Вещественная арифметика в ДССП для троичной машины

А.А.Бурцев[1]

[1] ФГУ ФНЦ НИИСИ РАН, Москва, Россия, burtsev@niisi.msk.ru

Аннотация. Созданный в НИЛ троичной информатики на факультете ВМК МГУ программный комплекс ДССП-ТВМ можно использовать в качестве среды разработки и прогона программ для троичного компьютера. Ранее в нём были предусмотрены лишь операции целочисленной арифметики. Теперь для ДССП-ТВМ создан пакет, дополняющий словарь ДССП операциями вещественной арифметики с плавающей точкой.

В статье характеризуются основные возможности представленного пакета, а также поясняются ключевые аспекты его реализации на языке ДССП-Т в интерпретаторе ДССП/ТВМ.

Ключевые слова: троичная симметричная система счисления, троичная машина, троичная логика, троичная арифметика, ДССП, вещественная арифметика с плавающей точкой.

1.Введение

Двоичная микроэлектроника уже исчерпала весь могучий потенциал своих технологических возможностей. В качестве одного из возможных путей дальнейшего развития вычислительной техники предлагается вновь обратиться к троичной симметричной системе счисления (ТССС) с цифрами (–1,0,+1). Перспективность такой системы счисления когда-то отметил ещё Дональд Кнут [1], а её вычислительная эффективность была убедительно подтверждена практической разработкой и применением троичных цифровых машин «Сетунь» и «Сетунь-70», созданных более полувека назад [2] в НИЛ ЭВМ МГУ под руководством Брусенцова Н.П.

О преимуществах троичных машин написано немало книг и статей. Как самим Брусенцовым [3], так и его соратниками и учениками [4,5]. Проявляются они не только в упрощении аппаратной реализации компьютерных блоков (микросхем) и в построении ПО троичного компьютера. Главный выигрыш троичности состоит в том, что она открывает ряд новых возможностей [6], которые привносит в мир вычислительной техники троичная логика и троичная арифметика.

Обычно в списке основных преимуществ, обеспечиваемых троичной (симметричной) системой счисления, отмечают следующие.

1. Троичная – самая экономичная система счисления (т.к. ближе к числу e=2.71828) [7].

2. Тем же количеством разрядов можно кодировать больший диапазон чисел.

3. Можно единообразно обрабатывать как положительные, так и отрицательные числа, для представления которых уже не требуется прибегать к каким-либо «ухищрениям» (например, так называемому дополнительному коду).

4. Возможность непосредственного воплощения логики с тремя значениями: (+1, -1, 0), которые могут быть истолкованы как результат троичного варианта ответа на вопрос: “да”, “нет”, “не знаю” или “может быть”.

5. Упрощённая реализация ряда арифметических операций (сдвига, сравнения чисел, смены знака), а также трёхзначность функции “знак числа”.

6. Оптимальное округление чисел простым отсечением младших разрядов и взаимокомпенсируемость погрешностей округления в процессе вычислений [8]. Особую значимость такое преимущество приобретает при реализации операций вещественной арифметики.

Недавно в преимуществах троичной вычислительной техники убедились и разработчики квантовых компьютеров, опубликовав на просторах Интернета статью с вызывающим заголовком: “будущее квантовых компьютеров — в троичных вычислениях” [9]. Троичность может положительно сказаться и при построении биоморфных нейропроцессоров, поскольку учёные (нейробиологии) сделали важное открытие, что нейрон мозга человека тоже троичен [10].

Всё это даёт серьёзные основания утверждать, что с троичными вычислениями и троичными компьютерами будет связано будущее вычислительной техники. Поэтому проводимые в мире исследования возможных путей дальнейшего развития вычислительной техники не обходят вниманием и вопросы возможной аппаратной реализации троичных элементов на современной элементной базе [11].

В НИЛ Троичной Информатики ВМК МГУ на протяжении ряда лет (с 2010 по 2017 гг.) Масловым С.П. (соратником Брусенцова Н.П.) были проведены исследования, в результате которых были предложены [12] способы реали-

зации на современной элементной базе таких аппаратных троичных элементов, которые могли бы функционировать аналогично тем, что были реализованы в машинах семейства «Сетунь».

С 2011 года в качестве среды разработки и прогона троичных программ (т.е. программ для троичного компьютера) можно использовать программный комплекс ДССП-ТВМ [5], созданный в НИЛ ТИ на факультете ВМК МГУ. Он включает имитатор троичной машины ТВМ и систему разработки программ для нее (кросс-компилятор и интерпретатор) на языке ДССП-Т – троичном варианте языка ДССП. ДССП – Диалоговая Система Структурированного Программирования, созданная в НИЛ ЭВМ МГУ под руководством Брусенцова в 80-х годах XX века и затем реализованная на микрокомпьютерах самых разнообразных архитектур [13].

Ранее в ДССП-ТВМ можно было создавать троичные программы, используя лишь операции целочисленной арифметики. Теперь для ДССП-ТВМ создан пакет, дополняющий словарь ДССП операциями вещественной арифметики с плавающей точкой.

В статье характеризуются основные возможности представленного пакета, а также поясняются ключевые аспекты его реализации на языке ДССП-Т в интерпретаторе ДССП/ТВМ.

2.Общая характеристика пакета вещественной арифметики

Этот пакет загружается в интепретаторе ДССП командой **LOAD Real** :

```
TVM - Ternary virtual machine
tvm (PC=#00000000000000)> go; q;
****** DSSP/TVM *******  v.3d
* LOAD Real
* SAVE DSSP-3dR
```

А команда **SAVE** позволяет сохранить (в файле **DSSP-3dR.nth**) обновлённую версию интерпретатора, уже включающую этот пакет.

Подлежащие обработке вещественные значения размещаются в стеке операндов как особым образом закодированные 27-тритные величины. И все предусмотренные в пакете вещественные операции действуют над величинами, помещёнными в стек.

Двуместные вещественные операции сложения, вычитания, умножения и деления **R+ R- R* R/** принимают из стека два верхних его элемента (вершину и подвершину) в качестве аргументов, а затем результат тоже помещают в стек. А одноместные операции смены знака **RNEG** и получения абсолютной величины **RABS** действуют только над вершиной стека.

Помимо этих основных операций предусмотрены также эффективно реализуемые дополнительные операции умножения на 3, 9, 10: **R*3 R*9 R*10**, а также на их целочисленные степени: **R*3^ R*9^ R*10^**. И аналогичные операции деления на 3, 9, 10: **R/3 R/9 R/10**.

Для сравнения вещественных значений предусмотрен такой же богатый ассортимент операций, как и для сравнения целочисленных значений. Он включает не только привычные операции: **R< R= R> R<= R>=**, дающие двоичный логический ответ (0 или 1). Но и операции, выдающие троичный результат: **RSGN RCMP** :

```
{z} RSGN { = -1 (z<0), 0 (z=0), +1 (z>0) }
{x,y} RCMP { = -1 (x<y), 0 (x=y), +1 (x>y) }
```

Предусмотрены также и полезные операции для двоичной **RSEG** и троичных проверок **RTSEG RTSEG!** попадания точки в сегмент:

```
{a,b,x} RSEG { = +1 if x in[a,b] else 0 }
{a,b,x} RTSEG {=+1 (a<x<b), 0(x=a|x=b), else 0 }
{a,b,x} RTSEG! {= -1(x<a),0(x in[a,b]),+1(b<x) }
```

С помощью операции **mkReal#** вещественное значение **X** с плавающей точкой можно сформировать явно из двух составляющих его величин: мантиссы **m** и порядка, приготовленных в закодированном виде. А из сформированного вещественного значения можно получить его закодированные значения мантиссы и порядка с помощью операций **Rp** и **Rm** :

```
{p,m} mkReal# {X}
{X} Rp {p}     {X} Rm {m}
```

С помощью операции **Int->Real** вещественное значение можно получить из целочисленного. А с помощью операции **Real->Int** можно, наоборот, получить целочисленное значение из целой части вещественного:

```
{N} Int->Real {X}
{X} Real->Int {N}
```

Логическая операция **RNorm?** позволяет проверить, нормализовано ли вещественное число, а операция **RNorm** позволяет, если нужно, привести его к нормализованному виду.

Для обеспечения ввода и вывода вещественных чисел в привычном десятичном виде предусмотрены операция преобразования строки символов в вещественное значение **Str->Real** и операция получения из вещественного числа строки **Real->Str**, его представляющей. Эти операции используются интерпретатором при формировании вещественного значения в стеке при вводе из входного потока изображающей его строки, а также при выдаче на терминал вещественного значения вершины стека в обычном десятичном формате с помощью слов: **.Real .Realp .R .?R** .

В случае невозможности исполнения какой-либо операции предусмотрено возбуждение соответствующей исключительной ситуации.

Например, если операции **Str->Real** не удаётся сформировать вещественное число из заданной строки, то возникает ситуация **WrongReal!**.

Операция **Str->Real** воспринимает вещественное число в любом из следующих привычных форматов его представления (см. табл.1).

Таблица 1.Форматы вещественного числа

Форматы	**Примеры**
±dd.	2. 123. +17. –5 –1234.
±dd.ff	+3.14 0.5 –1234.567
±dd±Epp	1E+6 –1E-7 12E34
±dd.ff±Epp	5.6E+12 –0.1E-9 +12.34E56
± – возможный знак + или – **dd** – десятичные цифры целой части **ff** – десятичные цифры дробной части **pp** – десятичные цифры порядка	

Для некоторых особо употребительных вещественных констант в словарь ДССП пакетом уже занесены представляющие их слова:

```
0.0 1.0 0.1 10.0
Pi 2Pi Pi/2 EXP1 LN2 LN3 LN10
+MAXREAL –MAXREAL
+MINREAL –MINREAL
```

Перед началом использования операций пакета должно вызываться слово **InitReal**.

Вместе с пакетом **Real** библиотека ДССП оснащена дополнительным пакетом **RealF**, в котором представлены типичные часто используемые функции вещественной арифметики. Командой **LOAD RealF** (в интерпретаторе ДССП/ТВМ) можно осуществить загрузку этого дополнительного пакета, вызвав затем командой **InitRealFunc** его инициализацию.

В настоящее время в этом пакете уже определены слова для вычисления экспоненты (**EXP**), натурального логарифма (**LN**), квадратного корня (**SQRT**), синуса (**SIN**), косинуса (**COS**). Предусмотрены также функции возведения произвольного вещественного значения в натуральную степень (**R^**) и эффективного вычисления полинома (**PLNM**), т.е. многочлена с заданными коэффициентами по схеме Горнера.

Для некоторых функций определены слова-дубликаты: **EXPn LNn SQRTn SINn COSn**, в которых дополнительным параметром **n** явно задаётся, сколько итераций осуществить (при исполнении функции) для достижения нужной точности вычислений. При вызове основного слова функции (например,**EXP**) точность вычисления определяется значением, установленным в переменной EPS (по умолчанию равной константе EPS_=1E-6), а количество итераций не превосходит максимального значения, установленного константой MAXRFCnt (=99).

Вычисление функции можно осуществлять в отладочном режиме с печатью важнейших промежуточных значений на каждой итерации, если перед вызовом функции установить (в 1) переменную **RFDbg**. В другой переменной **RFCnt** фиксируется реальное количество итераций, затраченных на исполнение функции.

Если вычислить вызванную функцию не представляется возможным, возбуждаются соответствующие исключительные ситуации. В настоящее время в пакете **RealF** предусмотрены две таких ситуации: 1) ситуация **SQRT<0!** возбуждается, если вызывается функция **SQRT** с отрицательным аргументом; 2) а ситуация **LN<=0!** возбуждается, если вызывается функция **LN** с неположительным аргументом (<=0).

3.Реализация основных операций вещественной арифметики

В настоящее время все операции над троичными вещественными числами в форме с плавающей точкой реализованы в ДССП-ТВМ на основе операций целочисленной арифметики на языке ДССП-Т. Познакомимся подробнее с приёмами реализации основных арифметических операций вещественной арифметики.

3.1.Представление вещественного числа в форме с плавающей точкой

В ДССП для троичной машины вещественное число в форме с плавающей точкой представляется в памяти одним 27-тритном машинным словом, состоящем из 3-х трайтов: в старшем трайте слова (в разрядах 18-26) записывается значение порядка **p**, а в двух младших трайтах (в разрядах 0-17) располагается значение мантиссы **m** (см. рис.1). Значение представленного таким способом троичного вещественного числа **X** формируется как произведение $\mathbf{X}=\mathbf{m}\times 3^{\mathbf{P}}$.

Нормализованное троичное вещественное число – это число, мантисса которого содержит не ноль в первой старшей цифре (17-м разряде). Такая мантисса **m** по абсолютной величине является дробным числом в диапазоне от 1/6 до 1/2, т.е. $1/6 < |\mathbf{m}| < 1/2$.

Обеспечиваемая такой 18-тритной мантиссой точность представления вещественного числа равна 3^{-18} ($\approx 2.58\times 10^{-9}$); и это уже значительно лучше, чем 2^{-24} ($\approx 59.9\times 10^{-9}$) в случае 32-битного формата single float, применяемого в современных двоичных компьютерах.

Используемое 9-тритное значение порядка, варьируемое в пределах [-9841,+9841], позволяет задавать диапазон вещественных чисел (по абсолютной величине) от минимального значения 0.5×3^{-9842} ($\approx 0.22\times 10^{-4695}$) до максимального значения $0.5\times 3^{+9841}$ ($\approx 1.12\times 10^{+4695}$), что значительно перекрывает диапазон вещественных чисел формата single float двоичных машин.

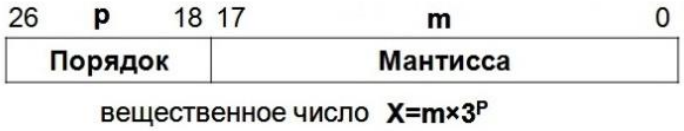

Рис. 1.Схема представления троичного вещественного числа в форме с плавающей точкой

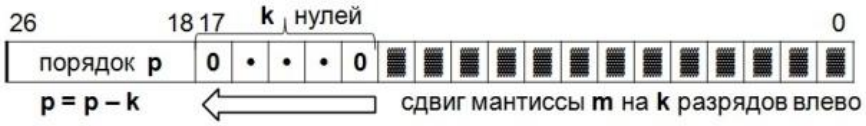

Рис. 2.Схема выполнения операции нормализации вещественных чисел

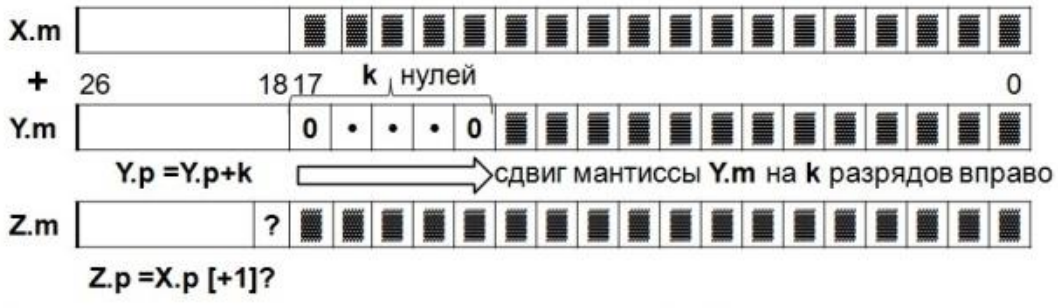

Рис. 3.Схема выполнения операции сложения вещественных чисел

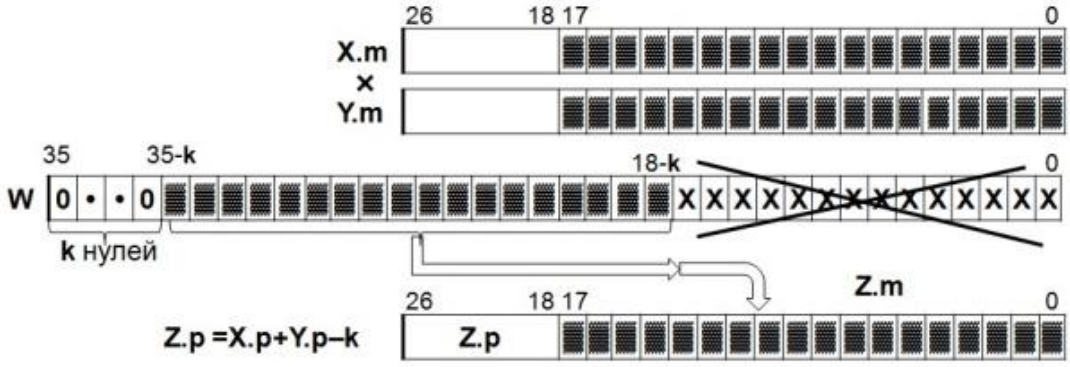

Рис. 4.Схема выполнения операции умножения вещественных чисел

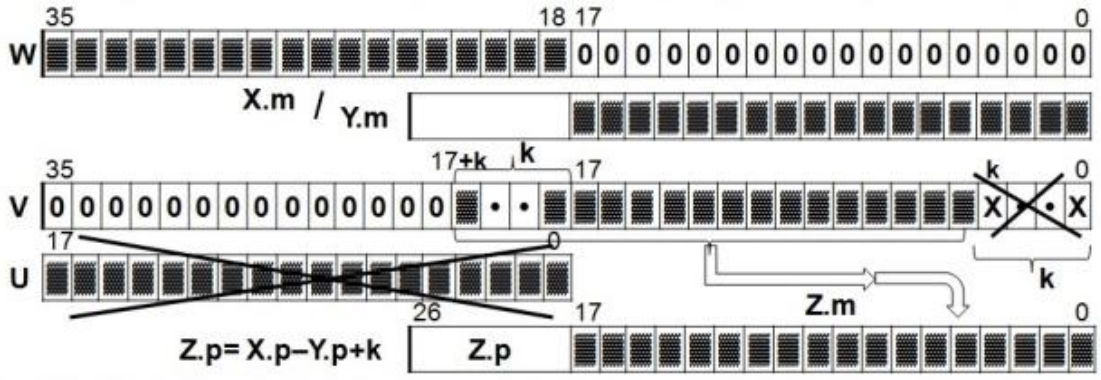

Рис. 5.Схема выполнения операции деления вещественных чисел

Для представления нулевого вещественного числа (0.0) сделано исключение: оно представляется нулевой мантиссой (m=0) и нулевым порядком (p=0), т.е. нулями во всех разрядах.

3.2.Операция нормализации

Далее предполагается, что основные арифметические операции выполняются над вещественными значениями, представленными в нормализованном виде. Для приведения ненулевого вещественного значения к нормализованному виду потребуется совершить ряд действий (см. рис.2).

1. Сдвинуть мантиссу **m** влево на несколько (**k**) разрядов, пока в её старшем (17-ом) разряде не появится ненулевое значение (-1 или +1).

2. И затем уменьшить значение порядка **p** на значение количества **k** передвинутых разрядов.

3.3.Операция сложения и вычитания

Заметим, что когда один из аргументов операции сложения нулевой, тогда другой аргумент просто будет результатом их сложения. Поэтому предполагается, что в описываемой ниже схеме операции сложения участвуют ненулевые нормализованные вещественные значения.

Для сложения двух вещественных значений **X+Y** совершим следующий ряд действий (см. рис.3).

1. Выясним, у какого вещественного значения больше порядок; предположим, что большим будет порядок числа **X: X.p>Y.p**; выясним, на какое значение: **k=X.p–Y.p**.

2. Уравниваем порядки складываемых чисел: **Y.p=Y.p+k.**

3. И сдвигаем мантиссу **Y.m** вправо на **k** разрядов.

4. Теперь складываем мантиссы как 18-тритные целые числа: **Z.m=X.m+Y.m**.

5. Если при этом возник перенос (+1 или -1) в 18-й разряде, то сдвигаем мантиссу **Z.m** вправо на 1 разряд и корректируем порядок: **Z.p=Z.p+1**.

6. Если получено ненормализованное вещественное значение Z (17-й разряд нулевой), то выполняем над ним операцию нормализации.

Для исполнения же операции вычитания двух вещественных значений **X–Y** выполним два действия: сначала изменим знак у 2-го аргумента: **–Y** (для этого инвертируем его мантиссу); а затем уже выполним операцию сложения: **X+(–Y).**

3.4.Операция умножения

Для умножения двух вещественных значений **X×Y** выполним следующий ряд действий (см. рис.4).

1. Складываем их порядки: **Z.p= X.p + Y.p**.

2. Перемножаем их мантиссы как 18-разрядные целые числа: **W= X.m × Y.m**.

3. У полученного 36-разрядного целого числа **W**[35..0] выделяем 18-разрядный участок, который начинается со старшего (**35-k**)-го ненулевого разряда.

4. Принимаем эту выделенную старшую часть **W** в качестве мантиссы результата: **Z.m= W[35-k..18-k]**; а оставшуюся младшую часть **W** отбрасываем.

5. Проведя такую нормализацию **W**, корректируем порядок результата **Z.p= Z.p – k**.

Заметим, что простота округления вещественного числа в ТССС как раз и выражается в том, что младшую часть **W** здесь анализировать не надо! (т.к. её можно просто отбросить!)

3.5.Операция деления

Для деления вещественного значения **X** на ненулевое вещественное значение **Y** выполним следующий ряд действий (см. рис.5).

1. Вычитаем порядки: **Z.p= X.p – Y.p** .

2. Мантиссу **X.m** превращаем в 36-разрядное целое число **W= X.m × 3^{18}**.

3. Делим 36-разрядное целое **W** на мантиссу **X.m** как на 18-разрядное целое число, получая 36-разрядное целое частное **V** и 18-разрядное целое **U** в качестве остатка.

4. У величины **V** выделяем 18-разрядный участок, который начинается со старшего (**17+k**)-го ненулевого разряда, и принимаем эту часть **V** в качестве мантиссы результата: **Z.m= V[17+k.. k]**; оставшуюся часть **V** (как и величину **U**) отбрасываем

5. Проведя такую нормализацию мантиссы, корректируем порядок результата: **Z.p= Z.p + k**.

Заметим, что простота округления вещественного числа в ТССС позволяет просто отбросить величину **U** и младшую часть **V** (т.к. их анализировать не надо!) и тем самым упростить реализацию операции деления.

4.Представление вещественных чисел в виде символьных строк

Познакомившись с тем, в каком закодированном виде вещественные значения хранятся в памяти троичной машины и как над ними исполняются арифметические действия, обратимся теперь к проблемам представления таких значений в символьном виде, удобном для восприятия их человеком.

4.1. Получение значения вещественного числа из строки символов

Операция **Str->Real** позволяет получить значение вещественного числа из строки символов, соответствующей одному из принятых форматов, представленных в таблице 1. Приве-

дём примеры применения этой операции для получения вещественных значений:

```
"123.45E+7" Str->Real .R D CR
"-12345." Str->Real .R D CR
"123.45E-5" Str->Real .R D CR
"-12345," Str->Real .R D CR
```

А также примеры выдачи их на терминал с помощью операции **.R**, которая печатает вершину стека как вещественное число:

```
0.12345E+10
-12345.
0.0012345
WrongReal!"-12345,"-12345.
```

В последней строке выдачи отображена стандартная реакция на исключительную ситуацию **WrongReal!** с печатью «ошибочной» строки, вызвавшей исключение.

Введение в язык ДССП-Т операций вещественной арифметики влечёт за собой необходимость распознавать в тексте ДССП-программы вещественные константы-литералы. Значит, интерпретатор и кросс-компилятор ДССП следует «научить» понимать вещественные числа, встречающиеся в программе, как константы-литералы и формировать для них 27-тритный код, состоящий из мантиссы и порядка.

В ходе разбора символьной строки вида «**±dd.ff±Epp**» для формирования вещественного значения приходится выполнять операции сложения, умножения на 10 (для целой части) и деления на 10 (для дробной части) над троичными вещественными значениями. Интерпретатор ДССП/ТВМ с загруженным пакетом **Real** становится способным исполнять такие операции. Поэтому в интерпретаторе можно реализовать операцию преобразования **Str->Real**.

Но ещё надо «надоумить» интерпретатор вызывать эту операцию всякий раз, когда ему приходится разбирать введённую строку, анализируя, представляет ли она собой константу-литерал. Прежний алгоритм интерпретатора при разборе введённой строки-литерала не учитывал, что могут встретиться строки, представляющие вещественные константы. Как можно теперь «научить» интерпретатор правильно понимать и обрабатывать такие строки?

Для решения такой проблемы пришлось модифицировать алгоритм разбора констант-литералов интерпретатора следующим образом. В блок распознавания литералов интерпретатора добавлена возможность вызывать свою оригинальную процедуру распознавания строки-константы. Так чтобы саму подобную процедуру стало возможным задавать позднее непосредственно в ДССП-программе, обрабатываемой интерпретатором.

Продемонстрируем, как такая процедура назначается при инициализации пакета **Real** :

```
: GetRealVal{Adr,len}
   EON WrongReal! WrongRealCode!
   Str->Real +1 {RVal,Code=+1};
: WrongRealCode! {RealVal,Code}
   T-1 {RVal, Code=-1} ;
: InitGetRealVal {вызывается в InitReal}
 '' GetRealVal !GetSpecValueProc ;
```

Теперь (после такого назначения) интерпретатор будет вызывать процедуру **GetRealVal** всякий раз, когда он не сумеет во введённой строке распознать какую-либо константу-литерал прежними способами. Если процедура **GetRealVal**, вызвав операцию **Str->Real**, распознает в заданной строке вещественную константу, она вернёт сформированное для неё 27-тритное значение (RealVal) и успешный код завершения (Code=+1). Если не удастся распознать в строке вещественное число, то возникнет ситуация **WrongReal!**, в результате обработки которой будет установлен отрицательный код завершения (Code=-1), означающий, что распознать корректное вещественное число в заданной строке, увы, не удалось.

Таким образом, интерпретатор ДССП/ТВМ удаётся «научить» понимать вещественные константы. А вот «обучить» этому свойству кросс-компилятор ДССП-ТВМ, оказывается, не так-то просто. Ведь компилятор должен не только провести синтаксический разбор заданной строки, представляющей вещественную константу, но и получить её троичное представление в форме 27-тритного значения, складывающегося из закодированных троичных значений мантиссы и порядка. А для этого компилятор должен уметь исполнять вещественные операции сложения, умножения и деления с троичными числами. Понятно, что кросс-компилятор, функционирующий на двоичной машине, этих операций, увы, сделать не сможет. Получается, чтобы решить обозначенную проблему, компилятор надо снабдить собственным имитатором троичных операций вещественной арифметики. А это задача сама по себе весьма трудоёмкая.

4.2. Получение из вещественного числа символьной строки для печати

Чтобы полученные в результате вычислений вещественные значения представить в визуальном виде, необходимо уметь выводить их на печать (например, на экран терминала). В простейшем случае вещественное значение вершины стека можно просто изобразить в виде двух троичных или 9-тичных значений мантиссы и порядка, что обеспечивается операцией **.Real#.** Например, фрагмент ДССП-программы:

```
."3.=" 3. .Real# CR
."-27.=" -27. .Real# CR
```

```
 ."-1/9=" 1. 9. R/ .Real# CR
 ."-1/81=" -1. 81. R/ .Real# CR
```

выдаст на печать:

```
3.=0.#300000000E#2
-27.=0.#600000000E#4
1/9=0.#300000000E#8
-1/81=0.#600000000E#6
```

Однако, желательно всё же изображать вещественные значения на печати в традиционной десятичной форме их записи. Поэтому в пакете **Real** предусмотрена операция преобразования **Real->Str**, которая позволяет получить строку с изображением вещественного значения в десятичном виде в формате с фиксированной или плавающей точкой (как в табл.1):

{X,p,Str} **Real->Str** {Str,Len} **TOS**

При этом параметр p задаёт количество значащих цифр в десятичной мантиссе. Если p>0, то число (если это возможно) изображается в форме с фиксированной точкой (и без незначащих нулей в дробной части). Если p<0, то число изображается обязательно в форме с плавающей точкой. А при p=0 значение p берётся по умолчанию (=9).

Сформированную такой операцией строку {Str,Len} можно сразу выдавать на печать командой **TOS**. Именно так и исполняется основная команда печати вещественного значения вершины стека **.Realp** :

```
: .Realp {Y,p} 'RealStr Real->Str TOS { };
: .Real {Y} 0 .Realp ;
: .R C .Real ;
: .?R .? SP .R ;
```

Остальные команды печати используют её с параметром p=0. Команда **.R** оставляет печатаемое значение в вершине стека. А команда **.?R** дополняет действие команды **.?**, изображающей вершину стека в различных целочисленных форматах (троичном, 9-ном, 10-ном и 16-ном), после чего распечатывает вершину стека ещё и как 10-ное вещественное число.

Приведём примеры их использования:

```
12345.67  7 .Realp CR
12345.67 -7 .Realp CR
12345.67 .Real CR
```

вместе с результатами выдачи на печать, которые характеризуют различие этих команд:

```
12345.67
0.1234567E+5
12345.6702
```

Заметим, что не всякое вещественное число, записанное в десятичной системе счисления (например, числа 0.5 и 0.1), может быть точно представлено в троичном коде. И наоборот, некоторые числа, точно представимые в троичном коде (например, 1/3 или 1/9), невозможно точно изобразить вещественным десятичным числом. Так что в результате исполнения вычислений:

```
."1/3=" 1. 3. R/ .R CR
."1/9=" 1. 9. R/ .R CR
```

получим такую выдачу на печать:

```
1/3=0.333333333
1/9=0.111111111
```

При реализации операции преобразования **Real->Str** возникает необходимость решить следующую задачу: как из троичного вещественного числа $\mathbf{X=m\times3^p}$ получить его десятичное представление в виде: $\mathbf{X=z\times10^q}$ такое, чтобы мантисса **z** оказалась по абсолютной величине в диапазоне: $0.1\le|\mathbf{z}|<1.0$. В результате надо получить значение мантиссы $\mathbf{z=X/10^q}$ и само значение порядка **q**.

Для решения этой задачи можно предложить такой алгоритм действий. При p>0 многократно (в цикле) делить X на 10., пока не станет |z|<1.0, подсчитывая при этом, сколько раз (**q**) было исполнено такое деление. А при p<0, наоборот, умножать X на 10., пока не станет $|\mathbf{z}|\ge0.1$, и подсчитывать при этом отрицательное значение порядка **q**.

Но при больших значениях порядка **p** такой алгоритм будет работать слишком медленно. Для его ускорения можно предложить предварительно скорректировать значение X, поделив (или умножив) его сразу на величину $\mathbf{10^t}$, взяв **t=p**/2 (**p=2t+j**). Тогда в качестве первого приближения для **z** можно получить величину $\mathbf{z=m\times3^j\times9^t/10^t}$ и **q=t**. А далее уже продолжить действия по описанному алгоритму.

5.Примеры применения пакета вещественной арифметики

Проиллюстрируем применение операций вещественной арифметики примерами разработки некоторых программ в ДССП-ТВМ.

5.1. Пример реализации стандартной функции EXP

Приведённый ниже фрагмент программного кода (на языке ДССП-Т) определяет слово **EXPn**, которое реализует функцию вычисления экспоненты в виде ряда:

$$\sum_{k=0}^{n}\frac{x^k}{k!}=1+\frac{x}{1!}+\frac{x^2}{2!}+\cdots+\frac{x^n}{n!}$$

Вычисление ряда заканчивается при достижении заданного (n) максимального количества слагаемых или возможно раньше, когда очередной член ряда, добавляемый в накапливаемую сумму (Sk), уже оказывается меньше (по абсолютной величине) величины желаемой точности, предварительно заданной словом EPS.

```
: EXP {X} MAXRFCnt EXPn {exp(X)} ;
: EXPn {X,n} 1.0 C C E4 0 E2
   {X,Fk=1!=1.0,Xk=1.0,Sk=1.0,k=0,n}
  DO- EXPk ! RFCnt E4 DDD {Res=Sk};
: EXPk {X,Fk=k!,Xk=X^k,Sk,k,i} E2 1+ E2
  5 ET C2 Int->Real R* 5 ET E4
  6 CT R* E4 C4 6 CT R/ E4 C4 R+
  E4 C EPS? BR+ EPS! D ;
: EPS? {Z} RABS EPS R<= {|Z|<=EPS?} ;
: EPS! {i,Z} DD 0 {i=0} ;
```

А словом **EXP** функция **EXPn** вызывается с максимально предусмотренным параметром **n**, задаваемым словом-константой MAXRFCnt.

Примеры вызовов функций **EXP** и **EXPn** :

```
." EXP(1.)= " 1.0 EXP .R CR
." EXP(-1.)= " -1.0 EXP .R CR
." EXP(-2.)= " -2.0 9 EXPn .R CR
." EXP(3.)= " 3. 20 EXPn .R D .RFCnt CR
." EXP(5.0)= " 5.0 EXP .R D .RFCnt CR
```

В итоге этих вызовов на экран терминала выдаются следующие результаты:

```
EXP(1.)= 2.7182818
EXP(-1.)= 0.36787919
EXP(-2.)= 0.135097
EXP(3.0)= 20.085537 [17]
EXP(5.0)= 148.41316 [23]
```

Заметим, что при вызове слова .RFCnt здесь печатается (в квадратных скобках) количество слагаемых, которое было учтено в итоговой сумме при вычислении ряда функции.

5.2. Пример процедуры поиска корня уравнения

Продемонстрируем применение операций вещественной арифметики для реализации в ДССП-ТВМ процедуры поиска единственного корня уравнения F(X)=0 на отрезке [a,b] простым методом деления отрезка пополам. При вызове этой процедуры **EQUROOT** предусмотрена проверка, что на концах этого отрезка функция принимает значения разных знаков. Если это условие нарушено, то вызывается исключительная ситуация **NoRoot!**, означающая, что вызванная процедура не сможет найти требуемый корень на заданном отрезке.

```
SITUATION NoRoot! .NoRoot{Нет корня!}
ACT VAR F {указатель на тело функции F(X) }
: EQUROOT_{a,b} GTP {a,b,F} EQUROOT {X} ;
: EQUROOT {a,b,F} ! F C2 F C2 F
  isRoot? BR+ FindRoot NoRoot! {X};
: isRoot? {a,b,Fa,Fb} C2 RSGN C2 RSGN
   NEG = {(sign(Fa)=-sign(Fb)?} ;
: FindRoot {a,b,Fa,Fb} C2 C2 R>
    {Fa>Fb?} IF+ ba {swap [a,b]}
  {a,b,Fa,Fb} getFcRoot? DW new[a,b]
    D 5 ET DDDD {X=c} ;
: ba{a,b,Fa,Fb} E2 E3 E4 E3 {b,a,Fb,Fa};
: getFc {a,b,Fa,Fb} C4 C4 R+ 2. R/
  C F {..,c=a+b)/2,Fc=F(c)} ;
: getFcRoot? getFc {a,b,Fa,Fb,c,Fc} 6 CT
 6 CT =?EPS C2 EPS? OR NOT {..,~Lv} ;
   { Lv= (|a-b|<=EPS?) or (|Fc|<=EPS?) }
: =?EPS {a,b} R- RABS EPS R<= ;
: new[a,b]{a,b,Fa,Fb,c,Fc} C RSGN
  BR+ [a,c] [c,b] {a',b',Fa',Fb'} ;
: [a,c] E3 D E4 D {a,b=c,Fa,Fb=Fc} ;
: [c,b] E4 D 5 ET D {a=c,b,Fa=Fc,Fb} ;
```

В переменной F запоминается указатель на тело заданной функции. Предусмотрена также и другая процедура **EQUROOT_**, при вызове которой имя-указатель функции можно передавать не в стеке, а задавать тут же в теле на месте вызова этой процедуры.

Применим эти процедуры для решения двух трансцендентных уравнений:

(1) $\sin x = x^2 - 1$ на отрезке $[0,\pi]$

(2) $e^x = x + 2$ на отрезке [1,2]

Для этого определим слова F1 и F2, задающие вычисление соответствующих функций:

```
{ F1(X)= sin(X) - X^2 + 1 }
: F1{X} C SIN E2 C R* R- 1. R+ ;
{ F2(X)= exp(X) - X- 2 }
: F2{X} C EXP E2 R- 2. R- ;
```

И осуществим с ними вызов рассмотренных процедур поиска корня:

```
."root for sin(X)-X^2+1.=0 in[0.,Pi]=" CR
 SP 0. Pi EQUROOT_ F1 .R CR
."root for exp(X)-X-2.=0 in[1.,2.]=" CR
 SP 1. 2. '' F2 EQUROOT .R CR
```

В итоге этих вызовов получим такой вот результат выдачи на экран терминала:

```
root for sin(X)-X^2+1=0 in[0.,Pi]=
 1.409624
root for exp(X)-X-2=0 in[1.,2.]=
 1.14619325
```

Проверочные вызовы функций с этими значениями корней выдают нулевые значения.

6.Заключение

В статье представлены основные возможности пакета операций вещественной арифметики для ДССП троичной машины (ТВМ). Пояснены важнейшие аспекты его реализации на языке ДССП-Т в интерпретаторе ДССП/ТВМ. Продемонстрированы примеры его применения для построения типовых вычислительных программ в интерпретаторе ДССП/ТВМ.

Таким образом, теперь для троичной машины в ДССП/ТВМ стало возможным создавать вычислительные программы с использованием операций вещественной арифметики.

Публикация выполнена в рамках государст-

венного задания ФГУ ФНЦ НИИСИ РАН «Проведение фундаментальных научных исследований (47 ГП)» по теме № FNEF-2021-0004 «Разработка архитектуры, системных решений и методов для создания микропроцессорных ядер и коммуникационных средств семейства систем на кристалле двойного назначения. 0580-2021-0004», Рег. № 121031300049-0.

Real arithmetic in DSSP for ternary machine

A.A. Burtsev

Abstract. The DSSP-TVM software complex created at the Research Laboratory of Ternary Informatics at the Faculty of Computational Mathematics and Cybernetics of Moscow State University can be used as an environment for developing and running programs for a ternary computer. Previously, it only provided integer arithmetic operations. Now for DSPP-TVM a package has been created that supplements the DSSP dictionary with operations of real floating point arithmetic.

The article describes the main features of the presented package, and also explains the key aspects of its implementation in the DSPP-T language in the DSPP/TVM interpreter.

Keywords: ternary symmetric number system, ternary machine, ternary logic, ternary arithmetic, DSSP, floating point real arithmetic.

Литература

1. Кнут Д. Искусство программирования на ЭВМ. Т.2 Получисленные алгоритмы. М., Мир, 1977, п.4.1, 216-219.

2. Н.П. Брусенцов Н.П., Х. Рамиль Альварес. Троичные ЭВМ “Сетунь” и “Сетунь 70”. «Первая Международная конференция "Развитие вычислительной техники в России и странах бывшего СССР: история и перспективы" SORUCOM-2006», Россия, Петрозаводск, 3-7 июля 2006. Петрозаводск, Изд-во ПетрГУ, 2006, ч.1, 45-51.

3. Брусенцов Н.П. Заметки о троичной цифровой технике. «Вычислительная техника и вопросы кибернетики», Т.15 (1978), 145–155.

4. Владимирова Ю.С. Введение в троичную информатику: учебное пособие. М., АРГАМАК-МЕДИА; 2015.

5. Бурцев А. А., Сидоров С. А. Троичная виртуальная машина и троичная ДССП. «Программные системы: теория и приложения» Т.6 (2015), №4, 29–97, http://psta.psiras.ru/read/psta2015_4_29-97.pdf.

6. Бурцев А. А. Особенности программирования троичной машины: новые возможности и новые задачи. «Труды НИИСИ РАН», Т. 10 (2020), № 3, 49–60.

7. А.А. Бурцев, В.А. Бурцев. О преимуществах троичных машин и эффективности троичных вычислений. «Труды НИИСИ РАН», Т. 10 (2020), № 3, 60–65.

8. Ким Г.Д., Воеводин В.В. Машинные операции с точки зрения математика. «Вычислительные методы и программирование», Т. 26 (1977), 31-35.

9. Будущее квантовых компьютеров — в троичных вычислениях, http://www.infuture.ru/news.php?news_id=475pdf.

10. Brain circuitry findings could shape computer design, http://cbcl.mit.edu/news/files/liu-tp-picower.html

11. Zarin Tasnim Sandhie, Jill Arvindbhai Patel, Farid Uddin Ahmed, Masud H. Chowdhury. Investigation of Multiple-valued Logic Technologies for Beyond-binary Era. ACM Comput. Surv. 54, 1, Article 16 (January 2021), 30 pages, https://doi.org/10.1145/3431230.

12. Маслов С.П. Об одной возможности реализации троичных цифровых устройств. «Программные системы и инструменты» Т.12 (2011), 222-227.

13. Бурцев А.А., Сидоров С.А. История создания и развития ДССП: от "Сетуни-70" до троичной виртуальной машины. «Вторая Международная конференция “Развитие вычислительной техники и её программного обеспечения в России и странах бывшего СССР” SORUCOM-2011», Россия, В.Новгород, 12-16 сентября 2011. В.Новгород, Изд-во НовГУ, 2011, 83-88.