Комплексная 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. Первоначально в нём были предусмотрены лишь операции целочисленной арифметики. Затем ДССП-ТВМ был дополнен пакетом операций вещественной арифметики с плавающей точкой. Теперь на его основе создан пакет комплексной арифметики. В статье характеризуются основные возможности предлагаемого пакета, а также поясняются ключевые аспекты его реализации на языке ДССП-Т в среде интерпретатора ДССП/ТВМ.

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

1.Введение

В настоящее время, когда двоичная микроэлектроника приблизилась к потолку своих потенциальных возможностей, в качестве одного из перспективных путей дальнейшего развития вычислительной техники исследуется возможность построения цифровых машин на основе троичной симметричной системы счисления (ТССС с цифрами –1,0,+1). Перспективность такого направления была в своё время отмечена ещё Дональдом Кнутом [1], а реальная возможность его осуществления была обозначена практической разработкой и применением троичных цифровых машин «Сетунь» и «Сетунь-70» [2], созданных более полувека назад в НИЛ ЭВМ МГУ под руководством Брусенцова Н.П.

Вопросы построения троичной вычислительной техники по-прежнему являются предметом проводимых научных исследований. Так, в работах Маслова С.П. [3], одного из сотрудников лаборатории Брусенцова Н.П., были предложены способы реализации на современной элементной базе таких аппаратных троичных элементов, которые могли бы функционировать аналогично тем, что были реализованы в машинах семейства «Сетунь». Это вселяет весомую надежду на создание в скором будущем современного троичного компьютера.

В числе достоинств цифровых машин, построенных на основе троичной симметричной системы счисления, обычно отмечают [4-6] следующие её особенности. Троичная – самая экономичная система счисления (подробнее это объясняется, например, в [6]). В ней можно тем же количеством разрядов закодировать больший диапазон чисел и проще реализовать ряд арифметических операций. Можно единообразно обрабатывать как положительные, так и отрицательные числа, и непосредственно реализовать операции троичной логики.

Но для области вычислительной обработки самое ценное качество этой системы счисления проявляется в том, что она позволяет сделать оптимальное округление чисел простым отсечением младших разрядов, а также обеспечивает взаимокомпенсируемость погрешностей округления в процессе вычислений.

Поэтому на троичном компьютере можно выполнять вычисления с лучшей точностью и избавиться, наконец, от назойливых проблем округления, которые были навязаны миру вычислительной техники применением двоичных машин. Такой вывод был сделан специалистами ВЦ МГУ [7] ещё в те годы XX века, когда им предоставлялась возможность решать вычислительные задачи как на двоичных, так и на троичных компьютерах.

В настоящее время в качестве среды разработки и прогона программ для троичного компьютера можно использовать программный комплекс ДССП-ТВМ [8], созданный в НИЛ троичной информатики МГУ. Он включает имитатор троичной машины ТВМ, кросс-компилятор и интерпретатор, позволяя разрабатывать программы для ТВМ на языке ДССП-Т – троичном варианте языка ДССП [9].

Первоначально в ДССП-ТВМ можно было создавать троичные программы, используя лишь операции целочисленной арифметики. Затем ДССП-ТВМ была дополнена пакетом операций вещественной арифметики с плавающей точкой [10]. А на его основе теперь создан пакет комплексной арифметики.

В итоге в настоящее время в ДССП-ТВМ обеспечена возможность разрабатывать для троичной машины и прогонять на ней вычислительные программы, в которых требуется обрабатывать как вещественные, так и комплексные числа.

В статье характеризуются основные возможности предлагаемого пакета комплексной арифметики, а также поясняются важнейшие аспекты его реализации на языке ДССП-Т в среде интерпретатора ДССП/ТВМ.

2.Общая характеристика пакета комплексной арифметики

Данный пакет комплексной арифметики предлагается использовать в среде интепретатора ДССП-ТВМ, который уже оснащён операциями вещественной арифметики. После загрузки этого пакета командой **LOAD Cmplx** (из файла **Cmplx.dsp**) будет подготовлена новая среда интерпретатора с операциями комплексной арифметики, версию которой затем можно сохранить командой **SAVE** (в файле **DSSP-3dC.nth**) :

```
****** DSSP/TVM *******  v.3d
* LOAD Cmplx
* SAVE DSSP-3dC
*
```

Подлежащие обработке значения комплексных чисел занимают в стеке две позиции: одна из них представляет вещественное значение реальной части комплексного числа, а другая – мнимой. А каждое из этих вещественных значений, в свою очередь, представляется в 27-тритном машинном слове как вещественное число в форме с плавающей точкой. При этом значение порядка помещается в его старшем трайте (в разрядах с 18-го по 26-й), а значение мантиссы располагается в двух младших трайтах (в разрядах с 0-го по 17-й) такого троичного слова (подробнее об этом см. в п.3.1 в [10]).

Все предусмотренные в пакете операции с комплексными числами действуют над величинами, помещёнными в стек. При загрузке комплексного значения в стек его реальная часть помещается в подвершину, а мнимая часть – в вершину стека.

Поэтому одноместные операции комплексной арифметики затрагивают две верхние позиции стека, двухместные операции – четыре, а операции, требующие 3 аргумента, действуют соответственно над шестью верхними позициями стека.

Чтобы было удобно копировать и переставлять в стеке комплексные числа, в пакете предусмотрены соответствующие дополнительные операции, позволяющие обрабатывать позиции стека парами. Представляя верхние позиции стека так, как будто они содержат одно, два или три комплексных числа, операции **CC CC2 CC3** позволяют скопировать наверх стека комплексное значение, лежащее в нём на глубине соответственно 1, 2 или 3:

```
{Re,Im} CC { Re,Im, Re,Im }
{Re,Im,a,b} CC2 {Re,Im,a,b,Re,Im}
{Re,Im,a,b,c,d} CC3 {Re,Im,a,b,c,d,Re,Im}
```

А операции **CE2** и **CE3** позволяют переставить в стеке комплексные числа, лежащие на глубине 1-2 и 1-3 соответственно:

```
{XRe,XIm,YRe,YIm } CE2 {YRe,YIm,XRe,XIm}
{XRe,XIm,a,b,YRe,YIm} CE3
                      {YRe,YIm,a,b,XRe,XIm}
```

Комплексное значение состоит из пары вещественных и занимает в стеке две 27-тритных позиции. И для занесения такого парного значения в стек из памяти по указанному адресу (A), а также для записи такой пары из стека в память тоже предусмотрены аналогичные дополнительные операции **@C !C** :

```
{A} @C {Re,Im}  { Re=M[A], Im=M[A+3] }
{Re,Im,A} !C { } { M[A]=Re, M[A+3]=Im }
```

В пакете представлены две одноместные операции над комплексным числом. Операция, обозначаемая словом **CNEG**, изменяет знак числа, а операция **C'** получает из него сопряжённое комплексное значение:

```
{XRe,XIm} CNEG {–XRe,–XIm}
{XRe,XIm} C' {XRe,–XIm}
```

Для сложения и вычитания двух комплексных чисел предусмотрены двухместные операции, обозначаемые словами **C+** и **C–** :

```
{XRe,XIm,YRe,YIm} C+ {XRe+YRe,XIm+YIm}
{XRe,XIm,YRe,YIm} C- {XRe–YRe,XIm–YIm}
```

Операция умножения двух комплексных чисел осуществляется словом **C*** :

```
{XRe,XIm,YRe,YIm} C*
{XRe*YRe–XIm*YIm, XRe*YIm+XIm*YRe}
```

Предусмотрены также операции, сочетающие умножение двух комплексных чисел вместе со сложением (или вычитанием) получаемого результата с третьим заданным числом. Они обозначаются словами **C*+** и **C*–** :

```
{ZRe,ZIm, XRe,XIm,YRe,YIm} C*+
{ ZRe + (XRe*YRe–XIm*YIm),
  ZIm + (XRe*YIm+XIm*YRe) }
{ZRe,ZIm, XRe,XIm,YRe,YIm } C*-
{ ZRe – (XRe*YRe–XIm*YIm) ,
  ZIm – (XRe*YIm+XIm*YRe) }
```

Заметим, что эти операции используют три помещённых в стек комплексных значения, оставляя вместо них в стеке лишь одно комплексное значение в качестве результата.

В пакет включена и самая нагруженная операция комплексной арифметики, совмещающая

в себе умножение со сложением и вычитанием комплексных чисел, которая одна реализует сразу обе рассмотренные выше операции. Она обозначается словом **C*+-** :

```
{ZRe,ZIm, XRe,XIm,YRe,YIm} C*+-
{ ZRe + (XRe*YRe–XIm*YIm) ,
  ZIm + (XRe*YIm+XIm*YRe) , }
{ ZRe – (XRe*YRe–XIm*YIm) ,
  ZIm – (XRe*YIm+XIm*YRe) }
```

Эта операция тоже использует три помещённых в стек комплексных значения, но оставляет вместо них в стеке не одно, а два комплексных значения в качестве результата.

Заметим, что вызовом слова **C*+-** можно исполнить одну из важнейших базовых операций, применяемых в задачах обработки сигналов, которая получила образное название «бабочка Фурье».

В пакете имеются также операции, в которых наряду с комплексными числами в качестве параметров участвуют и вещественные числа. Так, для умножения обеих компонентов комплексного числа на вещественное значение предназначена операция **C*S** :

```
{ XRe,XIm,S } C*S { XRe*S,XIm*S }
```

Операция, обозначаемая словом **CMOD^2**, позволяет получить вещественное значение, являющееся квадратом модуля заданного комплексного числа:

```
{ XRe,XIm } CMOD^2 { XRe^2+XIm^2 }
```

А операция, обозначаемая словом **|C|**, позволяет получить сумму абсолютных значений его реальной и мнимой частей:

```
{ XRe,XIm } |C| { |XRe|+|XIm| }
```

Для обеспечения ввода и вывода комплексных чисел в привычном десятичном виде предусмотрены операция **Str->Cmplx** для преобразования строки символов в комплексное значение (состоящее из двух вещественных), а также ряд операций для печати комплексного значения: **.C**, **.Cmplx** (с удалением его из стека) и **.Cmplxp** (с указанием числа позиций):

```
{Str,Len} Str->Cmplx {Re,Im}
{Re,Im} .C {Re,Im}
{Re,Im} .Cmplx { }
{Re,Im,p} .Cmplxp { }
```

При этом строка, изображающая комплексное число, представляется в привычном формате (см. табл. 1) в виде двух вещественных чисел, соединённых знаком + или –, второе из которых завершается символом мнимой единицы (**i** или **I**). Вещественное число реальной части может отсутствовать (тогда оно полагается равным нулю 0.). А при отсутствии числа в мнимой части (перед символом мнимой единицы), оно полагается равным единице (1.0).

В случае невозможности преобразования заданной строки в комплексное число (при исполнении операции **Str->Cmplx**), возбуждается исключительная ситуация **WrongCmplx!**.

Таблица 1.Форматы комплексного числа

Форматы	**Примеры**
±R±Ri	`3.-2i 1E-2+1.i` `-5.4+3.2I 1.2+1E+3i`
±Ri	`+3.i -1E-3i`
±R±i	`1.+i -2E+3-I`
±i	`+I -I +i -i I i`
± – возможный знак + или – **R** – вещественное число (без знака) **i** – обозначение мнимой единицы: `i` или `I`	

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

```
i {0.,1.0}    -i {0.,-1.0}
1+0i {1.,0.} -1+0i {-1.,0.}
```

Для обеспечения возможности определения именованных комплексных констант создано компилирующее слово **DVALUE**, позволяющее задать слово-константу, при вызове которого в стек посылается два представляющих его 27-тритных значения.

Например, в результате такого определения:

```
1. -1. DVALUE 1-i
```

в словарь ДССП добавляется слово **1-i**, вызов которого:

```
1-i {+1.0,–1.0}
```

заносит в стек два вещественные значения 1.0 (в подвершину) и –1.0 (в вершину), представляющие комплексное значение 1.0–1.0i :

В пакете также определяется новый тип данных **COMPLEX**, что позволяет объявлять в ДССП-программе переменные и массивы для работы с комплексными числами.

Представим примеры подобных объявлений:

```
COMPLEX VAR X   COMPLEX VAR Y
32 COMPLEX VCTR V { V[0:32] }
```

И продемонстрируем типовые действия с переменными и векторами комплексного типа:

```
X {XRe,XIm} ! Y {Y:=X}
1 V {V[1].Re,V[1].Im} 31 ! V { V[31]:=V[1]}
```

В первом примере вызов имени переменной **X** типа **COMPLEX** засылает в стек два вещественных значения XRe и XIm, представляющих реальную и мнимую часть её комплексного значения. А командой, обозначаемой словом **!**, эти два вещественных значения помещаются в реальную и мнимую часть переменной **Y**. А во втором примере комплексное значение, взятое из элемента вектора **V**`[1]`, записывается как новое значение для элемента **V**`[31]`.

3. Описание реализации пакета комплексной арифметики

Рассмотрим поподробнее ключевые аспекты реализации представленного пакета комплексной арифметики.

3.1. Реализация основных операций комплексной арифметики

В настоящее время все операции над троичными комплексными числами реализованы в ДССП-ТВМ на основе операций вещественной арифметики на языке ДССП-Т. Продемонстрируем приёмы реализации основных операций комплексной арифметики.

Реализация одноместной операции **C'** получения сопряжённого комплексного значения заключается в смене знака вещественного значения лишь его мнимой части, а для реализации операции смены знака **CNEG** требуется поменять знак у обеих вещественных компонент комплексного значения:

```
: C' {Re,Im} RNEG {Re,–Im} ;
: CNEG {Re,Im} RNEG {Re,–Im}
      {Re,–Im} E2 RNEG E2 {–Re,–Im} ;
```

Реализация одноместных операций **CMOD^2** и **|C|** сводится к вычислению одного вещественного значения с применением операций **R+ RABS** и **R*** над вещественными значениями его реальной и мнимой частей:

```
: CMOD^2 {Re,Im} C R*
        E2 C R* R+ {Re^2+Im^2} ;
: |C| {Re,Im} RABS
        E2 RABS R+ {|XRe|+|XIm|} ;
```

Реализация двухместных операций сложения **C+** и вычитания **C–** заключается в исполнении вещественных операций сложения **R+** и вычитания **R–** по отдельности над их вещественными компонентами:

```
: C+ {XRe,XIm,YRe,YIm} E4 R+
     E3 R+ {XRe+YRe,XIm+YIm} ;
: C– {XRe,XIm,YRe,YIm} E4 E2 R–
     E3 R– {XRe–YRe,XIm–YIm} ;
```

Умножение комплексного значения на вещественное значение осуществляется применением вещественной операции **R*** для умножения на это значение каждой из его вещественных компонент:

```
: C*S {Re,Im,S} E3 C3 R* E3
     {Re*S,Im,S} R* {Re*S,Im*S} ;
```

А вот при реализации операции комплексного умножения приходится уже воспользоваться всеми тремя вещественными операциями умножения **R***, сложения **R+** и вычитания **R–**, чтобы получить итоговые вещественные значения реальной и мнимой частей результата комплексного умножения по известным формулам:

```
: C* {XRe,XIm,YRe,YIm} C4 C3 R*
   C4 C3 R* R– {.., XRe*YRe–XIm*YIm }
   5 ET R* E3 R* R+
{XRe*YRe–XIm*YIm, XRe*YIm+XIm*YRe} ;
```

На основе этих базовых операций комплексной арифметики (уже реализованных **C+ C– C***) можно далее реализовать более сложные комплексные операции умножения с накоплением:

```
{ Z=(Z.Re,Z.Im),X=(X.Re,X.Im),Y=(Y.Re,Y.Im) }
: C*+ {Z,X,Y} C* C+ {Z'=Z+X*Y} ;
: C*– {Z,X,Y} C* C– {Z'=Z–X*Y} ;
: C*+– {Z,X,Y} C* CE2 {X*Y,Z}
   CC2 CC2 C+ CE3 C– {Z+X*Y,Z–X*Y} ;
```

Заметим, что здесь используются также операции копирования и перестановки в стеке комплексных значений, занимающих в стеке пару позиций. Продемонстрируем приёмы реализации таких операций над парами позиций стека:

```
: CC2 {Re,Im,a,b} C4 C4 {Re,Im,a,b,Re,Im} ;
: CE2 {XRe,XIm,YRe,YIm} E3
      E2 E4 E2 {YRe,YIm,XRe,XIm} ;
: CE3 {XRe,XIm,a,b,YRe,YIm} E2 6 ET
      E2 5 ET {YRe,YIm,a,b,XRe,XIm} ;
```

А также представим реализацию операций чтения комплексных значений из памяти в стек и записи из стека в память:

```
: @C {Adr} C @W E2 3+ @W {Re,Im} ;
        { Re=Mem[Adr], Im=Mem[Adr+3] }
: !C {Re,Im,Adr} E2 C2 3+ !W !W { } ;
        { Mem[Adr]=Re, Mem[Adr+3]=Im }
```

3.2. Представление комплексных значений в виде символьных строк

Чтобы полученные в результате вычислений комплексные значения представить в визуальном виде, необходимо уметь выводить их на печать (например, на экран терминала). Операции печати комплексного значения используют уже имеющиеся операции печати вещественного значения и дополняют образуемую ими строку вывода до изображения комплексного значения в виде двух соединённых символом знака вещественных значений, оканчивающихся символом мнимой единицы:

```
: .Cmplxp {Re,Im,p}
   E3 C Rm=0? BR+ D .Re.Realp C
   |Im|=1? BR+ .Im* .Im.Realp .'i' { } ;
 : .Re.Realp {p,Im,Re} C3 .Realp {p,Im} ;
 : |Im|=1? {Im} RABS 1.0 R= {|Im|=1.0?} ;
 : .Im.Realp {p,Im} C RSGN
    BRS NOP .'+' .'+' E2 .Realp { } ;
 : .Im* {p,Im} RSGN
    BRS .'–' .'+0' .'+' D { } ;
  : .'–' '–' TOB ;  : .'+' '+' TOB ;
  : .'i' 'i' TOB ;
  : .'+0' '+' TOB '0' TOB ;
: .Cmplx {Re,Im} 8 .Cmplxp { } ;
: .C {Re,Im} CC .Cmplx {Re,Im} ;
```

Для ввода комплексного числа необходимо уметь преобразовывать вводимую строку в комплексное значение. При реализации такой операции преобразования **Str->Cmplx** два раза вызывается операция преобразования строки в вещественное значение **Str->Real** (сначала для реальной, а потом для мнимой части), а также осуществляется проверка соблюдения формата для задаваемого комплексного числа (согласно таблице 1):

```
VAR pBCS { указатель на начало заданной строки }
VAR pECS { указатель на конец заданной строки }
VAR pCS { указатель на текущий символ строки }
: Str->Cmplx {Str,Len}
   EON WrongCmplx! RERAISE
   C2 ! pBCS C2 C2 + ! pECS
   {Str,Len} Get.Re 0. {XRe,XIm=0.}
   Str->Real_pS ! pCS Str->Real_Ch {Ch}
   BR '+' Get.ImReal '-' Get.ImReal
      'i' E2 'I' E2 'SP' NOP NUL NOP
   ELSE WrongCmplx! {XRe,XIm)} ;
: Get.Re {ReStr,Len} Str->Real {XRe} ;
: Get.ImReal {XRe,0.} pCS 1- pECS C2 -
   Get.Im E2D Str->Real_pS ! pCS
   Str->Real_Ch =I? IF0 WrongCmplx!
   Ch=SP|NUL? IF0 WrongCmplx! {XRe,XIm} ;
 : Get.Im {ImStr,ImLen} Str->Real {XIm} ;
 : =I? {Ch} C 'i' = E2 'I' = OR ;
 : Ch=SP|NUL? pCS pECS C2 <=
    {pCS,pECS<=pCS?} BR+ T1 =SP? ;
   : =SP? @T 'SP' = ;
```

Вместе с вызовами процедуры **Str->Real** используются также вспомогательные процедуры Str->Real_pS и Str->Real_pS, которые позволяют узнать, где в заданной строке и на каком символе остановилось распознавание вещественного числа с тем, чтобы иметь возможность продолжать в заданной строке дальнейшее распознавание комплексного значения.

3.3. Реализация типа COMPLEX

Реализация типа данных, представляющего комплексное значение, осуществлялась с применением имеющихся в ДССП-ТВМ средств объектно-ориентированного программирования, подробно описанных в [11].

Тип данных **COMPLEX**, используемый для объявления переменных и массивов комплексного типа, определяется как класс с полями, представляющими вещественные значения его реальной и мнимой частей, и методами, предназначенными для чтения комплексного значения в стек и записи нового значения из стека:

```
CLASS: COMPLEX
          VAR .Re  VAR .Im
        0 METHOD# @Val
        1 METHOD# !Val
;CLASS
```

Процедура, вызываемая по имени объекта (типа **COMPLEX**), как исполнитель метода с номером 0, в начале своего выполнения получает в вершине стека адрес объекта, используя который, считывает два вещественных значения из полей объекта **.Re** и **.Im**, чтобы занести их соответственно в подвершину и вершину стека:

```
COMPLEX :M: @Val
  {'X} C .Re E2 .Im {X.Re,X.Im} ;
```

А процедура, исполняющая метод (с номером 1), вызываемый префиксной операцией над объектом, обозначаемой словом **!** , приготовленную в стеке пару вещественных значений Re и Im записывает в память (соответственно в поля **.Re** и **.Im**) по адресу объекта, передаваемого ей в вершине стека:

```
COMPLEX :M: !Val { X.Re:=Re; X.Im:=Im }
  {Re,Im,'X} E2 C2 ! .Im ! .Re { } ;
```

3.4. Реализация компилирующего слова DVALUE

В ассортименте компилирующих средств ДССП-ТВМ уже предусмотрено слово **VALUE**, позволяющее создавать именованные слова-константы, при вызове которых в стек засылается одно 27-тритное значение, определённое при объявлении такой константы.

Но комплексная константа состоит из двух вещественных значений. Значит, при вызове слова, представляющего именованную комплексную константу, в стек должно засылаться два вещественных 27-тритных значения.

Поэтому возникла необходимость создать в ДССП-ТВМ новое компилирующее слово (названное **DVALUE** по аналогии со словом **VALUE**), позволяющее определять слова-константы, которые при вызове своего имени посылали бы в стек не одно, а два 27-тритных значения.

Приведём реализацию этого компилирующего слова в среде интерпретатора ДССП-ТВМ на языке ДССП-Т:

```
: DVALUE HEAD {Val1,Val2} ''LIT , E2 ,
  {Val2} ''LIT , {Val2} , ''; ,{ } ;
```

Далее при вызове такого слова **DVALUE**:

```
Val1 Val2 DVALUE DCNST
```

будет создаваться слово DCNST, посылающее в стек два значения Val1 и Val2, тело которого будет сформировано, как и при определении:

```
: DCNST Val1 Val2 ;
```

Реализация подобных компилирующих слов в среде интерпретатора осуществляется при помощи комплекта вспомогательных компилирующих средств, предусмотренных в ДССП-ТВМ (и описанных в [12, п.7.5.]). А с приёмами построения и функционирования тел, формируемых такими словами-компиляторами, можно подробно ознакомиться в [13].

4.Примеры применения пакета комплексной арифметики

Проиллюстрируем применение операций комплексной арифметики примерами разработки в ДССП-ТВМ программ, выполняющих дискретное преобразование Фурье над векторами комплексных чисел длиной $N=2^P$.

Сначала приведём реализацию процедуры простого дискретного преобразования Фурье (ДПФ) квадратичной сложности $O(N^2)$, которая вычисляет комплексные значения результирующего вектора напрямую по приведённым далее формулам. А затем рассмотрим процедуру быстрого преобразования Фурье (БПФ) логарифмической сложности $O(N \cdot \log_2 N)$, исполнение которой осуществляется путём многократного применения операции комплексной арифметики, называемой «бабочка Фурье».

4.1. Пример реализации процедуры простого преобразования Фурье

Операция дискретного преобразования Фурье над заданным вектором комплексных чисел Y_N заключается в получении результирующего вектора X_N комплексных чисел, элементы которого X_k вычисляются на основе исходного вектора Y по правилу:

$$X_k = \sum_{j=0}^{N-1} \{Y_j \times W_N^{k \cdot j}\}$$

в котором так называемые весовые коэффициенты вида $W^q{}_N$ вычисляются по формуле: $W_N^q = e^{-i \cdot 2\pi \cdot q/N}$ или $W_N^q = \cos\frac{2\pi q}{N} - i \cdot \sin\frac{2\pi q}{N}$, т.к. $e^{i \cdot \varphi} = \cos\varphi + i \cdot \sin\varphi$, где i – мнимая комплексная единица.

Будем считать, что эти величины вычислены заранее и приготовлены в виде массива комплексных чисел W размером от 0 до N. Для осуществления ДПФ можно предложить простой алгоритм, состоящий из двух вложенных циклов, который вычисляет каждый элемент результирующего вектора напрямую по описанному правилу. Выразим этот алгоритм отдельной функцией **DFT** на языке С:

```
void DFT(int N, complex *X,
       complex *Y, complex *W) {
int k,j; complex Xk, V;
for (k=0; k<N; k++) {
 Xk= 0.0 + I*0.0;
 for (j=0; j<N; j++) {
   V=W[(k*j)%N]; Xk= Xk+Y[j]*V;
 }//for j
 X[k]= Xk;
}//for k
}//DFT
```

А теперь продемонстрируем реализацию по такому же алгоритму процедуры **DFT** в среде ДССП-ТВМ с применением пакета комплексной арифметики:

```
: DFT {W,Y,X,N} C DO- X[k] DDDD { } ;
: X[k] {W,Y,X,N,k'} {k'=N-1,..,1,0}
   C2 1- C2 - 0. 0. {..,k',k=N-1-k',Xk=(0.,0.)}
   5 CT DO- +X[k] {..,k',k,Xk=(Re,Im)}
   C3 6 * 7 CT +  {..,'X[k]=X+6k}
  {W,Y,X,N,k',k,Xk,'X[k]} !C {X[k]:=Xk}
   D {W,Y,X,N,k'} ;
: +X[k] {W,Y,X,N,k',k,Xk=(Re,Im),j'} {j'=N-1,..,1,0}
   6 CT 1- C2 - {..,j=N-1-j'}
   W[(k*j)%N] {..,k',k,(Xk),j',j,V=(VRe,VIm)}
   C3 6 * 12 CT + {..,'Y[j]} @C
  {..,(Xk),j',j,(V),(Y[j])} C* CC3 C+ CE3 DDD
  {W,Y,X,N,k',k,(Xk)=(Xk)+(V)*(Y[j]),j'} ;
: W[(k*j)%N] {W,Y,X,N,k',k,Xk=(Re,Im),j',j}
   C 6 CT * 8 CT /DivMod E2D {..,(k*j)%N}
   6 * 11 CT + @C {..,V=W[(k*j)%N] } ;
```

В этом алгоритме для получения результирующего вектора длины N приходится исполнять порядка $O(N^2)$ операций с комплексными числами. Вот почему такой алгоритм характеризуется вычислительной сложностью $O(N^2)$.

4.2. Пример реализации процедуры быстрого преобразования Фурье

Для выполнения БПФ будем использовать вариант известного алгоритма Кули-Тьюки с прореживанием по времени для векторов комплексных чисел длиной $N=2^P$. В этом алгоритме сначала выполняется так называемое битреверсное копирование (или перестановка) вектора, а затем в несколько ($P=\log_2 N$) этапов осуществляется операция «бабочка Фурье» (**BF**) над всевозможными парами элементов вектора.

Подробнее с таким алгоритмом можно ознакомиться, например, в п.2 в [14]. Здесь же мы ограничимся его описанием в виде отдельной функции на языке Си:

```
void FFT(int N, complex *X,
 complex *Y,complex *W)
{long int P,t,s,h,G,R,d,k,u,j,a,b;
 complex V,T;
// Часть 1. Бит-реверсное копирование Y=>X:
 BRVPRST(N,X,Y);
//ч2.Основной цикл исполнения бабочек Фурье:
 P=iLog2(N); // P= log2N (так что N=2^P)
 s=1;h=2;G=1;R=N/2;d=N/2;
 for(t=1;t<=P;t++) { // на t-ом этапе:
  for(k=0,u=0;k<G;k++,u=u+d){
   V=W[u];
   for(j=0,a=k;j<R;j++,a=a+h)
   {b=a+s; BF(X[a],X[b],V,T);}
  }//for k
  h=h*2;s=s*2;G=G*2;R=R/2;d=d/2;
 }//for t
}//FFT
```

Для заданного комплексного вектора **Y** длиной **N** и таблицы комплексных коэффициентов

W функция **FFT** получает в качестве результата комплексный вектор **X**. При своём исполнении она вызывает процедуру бит-реверсного копирования **BRVPRST** (которая здесь не уточняется) и многократно осуществляет над парами комплексных элементов операцию **BF** («бабочку Фурье»), которую можно выразить в виде такого макроопределения на языке Си:

```
#define BF(XA,XB,V,Tmp) \
 { Tmp=XB*V; XB=XA-Tmp; XA=XA+Tmp; }
```

На каждом из $P=\log_2 N$ этапов алгоритма БПФ операция **BF** осуществляется над всеми N/2 парами элементов вектора, поэтому вычислительная сложность такого алгоритма оценивается величиной $O(N\log_2 N)$.

А теперь продемонстрируем реализацию процедуры **FFT**, действующей по такому алгоритму БПФ, в среде ДССП-ТВМ с применением пакета комплексной арифметики:

```
: FFT {W,Y,X,N} {{ P= log2(N), так что N=2^P
 {{ч1. Бит-реверсное копирование X<=BRVCopy(Y)
   'BRT C4 C4 C4 BrvCopyCVctr
 {{ч2. Основной цикл исполнения бабочек Фурье:
   1 2 1 N /2 C {..,s=1,h=2,G=1,R=N/2,d=N/2}
   P DO- DoStage(t) {t'=P-1,..,1,0; t=0,1,..P-1}
   DDD DD DDDD { } ;
: DoStage(t) {W,Y,X,N,s,h,G,R,d,t'}
   0 5 CT {..,u=0,G} DO- DoGroup(k) D
   5 ET 6 ET 4 * 5 ET
   E4 *2 E4 E3 /2 E3 E2 /2 E2
 {W,Y,X,N, s=s*2,h=2*s,G=G*2,R=R/2,d=d/2,t'} ;
: DoGroup(k) {W,Y,X,N, s,h,G,R,d,t',u,k'}
   C2 6 * 13 CT + @C {..,(V)=W[u]}
   8 CT 1- C4 - {..,(V),a=k}
   8 CT {R} DO- DoBTFLY(j) DDD
   {..,d,t,u,k'} E2 C4 + E2 {..,d,t,u=u+d,k'} ;
: DoBTFLY(j) {W,Y,X,N, s,h,G,R,d,t',u,k',(V),a,j'}
   12 CT C3 + {..,b=a+s} 15 CT C C3 6 * +
   E3 D C4 6 * + {..,(V),a,j','X[b],'X[a]}
   C @C C4 @C 10 CT 10 CT
  {..,(V),a,j','X[b],'X[a],Xa=(X[a]),Xb=(X[b]),(V)}
   BF {..,(V),a,j','X[b],'X[a], (Xa$),(Xb$)}
   6 CT !C {X[b]:=(Xb$)} C3 !C {X[a]:=(Xa$)}
  {..,(V),a,j','X[b],'X[a]} DD E2 11 CT + E2
  {..,(V),a'=a+h,j'} ;
: BF {(Xa),(Xb),(V)} C*+- {(newXa),(newXb)} ;
```

4.3. Сравнение результатов разных алгоритмов преобразования Фурье

В отличие от целых чисел, вычисления над которыми в компьютере можно осуществлять точно (в рамках разрешённого диапазона), с вещественными числами (представленными в форме с плавающей точкой) произвести точных вычислений на компьютере, вообще говоря, не удаётся. Можно рассчитывать получить лишь такой результат, который с той или иной степенью погрешности будет совпадать с действительным.

Другими словами, все вычисления, совершаемые с такими вещественными числами в компьютере, осуществляются не точно, а приближённо. Почему так? Во-первых, само значение вещественного числа в форме с плавающей точкой уже представляется в компьютере с некоторой погрешностью. А во-вторых, многие операции над такими числами осуществляются с применением округления.

К сожалению, из-за накапливаемых погрешностей округления возможны даже расхождения в результатах вычислений, которые выполняются по одинаковым формулам, но разными способами (алгоритмами), предполагающими различный порядок исполнения операций (например, другой порядок суммирования слагаемых).

Подобное расхождение вычислительных результатов наблюдается и при выполнении расчётов с применением комплексных чисел (представляемых парой вещественных). Например, вектора комплексных чисел, получаемые в результате преобразования Фурье двумя разными алгоритмами (ДПФ и БПФ), могут в точности и не совпадать.

Поэтому сравнивать результаты вычислений, полученные разными способами, можно лишь с некоторой точностью, оговаривая при этом определённую степень близости значений (вещественных или комплексных) чисел, при соблюдении которой считать их равными.

Для вещественных чисел в качестве меры близости обычно принимают абсолютную величину разности их значений. И полагают, что два числа X и Y совпадают с заданной точностью EPS, если соблюдается условие: $|X-Y| \le EPS$.

В качестве меры близости двух комплексных значений $X=(X.Re,X.Im)$ и $Y=(Y.Re,Y.Im)$ обычно принимают модуль их разницы, вычисляемый как расстояние между ними на комплексной плоскости по формуле:

$$M(X,Y) = \sqrt{(X.Re - Y.Re)^2 + (X.Im - Y.Im)^2}$$

И полагают два комплексных числа X и Y равными с точностью EPS при соблюдении условия: $M(X,Y) \le EPS$.

С использованием такого критерия совпадения комплексных значений были проведены сравнения комплексных векторов, полученных в качестве результатов преобразования Фурье рассмотренными выше алгоритмами ДПФ и БПФ. Прогоны и сравнения результатов выполнялись на двоичной машине архитектуры Intel и на троичной виртуальной машине ТВМ.

Вычисления на двоичной машине осуществлялись программой на языке Си, подготовленной в среде MS Visual Studio 2017, в которой для представления комплексных чисел (типа complex) использовалась пара вещественных

32-битных значений одинарной точности (типа float). Для исполнения ДПФ и БПФ в этой программе вызывались Си-функции DFT и FFT, представленные в п.4.1. и 4.2.

Вычисления на троичной машине выполнялись в среде интерпретатора ДССП-ТВМ с использованием операций пакета комплексной арифметики, в которых комплексные числа представлены двумя вещественными 27-тритными значениями. Для исполнения ДПФ и БПФ в ДССП-программе вызывались слова-процедуры DFT и FFT, реализация которых была представлена ранее в п.4.1. и 4.2.

Для выявления несовпадений комплексных значений в ДССП-программе определялось слово **CmpCVError?** :

```
: CmpCVError? {XRe,XIm,YRe,YIm}
   CC2 CC2 C- CMOD^2 CEPS C R* R>
   {((XRe−YRe)^2+(XIm−YIm)^2)>CEPS^2};
```

которое для двух комплексных чисел X=(XRe,XIm) и Y=(YRe,YIm), размещённых в стеке, проверяло, совпадают ли они с заданной точностью CEPS. Причём проверка на несовпадение выполнялась по упрощённой формуле:

$$((XRe - YRe)^2 + (XIm - YIm)^2) > CEPS^2$$

без вычисления квадратного корня.

И в Си-программе (на двоичной машине) и в ДССП-программе (на троичной машине) преобразование Фурье выполнялось над векторами комплексных чисел длиной вида $N=2^P$ для значений N от 32 до 4096, а исходный вектор Y предварительно инициализировался по одному и тому же правилу:

Y[k].Re= +**cos**(−π+k·h),

Y[k].Im= −**sin**(−π +k·h),

где h= 2·π/N , k=0,1,..,N-1

Комплексные вектора, полученные в результате выполнения преобразования Фурье алгоритмами DFT и FFT, сравнивались поэлементно с точностью EPS= 10^{-4}, 10^{-5}, 10^{-6}. При этом для каждой пары таких векторов подсчитывалось количество элементов, в которых комплексные значения не совпадали с заданной точностью.

Выявленное в итоге проведённых сравнений количество несовпадений, характеризующее расхождение результатов ДПФ и БПФ для векторов длины вида $N=2^P$ на двоичной машине (Intel) и троичной машине (ТВМ), представлены соответственно в таблицах 2 и 3.

Таблица 2. Расхождение результатов ДПФ и БПФ на двоичной машине архитектуры Intel

длина		точность сравнения **EPS**		
P	**$N=2^P$**	**10^{-4}**	**10^{-5}**	**10^{-6}**
5	32	0	0	1
6	64	0	0	11
7	128	0	1	47
8	256	0	4	121
9	512	0	18	329
10	1024	4	51	783
11	2048	9	188	1759
12	4096	32	551	3802

Таблица 3. Расхождение результатов ДПФ и БПФ на троичной виртуальной машине ТВМ

длина		точность сравнения **EPS**		
P	**$N=2^P$**	**10^{-4}**	**10^{-5}**	**10^{-6}**
5	32	0	0	2
6	64	0	0	4
7	128	0	1	18
8	256	0	0	32
9	512	0	3	106
10	1024	1	16	246
11	2048	1	49	713
12	4096	3	110	2040

Анализируя данные, представленные в этих таблицах, можно сделать следующие выводы.

1. При установлении более строгого критерия равенства комплексных значений, выражающегося в задании меньшего значения EPS, возрастает количество выявляемых случаев несовпадения значений в векторах-результатах.

2. С ростом длины векторов (N) возрастает и количество обнаруживаемых ошибок несовпадения значений в векторах-результатах, что объясняется как следствие возросшего объёма вычислительных операций.

3. При выполнении вычислений на троичной машине выявляется значительно меньше ошибок расхождения результатов, чем при выполнении тех же вычислений на двоичной машине.

5.Заключение

В статье представлены основные возможности пакета комплексной арифметики, созданного в ДССП-ТВМ для троичной виртуальной машины. Пояснены ключевые аспекты реализации этого пакета на языке ДССП-Т. Продемонстрированы примеры его применения для построения вычислительных программ в интерпретаторе ДССП/ТВМ.

На примерах программ, выполняющих преобразование Фурье для комплексных векторов, показано, что вычисления на троичной машине можно производить с лучшей точностью, чем аналогичные вычисления на двоичной машине.

Публикация выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН «Проведение фундаментальных научных исследований (47 ГП)» по теме № FNEF-2022-0004 «Разработка архитектуры, системных решений и методов для создания микропроцессорных ядер и коммуникационных средств семейства систем на кристалле двойного назначения», Рег. № 122041100063-0.

Complex 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. Initially, it provided only operations of integer arithmetic. Then the DSSP-TVM was supplemented with a package of real floating point arithmetic operations. Now a package of complex arithmetic has been created on its basis. The article describes the main possibilities of the proposed package, as well as explains the most important aspects of its implementation in the DSSP-T language in the environment of the DSSP/TVM interpreter.

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

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

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

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

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

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

8. 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.

9. 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.

10. А.А Бурцев. Вещественная арифметика в ДССП для троичной машины. «Труды НИИСИ РАН», Т. 11 (2021), № 2, 33–41.

11. Бурцев А.А., Рамиль Альварес Х. Средства объектно-ориентированного программирования в ДССП. «Программные системы и инструменты. Тематический сборник №4», М., Изд-во факультета ВМК МГУ, 2003, 166-175.

12. А.А. Бурцев, М.А. Бурцев. ДССП для троичной виртуальной машины. «Труды НИИСИ РАН», Т. 2 (2012), № 1, 73–82.

13. А.А. Бурцев. Разработка собственных управляющих конструкций в среде ДССП для троичной машины. «Труды НИИСИ РАН», Т. 8 (2018), № 2, 52–64.

14. А.А. Бурцев. О возможности применения векторного сопроцессора для ускорения операции быстрого преобразования Фурье. «Труды НИИСИ РАН», Т. 5 (2015), № 2, 138–147.