Бурцев А. А., Рамиль Альварес Х.

Кросс-система разработки программ на языке ДССП для троичной виртуальной машины

Введение

В настоящее время, когда двоичные цифровые машины (ЦМ) приблизились к потолку своих потенциальных возможностей, назрела необходимость построения троичного процессора на современной элементной базе, а также разработки программного оснащения для него.

В НИЛ ЭВМ МГУ разработан (и реализован на языке Си) программный комплекс ТВМ (Троичная Виртуальная Машина), 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” [1].

В качестве первичного инструментального средства подготовки программ для троичной машины вместе с ТВМ был разработан и ассемблер для неё. Но уже при создании ТВМ предполагалось, что основным языком разработки программ для неё должен стать троичный вариант языка ДССП [2-7], названный впоследствии ДССП-Т.

Чтобы обеспечить разработку программ для ТВМ на языке ДССП-Т, был сконструирован (и реализован на языке Си) кросс-компилятор с этого языка на язык ассемблера ТВМ. Вместе с ним были разработаны ассемблерное ядро, ДССП-библиотека стандартных модулей и командный монитор (ДКМОН), позволяющий прогонять получаемые ДССП-программы в режиме диалога. Все эти программные средства вместе с ассемблером и имитатором ТВМ составляют единую кросс-систему разработки программ для троичной виртуальной машины, получившую название ДССП-ТВМ.

1. Основные принципы построения системы

При создании ДССП-ТВМ учитывались перечисленные ниже требования как своеобразные принципы построения системы.

1. Кросс-компилятор языка ДССП-Т требовалось реализовать на языке Си, чтобы обеспечить возможность его функционирования в любой современной операционной среде (Windows, Linux).

2. Кросс-компилятор должен формировать внутреннее представление обрабатываемой ДССП-программы на языке ассемблера, чтобы избежать его зависимости от машинной кодировки команд ТВМ.

3. ДССП относится к классу интерпретируемых систем. В них исходная программа сначала переводится во внутреннее представление,

которое затем используется внутренним интерпретатором, обеспечивающим непосредственное исполнение программы. При создании ДССП-ТВМ большинство функций внутреннего интерпретатора предполагалось возложить на ТВМ.

4. Для внутреннего представления ДССП-программ всегда использовался так называемый сшитый (threaded) код. В ДССП сшитый код формируется по принципу “1-1”, т.е. одно слово исходного текста переводится в одно слово кода. Требовалось соблюдать такой принцип и в ДССП-ТВМ при формировании (кросс-компилятором) внутреннего представления ДССП-программ.

5. ДССП для организации ветвлений и циклов предлагает специфические команды структурированного управления, для которых тела ветвлений и циклов должны оформляться в форме отдельных процедур. Исполнение большинства таких команд предполагалось обеспечить соответствующими командами ТВМ условного и многократного вызова процедуры, полностью исключая потребность в командах условного и безусловного переходов.

6. Кросс-компилятор должен допускать возможность формирования (на языке ассемблера ТВМ) не только программного кода ДССП-программы, но и соответствующего ей словаря, который мог бы использоваться далее при прогоне полученной ДССП-программы на ТВМ, а также при её отладке.

7. Создаваемая компилятором ассемблерная программа должна позволять подключать к ней другие модули на языке ассемблера ТВМ.

8. Допускалось использование стандартного препроцессора (языка Си) перед обработкой полученной кросс-компилятором программы ассемблером ТВМ.

2. Структура и основные компоненты системы

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

- кросс-компилятор или ДССП-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 некоторого стандартного ассортимента наиболее употребительных слов;
- препроцессор (**cpp.exe**) – стандартный препроцессор 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**).

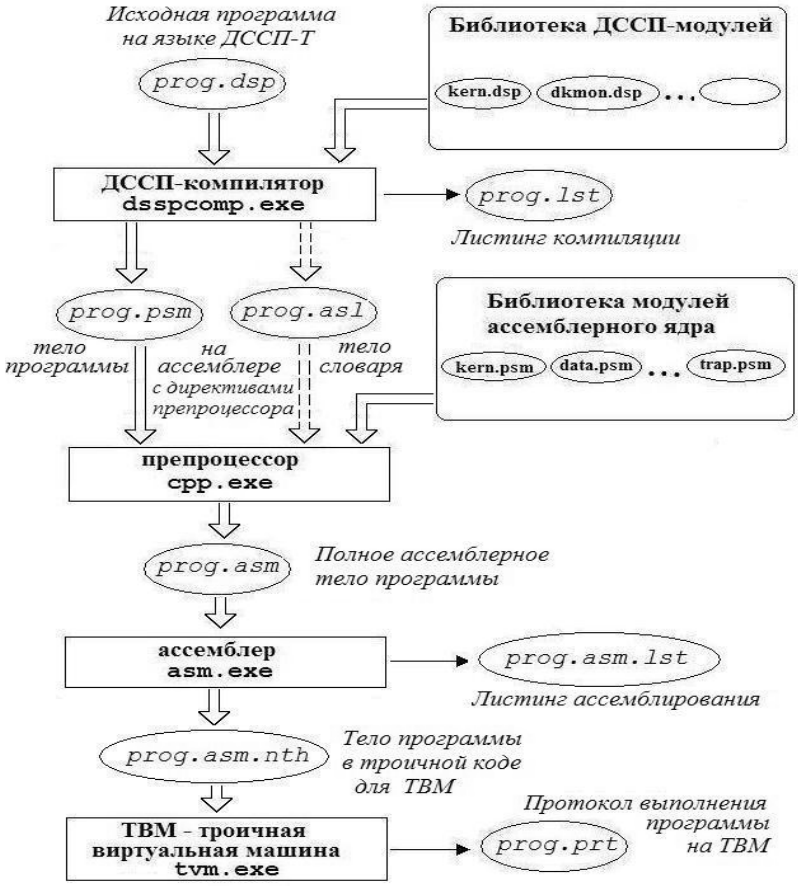

рис. 1. Схема прогона программы в ДССП-ТВМ.

Программа, составленная на языке ДССП-Т, (файл **prog.dsp**) проходит несколько стадий обработки в ДССП-ТВМ (см. рис. 1). Сначала она вместе с файлами ДССП-библиотеки обрабатывается ДССП-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 троич-

ном процессоре.

Наконец, программный 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.`

3. Особенности языка ДССП-Т

Язык ДССП-Т разрабатывался как троичный вариант языка ДССП, в котором было решено сохранить лучшие черты различных версий ДССП [2-7] для двоичных машин. Предполагалось также по возможности придать ему характерные свойства языков высокого уровня и отразить в нём привычные выразительные конструкции, присущие таким широко распространённым языкам программирования как Паскаль и Си. В качестве краткой характеристики языка ДССП-Т представим основные особенности, которые отличают его от языка ДССП, описанного в [4].

1. 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 пока без разделения на подсловари и секции, но в нём можно осуществлять очистку и обеспечена возможность помечать слова (**::**), которые всегда должны оставаться видимыми в словаре.

2. В языке ДССП-Т было решено допустить 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.

3. В ДССП-Т появилась необходимость помимо десятичных и 16-ных, задавать также троичные и 9-ные числа, характеризующие значения в так называемой троичной симметричной системе счисления. Было решено различать эти числа по специальным префиксам:

{ девятеричные числа: } #1234 #8765 **0t**qwer **0T**QWER

{ троичные числа: } **.**+0-. **.**+0- **0n**+0- **0N**+-0

{ 16-ные числа: } **$**12345678 **0x**90abcdef **0X**09ABCDEF

{ 10-ные числа: } 12345678 +90 -90

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

Для представления символьных констант используется общепринятая форма их записи в апострофах ‘X’, которая укоренилась во многих языках программирования (Си, Паскаль). Это позволило освободившийся знак # назначить в качестве префикса 9-ного числа.

4. 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 (трайт содержит 9 тритов). 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.

5. В словаре 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:} 17 / 3 {[6,-1] вместо [5,2]}
{сдвиг вправо= /3 :} 17 SHR { [6] вместо [8] }
{сдвиг влево = *3 :} 5 SHL { [15] вместо [10] }

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

6. 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 обработки очередного троичного слова памяти к значению адреса надо добавлять 3 (а не 4!).

7. При первоначальной реализации ДССП-Т было решено сократить до минимума набор префиксных операций над переменными (и массивами), ограничившись лишь операцией взятия значения (по имени), адреса (**‘**) и присвоения значения (**!**). Для объявления переменных и в качестве элементов массивов предусмотрены пока только два типа данных: **TRYTE** и **TWORD** для работы с трайтами и троичными словами.

8. В настоящее время язык ДССП-Т ориентирован на обработку его компилятором (а не интерпретатором, как это обычно осуществлялось для всех предыдущих версий ДССП). Поэтому в нём отсутствуют средства построения новых компилирующих слов, которые могли бы исполняться сразу же на стадии компоновки программы.

Кросс-компилятор пока обеспечивает действие лишь самого минимально необходимого набора компилирующих слов. Ассортимент таких слов языка ДССП-Т можно расширить только путём модификации непосредственно самого кросс-компилятора, в котором их действия реализуются соответствующими Си-функциями.

9. В языке ДССП-Т представлен весь ассортимент слов-команд условного вызова процедур, предусмотренных для организации ветвлений. Причём все команды одиночного (**IF- IF0 IF+**), двоичного (**BR- BR0 BR+**) и троичного (**BRS**) ветвления исполняются непосредственно как машинные команды самой ТВМ, и лишь команда многозначного ветвления (**BR** ... **ELSE**) реализуется соответствующей процедурой ассемблерного ядра.

Чтобы обеспечить возможности вычисления логических условий, эквивалентные средствам построения логических выражений языков высокого уровня, в ДССП-Т сохранены 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 (**CMP**) и оценки знака числа (**SGN**), предусматривающие троичный результат: меньше (-1), равно (0), больше (+1). Такие троичные результаты могут далее использоваться для вычисления сложного троичного условия путём применения потритных операций (**TMIN TMAX TADD TMUL NEG**).

10. 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 [2], 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 [5] появилась конструкция: **WHILE** условие **DO** тело, которая впоследствии (в ДССП/С [6,7]) 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 [3]. В ней появились команды (**DO RP**) для повторного вызова процедуры тела цикла заданное или бесконечное число раз. Были предусмотрены специальные команды для экстренного выхода из цикла по условию (**EX- EX0 EX+**) или безусловно (**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 описанных выше концепций: 1) **DW DO- LOOP** ; 2) **RP DO EX EX+ EX- EX0**. Причём команда **DW** исполняется как машинная команда ТВМ, а команды **RP** и **DO** реализованы на основе механизма ситуаций, чтобы обеспечить единый механизм установки и поиска ловушек в управляющем стеке.

11. В языке ДССП-Т предложен полный вариант механизма ситуаций (как в ДССП/С [6,7]). Объявлять ситуацию можно словами: **SITUATION** или **TRAP**. 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**). Кроме того, данный механизм дополнен особым типом ситуации, которая возбуждается для реализации экстренных выходов из циклов (**RP DO**) по командам (**EX EX+ EX0 EX-**).

Функционирование механизма ситуаций решено было пока обеспечить программным способом, не требуя от ТВМ никаких специальных команд для установки ловушек и их последующего поиска в управляющем стеке. Эти действия в настоящее время выполняются процедурами ассемблерного ядра.

12. 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, было решено обеспечить ей диалоговое общение через консоль терминала и операции доступа к файлам операционной среды ТВМ для последовательного чтения и записи.

13. 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**.

14. Обычно ДССП функционирует в режиме диалога, исполняя в качестве команд ту последовательность слов, которую пользователь задаёт в командной строке. В таком случае можно дать интерпретатору ДССП задание исполнить любое слово, приготовленное в словаре.

Но на языке ДССП-Т, вообще говоря, можно создавать любые программы, в том числе и такие, которые могут функционировать без командного интерпретатора ДССП. В таких программах необходимо указать, какое слово, определённое в программе, является главным, с которого и предстоит начать исполнение всей программы. В языке ДССП-Т предлагается задавать имя главного слова программы в первой строке исходной программы после служебного слова **PROGRAM**.

4. Характеристика ДССП-библиотеки

Возможности языка ДССП-Т не ограничиваются только словами базового словаря. Словарь компонуемой программы можно дополнить определениями слов, заготовленных в файлах библиотеки, загрузив их командой **LOAD**. Ассортимент этих слов постоянно пополняется. В настоящее время он включает операции:

- проверок над символами и строками (isBlank? strCmp);
- вывода на терминал строк и чисел (TOS TON TON9) ;
- ввода с терминала строк и чисел (TIS LTIS TIN) ;
- преобразования строки в число заданного формата (Str->Num9) ;
- преобразования числа заданного формата в строку (Num9->Str) ;
- печати содержимого арифметического и управляющего стеков и заданных ячеек памяти (. .? .. ..? ..CS .m .M ..m ...m ...M) ;
- а также определения некоторых общезначимых констант (‘LF’) и ряд других разнообразных часто используемых операций (E3DD >0?).

5. Специфика реализации ДССП-ТВМ

ДССП-ТВМ существенно отличается от всех предыдущих версий ДССП ещё и особенностями своей реализации.

Ранее ДССП всегда функционировала как резидентная система программирования, в которой входная программа (сразу вся или по частям) сначала переводилась во внутреннее представление, а исполнение такой закодированной ДССП-программы обеспечивалось специальным ПО, называемым внутренним интерпретатором. В ДССП-ТВМ роль внутреннего интерпретатора фактически возложена на саму ТВМ, а внутреннее представление всей программы подготавливается кросс-компилятором, т.е. отдельной компонентой системы, функционирующей не на самой ТВМ, а на другой машине.

Для внутреннего представления программ в ДССП всегда применялся так называемый сшитый (threaded) код. Хотя в разных версиях ДССП использовались та или иная разновидность сшитого кода (прямой [3], знаково-косвенный [6], двойной косвенности [2]), всех их объединяло одно характерное свойство: в сшитом коде в закодированном виде представлялись адреса процедур, которые интерпретатору предстояло вызывать при исполнении программы.

При разработке ДССП-ТВМ предстояло решить проблему, какую же разновидность сшитого кода применять для внутреннего представления, чтобы ТВМ могла напрямую (т.е. без специального интерпретатора) вызывать процедуры по встречающимся в этом коде адресам. Для решения этой проблемы было предложено применить в ТВМ такую кодировку команд, чтобы значение адреса воспринималась ею и как команда вызова процедуры с этого адреса. В результате для внутреннего представления программ в ДССП-ТВМ стал использоваться так называемый

процедурный сшитый код (STC [8]), который может непосредственно исполняться троичной машиной.

Кросс-компилятор формирует такой код на языке ассемблера ТВМ. В нём для каждого слова-примитива ДССП-ядра размещается одна машинная команда, которая реализует действие этого примитива. Для тех примитивов, действие которых невозможно исполнить одной машинной командой, заготовлены ассемблерные процедуры, их реализующие, а в сшитый код вставляется адрес (или команда вызова) соответствующей ей процедуры. Совокупность таких ассемблерных процедур образует ассемблерное ядро, которое добавляется к каждой откомпилированной ДССП-программе.

Аналогично формируются и тела новых слов, определяемых с помощью компилирующего слова **:** и представляющих собой процедуры в ДССП-программе. Каждой новой процедуре назначается адрес, начиная с которого располагается её тело и который вставляется в тела других процедур в том месте, где она из них вызывается.

Более сложное строение имеют тела слов, представляющих другие сущности ДССП-программы: переменные, массивы, исключительные ситуации, числовые и текстовые константы-строки. Каждому из этих слов тоже сопоставляется уникальный адрес его тела. Но по этому адресу предполагается не только вызывать процедуру, с которой начинается тело слова, но и получать доступ к тем или иным дополнительным характеристикам слова, сформированным в его теле (например, адресу переменной, границам массива и т.д.).

Заключение

В настоящее время ДССП-ТВМ функционирует в среде ОС Windows. Создаваемые на языке ДССП-Т программы можно прогонять на ТВМ как автономно, так и в режиме привычного для ДССП диалога, если скомпоновать и запускать их вместе со специально разработанным (на языке ДССП-Т) диалоговым командным монитором (ДКМОН).

В результате решена проблема целевой компиляции ДССП-программ для ТВМ и подготовлена инструментальная среда для построения версии ДССП, способной функционировать на самой ТВМ в качестве резидентной системы программирования.

Литература

1. 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.

2. 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.

3. Брусенцов Н.П., Захаров В.Б., Руднев И.А., Сидоров С.А. Диало-

говая система структурированного программирования ДССП-80. // Диалоговые микрокомпьютерные системы. М.: Изд-во МГУ, 1986, с.3-21.

4. 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.

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

6. 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.

7. 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.

8. Ritter T., Walker G. Varieties of threaded code for language implementation. // BYTE, 1980, v.5, No.9, p.206.

Опубликовано: Программные системы и инструменты. Тематический сборник № 12. М.: Изд-во факультета ВМиК МГУ, 2011. С. 183-193.