Бурцев А.А.[1], Сидоров С.А[2]

[1] Федеральный Научный Центр НИИ Системных Исследований (ФГУ ФНЦ НИИСИ РАН), г. Москва, к.ф-м.н., доцент, с.н.с. Отдела Архитектур Высокопроизводительных Микропроцессоров (ОАВМ) Отделения Разработки Вычислительных Систем (ОРВС), burtsev@niisi.msk.ru

[2] ФГУ ФНЦ НИИСИ РАН, г. Москва, к.ф-м.н., зав. Отделом Сопровождения Разработок (ОСР) Отделения Разработки Вычислительных Систем ОРВС, sidorov@niisi.msk.ru

Программный комплекс ДССП-ТВМ для структурированного программирования троичной [виртуальной] машины.

КЛЮЧЕВЫЕ СЛОВА:

Троичная симметричная система счисления, троичный компьютер, ТВМ – троичная виртуальная машина, структурированное программирование, ДССП – Диалоговая система структурированного программирования.

АННОТАЦИЯ:

В статье подчёркивается значимость троичной информатики, объясняются преимущества троичных машин (перед двоичными), и в качестве одного из возможных путей дальнейшего развития отечественной компьютерной индустрии рассматривается предпринятый (в 2010-2013 гг.) в научно-исследовательской лаборатории троичной информатики (НИЛ ТИ) ВМК МГУ вариант построения троичного компьютера в виде имитационной программной модели (ТВМ – троичной виртуальной машины), сопровождаемой комплексом программных инструментов (ДССП-ТВМ), предназначенных для разработки структурированных программ для троичной машины на языке ДССП-Т – троичном варианте языка ДССП – Диалоговой системы структурированного программирования, созданной в начале 80-х годов XX века в проблемной лаборатории ЭВМ ВМК МГУ под руководством Николая Петровича Брусенцова (07.02.1925–04.12.2014) – творца первых в мире троичных компьютеров «Сетунь» и «Сетунь-70».

Рассматривается состав и структура комплекса, даётся характеристика системы команд троичной виртуальной машины, её языка ассемблера, а также языка ДССП-Т.

Описываются основные возможности кросс-компилятора и наполняющей ДССП-словарь библиотеки модулей (на языке ДССП-Т), обеспечивающие построение троичных программ с последующим их исполнением не только на самой «голой» ТВМ (автономно), но и в среде диалогового командного монитора (ДКМОН) или диалогового ДССП-интерпретатора, способных функционировать на троичной машине в качестве её резидентного ПО (программного оснащения).

ДССП-ТВМ уже сейчас может использоваться в качестве учебной среды программирования, служить своеобразным «полигоном» для освоения «искусства программирования» троичных машин, и тем самым способствовать подготовке квалифицированных специалистов в области троичной информатики.

Статья сопровождается двумя приложениями, в одном из которых демонстрируются примеры программ на языке ДССП-Т, а в другом – вариант учебного плана (или учебной программы) спецкурса с условным названием «Программирование в ДССП для троичной машины», который предлагался (и предлагается!) для изучения студентам старших курсов (университетов или технических вузов), освоившим базовый курс программирования.

В настоящее время, когда традиционная (кремниевая) микроэлектроника, построенная на двоичной системе счисления, подходит к пределу своих технологических возможностей, возрождается интерес к исследованию принципиально иных путей построения вычислительных средств, основанных на других системах счисления. Как одна из наиболее перспективных, многих исследователей давно привлекает троичная симметричная система счисления (с цифрами -1,0,+1), которую когда-то Д.Кнут назвал “самой изящной” [1], и которая была полвека назад успешно воплощена в троичных цифровых машинах (ЦМ) “Сетунь” и “Сетунь-70” [2], разработанных в НИЛ ЭВМ МГУ под руководством Брусенцова Н.П.

Использование симметричного троичного кода и трёхзначной логики даёт ряд неоспоримых преимуществ (перед двоичными системами), которые были подтверждены практической эксплуатацией этих машин. Среди множества достоинств рассматриваемой троичной симметричной системы счисления можно выделить следующие:

- тем же количеством разрядов можно кодировать больший диапазон чисел;
- единообразное представление отрицательных и положительных чисел;
- непосредственное воплощение логики с тремя значениями: да (+1), нет (−1), не знаю [может быть?] (0);
- упрощённая реализация ряда арифметических операций (сдвига, сравнения чисел, смены знака); трёхзначность функции “знак числа”;
- оптимальное округление чисел простым отсечением младших разрядов и взаимокомпенсируемость погрешностей округления в процессе вычислений.

С осознанием этих преимуществ приходит понимание, что с троичными вычислениями и троичными компьютерами связано будущее вычислительной техники. И горизонты этого будущего открываются сегодня всё большему кругу научных работников и специалистов, причём не только сферы информационных технологий. Так, например, в Интернете появились статьи на тему: “Будущее квантовых компьютеров — в троичных вычислениях”[3]. Встречаются также публикации о преимуществах троичных машин, в которых в качестве одного из доводов в пользу троичности приводится упоминание о троичности нейрона человеческого мозга [4].

В НИЛ ТИ ВМК МГУ (ранее НИЛ ЭВМ), где продолжаются исследования возможных способов реализации троичного процессора на современной элементной базе [5-6], была предпринята попытка создания имитационной модели троичного процессора, а также системы разработки троичных программ для него. В результате проведённой (в 2010-2013 гг.) работы был создан программный комплекс ТВМ (Троичная Виртуальная Машина) [7], имитирующий функционирование современного варианта троичного процессора двухстековой архитектуры с поддержкой структурированного программирования на уровне машинных команд, аналогичной той, что была обеспечена в троичной ЦМ “Сетунь-70”.

В той же НИЛ ЭВМ МГУ ранее (в 80-х годах XX века) была разработана система программирования ДССП [8], которая на протяжении ряда лет применялась для программирования двоичных мини- и микрокомпьютеров самой разнообразной архитектуры и в различных операционных средах. ДССП убедительно показала, как важны для успешного построения компьютерных программ интерактивная среда разработки, двухстековая архитектура, структурированный набор команд управления и поддержка нисходящей разработки по принципу “сверху-вниз” (top-down). Поэтому

в качестве основного средства разработки троичных программ для ТВМ было решено создать троичный вариант этой системы программирования, т.е. троичную ДССП с языком ДССП-Т – троичным вариантом языка ДССП.

Работа по созданию системы разработки программ для ТВМ на языке ДССП-Т была выполнена в два этапа. Сначала (на 1-ом этапе) была построена кросс-система ДССП-ТВМ [9], позволяющая создавать программы для ТВМ на языке ДССП-Т с помощью кросс-компилятора. Затем (на 2-ом этапе), используя уже саму ДССП-ТВМ как среду разработки, был создан диалоговый интерпретатор ДССП/ТВМ [12], способный функционировать на троичной машине (ТВМ) в качестве её резидентной системы программирования.

В ДССП-ТВМ в качестве составляющих её компонент используются следующие программные средства (см. рис.1):

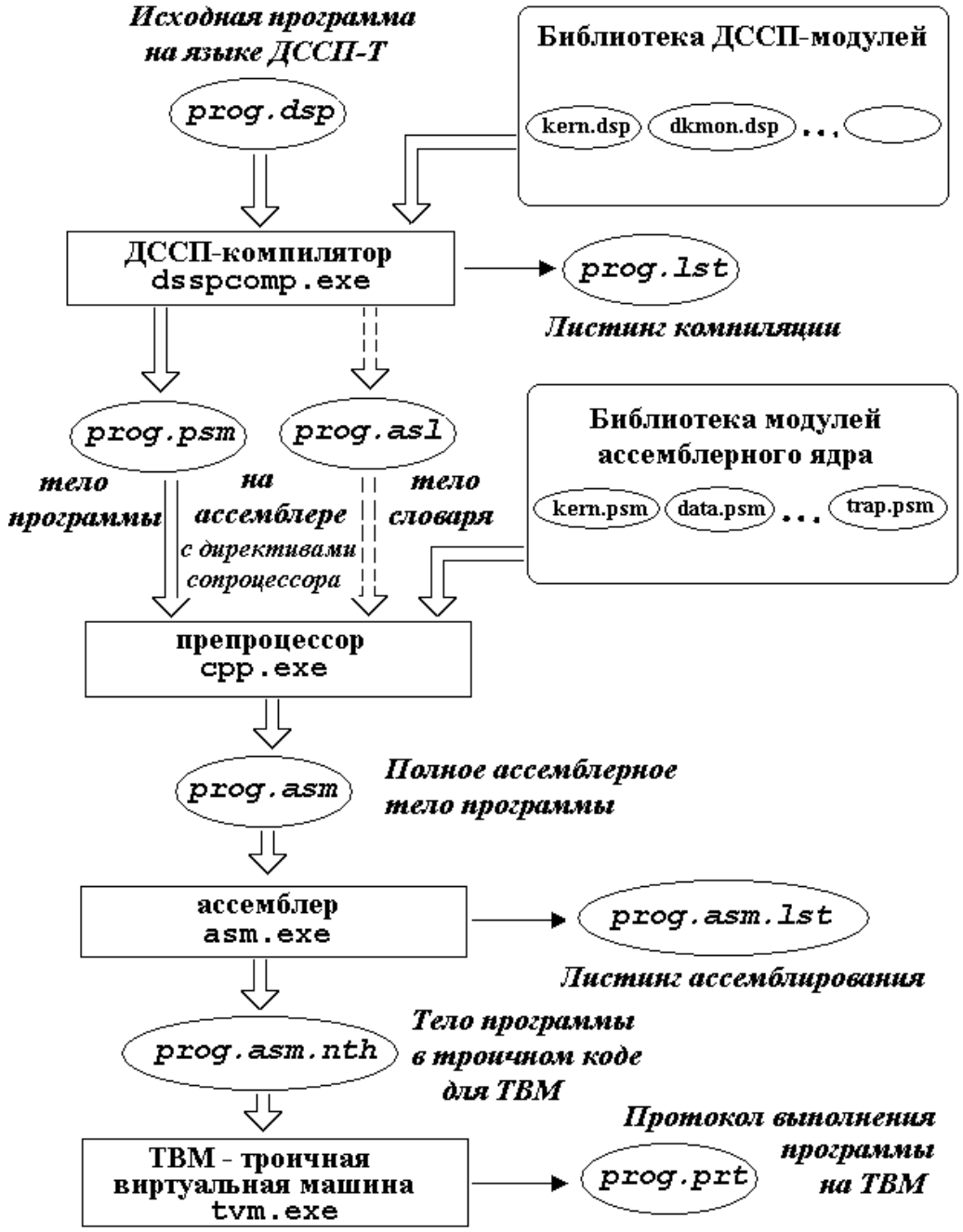

Рис.1. Основные компоненты ДССП-ТВМ и схема прогона ДССП-программы

- кросс-компилятор (dsspcomp.exe);
- ДССП-библиотека (kern.dsp,...) – набор файлов на языке ДССП-Т (dsp-файлы), подключаемых в ДССП-программу командой LOAD в ходе её компиляции, которые содержат определения стандартного ассортимента наиболее употребительных слов;
- стандартный препроцессор (cpp.exe) языка Си ;
- ассемблерное ядро ДССП (kern.psm,...) – совокупность ассемблерных файлов (psm-файлов), подключаемых в ассемблерную программу на стадии препроцессорной обработки, которые содержат тела процедур, необходимые для исполнения ряда примитивов ядра ДССП;
- ассемблер (asm.exe) троичной машины;
- имитатор троичной машины (tvm.exe).

Программа (файл prog.dsp), составленная на языке ДССП-Т, проходит несколько стадий обработки в ДССП-ТВМ (см. рис.1). Сначала она обрабатывается кросс-компилятором, который создаёт ассемблерную программу (файл prog.psm), содержащую также препроцессорные директивы. Компилятор формирует листинг сообщений о ходе компиляции (файл prog.lst), а также может создать ещё и ассемблерное тело словаря (файл prog.asl), сформированного программой.

Далее ассемблерная программа (файл prog.psm) вместе с файлами ассемблерного ядра ДССП обрабатывается препроцессором для получения единой ассемблерной программы (файл prog.asm), которая затем обрабатывается ассемблером троичного процессора, в результате чего создаётся листинг ассемблирования программы (файл prog.asm.lst) и троичный код (файл prog.asm.nth), готовый для исполнения на троичном процессоре. Наконец, имитатор троичного процессора (ТВМ) исполняет программу, представленную в троичном коде. Исполнение троичной программы на ТВМ можно отслеживать (на экране) не только в режиме диалога, но и получить протокол её работы (файл prog.prt).

Для автоматизации прохождения всех стадий обработки ДССП-программы (от её компиляции до исполнения на ТВМ), подготовлены соответствующие командные файлы. Так что полный прогон ДССП-программы можно вызвать всего одной командой: dssp-tvm prog.

Подробнее с возможностями ТВМ, ДССП-ТВМ и выразительными средствами языка ДССП-Т можно ознакомиться на Интернет-сайте НИЛ троичной информатики по адресу http://ternarycomp.cs.msu.ru/, а также по материалам серии статей, опубликованных в тематических сборниках «Программные системы и инструменты» факультета ВМК МГУ» [9-11] и «Труды НИИСИ РАН» ФНЦ НИИСИ РАН [12], а также доклада [13], представленного на научно-практической конференции «Посткремниевые вычисления», которая состоялась в рамках прошедшего в г. Переславль-Залесский в ноябре 2014 года Национального Суперкомпьютерного Форума (НСКФ-2014). Для первоначального краткого знакомства с языком ДССП-Т в приложении №1 размещены исходные тексты примеров учебных программ.

Созданный в НИЛ ТИ ВМК МГУ программный комплекс ДССП-ТВМ может использоваться не только в качестве среды опережающей подготовки и отладки программного оснащения для создаваемой «в кремнии» (т.е. аппаратно реализованной) троичной машины. В настоящее время ДССП-ТВМ может применяться ещё и в образовательном процессе в качестве учебной среды программирования. В языке ДССП-Т предусмотрены конструкции для структурированного, модульного, объектно-ориентированного и параллельного

программирования, а также средства для структурированной обработки исключительных ситуаций. Таким образом, ДССП-ТВМ может служить своеобразным «полигоном» для освоения современных методов программирования с учётом специфических особенностей троичных машин и тем самым способствовать подготовке квалифицированных специалистов в области троичной информатики.

Опыт использования (студентами кафедр АСВК и алгоритмических языков во время прохождения ими преддипломной практики в НИЛ ТИ) ДССП-ТВМ в деле создания ряда учебно-исследовательских программ для ТВМ, с одной стороны, подтвердил, что и сама система разработки, и её язык относительно легко могут усваиваться новичками. Но с другой стороны, всё же выявил потребность во внедрении в учебный процесс особого спецкурса, благодаря которому заинтересованные студенты могли бы гораздо быстрее освоить специфику структурированной разработки программ для троичной машины.

В целях удовлетворения обозначенной потребности и последующего внедрения ДССП-ТВМ в учебный процесс был составлен и предложен (на факультете ВМК МГУ) учебный план годового (возможно и полугодового) спецкурса с условным названием "Программирование в ДССП для троичной машины" (см. приложение №2). Этот спецкурс нацелен на ознакомление слушателей с троичной симметричной системой счисления, характеристиками троичных ЦМ "Сетунь" и "Сетунь-70", архитектурой и системой команд троичной виртуальной машины (ТВМ), а также возможностями среды ДССП-ТВМ и её языка ДССП-Т. Основная задача спецкурса заключается в освоении слушателями выразительных средств и приёмов разработки структурированных программ в ДССП для троичной машины. Спецкурс ориентирован на слушателей, усвоивших базовый курс программирования (например, на языке Паскаль), и может быть предложен для освоения студентам 2-4 курсов университетов или технических ВУЗов, обучающихся по направлениям: «Прикладная математика и информатика» и/или «Информатика и вычислительная техника».

Литература

1. Кнут Д. Искусство программирования на ЭВМ. т.2 Получисленные алгоритмы. п. 4.1. с. 216-219.
2. Брусенцов Н.П., Рамиль Альварес Х. Троичные ЭВМ “Сетунь” и “Сетунь 70”. // Первая Международная конференция "Развитие вычислительной техники в России и странах бывшего СССР: история и перспективы" SORUCOM-2006 (3-7 июля 2006 г., г. Петрозаводск, Россия): Труды конференции в 2-х ч. Петрозаводск: Изд-во ПетрГУ, 2006. ч.1, с. 45-51.
3. Будущее квантовых компьютеров — в троичных вычислениях. URL: http://www.infuture.ru/news.php?news_id=475 (дата обращения: 02.11.2014).
4. Малашевич Д.Б. Недвоичные системы в вычислительной технике. URL: http://www.computer-museum.ru/books/archiv/sokcon27.pdf (дата обращения: 02.11.2014)
5. Маслов С.П. Об одной возможности реализации троичных цифровых устройств // Программные системы и инструменты. Тематический сборник №12, М.: Изд-во факультета ВМК МГУ, 2011. С.222-227.
6. Маслов С.П. Троичная схемотехника // Программные системы и инструменты. Тематический сборник №13, М.: Изд-во факультета ВМК МГУ, 2012. С.152-158.
7. Сидоров С.А., Владимирова Ю.С. Троичная виртуальная машина // Программные системы и инструменты. Тематический сборник №12, М.: Изд-во ф-та ВМК МГУ, 2011. С.46-55.
8. 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 (12-16 сентября 2011 г., г. Великий Новгород, Россия): Труды конференции. В.Новгород: Изд-во НовГУ, 2011. с. 83-88.
9. Бурцев А.А., Рамиль Альварес Х. Кросс-система разработки программ на языке ДССП для троичной виртуальной машины // Программные системы и инструменты. Тематический сборник №12, М.: Изд-во факультета ВМК МГУ, 2011. С.183-193.

10. Бурцев А.А., Рамиль Альварес Х. Реализация средств объектно-ориентированного программирования в кросс-компиляторе языка ДССП-Т. // Программные системы и инструменты. Тематический сборник №13, М.: Изд-во факультета ВМК МГУ, 2012. с.28-37.
11. Бурцев А.А., Рамиль Альварес Х. Сопрограммный механизм в системе структурированного программирования для троичной машины. // Программные системы и инструменты. Тематический сборник №14, М.: Изд-во факультета ВМК МГУ, 2013. с.193-208.
12. Бурцев А.А., Бурцев М.А. ДССП для троичной виртуальной машины. // Труды НИИСИ РАН, т.2, №1. М.: Изд-во НИИСИ РАН, 2012. с.73-82.
13. А.А. Бурцев, С.А. Сидоров. ТРОИЧНАЯ ВИРТУАЛЬНАЯ МАШИНА И ТРОИЧНАЯ ДССП. URL: http://2014.nscf.ru/TesisAll/0_PostMoore_Plenar/06_208_BurtsevAA.pdf (дата обращения 20.10.2015)

Приложение состоит из двух частей:

1. Приложение №1. Примеры программ на языке ДССП-Т:

Примеры использования цикла DW с условием

```
PROGRAM DEMO_DW
LOAD kern
  ." примеры использования цикла DW  " CR
{{ ------------ Наибольший общий делитель
  : NOD {{  [ x,y ] => [ z ] , z= nod(x,y)
        x<>y? DW x-y|y-x D ;
   : x<>y? {[ x,y ]} C2 C2 <> ;
   : x-y|y-x {[ x,y ]} x>y? BR+ x-y y-x ;
    : x>y?  {[ x,y ]} C2 C2 > ;
    : x-y {[ x,y ]} E2 C2 - E2 {[ x-y,y ]} ;
    : y-x {[ x,y ]} C2 - {[ x,y-x ]} ;
{{ ------------ Число Фиббоначи (n-ое по порядку)
{{              F(0)=1, F(1)=1; F(k+1)= F(k)+F(k-1)
  : Fib {{  [ n ] => [ F(n) ] 1 1 E3 1- {[ FP,FT,k ]} >0? DW newF D E2D ;
    : >0? C 0 > ;  : newF {[ FP,FT,k ]} E2 E3 C3 + E2 1- {[ FT,FN,k-1 ]} ;
{{ ------------ Число Фиббоначи (первое по порядку, превыщающее M)
{{              F(0)=1, F(1)=1; F(k+1)= F(k)+F(k-1)
  : Fib> {{  [ M ] => [ Fi ] , Fi=F(i) > M
         1 1 {[ M,FP,FT ]} <=M? DW newFi E3 DD ;
    : <=M? C C4 <= ;  : newFi {[ FP,FT ]} E2 C2 + {[ FT,FN ]} ;
{{ ------------ Факториал
  : Fctrl {{  [ n ] => [ n! ] , n!= 1*2*...*n   1 E2 {[ F,k ]} >0? DW F*k D ;
    : F*k {[ F,k ]} E2 C2 * E2 1- {[ F*k, k-1 ]} ;
  : DEMO_DW 77 55 NOD . 7 Fib . { 21 } 20 Fib> . 5 Fctrl . ;
UNDEF CLEAR
$$
```

Примеры использования цикла DO- с параметром-счетчиком

```
PROGRAM DEMO_DO- { DKMON }
[] [*****   примеры использования цикла DO- с параметром-счетчиком.  *****]
[***   Вывод строки заглавных латинских букв ***]
:: : .LatString 26 DO- .LatLetter ;
    : .LatLetter [i] 'Z' C2 - TOB [i] ; [печать i-ой латинской буквы]
[***   Сумма квадратов всех целых чисел от 0 до N ***]
:: : SumKv [N] 0 E2 [S,N] 1+ DO- SumI [S] ;
    : SumI [S,i] E2 C2 C * [i,S,i*i] + E2 [S+i*i,i] ; [ i=N,N-1,..,1,0. ]
[***   Сумма квадратов всех целых чисел заданного диапазона (M..N) ***]
:: : SumKvD [M,N] C2 - 1+ 0 E2 [M,0,N-M+1] DO- SumID [M,S] E2 D [S] ;
    : SumID [M,S,i] E2 C3 C3 + [M+i] C *
           [M,i,S,(M+i)*(M+i)] + E2 [M,S+(M+i)*(M+i),i] ; [ i=N-M,..,1,0. ]
[***   Вычисление факториала ***]
::  : Factor [N] 1 E2 [1,N] DO- FI [N!] ;
     : FI [F,I] E2 C2 1+ * E2 [F*(I+1),I] ; [при I=N-1,..,1,0]
[***   Вычисление полинома по схеме Горнера ***]
    [ P(X) = (..(A9*X+A8)*X+ ..)*X+A0 ]
      9 VALUE n  [задает размер массива коэффициентов ]
    [массив коэффициэнтов A(0..9), определяющих полином : ]
    LONG CNST A 5 -2 0 1 0 0 0 0 0 0 ; [ X^3-2X+5 ]
   : Plnm [X] n A [A(n)] n [X,A(n),n] DO- PI [P(X)] ;
    : PI [X,S,i] E2 C3 * [X,i,S*X] C2 A + [S*X+A(i)] E2 [X,S,i] ;
[***   Сортировка массива методом простого выбора ***]
  9 VALUE N [размер массива]  N VCTR S [сортируемый массив S(0..N) ]
::  : SORT N 1+ [N+1] DO- SORTi ;
     : SORTi [i] MAXi  [i,k] EXCHANGEi D [i] ; [i=N,..,0]
      : MAXi [определяет индекс максим.элемента среди S0..Si ]
```

```
              [i] C [i,k] C2 [i] DO- MAXj [i,k] ;
       : MAXj [k,j] C2 S C2 S [S(k),S(j)] < IF+ k:=j [k,j] ;
        : k:=j [k,j] E2 D C [j,j] ;
      : EXCHANGEi [ переставляет i-ый и k-ый элементы массива ]
           [i,k] C S C3 S [i,k,S(k),S(i)] C3 ! S [S(i)=>S(k)]
                [i,k,S(k)] C3 ! S [S(k)=>S(i)]  [i,k] ;
[***   ввод массива S из стека ***]
::  : GETS [S0,S1,..,SN] N 1+ [N+1] DO- GETSi ;
     : GETSi [Si,i] E2 C2 ! S [i] ;
 [***  вывод массива S на терминал ***]
::  : PUT  N 1+ [N+1] DO- PUTi ;  : PUTi [i] N C2 - S .N ","  [i] ;
      : .N C IF- .- .N2 ;   : .N2 2 TON10 ;  : .-  ."-" NEG ;
[***   ввод массива S с терминала ***]
::  : GET  N 1+ [N+1] DO- GETi ;
     : GETi [i] "=" TIN N C3 - ! S [i] ;
::  : DEMO_DO-      ." LatString=" .LatString CR
         ." SumKv(10)=" 10 SumKv 4 TON10 CR
         ." Factor(7)=" 7 Factor 6 TON10 CR
         ." Plnm(3)="   3 Plnm   6 TON10 CR
         ." Сортировка массива " CR 55 23 67 12 99 07 66 78 33 47 GETS
         ."S[0:9]=" PUT CR ."SORT_S[0:9]" SORT CR ."S[0:9]=" PUT CR    ;
UNDEF  CLEAR
$$
```

Примеры программ перестановок в одномерном массиве

```
PROGRAM Test_PrstVctr { DKMON }
LOAD DKMON
{----- Ввод массива из стека -----}
::  : GetVctr {A0,A1,..,An,'A,k} {{'A=адрес,k=длина (кол-во эл-тов)
  {A0,..,An,'A,n+1} DO- Get_A[i] D { }; {for i=n,..,1,0 A[i]:=val from stack}
     : Get_A[i] {.., X, 'A, i } E2 E3 {.., 'A, i, X } C3 C3 *3 +
        {.., 'A, i, X, 'A+i*3 } !W { A[i]:=X } {.., 'A, i } ;
{----- Печать массива как десятичных чисел -----}
::  : PrintVctr {'A,k} DO- Print_A[i] D { } ; {{'A=адрес,k=длина(к-во эл-тов)
     : Print_A[i] { 'A[i], i } C2 3+ { 'A[i+1] } E3
        { 'A[i+1], i, 'A[i] } @W 0 TON10 SP  { 'A[i+1], i } ;
{------- Инверсия одномерного массива -------------}
:: : InvrVctr { A,k } {{ A - адрес массива, k= кол-во элементов , n=k-1
       {{ массив A[0:n]= [ A0,A1,..,An ] => [ An,..,A1,A0 ]
       { A,k } C 1- E2 /2 { A,n, k div2 } DO- A[i]<=>A[n-i] DD { } ;
    : A[i]<=>A[n-i] { for i= k div2,..,1,0 }
       { A,n,i } C3 C3 C3 - *3 + {..,A+(n-i)*3 } C @W {..,A[n-i] }
       { A,n,i,'A[n-i],A[n-i] } 5 CT C4 *3 + {..,A+i*3 } C @W {..,A[i] }
       { A,n,i,'A[n-i],A[n-i],'A[i],A[i] } E3 E2 !W { A[i]:=A[n-i] }
       { A,n,i,'A[n-i],A[i] } E2 !W { A[n-i]:=A[i] }      ;
{--- Циклический сдвиг одномерного массива влево. Простой вариант. ---}
:: : LShiftVctr {A,n,s} {{A-адрес массива, k=к-во эл-тов, s=число поз.сдвига
       {{ массив A[0:n]= [ A0,A1,..,A(n-1) ] => [ As,..,A(n-1),A0,..,A(s-1) ]
       { A,n,s } DO- LShift1Vctr DD { } ;
    : LShift1Vctr {A,n,i} C3 @W {A0} C4 {'A[0]} C4 {n} 1- DO- LShift_A[j]
       { A,n,i, A0, 'A[n] } !W { A[n]:= A0 } { A,n,i } ; { for i= s-1,.,1,0 }
    : LShift_A[j] {.., 'A[j],j } { j=n-1,..,1,0 }
       {.., 'A[j],j } C2 3+ C @W {.., 'A[j],j, 'A[j+1], A[j+1] } E2 E4
       {.., 'A[j+1],j, A[j+1], 'A[j] } !W { A[j]:=A[j+1] } {.., 'A[j+1],j } ;
{--- Циклический сдвиг одномерного массива влево. Эффективный вариант. ---}
:: : LShift!Vctr {A,n,s}{{A=адрес массива, k=к-во эл-тов,s=число поз. сдвига
        { A,n,s } C3 C2 {.., A,s } InvrVctr
        { A,n,s } C3 C2 *3 + C3 C3 - {.., 'A[s],n-s } InvrVctr
        { A,n,s } D { A,n } InvrVctr ;
   100 VCTR A  : 'A 0 ' A ;
:: : Test_PrstVctr CR ."---Перестановки в одномерном массиве (векторе)---" CR
       10 11 22 33 44 55 66 77 88 99  'A 10 GetVctr  'A 10 PrintVctr CR
       'A 10 InvrVctr ." A[0:9]=> "  'A 10 PrintVctr CR
       10 11 22 33 44 55 66 77 88 99 'A 10 GetVctr  'A 10 PrintVctr CR
       'A 10 3 LShiftVctr  'A 10 PrintVctr CR
       'A 10 2 LShift1Vctr 'A 10 PrintVctr CR
       'A 10 1 LShift1Vctr 'A 10 PrintVctr CR
       'A 10 0 LShift1Vctr 'A 10 PrintVctr CR
          10 11 22 33 44 55 66 77 88 99 'A 10 GetVctr  'A 10 PrintVctr CR
       'A 10 4 LShift!Vctr 'A 10 PrintVctr CR    ;
  UNDEF  CLEAR
$$
```

Троичная проверка попадания точки в отрезок

```
PROGRAM Test_TSEG
LOAD KERN
  ." Test_TSEG Троичная проверка попадания точки в отрезок " CR
{{ { a,b, x } TSEG { r } {{ r = +1 (a<x<b) или 0 (x=a|x=b) или -1 (x<a|b<x)
   :: : TSEG { a,b,x } C E4 { x,b, x,a } CMP { x?a } E3
              { x?a, b,x } CMP { x?a,b?x } TMIN { r } ;
  10 VALUE N   { константа, задающая количество точек }
   VAR A VAR B { переменные, задающие границы отрезка [A,B] }
   9  VCTR V   { одномерный массив точек V[0:9] }
  { приём значений точек из стека данных в массив V }
  : GetV {V0,V1,..,V9} N DO- GetV[i] ;
    : GetV[i] {..,Vi,i} E2 C2 ! V {..,i} ; { i=9,8,..,1,0 }
  : .10 {Num} 0 TON10 {Num} ; { печать 10-ного числа }
  { проверка попадания точек массива в отрезок: }
  : .CheckV ."--- для отрезка [" A .10 ',' TOB B .10 ']' TOB CR CheckV ;
  : CheckV N DO- CheckV[i] { i=9,8,..,1,0 } ;
   : CheckV[i] {i} A B ." точка " C3 V {i,A,B,V[i]} C .10
    {i,A,B,V[i]} TSEG {-1/0/1} BRS )A,B( |A,B| (A,B) CR {i} ;
    : )A,B( ." вне отрезка " ;    : |A,B| ." на границе отрезка " ;
    : (A,B) ." внутри отрезка " ;
:: : Test_TSEG ."===== Проверки попадания точек в отрезок: =====" CR
     { задание массива точек V[0:9]=} 5 4 7 3 8 -4 -7 0 6 -6 GetV
     { задание границ отрезка: }  4 ! A 7 ! B  { A:= 4; B:= 7 }
     { проверки попадания точек массива в отрезок:  [4,7] } .CheckV
     { задание границ отрезка: } -6 ! A 6 ! B  { A:=-6; B:=+6 }
     { проверки попадания точек массива в отрезок: [-6,6] } .CheckV  ;
 CLEAR UNDEF
 $$
```

Новые типы данных очередь (QUEUE) и дек (DEQ) как классы

```
PROGRAM DKMON {QDEMO}
[] [* ОЧЕРЕДЬ как Пример построения нового типа данных с помощью КЛАССА *]
LOAD DKMON
   CR ." Новый Тип данных QUEUE (Очередь) " CR :: : QDEMO? 1 ;
 [--- ОЧЕРЕДЬ элементов типа TWORD (3-х трайтовых целых) ---]
 100 VALUE MaxLen [ максимальная длина очереди ]
                  [--- Объявление класса ОЧЕРЕДЬ ---]
 ." -------- Объявление класса  QUEUE " CR
:: CLASS:  QUEUE  SUBCLASS TCLASS
     [--- поля данных, составляющих очередь ---]
      VAR .cnt [ кол-во элементов в очереди ]
      VAR .n  [ индекс-указатель начала очереди в массиве ]
      VAR .k  [ индекс-указатель конца очереди в массиве ]
      100 VCTR .MI  [ массив для хранения элементов очереди ]
     [--- методы , представляющие операции над очередью ---]
   ::  METHOD Init   [ инициировать, начать работу очереди ]
   ::  METHOD Done   [ завершить, закончить работу очереди ]
   ::  METHOD Show   [ показать на экране состояние очереди ]
   ::  METHOD Put    [ поместить элемент x из вершины стека в очередь Q ]
                    [ Пример вызова: {x} Put Q  ]
   ::  METHOD Get    [ взять элемент y из очереди Q на вершину стека ]
                    [ Пример вызова:  Get Q {y} ]
   ::  METHOD Empty? [ проверить, пуста ли очередь ]
   ::  METHOD Full?  [ проверить, полна ли очередь ]
   ::  METHOD= ! =>  [  копирование очереди ]
   ;CLASS
 ." -------- Объявление ситуаций " CR
      SITUATION  Empty! .Empty!   : .Empty! ." Empty Queue! " HALT ;
      SITUATION  Full!  .Full!    : .Full!  ." Full Queue! "  HALT ;
                [--- Реализация методов класса ОЧЕРЕДЬ ---]
  ." -------- Реализация методов QUEUE " CR
 QUEUE :M: Init [Q] 0 C2 ! .cnt                [ Q.cnt:=0 ]
                    0 C2 ! .k 1 E2 [Q] ! .n  [ ] ; [ Q.k:=0; Q.n:=1 ]
 QUEUE :M: Done [Q] D ;
 QUEUE :M: Show [Q] ."Queue:"  ShowQueue [] ;
           : ShowQueue [Q] ." (" C .cnt 2 TON10 .")= < "
                  C .n C2 .cnt [Q,n,cnt] DO- ShowItem DD ."> " []  ;
           : ShowItem [Q,t,i] C2 C4 [...,t,Q] .MI [M(t)] 3 TON10 ."  "
                     [Q,t,i] E2 +1mod E2 [Q,(t+1)mod,i] ;
 QUEUE :M: Put [x,Q] C .Full? IF+ Full! [ проверка на искл.ситуацию ]
```

```
                    C .k +1mod C C3 ! .k [ Q.k:=(Q.k+1)mod Maxlen ]
                [x,Q,k] C2 .cnt 1+ C3 ! .cnt      [ Q.cnt:=Q.cnt+1 ]
                        E2 [x,k,Q] ! .MI          [ Q.M(k):=x ] [ ] ;
QUEUE :M: Get [Q] C .Empty? IF+ Empty! [ проверка на искл.ситуацию ]
                    C .n C2 .MI E2    [ y:= Q.M(Q.n) ]
             [y,Q] C .cnt 1- C2 ! .cnt [ Q.cnt:=Q.cnt-1 ]
              C .n +1mod E2 [y,n+1,Q] ! .n [Q.n:=(Q.n+1)mod Maxlen] [y] ;
        : .Empty? [Q] .cnt 0 <= [1/0] ;    [ Q.cnt<=0 ? ]
 QUEUE :M= Empty? .Empty?
        : .Full? [Q] .cnt MaxLen >= [1/0] ;    [ Q.cnt>=MaxLen ? ]
 QUEUE :M= Full? .Full?
                        [--- Вспомогательные процедуры  ---]
 : +1mod [z] 1+ C MaxLen >= IF+ T0 [z+1 или 0, если z>=MaxLen-1 ] ;
 : -1mod [z] C IF0 +MaxLen 1-      [z-1 или MaxLen-1, если z=0 ]  ;
  : +MaxLen MaxLen + ;
[-------  Примеры объявленных экземпляров классов QUEUE ----------]
 ." Примеры использования QUEUE " CR
  QUEUE VAR Q    QUEUE VAR Q2
  9 QUEUE VCTR VQ  [ массив очередей VQ(0:9) ]
  3 4 2 QUEUE ARR AQ  [ матрица очередей AQ(0:3,0:4) ]
 :: : QDEMO CR ."Демонстрация операций с очередями " CR
       Init Q 33 Put Q 44 Put Q 55 Put Q 66 Put Q 77 Put Q ."Q= " Show Q CR
       9 DO- InitVQ(i)
       Q 3 => VQ [ VQ(3):= Q ] -99 3 Put VQ  ."VQ(3)= " 3 Show VQ CR
       -44 4 Put VQ  3 Get VQ . SP 4 Put VQ 3 Get VQ . SP 4 Put VQ CR
       ."VQ(3)= " 3 Show VQ CR ."VQ(4)= " 4 Show VQ CR
      [ действия с одним элементом матрицы очередей: ]
      1 2 Init AQ     [ A(1,2).Init  ]
      4 VQ 1 2 => AQ  [ A(1,2):=VQ(4) ]
      666 1 2 Put AQ  [ AQ(1,2).Put(666) ]
       ."AQ(1,2)= " 1 2 Show AQ CR   ;
     : InitVQ(i) [i] C Init VQ [i] ; [ инициализация i-ой очереди ]
```

```
." Новый Тип данных DEQ на основе QUEUE " CR  :: : DQDEMO? 1 ;
                    [--- Объявление класса ДЕК ---]
 [--- ДЕК элементов типа LONG (4-х байтовых целых) ---]
:: CLASS:  DEQ  SUBCLASS QUEUE [ наследует класс QUEUE ]
     [--- методы , дополняющие операции над очередью ---]
   ::  METHOD Put_ [ поместить элемент из вершины стека в начало очереди ]
   ::  METHOD Get_  [ взять элемент с конца очереди на вершину стека ]
   ::  METHOD Count [ определить количество элементов в очереди ]
   ;CLASS
                [--- Реализация методов класса ДЕК ---]
 DEQ :M= Count .cnt
 DEQ :M: Show [Q] ."Double-End-"  Show AS QUEUE [] ;
 DEQ :M: Put_ [x,Q] C .Full? IF+ Full! [ проверка на искл.ситуацию ]
                    C .n -1mod C C3 ! .n [ Q.n:=(Q.n-1)mod MaxLen ]
                [x,Q,n] C2 .cnt 1+ C3 ! .cnt [ Q.cnt:=Q.cnt+1 ]
                        E2 [x,n,Q] ! .MI     [ Q.M(n):=x ] [ ] ;
DEQ :M: Get_ [Q] C .Empty? IF+ Empty! [ проверка на искл.ситуацию ]
                    C .k C2 .MI E2   [ y:= Q.M(Q.k) ]
            [y,Q] C .cnt 1- C2 ! .cnt  [ Q.cnt:=Q.cnt-1 ]
            C .k -1mod E2 [y,k+1,Q] ! .k  [Q.k:=(Q.k-1)mod MaxLen] [y] ;
 [-------  Примеры объявленных экземпляров класса DEQ ----------]
 ." Примеры использования  QUEUE и DEQ " CR
  DEQ VAR DQ  [ дек , который может использоваться и как очередь ]
 104 VCTR ZA [ произвольная область памяти, которая используется как дек ]
       : Z 0 ' ZA ;  [ её адрес ]
 :: : DQDEMO CR ."Демонстрация операций с очередями и деком " CR
       Init Q 33 Put Q  44 Put Q  55 Put Q  66 Put Q  77 Put Q ."Q= " Show Q CR
       Init DQ  Q DQ => AS QUEUE [ DQ:= Q as QUEUE ]
       88 Put DQ 22 Put_ DQ 11 Put_ DQ 99 Put DQ ."DQ= " Show DQ CR
       Get DQ 3 TON10 ." <- " Get_ DQ 2 TON10 ." -> "
       Get DQ 3 TON10  ." <- " CR  ."DQ= " Show DQ CR
       DQ => Q [ Q:= DQ as QUEUE ] Get Q 3 TON10 CR ."Q= " Show Q CR
       9 DO- InitVQ(i)
       Q 3 => VQ [ VQ(3):= Q ] -99 3 Put VQ  ."VQ(3)= " 3 Show VQ CR
       -44 4 Put VQ  3 Get VQ . 4 Put VQ SP
       3 Get VQ .  4 Put VQ CR ."VQ(3)= " 3 Show VQ CR
       ."VQ(4)= " 4 Show VQ CR ."VQ(4)= " 4 VQ Show AS DEQ CR ;
CR ." Для запуска демонстрации выполните QDEMO или DQDEMO " CR
UNDEF CLEAR $$
```

2. Учебный план спецкурса "Программирование в ДССП для троичной машины" ("Programming in DSSP for ternary computer"):

Учебный план спецкурса
"Программирование в ДССП для троичной машины"
("Programming in DSSP for ternary computer")

годовой (возможно, полугодовой) для студентов 2-4 курсов
лектор – к.ф.м.н. Бурцев Алексей Анатольевич

Спецкурс "Программирование в ДССП для троичной машины" нацелен на ознакомление слушателей с троичной симметричной системой счисления, характеристиками троичных ЦМ "Сетунь" и "Сетунь-70", архитектурой и системой команд троичной виртуальной машины (ТВМ), а также возможностями Диалоговой Системы Структурированного Программирования (ДССП), созданными в НИЛ троичной информатики факультета ВМК МГУ.

Основная задача спецкурса заключается в освоении слушателями выразительных средств и приёмов разработки структурированных программ в ДССП для троичной машины.

- 1. Введение. Особенности троичной симметричной системы счисления. Краткая характеристика ЦМ "Сетунь" и "Сетунь-70". История создания и развития ДССП.
- 2. Краткая характеристика ТВМ, ДССП-ТВМ, ДССП/ТВМ . ТВМ — программный комплекс, имитирующий современный вариант троичного процессора двухстековой архитектуры с поддержкой структурированного программирования на уровне машинных команд. ДССП-ТВМ — кросс-система разработки программ для него на языке ДССП-Т. Структура и состав ДССП-ТВМ, прогон ДССП-программ в ней. Диалоговый интерпретатор ДССП/ТВМ и приёмы простейшей работы в нём.
- 3. Словарная организация ДССП. Программирование в ДССП как наращивание словаря исполнителя. Простейшие приёмы определения новых слов.
- 4. ДССП как стековая машина. Операции целочисленной арифметики над арифметическим стеком. Представление числовых констант, символов и строк. Работа с ДССП как со стековым калькулятором. Отладочные средства просмотра содержимого стеков и памяти.
- 5. Процедуры в ДССП. Приёмы определения новых процедур. Принципы структурированного программирования. Пошаговая нисходящая разработка программ в ДССП.
- 6. Организация ветвлений в ДССП. Команды условного вызова процедур. Операции для вычисления логических условий (выражений). Поразрядные операции троичной логики и арифметики. Примеры разработки простейших программ с ветвлением.
- 7. Организация циклов в ДССП. Простейшие команды многократного вызова процедур. Примеры разработки простейших программ с циклами.
- 8. Переменные и массивы в ДССП. Объявления для определения новых слов-переменных и слов-массивов. Простейшие типы данных. Простейшие операции над переменными и массивами. Примеры программ обработки массивов.
- 9. Доступ к памяти по адресам. Примеры разработки программ для обработки произвольных массивов, задаваемых в качестве параметров. Приёмы создания реентерабельных программ в ДССП.
- 10. Простейшие операции ввода и вывода с терминалом. Ввод и вывод символов, строк, чисел. Операции анализа символов и обработки строк. Операции преобразования строки в число и числа в строку. Примеры программ обработки текстовых строк.
- 11. Команды управления загрузкой, чисткой и сохранением словаря. Управление видимостью имён.
- 12. Словарь ядра и ДССП-библиотека. Характеристика ассортимента слов базового словаря и возможностей модулей ДССП-библиотеки. Простой диалоговый командный монитор (ДКМОН).
- 13. Команды доступа к двоичным файлам среды. Примеры разработки программ обработки файлов. Особенности обработки текстовых файлов.
- 14. Механизм обработки исключительных ситуаций. Объявление слов-ситуаций. Команды установки реакций на ситуации. Команды возбуждения ситуаций. Примеры разработки программ с применением механизма ситуаций.
- 15. Команды циклов с выходами путём возбуждения ситуации. Примеры разработки программ, применяющих циклы с выходами по ситуации.
- 16. Объявление новых типов как структур. Операции над объектами-структурами и отдельными их полями. Примеры программ с применением структур.
- 17. Объявление новых типов как классов. Определение новых операций как методов класса. Определение процедур реализации методов. Объявление класса-наследника. Переопределение методов. Спецоперации (AS, CONSTRUCT) для работы с объектами классов в произвольном участке памяти. Примеры определения новых типов как классов.
- 18. Сопрограммный механизм в ДССП. Объявление сопрограммы. Операция контекстного переключения. Примеры применения сопрограмм.
- 19. Простейший монитор управления параллельными процессами. Объявление процесса и операции с ним. Примеры применения монитора. Приёмы его реализации на основе сопрограммного механизма.
- 20. Определение слов в коде ассемблера ТВМ. Характеристика архитектуры и системы команд ТВМ. Особенности входного языка ассемблера ТВМ. Примеры реализации слов ядра ДССП на ассемблере.

- 21. Особенности сшитого кода ДССП для ТВМ. Строение тел процедур, констант, переменных, массивов. Операции доступа к содержимому управляющего стека. Приёмы определения особого вида слов с получением доступа к содержимому управляющего стека и позициям тел сшитого кода.

Литература, предлагаемая слушателям спецкурса:

1. Кнут Д. Искусство программирования на ЭВМ. Т. 2: Получисленные алгоритмы. —М.:Мир,1977. п. 4.1. с. 216–219.
2. Дал У., Дейкстра Э., Хоор К. Структур(ирован)ное программирование.— М.: Изд-во Мир, 1975.— 247 с.
3. Брусенцов Н. П., Рамиль Альварес Х. Троичные ЭВМ «Сетунь» и «Сетунь-70» // Труды 1-ой международной конференции «Развитие вычислительной техники в России и странах бывшего СССР: история и перспективы» SORUCOM-2006, г. Петрозаводск, Россия, 3–7 июля 2006 г.— Петрозаводск: Изд-во ПетрГУ, 2006.— Ч.1, с. 45–51.
4. Бурцев А. А., Сидоров С. А. История создания и развития ДССП: от «Сетуни-70» до троичной виртуальной машины // Труды 2-ой международной конференции «Развитие вычислительной техники и её программного обеспечения в России и странах бывшего СССР» SORUCOM-2011, г. Великий Новгород, Россия, 12–16 сентября 2011 г.— В.Новгород: Изд-во НовГУ, 2011.— С. 83–88.
5. Сидоров С. А., Владимирова Ю. С. Троичная виртуальная машина // Программные системы и инструменты. Тематический сборник №12.— М.: Изд-во ф-та ВМК МГУ, 2011, с. 46–55.
6. Бурцев А. А., Рамиль Альварес Х. Кросс-система разработки программ на языке ДССП для троичной виртуальной машины // Программные системы и инструменты. Тематический сборник №12.— М.: Изд-во факультета ВМК МГУ, 2011, с. 183–193.
7. Бурцев А. А., Бурцев М. А. ДССП для троичной виртуальной машины // Труды НИИСИ РАН, т. 2, № 1.— М.: Изд-во НИИСИ РАН, 2012.— С. 73–82.
8. Бурцев А. А., Рамиль Альварес Х. Реализация средств объектно-ориентированного программирования в кросс-компиляторе языка ДССП-Т // Программные системы и инструменты. Тематический сборник № 13.— М.: Изд-во факультета ВМК МГУ, 2012, с. 28–37.
9. Бурцев А. А., Шумаков М. Н. Сопрограммный механизм в ДССП как основа для построения мониторов параллельных процессов // Вопросы кибернетики. Сборник статей / ред. В .Б. Бетелин.— М.: Изд-во НИИСИ, 1999.— С. 45–63.
10. Бурцев А. ., Рамиль Альварес Х. Сопрограммный механизм в системе структурированного программирования для троичной машины // Программные системы и инструменты. Тематический сборник № 14.— М.: Изд-во факультета ВМК МГУ, 2013.— С. 193–208.
11. Баранов С.Н., Ноздрунов Н.Р. Язык Форт и его реализации.— Л.: Машиностроение, 1988.— 157 с.
12. Материалы по троичной информатике на Интернет-сайте Научно-Исследовательской Лаборатории Троичной Информатики (НИЛ ТИ) факультета ВМК МГУ. URL: http://ternarycomp.cs.msu.ru/ (дата обращения 20.10.2015).