Брусенцов Н.П.

Вычислительная техника и
вопросы кибернетики. Вып. 15.
М.: Изд-во Моск. ун-та, 1978.

Н. П. БРУСЕНЦОВ, Х. РАМИЛЬ АЛЬВАРЕС

СТРУКТУРИРОВАННОЕ ПРОГРАММИРОВАНИЕ НА МАЛОЙ ЦИФРОВОЙ МАШИНЕ

Структурированное программирование [1] расценивается обычно как методика разработки больших, сложных программ, реализуемая на базе языка высокого уровня, например Фортрана или Кобола. При этом преимущества структурированных программ (регулярность логической структуры, легкость понимания, отладки, обслуживания и внесения изменений) достигаются ценой некоторого снижения их эффективности: структурированная программа, как правило, длиннее обычной и выполняется дольше. Не видно, однако, никаких доводов против реализации структурированного программирования на уровне машинного языка, чтобы использовать его на всех последующих уровнях. Есть основания надеяться, что эффективность структурированных программ при этом повысится.

В 1975 г. авторы произвели в указанном направлении успешный эксперимент, введя в набор команд машины «Сетунь-70» [2] новые команды, функционально равноценные операторам структурированного программирования. Настоящая статья посвящена изложению предпосылок и результатов этой работы.

Разработка машины «Сетунь-70» основывалась на опыте освоения и использования в вузах и в ряде промышленных организаций малой цифровой машины «Сетунь», выпускавшейся в 1962—1965 гг. Этот опыт убедительно подтвердил целесообразность тех принципов построения малой цифровой машины, которые определили архитектуру «Сетуни» [3]:

короткое машинное слово (9 тритов, что эквивалентно 14,3 битам),

компактный набор простых команд (24 команды),

страничная (зонная) структура памяти, выполнение памяти в виде двух ступеней (основная — на магнитном барабане, оперативная — в виде быстродействующего устройства небольшой емкости), обмен между которыми производится страницами,

наличие резерва скорости выполнения операций, позволяющего эффективно использовать интерпретирующие программы

для продолжения и развития в нужном направлении и до желаемого уровня вычислительных и управляющих функций, которые аппаратурой осуществляются в весьма элементарном виде.

Реализация данных принципов, составивших впоследствии основу архитектуры миникомпьютеров, не только позволила значительно удешевить машину и резко повысить величину отношения производительность/цена, но также обеспечила появление у нее таких важных качеств, как программная модифицируемость и приспосабливаемость к применениям (настраиваемость на применение). Не предназначавшаяся для системных применений «Сетунь» была успешно использована в ряде весьма разнохарактерных автоматизированных систем: статистической обработки данных, представленных графиками, структурного анализа кристаллов, прогноза погоды, проектирования строительных конструкций, программированного обучения и др. Как вычислительная машина, она в зависимости от используемой интерпретирующей программы могла работать с фиксированной и с плавающей запятой, с 6, 8 и 12 десятичными знаками, а также с комплексными числами. Существенно, что интерпретирующие программы не были «зашиты» в постоянном запоминающем устройстве (ПЗУ), а вводились по выбору с перфоленты на магнитный барабан. Благодаря этому, с одной стороны, машина была более простой и дешевой, а с другой стороны, легко настраивалась на желаемый вычислительный режим и для использования ее в той или иной системе не требовалось «перешивать» ПЗУ.

Принципы, определившие направление и характер разработки «Сетуни-70» (1966—1968 гг.), в общем были теми же, на которых базировалась разработка «Сетуни». Вместе с тем архитектура машины претерпела значительные изменения. Стремление повысить компактность представления программ и создать благоприятные условия для оптимальной компиляции привело к использованию в качестве машинного языка польской инверсной записи (ПОЛИЗ). Кроме того, были произведены многие другие усовершенствования, направленные на более эффективное использование оборудования, обеспечение возможности наращивать память и периферию, усиление приспосабливаемости к применениям и т. п. В частности, размер прямо адресуемой единицы памяти был уменьшен с 9 до 6 тритов, минимальная емкость главной памяти увеличена с 3 до 27 страниц (в том числе 18 страниц ПЗУ для системных программ) с возможностью дальнейшего расширения при неизменных форматах команд и незначительных изменениях аппаратуры; обеспечена возможность обрабатывать совместно данные длиной 6, 12 и 18 тритов и длиной результата до 36 тритов; введена приоритетная система прерыва-

ний, привилегированные режимы и аппарат программируемых макроопераций, призванный повысить эффективность использования интерпретирующих программ; реорганизована и значительно расширена периферийная часть машины, в состав которой включены три канала ввода-вывода, способные обслужить до 24 устройств, и три канала страничного обмена для устройств периферийной памяти.

Несмотря на то что та программная надстройка, на которую рассчитывались все эти средства, не была осуществлена, машину удалось эффективно использовать, причем в деле, специфика которого в процессе разработки учтена не была, а именно в автоматизированной системе обучения [4]. Правда, при этом почти не используются вычислительные ресурсы машины и свойственная ей простота программирования арифметических выражений в ПОЛИЗ, но в остальном архитектура оказалась хорошо отвечающей требованиям программированного управления учебой и построенная на основе ее обучающая система может быть охарактеризована как достаточно гибкая и экономичная. Опыт создания этой системы дает право утверждать, что машина окажется весьма подходящей и для многих других автоматизированных систем управления.

Тем не менее разработка программного оборудования обучающей системы в условиях, когда машина, рассчитанная на программную реализацию многих выполняемых обычно аппаратурой функций, вообще не оснащена программами, естественно была трудоемкой. Известные трудности составления и отладки логически сложных управляющих программ значительно увеличивались вследствие программирования на языке машинных команд и из-за страничной структуры памяти.

Стремясь уменьшить трудоемкость и повысить эффективность программистского труда, мы стали внедрять в практику нашей работы структурированное программирование. Однако управляющие команды машины плохо соответствовали операторам управления, используемым в структурированных программах, и поэтому машинная эффективность этих программ получалась низкой, что на малой машине с ее скудными ресурсами совершенно неприемлемо. Решили перестроить архитектуру машины применительно к требованиям структурированного программирования, чтобы структурированные программы могли стать машинно-эффективными.

Архитектура «Сетуни-70» оказалась исключительно благоприятной для указанной перестройки. То, что команды этой машины представляются сочетаниями адресных и операционных слогов и имеют варьируемую длину, позволило легко построить в машинном языке конструкции, прямо соответствующие основным операторам структурированного программирования (ветвление, повторение, выполнение подпрограммы).

Наличие же в машине стекового механизма для автоматического упрятывания и восстановления (операция «возврат») значений программного счетчика обеспечило возможность простой и эффективной реализации новых команд.

Перестройка значительно осложнялась тем, что ее надо было осуществить, не нарушив нормальной эксплуатации машины и избежав сколько-нибудь существенных переделок в имеющихся программах. Последнее условие казалось особенно трудным, потому что свободных кодов операций не было и новые операции можно было ввести, лишь исключив некоторые из имеющихся. К счастью, оказалось, что имеются операции либо вообще не использованные в программах, либо использованные исключительно редко. Анализ показал, что среди этих операций имеют место действительно бесполезные, утратившие значение, например, в результате того, что приемы программирования, для реализации которых они были предусмотрены, на практике оказались ненужными вследствие изобретения других, более удачных приемов. Так, практически не получили применения операции прибавления единицы к содержимому регистра E и вычитания единицы из этого регистра ($E+1$ и $E-1$). Незначительным оказалось применение операции, устанавливающей в E нуль ($E=0$). Не нашла употребления операция засылки в арифметический стек части текущего значения программного счетчика, определяющей адрес выполняемого слога программы в пределах страницы ($T=C$), предназначавшаяся для запоминания адреса возврата при обращении к подпрограмме. Именно эти четыре операции были изъяты из числа выполняемых машиной и заменены новыми операциями.

В число новых операций вошли операции ветвления, повторения (цикла), выполнения подпрограммы и обобщенного перехода. Последняя к структурированному программированию отношения не имеет и в дальнейшем не рассматривается.

Операция *ветвления* является троичным аналогом оператора *IF THEN ELSE*. Ветвление производится в зависимости от знака содержимого вершины T арифметического стека выбором одной из трех альтернатив:

if $T<0$ **then** $P1$ **else if** $T=0$ **then** $P2$ **else** $P3$.

Здесь $P1$, $P2$, $P3$ — идентификаторы подпрограмм. Попарное отождествление $P1$, $P2$, $P3$ позволяет получить любое из двухальтернативных ветвлений. Например,

при $P3 \equiv P2$ имеем **if** $T<0$ **then** $P1$ **else** $P2$,
при $P2 \equiv P1$ имеем **if** $T \leqslant 0$ **then** $P1$ **else** $P3$

и т. д.

Команда *ветвления* представляет собой последовательность четырех слогов: $K0$ $K1$ $K2$ $K3$. Слог $K0$ — операцион-

ный, код операции ветвления. Остальные три слога — адресные, начальные адреса подпрограмм соответственно $P1$, $P2$, $P3$. Выполняя команду ветвления, машина запоминает в системном стеке адрес возврата, равный $PC+4$, где PC — текущее значение программного счетчика (адрес слога $K0$), и засылает в программный счетчик один из адресных слогов команды — $K1$, $K2$, $K3$ в зависимости от знака содержимого вершины T арифметического стека. Таким образом происходит переход на одну из подпрограмм $P1$, $P2$, $P3$. Каждая из этих подпрограмм оканчивается слогом «возврат», вызывающим обратную пересылку адреса возврата из системного стека в программный счетчик, т. е. переход на продолжение программы, на команду, непосредственно следующую за командой ветвления.

Операция *повторения* реализована в виде **while** $T \neq 0$ **do** P, где P — индентификатор подпрограммы.

Команда *повторения* состоит из двух слогов — KO KA, операционного и адресного, указывающего начальный адрес подпрограммы P.

Если условие $T \neq 0$ выполнено, то текущее значение PC, являющееся адресом первого слога команды повторения, отсылается в системный стек, а его место в программном счетчике занимает второй слог этой команды — KA, т. е. происходит переход на подпрограмму P. Подпрограмма оканчивается операционным слогом «возврат», который вызывает обратный переход на команду повторения, таким образом замыкая цикл.

Если же при очередном выполнении слога KO команды повторения условие $T \neq 0$ окажется невыполненным, то машина пропускает слог KA этой команды и переходит к выполнению следующего за ним слога программы.

Команда *выполнения подпрограммы* также имеет вид KO KA, но интерпретируется иначе. Выполняя ее, машина не проверяет условия и в качестве адреса возврата запоминает в системном стеке не PC, а $PC+2$. Таким образом, переход на подпрограмму производится безусловно, а возврат осуществляется на команду, непосредственно следующую за командой выполнения подпрограммы. Другими словами, команда выполнения подпрограммы в процессе выполнения просто замещается указываемой ею подпрограммой.

Длина машинного слога — 6 тритов — достаточна для обеспечения прямой адресации в пределах 9 страниц по 81 слогу, т. е. во всей оперативной (чтение и запись) памяти машины. Доступ к подпрограммам, расположенным в ПЗУ, обеспечивается аппаратом макроопераций.

Практика разработки и эксплуатации программ в условиях модернизированной таким образом системы команд полностью подтвердила отмечаемые обычно преимущества струк-

турированного программирования: программы на самом деле отличаются легко прослеживаемой логикой, их проще проверять, отлаживать, модифицировать. Существенно, что все эти преимущества получены без снижения машинной эффективности программ, сопутствующей обычно применению структурированного программирования на уровне машинно-независимых языков. Но, кроме того, специфическая форма машинной реализации операторов структурированного программирования на основе стекового механизма обращения к подпрограмме обусловила возможность дальнейшего улучшения структуры программы и явилась избавлением от неудобств, которые при обычном программировании имели место из-за страничной организации памяти машины.

Как видно из данного выше описания, команды ветвления и цикла отличаются от соответствующих операторов структурированного программирования, в частности, тем, что тела ветвей и цикла в них заменены ссылками на подпрограммы (именами или адресами начала подпрограмм). С этим отличием связаны по меньшей мере два следующих преимущества. Во-первых, вынесение фрагментов программы, составляющих тела ветвей и циклов, в подпрограммы, на которые ссылается главная программа, значительно улучшает обозримость и понятность последней. Во-вторых, программа, представлявшая собой единую последовательность команд, отладка и выполнение которой с учетом имеющихся в ней ветвлений и циклов в условиях страничной памяти сопряжены с известными трудностями, теперь распадается на сравнительно небольшие части, которые можно автономно отлаживать и произвольно размещать в памяти. Ведущая часть (главная программа) состоит в основном из команд ветвления, цикла и безусловного выполнения подпрограммы, содержащих ссылки на другие части (подпрограммы первого уровня), которые в свою очередь могут содержать команды, ссылающиеся на подпрограммы второго уровня и т. д. Не исключены также ссылки в подпрограммах более высоких уровней на подпрограммы низших уровней.

ЛИТЕРАТУРА

1. Дал У., Дейкстра Э., Хоор К. Структур(ирован)ное программирование. М., «Мир», 1975.

2. Брусенцов Н. П., Жоголев Е. А. Структура и алгоритм функционирования малой вычислительной машины. — В кн.: Вычислительная техника и вопросы кибернетики, вып. 8. Изд-во ЛГУ, 1971, с. 34—51.

3. Брусенцов Н. П., Жоголев Е. А. и др. Малая автоматическая цифровая машина «Сетунь». — «Вестн. Моск. ун-та, сер. матем.» 1962, № 4; 3—12.

4. Брусенцов Н. П., Маслов С. П., Рамиль Альварес Х. Автоматизированная система обучения «Наставник». — В кн.: Вычислительная техника и вопросы кибернетики, вып. 13. Изд-во МГУ, 1976, с. 3—17.