaten zu konstruieren, der Folgen von arithmetischen Operationen ausführen konnte, wie sie zur Durchführung ingenieurmäßigen Berechnungen notwendig sind. Als Bauingenieur hatte er keine Ausbildung in Elektrotechnik oder Elektronik erhalten und war auch nicht mit der Technik vertraut, die in konventionellen mechanischen Rechnern benutzt wurde. Dieses nominelle Defizit zahlte sich jedoch zu seinem Vorteil aus, denn er mußte das ganze Problem der arithmetischen Berechnungen überdenken und kam so zu neuen Lösungen.

Zuse entschied sich, eine erste experimentelle Rechenmaschine zu bauen, die zwei wesentliche Ideen umsetzte: a) die Maschine arbeitete mit binären Zahlen; b) Rechen- und Steuereinheiten wurden vom Speicher getrennt. Jahre bevor John von Neumann die Vorteile einer Computerarchitektur schriftlich begründete, in der Prozessor und Speicher getrennt sind, war Zuse bereits auf den gleichen Gedanken gekommen. Es ist jedoch anzumerken, daß diese Idee auf Charles Babbage zurückgeht, der im vorigen Jahrhundert beim Entwurf der *Analytischen Maschine* zum selben Konstruktionsprinzip kam. Im Jahr 1936 wurde der Speicher[1] der von Zuse geplanten Maschine fertiggestellt. Es war ein mechanisches Gerät, aber nicht vom üblichen Typ. Zuse implementierte logische und arithmetische Operationen mit gestanzten Blechschienen anstelle von Zahnrädern (wie sie von Babbage im vorigen Jahrhundert benutzt wurden). Die Blechschienen konnten sich nur in zwei Richtungen bewegen (vorwärts und rückwärts) und waren deswegen ausreichend für eine binär arbeitende Maschine [16]. Der Prozessor der Z1 wurde einige Monate nach dem Speicher fertiggestellt und verwendete die gleiche Art von Technik für die einzelnen Komponenten. Er arbeitete zusammen mit dem Speicher, war aber niemals sehr zuverlässig. Das Hauptproblem war die präzise Synchronisation, die gebraucht wurde, um übermäßige mechanische Beanspruchungen der bewegenden Teile zu vermeiden. Es ist interessant anzumerken, daß im gleichen Jahr, als der Speicher der Z1 fertig wurde, Alan Turing seinen bahnbrechenden Artikel über berechenbare Zahlen schrieb, in dem er das intuitive Konzept von Berechenbarkeit formalisierte.

Obgleich die Z1 unzuverlässig blieb, zeigte sich, daß der Entwurf konsistent war und dies trieb Zuse dazu, andere mögliche Realisierungen zu erforschen. Er entschied sich für elektromechanische Relais, die vor und während des 2. Weltkrieges billiger und leichter als andere Komponenten zu

[1] Zuse bezeichnete ihn als "Speicherwerk". Der Begriff "Speicher" wird im Deutschem nach wie vor anstelle des antropomorphen Begriffs "memory" benutzt, den John von Neumann im Englischen einführte. Charles Babbage verwendete das Wort "store".

erhalten waren. Ein Versuchsmodell (später Z2 genannt) hatte einen Prozessor bestehend aus Relais und den mechanischen Speicher der Z1. Bald danach fing Zuse an die Z3 zu bauen, eine Maschine, die ausschließlich aus Relais bestand, die aber die gleiche logische Struktur wie die Z1 besaß. Die Z3 war 1941 fertig und einsatzbereit, vier Jahre vor der ENIAC.

Dieser Artikel bietet die erste detaillierte Darstellung der gemeinsamen Architektur von Z1 und Z3. Die Z1 wurde von Zuse selbst in den achtziger Jahren in Berlin rekonstruiert und ist nun eine der Austellungsattraktionen des Berliner Museums für Verkehr und Technik. Jedoch beschreiben die bisher verfügbaren Informationen nur den Entwurf des mechanischen Speichers [13]. Die Z3 wurde von Zuse in der Patentanmeldung Z-391 von 1941 dokumentiert, diese ist aber schwer zu durchschauen durch die nicht standardisierte Notation und Terminologie [15]. Czaudernas Buch über die Z3 ist eine gute Quelle, um das historische Umfeld von Zuses Erfindung zu verstehen, beschreibt aber die Z3 nicht im Detail [4]. Im folgenden wird nur auf die Z3 eingegangen, weil die Z1 und die Z3 vom Standpunkt der Logik und funktional praktisch äquivalent waren. Der maßgebliche Unterschied in der Architektur von Z1 und Z3 ist die Tatsache, daß die Quadratwurzeloperation in der Z1 fehlte. Es gab außerdem unwesentliche Unterschiede in der Anzahl der Bits, die der Prozessor für arithmetische Operationen benutzte (die Z1 verwendete ein Bit weniger für die Mantisse von Gleitkommazahlen) und in der Anzahl der Zyklen, die für jeden Befehl benötigt wurden. Mit diesen leichten Einwänden und hauptsächlich die architektonischen Eigenschaften betrachtend, kann von der Z1 und der Z3 als von äquivalenten Maschinen gesprochen werden, für deren Schaltungen Zuse auch eine einheitliche Beschreibung ("abstrakte Schaltgliedtechnik", [16], S. 56) entwickelte.

2 Überblick über die Architektur von Z1 und Z3

Dieser Abschnitt faßt die wichtigen architektonischen Eigenschaften der Z3 zusammen. Die Darstellung bewegt sich vom Einfachen zum Komplexen: zuerst wird ein Überblick geboten und in Abschnitt 3 werden die Details besprochen. Um einen einfachen Satzbau zu ermöglichen wird von der Z3 im Präsens gesprochen.

Struktureller Aufbau

Die Z3 ist eine Maschine, die Gleitkommazahlen verarbeitet. Während die anderen damaligen Rechenautomaten wie der Mark I, der ABC und der ENIAC mit Festkommazahlen arbeiteten, hatte Zuse sich bereits viel früher dafür entschieden, die "halblogarithmische" Notation zu verwenden. Diese entspricht der modernen Darstellung von numerischen Größen als Gleitkommazahlen.

Abbildung 1 zeigt einen Überblick der Grundstruktur der Z3. Das erste relevante Merkmal ist die Trennung von Prozessor und Speicher. Die Z3 besteht aus einer binären Speichereinheit (mit einem Speichervermögen von 64 Gleitkommazahlen), einem binären Gleitkommazahlprozessor, einer Steuereinheit und Ein- bzw. Ausgabegeräten. Speicher und arithmetische Einheit sind über einen Datenbus verbunden, der Exponent und Mantisse der Gleitkommazahldarstellung überträgt. Die Steuereinheit enthält für jeden Befehl einen Mikrosequenzer. Signalleitungen, die von der Steuereinheit zum Prozessor, dem Speicher und den Ein- bzw. Ausgabegeräten gehen, sorgen für die richtige Synchronisation aller Geräte. Der Lochstreifenleser liefert den Befehlskode für jeden Befehl ebenso wie die Adresse für Speicherzugriffe. Die Ein- bzw. Ausgabegeräte sind durch den Datenbus mit der Recheneinheit verbunden.

Darstellung von Gleitkommazahlen

Abbildung 2 zeigt die Darstellung der Gleitkommazahlen im Speicher der Z3. Das erste Bit wird benützt, um das Vorzeichen der Zahl zu speichern, die folgenden 7 Bits sind für den Exponenten und die letzten 14 Bits für die Mantisse (nur die 14 Stellen rechts von Dezimalpunkt). Die Bits vom Exponenten werden "Teil A" der Zahl genannt und als $a_6, \ldots, a_0$ bezeichnet. Die Bits der Mantisse werden "Teil B" genannt und als $b_0, b_{-1}, \ldots, b_{-14}$ geschrieben. Der Exponent ist eine

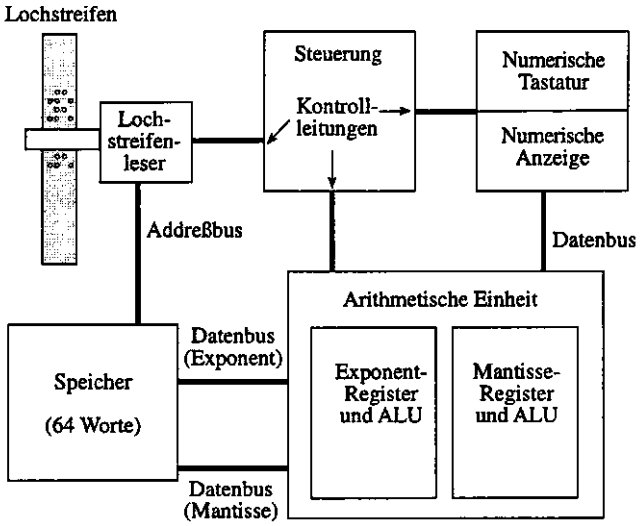

Abbildung 1: Die Bausteine der Z3

Zahl im Zweierkomplement. Der Zahlenumfang reicht deswegen von 2^{-64} bis 2^{63}. Die Mantisse wird in normalisierter Form gespeichert, das heißt, die erste Ziffer vor dem Dezimalpunkt (b_0) muß immer eine 1 sein. Diese Ziffer braucht nicht gespeichert zu werden (und erscheint deswegen nicht in Abbildung 2), so daß der effektive Zahlenbereich im Speicher äquivalent zu einer Mantisse von 15 Bits ist. Es gibt jedoch ein Problem mit der Zahl Null, die nicht mit einer normalisierten Mantisse dargestellt werden kann. Die Z3 benutzt die Konvention, daß jede Mantisse mit dem Exponenten -64 als Null zu interpretieren ist. Außerdem gilt jede Zahl mit dem Exponenten 63 als die Zahl Unendlich. Operationen, die Null oder Unendlich beinhalten, werden als Ausnahmen behandelt. Spezielle Hardware überwacht jede in den Prozessor eingelesene Zahl, um die Ausnahmebits (siehe Abschnitt 4) zu setzen.

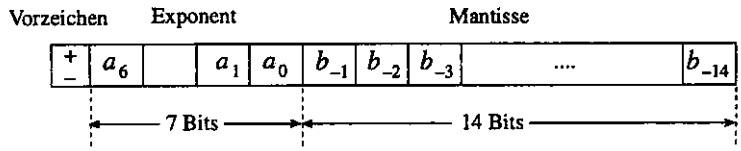

Abbildung 2: Die Darstellung von Gleitkommazahlen im Speicher

Mit dieser Konvention ist die kleinste darstellbare Zahl im Speicher der Z3 die Zahl $2^{-63} \approx 1.08 \times 10^{-19}$ und die größte ist $1,999 \times 2^{62} \approx 9.2 \times 10^{18}$. Die Argumente für Berechnungen können als Dezimalzahlen (vier Ziffern) an der Tastatur der Z3 eingegeben werden. Der Exponent der dezimalen Darstellung wird durch Drücken der entsprechenden Position in einer mit $-8, -7, \ldots, 7, 8$ beschrifteten Reihe von Tasten eingegeben. Die ursprüngliche Z3 konnte lediglich Eingaben zwischen 1×10^{-8} und 9999×10^{8} annehmen. Die Rekonstruktion der Z3, die Zuse für das Deutsche Museum im München baute, bietet genügend Tasten für größere Exponenten. Mit dieser Anordnung kann die numerische Kapazität der Maschine bei der Eingabe ausgeschöpft werden. Das gleiche kann über die Ausgabe gesagt werden. Die Z3 kann das vom Programm produzierte numerische Ergebnis nicht drucken. Statt dessen wird eine einzelne Zahl mit einer Matrix von Lampen angezeigt, die die Ziffern 0 bis 9 darstellen. Die größte Zahl, die angezeigt werden kann, ist 19999 und die kleinste ist 00001. Der größte Exponent, der angezeigt werden kann, ist $+8$, der kleinste -8. Die Maschine in München benutzt für die Ausgabe des Exponenten wiederum einen größeren Ausgabebereich.

Befehlssatz

Das Programm für die Z3 befindet sich auf einem Lochstreifen. Die acht Bits in jeder Zeile kodieren einen Befehl. Der Befehlssatz der Z3 besteht aus den neun in Tabelle 1 aufgelisteten Befehlen. Es gibt drei Klassen davon: Ein-/Ausgabe-, Speicher- und Arithmetikbefehle. Der Befehlskode hat eine variable Länge von zwei bis fünf Bits. Speicheroperationen beinhalten die Adresse eines Speicherwortes in den niederwertigen sechs Bits, d.h. der Adreßraum hat wie bereits erwähnt eine maximale Größe von 64 Worten.

Tabelle 1: Befehlssatz und Befehlskodes der Z3

Klasse	Befehl	Beschreibung	Befehlskode
Ein-/Ausgabe	Lu	Lesen von Tastatur	01 110000
	Ld	Ergebnis anzeigen	01 111000
Speicher	Pr z	Lesen von Adresse z	11 $z_6z_5z_4z_3z_2z_1$
	Ps z	Speichern in Adresse z	10 $z_6z_5z_4z_3z_2z_1$
Arithmetik	Lm	Multiplikation	01 001000
	Li	Division	01 010000
	Lw	Quadratwurzel	01 011000
	Ls_1	Addition	01 100000
	Ls_2	Subtraktion	01 101000

Die Befehle können in beliebiger Reihenfolge kombiniert werden. Die Befehle Lu und Ld (Lese von Tastatur, Anzeige des Ergebnisses) halten die Maschine an, so daß der Bediener genügend Zeit hat, um eine Zahl einzugeben oder das Ergebnis aufzuschreiben. Die Maschine wird dann neu gestartet und setzt die Ausführung des Programms fort.

Die offensichtlichste Lücke in dem Befehlssatz der Z3 ist das Nichtvorhandensein von Verzweigungen. Eine Schleife kann einfach durch das Zusammenkleben der beiden Enden eines Lochstreifens realisiert werden, aber es gibt keine Möglichkeit zur Implementierung von bedingten Sprüngen in Befehlsfolgen. Schon aus diesem Grund kann man die Z3 nicht als einen Universalrechner im Sinne von Turing einstufen.

Anzahl der Zyklen

Die Z3 ist eine getaktete Maschine. Jeder Takt (ein Spiel in der Terminologie von Zuse) ist in die fünf römisch nummerierten "Stufen" I, II, III, IV und V unterteilt. Der momentane Befehl auf dem Lochstreifen wird in Stufe I eines Zyklus dekodiert. Die zwei grundlegenden arithmetischen Operationen der Maschine sind Addition und Subtraktion von Exponent und Mantisse. Diese Operationen können in den ersten drei Stufen eines jeden Zyklus ausgeführt werden. Die Stufen IV und V werden benutzt, um die Argumente für die nächste Operation vorzubereiten oder um ein Ergebnis in die Register oder in den Speicher zurückzuschreiben.

Die in der Z3 implementierten Befehle benötigen die folgende Anzahl von Zyklen:

Multiplikation:	16 Zyklen
Division:	18 Zyklen
Quadratwurzel:	20 Zyklen
Addition:	3 Zyklen
Subtraktion:	4 oder 5 Zyklen (abhängig vom Ergebnis)
Lesen von Tastatur (Eingabe):	9 bis 41 Zyklen (abhängig vom Exponenten)
Ergebnis anzeigen (Ausgabe):	9 bis 41 Zyklen (abhängig vom Exponenten)

Lesen aus Speicher:	1 Zyklus
Schreiben in Speicher:	0 oder 1 Zyklus

Nach Aussage von Zuse wurden für eine Multiplikation drei Sekunden benötigt. Da eine Multiplikation 16 Zyklen braucht, können wir annehmen, daß die Z3 eine Taktfrequenz von (16/3) ≈5.33 Hz erreicht hat[2].

Die Anzahl der Zyklen für Ein- und Ausgabebefehle ist variabel, weil sie vom Exponenten der Argumente abhängig ist. Da die Eingabe von der Dezimal- in die Binärdarstellung umgewandelt werden muß, wird die Anzahl der Multiplikationen, die mit dem Faktor 10 oder 0.1 benötigt werden, vom dezimalen Exponenten bestimmt (siehe auch Abschnitt 4).

Addition und Subtraktion benötigen mehr als einen Zyklus, weil im Falle von Gleitkommazahlen der Exponent von beiden Argumenten vereinheitlich werden muß. Dies erfordert einige zusätzliche Vergleiche und Verschiebungen.

Eine Zahl kann in *Null* Zyklen in den Speicher geschrieben werden, wenn das Ergebnis der letzten arithmetischen Operation auf die gewünschte Speicheradresse umgeleitet werden kann. Der für das Speichern benötigte Zyklus überlappt sich mit dem letzten Zyklus der arithmetischen Operation.

Programmiermodell

Es ist sehr wichtig, das Programmiermodell, d.h. die für den Programmierer abstrakt vorhandenen Bestandteile der Maschine zu beschreiben. Vom Standpunkt der Software besteht die Z3 aus 64 Speicherworten, die in zwei Gleitkommaregister (bezeichnen wir sie einfach mit R1 und R2) geladen werden können. Diese zwei Register beinhalten die beiden Argumente der dazugehörigen arithmetischen Operation. Der Programmierer kann eine beliebige Folge von Befehlen schreiben, aber er muß dabei den Zustand der Register im Auge behalten.

Vor allem muß der folgende Punkt beachtet werden: die erste Speicher-Leseoperation in einem Programm (Pr z) überträgt den Inhalt der Adresse z nach R1. Jede nachfolgende Speicher-Leseoperation überträgt ein Speicherwort vom Speicher nach R2. Ein Befehl zum Lesen von der numerischen Tastatur überträgt die numerische Eingabe in das Register R1 und *löscht* den Inhalt von R2.

Die Argumente für arithmetische Operationen sind im Befehlskode nicht explizit angegeben. Ihre implizite Semantik ist die folgende:

Multiplikation:	R1:=R1×R2
Division:	R1:=R1/R2
Addition:	R1:=R1+R2
Subtraktion:	R1:=R1−R2
Quadratwurzel:	R1:=$\sqrt{\text{R1}}$

R2 wird nach jeder arithmetischen Operation auf 0 gesetzt, während das Ergebnis in R1 gespeichert wird. Nachfolgende Leseoperationen überschreiben R2. Der Speichern-Befehl und die Ausgabe des Ergebnisses beziehen sich immer auf R1. Nach einem Speichern- oder Ausgabebefehl wird R1 auf 0 gesetzt. Die nächste Leseoperation bezieht sich dann auf R1.

Ein Beispiel ist an dieser Stelle besser als weitere Erklärungen, um das Programmiermodell der Z3 zu verdeutlichen. Nehmen wir an, wir wollen ein Polynom nach der Methode von Horner berechnen:

$$((a_4x + a_3)x + a_2)x + a_1$$

Nehmen wir weiterhin an, daß die Konstanten a_4, a_3, a_2 und a_1 in den Adressen 4, 3, 2 bzw. 1 gespeichert sind. Die Zahl x ist in der Adresse 5 gespeichert. Das Programm für die gewünschte Berechnung lautet:

[2]Es ist sicherlich ein kurioser Zufall, daß die Gatterebenensimulation, die meine Studenten für die Z3 implementierten, ebenfalls ca. drei Sekunden für eine Multiplikation benötigte!

Pr 4	Lade a_4 in R1
Pr 5	Lade x in R2
Lm	Multipliziere R1 und R2, Ergebnis in R1
Pr 3	Lade a_3 in R2
Ls_1	Addiere R1 und R2, Ergebnis in R1
Pr 5	Lade x in R2
Lm	Multipliziere R1 und R2, Ergebnis in R1
Pr 2	Lade a_2 in R2
Ls_1	Addiere R1 und R2, Ergebnis in R1
Pr 5	Lade x in R2
Lm	Multipliziere R1 und R2, Ergebnis in R1
Pr 1	Lade a_1 in R2
Ls_1	Addiere R1 und R2, Ergebnis in R1
Ld	Ausgabe des Ergebnisses

Nach der Ausführung des letzten Befehls wird der Prozessor in seinen Ausgangszustand zurückversetzt. Eine neues Programm kann gestartet werden.

3 Blockdiagramm der Z3

In diesem Abschnitt beschäftigen wir uns genauer mit der Struktur der Z3 und beschreiben detaillierter ihre wesentlichen Bausteine. Der wichtigste Punkt betrifft die Sicherstellung der richtigen Synchronisation der verfügbaren Komponenten.

Der Prozessor

Abbildung 3 zeigt eine vereinfachte Darstellung der arithmetischen Einheit der Z3. Es gibt zwei Teile: die linke Seite wird für Operationen mit den Exponenten von Gleitkommazahlen benutzt, die rechte Seite für Operationen mit den Mantissen. Af und Bf sind Register, die jeweils für die Speicherung von Exponenten und Mantisse derselben Gleitkommazahl benutzt werden. Dies ist aus Sicht des Programmierers das Register R1. Im folgenden wird deswegen auf R1 als Registerpaar <Af,Bf> Bezug genommen. Das Registerpaar <Ab,Bb> speichert Exponenten und Mantisse vom Register R2 des Programmiermodells. Das Paar <Aa,Ba> enthält den Exponenten und die Mantisse eines temporären dritten Gleitkommazahlregisters, das dem Programmierer verborgen bleibt. Die beiden Addierer A und B werden jeweils für die Addition und die Subtraktion von Exponenten und Mantisse benutzt. Das Ergebnis für den Exponentenanteil einer Operation wird in Ae abgelegt. Für den Mantissenanteil wird das Ergebnis in Be abgelegt. Im Teil B erlaubt ein Multiplexer die Wahl, ob Ba oder die Ausgabe vom Addierer B als Ergebnis einer Operation in Be selektiert wird. Der Multiplexer wird vom Relais Bt gesteuert (wenn Bt=0, dann wird Ba nach Be übertragen).

Die kleinen mit Ea, Eb, Ec, Ed, Ef, Fa, Fb, Fc, Fd und Ff beschrifteten Kästchen stellen Relaisschalter dar, die den Datenbus öffnen oder schließen. Wenn zum Beispiel der Inhalt von Register Af zum Register Aa übertragen werden soll, dann wird der Relaisschalter Ea auf 1 gesetzt und das Ergebnis ist Aa:=Af. Wie im Diagramm zu erkennen ist, kann der Inhalt von Af entsprechend dem Zustand der Relaisschalter nach Aa oder Ab übertragen werden. Der Inhalt von Ae kann nach Aa, Ab oder Af übertragen werden. Die Struktur von Teil B der arithmetischen Einheit ist der von Teil A sehr ähnlich, aber zusätzlich zum von Bt kontrollierten Multiplexer gibt es einen Schifter zwischen Bf und Ba und ebenso einen zwischen Bf und Bb. Der erste Schifter kann die Mantisse um bis zu zwei Stellen nach rechts oder um eine Stelle nach links verschieben. Dies bedeutet entweder eine Division durch 4 oder eine Multiplikation mit 2. Der zweite Schifter kann die Mantisse in Af zwischen einer und 15 Stellen nach links sowie zwischen einer und 16 Stellen nach rechts verschieben. Diese Verschiebungen sind für die Addition und die Subtraktion von Gleitkommazahlen notwendig. Die Multiplikation und die Division um den Faktor 2 sind auf

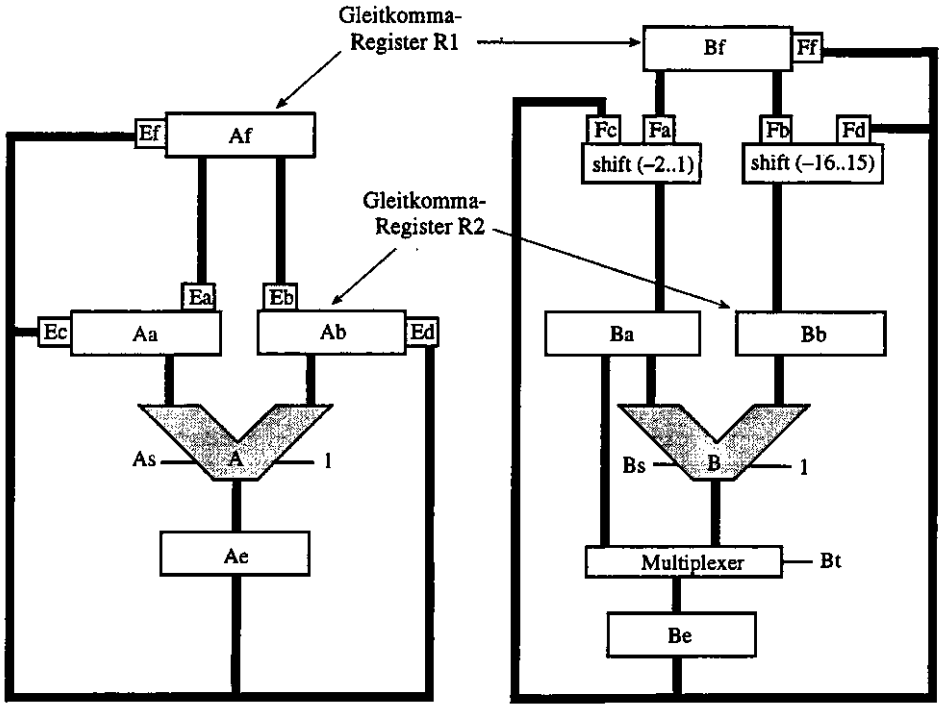

Abbildung 3: Register und Rechenwerk

diese Weise trivial zu berechnen, während die Operanden für die nächste arithmetische Operation geholt werden, und verbrauchen deswegen keine zusätzliche Zeit.

Die Bitlänge der Register ist folgende:

Af	7 Bits	Bf	17 Bits
Aa	7	Ba	19
Ab	7	Bb	18
Ae	8	Be	18

Wie aus dieser Liste ersichtlich ist, benutzt Ae ein extra Bit, um die Addition der Exponenten zu bearbeiten. Teil B des Prozessors benutzt zwei extra Bits für die Mantissenanteile (b_{-15}, b_{-16}) und verwendet explizit Bit b_0, das nicht gespeichert wird. Die extra Bits an den Positionen -15 und -16 erlauben es, die Genauigkeit der Berechnung zu erhöhen. Damit beträgt die Anzahl der Bits, die für das Speichern des Ergebnisses einer arithmetischen Operation im Register Bf gebraucht werden, 17 Bits. Register Ba und Bb benötigen weitere extra Bits (ba_2, ba_1, und bb_1), um die Zwischenergebnisse einiger numerischen Algorithmen aufnehmen zu können. Insbesondere der Quadratwurzelalgorithmus kann in Ba zu Teilberechnungen führen, die bis zu drei Einsen links vom Dezimalpunkt enthalten.

Die wesentlichen primitiven Operationen in diesem Kreislauf sind Addition und Subtraktion von Exponenten und Mantissen. Falls das Relais As (Bs) gesetzt ist, wird der Negativwert des zweiten Arguments Ab (Bb) in den Addierer geladen. Der Addierer im Teil A subtrahiert also die Argumente, wenn das Relais As auf 1 gesetzt ist, ansonsten werden die Argumente addiert. Das

gleiche gilt für den Teil B und das Relais Bs. Die in der Abbildung dargestellte Konstante 1 wird gebraucht, um das Zweierkomplement einer Zahl zu erzeugen.

Nehmen wir an, daß zwei Zahlen mit dem gleichen Exponenten addiert werden sollen. Der erste Exponent wird in Af gespeichert, der zweite in Ab. Im Teil A der Maschine muß keine besondere Aktion getätigt werden, weil beide Exponenten gleich sind. Im Teil B wird die Mantisse der ersten Zahl in Bf gespeichert und die Mantisse der zweiten in Bb. Der erste Schritt besteht darin, Ba mit Bf zu laden, indem der Relaisschalter Fa auf 1 gesetzt wird. Die Addition wird als nächstes ausgeführt, das Relais Bt wird auf 1 gesetzt und damit Be das Ergebnis Ba+Bb zugewiesen. Der Relaisschalter Ff wird nun auf 1 gesetzt und das Ergebnis in Bf gespeichert. Wie zu sehen ist, kann die Information zwischen den Registern bewegt werden und zirkuliert damit in einem Kreislauf. Der Computerarchitekt muß die richtige Konfiguration der Relaisschalter bereitstellen, um die gewünschte Operation zu erhalten. Dies wird in der Z3 mit einer Technik bewerkstelligt, die der Mikroprogrammierung sehr ähnlich ist.

Die Steuereinheit

Abbildung 4 zeigt ein Detaildiagramm der Steuereinheit und der Ein-/Ausgabeeinheiten. Die Steuereinheit bestimmt die richtige Mikrosequenzierung der Befehle. Es gibt spezielle Schaltkreise für jeden Befehl aus dem Befehlssatz. Der Schaltkreis Pa dekodiert den Befehlskode des vom Lochstreifen gelesenen Befehls. Wenn es ein Speicherbefehl ist, setzt Pb den Adreßbus auf den Wert der niederwertigen sechs Bits des Befehlskodes.

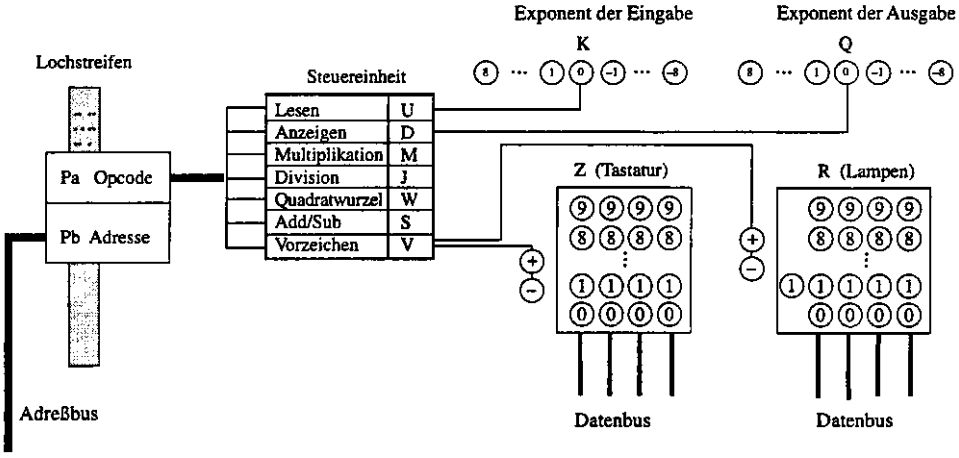

Abbildung 4: Die Steuereinheit und die Ein-/Ausgabeeinheiten

Der Schaltkreis Z stellt die Tastatur dar, die für die Eingabe von Dezimalzahlen in die Maschine benutzt wird. Nur ein Schalter in jeder der vier Spalten kann aktiviert werden. Der Exponent wird im Schaltkreis K durch das Drücken einer der mit -8 bis 8 beschrifteten Tasten gesetzt. Die Ausgabeeinheit ist ähnlich der Eingabeeinheit, allerdings zeigen hier aufleuchtende Lampen die entsprechenden Dezimalziffern, den Exponenten der Zahl (Schaltkreis Q) und das Vorzeichen an. Zu beachten ist eine fünfte Ziffer für die Ausgabe, die bei der Z3 nur 1 oder 0 sein kann.

Wenn einmal eine Dezimalzahl eingeben wurde, dann überträgt der Datenbus die Ziffern in das Register Ba und eine komplexe Serie von Operationen wird gestartet. Die Dezimaleingabe muß in eine Binärzahl umgewandelt werden. Dies bedeutet eine Reihe von Multiplikationen, deren Anzahl proportional zum absoluten Wert des Exponenten ist. Für den Exponenten 0 braucht die ganze Umwandlung 9 Zyklen. Für den Exponenten 8 sind jedoch $9 + 4 \times 8 = 41$ Zyklen notwendig.

Mikroprogrammsteuerung der Z3

Das Herz der Steuereinheit sind Mikrosequenzer. Bevor ihre Arbeitsweise beschrieben wird, ist es nötig, einen Blick auf die Verkettung von arithmetischen Befehlen in der Z3 zu werfen. Abbildung 5 zeigt die Grundidee. Jeder Zyklus in der Z3 wird in 5 Stufen aufgeteilt. Die Stufen IV und V werden benutzt, um Information von einem Teil der Maschine zu einem anderen zu übertragen. Während der Stufen I, II und III wird eine Addition oder Subtraktion im Teil A und eine andere im Teil B der Z3 ausgeführt. Wir bezeichnen dies als die "Ausführungsphase" des Befehls. Ein typischer Befehl holt seine Argumente, führt eine Operation aus und schreibt das Ergebnis zurück. Zuse hat großen Wert darauf gelegt, Ausführungszeit dadurch zu sparen, daß die Argumentevorbereitungsphase des nachfolgenden Befehls mit der Ergebnissicherungsphase des gegenwärtigen Befehls überlappt wird. Wir können uns einen Befehlsausführungszyklus denken, der aus lediglich zwei Phasen besteht, wie in Abbildung 5 zu sehen ist, in dem die ersten beiden Zyklen einer Reihe von Befehlen dargestellt werden. Diese Konvention wurde in den Tabellen über die numerischen Algorithmen übernommen, die weiter unten diskutiert werden.

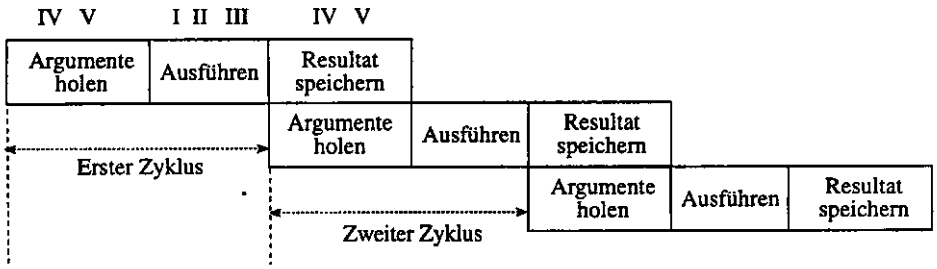

Abbildung 5: Die Ausführungspipeline der Z3

Die Mikrosequenzierung erfolgt durch spezielle Schrittschalter. Es gibt eines für den Multiplikationsalgorithmus, ein anderes für die Division und ein weiteres für die Quadratwurzeloperation. Der in Abbildung 6 gezeigte bewegliche Arm fängt an, sich im Uhrzeigersinn zu bewegen, sobald die Steuereinheit den entsprechenden Befehl dekodiert. In jedem Zyklus bewegt sich der Arm von einer Position zur nächsten. Der Arm steht unter elektrischer Spannung und aktiviert die Schaltkreise, mit denen er in Berührung kommt. In dem Beispiel in der Abbildung setzt der bewegte Arm den Relaisschalter Ea auf 1 im ersten Zyklus. Dies führt zur Übertragung des Inhalts von Register Af nach Aa. Im nächsten Zyklus werden die Relaisschalter Ec und Fc aktiviert. Auf diese Weise werden die Ergebnisse der Operation in den Teil A und B der Maschine in die Register Aa bzw. Bb zurückgeschrieben. Wie man leicht sehen kann, bietet ein derartiges Schrittschalter eine komfortable Plattform zur Veränderung der genauen Folge von Schaltereignissen für eine Operation. Die Schrittschalter entsprechen den Mikrosequenzern, die in heutigen Mikroprozessoren benutzt werden. Ich möchte dies nicht als echte Mikroprogrammierung bezeichnen, weil in diesem Fall die Mikrosequenzen fest verschaltet sind, es ist jedoch klar, daß Mikroprogrammierung und Mikrosequenzierung nahe miteinander verwandt sind.

Die umfassende Nutzung der Mikrosequenzierung erlaubte es Zuse, die Z3 zu vereinfachen. Sobald die grundsätzlichen Schaltkreise entworfen waren, war es lediglich eine Frage der Verfeinerung der Steuerung, bis die optimalen Folgen von Schaltereignisse gefunden waren. Es gibt einige Details, die vom Programmierer beim Entwurf des "Mikroprogramms" beachtet werden müssen, weil sonst Kurzschlüsse die Hardware zerstören können. Die Z1 mit ihrem mechanischen Design war in diesem Sinne noch weit sensibler als die Z3. Auch nachdem sie fertiggestellt war, gab es Befehlsfolgen, die ein Programmierer vermeiden mußte, um die Hardware nicht zu beschädigen. Eine solche Befehlsfolge wurde versehentlich 1994 in der rekonstruierten Z1 im Berliner Museum für Verkehr und Technik gestartet und führte zu einer leichten Beschädigung der Maschine.

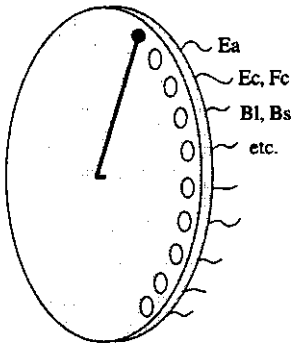

Abbildung 6: Schrittschalter für die Mikrosequenzer

Teil II

4 Arithmetische Algorithmen

Übertrag bei der Addition

Ein wichtiger Aspekt der Z3 ist der Entwurf eines Addierers, der in der Lage ist, Additionen und Subtraktionen unter Nutzung der Technik der *carry look-ahead* zu berechnen. Wird die parallele binäre Addition direkt und ohne eine solche Optimierung implementiert, dann müssen Übertragungsbits von einer Bitposition zur nächsten weitergegeben werden. Für die Mantisse wären dann mindestens 16 Zyklen für die sichere Weitergabe der Übertragsbits notwendig.

Die Addierer der Z3 sind viel schneller - sie leisten eine Addition oder eine Subtraktion in den Stufen I, II oder III von einem einzigen Zyklus. Die Subtraktion wird durch Komplementierung des zweiten Argumentes und Addition einer zusätzlichen 1 auf der niederstwertigen Bitposition auf eine Addition reduziert.

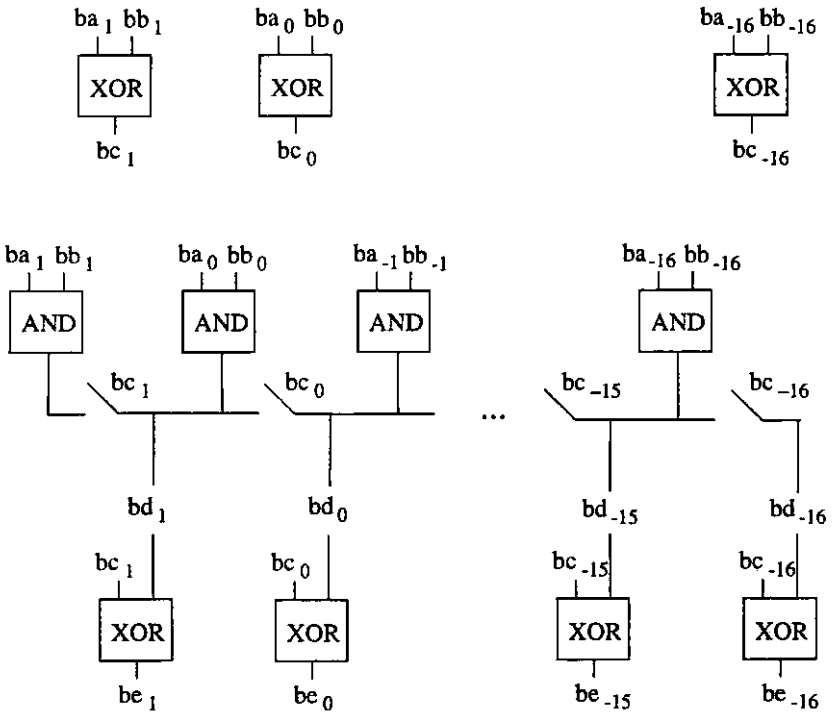

Abbildung 7: Die Schaltung für *carry look-ahead* (entspricht Bild 19 in [16])

Betrachten wir die Addition der Register Ba und Bb. Wir beziehen uns mit bb_i oder Bb[i] auf das i-te Bit vom Register Bb, je nachdem welche Form bequemer ist. Die gleiche Notation wird für die anderen Register benutzt. Zuerst wird ein Zwischenergebnis berechnet, das die bitweise XOR Operation auf beiden Registern darstellt, d.h. $bc_i = (ba_i \text{ XOR } bb_i)$. Ein zweites Zwischenergebnis ist die bitweise AND Operation, angewendet auf beide Register, i.e. ba_i AND bb_i. Die nächste Operation betrifft die Ermittlung der Bitpositionen, für die ein Übertrag benötigt wird. Die Zwischenergebnisse bd_i werden mit Hilfe der in Abbildung 7 gezeigten Schaltung berechnet. Bitte beachten Sie, daß wenn ein Bit 1 ist, die entsprechende Leitung unter Spannung steht, ansonsten ist die Leitung von der Spannungsquelle getrennt (sog. *three-state* Schalter). Nur so kann die Arbeitsweise der Schaltung verstanden werden. Die Ruheposition der Relais $bc_1, \ldots, bc_{-16}$ ist in der Abbildung zu sehen. Falls Bit bc_i gleich 1 ist, wird das entsprechende Relais geschlossen.

Das Endergebnis ist $be_i = bd_i$ XOR bc_i. Es ist beachtlich, wie einfach die Benutzung der Relais die Verteilung des Übertrags bis zur letzten Bitposition macht. Der Übertrag wird beim Weitergeben von einer Position zur nächsten nicht verzögert, weil alle Relais gleichzeitig aktiviert werden.

In der Z4 verwendete Zuse dann eine weiter vereinfachte Relaisschaltung, die pro Stelle mit zwei Relais für Ba[i] und Bb[i] mit je vier Umschaltkontakten auskommt ([16], Bild 20). Die originale Z4 ist im Deutschen Museum in München ausgestellt.

Im unteren werden die Algorithmen beschrieben, die bei Operationen mit Gleitkommazahlen in der Z3 benutzt werden. Sie sind ohne Ausnahme die gleichen wie sie heutzutage in einfachen Gleitkommaprozessoren verwendet werden [6].

Ausnahmebedingungen bei Gleitkommazahlen

Das Problem mit Zuses Notation von Gleitkommazahlen ist, daß spezielle Konventionen für die Darstellung der Zahl Null angewendet werden müssen. Die Z3 löst dieses Problem und behandelt andere Ausnahmebedingungen (Überlauf, Unterlauf) durch die Überwachung des Wertes des Exponenten nach jeder arithmetischen Operation oder nach jedem Lesezugriff auf den Speicher. Eine spezielle Schaltung überwacht den Zustand von Bus Ae und fängt die Ausnahmebedingungen ab. Jede Zahl mit dem Exponenten -64 wird als Null betrachtet: ein Relais (bezeichnet mit Nn_1) wird auf 1 gesetzt, wenn die gelesene Zahl im Registerpaar <Af,Bf> gespeichert wird. Falls aber die gelesene Zahl im Registerpaar <Ab,Bb> gespeichert wird, dann wird das Relais Nn_2 auf 1 gesetzt. Damit wissen wir jederzeit, ob ein oder beide Argumente einer arithmetischen Operation 0 sind. Etwas ähnliches wird für Exponenten mit dem Wert 63 gemacht (eine Zahl mit Wert Unendlich gemäß Zuses Konvention). In diesem Fall wird das Relais Ni_1 oder Ni_2 auf 1 gesetzt, je nachdem in welchem Registerpaar die Zahl gespeichert wird.

Operationen, die eine "Ausnahmezahl" (Null oder Unendlich) beinhalten, werden wie üblich ausgeführt, aber das Ergebnis wird von der Überwachungsschaltung überschrieben. Sei zum Beispiel angenommen, daß eine Multiplikation berechnet wird und daß das erste Argument 0 ist (Nn_1 wurde auf 1 gesetzt). Die Berechnung geht weiter wie üblich, aber in jedem Zyklus liefert die Überwachungsschaltung das Ergebnis -64 am Ausgang des Addierers im Teil A. Es ist unerheblich, welche Operation mit der Mantisse ausgeführt wird, weil der Exponent des Ergebnisses auf -64 gesetzt wird und damit das Endergebnis 0 ist. Die Division durch eine unendliche Zahl wird auf ähnliche Weise abgearbeitet. Die Z3 erkennt nichtdefinierte Operationen wie $0/0$, $\infty - \infty$, ∞/∞ und $0 \times \infty$. In allen diesen Fällen leuchtet ein passend beschriftetes Ausnahmelämpchen an der Ausgabeeinheit auf und die Maschine wird angehalten. Die Z3 produziert immer das richtige Ergebnis, wenn ein Argument 0 oder ∞ ist und das andere Argument innerhalb der numerischen Grenzen liegt[3].

Eine zusätzliche Schaltung überwacht den Exponenten am Ausgang des Exponentenaddierers. Falls der Exponent größer oder gleich 63 ist, dann ist ein Überlauf eingetreten und das Ergebnis muß auf ∞ gesetzt werden. Falls der Exponent kleiner als -64 ist, dann ist ein Unterlauf eingetreten und das Ergebnis muß auf 0 gesetzt werden. Um dies zu erlauben, wird das entsprechende Relais (Nn_1 oder Ni_1) auf 1 gesetzt.

Es ist Zuse gelungen, eine Ausnahmebehandlung zu implementieren, die nur ein paar Relais benutzte. Diese Eigenschaft der Z3 ist wohl eine der elegantesten im ganzen Entwurf der Maschine. Viele der ersten Mikroprozessoren in den siebziger Jahren konnten keine Ausnahmebehandlungen bearbeiten und überließen dies der Software. Zuses Ansatz ist besser, denn er befreit den Programmierer von der Last der Prüfung auf numerische Grenzüberschreitungen vor jeder Operation.

[3] Dies gilt nicht für die Z1. Zuse dachte daran, aber implementierte keine Ausnahmebehandlung in der Z1. Die Maschine konnte nicht richtig arbeiten bei Berechnungen, die 0 enthielten [pers. Mitt. Zuse].

Addition und Subtraktion

Um zwei Gleitkommazahlen x und y zu addieren oder zu subtrahieren, muß ihre Darstellung auf einen gemeinsamen Exponenten gebracht werden. Danach müssen nur noch die Mantissen addiert oder subtrahiert werden. Falls sich die Exponenten unterscheiden, muß die Mantisse der kleineren Zahl um entsprechend viele Stellen nach rechts verschoben werden und ihr Exponent entsprechend erhöht werden (um den Wert der Zahl zu erhalten), bis beide Exponenten gleich sind. Nach 17 Verschiebungen nach rechts kann es natürlich passieren, daß die kleinere Zahl Null wird.

Die Vorzeichen der beiden Zahlen werden verglichen, bevor über die Art der auszuführenden Operation entschieden wird. Falls eine Addition verlangt wurde und die Vorzeichen gleich sind, dann wird die Addition ausgeführt; falls die Vorzeichen verschieden sind, wird eine Subtraktion ausgeführt. Falls eine Subtraktion verlangt wird und die Vorzeichen verschieden sind, dann wird eine Addition durchgeführt; falls die Vorzeichen gleich sind, wird eine Subtraktion durchgeführt. Eine spezielle Schaltung setzt das Vorzeichen für das Endergebnis in Abhängigkeit von den Vorzeichen der Argumente und vom Vorzeichen des partiellen Ergebnisses.

Addition und Subtraktion werden von einer Relaiskette gesteuert (nicht durch ein Schrittschalter), weil die Anzahl der maximal möglichen Zyklen gering ist. Abbildung 8 zeigt die Synchronisation für die Addition zweier Zahlen. Anfänglich sind die Argumente für die Addition in den Registerpaaren <Af,Bf> und <Ab,Bb> gespeichert. Im ersten Zyklus werden die Exponenten subtrahiert. Im zweiten Zyklus wird die Mantisse des größeren Exponenten in das Register Ba geladen und die Mantisse des kleineren Exponenten in das Register Bb. Die Mantisse im Register Bb wird um so viele Stellen nach rechts verschoben, wie es der absoluten Differenz der Exponenten entspricht (die Ausnahmebehandlung beschäftigt sich mit den Fällen, in denen die kleinere Zahl durch die Verschiebung 0 wird). In den Stufen I, II und III des zweiten Zyklus werden die Mantissen addiert und schließlich testet der Prozessor, ob das Ergebnis größer als 2 ist. In diesem Fall wird die Mantisse des Ergebnisses um eine Stelle nach rechts verschoben und der Exponent um 1 erniedrigt. Es ist zu beachten, daß der Test "ist(Be$\geq$2)" im Teil A der arithmetischen Einheit ausgeführt wird, *nachdem* Be bereits im Teil B während der Stufen I, II und III des zweiten Zyklus berechnet wurde.

Im Falle einer Subtraktion sind vier oder fünf Zyklen nötig. Abbildung 9 demonstriert die Synchronisation, die für eine Subtraktion gebraucht wird. 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. Im Endeffekt soll Zyklus 3 lediglich die Mantisse des Ergebnisses positiv machen. Zyklus 4 ist sehr wichtig: die Differenz von zwei normalisierten Mantissen kann einige Nullen auf den höchstwertigen Bitpositionen enthalten. Das Ergebnis wird normalisiert, indem Be entsprechend viele Stellen nach links verschoben wird (dies erfolgt mit dem Schifter zwischen der Relaisschaltung Fd und dem Register Bb). Die Anzahl der bitweisen Verschiebungen wird vom Exponenten im Teil A des Prozessors subtrahiert. Im Zyklus 5 wird das Ergebnis im Registerpaar <Af,Bf> gespeichert.

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:=Bf, Bb:=Be (verschoben) sonst Ba:=Be, Bb:=Bf (verschoben) (Be oder Bf werden |Ae| Stellen nach rechts verschoben)
	I,II,III	wenn (Be≥2) dann Ae:=Aa+Ab+1 sonst Ae:=Aa+Ab	Be:=Ba+Bb
3	IV,V	Af:=Ae	wenn (Be≥2) dann Bf:=Be/2 sonst Bf:=Be

Abbildung 8: Die drei Zyklen, die für eine Addition gebraucht werden. Die Argumente für die Addition werden in den Registerpaaren <Af,Bf> und <Ab,Bb> gespeichert, bevor die Operation ausgeführt wird.

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:= 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

Abbildung 9: Die vier bzw. fünf Zyklen, die für eine Subtraktion gebraucht werden. Die Argumente für die Subtraktion werden in den Registerpaaren <Af,Bf> und <Ab,Bb> gespeichert, bevor die Operation ausgeführt wird.

Multiplikation

Der Multiplikationsalgorithmus der Z3 arbeitet wie für eine dezimale Multiplikation mit Papier und Bleistift. Er besteht aus der wiederholten Addition des Multiplikators entsprechend den einzelnen Ziffern des Multiplikanden. Am Anfang des Algorithmus wird das erste Argument im Registerpaar <Af,Bf> und das zweite Argument im Registerpaar <Ab,Bb> gespeichert. Das temporäre Registerpaar <Aa,Ba> wird auf 0 gesetzt. Abbildung 10 stellt die Mikrosequenzierung durch den Schrittschalter für die Multiplikation dar. Der Algorithmus benötigt 16 Zyklen für einen Durchgang. Anzumerken ist, daß nur die Bits der Stellen -14 bis 0 des Multiplikanden benutzt werden. Die Exponenten werden im ersten Zyklus addiert und das Ergebnis kreist danach im Teil A der arithmetischen Einheit. Die Mantissen werden im Teil B bearbeitet. Register Ba enthält das Teilergebnis der Berechnung. Die Hauptschleife der Multiplikation hat folgende Form:

Ba:=Be/2
Be:=Ba + Bb×(i-tes Bit von Bf)

für $i = -14, \ldots, 0$. Das Teilergebnis Be wird um eine Stelle nach rechts verschoben, um Ba:=Be/2 zu produzieren. Dies erfolgt durch den Schifter, der mit der Relaisschaltung Fc verbunden ist.

Das Ergebnis der Multiplikation ist eine Zahl $1 \leq r < 4$ (für Argumente innerhalb der numerischen Grenzen). Im letzten Zyklus wird geprüft, ob $r \geq 2$. In diesem Fall wird das Ergebnis um eine Stelle nach rechts verschoben und 1 zum Exponenten des Ergebnisses addiert.

Division

Der Divisionsalgorithmus ist analog zum Multiplikationsalgorithmus, nur werden wiederholt Subtraktionen anstelle von Additionen ausgeführt. Am Anfang wird der Dividend im Registerpaar <Af,Bf> und der Divisor im Registerpaar <Ab,Bb> gespeichert. Das temporäre Registerpaar <Aa,Ba> wird auf 0 gesetzt. Abbildung 11 stellt die Mikrosequenzierung durch den Divisionsschrittschalter dar. Der Algorithmus braucht 18 Zyklen.

Die Grundidee des Algorithmus ist einfach. Der Exponent des Ergebnisses wird durch Subtraktion der Exponenten von Dividend und Divisor berechnet. Und nun zur Mantisse: wir möchten x/y für die normalisierten Mantissen x und y berechnen. Da wir mit normalisierten Zahlen arbeiten, ist die erste Ziffer des Ergebnisses gleich 1, wenn $x \geq y$ ist und gleich 0, wenn $x < y$ ist. Im ersten Fall setzen wir die Ziffer des Ergebnisses auf 1 und berechnen den Rest, der $x - y$ ist. Der Rest wird wiederum durch y geteilt. Dafür wird der Rest um eine Stelle nach links verschoben und das neue Ergebnisbit wird an der Stelle [−1] im Register Bf gespeichert (was damit den Effekt des Verschiebens wieder aufhebt). Falls das Ergebnisbit 0 ist, ist der Rest nur x und die Division geht weiter wie im ersten Fall.

Die Hauptschleife der Division hat die folgende Form:

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

für $i = 0, \ldots, -14$. Das Teilergebnis Be wird um eine Stelle nach links verschoben, um Ba:=2×Be zu produzieren. Dies erfolgt über den Schifter, der mit der Relaisschaltung Fc verbunden ist.

Das Ergebnis der Division der Mantissen ist eine Zahl $1/2 < r < 2$. Diese Bedingung wird in den Zyklen 17 und 18 getestet. Falls $r < 1$, dann wird 1 vom Exponenten subtrahiert und das Ergebnis um eine Stelle nach links verschoben, um wiederum eine normalisierte Zahl zu erhalten.

Quadratwurzelberechnung

Der Algorithmus für die Berechnung der Quadratwurzel ist das Glanzstück der Z3. Abbildung 12 zeigt die Mikrosequenzierung des entsprechenden Befehls, der 20 Zyklen benötigt. Das Argument für die Operation wird im Registerpaar <Af,Bf> gespeichert. Das Registerpaar <Aa,Ba> wird mit 0 initialisiert. Der Algorithmus berechnet die Quadratwurzel von Zahlen mit einem geraden Exponenten. Falls der Exponent ungerade ist, dann wird die Mantisse um eine Stelle nach links verschoben und der Exponent um 1 erhöht. Der Exponent des Ergebnisses (berechnet in Zyklus 19) ist halb so groß wie der Anfangsexponent.

Die Grundidee des Algorithmus ist es, die Quadratwurzelberechnung auf eine Division zu reduzieren[4]. Wenn wir die Wurzel von x berechnen wollen, dann suchen wir eine Zahl derart, daß $x/Q = Q$ ist. Das Ergebnis Q wird erhalten, indem nacheinander das i-te Bit auf 1 gesetzt wird und dann geprüft wird, ob die Bedingung $x > Q^2$ noch gilt. Falls dies nicht der Fall ist, dann muß das i-te Bit auf 0 gesetzt werden.

Nehmen wir an, wir haben bereits Bit 0 bis Bit $-i+1$ des Endergebnisses berechnet. Die Mantisse sei bezeichnet mit Q_{-i+1}.

$$Q_{-i+1} = \mathrm{Bf}[0] \times 2^0 + \mathrm{Bf}[-1] \times 2^{-1} + \cdots + \mathrm{Bf}[-i+1]2^{-i+1}.$$

Bit $-i$ wird dann auf q_{-i} gesetzt und es muß gelten, daß

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

Dies ist erfüllt, wenn

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

Definieren wir t_{-i} durch den Ausdruck

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

Dies kann geschrieben werden als

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

wobei wir die rekursive Definition $2^{-i+1}t_{-i+1} = (x - Q^2_{-i+1})$ benutzt haben. Vereinfachen wir den letzten Ausdruck, dann erhalten wir schließlich:

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

Falls t_{-i} positiv ist für $q_{-i} = 1$, dann setzen wir Bit $-i$ des Endergebnisses auf 1, d.h. Bf[$-i$]:=1. Falls t_{-i} negativ ist, dann setzen wir Bf[$-i$]:=0. Die rekursive Berechnung wird mit $t_0 = x$ gestartet. Q_{-i+1} stellt in jedem Schritt das Teilergebnis im Register Bf dar. Bit $-i$ wird versuchsweise auf 1 gesetzt und das Vorzeichen von t_{-i} wird geprüft.

Die Hauptschleife für den Algorithmus für die Quadratwurzel hat deswegen 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
```

Alle Bits des Registers Bf werden für die Berechnung der Quadratwurzel benutzt. Falls die Ausgangszahl innerhalb der numerischen Grenzen liegt, ist auch das Ergebnis innerhalb der Grenzen.

[4]Diese Idee liegt auch der früher im Unterricht des Gymnasiums gelehrten Methode für die Quadratwurzelberechnung im Dezimalsystem zugrunde

Ein- und Ausgabebefehle

Die zwei kompliziertesten Befehle der Z3 gibt es in Verbindung mit der Ein- und Ausgabe von Dezimalzahlen. Eine Dezimalzahl mit vier Ziffern, die über die Tastatur einzugegeben ist, wird zuerst in eine binäre ganze Zahl umgewandelt. Dies erfolgt durch aufeinanderfolgendes einlesen einer einzelnen Ziffer, umwandeln dieser in eine Binärzahl und speichern in den Bits Ba[−10], Ba[−11], Ba[−12], und Ba[−13] des Registers Ba. Die Zahl im Register Ba wird mit 10 multipliziert und die Prozedur wird wiederholt für alle weiteren Ziffern. Nach vier Durchgängen ist die dezimale Eingabe in eine binäre Zahl umgewandelt worden. Die Schwierigkeiten entstehen für den Exponenten. Falls der Exponent e positiv ist, dann muß die Mantisse e mal mit 10 multipliziert werden. Falls der Exponent negativ ist, dann muß die Mantisse $|e|$ mal mit 0.1 multipliziert werden. Die Multiplikation mit 10 ist recht einfach: die Mantisse kann in Be um ein Bit verschoben werden und dann in Ba gespeichert werden (d.h. Ba:=2×Be). Zur gleichen Zeit kann Be um drei Stellen nach links verschoben werden und in Bb gespeichert werden (d.h. Bb:=8×Be). Die Addition von Ba und Bb ergibt dann das gewünschte Ergebnis: die Multiplikation der Ursprungszahl in Be mit 10. Diese Prozedur braucht 4 Zyklen für eine Multiplikation, das wären zum Beispiel 32 Zyklen für den dezimalen Exponenten +8. Dies heißt, daß eine Dezimalzahl mit dem Exponenten +8 in 41 Zyklen eingelesen wird, weil eine Leseoperation mindestens neun Zyklen braucht.

Im Fall eines negativen Exponenten wird die Multiplikation mit der Konstanten 0.1 mit Hilfe der Schifter und der Addierer geleistet. Diese Multiplikation ist etwas komplexer, weil 0.1 im Binärsystem eine nicht triviale periodische Zahl ist. Die Beschreibung dieser Mikrosequenz würde uns zu weit vom Thema abbringen, deswegen wird sie ausgelassen.

Die Ausgabe arbeitet mit der wiederholten Multiplikation oder Division durch 10. Falls der binäre Exponent der Zahl im Register R1 positiv ist, dann wird die Zahl so lange mit 0.1 multipliziert, bis der binäre Exponent gleich 2 ist und bis die ersten vier Bit im Register Bf eine Zahl zwischen 0 und 9 (0000 und 1001) enthalten. Dies ist die dezimale Ziffer, die in der nächsten Spalte von den Ausgabelampen angezeigt wird. Die Ziffer wird von der Mantisse subtrahiert und die Prozedur setzt sich für die folgenden Ziffern fort. Falls der binäre Exponent der Zahl im Register R1 negativ ist, dann ist die Prozedur genau dieselbe, nur das Multiplizieren erfolgt mit der Konstanten 10.

5 Die vollständige Architektur der Z3

Wir sind nun in der Lage, das Detaildiagramm der Z3 in Abbildung 13 besser zu verstehen. Wir sehen einige Komponenten, die in den vorangehenden Abschnitten besprochen wurden.

Die Steuereinheit und die Ein- bzw. Ausgabeeinheiten wurden bereits besprochen. Die vier Dezimalziffern der Eingabetastatur werden mit Hilfe der Relaisschaltungen Za, Zb, Zc und Zd übertragen. Diese Relaisschaltungen werden eine nach der anderen aktiviert.

Die Relaisschaltungen Eg und Ei werden benutzt, um zwei nützliche Konstanten (+13 und −4) direkt in das Exponentenregister zu bringen. Der Schifter Ee zwischen den Registern Af und Aa wird für die Quadratwurzelberechnung benutzt. Der Exponent vom Ergebnis (Aa) ist dann halb so groß wie der Exponent der Ausgangszahl (Af).

Ah_1 ist ein Relais, das als Flip-Flop arbeitet. Wenn es auf 0 gesetzt ist, dann wird das Registerpaar <Af,Bf> für eine Leseoperation zugänglich. Wenn es auf 1 gesetzt ist, dann wird auf das Registerpaar <Ab,Bb> zugegriffen. Dieses Relais wird über die Signalleitung a_i auf 0 gesetzt. Die Signalleitungen a_l, a_j, b_l, und b_j werden gebraucht, um bei Bedarf die Register Af, Ab, Bf und Bb zu löschen.

Der mit "Null, Unendlich" bezeichnete Kasten unterhalb von Ae stellt die Schaltung für die Ausnahmebehandlung dar. Sie überwacht ständig den Datenbus (hinsichtlich der Ergebnisse von Operationen und des Datentransfers vom Speicher) und setzt ggf. die entsprechende Ausnahmebedingung. Der Schifter unter Be wird benutzt, um die Mantisse um eine Stelle nach rechts zu verschieben. Dies erbringt die nötige Normalisierung der Mantisse jedesmal, wenn Be≥2.

Fp und Fq sind Relais, die Anzahl und Richtung der Verschiebungen im Schifter unter den

Relaisschaltungen Fc und Fa kontrollieren. Fh, Fi, Fk, Fl und Fm haben die gleiche Funktion im Zusammenhang mit den anderen Schiftern. Mit diesen fünf Bits können die Zahlen zwischen -16 und 15 dargestellt werden und dies entspricht der Anzahl und Richtung der möglichen Verschiebungen des zweiten Schifters. Jedesmal, wenn eine derartige Verschiebung ausgeführt wird, wird die Zahl, die die Relais Fh bis Fm darstellen, durch die Relaisschaltung Bn in das Register Ab übertragen. Dies erfolgt, um den Exponenten des Ergebnisses entsprechend zu ändern. Falls die Zahl um 10 Stellen nach links verschoben wird, dann wird $+10$ vom Exponenten des Ergebnisses subtrahiert. Solche großen Verschiebungen werden häufig nach Subtraktionen benötigt.

Schauen wir uns noch einmal das Diagramm für die Z3 an. Alles macht nun einen Sinn und sieht ebenso konventionell aus wie in jedem einfachen modernen Gleitkommaprozessor. Es ist in der Tat erstaunlich, wie Konrad Zuse in der Lage war, eine sehr effiziente Architektur gleich zu Beginn zu finden. Der Prozessor der Z3 enthält bloß 600 Relais, der Speicher braucht dreimal soviele. Zuse war *gezwungen*, immer wieder die logische Struktur seiner Maschine zu überdenken, weil der Entwurf dahingehend optimiert werden mußte, soweit als möglich Hardware zu sparen. Er hatte nicht den Luxus der fast unbegrenzten Unterstützung, die vom amerikanischem Militär für die Entwicklung des ENIAC oder von IBM für die Mark I bereitgestellt wurden. Er war ganz auf sich allein gestellt, und obwohl dies sicher zu seinem Vorteil auf der konzeptionellen Seite wirkte, es gelangte zu seinem Nachteil, wenn der geringe Einfluß der Z1 und der Z3 auf die aufblühende amerikanische Computerindustrie nach dem Weltkrieg in Betracht gezogen wird [14].

6 Die Erfindung des Computers

Das größte Manko der Z3 war das Fehlen von Befehlen für bedingte Sprünge im Befehlssatz. Es wäre nicht unmöglich gewesen, dies zu implementieren: obwohl es bei einem auf einem Lochstreifen gespeicherten Programm schwerfällig ist, wären nur eine paar zusätzliche Schaltkreise nötig gewesen. Manchmal wird die Trennlinie zwischen einer bloßen Rechenmaschine und einem universellen Computer mit der Unterscheidung nach intern oder extern gespeichertem Programm gezogen. Ich habe an anderer Stelle argumentiert [10], daß dies kein verbindliches Kriterium ist. Ein externes Programm kann wie ein Interpreter numerischer Daten arbeiten. Das externe Programm wird fester Bestandteil des Prozessors und die Daten werden selbst zum Programm, ganz ähnlich wie eine universelle Turing-Maschine als Interpreter arbeitet. Ich habe argumentiert [11], daß universelle Maschinen einen minimalen Befehlssatz und indirekte Adressierung benötigen. Indirekte Adressierung kann durch ein sich selbst modifizierendes Programm simuliert werden, sodaß der Befehlssatz das entscheidende Kriterium ist. Eine Maschine mit ausreichendem Speicher, der sowohl Daten wie Befehle faßt, einem Addierer und fähig zur Ausführung der Befehle CLR (löschen), INC (inkrementieren), LOAD (lesen), STORE (speichern) und BR (springen falls Null) ist eine universelle Maschine (dieser Befehlssatz kann weiter reduziert werden [12]). In diesem Sinne war die Z1 keine universelle Maschine, aber ebenso war es keine der anderen frühen Rechenmaschinen. Der ABC war eine spezielle Maschine für die Gaußelimination und der Mark I aus Harvard fehlte der bedingte Sprungbefehl, obwohl sie sozusagen FOR-Schleifen ausführen konnte. Der ENIAC war noch nicht einmal über Software programmierbar: die Bausteine mußten nach Datenflußart fest verbunden werden. Bedingte Sprünge waren in der ENIAC nur begrenzt verfügbar und sich selbst modifizierende Programme standen nicht zur Diskussion.

Tabelle 2 und 3 fassen die wichtigsten Informationen für die frühen Rechner zusammen, die im Abschnitt 1 erwähnt wurden. Wie aus diesen Tabellen deutlich abzulesen ist, erfüllt keine Maschine alle Anforderungen für einen Universalrechner. Die in Manchester zwischen 1946 und 1948 gebaute 'Baby' Mark 1 wurde in die Tabelle aufgenommen, weil sie (soweit es dem Autor bekannt ist) die erste Maschine war, die die Definition eines universellen Computers erfüllt. Die 'Baby' Mark 1 wie auch die spätere Ferranti MARK I, wurde unter der Leitung von F. C. Williams und T. Kilburn gebaut. Die Programme wurden in einen digitalen Speicher mit wahlfreiem Zugriff geladen, der mit Kathodenstrahlröhren implementiert wurde. Alle notwendigen primitiven Befehle waren vorhanden (in veränderter Form) und obwohl es keine indirekte Adressierung gab, konnten sich selbst modifizierende Programme geschrieben werden. Das erste Programm lief am 21. Juni 1948

Tabelle 2: Vergleich der architektonischen Eigenschaften

Maschine	Speicher und CPU getrennt?	Bedingte Sprünge?	Soft- oder Hardware-programmierung	Sich selbst modifizierende Programme?	Indirekte Adressierung?
Zuses Z1	√	×	Software	×	×
Atanasoff	√	×	Hardware	×	×
Harvard Mark I	×	×	Software	×	×
ENIAC	×	teilweise	Hardware	×	×
Manchester Mark 1	√	√	Software	√	×

Tabelle 3: Einige zusätzliche Eigenschaften

Maschine	Interne Kodierung	Fest- oder Gleitkomma?	Bit-Sequentielle Arithmetik?	Architektur	Technologie
Zuses Z1	Binär	Gleitkomma	nein	sequentiell	Mechanisch
Atanasoff	Binär	Festkomma	ja	vektoriell	Elektronisch
Harvard Mark I	Dezimal	Festkomma	nein	parallel	Elektromechanisch
ENIAC	Dezimal	Festkomma	nein	Datenfluß	Elektronisch
Manchester Mark 1	Binär	Festkomma	ja	sequentiell	Elektronisch

und berechnete den größten Faktor von 2^{18} mit 3.5 Millionen Operationen, wozu es 52 Minuten brauchte [8]. Im September wurde Alan Turing zum Dozenten für Mathematik in Manchester berufen und schrieb einige Programme für den ersten vollwertigen Computer der Welt. Seine Vision eines Universalrechners, die 1936 publiziert wurde, dem selben Jahr, in dem die Speichereinheit der Z1 fertig wurde, wurde mit der Mark 1 verwirklicht. Die Tabellen 2 und 3 zeigen deutlich, daß die Erfindung des Computers eine kollektive Errungenschaft war, die zwei Kontinente und 12 Jahre umfaßte.

Danksagung

Es war mir nur durch die Zusammenarbeit mit einigen meiner Studenten an den Universitäten in Halle und Berlin möglich, die skizzenhafte Dokumentation der Z3 zu verstehen. Ich danke Alexander Thum und Axel Bauer, die eine Gatterebenensimulation des Prozessors der Z3 implementiert haben. Uns wurden die Synchronisationsprobleme bewußt, nachdem die Simulation nicht in Gang kommen wollte. Ich bedanke mich auch bei Frank Kosnieczny, Reimund Spitzer und Roland Schultes, die einen Teil einer eigenständigen Simulation des Prozessors in C schrieben. Axel Schalt hat den Originalartikel vom Englischen ins Deutsche übersetzt. Prof. Friedrich L. Bauer hat wertvolle Hinweise für die Überarbeitung des Manuskripts gegeben. Meine Studenten und ich fingen die Arbeit an der Z3 mit der Hilfe von Konrad Zuse an, der unsere Fragen gerne beantwortete. Es war erstaunlich zu beobachten, wie er nach fast 60 Jahren den kompletten Entwurf der Z3 noch immer im Gedächtnis hatte. Leider starb Konrad Zuse im Dezember 1995, bevor dieser Artikel über sein Werk fertig wurde. Diese Arbeit ist seinem Andenken gewidmet.

Literatur

[1] H. Aiken und G. Hopper, "The Automatic Sequence Controlled Calculator", nachgedruckt in [9], S. 203-222.

[2] A. W. Burks und A. R. Burks, "The ENIAC: First General-Purpose Electronic Computer", *Annals of the History of Computing*, Vol. 3, N. 4, 1981, S. 310-399.

[3] A. W. Burks und A. R. Burks, *The First Electronic Computer – The Atanasoff Story*, The University of Michigan Press, Ann Arbor, 1988.

[4] K.-H. Czauderna, *Konrad Zuse, der Weg zu seinem Computer Z3*, Oldenbourg Verlag, München, 1979.

[5] D. Knuth, *The Art of Computer Programming – Seminumerical Algorithms*, Vol. 2, 1981.

[6] I. Koren, *Computer Arithmetic Algorithms*, Prentice Hall, Englewood Cliffs, NJ, 1993.

[7] S.H. Lavington, *A history of Manchester computers*, NCC Publications, Manchester, 1975.

[8] S.H. Lavington, *Early British Computers*, Digital Press, Manchester, 1980.

[9] B. Randell, *The Origins of Digital Computers*, Springer-Verlag, Berlin, 1982.

[10] R. Rojas, "Who invented the computer? The debate from the viewpoint of computer architecture", in W. Gautschi (ed.), *Fifty Years Mathematics of Computation*, Proceedings of Symposia in Applied Mathematics, AMS, 1993, S. 361–366.

[11] R. Rojas, "On Basic Concepts of Early Computers in Relation to Contemporary Computer Architectures", *Proceedings of the 13th World Computer Congress*, Hamburg, Vol. II, S. 324–331.

[12] R. Rojas, "Conditional Branching ist not Necessary for Universal Computation in von Neumann Computers", erscheint in *Journal of Universal Computer Science.*

[13] U. Schweier und D. Saupe, "Funktions-und Konstruktionsprinzipien der programmgesteuerten mechanischen Rechenmaschine Z1", Arbeitspapiere der GMD 321, Bonn, 1988.

[14] N. Stern, *From ENIAC to UNIVAC*, Digital Press, Bedford, 1981.

[15] K. Zuse, *Patentanmeldung Z-391*, Deutsches Patentamt , Berlin, 1941.

[16] K. Zuse, *Der Computer mein Lebenswerk*, Verlag Moderne Industrie, München, 1970.

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

Abbildung 10: Die 16 Zyklen für die Multiplikation. Das i-te Bit des Registers Bf wird mit Bf[i] bezeichnet. Der Multiplikator wird im Registerpaar <Af,Bf> gespeichert und der Multiplikand in <Ab,Bb>, bevor die Operation durchgeführt wird.

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	Ba:=Bf, Bb:=0
	I,II,III	Ae:=Aa+Ab	Be:=Ba+Bb
18	IV,V	Af:=Ae	wenn (Bf[0]=0) dann Bf:=2×Be sonst Bf:=Be

Abbildung 11: Die 18 Zyklen für die Division. Das i-te Bit des Registers Bf wird mit Bf[i] bezeichnet. Der Dividend wird im Registerpaar <Af,Bf> gespeichert und der Divisor in <Ab,Bb>, bevor die Operation durchgeführt wird.

Zyklus	Stufe	Exponent	Mantisse
0	I,II,III		
1	IV,V		If (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 Ba:=2×Be, Bb:=2×Bf, Bb[−1]:=1
	I,II,III		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

Abbildung 12: Die 20 Zyklen für die Berechnung der Quadratwurzel. Das i-te Bit der Register Af und Bf wird mit Af[i] bzw. Bf[i] bezeichnet. Das Argument wird im Registerpaar <Af,Bf> gespeichert, bevor die Operation ausgeführt wird.

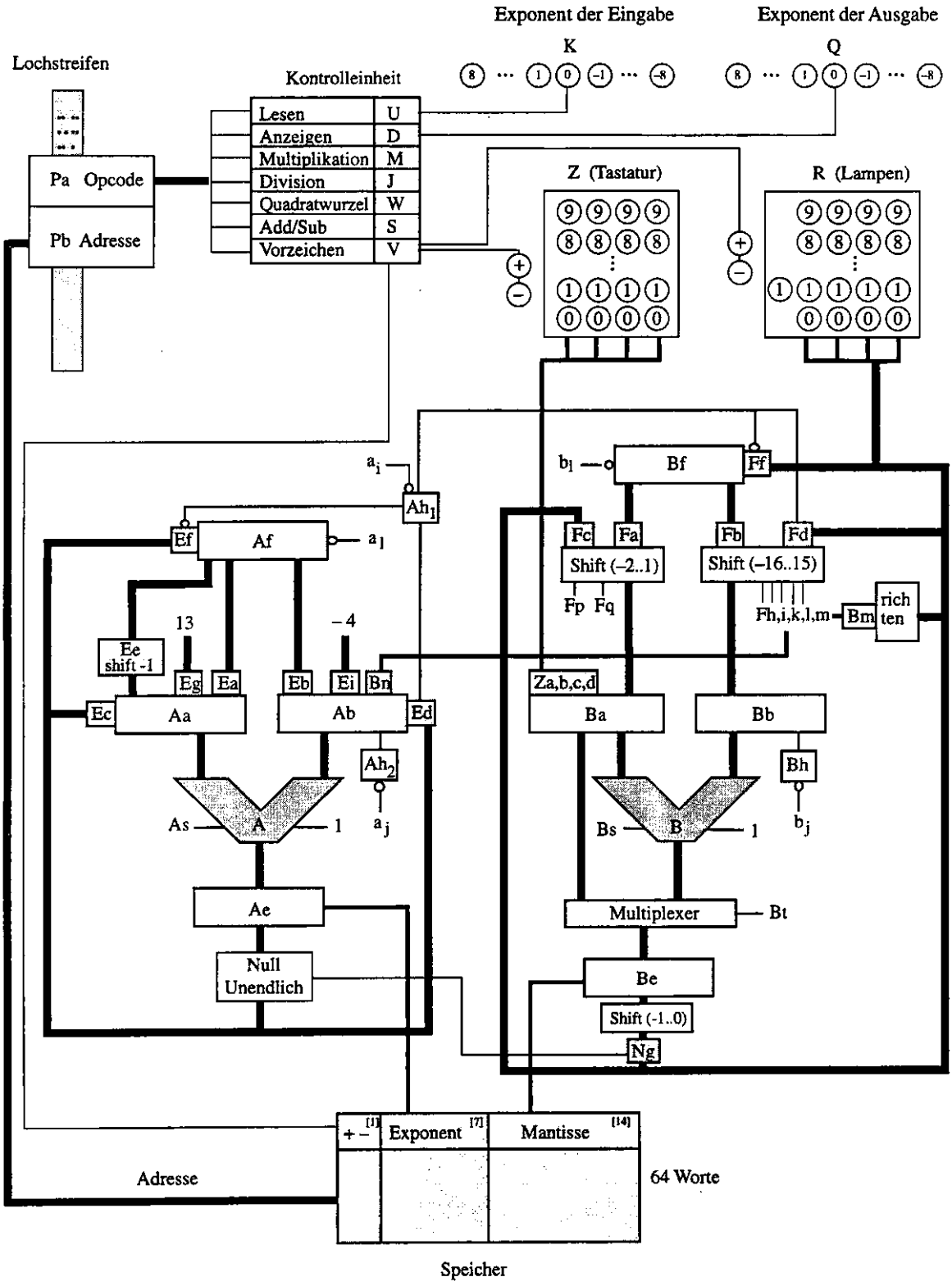

Abbildung 13: Die vollständige Architektur der Z3