ДССП для троичной виртуальной машины

А.А. Бурцев, к. ф.-м. н., доц.; М.А. Бурцев.

В НИЛ ЭВМ МГУ разработан программный комплекс ТВМ (Троичная Виртуальная Машина), имитирующий функционирование современного варианта троичного процессора двухстековой архитектуры. В статье представляются два варианта ДССП (Диалоговой Системы Структурированного Программирования) для ТВМ. ДССП-ТВМ - кросс-система, позволяющая создавать программы для ТВМ с помощью кросс-компилятора. ДССП/ТВМ - интерпретатор языка ДССП, функционирующий на ТВМ как резидентное ПО. Даётся характеристика их входного языка ДССП-Т как троичной версии языка ДССП.

1. Введение

В настоящее время, когда традиционная микроэлектроника, построенная на двоичной системе счисления, подходит к пределу своих технологических возможностей, в специальных изданиях и Интернете отмечается заметное повышение научной и инженерной активности в области исследований 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”, и которая была полвека назад успешно воплощена в троичных цифровых машинах (ЦМ) “Сетунь” и “Сетунь-70” [1]. 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;
- 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 вычисления.

В научно-исследовательской лаборатории (НИЛ) троичной информатики (ранее НИЛ ЭВМ) на факультете ВМК МГУ создан программный комплекс ТВМ (Троичная Виртуальная Машина) [2], имитирующий функционирование современного варианта троичного процессора двухстековой архитектуры с поддержкой структурированного программирования на уровне машинных команд [3], аналогичной той, что была обеспечена в троичной ЦМ “Сетунь-70”.

Предполагалось, что основным языком разработки программ для ТВМ станет ДССП-Т – троичный вариант языка ДССП [4-6]. Требовалось создать систему разработки программ для ТВМ на языке ДССП-Т. Эта работа была выполнена в два этапа.

Сначала (на 1-ом этапе) была построена кросс-система ДССП-ТВМ [7], позволяющая создавать программы для ТВМ на языке ДССП-Т с помощью кросс-компилятора. На 2-ом этапе, используя уже саму ДССП-ТВМ как среду разработки, был создан диалоговый интерпретатор ДССП/ТВМ, способный функционировать на троичной машине (ТВМ) в качестве её резидентной системы программирования.

2. Общая характеристика ДССП

Диалоговая система структурированного программирования (ДССП) была создана в начале 80-х годов XX века в проблемной лаборатории ЭВМ на факультете ВМК МГУ под руководством Брусенцова Н.П. ДССП была призвана существенно облегчить разработку ПО для широкого класса малых цифровых машин – мини- и микрокомпьютеров.

При создании ДССП преследовалось сразу несколько целей. Во-первых, ДССП создавалась в качестве программного имитатора (эмулятора) новой двоичной цифровой минимашины, в которой предполагалось воплотить ценные идеи двухстековой архитектуры и структурированных команд управления, унаследованные от модернизированной троичной ЦМ “Сетунь-70”. В роли такого имитатора или образно говоря виртуального процессора по сути и функционирует внутренний интерпретатор ДССП, являющийся сердцевиной (ядром) системы.

Во-вторых, ДССП создавалась как интегрированная среда разработки программ, легко доступная для освоения широкому классу пользователей микрокомпьютерной техники и персональных компьютеров. Поэтому при создании ДССП была заимствована идея словарной организации системы ФОРТ (Forth) [8] и обеспечен диалоговый характер взаимодействия с пользователем на всех стадиях разработки программы: начиная от её редактирования и вплоть до её окончательной отладки. Такой диалог обеспечивается внешним интерпретатором ДССП, функционирующим в режиме исполнения потока слов-команд.

Программа, составленная на входном языке ДССП, сначала преобразовывается во внутреннее представление, для характеристики которого применяется специальный термин – сшитый (threaded) код. Совокупность программных компонент, выполняющих такое преобразование, принято называть компилирующими средствами или просто компилятором системы. Внутрен-

ний и внешний интерпретатор, компилятор и словарь ядра, а также встроенные редактор текстов и отладчик вместе составляют базовый комплект ПО системы. На его основе путём наращивания словаря каждый пользователь может создать свой вариант расширения ДССП, отвечающим нуждам его прикладной области.

Программирование в ДССП заключается в наполнении её словаря новыми словами. Всякая операция (процедура), поддерживаемая системой, обозначается отдельным словом (слово – это любая последовательность печатных символов, отличных от пробела). Совокупность слов, понимаемых системой, и составляет ее язык (словарь). Система расширяется путем определения новых слов, которые сразу же после добавления в словарь могут быть использованы в системе.

Любое действие, совершаемое системой, вызывается словом. Предусмотренные в системе слова исполняют самые разнообразные функции. Они выполняют вычисления или обработку данных в стеке, доступ к памяти или к переменным, служат для организации ветвлений и циклов, обеспечивают построение тел для новых слов, вносимых в словарь, управляют состоянием словаря и даже самим процессом выполнения задания в системе.

ДССП характеризуется двухстековой архитектурой. Стек операндов, над которым действуют все операции по обработке данных, приводит к компактной польской инверсной записи арифметических выражений, а управляющий стек служит для сохранения адресов возвратов из процедур.

Следующие две характерные черты ДССП выгодно отличают её от системы ФОРТ: 1) специфическая процедурная версия структурированных команд управления; 2) и поддержка пошаговой (step-by-step) нисходящей (top-down) разработки программы.

В ДССП каждая из команд ветвлений вместо перехода по телу сшитого кода (как это обычно делается в системе ФОРТ) по сути осуществляет вызов процедуры слова, выбранного по условию, а каждая команда цикла организует многократный вызов процедуры слова, пока такой вызов допускается условием цикла. Таким образом, в ДССП поддержка структурированного программирования осуществляется непосредственно на уровне сшитого кода аналогично тому, как такая поддержка была обеспечена в ЦМ “Сетунь-70” на уровне машинных команд.

Определение нового слова в ДССП и в языке ФОРТ синтаксически выглядят одинаково:

: P P1 P2 **...** Pn **;** [определение процедуры с именем P]

Однако ФОРТ допускает подобное определение, только если все слова P1 P2 ... Pn уже были определены и стали известны системе. А ДССП допускает, что в этой последовательности слов P1 P2 **...** Pn могут встречаться в том числе и такие слова, которые еще не были определены и предполагается, что будут определены впоследствии. Именно благодаря такой особенности ДССП и позволяет разрабатывать программу методом нисходящего анализа: сначала определить главное слово, затем слова следующего уровня, из которых оно определяется, и так постепенно спуститься до детального определения самых простых слов.

Первая версия ДССП (ДССП-НЦ [4]) была создана для микрокомпьютеров “Электроника НЦ-03Д”. Уже с этой версии в ДССП нашли отражение характерные черты языка ассемблера ЦМ “Сетунь-70”: его структурированные команды управления для организации ветвления и цикла по условию: **BRS DW0N** .

На следующем этапе своего развития [9] ДССП создавалась уже для микрокомпьютеров так называемой унифицированной архитектуры (PDP-11). Причём наряду с основной версией ДССП-80 [5] разрабатывались также и экспериментальные версии, представляющие собой варианты развития ДССП, ориентированные на построение программ управления периферией (ДССП-ПМ [10]) и операционных систем реального времени (ДССП-РВ [11]).

В ДССП-80 были предложены новые команды **RP DO** для организации циклов с выходами по условию **EX- EX0 EX+** , а также типизированный набор префиксных операций: **! ’ ...** для единообразной работы с переменными и массивами различных типов. Была усовершенствована организация словаря (он стал перемещаемым, в нём появились подсловари и секции), позволившая упростить разработку сложных многомодульных программ. Для автоматизации сборки как самой ДССП, так и законченной прикладной программы, были предложены специальные инструментальные средства: компоновщик и целевой компилятор

Хотя экспериментальные версии ДССП-ПМ и ДССП-РВ не получили самостоятельного развития, предложенные в них новые программные механизмы были частично реализованы в последующих версиях ДССП [12,13]. Это механизм исключительных ситуаций, сопрограммный механизм, механизм прерываний на уровне сшитого кода, а также разнообразные средства взаимодействия параллельных процессов.

С начала 90-х годов ДССП стала активно использоваться в НИИСИ РАН в качестве инструментальной среды программирования для разработки внутреннего программного оснащения ряда компьютерных модулей, создаваемых на основе разных микропроцессоров (MC68020, R3000, Intel80386) [14].

В дальнейшем над развитием ДССП и распространением её на другие платформы трудились как сотрудники НИЛ ЭВМ МГУ вместе со студентами и аспирантами, так и сотрудники НИИСИ РАН [15]. В результате на протяжении ряда лет ДССП была реализована для компьютеров самой разнообразной архитектуры и в различных операционных средах. К середине 90-х годов ДССП функционировала почти на всех компьютерных платформах (PDP-11, Intel8080, Intel80x86, Motorola 68020, VAX, R3000) и в операционных средах (RT-11, RSX-11, UNIX, CP/M, MSDOS,Windows), получивших распространение в нашей стране.

Отдельные версии ДССП для разных платформ были плохо совместимы друг с другом. Их общий недостаток заключался в том, что ядро каждой из них создавалось на языке ассемблера базовой машины, что затрудняло перенос созданного ПО на другие компьютерные платформы. В 1998 г. в НИИСИ РАН была создана мобильная версия ДССП (ДССП/С), ядро которой было написано на языке Си [12]. При разработке ДССП/С была проведена такая модернизация работы с данными, благодаря которой в ДССП впоследствии были обеспечены возможности построения новых типов [16], аналогичные средствам объектно-

ориентированного программирования.

На протяжении почти 20-летнего периода своего развития [9] ДССП постоянно совершенствовалась, приобретая новые возможности, присущие современным языкам и системам программирования. В результате ДССП превратилась в среду программирования [13], позволяющую применять широкий спектр современных методов структурированной разработки сложных программных систем, сосредоточив в себе средства для структурированного, модульного, объектно-ориентированного и параллельного программирования, а также средства для структурированной обработки исключительных ситуаций.

3. Троичная виртуальная машина

Троичная виртуальная машина [2] (ТВМ, TVM - Ternary Virtual Machine) представляет собой программный эмулятор архитектуры троичного компьютера, основанного на троичной симметричной системе счисления и наследующего некоторые принципы троичных ЭВМ «Сетунь» и «Сетунь-70».

ТВМ включает в себя эмулятор троичного процессора, транслятор с языка ассемблера и интерактивный монитор для диалогового взаимодействия с пользователем, которые в совокупности образуют среду разработки и отладки программ.

Программный эмулятор работает на обычных двоичных компьютерах, полностью воссоздавая троичную среду на уровне машинных команд, регистрового файла, памяти и адресного пространства. Так что программист ТВМ имеет дело только с троичными объектами и не беспокоится об их реальном представлении.

В ТВМ используется троичная симметричная система счисления. Минимальной единицей информации в троичной машине является трит. Трит принимает одно из трёх значений: +1, 0, -1. Трайт в ТВМ состоит из 9 тритов. Основной единицей данных в ТВМ является слово, состоящее из 3 трайтов (27 тритов). ТВМ имеет двухстековую архитектуру. Для обработки данных используется стек данных. Арифметические и логические команды берут свои аргументы из стека данных и засылают в него обработанные результаты. Второй стек, называемый управляющим стеком или стеком возвратов, используется для сохранения адреса возврата из подпрограммы, сохранения контекста. Элементом стека возвратов является троичное слово.

Система команд включает в себя команды пересылок данных, арифметические и логические операции, группу команд для работы со стеком. Для управления ходом исполнения программы ТВМ предлагает команды, соответствующие основным управляющим конструкциям структурированного программирования: команды вызова подпрограммы, ряд команд для условного вызова одной из процедур и команду многократного вызова процедуры по условию.

4. Кросс-система ДССП-ТВМ

Система программирования ДССП-ТВМ предоставляет возможность разработки программ для троичного процессора на языке ДССП-Т и их исполнения на имитационной модели троичного процессора – ТВМ.

В ДССП-ТВМ в качестве составляющих её компонент используются следующие программные средства:

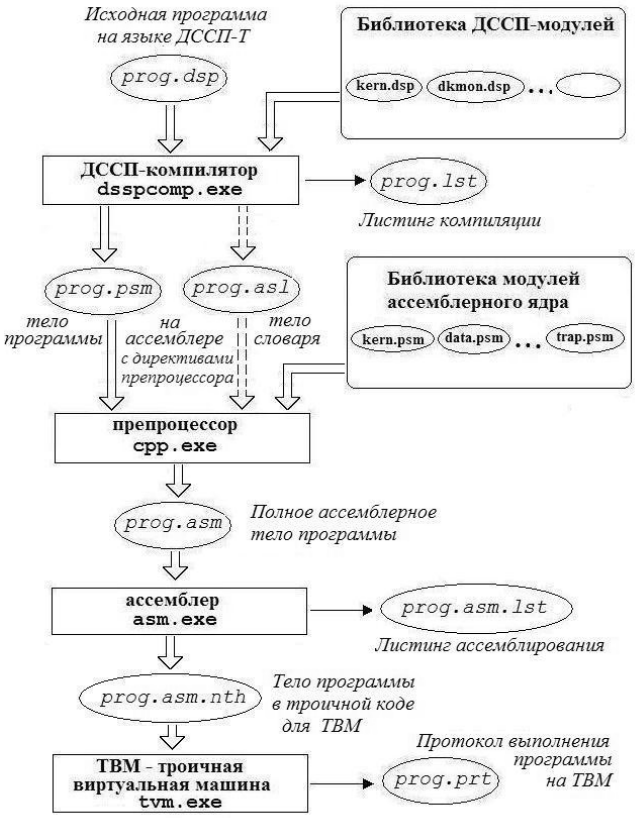

Рис.1. Основные компоненты ДССП-ТВМ и схема прогона ДССП-программы.

- 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, составленная на языке ДССП-Т, (файл **prog.dsp**) 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 за-

тем 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** .

5. Язык ДССП-Т

Язык ДССП-Т представляет собой ориентированный на троичную машину вариант языка ДССП, в котором удалось сохранить лучшие черты различных версий ДССП [4-6,10-15], разработанных для двоичных машин. Рассмотрим основные черты языка ДССП-Т, выделяя те его особенности, которые существенно отличают его от языка ДССП, описанного в [6].

Синтаксис языка ДССП-Т такой же простой, как и у обычной ДССП. Программа записывается как последовательность слов в свободном формате, разделителями слов являются пробелы (или приравненные к ним символы), а также комментарии. Словарь построен без разделения на подсловари и секции, но в нём можно осуществлять очистку, и обеспечена возможность помечать слова, которые всегда должны оставаться видимыми (неудаляемыми) в словаре.

В языке ДССП-Т допускаются разнообразные варианты комментаторных скобок так, чтобы никакие из символов: **([{}])** , употребляемых для комментариев, не запрещать программисту использовать в своих словах. Для настройки желаемого вида комментариев предусмотрены служебные слова: **() [] {}** . Допускаются также и комментарии, отделяемые двойными открывающимися скобками: **(([[{{** до конца текущей строки.

ДССП имеет двухстековую архитектуру. Все основные операции осуществляют действия над данными, помещёнными в стек операндов (арифметический стек). Одному элементу стека сопоставлено одно машинное слово из 27-тритов или 3-х трайтов. Стек операндов растёт в сторону увеличения адресов.

Управляющий стек (или стек возвратов) расположен в памяти так, чтобы расти навстречу стеку операндов (в сторону убывания адресов). Элементами стека являются 27-тритные слова, которые, как правило, трактуются как адреса возвратов из процедур. Предполагается, что помимо адресов возвратов в управляющий стек могут помещаться специального вида структуры – так называемые ловушки ситуаций.

Для объявления новых слов, представляющих те или иные сущности в ДССП-программе, используются специальные служебные слова, так называемые компилирующие слова или слова-компиляторы:

PROGRAM : VALUE VAR VCTR ARR TEXT STRUCT: ;STRUCT CLASS: SUBCLASS ;CLASS METHOD METHOD# METHOD= :M: :M= SITUATION ASM ;ASM

В языке ДССП-Т имя главного слова программы, с которого и предстоит начать исполнение всей программы, принято задавать в первой строке исходной программы после служебного слова **PROGRAM** :

PROGRAM DKINT

Слово-процедура в ДССП-программе определяется следующим образом:

: P P1 P2 ... PN **;**

При своём выполнении такая процедура P будет последовательно вызывать исполнение слов, из которых построено её тело: P1 ... PN. Причём в ДССП-программе не требуется, чтобы все используемые в таком определении слова должны уже быть определены заранее. В определяемом теле новой процедуры разрешается употреблять и те слова, которые могут быть определены позднее.

В общем виде объявление переменной в ДССП-программе выглядит так:

ИмяТипа **VAR** ИмяПерем {Пример:} TRYTE **VAR** X

Тип переменной зависит от того, какое слово с именем типа употреблялось перед её объявлением. Если тип не был задан, то он принимается по умолчанию (в настоящее время это тип TWORD , задающий троичное слово).

При объявлении вектора (одномерного массива):

{N} **VCTR** ИмяВектора {Пример:} 10 **VCTR** Y

в стеке задаётся его верхняя граница N (нижняя граница = 0). Так что в данном примере объявляется вектор Y[0:N] с N+1 элементом.

При объявлении многомерного массива вида:

{L1,..,Lm,m} **ARR** ИмяМас {Пример:} 10 20 2 **ARR** M

в стеке задаются его размерность m и m верхних границ по каждому измерению. Так что в примере объявляется массив M[0:10,0:20].

Тип элементов объявляемого массива определяется тем словом с именем типа, которое было употреблено непосредственно перед объявлением.

В текущей версии языка ДССП-Т (v.3a) предусмотрен лишь минимальный набор префиксных операций над переменными и массивами: операции взятия значения (по имени), адреса (**‘**) и присвоения значения (**!**). А в базовом словаре языка для объявления переменных и в качестве элементов массивов предусмотрены лишь слова-типы, представляющие возможность работать с трайтами, двухтрайтовыми значениями и троичными словами: **TRYTE DTRYTE TWORD .**

В языке ДССП-Т помимо десятеричных и 16-ных можно задавать также троичные и 9-ные числа, характеризующие значения в так называемой троичной симметричной системе счисления. Каждое из таких чисел опознаётся по специальным префиксам:

{девятеричные числа:} #8765 **0t**qwer **0T**QWER #1234
{троичные числа:} **.**++0--**. .**++0-- **0n**++0-- **0N**++0--
{16-ные числа:} **$**13579ace **0x**1290abcd **0X**345678EF
{10-ные числа:} 56789 +24680 -13579

Большие и малые латинские буквы в записи числа не различаются, а для представления отрицательных девятеричных цифр (-1,-2,-3,-4) можно использовать как цифры (8765), так и буквы (qwer или QWER).

Для представления символьных констант используется общепринятая форма их записи в апострофах ‘X’ (как в языках Си и Паскаль).

Хотя числовые константы-литеры автоматически распознаются (по своим специфическим изображениям) как особые слова, в языке ДССП-Т можно объявить слово, представляющее константу, придав ему нужное имя:

{N} **VALUE** ИмяКнст {Пример:} 10 **VALUE** Ten

При таком объявлении в стеке задаётся то значение константы N(10), которое будет помещаться в арифметический стек при вызове слова-константы (Ten).

В языке ДССП-Т можно определить и слово, задающее константу-строку:

TEXT ИмяКС Строка

{Пример:} **TEXT** Red “Красный”

При вызове имени такой константы-строки в арифметический стек будут помещаться её параметры: адрес начала строки и длина (количество символов).

В языке ДССП-Т сохранены привычные обозначения слов для операций целочисленной арифметики, но выполняются они над числами, представленными в троичной симметричной системе счисления, так что результаты некоторых операций (например, сдвигов и делений с остатком), могут оказаться иными, чем в двоичной машине:

{деление с остатком:} 47 / 3 { [16,-1] вместо [15,2] }
{сдвиг вправо= /3 :} 47 SHR { [16] вместо [23] }
{сдвиг влево = *3 :} 13 SHL { [39] вместо [26] }

В языке ДССП-Т нет слов (**& &0 ‘+’ INV**), выполнявших привычные действия над отдельными битами машинного слова. Вместо них присутствуют слова, позволяющие выполнять разнообразные действия над тритами машинного слова:

TMIN TMAX TADD TMUL NEG .

Для доступа к одиночным и двойным трайтам, а также к словам троичной памяти в ДССП-Т предусмотрены операции чтения: **@T @TT @W** и записи: **!T !TT !W** . Причём для продвижения по памяти к очередному троичному слову надо добавлять 3 к значению адреса.

В языке ДССП-Т представлен полный ассортимент ДССП-команд условного вызова процедур, предусмотренных для организации ветвлений в ДССП-программе: **IF- IF0 IF+ BR- BR0 BR+ BRS BR ELSE** .

Чтобы обеспечить возможности вычисления логических условий, эквивалентные средствам построения логических выражений языков высокого уровня, в ДССП-Т имеются операции отношений для сравнения двух чисел: **< <= = <> >= >** , а также операции логических связок: **NOT AND OR** , вырабатывающие значение 1(ДА) или 0 (НЕТ).

Кроме того, в языке ДССП-Т предусмотрены также слова-операции для сравнения двух чисел и оценки знака числа: **CMP SGN** , предусматривающие троичные результаты: -1 (меньше), 0 (равно), +1 (больше), которые далее могут использоваться для вычисления сложного троичного условия путём применения потритных операций: **TMIN TMAX TADD TMUL NEG** .

В ходе создания и развития ДССП применялись две различные концепции организации циклов. Первая основывается на том, что условие цикла должно задаваться в самой команде цикла. Вторая предполагает, что выход из цикла может осуществляться с любого места тела цикла по специальным командам, а сама команда цикла должна лишь обеспечивать повторный вызов процедуры её тела.

Первая концепция обозначилась командой **DWON** уже в первой версии ДССП-НЦ [4], которая по сути унаследовала команду цикла DW машины “Сетунь-70”. Эта команда многократно вызывала стоящую следом процедуру, пока вершина стека оставалась ненулевой. Затем было предложено усовершенствовать эту команду цикла и выделить вычисление условия в отдельную процедуру, а значение условия удалять из стека после его проверки. Так в ДССП [10] появилась конструкция: **WHILE условие DO тело**, которая впоследствии (в ДССП/С [13]) превратилась в команду: **условие DW тело**. Вместе с ней были предложены команда **DO-** для организации цикла со счётчиком, а также команда бесконечного цикла **LOOP**, которые исполняют эти виды циклов эффективней, чем универсальная команда **DW**.

Вторая концепция организации циклов была предложена в версии ДССП-80 [5]. В ней появились команды для повторного вызова процедуры тела цикла заданное или бесконечное число раз: **DO RP** . Были предусмотрены специальные команды для экстренного выхода из цикла по условию или безусловно: **EX- EX0 EX+ EX** . Эти команды могли осуществлять требуемый выход из цикла с любой точки как самой процедуры тела цикла, так и вызванных ею процедур.

Обе концепции организации циклов применялись в ДССП на протяжении достаточно длительного периода и потому стали привычными для многих разработчиков ДССП-программ. Чтобы предоставить каждому из них желаемый способ организации циклов, в языке ДССП-Т обеспечены команды циклов обеих этих концепций: **DW DO- LOOP RP DO EX EX+ EX- EX0** .

В языке ДССП-Т поддерживается полный вариант (как в ДССП/С [13]) механизма работы с исключительными ситуациями. Предусмотрены команды: **ON EON RON** для установки трёх типов реакций (Notify, Escape, Retry) на заданную (следом по телу) ситуацию; на ситуацию, переданную через стек: **ON_ _ESCAPE _NOTIFY _RETRY** ; на произвольную ситуацию **ANY** ; а также команда для повторного возбуждения ситуации **RERAISE .**

Объявлять ситуацию можно так:

SITUATION ИмяСит КнРеак

При таком объявлении помимо имени ситуации задаётся также имя слова-процедуры конечной реакции, которая будет вызвана в случае, когда возбуждённая ситуация не будет обработана в ДССП-программе.

Программа, разработанная в ДССП-ТВМ, должна уметь общаться с внешними устройствами, которые ей может предоставить имеющаяся в настоящее время среда исполнения. В качестве минимального набора устройств ввода и вывода, доступных исполняемой ДССП-программе, в настоящее время ей обеспечивается диалоговое общение через консоль терминала и операции доступа к файлам операционной среды для последовательного чтения и записи.

При построении тел некоторых новых слов ДССП-программы может возникнуть потребность запрограммировать такие действия, выразить которые возможно

удастся лишь командами базовой машины (т.е. командами ТВМ). Поэтому язык ДССП-Т предоставляет возможность определить новое ДССП-слово, задав процедуру его исполнения на языке ассемблера ТВМ. Такое определение задаётся между служебными словами **:ASM** и **;ASM** .

Объявление нового типа как структуры начинается словом **STRUCT:** и завершается словом **;STRUCT** , а располагаемые между ними определения объектов данных (с помощью слов **VAR VCTR ARR**) объявляют в этом случае не самостоятельные переменные и массивы, а поля определяемой структуры:

STRUCT: ИмяТипаСтрукт **VAR** ИмяПоля1 . . . **VAR** ИмяПоляK **;STRUCT**	**STRUCT:** TPoint TRYTE VAR .X TRYTE VAR .Y **;STRUCT**

После объявления нового типа данных можно использовать его имя для объявления объектов (переменных и массивов) такого типа:

```
TPoint VAR P      TPoint 100 VCTR V {вектор точек}
```

Имеющиеся в языке ДССП-Т префиксные операции над переменными и массивами можно теперь применять и для доступа как ко всему объекту типа структуры целиком, так и к отдельным его полям:

P {‘P} 3 ! V {V[3]:=P} 5 V ! P {P:=V[5]} i V j ! V {V[j]:=V[i]}	P .X {P.X} 55 P ! .Y {P.Y:=55} P .X 6 V ! .X {V[6].X:=P.X} 8 V .Y P ! .Y {P.Y:=V[8].Y}

Ещё одним средством построения новых типов данных в языке ДССП-Т являются классы. Объявление нового типа как класса начинается со слова **CLASS:** и заканчивается словом **;CLASS** , а внутри такого объявления можно задавать не только поля (как и при объявлении структуры), но и методы класса:

CLASS: ИмяТипаКласса **VAR** ИмяПоля1 . . . **VAR** ИмяПоляK **METHOD** ИмяМетодаI . . . **METHOD** ИмяМетодаJ **;CLASS**	**CLASS:** TPoint **VAR** .X **VAR** .Y **METHOD** Init **METHOD** Show **METHOD=** Show Print 4 **METHOD#** >> **;CLASS**

Словом **METHOD** объявляется новая операция, которую можно будет совершать над объектом этого типа-класса помимо тех методов (префиксных операций **!** и **‘**), которые над ним уже предусмотрены изначально (как и над структурами). Слово **METHOD=** позволяет назначить новое имя методу, используя его прежнее имя, а слово **METHOD#** – задав номер метода. Так в примере для метода Show было назначено два новых имени: Print и >> .

Далее необходимо с помощью компилирующих слов **:M:** и **:M=** определить процедуры, которые будут назначены для исполнения новых методов над объектами объявленного класса:

```
TPoint :M: Init {P} 0 C2 ! .X 0 E2 ! .Y { } ;
: .Show {P} ‘(’ TOB C .X . D ‘,’ TOB .Y . D ‘)’ TOB {};
TPoint :M= Show .Show
```

Имя типа класса следует задать перед употреблением слова **:M:** (или **:M=**). Слово **:M:** создаёт тело новой процедуры, компилируя его как и слово **:** (до **;**), а слово **:M=** принимает имя уже существующей процедуры (.Show), чтобы назначать её в качестве исполнителя метода.

Внутри объявления нового типа как класса с помощью слова **SUBCLASS** можно задать, какой уже известный тип использовать для наследования:

```
CLASS: T3DPoint SUBCLASS TPoint VAR .Z ;CLASS
T3DPoint :M: Init {P} C Init AS TPoint 0 E2 ! .Z ;
T3DPoint :M: Show {P} {реализуем по-новому:}
 ‘[’ TOB C .X . D ‘,’ C .Y . D ‘,’ .Z . D ‘]’ TOB ;
```

Такое наследование означает, что у нового объявляемого типа изначально считаются уже заданными все поля и методы, которыми обладает наследуемый тип, после чего можно добавлять к его определению новые поля, задавать новые методы, а также переопределять исполнителей прежних методов (Show).

6. ДССП-библиотека

Возможности языка ДССП-Т не ограничиваются только словами базового словаря. Словарь компонуемой программы можно дополнить определениями слов, заготовленных в файлах стандартной библиотеки (ДССП-библиотеки). Для этого достаточно загрузить их определения из требуемого файла командой:

LOAD Имя_Файла

В настоящее время ДССП-библиотека состоит из двух частей: ДССП-ядра (файл kern.dsp) и диалогового командного монитора (файл dkmon.dsp).

Ядро ДССП складывается как бы из двух частей: первая часть содержит тела процедур ядра, составленные на языке ассемблера ТВМ, а вторая - определения слов ядра на языке ДССП-Т. Ассортимент этих слов весьма разнообразен и достаточно обширен. В настоящее время он охватывает:

- определения ряда констант;
- операции вывода на терминал управляющих символов, строк, чисел ;
- операции ввода с терминала строк;
- операции преобразования строки в число заданного формата;
- операции преобразования числа в строку заданного формата;
- операции сравнения и копирования строк;
- операции проверки символов;
- операции над нумерованными тритами троичного слова;
- определения новых типов данных **ACT** и **TSET** с операциями над ними;
- операции с файлами операционной среды ТВМ;
- а также разнообразные вспомогательные операции.

Диалоговый Командный МОНитор (далее ДКМОН) предназначен для исполнения скомпонованной ДССП-программы в режиме привычного диалога. Такой диалог предполагает, что в ответ на стандартное ДССП-приглашение (в виде символа * с новой строки) можно последовательно ввести в командной строке и исполнить любое слово скомпонованной программы (или несколько слов), просмотреть содержимое стека операндов, стека возвратов, ячеек памяти.

ДКМОН не обеспечивает всех тех функций, которыми обычно должен обладать внешний интерпретатор ДССП. Кроме слов, приготовленных в компонуемой программе, он умеет распознавать только числа (в этом случае в качестве действия он посылает в стек

значение принятого числа). Ни комментарии, ни строки вида **"…"**, ни текст для печати вида **."…"** он не воспринимает. На любое непонятное слово он печатает сообщение об ошибке.

ДКМОН принимает слова из командной строки по одному. Встретив известное слово, ДКМОН сначала проверяет, можно ли это отдельное слово исполнить. Если нельзя, сообщает об ошибке, что это неисполнимое слово.

ДКМОНу известны только те слова, которые остались видимыми в словаре после компоновки программы. Из них исполнимыми считаются только те слова, которые могут выполняться в одиночку. В частности, ДКМОН не позволит исполнить введённую в командной строке последовательность слов, содержащую какие-либо управляющие команды: **IF+ BR- BRS DW DO RP** или префиксные операции **` !** над переменными, либо какие-то другие слова, требующие доступа к последующим словам по телу сшитого кода.

Модуль ДКМОНа можно использовать и просто как ДССП-библиотеку, загрузив в программу содержащиеся в нём определения слов (командой `LOAD`) без обязательного последующего запуска монитора при старте программы.

7. Интерпретатор ДССП/ТВМ

При разработке интерпретатора ДССП/ТВМ предполагалось обеспечить совместимость его входного языка с той текущей версией (v.3a) языка ДССП-Т, которая в настоящее время поддерживается кросс-компилятором ДССП-ТВМ. Такая совместимость означает, что интерпретатор должен уметь воспринимать и обрабатывать любую ДССП-программу, ранее созданную с помощью кросс-компилятора. И обеспечивать при этом точно такой же эффект исполнения данной программы при её запуске в среде интерпретатора, который проявлялся ранее при её прогоне в кросс-системе ДССП-ТВМ, где она могла запускаться на ТВМ автономно или вызываться в диалоговом командном мониторе.

ДССП-программа подаётся на вход интерпретатору как последовательность слов, которую он должен суметь проанализировать за один проход. Основной цикл ДССП-интерпретатора заключается в приёме очередного слова из входного потока и исполнении действия, соответствующего этому слову.

Интерпретатор поддерживает словарь известных ему слов. Каждому такому слову словаря сопоставлена процедура, которую интерпретатору надлежит вызывать, когда он встретит такое слово во входном потоке.

Помимо слов, накопленных в его словаре, интерпретатор умеет также распознавать особые слова-литеры, представляющие изображения чисел, символов, строк текста. Интерпретатор заранее знает, какое следует исполнить действие, встречая слово-литеру. Для числовой константы требуется занести в стек изображаемое ею значение, для символьной – её код, для обычной строки – поместить в стек её адрес и длину, а для строки, изображающей печатный текст, требуется напечатать этот текст.

Все слова, вызывающие исполнение тех или иных функций интерпретатора, можно условно разделить на несколько групп.

7.1. Средства построения тела программы

Основная задача, которую должен уметь выполнять интерпретатор языка ДССП, заключается в том, чтобы переводить анализируемый текст задаваемой во входном потоке ДССП-программы во внутреннее представление (в виде так называемого сшитого кода) с тем, чтобы запускать потом исполнение этого сформированного кода на троичной машине (ТВМ) при поддержке процедур ассемблерного ядра ДССП.

Предполагается, что программа, подаваемая во входном потоке интерпретатору, построена с использованием того же ассортимента компилирующих слов, которые предусмотрены в языке ДССП-Т и которые уже умеет понимать кросс-компилятор системы ДССП-ТВМ. Такие слова-компиляторы предназначены для объявления (определения) новых сущностей в программе. Каждое из них даёт указание интерпретатору создать новое слово в словаре и сформировать соответствующее ему тело во внутреннем представлении:

- слово **:** определяет новую процедуру в ДССП-программе;
- слово **VAR** определяет новую переменную;
- слово **VCTR** определяет одномерный массив;
- слово **ARR** определяет многомерный массив;
- слово **VALUE** определяет числовую константу;
- слово **TEXT** определяет константу-строку;
- слово **SITUATION** определяет исключительную ситуацию.

Обрабатывая эти слова, кросс-компилятор формировал тело компонуемой программы в виде файла ассемблерного кода. А интерпретатор, обрабатывая такие слова, должен формировать тело принимаемой программы во внутреннем представлении в особой области памяти, называемой областью тел.

Комплект процедур, обеспечивающих реализацию действий слов-компиляторов, как раз и позволяет интерпретатору осуществлять свою основную задачу. Этот комплект и составляет, по сути, основное ядро, образно говоря, сердцевину интерпретатора.

Встречая во входном потоке компилирующее слово из перечисленного выше набора, интерпретатор должен выполнить ряд характерных действий: 1) создать в словаре заголовок для вновь определяемого слова; 2) сформировать для него тело во внутреннем представлении и сохранить адрес этого тела в заголовке слова; 3) распределить память для объекта (переменной или массива) в особой области данных.

7.2. Слова управления входным потоком

В качестве основного способа взаимодействия с пользователем ДССП-интерпретатор предлагает диалоговый режим работы. Это значит, что входной поток слов, предназначенных для исполнения, интерпретатору можно задавать вводом строки с терминала, а получать от него ответы – в виде сообщений на терминал. Первоначально после своего запуска и всякий раз, когда интерпретатор входит в диалоговый режим, он вы-

даёт приглашение (обычно в виде символа * с новой строки), означающее, что он ожидает от пользователя ввода очередных слов-команд с терминала.

Обычно с терминала удобно задавать короткие последовательности команд, чтобы тут же в ходе диалога наблюдать эффект их исполнения в среде интерпретатора. Но вводить с терминала весь тот огромный поток слов, с помощью которого строится достаточно большая ДССП-программа, не представляется разумным. Такую программу лучше предварительно подготовить в виде отдельного текстового файла, а потом подавать целиком на обработку интерпретатору.

Чтобы воспринимать входной поток с файла, в интерпретаторе предусмотрена операция:

LOAD ИмяФайла {Пример:} **LOAD** MyProg

Эта операция запоминает параметры, характеризующие текущее состояние входного потока, и переключает интерпретатор на чтение входного потока слов из заданного файла. Предполагается, что, завершив чтение слов из этого файла, интерпретатор вернётся к обработке прежнего входного потока.

Интерпретатор воспринимает поток слов из файла, пока не встретит в нём специальное завершающее слово (**$$**) или не достигнет конца файла. Если при обработке очередного файла снова встретится слово-операция **LOAD**, то интерпретатор переключится на чтение нового потока из другого файла, завершив который, вернётся к продолжению обработки слов предыдущего файла. Т.е. интерпретатор допускает возможность вложенных вызовов для операции **LOAD**.

С помощью специальных слов: **{} [] ()** можно указать интерпретатору, какой вид комментариев учитывать далее при обработке входного потока. Текущий установленный вид комментариев запоминается интерпретатором при переключении входного потока на другой файл и восстанавливается при возвращении на обработку прежнего потока.

7.3. Слова управления состоянием словаря

Словарь ДССП-программы является одной из важнейших её компонент. Почти все компилирующие операции добавляют в него новые слова. Но в языке ДССП-Т предусмотрены также и особые слова-операции для управления словарём:

- операция :: помечает следующее определяемое слово так, чтобы оно всегда оставалось в словаре;
- операция **CLEAR** очищает словарь, оставляя в нём лишь заголовки тех слов, которые специально были помечены (операцией **::**) как неудаляемые; заметим, что удаляя из словаря заголовки остальных слов, интерпретатор оставляет всё же их тела;
- операция **UNDEF** позволяет узнать, остались ли на данный момент в словаре неопределённые слова (т.е. слова, которые использовались в описании других слов, но сами описаны не были).

Диалоговый интерпретатор дополняет язык ДССП-Т новыми операциями, которые позволяют запоминать состояние словаря, достигнутое в ходе диалога, так, чтобы впоследствии можно было бы восстановить нужное состояние словаря (запомненное ранее) перед тем, как продолжать дальнейшую работу. Необходимость в таком восстановлении требуемого состояния словаря возникает, например, когда требуется повторно загрузить из файла (командой LOAD) ту программу, которая уже когда-то ранее загружалась в ходе текущего диалогового сеанса работы с интерпретатором. (Предполагается, что эта программа содержала ошибки, которые удалось обнаружить и исправить).

В языке ФОРТ [8] и в версии ДССП-НЦ [4], которая имела такое же простое линейное строение словаря, для восстановления состояния словаря использовалась операция **FORGET** с указанием имени обычного слова, которая удаляла из словаря это слово вместе со всеми словами, созданными после него.

Интерпретатор ДССП/ТВМ обеспечивает аналогичную возможность, но разрешает применять операцию **FORGET** не с обычным, а со специальным словом (назовём его маркером), которое создаётся в словаре (операцией **GROW**) именно с целью фиксации в его теле параметров, характеризующих текущее состояние словаря. Восстанавливая значения этих параметров из тела указанного слова, операция **FORGET** сумеет вернуть словарь в то состояние, которое было зафиксировано на момент создания этого слова.

Чтобы повторно загружаемая (командой **LOAD**) ДССП-программа могла корректно обрабатываться интерпретатором, обычно в её начало помещают последовательность команд **FORGET** и **GROW** с одним и тем же словом: **FORGET** $PROG **GROW** $PROG .

Для краткой записи таких действий предлагается команда: **PROGRAM** $PROG . Это позволяет интерпретатору корректно обрабатывать любые программы, разработанные ранее в системе ДССП-ТВМ.

7.4. Операция сохранения тела и словаря

Диалоговый интерпретатор обладает ещё одной важнейшей операцией, которая совсем не предусматривалась во входном языке кросс-компилятора. Но такая операция является характерной особенностью диалоговой среды разработки интерпретируемых программ. Она имеется во всех интерпретируемых версиях языков ФОРТ и ДССП и обычно называется **SAVE** :

SAVE ИмяФайла {Пример:} **SAVE** mydssp

Эта операция позволяет сохранить результаты диалогового сеанса общения с интерпретатором так, чтобы начать с ним следующий диалоговый сеанс в том состоянии, в котором был завершён предыдущий. Для этого в указанном файле сохраняются все компоненты, характеризующие состояние диалоговой среды ДССП-интерпретатора: программный код самого интерпретатора, область его системных переменных, словарь и область сформированных тел ДССП-программы.

Существенно, что такой файл обычно формируется в формате, пригодном для последующей его загрузки в память с целью непосредственного исполнения на той же машине или в той же операционной среде, в которой и запускался ранее диалоговый интерпретатор. Поэтому можно сказать, что операцией **SAVE** на самом деле создаётся файл новой версии интерпретатора, в словарь которой каждый пользователь может включить свои новые слова, созданные им в результате проведённого сеанса работы.

Интерпретатор ДССП/ТВМ, выполняя операцию сохранения, создаёт файл в формате троичного кода (nth-файл), который впоследствии может быть загружен для исполнения троичной машиной (ТВМ).

7.5. Операции для создания новых слов-компиляторов

При разработке процедур исполнения компилирующих слов в интерпретаторе были предусмотрены вспомогательные слова-операции, которые могут оказаться полезными для разработчиков ДССП-программ, если им потребуется создавать свои собственные слова-компиляторы.

Так, например, для работы с формируемым телом уже предусмотрен следующий набор вспомогательных операций:

- операция **,** заносит в формируемое тело троичное слово, взятое из стека;
- операция **,T** заносит в формируемое тело трайт, взятый из вершины стека;
- операция **,Str** копирует в формируемое тело строку из трайтов, адрес и длина которой задаются в стеке;
- операция **,WBlk** копирует в формируемое тело блок троичных слов, адрес и длина (количество слов) которого задаются в стеке;
- операция BA засылает в стек адрес текущей позиции формируемого тела.

Другой набор вспомогательных операций позволяет прочитать слово из входного потока и выполнить с ним необходимые действия в словаре:

- операция **GETWORD** считывает следующее слово из входного потока;
- операция **_WORD** выдаёт в стек адрес и длину прочитанного слова;
- операция **FINDWORD** производит поиск в словаре введённого ранее слова;
- операция **WORD?** считывает слово и осуществляет его поиск по словарю.
- операция **NewWord** добавляет в словарь заголовок нового слова с заданной строкой его имени и заданной для него командой (адресом тела) в стеке;
- операция **HEAD** считывает слово из входного потока и подготавливает его заголовок в словаре для формирования тела этого слова;
- операция **WORDCODE** считывает слово из входного потока, находит его в словаре и помещает в стек сопоставленную ему команду (адрес тела).

В качестве вспомогательных в словаре интерпретатора предусмотрены даже такие операции, которые осуществляют компиляцию тела как для отдельного слова (**COMPILEWORD**), так и для последовательности слов (**COMPILEBODY**), заканчивающейся **;** . А также ряд операций, помогающих выделить память (**MUSE**) и назначить адрес (**LOCATE**) для объявляемого объекта данных.

7.6. Слова для определения новых типов

Интерпретатор обеспечивает средства построения новых типов данных как структур и классов, появившиеся в новой версии (3а) языка ДССП-Т.

Встречая компилирующее слово **STRUCT:**, он создаёт слово-имя для нового типа данных и переключается в особый режим обработки слов, объявляющих данные, пока не встретит слово **;STRUCT**, завершающее определение нового типа. В этом особом режиме он создаёт для объявляемых объектов данных такие тела, которые позволяют им функционировать не в качестве самостоятельных переменных, а выступать в роли полей объявляемой структуры. Завершая обработку структуры, интерпретатор формирует в области тел дескриптор для нового типа данных.

Создание нового типа данных как класса интерпретатор осуществляет, обрабатывая целый комплект компилирующих слов: **CLASS: SUBCLASS ;CLASS METHOD METHOD# METHOD= :M: :M=** .

Но в этом случае помимо слова-имени для нового типа и слов-имён полей он создаёт ещё и слова с именами новых методов, объявляемых в определении класса, а также тела процедур, назначаемых в качестве их исполнителей.

Формируя дескриптор для нового типа-класса, интерпретатор сначала заимствует дескриптор наследуемого типа-класса, а затем расширяет его, добавляя позиции для новых методов. Впоследствии интерпретатор заполняет эти позиции ссылками на тела процедур-исполнителей методов, когда обрабатывает слова-компиляторы **:M: :M=** .

7.7. Усовершенствованные возможности командного монитора

Созданный интерпретатор не только поддерживает все функции по прогону построенной ДССП-программы, который умел ранее выполнять диалоговый командный монитор ДКМОН, но и обеспечивает для этого ряд новых возможностей, которыми ДКМОН не обладал.

При разработке интерпретатора были преодолены отмеченные ранее недостатки монитора. Так что интерпретатор умеет корректно распознавать и комментарии, и любые слова-литералы языка ДССП-Т, в том числе: большие числа, строки, печатный текст.

Существенно, что интерпретатор позволяет исполнять в режиме монитора так называемые Р-слова, т.е. особые слова, помеченные Р-флагом, которые требуют для своего исполнения доступа к следующему слову по телу сформированного сшитого кода. Для исполнения таких Р-слов в режиме монитора интерпретатор создаёт на некоторое время тело невидимой процедуры (без имени), а затем запускает её исполнение.

Большинство Р-слов требуют доступа лишь к одному последующему слову. Поэтому, встретив в командной строке слово, помеченное Р-флагом, интерпретатор помещает в тело такой временно создаваемой процедуры результат компиляции самого этого слова вместе со словом, следующим за ним, а также команду возврата, после чего запускает исполнение этого тела.

Однако, такой алгоритм обработки Р-слов не позволяет интерпретатору, вообще говоря, корректно исполнять (в режиме монитора) слова, требующие доступа более чем к одному слову по телу сшитого кода. Например, было невозможно исполнять последова-

тельности слов, содержащие управляющие команды ветвления и цикла: **BR+ BRS BR DW** .

Чтобы обеспечить исполнение всевозможных Р-слов в режиме монитора, интерпретатор предоставляет дополнительно слово-операцию с именем **;:** , которая создаёт на время процедуру без имени из последовательности слов, заканчивающейся, как и обычная процедура, словом **;** , а затем исполняет её.

8. Заключение

В работе представлены ДССП-ТВМ - кросс-система разработки программ для троичной машины (ТВМ) и диалоговый интерпретатор ДССП/ТВМ, способный функционировать на троичной машине в качестве её резидентной системы программирования.

Хотя ДССП-ТВМ сейчас функционирует в среде ОС Windows, её можно легко перенести в другую операционную среду (Linux), т.к. кросс-компилятор, ассемблер и имитатор ТВМ написаны на языке Си.

Опыт создания интерпретатора ДССП/ТВМ в кросс-системе разработки ДССП-ТВМ подтверждает, что эта кросс-система (как и созданный интерпретатор) могут служить основой для построения полноценного первичного программного оснащения аппаратно реализованной троичной машины той же архитектуры.

Литература

1. 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.

2. Сидоров С.А., Владимирова Ю.С. Троичная виртуальная машина // Программные системы и инструменты. Тематический сборник №12, М.: Изд-во факультета ВМиК МГУ,2011. С.46-55.

3. Брусенцов Н.П., Рамиль Альварес Х. Структурированное программирование на малой цифровой машине. // Вычислительная техника и вопросы кибернетики. Вып. 15. М.: Изд-во МГУ, 1978, с.3-8.

4. Брусенцов Н.П., Златкус Г.В, Руднев И.А. ДССП - диалоговая система структурированного программирования. // Программное оснащение микрокомпьютеров. М.: Изд-во МГУ, 1982, с.11-40.

5. Брусенцов Н.П., Захаров В.Б., Руднев И.А., Сидоров С.А. Диалоговая система структурированного программирования ДССП-80. // Диалоговые микрокомпьютерные системы. М.: Изд-во МГУ, 1986, с.3-21.

6. Брусенцов Н.П., Захаров В.Б., Руднев С.А., Сидоров С.А., Чанышев Н.А. Развиваемый адаптивный язык РАЯ диалоговой системы программирования ДССП. М.: Изд-во Моск. Ун-та, 1987 г. – 80 с.

7. 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 ВМиК МГУ,2011. С.183-193.

8. Баранов С.Н., Ноздрунов Н.Р. Язык Форт и его реализации. – Л.: Машиностроение, 1988. – 157 с.

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 (12-16 сентября 2011 г., г. Великий Новгород, Россия): Труды конференции. В.Новгород: Изд-во НовГУ, 2011. с. 83-88.

10. Бурцев А.А. Периферийный монитор – развитие архитектуры ввода/вывода ДССП. // Диалоговые микрокомпьютерные системы. М.: Изд-во МГУ, 1986, с.42-51.

11. Борисов А.В. Диалоговая система структурированного программирования в реальном времени – ДССП-РВ. // Диалоговые микрокомпьютерные системы. М.: Изд-во МГУ, 1986, с.51-62.

12. Бурцев А.А., Франтов Д.В., Шумаков М.Н. Разработка интерпретатора сшитого кода на языке Си. // Вопросы кибернетики. Сб. статей под ред. В.Б.Бетелина. М., 1999. с.64-76.

13. Бурцев А.А. ДССП – среда структурированной разработки программ как сложных систем. // Вторая Международная конференция “Системный анализ и информационные технологии” САИТ-2007 (10-14 сентября 2007 г., Обнинск, Россия): Труды конференции. Изд-во ЛКИ, 2007. т. 2. с.190-194.

14. Лякина Е.Н., Пшеничный К.А., Сидоров С.А., Шумаков М.Н. Система внутреннего программного оснащения DPROM. // Вопросы кибернетики. Сб. статей под ред. В.Б.Бетелина. М., 1999. с.77-86.

15. Сидоров С.А., Шумаков М.Н. ДССП как открытая система. // Дискретные модели. Анализ, синтез и оптимизация. Спб.: СпбГУ, 1998. с.191-201.

16. 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. с.166-175.

DSSP for ternary virtual machine

Burtsev A.A., Burtsev M.A.

Abstract. Ternary virtual machine (TVM) is a simulator of ternary computer, which architecture has two stacks (data stack and control stack) and machine commands for structured programming like “Setun-70”. TVM has been constructed as computer program in the computer laboratory leaded by Brusentsov N.P. (at the Moscow State University). Two variants of Dialogue System of Structured Programming (DSSP) for TVM are considered. DSSP-TVM let it possible to create DSSP-program for TVM by means of cross-compiler. DSSP/TVM is interpreter, which can run on TVM as its resident software. Programming language DSSP-T used in both systems is described as ternary version of language of the DSSP.

Опубликовано: Труды НИИСИ РАН, т.2, №1. М.: Изд-во НИИСИ РАН, 2012. с.73-82.