ДИАЛОГОВАЯ СИСТЕМА СТРУКТУРИРОВАННОГО ПРОГРАММИРОВАНИЯ

Юлия Сергеевна Владимирова
Факультет ВМК МГУ им. М. В. Ломоносова, Москва, Российская Федерация, vladimirova@cs.msu.ru

Аннотация **– Диалоговая система структурированного программирования ДССП была создана в начале 1980-х годов в МГУ под руководством главного конструктора троичных машин «Сетунь» и «Сетунь-70» Н.П. Брусенцова. ДССП разрабатывалась как средство программирования мини- и микрокомпьютеров, в основе были идеи, примененные в машинном языке «Сетуни 70». После переноса на персональные ЭВМ, она стала развиваемой системой программирования, обладающей рядом преимуществ, актуальных до настоящего времени. Существует несколько версий ДССП, в том числе поддерживающие троичную арифметику и логику.**

Ключевые слова **– программное обеспечение, структурированное программирование.**

1. Предыстория

В статьях [1-3] рассказывалось об уникальных троичных вычислительных машинах «Сетунь» и «Сетунь-70», оперировавших числами в троичной симметричной системе счисления. Диалоговая система структурированного программирования ДССП многие свои черты унаследовала от машинного языка «Сетуни-70». Это, в частности, двухстековая архитектура и синтаксис языка, основанный на бесскобочной польской инверсной записи ПОЛИЗ [4].

Первый вариант основанной на ПОЛИЗ системы автоматического кодирования был реализован раньше, в 1963-64 г. на «Сетуни» в качестве одной из интерпретирующих систем (ИП) [4]. В состав этой ИП входил входной язык СИМПОЛИЗ-64, внутренний троичный язык ИНПОЛИЗ и операционная система ПОЛИЗ, в которую в свою очередь входил интерпретатор и программные библиотеки.

Данная ИП предназначалась для тех машин «Сетунь», на которых предполагалось использовать транслятор с языка АЛГОЛ. Польская инверсная запись была применена в качестве его входного языка, так как это существенно упрощало алгоритмы трансляции. В то же время, интерпретация ПОЛИЗ осуществлялась на «Сетуни» весьма просто.

Неизбежное замедление выполнения программ в режиме полной интерпретации было небольшим – в среднем в 2,5-3 раза по сравнению с выполнением на ИП-3 [3] и компенсировалось тем, что программы на ПОЛИЗе были в 2-3 раза короче, чем на машинном языке. Наиболее существенным выигрышем от применения ПОЛИЗ было заметное упрощение создания и сопровождения программ [4].

Опыт использования ИП ПОЛИЗ был учтен при разработке в 1967 году троичной машины «Сетунь-70» [2]: польская инверсная запись была использована в качестве машинного языка также с целью повышения компактности программ и эффективности их трансляции. Такое решение потребовало ввести в архитектуру арифметический стек и использовать вместо стандартных машинных команд с кодами операций и адресами безадресных слогов, которые оперировали значениями в стеке или осуществляли пересылки значений в стек из памяти.

Единственным применением «Сетуни-70» была автоматизированная система обучения «Наставник» [5]. Ее разработка изначально была затруднена тем, что велась на машинном языке, так как для «Сетуни-70» не было создано базового программного обеспечения. Упростить разработку удалось благодаря внедрению идей предложенного Э. Дейкстрой структурированного программирования, целью которого было сокращение трудоёмкости создания программ, обеспечение легкости их понимания и модификации [6].

Принципы структурированного программирования оказалось достаточно несложно внедрить в стековую архитектуру: для этого потребовалось добавить системный стек и пополнить машинный язык командами ветвления, цикла и перехода к подпрограмме [2]. Соответствующая перестройка архитектуры «Сетуни-70» была осуществлена в 1975 г. После указанных изменений её машинный язык и архитектура послужили прототипом ДССП.

2. Структурированное программирование

Внедрение команд структурированного программирования на уровне машинного языка было необычным решением. Как правило, структурированное программирование понимается как технология составления и сопровождения больших программ на высокоуровневых языках программирования.

Суть ее состоит в последовательной декомпозиции программ: выполняемое программой действие разбивается на несколько обособленных поддействий, которые в свою очередь также разбиваются на составные части до тех пор, пока подпрограммы не окажутся состоящими из нескольких команд языка программирования. Этот способ программирования называется нисходящим («*top-down*»). Результатом оказывается представление программы в виде иерархической структуры программных блоков, каждый из которых выполняет чётко определённую функцию, чем и обеспечивается понятность, простота отладки и модификации программ [7].

Возможен и противоположный, восходящий способ («*bottom-up*»), который используется, в частности, для развития программных систем, а также при отладке программ.

Применение технологии структурированного программирования, как правило, приводит к некоторому снижению эффективности программ: структурированная программа длиннее обычной и выполняется дольше. В большинстве случаев это с запасом компенсируется повышением качества программы в течение всего её жизненного цикла.

ДССП. Первый вариант ДССП был разработан в 1980-82 гг. как система программирования мини- и микрокомпьютеров [8]. Эти цифровые машины были выполнены на интегральных схемах, благодаря чему производились большими тиражами и использовались для решения широкого круга прикладных задач. Их дешевизна достигалась не только высокоавтоматизированной технологией, но и существенным ограничением ресурсов и аппаратных функций. Универсальность обеспечивалась наличием в них набора лишь самых примитивных операций и типов данных, поэтому при разработке прикладных программ было необходимо основательное понимание устройства их архитектуры и машинного языка. С другой стороны, многообразие применений предполагало не только наличие большого количество программного обеспечения, но и возможность их программирования специалистами в той или иной прикладной области, не являющихся профессиональными программистами. Кроме того, были распространены микрокомпьютеры различных архитектур, поэтому актуальной была также и проблема переносимости программного обеспечения.

В связи с этим к программному обеспечению предъявлялись достаточно высокие требования: оно должно было предоставлять средства решения прикладных задач при минимальных аппаратных функциях, обладать высокой эффективностью, доступностью для широкого круга пользователей и переносимостью. Необходимо было обеспечить снижение трудоемкости разработки программ и их хорошую модифицируемость по сравнению с программированием на языке ассемблера.

Существовавшее для микрокомпьютеров программное обеспечение полностью указанные проблемы не решало [9]. Для программирования использовались достаточно сложный в освоении ассемблер и языки высокого уровня, такие как паскаль, фортран или бейсик. Последний специально разрабатывался для использования непрофессиональными программистами, но его удобство было получено ценой существенного синтаксического обеднения.

Следует также отметить, что не все существовавшие на тот момент языки обеспечивали хороший стиль программирования. Часто плохо решалась и задача переносимости. Для запуска программ, предназначенных для одной архитектуры, на другой, использовались кросс-системы, в состав которых входил эмулятор той системы команд, для которой была предназначена система изначально.

В [9] отмечается, что один из удачных способов решения этой проблемы был применен в языке FORTH [10], являющемся, по сути, программным эмулятором двухстековой машины, для которой реализованы интерпретатор и компилятор. Простота реализации эмулятора обеспечивает хорошую его переносимость. Простота синтаксиса языка, похожего на ПОЛИЗ, в сочетании с мощными средствами программирования являются неоспоримыми достоинствами этого языка.

FORTH – чрезвычайно гибкий язык программирования. Например, наличие встроенного ассемблера позволяет повышать компактность и быстродействие программ на FORTH путем вставки в них фрагментов на ассемблере. Другой пример – программная доступность стека возвратов.

Программирование на FORTH предполагает наличие этапа проектирования, после которого продуманная заранее программа конструируется восходящим способом, начиная с самых мелких ее составляющих. Эти принципы и средства разработки позволяют получать чрезвычайно эффективные программы, но они предназначены для использования профессиональными программистами и не были доступны для неспециалистов.

Все перечисленные выше требования оказались достижимыми благодаря применению приёмов, выработанных в ходе разработки архитектуры ЭВМ «Сетунь-70», в частности, средств поддержки структурированного программирования, получивших воплощение в ДССП.

Структура ДССП. Ядром ДССП является стековый ДССП-процессор. На его вход подаются либо команды непосредственно с терминала, либо последовательность команд в виде скомпилированной программы. В ДССП-процессоре имеются два стека: стек операндов (аналог арифметического стека ЭВМ «Сетунь-70»), предназначенный для обмена значениями между командами и стек возвратов (аналог системного стека), обеспечивающий вызов подпрограмм, ветвления и циклы. ДССП-процессор последовательно выполняет подаваемые на его вход команды.

В состав ДССП обычно включаются также компилятор, словарь базисных процедур, редактор и отладчик. По сути ДССП – больше, чем язык программирования, это виртуальный процессор с собственной архитектурой, адресацией памяти, системой команд и набором средств составления и отладки программ. При этом ядро ДССП – это небольшой компактный набор программ, которые реализуются достаточно просто. Остальные составляющие, такие, как редактор, являются надстройкой над ядром, что обеспечивает простоту переносимости системы. Кроме того, компактность ядра делает возможной его реализацию на оборудовании с очень небольшими ресурсами.

Еще одной важной особенностью ДССП является простота синтаксиса языка, благодаря которой система оказалась не только лёгкой в освоении пользователями, но и обеспечивала следование ими структурированной технологии программирования. Это было достигнуто применением так называемого процедурного программирования.

3. Процедурное программирование

Синтаксис языка ДССП предельно прост: единственным конструктивом в нем является процедура, т.е. поименованное действие. Существует несколько типов процедур. Обычная процедура выполняет какие-либо действия над стеком операндов. Например, процедура + складывает два числа в вершине и подвершине стека:

[a, b] + [a+b]

В квадратных скобках в качестве комментария перечисляются значения в стеке операндов. Значения в стеке принято перечислять через запятую слева направо, начиная с элемента, находящегося на наибольшей глубине, так что вершина стека оказывается самой правой в списке.

Процедуры с собственной памятью обеспечивают работу с хранимыми в памяти значениями. Например, константа 5 является в ДССП процедурой, единственное действие которой – это засылка значения 5 в вершину стека операндов. Процедуры, обладающие памятью, играют роль переменных различного формата. Например, переменная X, объявленная как целое число, засылает свое значение в вершину стека:

[ ] X [значение X]

Процедуры-префиксы влияют определённым образом на выполнение одной или нескольких процедур, непосредственно следующих за ними во входном потоке. С их помощью реализуются, например, процедуры создания новых процедур и доступа к памяти. Команда

VARX

объявляет процедуру с памятью формата «4-байтное число».

Другой пример: команда

[5] !X

осуществляет запись в память переменной X значения 5.

Префиксными процедурами реализуются также ветвления и циклы. Например, команда

[x] BRSP1 P2 P3

осуществляет ветвление по знаку числа в вершине стека. Если x<0, то будет выполнена процедура P1, и произойдет возврат в точку после процедуры P3. Если x=0, будет выполнена P2, если x>0, то выполнится P3.

Ядро ДССП содержит небольшой набор процедур, называемых примитивами, и реализованных непосредственно на машинном языке. Все остальные процедуры определяются как последовательность других процедур. Каждая вложенная процедура является либо примитивом, либо, в свою очередь, определяется как последовательность вложенных процедур. Таким образом, программа на ДССП выстраивается в иерархическую структуру, на верхнем уровне которой головная процедура, а на нижнем – примитивы.

Для внутреннего представления программ в ДССП используется сшитый код (threaded code), для которого характерно представление тел процедур в виде последовательности вызовов вложенных процедур. Такой способ обладает рядом преимуществ, в частности, допускает декомпиляцию процедур, а также позволяет компилировать и выполнять процедуру, даже в случае, когда не все вложенные в неё процедуры определены.

Стек *операндов и комментарии*. В ДССП стек операндов предназначен для обмена значениями между процедурами. Все элементы стека операндов имеют одинаковый размер, зависящий от версии ДССП. Контроль со стороны системы за типами операндов отсутствует – интерпретация значений в стеке полностью возлагается на обрабатывающие их процедуры. Это придает языку дополнительную гибкость, но уменьшает наглядность программ. Для описания операндов процедур и промежуточных состояний стека в ДССП используются комментарии, наличие которых считается обязательным.

4. Словарь ДССП

Программа на ДССП представляет собой последовательность слов, понимаемых как имена процедур. Совокупность всех слов, известных процессору, составляет словарь ДССП. Словарь включает небольшой набор примитивов, наборы, реализующие компоненты ДССП, например, отладчик. Каждый такой набор называется подсловарём. В ДССП имеются команды управления подсловарями, в частности, команда создания нового подсловаря, в который будут помещаться все компилируемые процедуры. Существуют команды открытия подсловаря для использования и для пополнения новыми словами, команда закрытия и удаления подсловаря.

В ДССП процедуры, определенные посредством других процедур, по своему действию неотличимы от примитивов. Эта регулярность языка обеспечивает высокую степень развиваемости и мобильности ДССП. Пакет процедур, решающих определённую задачу, может дозагружаться к базовой ДССП и использоваться наряду с процедурами базового набора без каких-либо ограничений.

Так как основой ДССП является бесскобочная запись арифметических выражений, в ней отсутствуют и операторные скобки, позволяющие сгруппировать несколько слов в блок. Любой такой блок должен определяться как процедура.

В этом есть некоторые неудобства – в программе появляется большое количество коротких процедур. Например, при необходимости присвоить переменной X значения из вершины стека, если это значение положительно, оператор присваивания, реализуемый префиксной процедурой !X, требуется вынести в отдельную процедуру:

```
…[x] CIF+ P …
: P [x] !X[ ];
```

И таких процедур может возникнуть достаточно много. С другой стороны, при отсутствии операторных скобок становятся ненужными иерархии процедур и присущие высокоуровневым языкам правила видимости. Все процедуры открытого подсловаря оказываются доступными для использования.

Как правило, подсловарь представляет собой программную библиотеку, решающую некоторую задачу. В нем имеется небольшой набор головных процедур, и вложенные в них процедуры, использование которых дальше не предполагается. Существуют средства чистки подсловаря – удаления из доступа имен вложенных процедур.

Отладка. Поддерживаемое ДССП структурированное программирование и иерархическая структура ДССП-программ предполагает их отладку последовательной восходящей проверкой. Сначала проверяются процедуры самого нижнего уровня, затем те, компонентами которых они являются, и так далее, пока не будут отлажены процедуры самого высокого уровня. Такой способ отладки оказывается более эффективным, чем традиционное тестирование. Кроме того, структура программы в виде иерархии вложенных подпрограмм способствует исчерпывающей проверке всех ветвей программы.

Использование для внутреннего представления сшитого кода сделало возможной отладку недоопределенных процедур. Процедура, в теле которой есть вызов еще не определенной процедуры, может быть скомпилирована и выполнена. Тело неопределенной процедуры заменяется компилятором процедурой-заглушкой, вызывающей останов работы программы. Если неопределенная процедура вызывается в ветке, переход к которой не предполагается, остальная часть программы может отлаживаться и использоваться. Исчерпывающее описание ДССП приведено в [11].

5. Развитие ДССП

Первая версия была разработана в 1980-82 гг. для миникомпьютера «Электроника НЦ-03Д», последующая – для унифицированной архитектуры PDP-11. До середины 1990-х ДССП развивалась достаточно интенсивно, в частности, благодаря связанным с ней научным исследованиям. К середине 1990-х существовали версии ДССП для большинства распространенных на тот момент платформ и операционных систем. Подробно различные версии ДССП, а также ее применения перечислены в [12]. Несмотря на все достоинства ДССП, после 1990-х годов ее развитие и использование практически прекратилось.

Следует отметить две версии, поддерживающие троичные вычисления, которые связаны с продолжением исследований троичных вычислений и троичной элементной базы, осуществлявшейся под руководством Н.П. Брусенцова до начала 2010-х гг. Широко известно, что современные технологии создания вычислительной техники подошли к пределу своих возможностей, и троичная техника рассматривается как одна из вариантов дальнейшего развития. В связи с этим для осуществления исследований троичных алгоритмов были созданы две версии ДССП, поддерживающие вычисления в троичной симметричной системе счисления.

Первая из упомянутых версия ДССП содержит набор операций для работы с троичными числами, отдельный троичный стек и возможность пересылки значений из стека обычного операндов в троичный и обратно.

Вторая версия, получившая название ДССП-Т [13], была разработана в МГУ в 2010 г. в рамках проекта разработки имитационной модели троичного процессора ТВМ [14]. ДССП-Т была принята в качестве основного для языка программирования для ТВМ.

Автор выражает искреннюю благодарность участнику разработки программного оснащения вычислительных машин «Сетунь» и «Сетунь-70», руководителю разработки программного оснащения системы обучения «Наставник», автору версии ДССП, оперирующей троичными числами, Хосе Рамилю Альваресу за неоценимую помощь в написании статьи.

Список литературы

1. Брусенцов Н.П., Рамиль Альварес Х. Троичные ЭВМ «Сетунь» и «Сетунь-70» // SORUCOM-2006: Развитие вычислительной техники в России и странах бывшего СССР: история и перспективы. В 2 ч. Ч. 1. Петрозаводск, 2006. С. 45-51.
2.Брусенцов Н.П., Рамиль Альварес Х. Троичная ЭВМ «Сетунь-70» // Труды SORUCOM-2011. Вторая Международная конференция «Развитие вычислительной техники и её программного обеспечения в России и странах бывшего СССР». Великий Новгород, 2011. С. 71-75.
3. Рамиль Альварес Х., Владимирова Ю.С. Программное обеспечение малой ЭВМ «Сетунь» // Труды SORUCOM-2014. Третья Международная конференция «Развитие вычислительной техники и ее программного обеспечения в России и странах бывшего СССР: история и перспективы», 13-17 октября, Казань, Россия. Казань, 2014. С. 315-318.
4. Жоголев Е.А., Лебедева Н.Б. СИМПОЛИЗ 64 – язык для программирования в символических обозначениях. Серия: Математическое обслуживание машины «Сетунь». Под общей редакцией Е.А. Жоголева. Вып. 10. М: Изд-во Моск. ун-та, 1965.
5. Брусенцов Н.П., Маслов С.П., Рамиль Альварес Х. Микрокомпьютерная система обучения «Наставник». М.: Наука, 1990.
6. Dijkstra E.W. Notes on structured programming. EWD 249 – Technical University, Eindhowen, Netherland, 1969.
7. Брусенцов Н.П., Рамиль Альварес Х. Структурированное программирование на малой цифровой машине // Вычислительная техника и вопросы кибернетики. Вып. 15. М.: Изд-во Моск. ун-та, 1978. С. 3-8.
8. Брусенцов Н.П. Микрокомпьютеры Учеб. пособие для студ. ВУЗов. М.: Наука, 1985. 206 с.
9. Брусенцов Н.П. О программном оснащении микрокомпьютеров. // Программное оснащение микрокомпьютеров. Под ред. Брусенцова Н.П., Маслова С.П. М.: Изд-во Моск. ун-та, 1982. С. 3-10.
10. Келли М., Спайс Н. Язык программирования Форт. М: Радио и связь, 1993. 320 с.
11. Брусенцов Н.П., Захаров В.Б., Руднев И.А., Сидоров С.А., Чанышев Н.А. Развиваемый адаптивный язык РАЯ диалоговой системы программирования ДССП. Учебное пособие. М.: Изд-во Моск. ун-та, 1987.
12. Бурцев А.А., Сидоров С.А. История создания и развития ДССП: от «Сетуни-70» до троичной виртуальной машины // Труды SORUCOM-2011. Вторая Международная конференция «Развитие вычислительной техники и её программного обеспечения в России и странах бывшего СССР» (12-16 сентября 2011 г., г. Великий Новгород, Россия). С. 83-88.
13. Бурцев А.А., Рамиль Альварес Х., Кросс-система разработки программ на языкеДССП для троичной виртуальной машины // Программные системы и инструменты. Тематический сборник № 12. М.: Изд-во факультета ВМиК МГУ, 2011. С. 183-193.
14. 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.