АРХИТЕКТУРА ВЫЧИСЛИТЕЛЬНЫХ МАШИН КОНРАДА ЦУЗЕ Z1 И Z3

Рауль Рохас*

Данная статья предлагает первое подробное описание архитектуры вычислительных машин Z1 и Z3, спроектированных Конрадом Цузе в период с 1936 по 1941 гг. Необходимая информация взята из патента Цузе, зарегистрированного в 1941 году, содержащего исчерпывающие пояснения. Дополнительные сведения почерпнуты из программной имитации машинной логики. Z1 была построена исключительно из механических деталей, в Z3 использованы электромагнитные реле. Однако обе машины имели общую логическую структуру, и идентичную модель программирования. Ниже будет показано, что Z1 и Z3 обладали свойствами, типичными для современных компьютеров: память и процессор были представлены отдельными компонентами, процессор мог обрабатывать числа с плавающей запятой и выполнять четыре основные арифметические операции, а также вычислять квадратный корень. Программа записывалась на перфоленте и последовательно считывалась. В заключительной части данной статьи архитектура Z1 и Z3 будет рассмотрена в историческом контексте, в сравнении с другими машинами.

1. Первые вычислительные машины

Конрад Цузе признан в Германии „Отцом компьютеров", а его Z1, программируемый автомат, построенный в 1936-1938 годах, назван „первым компьютером" в мире. Другие страны присваивают это звание одному из своих ученых, и уже долго длятся дебаты об „истином" изобретателе компьютера. Иногда дискуссия предворяется подробным описанием технических черт конкретной машины. ENIAC (*Electronic Numerical Integrator and Computer)*, например, был назван первым в мире *крупномасштабным электронным компьютером широкого назначения* [2]. Эта машина была построена в *Moore School of Electrical Engineering* в Университете Пенсильвании с мая 1943 по 1945. ENIAC разрешил свою первую задчу в декабре 1945 и был официально представлен в феврале 1946.

Еще один претендент на звание первого компьютера – МаркI (MarkI), построенный Хорвардом Айкен в Гарвардском университете в период с 1939 по 1944. МаркI был электро-механической машиной, т.е. он был не полностью механическим как более ранние вычислительные машины, но и содержал доступные в то время электронные компоненты [1]. В машине Джона Атанасова (позже названной ABC), построенной в 1938-1942 гг. в *Iowa State College,* использовались вакуумные трубки. Она могла только складывать и вычитать вектора и имела структуру не подходящую для общих вычислений [3]. В противоположность этим трем машинам Z1 и

* Рауль Рохас
Профессор Берлинского Университета (Freie Universität Berlin),
факультет математики и информатики

Z3 были намного более гибкими; они могли выполнять длинные последовательности команд, записаных на перфоленте. Машины Цузе были механическими или электро-механическими и имели небольшие размеры. Так как Z1 была изготовлена раньше, чем МаркI, то она названа первой *программируемой* вычислительной машиной. Конечно, эта статья не прекратит старый спор, но в следующих главах будет показано насколько прогрессивными были машины Цузе с точки зрения современной компьютерной архитектуры и по сравнению с другими разработками того времени.

В 30-х годах, еще будучи студентом, начал Конрад Цузе думать о вычислительных машинах. Он считал, что он мог построить автомат, способный выполнять арифметические операции, необходимые для инженерных вычислений. Как инженер-строитель он не имел специального образования в электротехнике или электронике и не был знаком с техникой, обычно используемой тогда в вычислительных машинах. Однако этот дефицит знаний стал его преимуществом, так как он вынужден был обдумать проблему арифметических вычислений заново и пришел к новым решениям.

Цузе решил построить экспериментальную вычислительную машину, которая реализовывала бы две основные идеи: a) машина должна была работать с двоичными числами; b) блок управления и вычисления должен был быть отдельным от запоминающего устройства. За годы до того, как Джон фон Нойман обосновал преимущества компьютерной архитектуры, в которой процессор и память разделены, Цузе пришел к той же идее. Нужно отметить, что еще Чарльз Бэббидж в прошлом столетии при разработке *Аналитической Машины* использовал тот же принцип. В 1936 году была изготовлена память[1] для запланированной Цузе машины. Это был механический прибор, но нетипичный для того времени. Вместо шестерней (использованных Бэббиджем в прошлом столетии) Цузе реализовал логические и арифметические операции с помощью подвижных металлических планок. Планки могли двигаться только в двух направлениях (вперед и назад) и были потому достаточны для двоичной машины [16]. Процессор машины, впоследствии названой Z1, был изготовлен на несколько месяцев позже памяти с использованием тех же технических компонентов. Он работал с памятью но не был очень надежен. Основной проблемой была точная синхронизация, необходимая для исключения черезмерной механической нагрузки на подвижные части. Интересно отметить, что в том же году, когда была изготовлена память для Z1, Алан Тьюринг написал свою статью об исчислимых числах, в которой он формализовал интуитивное понятие об исчислимости.
Хотя Z1 и была ненадежна, она показала, что проект был стоящим и побудил Цузе к дальнейшим исследованиям возможных реализаций. Он решил использовать электромеханические реле, так как до и во время второй мировой войны их было легче получить и они были дешевле, чем другие компоненты. Пробная модель (названная впоследствии Z2) имела процессор, построенный на реле, и механическую память Z1. Вскоре после этого Цузе начал строить Z3, машину, основанную исключительно на реле, но имеющую такую же логическую структуру, как и Z1. Z3 была готова и работала уже в 1941 году, на четыре года раньше ENIAC.

Эта статья предлагает первое подробное представление общей архитектуры Z1 и Z3. Цузе сам реконструировал Z1 в 80-х годах в Берлине. Сейчас Z1 – один из выставочных экспонатов в Берлинском музее транспорта и техники, однако имеющаяся там информация описывает только разработку механической памяти [13]. Z3 была

[1] Цузе назвал его „Speicherwerk“ (устройство памяти). Слово „Speicher“ (память) используется в немецком вместо английского „memory“

документирована Цузе в патенте Z-391 1941 года, однако эту документацию трудно читать из-за нестандартной формы записи и терминологии [15]. Книга Чаудерна о Z3 – хороший источник для понимания исторической среды, в которой появились изобретения Цузе, но не содержит подробного описания Z3 [4]. Ниже будет идти речь только о Z3, т.к. Z1 и Z3 с точки зрения логики практически эквивалентны. Главное различие в архитектуре Z1 и Z3 состоит в том, что Z1 не вычислял квадратный корень. Кроме того имелось незначительное отличие в количестве бит, используемых процессором для выполнения арифметических операций (в Z1 мантиса чисел с плавающей запятой была на 1 бит меньше) и в числе циклов, необходимых для каждой команды. С этими небольшими оговорками и принимая во внимание только архитектурные свойства, можно говорить о Z1 и Z3 как об эквивалентных машинах.

2. Обзор архитектуры Z1 и Z3

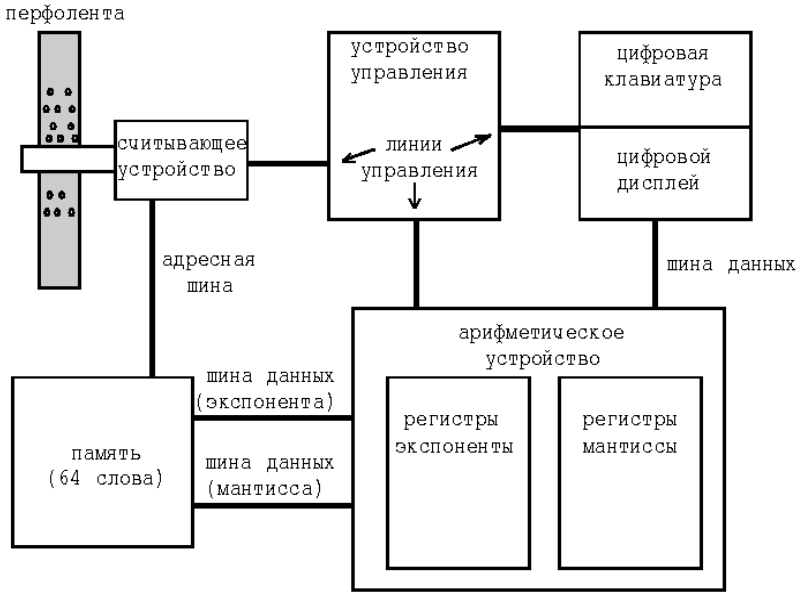
В этом разделе описываются важнейшие свойства архитектуры Z3. Описание ведется от простого к сложному: вначале предлагается обзор, а в третьем разделе обсуждаются детали. Для простоты будем говорить о Z3 в настоящем времни.

2.1. Блоковая структура

Z3 – машина, работающая с числами с плавающей запятой. В то время как другие вычислительные машины (МаркI, ABC, ENIAC) работали с числами с фиксированной запятой, Цузе уже решил применять то, что он называл „полулогарифмическим" представлением, и что соответствует современному представлению чисел с плавающей запятой.

Рисунок 1 схематично показывает структуру Z3. Первое значимое свойство - разделение процессора и памяти. Z3 состоит из двоичного запоминающего устройства (с емкостью 64 числа с плавающей запятой), двоичного процессора, элемента управления и устройства ввода/вывода. Память и арифметический элемент связаны шиной данных, которая передает экспоненту и мантиссу числа. Блок управления содержит контур или схему для каждой команды. Сигнальные линии, которые идут от элемента управления к запоминающему устройству, процессору и устройствам ввода/вывода, отвечают за правильную синхронизацию всех приборов. Прибор считывания с перфоленты поставляет коды команд для каждой команды, а также адрес для доступа в память. Устройства ввода/вывода связаны шиной данных с процессором.

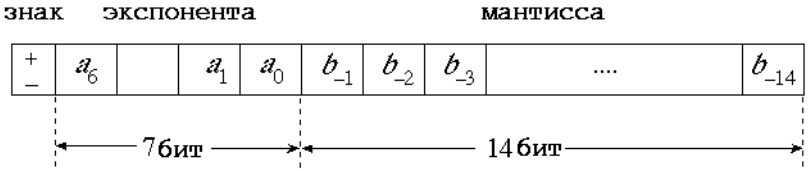
Рис.1 Основные элементы Z3



2.2. Представление чисел с плавающей запятой

Рисунок 2 показывает представление чисел с плавающей запятой в памяти Z3. В первом бите сохраняется знак числа, в следующих 7-ми битах – экспонента, и в последних 14-ти битах - мантисса числа (запоминаются только 14 знаков после запятой). Биты, в которых хранится экспонента числа, названы „часть A" числа и обозначены $a_6,...a_0$. Биты мантиссы названы „часть B" числа и обозначены $b_0,b_{-1},...b_{-14}$. Экспонента – степень числа 2. В памяти соответственно могут храниться числа от 2^{-64} до 2^{63} (по модулю). Мантисса записывается в „нормализованной" форме, т.е. цифра перед запятой (b_0) всегда 1, поэтому ее не нужно запоминать (и поэтому она не обозначена на рис.2). Таким образом фактически в памяти может храниться число с 15-ти битной мантиссой. Проблемой является число 0, т.к. оно не может быть представлено в виде числа с нормализованной мантиссой. В Z3 операции с числом 0 или с бесконечностью обрабатываются как исключения. Любое число с экспонентой -64 интерпритируется как 0, число с экспонентой 63 – как бесконечность. Машина проверяет каждое прочитанное число и, если необходимо, устанавливает бит исключения.

Рис.2 Представление чисел с плавающей запятой в памяти



При этих соглашениях наименьшее представимое в памяти Z3 число это $2^{-63} \approx 1{,}08 \times 10^{-19}$, а самое большое $1{,}999 \times 2^{62} \approx 9{,}2 \times 10^{18}$. Операнды могут быть введены с клавиатуры Z3 как четырехзначные десятичные числа. Экспонента десятичного числа вводится нажатием соответствующей

клавиши из ряда, помеченного -8,-7,...,7,8. Первая версия Z3 могла принимать числа от 1×10^{-8} до 9999×10^{8}. Реконструированная Z3, построенная Цузе в 1962 году для Немецкого Музея, предлагает достаточно клавиш и для бóльших экспонент. При таких соглашениях полная числовая емкость машины может быть представлена широтой допустимых значений ее ввода. То же самое можно сказать и о выводе. Однако, Z3 не может печатать числовой результат, вычисленный программой. Вместо этого одиночные числа высвечиваются матрицей из ламп, которая может отображать цифры от 0 до 9. Наибольшее представимое число – это 19999, а наименьшее 00001. Наибольшая представимая экспонента – это +8, наименьшая -8. Машина в Мюнхене может выводить экспоненты в более широком диапозоне значений.

2.3. Набор команд

Программа для Z3 записана на перфоленте. Восемь бит в каждой строке кодируют одну команду. Набор команд для Z3 состоит из девяти приведенных в таблице 1 команд. Они делятся на три класса: команды ввода/вывода, команды работы с памятью и арифметические команды. Код команды имеет длину от 2 до 5 бит. Операции с памятью содержат в нижних шести битах адрес „слова" в памяти, т.е. в адресном пространстве, которое как уже было отмечено, имеет максимальный размер 64 слова.

Команды можно комбинировать в любом порядке. Команды Lu и Ld (чтение с клавиатуры, выдача результата) приостанавливают машину, чтобы пользователь успел ввести число или считать результат. После этого машина продолжает выполнение программы.

Таблица 1 Набор команд для Z3

Класс	Команда	Описание	Код команды
Ввод/вывод	Lu	Чтение с клавиатуры	01 110000
	Ld	Выдача результата	01 111000
Память	Pr *z*	Чтение из памяти с адресом z	11 *Z6Z5Z4Z3Z2Z1*
	Ps *z*	Запись в память по адресу z	10 *Z6Z5Z4Z3Z2Z1*
арифметические команды	Lm	Умножение	01 001000
	Li	Деление	01 010000
	Lw	Вычисление квадратного корня	01 011000
	Ls1	Сложение	01 100000
	Ls2	Вычитание	01 101000

Очевидный недостаток в наборе команд Z3 – это отсутствие условного разветвления. Цикл может быть реализован путем соединения концов перфоленты, но возможность реализовать условный переход отсутствует. По этой причине Z3 не является универсальной вычислительной машиной по Тюрингу.

2.4. Число циклов

Z3 – это тактовая машина. Каждый такт (в терминах Цузе – одна игра) – разделяется на пять нумерованых римскими цифрами ступеней: I, II, III, IV и V. Текущая команда, считываемая с перфоленты, декодируется первой ступенью цикла.

Две основные арифметические операции Z3 – это сложение и вычитание экспоненты и мантиссы. Эти операции могут быть выполнены на первых трех ступенях каждого цикла. Ступени IV и V используются

для подготовки операндов для следующей операции или для записи результата в регистр или память. Доступные в Z3 команды требуют для своего исполнения следующее число циклов:

Умножение:	16 циклов
Деление:	18 циклов
Квадратный корень:	20 циклов
Сложение:	3 цикла
Вычитание:	4 или 5 циклов, в зависимости от результата
Считывание с клавиатуры:	от 9 до 41 циклов, в зависимости от экспоненты
Отображение результата:	от 9 до 41 циклов, в зависимости от экспоненты
Чтение из памяти:	1 цикл
Запись в память:	0 или 1 цикл

По словам Цузе, умножение занимает 3 секунды[2]. Учитывая то, что умножение выполняется за 16 циклов, можно считать, что тактовая частота Z3 составляла (16/3) ≈ 5.33 Гц.

Число циклов для команд ввода/вывода переменно, так как оно зависит от экспоненты аргумента. Так как введенное число должно быть преобразовано из десятичной в двоичную форму, количество необходимых умножений на 10 или 0.1 зависит от десятичной экспоненты (см. раздел 4).

Сложение и вычитание требуют более, чем один цикл, так как в случае чисел с плавающей запятой экспоненты обоих аргументов должны быть сделаны равными, что требует нескольких дополнительных сравнений и сдвигов.

Число может быть записано в память за *нуль* циклов, т.к. результат последней арифметической операции может быть перенаправлен по желаемому адресу в памяти. Таким образом запись в память происходит одновременно с последним циклом арифметической операции.

2.5. Модель программирования

Важную роль играет описание модели программирования, т.е. части машины, видимой для программиста. С точки зрения програмного обеспечения Z3 состоит из 64 слов памяти, которые могут быть загружены в два регистра, назовем их R1 и R2. Эти два регистра содержат два аргумента необходимых для арифметических операций. Программист может писать любые последовательности команд, учитывая при этом состояние регистров.

Прежде всего нужно помнить следующее: при выполнении первой в программе операции чтения из памяти (Pr *z*), содержимое ячейки памяти с адресом *z* будет перенесено в R1. Каждая следующая операция чтения из памяти переносит слово из памяти в R2. Операция „чтение с клавиатуры“ записывает введенное число в R1 и удаляет содержимое R2. Аргументы арифметических операций не указываются в коде команды. Команды имеют следующую семантику:

Умножение:	R1:=R1*R2
Деление:	R1:=R1/R2
Сложение:	R1:=R1+R2

[2] Это, конечно же,просто курьезное совпадение, что програмная симуляция Z3, осуществленная моими студентами, выполняет умножение также за три секунды!

Вычитание:	R1:=R1-R2
Вычисление квадратного корня:	$R1:=\sqrt{R1}$

После каждой арифметической операции R2 обнуляется, а результат записывается в R1. Последующие операции чтения переписывают R2. Операции „запись" и „выдача результата" обращаются всегда к R1. После операции „запись" и операции „выдача результата" в R1 записывается нуль. Следующая операция чтения переписывает R1.

Чтобы лучше понять модель программирования Z3, рассмотрим пример. Допустим мы хотим посчитать значение полинома методом Горнера:

$$((a_4 x + a_3)x + a_2)x + a_1$$

Предположим, что константы a_4, a_3, a_2, a_1 записаны в ячейках памяти с адресами 4,3,2,1 соответственно. Число x записано в ячейке памяти с адресом 5. Тогда наша программа выглядит так:

Pr4	Записать a_4 в R1
Pr5	Записать x в R2
Lm	Умножить R1 и R2, результат в R1
Pr3	Записать a_3 в R2
Ls_1	Сложить R1 и R2, результат в R1
Pr5	Записать x в R2
Lm	Умножить R1 и R2, результат в R1
Pr2	Записать a_2 в R2
Ls_1	Сложить R1 и R2, результат в R1
Pr5	Записать x в R2
Lm	Умножить R1 и R2, результат в R1
Pr1	Записать a_1 в R2
Ls_1	Сложить R1 и R2, результат в R1
Ld	Выдать результат

После выполнения последней команды процессор перейдет в начальное состояние. Можно запускать следующую программу.

3. Блоковая диаграмма Z3

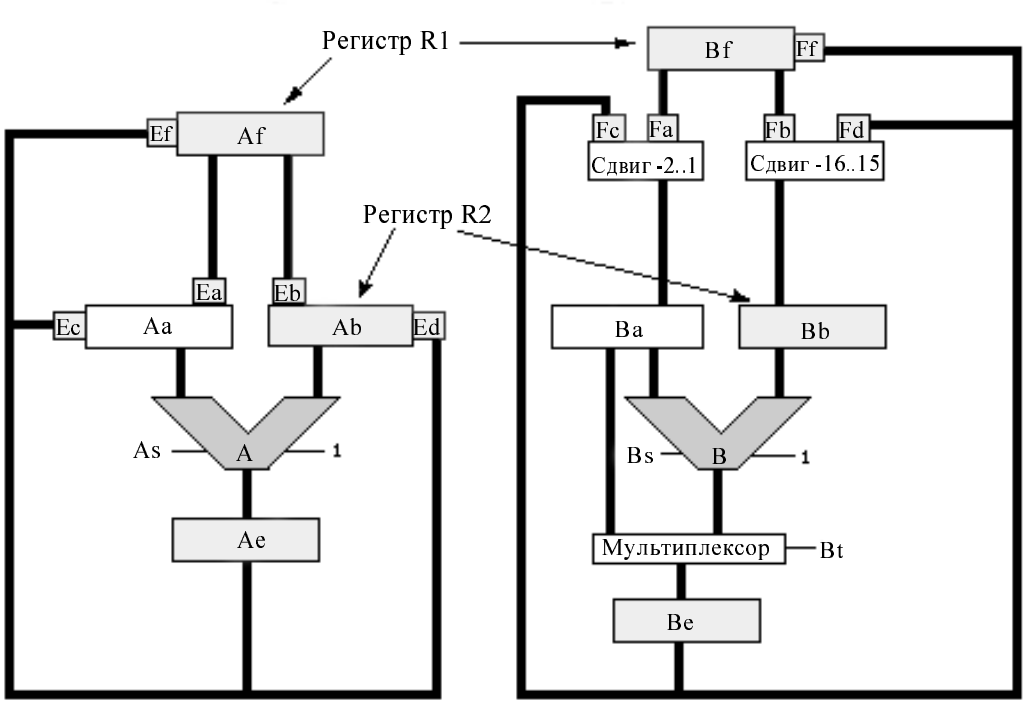
В этой части мы рассмотрим подробней структуру Z3 и опишем его существенные компоненты. В центре внимания находится осуществление правильной синхронизации между имеющимися компонентами.

3.1. Процессор

На рисунке 3 схематично изображен арифметический элемент Z3. Он состоит из двух частей: левая сторона производит операции над экспонентами чисел с плавающей запятой, правая – над мантиссами. Af и Bf – регистры, используемые для записи экспоненты и мантиссы одного числа с плавающей запятой, и составляющие регистр R1 модели программирования. Поэтому в дальнейшем мы будем обозначать R1 как пару регистров <Af,Bf>. Пара регистров <Ab,Bb> содержит экспоненту и

мантиссу числа, находящегося в регистре R2. Пара <Aa,Ba> содержит экспоненту и мантиссу третьего, временного, регистра, невидимого для программиста. Сумматоры A и B выполняют сложение и вычитание экспонент и мантисс. Результат операции над экспонентами заносится в Ae, над мантиссами - в Be. В части B переключатель(multiplexer) позволяет заносить в Be (результат операции) содержимое Ba или результат операции из сумматора B. Переключатель регулируется реле Bt (если Bt=0, Ba переносится в Be).

Рис.3 Регистры и вычислительное устройство



Элементы, обозначенные Ea,Eb,Ec,Ed,Ef,Fa,Fb,Fc,Fd и Ff, представляют релейные переключатели, которые открывают и закрывают шину данных. Если, например, нужно передать содержимое регистра Af в регистр Aa, переключатель Ea будет установлен на 1 (Aa:=Af). Как видно из диаграммы, содержимое регистра Af с помощью переключателя может быть перенесено в Aa или Ab. Содержимое Ae может быть перенесено в Aa, Ab или Af. Структура части B похожа на структуру части A, но кроме переключателя, управляемого Bt, в ней есть еще сдвигатель между Bf и Ba, и между Bf и Bb. Первый сдвигатель может передвинуть мантиссу до двух знаков вправо или на один знак влево. Это означает деление на 4 или умножение на 2.

Второй сдвигатель может передвинуть мантиссу от одного до 15 знаков влево и от одного до 16 знаков вправо. Эти передвижения необходимы для сложения и вычитания чисел с плавающей запятой. Таким образом умножение или деление на 2 легко выполняется при считывании операндов для следующей арифметической операции, и не требует дополнительного времени.

Регистры имеют следующую длину:

Af	7 бит	Bf	17 бит
Aa	7 бит	Ba	19 бит
Ab	7 бит	Bb	18 бит

Ae 8 бит Be 18 бит

Ae имеет дополнительный бит для сложения экспонент. В части B процессор имеет два дополнительных бита для мантиссы (b-15,b-16) и использует бит b_0, который не запоминается. Биты b-15 и b-16 позволяют увеличить точность вычислений. Поэтому общее число бит, необходимых для записи результата арифметической операции, в Bf – 17. В регистрах Ba и Bb требуются дополнительные биты (ba_2,ba_1,bb_1) для работы с промежуточными результатами арифметических операций. Например алгоритм вычисления квадратного корня, может привести к тому, что при промежуточных вычислениях в Ba будут использованы до трех бит слева от запятой. Основные элементарные операции – сложение и вычитание экспонент и мантисс. Если реле As(Bs) установлено на 1, значение второго аргумента Ab(Bb) будет загружено в блок A(B) со знаком „-“. Таким образом сумматор A выполняет вычитание аргументов, если реле As установлено на 1, в противном случае выполняется сложение. Аналогично для части B и реле Bs. Изображенная на рисунке 3 константа 1 используется для построения числа, которое при сложении по модулю 2 с исходным числом дает 0.

Предположим, нужно сложить два числа с одинаковыми экспонентами. Первая экспонента будет запомнена в Af, вторая - в Ab. Поскольку экспоненты равны, в части A никакие операции не требуются. В части B мантисса первого числа будет запомнена в Bf, мантисса второго - в Bb. Первый шаг состоит в том, чтобы поместить содержимое Bf в Ba путем установления Fa на 1.

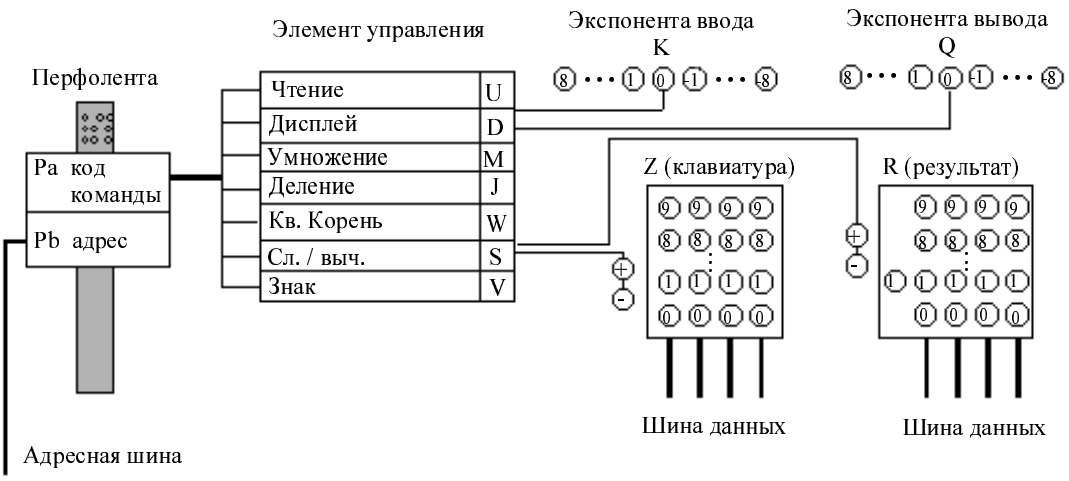
Затем будет произведено сложение, реле Bt установлено на 1, и результат Ba+Bb помещен в Be. Реле Ff будет теперь установлено на 1 и результат занесен в Bf. Как видно из примера, информация может перемещаться между регистрами. Опрелеоенная конфигурация релейных переключателей приводит к выполнению желаемых операций. В Z3 для этого использована технология, похожая на микропрограммирование.

3.2. Элемент управления

На рисунке 4 изображена подробная диаграмма элемента управления и устройств ввода/вывода. Элемент управления отвечает за правильное исполнение команд. Для каждой команды существует специальный контур. Контур Pa декодирует код команды, считанной с перфоленты. Если это команда работы с памятью, Pb передает шине данных значение нижних шести бит кода команды.

Схема Z представляет клавиатуру, используемую для ввода десятичных чисел. В каждом столбце может быть активирована только одна клавиша. Экспонента устанавливается нажатием одной из клавиш от -8 до 8 в схеме K. Устройство вывода похоже на устройство ввода, только в нем включенные лампы показывают десятичное число, экспоненту (схема Q) и знак числа. В устройстве вывода есть еще пятая цифра, которая может быть 0 или 1.

Рис.4 Блок управления и устройства ввода/вывода

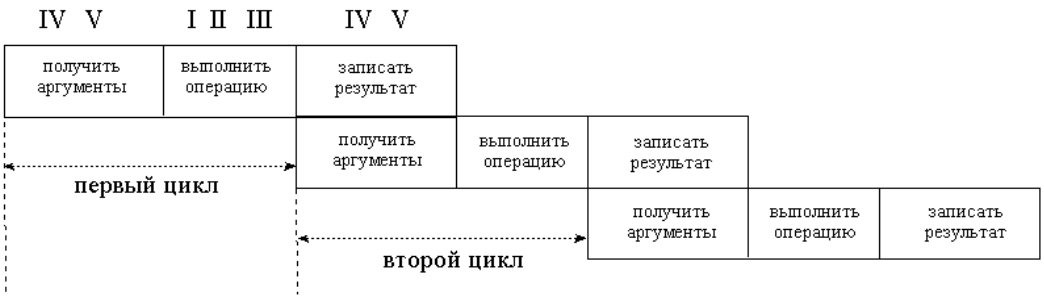


Как только десятичное число введено, шина данных передает цифры в регистр Ba и начинается серия операций. Десятичное число должно быть переведено в двоичную систему. Для этого требуется ряд умножений, число которых пропорционально абсолютному значению экспоненты. Для экспоненты равной нулю необходимо 9 циклов, а для экспоненты 8 требуется уже 9+4*8=41 цикл.

3.3. Управление Z3 посредством микрокоманд

Сердце элемнта управления – это его микросиквенсеры (блоки управления микрокомандами). Прежде чем описывать их способ работы, необходимо взглянуть на связь арифметических команд в Z3. Каждый цикл в Z3 разделяется на пять ступеней. Ступени IV и V используются для переноса информации из одной части машины в другую. В течение ступеней I, II и III выполняются сложение или вычитание экспонент в части A Z3, и мантисс в части B. Назовем это „исполнительной частью" команды. Типичная команда принимает свои аргументы, выполняет операцию и выдает результат. Цузе уделил большое внимание уменьшению времени исполнения, он реализовал одновременное выполнение фазы подготовки аргументов следующей команды и фазы записи результата

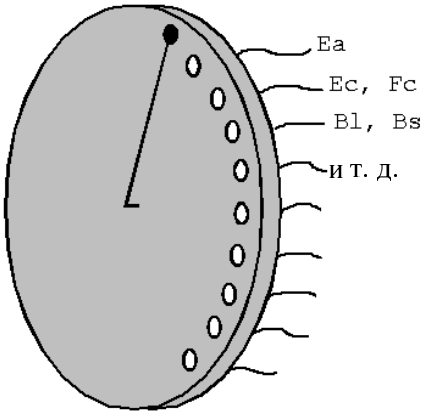
Рис.5 Конвейер выполнения команд Z3



текущей. Можно сказать, что цикл исполнения команды состоит из двух фаз, как показано на рис. 5; первые два цикла представлены последовательностью входящих в них команд. Это соглашение исползуется в таблицах числовых алгоритмов, которые мы обсудим позже.

Управление микрокомандами осуществляется с помощью особых колес. Имеется отдельное колесо для алгоритма умножения, одно для вычитания и еще одно для извлечения квадратного корня. Показанная на рис. 6 подвижная стрелка начинает двигаться по часовой стрелке, как только элемент управления начинает декодировать соответствующую команду. С каждым циклом стрелка передвигается из одной позиции в другую. Стрелка находится под напряжением и активирует схемы, с которыми входит в контакт. В примере, привденном на рисунке, стрелка устанавливает реле Ea на 1 в первом цикле. Это влечет перенос содержимого регистра Af в Aa. В следующем цикле активируются реле Ec и Fc. Таким образом результаты операций в частях A и B машины записываются соответственно в регистры Aa и Bb. Как легко видеть, такие колеса управления предоставляют удобную основу для изменения точной последовательности событий в течение операции. Они соответствуют микросиквенсерам, которые используются в современных микропроцессорах. Я не хочу называть это настоящим микропрограммированием, так как в этом случае микропрограмма „встроена в железо", однако очевидно, что микропрограммирование и микросиквенсирование тесно связаны.

Рис.6 Колесо управления микрокомандами



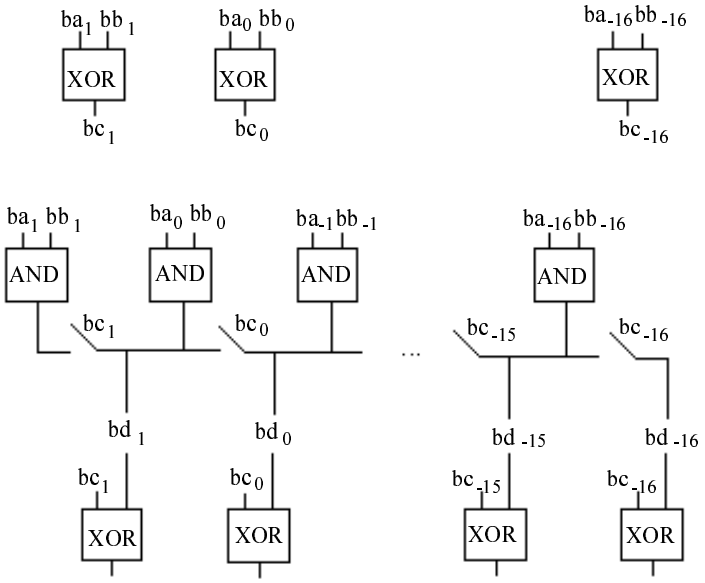
Обширное использование микросиквенсеров позволило Цузе упростить Z3. После того как были спроектикованы основные схемы, оставалось только улучшать управление, пока не были найдены оптимальные последовальности событий. Есть много деталей, на которые должен обратить внимание программист при разработке "Микропрограмм", так как в противном случае короткие замыкания могут повредить оборудование. Z1 с ее механическим дизайном была в этом смысле еще чувствительнее, чем Z3. Даже после того, как он был готов, существовали ошибочные последовательности, которые программист не должен был допускать, чтобы не повредить машину. Одна такая ошибочная последовательность была по недосмотру запущена на реконструированной Z1 в Берлинском Музее Техники и Транспорта и привела к легкому повреждению машины.

4. Арифметические алгоритмы.

4.1. Перенос при сложении.

Важное свойство Z3 - это наличие сумматоров, которые в состоянии производить сложение и вычитание с использованием техники carry look-ahead. Если бы параллельное двоичное сложение было реализовано напрямую без такой оптимизации, то биты переноса должны были бы передаваться от одной битовой позиции к другой. Для мантиссы требовалось бы в этом случае минимум 16 циклов для надежного переноса битов.

Рис.7 Схема, реализующая *carry look ahead* (соответствует рис. 19 в [16])



Сумматоры Z3 значительно быстрее - они производят одно сложение или вычитание на ступенях I,II и III одного единственного цикла. Вычитание сводится к сложению конвертированием каждого бита второго аргумента и прибавлением 1 к его младшему биту.

Рассмотрим сложение регистров Ba и Bb. Будем обозначать i-ый бит регистра Bb через bb*i* или Bb[i], в зависимости от того, какая форма будет для нас удобнее. То же соглашение будем использовать и для других регистров. Сперва вычисляется промежуточный результат, который представляет собой побитовый XOR обоих регистров, то есть bc*i*=(ba*i* XOR bb*i*). Второй промежуточный результат это побитовый AND обоих регистров: ba*i* AND bb*i*. Следующая операция касается определения битов, для которых необходим перенос. Промежуточные результаты bd*i* вычисляются с помощью показанной на рис. 7 схемы. Обратите внимание на то, что когда бит равен 1, соответствующая линия находится под напряжением, иначе она отключена от источника напряжения (так называемая схема с тремя состояниями). Это следует учитывать при рассмотрении данной схемы. На рисунке показано нерабочее состояние реле bc*1*,...,bc*-16*. Если бит bc*i* равен 1, замыкается соответствующее реле. Окончательный результат - be*i*=bd*i* XOR bc*i*. Обратите внимание, насколько легче делает использование реле перенос битов из одной

позиции в другую. Перенос не задерживается при передаче из одной позиции в другую, так как все реле активируются одновременно.

В Z4 Цузе разработал еще более упрощенную схему, которая обходится двумя реле (каждое с четырмя контактами) для Ba[i] и Bb[i] ([16], рис. 20). Z4 сейчас находится в Немецком Музее [5].

Ниже описаны алгоритмы, которые использованы в Z3 для операций с числами с плавающей запятой. Все они полностью соответствуют алгоритмам, используемым в сегодняшних простейших процессорах, работающих с числами с плавающей запятой [6].

4.2 Обработка исключений при работе с числами с плавающей запятой.

Проблема с нотацией чисел с плавающей запятой Цузе состоит в том, что для представления числа нуль должны использоваться особые соглашения. Z3 решает эту проблему и обрабатывает другие исключения (переполнение, вырождение в нуль) проверкой значения экспоненты после каждой арифметической операции или после каждого чтения из памяти. Специальная схема следит за состоянием шины Aa и перехватывает исключения. Каждое число с экспонентой -64 рассматривается как нуль: реле, помеченное Nn1, устанавливается в 1, если прочитанное число сохранено в паре регистов <Af,Bf>. Если же прочитанное число сохранено в паре регистров <Ab,Bb>, то в 1 будет установлено реле Nn2. Таким образом мы всегда знаем, если один или оба аргумента арифметической операции равны нулю. Подобные действия выполняются для экспонент со значением 63 (согласно нотации Цузе, бесконечность). В этом случае реле Ni1 или Ni2 устанавливается в 1, в зависимости от того, в какой паре регистров сохранено прочитанное число.

Операции, содержащие "исключительные" числа (нуль и бесконечность), выполняются как обычно, но их результат перезаписывается следящей схемой. Пусть, например, выполняется умножение и первый аргумент - нуль (Nn1 устанавливается в 1). Вычисление идет как обычно, но в каждом цикле схема наблюдения подает на выход сумматора части A результат -64. Неважно, какая операция с мантиссой производится, так как экспонента результата устанавливается в -64 и, таким образом, результат равен нулю. Деление на бесконечно большое число обрабатывается похожим образом. Z3 распознает такие неопределенности, как 0/0, ∞-∞, ∞/∞ и 0*∞. В каждом из этих случаев на устройстве вывода загорается соответствующим образом подписанная лампочка, и машина останавливается. Z3 выдает правильный результат и тогда, когда один из аргументов равен 0 или ∞, а другой - обыкновенное число[3].

На выходе сумматора экспонент за результатом наблюдает дополнительная схема. Если экспонента больше или равна 63, произошло переполнение и результат должен быть установлен в ∞. В случае, когда экспонента меньше, чем -64, возникает так называемый машинный нуль и результат выдается равным нулю. Для этого соответствующее реле (Nn1 или Ni1) устанавливается в 1.

Цузе смог обойтись всего парой реле для реализации обработки исключений. Это свойство – одно из самых элегантных во всей Z3.

[3] Это не верно для Z1. Цузе думал о реализации обработки исключений в Z1, но не сделал этого. Машина не могла правильно работать с вычислениями, которые содержали нуль.

Многие из первых микропроцессоров семидесятых годов не могли обрабатывать исключения и оставляли это программному обеспечению. Подход Цузе гораздо лучше, так как он освобождает программиста от утомительных проверок границ чисел перед каждой операцией.

4.3 Сложение и вычитание

Для сложения и вычитания чисел с плавающей запятой x и y необходимо сначала сделать равными их экспоненты. После этого надо сложить или вычесть их мантиссы. Если экспоненты различны, мантисса меньшего числа должна быть сдвинута на соответствующее число разрядов вправо и его экспонента соответственно увеличена (для сохранения значения числа); пока экспоненты не сравняются. После семнадцати сдвигов может, конечно, случиться, что число обратится в нуль.

Рис.8 Три цикла, необходимых для выполнения операции сложения. Перед выполнением операции аргументы записаны в регистрах <Af,Bf> и <Ab,Bb>.

Цикл	Ступень	Экспонента	Мантисса
0	I,II,III		
1	IV,V	Aa:=Af	
	I,II,III	Ae:=Aa-Ab	Be:=0+Bb
2	IV,V	если (Ae≥0) то Ab:=0,Aa:=Af иначе Aa:=0	если (Ae≥0) то Ba:=Bf,Bb:=Be(сдвинутый) иначе Ba:=Be,Bb:=Bf(сдвинутый) (Be или Bf сдвигаются на |Ae| мест вправо)
	I,II,III	если (Be≥0) то Ae:=Aa+Ab+1 иначе Ae:=Aa+Ab	Be:=Ba+Bb
3	IV,V	Af:=Ae	если (Be≥2) то Bf:=Be/2 иначе Bf:=Be

Перед тем, как выбрать вид операции, сравниваются знаки чисел. Если требуется произвести сложение и знаки чисел одинаковы, выполняется сложение; если знаки различны, выполняется вычитание. Если требуется произвести вычитание и знаки чисел различны, выполняется сложение; если знаки равны, выполняется вычитание. Специальная схема устанавливает знак результата в зависимости от знаков аргументов и знака промежуточного результата.

Сложение и вычитание управляется цепью реле (не колесом управления), так как число максимально возможных циклов невелико. На рис. 8 показано, как синхронизируется сложение двух чисел. Вначале аргументы сложения находятся в парах регистров <Af,Bf> и <Ab,Bb>. В первом цикле вычитаются экспоненты. Во втором цикле мантисса числа с большей экспонентой загружается в регистр Ba, мантисса числа с меньшей экспонентой в регистр Bb. Мантисса в регистре Bb сдвигается на столько разрядов, сколько составляет модуль разности экспонент (случаями, в которых число после сдвигов обращается в нуль, занимается обработка исключений).На ступенях I, II, III второго цикла складываются мантиссы и, наконец, процессор проверяет, превышает ли результат 2. Если да, то мантисса результата сдвигается на одну позицию вправо и экспонента увеличивается на 1. Стоит обратить внимание на то, что тест "Be≥2" производится в части A

арифметического элемента, после того как Be вычисляется в части B на ступенях I, II, III второго цикла.

Рис.9 четыре (или пять) циклов, необходимых для выполнения операции вычетания. Перед выполнением операции аргументы записаны в регистрах <Af,Bf> и <Ab,Bb>.

Цикл	Ступень	Экспонента	Мантисса
0	I,II,III		
1	IV,V	Aa:=Af	
	I,II,III	Ae:=Aa-Ab	Be:=0+Bb
2	IV,V	если (Ae≥0) то Ab:=0,Aa:=Af иначе Aa:=0	если (Ae≥0) то Ba:=Af,Bb:=Be(сдвинутый) иначе Ba:=Be,Bb:=Bf(сдвинутый) (Be или Bf сдвигаются на |Ae| мест вправо)
	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:=число сдвигов	Bb:=Be(сдвинутый) (Be нормализируется за счет сдвига влево)
	I,II,III	Ae:=Aa-Ab	Be:=0+Bb
5	IV,V	Af:=Ae	Bf:=Be

При вычитании необходимо произвести четыре или пять циклов. На рис. 9 показано, как синхронизируется вычитание. Первые два цикла почти идентичны первым двум циклам алгоритма сложения, в них вычитаются мантиссы. Цикл 3 выполняется только в том случае, когда разность мантисс отрицательна. Конечная задача третьего цикла – сделать мантиссу результата положительной. Цикл 4 очень важен: разность двух нормализированных мантисс может содержать нули в высших разрядах. Результат нормализируется, причем Be сдвигается влево на соответствующее число мест (это производится с помощью сдвигателя между реле Fd и регистром Bb). Число побитовых сдвигов вычитается из экспоненты в части A процессора. В цикле 5 результат сохраняется в паре регистров <Af,Bf>.

4.4 Умножение

Алгоритм умножения Z3 похож на умножение "в столбик". Он состоит из повторяющихся сложений второго множителя соответственно отдельным цифрам первого. В начале алгоритма аргументы находятся соответственно в парах регистров <Af,Bf> и <Ab,Bb>. Временный регистр <Aa,Ba> устанавливается в нуль. На рис. 10 показано микросеквенсирование с помощью шагового переключателя для умножения. Алгоритм требует 16 циклов для одного прохода. Заметьте, что используются только биты с (-14)-го по нулевой первого множителя. Экспоненты складываются в первом цикле и после этого результат „циркулирует“ в части A арифметического элемента. Мантиссы обрабатываются в части B. Регистр Ba содержит частичный результат вычисления. Основная схема умножения имеет следующую форму:

Ba:=Be/2
Be:=Ba+Bb*(i-ый бит Bf)

для i=-14,...,0. Частичный результат Be сдвигается на одну позицию вправо для получения Ba:=Be/2. Это производится с помощью сдвигателя, связаного с реле Fc.

Рис.10 Шестнадцать циклов для умножения. Перед выполнением операции второй аргумент записан в регистре <Af,Bf> и первый в <Ab,Bb>. *i*-тый бит регистра Bf обозначен Bf[*i*]

Цикл	Ступень	Экспонента	Мантисса
0	I,II,III		
1	IV,V	Aa:=Af	
	I,II,III	Ae:=Aa+Ab	если (Bf[-14]=1) то Be:=Ba+Bb иначе Be:=Ba
2	IV,V	Aa:=Ae,Af:=0,Ab:=0	Ba:=Be/2
	I,II,III	Ae:=Aa+Ab	если (Bf[-13]=1) то Be:=Ba+Bb иначе Be:=Ba
3	IV,V	Aa:=Ae	Ba:=Be/2
	I,II,III	Ae:=Aa+Ab	если (Bf[-12]=1) то Be:=Ba+Bb иначе Be:=Ba
:	:	:	:
I	IV,V	Aa:=Ae	Ba:=Be/2
	I,II,III	Ae:=Aa+Ab	если (Bf[*i*-15]=1) то Be:=Ba+Bb иначе Be:=Ba
:	:	:	:
15	IV,V	Aa:=Ae	Ba:=Be/2
	I,II,III	если (Be≥2) то Ae:=Aa+1	если (Bf[0]=1) то Be:=Ba+Bb иначе Be:=Ba
16	IV,V	Af:=Ae	если (Be≥2) то Bf:=Be/2 иначе Bf:=Be Bb:=0

Результат умножения - число 1≤r<4 (для аргументов в пределах допустимых числовых границ). В последнем цикле проверяется r≥2. Если да, то результат сдвигается на одно место вправо и к экспоненте результата добавляется 1.

4.5 Деление

Алгоритм деления аналогичен алгоритму умножения, только вместо сложения повторяется вычитание. Вначале делимое содержится в паре регистров <Af,Bf>, делитель в паре регистров <Ab,Bb>. Временная пара <Aa,Ba> устанавливается в нуль. На рис. 11 показано микросеквенсирование, осуществляемое шаговым переключателем деления. Алгоритм требует 18 циклов.

Основная идея алгоритма проста. Экспонента результата вычисляется как разность экспонент делимого и делителя. Переходим к мантиссе: мы хотим вычислить x/y для нормализованных мантис x и y. Так как мы работаем с нормализованными числами, то первая цифра результата равна 1, если x≥y, и равна нулю, если x<y. В первом случае мы устанавливаем первую цифру результата в 1 и вычисляем остаток по формуле x-y. Остаток опять же делим на y. Для этого остаток сдвигаем на одну позицию влево и новый бит результата заносится в [-1]-ю позицию регистра Bf (так мы исправляем эффект сдвига). В случае, если бит результата равен нулю, остаток это просто x и деление продолжается как в первом случае.

Рис.11 Восемнадцать циклов для деления. Перед выполнением операции делимое записано в регистре <Af,Bf>, делитель в <Ab,Bb>. *i*-тый бит регистра Bf обозначен Bf[*i*]

Цикл	Ступень	Экспонента	Мантисса
0	I,II,III		
1	IV,V	Aa:=Af	Ba:=Bf
	I,II,III	Ae:=Aa-Ab	если (Ba-Bb≥0) то Be:=Ba-Bb,bt:=1 иначе Be:=Ba, bt:=0
2	IV,V	Aa:=Ae,Ab:=0	Bf:=0 если (bt=1) то Bf[0]:=1 Ba:=2xBe
	I,II,III	Ae:=Aa+Ab	если (Ba-Bb≥0) то Be:=Ba-Bb,bt:=1 иначе Be:=Ba, bt:=0
3	IV,V	Aa:=Ae	если (bt=1) то Bf[-1]:=1 Ba:=2xBe
	I,II,III	Ae:=Aa+Ab	если (Ba-Bb≥0) то Be:=Ba-Bb,bt:=1 иначе Be:=Ba, bt:=0
:	:	:	:
i	IV,V	Aa:=Ae	если (bt=1) то Bf[2-*i*]:=1 Ba:=2xBe
	I,II,III	Ae:=Aa+Ab	если (Ba-Bb≥0) то Be:=Ba-Bb,bt:=1 иначе Be:=Ba, bt:=0
:	:	:	:
16	IV,V	Aa:=Ae	если (bt=1) то Bf[-14]:=1 Ba:=2xBe
	I,II,III	Ae:=Aa+Ab	если (Ba-Bb≥0) то Be:=Ba-Bb,bt:=1 иначе Be:=Ba, bt:=0
17	IV,V	если (Bf[0]=0) то Ab:=-1	Ba:=Bf,Bb:=0
	I,II,III	Ae:=Aa+Ab	Be:=Ba+Bb
18	IV,V	Af:=Ae	если (Bf[0]=0) то Bf:=2xBe иначе Bf:=Be

Основной цикл деления имеет следующую форму:

Ba:= 2 x Be

если (Ba-Bb≥0) то Be:= Ba-Bb, Bf[i]:=1
иначе Be:= Ba, Bf[i]:=0

для i=0,...,-14. Частичный результат Be сдвигается на одну позицию влево для получения Ba:=2xBe. Это производится с помощью сдвигателя, соединенного с реле Fc.

Результат деления мантисс это число 1/2≤r<2. Это условие проверяется в циклах 17 и 18. Если r<1, то для получения нормализированного числа от экспоненты отнимается 1 и результат сдвигается на одну позицию влево.

4.6 Вычисление квадратного корня

Алгоритм вычисления квадратного корня - это изюминка Z3. Рисунок 12 показывает микросеквенсирование соответствующих команд, всего требуются 20 циклов. Аргумент операции запоминается в паре регистров <Af,Bf>. Пара регистров <Aa,Ba> инициализируется нулем. Алгоритм вычисляет квадратный корень чисел с четными экспонентами.

Если экспонента нечетная, мантисса сдвигается на одну позицию влево и экспонента увеличивается на 1. Экспонента результата (вычисляется в цикле 19) в два раза меньше исходной экспоненты

Рис.12 Двадцать циклов для вычисления квадратного корня. Перед выполнением операции аргумент записан в регистре <Af,Bf>. *i*-тые биты регистров Af и Bf обозначены Af[*i*] и Bf[*i*] соответственно.

Цикл	Ступень	Экспонента	Мантисса
0	I,II,III		
1	IV,V		если (Af[0]=1) то Ba:=2xBf иначе Ba:=Bf Bb[0]:=1
	I,II,III		если (Ba-Bb≥0) то Be:=Ba-Bb,bt:=1 иначе Be:=Ba, bt:=0
2	IV,V		Bf:=0 если (bt=1) то Bf[0]:=1 Ba:=2xBe,Bb:=2xBf,Bb[-1]:=1
	I,II,III		если (Ba-Bb≥0) то Be:=Ba-Bb,bt:=1 иначе Be:=Ba, bt:=0
3	IV,V		если (bt=1) то Bf[-1]:=1 Ba:=2xBe,Bb:=2xBf,Bb[-2]:=1
	I,II,III		если (Ba-Bb≥0) то Be:=Ba-Bb,bt:=1 иначе Be:=Ba, bt:=0
:	:	:	:
i	IV,V		если (bt=1) то Bf[2-*i*]:=1 Ba:=2xBe,Bb:=2xBf,Bb[1-*i*]:=1
	I,II,III		если (Ba-Bb≥0) то Be:=Ba-Bb,bt:=1 иначе Be:=Ba, bt:=0
:	:	:	:
18	IV,V		если (bt=1) то Bf[-16]:=1 Ba:=2xBe,Bb:=2xBf
	I,II,III		если (Ba-Bb≥0) то Be:=Ba-Bb,bt:=1 иначе 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

Основная идея алгоритма - свести вычисление квадратного корня к делению[4]. Если мы хотим вычислить квадратный корень числа x, то мы ищем число Q, такое что $x/Q=Q$. Результат Q ищется следующим образом: i-тый бит устанавливается на 1 и проверяется, выполнено ли условие $x > Q^2$. Если нет, i-тый бит должен быть установлен на 0.

Предположим, мы уже нашли с нулевого по $-i+1$-й биты результата. Обозначим мантиссу Q_{-i+1}.

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

Затем $-i$-тый бит устанавливается равным q_{-i}, и должно выполняться

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

Это выполнено, если

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

[4] Эта идея лежит в основе метода вычисления квадратного корня, изучаемого ранее в гимназиях.

Определим t_{-i} следующим выражением

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

что можно иначе записать как:

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

использовав рекурсивное определение $2^{-i+1}t_{-i+1} = (x - Q_{-i+1}^2)$. Упростив последнее выражение, получим:

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

Если для $q_{-i} = 1$ t_{-i} положительно, то -*i*-тый бит результата устанавливаем равным 1, т.е. Bf[-i]:=1. Если t_{-i} отрицательно, устанавливаем Bf[-i]:=0. Рекурсивное вычисление начинается с $t_0 = x$. Q_{-i+1} представляет на каждом шаге частичный результат в регистре Bf. -*i*-тый бит устанавливается сперва на 1 и проверяется знак t_{-i}.

Основной цикл алгоритма вычисления квадратного корня имеет следующую форму:

```
Ba:=2xBe
Bb:=2xBf
Bb[-i]:=1
если (Ba-Bb≥0)  то    Be:=Ba-Bb,Bf[-i]:=1
                иначе Be:=Ba,   Bf[-i]:=0
```

Для вычисления квадратного корня используются все биты регистра Bf. Если исходное число лежит в допустимых пределах числовых границ, то и результат будет находиться в тех же пределах.

4.7 Команды ввода/вывода

Две наиболее сложных команды в Z3 связаны с вводом/выводом десятичных чисел. четырехзначное десятичное число, введенное с клавиатуры, сперва переводится в целое двоичное число. Это осуществляется следующим образом: цифры считываются последовательно, каждая считанная цифра переводится в двоичное число и запоминается в битах Ba[-10], Ba[-11], Ba[-12], Ba[-13] регистра Ba. число в регистре Ba умножается на 10, и процедура повторяется для остальных цифр. За четыре прохода десятичный ввод переводится в двоичное число. Сложнее с экспонентами. Если экспонента *e* положительна, то мантиссу нужно *e* раз умножить на 10. Если экспонента отрицательна, мантиссу нужно |*e*| раз умножить на 0.1 . Умножение на 10 довольно просто: мантисса может быть сдвинута в Be на один бит и записана в Ba (т.е. Ba:=2xBe). Одновременно Be может быть сдвинут на три бита влево и записан в Bb (т.е. Bb:=8xBe). Тогда сложение Ba и Bb дает желаемый результат: умножение исходного числа в Be на 10. Одно такое умножение требует 4 цикла, т.е. для десятичной экспоненты +8, например, потребовалось бы 32 цикла.Это значит, что одно десятичное число с экспонентой +8 будет считано за 41 цикл, так как операция чтения требует минимум 9 циклов.

В случае отрицательной экспоненты умножение на 0.1 осуществляется с помощью сдвигателя и сумматора. Это умножение сложнее, потому что 0.1 в двоичной системе нетривиальное периодическое число. Описание этой микропоследовательности увело бы нас далеко от темы, поэтому мы его опустим.

Вывод работает за счет повторяющихся умножений или делений на 10. Если двоичная экспонента числа в регистре R1 положительна, то число будет умножено на 0.1 столько раз, пока экспонента не будет равна 2 и первые четыре бита в регистре Bf будут содержать число от 0 до 9 (0000 и 1001). Это - десятичное число, которое будет высвечено в следующем столбце лампочек, означающих вывод. Показанная цифра вычитается из мантиссы и процедура продолжается для остальных цифр. Если двоичная экспонента числа в регистре R1 отрицательна, происходит то же самое, только с умножением на 10.

5. Полная архитектура Z3

Сейчас мы можем лучше понять подробную диаграмму Z3, изображенную на рисунке 13. Мы видим некоторые компоненты, рассмотренные в предыдущих частях.

Элемент управления и устройство ввода/вывода мы уже рассмотрели раньше. Заметьте, что четыре десятичные цифры, введенные с клавиатуры, передаются посредством последовательной активации релейных переключателей Za, Zb, Zc и Zd.

Реле Eg и Ei используются для занесения двух полезных констант +13 и -4 прямо в регистр экспоненты. Сдвигатель Ee между регистрами Af и Aa используется для вычисления квадратного корня. Экспонента результата (Aa) в два раза меньше экспоненты исходного числа (Af).

Реле Ah_1 работает как переключатель между парами регистров. Если оно установлено на 0, то для операции чтения доступна пара регистров <Af,Bf>. Если оно установлено на 1, то доступна пара <Ab,Bb>. Это реле устанавливается на 0 через сигнальную линию a_i. Сигнальные линии a_1,a_j,b_1 и b_j используются при необходимости для удаления содержимого регистров Af, Ab, Bf и Bb.

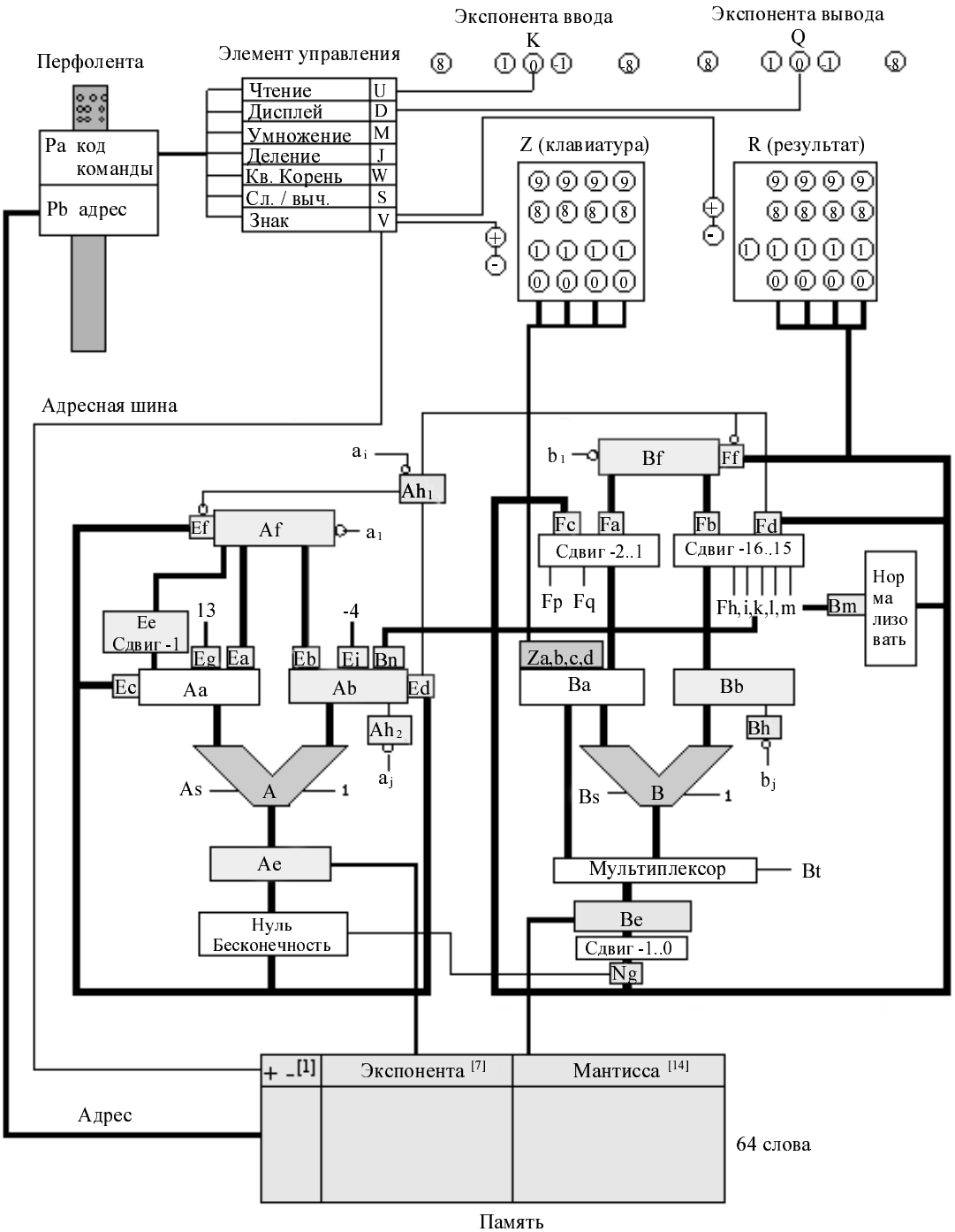
Элемент, обозначенный „нуль, бесконечность“, расположенный под Ae, представляет переключатели для обработки исключений. Они постоянно проверяют шину данных (результат операций и передачу данных из памяти) и, если надо, устанавливают соответствующие флажки. Сдвигатель под Be используется для передвижения мантиссы на один знак вправо. Это обеспечивает необходимую нормализацию мантиссы каждый раз, когда 2≥Be.

Реле Fp и Fq регулируют количество и направление передвижений в сдвигателе под релейными переключателями Fc и Fa. Fh, Fi, Fk, Fl и Fm выполняют ту же функцию по отношению к другому сдвигателю. С помощью этих пяти бит может быть представлено число от -16 до 15, что соответствует возможному количеству и направлению передвижений для второго сдвигателя. После выполнения такого передвижения, число, представленное реле с Fh по Fm, заносится через релейный переключатель Bn в регистр Ab, чтобы соответствующим образом изменить экспоненту результата. Если число передвигается на 10 знаков влево, то из экспоненты будет вычтено +10. Такие большие передвижения в основном нужны после вычитания.

Посмотрим еще раз на диаграмму Z3. Теперь все выглядит вполне обычно и осмысленно, как в любом простом современном процессоре. Удивительно, как Конрад Цузе с самого начала смог найти действенную архитектуру. Процессор Z3 содержит всего 600 реле, в памяти использовано в три раза больше. Цузе вынужден был все время пересматривать логическую структуру своей машины и оптимизировать

дезайн, чтобы на сколько можно уменьшить расходы на материалы. Он не имел такой роскоши, как почти неограниченные финансы, выделяемые американским военным министерством на разработку ENIAC'a, или IBM на MarkI. Он опирался только на себя, и хотя это было его преимуществом с концептуальной стороны, оно же было и недостатком, если учесть, какое ничтожное влияние оказали Z1 и Z3 на развивавшуюся американскую компьютерную промышленность после второй мировой войны [14].

Рис.13 Полная архитектура Z3



6. Изобретение компьютеров

Главным недостатком Z3 было отсутствие условного перехода в наборе его команд. Это было возможно реализовать, хотя для программы, записанной на перфоленте, это было бы несколько неуклюже, потребовались бы только несколько дополнительных схем. Иногда черту, разделяющую простую вычислительную машину и универсальный компьютер, проводят между машинами с внешним и внутренним сохранением программ. В другой своей работе [10] я показывал, что это – несущественный критерий. Внешняя программа может работать как интерпретатор численных данных. Внешняя программа занимает конкретную часть процессора и данные поступают в программу, так же как универсальная машина Тьюринга работает как интерпретатор. Я аргументировал [14], что для универсальной машины необходим минимальный набор команд и косвенная адресация. Косвенная адресация может быть реализована написанием самоизменяющейся программы, таким образом решающим критерием является набор команд. Машина с памятью, достаточной для хранения как данных, так и команд, сумматором и способная выполнять команды CLR(удалить), INC(увеличить на 1), LOAD(считать/загрузить), STORE(сохранить) и BR(переход в случае нуля) является универсальной машиной (данный набор команд может быть уменьшен[12]). В этом смысле Z1 не был универсальной машиной, также как и другие вычислительные машины того времени. ABC был специальной машиной для выполнения метода Гаусса, в МаркеI из Гарварда отсутствовал условный переход, хотя он мог выполнять FOR-циклы. ENIAC никогда не был программируем через software: его блоки должны были быть соединены соответственно потоку данных. Условные переходы были в ENIAC'е только ограничено выполнимы. А о самоизменяющейся программе не могло быть и речи.

В таблице 2 приведена сравнительная информация о машинах, упомянутых в части 1. Как видно из таблицы, ни одна из машин не выполняет условия для универсальной вычислительной машины. В таблице отсутствует еще одна важная машина: *Relay Interpolator*, изготовленная Джорджем Штибицом при *Bell Telephone Laboratories* в октябре 1939. Машина могла считывать последовательность арифметических команд с перфоленты и выполнять их. В ней использовались реле, специальное кодирование чисел, и она выполняла

Таблица 2 Сравнение особенностей ранних вычислительных машин

Машина	Память и процессор разделены ?	Условный переход ?	Программирование через soft- или hardware	Самоизменяющаяся программа ?	Косвенная адресация ?
Z1 Цузе	Да	нет	Software	нет	нет
Атанасов	Да	нет	Hardware	нет	нет
MarkI(Гарвард)	Нет	нет	Software	нет	нет
ENIAC	Нет	частично	Hardware	нет	нет
Mark1(Манчестер)	Да	да	Software	да	нет
EDSAC	Да	да	Software	да	нет

Машина	Внутреннее кодирование	числа с плавающей запятой ?	Побитовая арифметика ?	архитектура	Техника
Z1 Цузе	двоичное	да	нет	последовательная	механика
Атанасов	двоичное	нет	да	векторная	электроника
MarkI(Гарвард)	Десятичное	нет	нет	параллельная	Электро-механика
ENIAC	Десятичное	нет	нет	поток данных	электроника
Mark1(Манчестер)	двоичное	нет	да	последовательная	электроника
EDSAC	двоичное	нет	нет	последовательная	электроника

операции над числами не с плавающей запятой. Она была функцианально аналогична Z3 и могла выполнять небольшие условные переходы.

В последней строке приведены данные для EDSAC(*Electronic Delay Storage Automatac Calculator*). Эта вычислительная машина была построена в университете Кэмбридж. Руководителем группы был Морис Вилкес, который в 1946 г. имел возможность познакомиться с американскими машинами. EDSAC был двоичной серийной машиной, которая работала с целыми числами. Программы загружались в память, и среди многих других команд был также условный переход. Первая программа была успешно выполнена в мае 1949. Построенный в Манчестере с 1946 по 1948 „малыш" Mark1, тоже внесен в таблицу, т.к. он (насколько известно автору) был первой машиной, которая соответствовала определению универсальной вычислительной машины. „Малыш" Mark1, также как и позже Ferranti MARKI, был построен под руководством Ф.Вильямса и Т.Килбурна. Программы загружались в цифровую память со свободным доступом, реализованную с помощью катодно-лучевых трубок. Имелись все необходимые элементарные команды (в измененной форме), и хотя и не было косвенной адресации, существовала возможность писать самоизменяющиеся программы. Первая программа была запущена 21 июня 1948г., она вычисляла наибольший делитель числа 2^{18} и содержала 3,5 миллиона операций, время работы составило 52 минуты [8]. В сентябре Алан Тьюринг был приглашен на должность доцента математики в Манчестер и написал несколько программ для первого полноценного компьютера в мире. Его представление об универсальной вычислительной машине, опубликованное в 1936 г., в том же году, когда была изготовлена память для Z1, было воплощено в Марке1. Таблица ясно показывает, что изобретение компьютера было совместным достижением, охватившим два континента и двенадцать лет.

Литература

1. Айкен,Х., Хоппер, Г.: „Вычислительная машина, управляемая авто-последовательностью" („The Automatic Sequence Controlled Calculator" цитировано в [9], стр.203-222
2. Беркс,А.В., Беркс, А.Р.: „ENIAC: первый электронный компьютер широкого назначения" („The ENIAC: First General-Purpose Electronic Computer"), Annals of the History of the Computing, том 3, N.4, 1981, стр. 310-399
3. Беркс,А.В., Беркс, А.Р.: „Первый электронный компьютер – история Атанасова" (The First Electronic Computer – The Atanasoff Story), The University of Michigan Press, Анн Арбор,1988
4. Чаудерна,К.-Х.: „Конрад Цузе, Путь к Компьютеру Z3" („Konrad Zuse, der Weg zu seinem Computer Z3"), Oldenborn Verlag, Мюнхен,1979
5. Ваинхарт,К.: „Информатика и автоматика. Путеводитель по немецкому музею"(Informatik und Automatik. Führer durch die Ausstellung im Deutschen Museum). Мюнхен 1990, часть 6.6.4
6. Корен,И.: „Компьютерные Арифметические Алгоритмы" („Computer Arithmetic Algorithms"), Prentice Hall, Englewood Cliffs, NJ(Нью Йорк),1993
7. Лавингтон,С.Х.: "История Манчестерских Компьютеров" („A History of Manchester Computers"), NCC Publications, Манчестер, 1975
8. Лавингтон,С.Х.: „Ранние Британские Компьютеры" (Early British Computers), Digital Press, Манчестер,1980

9. Рандель, Б.: „Происхождение Цифровых Компьютеров“ („The Origins of Digital Computers“), Springer-Verlag, Берлин,1982
10. Рохас,Р.: „Кто изобрел компьютер? Обсуждение с точки зрения компьютерной архитектуры“ („Who invented the computer? The debate from viewpoint of computer architecture“), в В. Гаучи: „Пятьдесят Лет Математики Вычислений“ (Fifty Years Mathimatics of Computation), материалы симпозиума по прикладной математике, AMS,1993, S.361-366
11. Рохас,Р.: „Об основных концепциях первых компьютеров с точки зрения современной компьютерной архитектуры“(„On Basic Concepts of Early Computers in Relation to Contemporary Computer Architectures“), материалы 13-го всемирного компьютерного конгресса, Гамбург, том II, стр.324-331
12. Рохас,Р.: „Условное Разветвление не Необходимо для Универсальных Вычислений в Компьютерах фон Ноймана“ („Conditional Branching is not Necessary for Universal Computation in von Neumann Computers“), опубликовано в журнале компьютерных наук (Journal of Universal Computer Science)
13. Швайер,У., Заупе,Д.: „Функциональный принцип и конструкция программноуправляемой механической вычислительной машины Z1“ („Funktions- und Konstruktionsprinzipien der programmgesteuerten mechanischen Rechenmaschine Z1“), рабочий документ GMD 321, Бонн,1988
14. Штерн,Н.: „От ENIAC’a до UNIVAC’a“(„From ENIAC to UNIVAC“), Digital Press, Бедфорд,1981
15. Цузе,К.: Патент Z-391, Патентное ведомство Германии, Берлин,1941
16. Цузе,К.: „Компьютер - дело моей жизни“ („Der Computer mein Lebenswerk“), Verlag Moderne Industrie, Мюнхен,1970