Брусенцов Н.П

Вычислительная техника и
вопросы кибернетики. Вып. 13.
М.: Изд-во Моск. ун-та, 1977.

Н. П. БРУСЕНЦОВ

СТЕКОВЫЕ МАШИНЫ С ИЗМЕНЯЕМОЙ АДРЕСНОСТЬЮ КОМАНД

Адресность — это число адресов, содержащихся в команде цифровой машины.

Прежде команды цифровой машины были, как правило, одной и той же адресности, и поэтому говорили об адресности машины. Известны одно-, двух-, трех- и четырехадресные, а с появлением стековой архитектуры также нуль-адресные машины.

В литературе можно встретить рассуждения о том, какая адресность выгоднее, и доказательства исключительной целесообразности той или иной из них, иногда вполне убедительные. Например, в конце 50-х годов, когда уже стала ясной необходимость значительного увеличения емкости запоминающих устройств, но еще не были осознаны возможности непрямой адресации, вывод о том, что машина должна быть одноадресной [1] был, безусловно, правильным. Однако развитие в последующие годы относительной и косвенной адресации и применение собственной памяти процессора в виде общецелевых регистров с короткими адресами сделали этот вывод недействительным.

Попытки ответить на вопрос: «Какая адресность лучше?» — продолжаются [2], хотя в общем ясно, что каждая адресность хороша по-своему и выгодна в соответствующих условиях. Практика создания цифровых машин привела к компромиссному решению — в системе команд машины предусматривают команды разной адресности [3].

Наш ответ: адресность команды должна быть переменной, автоматически изменяющейся применительно к конкретным условиям использования команды. Настоящая статья посвящена обоснованию этого утверждения и изложению предложений по практическому осуществлению его.

1. Умолчание в машинных командах

Команда — это предписание, однозначно определяющее действия машины на протяжении некоторого этапа ее работы, называемого обычно циклом, или шагом. Машина производит обработку данных, выполняя последовательность шагов, заданных последовательностью команд (программой). За один шаг, как правило, осуществляется одна операция над одним или несколькими кусками данных. Команда однозначно задает как операцию, так и ее объекты — операнды. Операцию указывает код операции, представляющий на машинном языке имя операции. Операнды могут быть заданы адресами, т. е. машинными именами ячеек памяти, содержащих значения операндов, или непосредственно значениями, содержащимися в самой команде.

Системы адресации, т. е. определения ячеек памяти по указанному в команде адресу (или элементам адреса) операнда, в современных машинах сложны и разнообразны, однако заданная в команде адресная информация определяет соответствующую операнду ячейку при данном состоянии машины однозначно. Далее мы называем адресом или именем операнда именно ту адресную информацию, которой операнд представлен в команде.

Операндами мы будем называть не только входные объекты операций (аргументы), но и выходные (результаты).

При таком понимании данных терминов машинная команда может рассматриваться как содержащая операцию, представленную ее именем (кодом), и операнды, каждый из которых задан или его именем (адресом) или непосредственно значением. Выходные операнды могут быть заданы только именами.

Адресность команды, естественно, зависит от арности (числа аргументов) задаваемой командой операции. Но эта зависимость выражается, пожалуй, только в том, что для операции с большей арностью при равных прочих условиях может потребоваться команда с большей адресностью. Практически адресность определяется главным образом степенью использования неявного задания или умолчания операндов.

Примером команд с явным заданием всех операндов являются команды трехадресных и четырехадресных машин. Типичная трехадресная команда задает бинарную (двухместную) операцию, указывая явно оба ее аргумента и адрес результата.

Четырехадресная (три-плюс-один-адресная) команда использует возможность умолчания даже в меньшей степени, чем трехадресная — она в дополнение к содержимому последней включает адрес, указывающий следующую команду программы. Обеспечиваемая таким образом возможность произволь-

ного расположения команд программы в памяти машины существенна в отдельных специальных случаях, но, как правило, она не существенна. Отказавшись от этой возможности и предусмотрев в машине автомат, формирующий адрес каждой следующей команды, можно уменьшить длину команд изъятием из них соответствующего операнда.

Как известно, в современных цифровых машинах адрес следующей команды указывается только в командах перехода, а относительно других команд подразумевается, что этот адрес формируется программным счетчиком путем фиксированного приращения адреса выполняемой команды. По сравнению с явным заданием адресов команд, используемым четырехадресной системой, такой способ указания последовательности команд можно характеризовать как связанный с умолчанием адреса следующей команды. Отсутствие этого адреса в команде означает, что он должен быть получен от программного счетчика, а в случае перехода, т. е. когда адрес следующей команды необходимо указать явно, употребляется специальная команда, заведомо содержащая этот адрес.

В машине с переменной длиной команд можно не иметь специальной команды перехода, условившись, что наличие в выполняемой команде «лишнего» (нормально отсутствующего) адреса символизирует переход по данному адресу. При этом команда будет помимо основной ее операции задавать операцию перехода подобно тому, как принято в четырехадресной машине, но обеспечивая экономию адреса в случае, когда переход не производится.

В значительно большей степени умолчание используется в одноадресной машине. Задание двухместной операции в ней осуществляется командой с единственным операндом, который соответствует одному из аргументов. Другой аргумент подразумевается находящимся в специальном регистре — аккумуляторе, который является также регистром результата операции.

Можно сказать, что к аккумулятору относится все, о чем умалчивает команда, за исключением, конечно, умолчания адреса следующей команды, формирование которого осуществляет программный счетчик. Например, команда сложения, имеющая в одноадресном варианте вид: «сложить x», где x — имя-адрес одного из слагаемых, означает, что значение операнда x надо сложить с находящимся в аккумуляторе значением другого операнда, и результат поместить в аккумулятор. Другими словами надо прибавить значение операнда x к содержимому аккумулятора. Во всякой команде недосказанное означает аккумулятор: «послать x» значит послать значение x в аккумулятор, «послать в x» значит послать в ячейку x главной памяти содержимое аккумулятора и т. д.

В еще большей степени умолчание используется в так называемой нуль-адресной машине, которая обладает многоместным аккумулятором, организованным по принципу стека (магазина). Стек запоминает поступающие на его вход операнды, в частности, промежуточные результаты, и по требованию возвращает их в обратной последовательности

В нуль-адресной машине операция, как правило, вызывает значения своих аргументов из стека, а значение, получаемое в результате операции, отсылается в стек. При такой организации в команде умалчиваются все операнды, поэтому машина и названа нуль-адресной. Подразумевается, что одноместная операция всегда использует в качестве аргумента содержимое вершины стека и замещает его своим результатом. Двухместная операция использует в качестве аргументов содержимое вершины и подвершины и т. д.

Поскольку результат каждой операции отсылается в стек, то он автоматически становится аргументом одной из последующих операций. В одноадресной машине это возможно только в тех случаях, когда результат данной операции оказывается аргументом непосредственно следующей за ней операции, а в трехадресной машине все промежуточные результаты отсылаются в главную память и затем вызываются из нее, причем все эти пересылки задаются явно в командах.

Приняв для нуль-адресной машины, что аргументы операций вызываются из стека, а результаты отсылаются в стек, следует, очевидно, предусмотреть команды засылки данных из главной памяти в стек и обратно, из стека в главную память. Ясно, что эти команды должны содержать или, по меньшей мере, указывать адрес ячейки главной памяти, причем для засылки из главной памяти непременно должна быть команда с явным заданием адреса.

В духе максимума умолчания эта команда может содержать один только адрес [4], т. е. в ней умалчивается имя (код) операции. Адреса не встречаются в других командах, поэтому всякое появление адреса можно рассматривать как команду засылки в стек содержимого указываемой данным адресом ячейки главной памяти.

Другие команды, касающиеся главной памяти, при этом уже не могут содержать адреса, и должны быть именами операций, интерпретирующих один из предназначенных для них в стеке операндов как адрес главной памяти. Например, операцию засылки значения из стека в главную память можно определить как двухместную, которая, получив из стека два операнда, рассматривает первый как адрес ячейки и засылает в эту ячейку второй операнд. В другом варианте такая операция может засылать операнд, полученный из стека первым, по адресу, полученному в качестве второго операнда.

Аналогично определяются операции засылки в стек с выборкой из главной памяти по содержащемуся в стеке адресу и операции перехода с заданием адреса очередной команды через стек [4].

Как показывает изложенное, умолчание операндов и операций в командах сокращает объем команд, но при этом уменьшает их возможности. Например, трехадресная команда способна задать двухместную операцию над аргументами, находящимися в главной памяти, с отсылкой результата в главную память. Для задания эквивалентной процедуры в одноадресной машине необходимы три команды: засылка первого аргумента в аккумулятор, операция над содержимым аккумулятора и вторым аргументом, отсылка результата в главную память, а нуль-адресной машине требуется пять команд: засылка в стек первого аргумента, засылка в стек второго аргумента, операция над стеком, засылка в стек адреса результата, отсылка результата в главную память.

В свете этого примера может показаться, что умолчание нерационально, — сокращая объем команд, оно вызывает увеличение программы, реализующей данную процедуру, да еще требует наличия в машине специальных механизмов (стек, аккумулятор). Однако в действительности это не так. Дело в том, что используемая в примере процедура является наивыгоднейшей для трехадресной машины (точно соответствует ее команде) и одной из наименее выгодных для одноадресной и нуль-адресной машин. На других примерах можно с таким же успехом продемонстрировать преимущество одноадресной и нуль-адресных машин. Так, для задания арифметических действий в цикле процедуры вычисления значений полиномов по схеме Горнера на трехадресной и одноадресной машинах требуется по две команды, а на нуль-адресной — четыре команды. Поскольку трехадресная команда содержит одну операцию и три операнда, в то время как в одноадресной одна операция и один операнд, а в нуль-адресной — или одна операция, или один операнд, то оказывается что рассматриваемый фрагмент программы в трехадресном варианте будет содержать 8 элементов (2 операции и 6 операндов), а в одноадресном и нуль-адресном вариантах — только 4 элемента (2 операции и 2 операнда). Правда, работа, выполняемая трехадресной машиной, не вполне эквивалентна выполняемой одноадресной и нуль-адресной в том, что первая отсылает результат в главную память, а последним для этого требуется по окончании вычислений команда (нуль-адресной машине — две команды) отсылки результата в память. Но даже в случае полинома небольшой степени это практически не изменит указанного соотношения, да и не всегда нужно отсылать результат в главную память.

Ясно, что в зависимости от характера реализуемой про-

цедуры наиболее экономной по объему программы может быть то одна, то другая машина. Но программа для трехадресной машины может быть более компактной только в тех случаях, когда отсылка результатов в главную память производится весьма часто, т. е. когда результат операции, как правило, не используется в качестве аргумента следующей операцией. При этом компактность достигается за счет того, что операция в трехадресной команде означает кроме соответствующего преобразования аргументов в результат также выборку из главной памяти обоих аргументов и отсылку в эту память результата. В одноадресной машине требуется задать в виде отдельных операций выборку одного из аргументов и отсылку результата, в нуль-адресной машине — отсылку результата.

В случае одноадресной и нуль-адресной машин программа оказывается более компактной в силу других факторов, а именно благодаря исключению тех операндов трехадресной команды, которые служат для отсылки в главную память и затем выборки из нее промежуточных результатов. В одноадресной машине исключение этих бесполезных пересылок и связанных с ними операндов возможно всякий раз, когда результат одной операции используется непосредственно следующей за ней операцией, а в нуль-адресной машине с достаточно большой емкостью стека можно исключить практически все бесполезные пересылки и операнды. Если же на одноадресной машине промежуточное запоминание в главной памяти происходит, то оно требует двух команд (отсылка из аккумулятора в память и засылка в аккумулятор нового значения), т. е. сверх двух операндов, необходимых для промежуточного запоминания в трехадресной машине, требуются еще две операции.

Программы вычисления значений алгебраических выражений, содержащих не менее двух операций, на нуль-адресной машине получаются наиболее компактными, причем сокращение объема по сравнению с соответствующей программой для трехадресной машины тем значительнее, чем сложнее выражение. На одноадресной машине в зависимости от вида выражения объем программы может быть как меньше, так и больше объема соответствующей программы для трехадресной машины. Данные об объемах программы для вычисления значений некоторых выражений приведены в табл. 1. Кружок $\circ$ служит символом двухместной операции.

Приведенные данные свидетельствуют о безусловном превосходстве нуль-адресной машины в отношении компактности программ. Она во всех случаях превосходит одноадресную машину и уступает трехадресной машине в единственном случае, а именно когда реализуемая процедура в точности соответствует трехадресной команде, о чем уже говорилось.

Таблица 1

Структура выполняемой процедуры	Нуль-адресная машина			Одноадресная машина			Трехадресная машина		
	операций	операндов	всего	операций	операндов	всего	операций	операндов	всего
$z := a \circ b$	2	3	5	3	3	6	1	3	4
$z := a \circ b \circ c$	3	4	7	4	4	8	2	6	8
$z := a \circ b \circ c \circ d$	4	5	9	5	5	10	3	9	12
$z := (a \circ b) \circ (c \circ d)$	4	5	9	7	7	14	3	9	12
$z := (a \circ b) \circ (c \circ d) \circ e$	5	6	11	8	8	16	4	12	16
$z := (a \circ b) \circ (c \circ d) \circ (e \circ f)$	6	7	13	11	11	22	5	15	20

Более того, программы вычисления выражений на нуль-адресной машине являются в известном случае предельно компактными. Такая программа содержит ровно столько операций и операндов, сколько в реализуемой ею алгебраической формуле имеется вхождений знаков действий (включая знак «равно») и явно заданных объектов этих действий (значений или имен чисел). Дальнейшее сокращение объема программы можно получить путем усложнения машинных операций таким образом, чтобы одна машинная операция соответствовала более чем одному действию в формуле. Примером операций этого рода служат операции трехадресных команд, каждая из которых соответствует обычно двум операциям — арифметическому действию и присваиванию полученного результата некоторой переменной. В двухадресной машине применение подобных комбинированных операций в сочетании с аккумулятором и умолчанием операндов обеспечивает даже большую компактность программ, чем в случае нуль-адресной машины без комбинированных операций, вследствие чего двухадресная система нередко расценивается как наилучшая. Однако сокращение объема программы путем комбинированных операций связано с многократным увеличением количества вариантов каждой основной операции — известный недостаток двухадресной машины. Комбинированные операции с не меньшим успехом применимы, конечно, и в нуль-адресной машине. Например, введение единственного вида комбинации арифметического действия с отсылкой результата в главную память обеспечивает в случае нуль-адресной машины не худшую компактность программ, чем использование в двухадресной машине всех возможных комбинаций арифметических действий с засылкой в аккумулятор и отсылками в главную память.

Предельная компактность программ нуль-адресной машины отвечает максимальному умолчанию в командах этой ма-

шины — напомним, что в любой ее команде умалчивается (подразумевается) все, за исключением одной операции или одного операнда.

2. Недостатки нуль-адресной машины

Компактность программ, свойственная нуль-адресной машине, является очевидным ее преимуществом, но не означает, что этот тип машины безусловно лучше других. Правда, язык нуль-адресной машины обладает, кроме того, другим не менее важным преимуществом — трансляция с машинно независимых языков на этот язык существенно проще, чем на языки других машин. Обычно этот язык, называемый польской инверсной записью (ПОЛИЗ) [5], используется в трансляторах на другие машинные языки в качестве промежуточного.

Но и в сумме указанные преимущества не обеспечили нуль-адресной машине заметного предпочтения перед другими типами машин. Поэтому резонно проанализировать недостатки, свойственные нуль-адресной машине, и рассмотреть возможные пути их преодоления.

Перечень недостатков нуль-адресной машины или доводов против принятия стековой архитектуры имеется в [6]. Впрочем, не все эти доводы, выдвинутые сторонниками многорегистровой двухадресной архитектуры, действительны — мы рассмотрим только те из них, которые не были опровергнуты в [3], а именно:

1. Производительность обусловлена быстрыми регистрами, а не способом их использования.

2. Стековая организация слишком ограничивает и требует много операций копирования и взаимообмена.

3. Стек обладает поддоном (продолжением), размещение которого в более медленной памяти вызывает потерю производительности.

Первый довод, по-видимому, верен, если иметь в виду только быстродействие машины, но в отношении компактности программ и простоты компиляции способ организации регистров является определяющим и стековая организация обладает существенными преимуществами.

Второй довод действителен, когда речь идет о неоднократно используемых значениях аргументов и промежуточных результатов. Возможности усовершенствования стековой машины в этой части будут рассмотрены в конце статьи.

Третий довод касается снижения производительности стековой машины, происходящего вследствие переполнения части стека, выполненной на быстрых регистрах. Возникающие в этом случае пересылки из регистровой части стека в главную память, а затем обратно в регистры действительно вызывают

заметное снижение скорости, но переполнение происходит не потому, что регистры организованы по принципу стека, а потому что число их недостаточно велико. В многорегистровых машинах нестековой архитектуры ситуация, в которой для продолжения процесса приходится временно отсылать содержимое некоторых регистров в главную память, вовсе не исключена. При этом если в стековой машине продолжение стека в главной памяти несложно автоматизировать, умолчав о нем в программе, то в обычных машинах разгрузку и загрузку регистров, как правило, приходится выполнять при помощи команд.

Впрочем, недостатки нестековых машин не уменьшают недостатков стековой машины. Чтобы оценить и по возможности предотвратить вредный эффект переполнения регистровой части стека, мы займемся теперь выяснением того, насколько сильна опасность переполнения регистровой части, содержащей ограниченное число регистров, и каковы возможности уменьшения этой опасности.

Представленная в ПОЛИЗ программа нуль-адресной машины — это последовательность операндов (будем обозначать их буквами) и операций (ограничимся для простоты арифметическими операциями, имея в виду, что вообще операция может быть любой: пересылка, переход, обращение к подпрограмме и т. п.). Будем записывать указанную последовательность в строку, предполагая, что машина читает ее слева направо и воспринимает каждый символ как команду. Команда-операнд вызывает засылку значения соответствующего операнда в стек, а команда-операция — выполнение указанного ею действия над одним или двумя верхними значениями из стека с отправкой результата в стек.

Например, последовательность $ab+$ означает: «заслать a», «заслать b», «извлечь из стека два последних значения, получить их сумму и поместить эту сумму в стек».

Вычисление суммы трех слагаемых можно осуществить в двух вариантах: 1) $ab+c+$, 2) $abc++$. Первый вариант, называемый раннеоператорной ПОЛИЗ [7], требует двухрегистрового стека: сначала в стеке оказываются слагаемые a, b, затем их замещает сумма $(a+b)$, далее имеем $(a+b)$, c и наконец $(a+b+c)$.

Второй вариант (позднеоператорная ПОЛИЗ) требует трехрегистрового стека, ибо сначала идет засылка трех слагаемых, а затем получаем: a, $(b+c)$ и наконец $(a+b+c)$.

Раннеоператорная ПОЛИЗ заслуживает предпочтения, так как требует меньшей длины стека, чем позднеоператорная. Оба эти варианта ПОЛИЗ получаются из обычной в алгебре (инфиксной) записи перемещением знаков операций при неизменной последовательности операндов [7]. В раннеоператорном варианте каждая операция занимает самую левую из

возможных для нее позиций, а в позднеоператорном — самую правую.

Раннеоператорная ПОЛИЗ все же не снимает проблемы переполнения стека. Вычисление характерных для практики алгебраических выражений и в раннеоператорной записи также нередко требует стека значительной длины.

Однако применение раннеоператорной ПОЛИЗ не исчерпывает возможности уменьшения необходимой длины стека. Покажем это на примере программирования выражения $a+b+c\times(d-e)$. В позднеоператорной ПОЛИЗ оно имеет вид $abcde-\times++$ и требует 5 позиций стека. В раннеоператорной ПОЛИЗ имеем $ab+cde-\times+$ и соответственно 4 позиции. Кроме того, возможна запись $ab+de-c\times+$, для реализации которой требуется 3 позиции стека и наконец $de-c\times b+a+$, обходящаяся двумя позициями.

Последний вариант записи, обеспечивающий максимальную экономию необходимой длины стека, мы будем называть *совершенной* ПОЛИЗ. Она является инверсным вариантом *совершенной бесскобочной записи* (СБЗ), введенной нами в [8]. Рассматриваемое выражение в СБЗ имеет вид $-ed\times c++b+a$.

Перевод выражения, записанного в СБЗ, в совершенную ПОЛИЗ производится переносом каждого операнда из положения справа от знака операции в симметричное положение слева от этого знака. Перевод из совершенной ПОЛИЗ в СБЗ совершается путем обратного переноса.

Существенным признаком СБЗ и совершенной ПОЛИЗ является то, что в них все операнды максимально приближены к своим операциям. Благодаря этому исключены преждевременные засылки операндов в стек — каждый операнд засылается непосредственно перед использованием его операцией. При реализации совершенной записи выражений с одноместными и двухместными операциями в стеке не может оказаться одновременно более двух засланных программой операндов.

Таким образом, совершенная ПОЛИЗ радикально уменьшает опасность переполнения стека. Практически переполнение может возникать лишь в результате накопления в стеке значительного количества промежуточных результатов, что в действительности случается не часто. По-видимому, наиболее неблагоприятным в этом отношении является вычисление выражений вида $(((...)\circ(...))\circ((...)\circ(...)))$. Нетрудно подсчитать, что максимальное число одновременно находящихся в стеке промежуточных результатов при вычислении такого выражения равно числу заключений в скобки каждой из самых внутренних бесскобочных его частей. Например, для выражения $((a+b)\times(c+d))$ это число равно 2. При этом

волное число регистров стека, необходимых с учетом операндов, засылаемых программой, на 1 больше, т. е. для данного примера равно 3. Заметим, что для выражений рассматриваемого вида как раннеоператорная, так и совершенная ПОЛИЗ совпадает с позднеоператорной, которая для этих выражений, к счастью, оказывается достаточно экономной.

Экономия необходимой длины стека, обеспечиваемая совершенной ПОЛИЗ, достигается за счет перемещения операндов, т. е. основана на коммутативности операций. Для реализации аналогичной экономии в случае некоммутативных операций необходимы [8] оба варианта некоммутативных операций — левый и правый (имеются в виду двухместные операции). В машине, не располагающей обоими вариантами некоммутативных операций, можно воспользоваться *почти совершенной* записью [9], которая отличается от совершенной тем, что допускает преждевременные засылки в стек в тех случаях, в которых они неизбежны из-за отсутствия второго варианта некоммутативных операций.

В связи с использованием совершенной или почти совершенной форм записи естественно возникает вопрос о том, насколько сложна трансляция в эти формы. Разработка соответствующих алгоритмов [9, 10] показывает, что они незначительно сложнее алгоритмов трансляции в раннеоператорную ПОЛИЗ.

3. Машины с изменяемой адресностью команд

В цитированном выше перечне недостатков нуль-адресной машины упущен недостаток, заключающийся в том, что засылка операндов из главной памяти в стек отделена от использующих эти операнды операций. По сравнению с обычной машиной, отрабатывающей адресную и операционную части команды впараллель, это снижает скорость обработки и, следовательно, производительность машины. В этом отношении признается выгодным даже использование стека в сочетании с одноадресными и двухадресными командами [3], при этом утрачиваются практически все преимущества, свойственные нуль-адресной машине.

Имеется, однако, возможность так устранить указанный недостаток, что при полном сохранении всех достоинств нуль-адресной машины к ним прибавятся положительные свойства машин других типов. Получаемая путем такого усовершенствования машина [11], характеризуется переменной, а точнее — автоматически изменяемой адресностью команд. В зависимости от конкретного положения каждой содержащейся в программе операции эта операция воспринимается и выполняется машиной в составе команды с оптимальной для данного положения адресностью.

В реальной машине наибольшая допустимая адресность команды, конечно, ограничена и, по-видимому, не будет превосходить 3. Нуль-адресная машина представляет собой частный случай с предельно ограниченной адресностью.

Независимо от адресности, допустимой для данной конкретной машины, все машины этого типа работают с одной и той же программой, представленной в виде одного из вариантов ПОЛИЗ. Однако на машинах с разной допустимой адресностью выполняется эта программа в техническом отношении по-разному. Машины с ненулевой максимальной адресностью, в отличие от нуль-адресной машины, интерпретируют содержащиеся в программе операции и операнды в зависимости от контекста и допустимой для данной машины адресности, составляя из них *виртуальные* команды той или иной длины.

Общее правило образования виртуальных команд таково. Каждая операция присоединяет к себе непосредственно предшествующие ей операнды (если они имеются) в количестве, не превышающем ее арности и допустимой в данной машине адресности. Неприсоединенные операнды рассматриваются как команды засылки в стек. Относительно операций, присоединивших к себе меньше операндов, чем требуется в соответствии с их арностью, подразумевается, что недостающие их операнды находятся в стеке. В частности, операция не присоединившая ни одного операнда, относится только к операндам, извлекаемым из стека, как в нуль-адресной машине.

Поясним это правило на примере выражения $a/(b+c+d)$. В раннеоператорной и в совершенной ПОЛИЗ оно имеет вид соответственно $abc+d+/$ и $bc+d+a\backslash$, где $\backslash$ — символ левого деления: $x\backslash y=y/x$.

Обозначим вершину стека буквой t, подвершину — буквой s. Последовательности виртуальных команд и их интерпретация на машинах с допустимой адресностью 1 и 2 при раннеоператорной и совершенной формах записи программы будут следующими.

Виртуальные команды	Интерпретация виртуальных команд

а) раннеоператорная ПОЛИЗ, адресность 1:

a	$t:=a;$
b	$s:=t;\ \ t:=b;$
$c+$	$t:=t+c;$
$d+$	$t:=t+d;$
$/$	$t:=s/t;$

б) раннеоператорная ПОЛИЗ, адресность 2:

a	$t:=a;$
$bc+$	$s:=t;\ \ t:=b+c;$
$d+$	$t:=t+d;$
$/$	$t:=s/t;$

в) совершенная ПОЛИЗ, адресность 1:

b	$t:=b;$
$c+$	$t:=t+c;$
$d+$	$t:=t+d;$
$a\setminus$	$t:=t\setminus a;$

г) совершенная ПОЛИЗ, адресность 2:

$bc+$	$t:=b+c;$
$d+$	$t:=t+d;$
$a\setminus$	$t:=t\setminus a;$

С учетом того, что на нуль-адресной машине каждая из двух используемых в примере записей составит 7 команд, причем реализация раннеоператорной записи потребует 3, а совершенной — 2 ячейки стека, получаем сравнительную характеристику рассмотренных вариантов в виде табл. 2.

Таблица 2

Вид записи	Адресность	Число виртуальных команд	Длина стека
Раннеоператорная	0	7	3
	1	5	2
	2	4	2
Совершенная	0	7	2
	1	4	1
	2	3	1

Эта таблица показывает, что путем увеличения допустимой адресности можно ощутимо сократить количество виртуальных команд и несколько уменьшить необходимую длину стека. Надо иметь в виду, что данная таблица относится к частному примеру и поэтому может характеризовать общую тенденцию лишь в той мере, в какой этот пример типичен. Нетрудно указать примеры, в которых, скажем, адресность 2 не дает никакого выигрыша по сравнению с адресностью 1. Однако на примерах произвольного вида число виртуальных команд в среднем будет, конечно, существенно меньшим в случае адресности 2.

Наибольший выигрыш получается при переходе от нуль-адресной машины к машине с адресностью 1: резко сокращается длина программы в виртуальных командах и почти всегда укорачивается на 1 необходимая длина стека. Переход от допустимой адресности 1 к 2 обеспечивает дальнейшее сокращение длины программы и, следовательно, повышение быстродействия машины, но не сокращает необходимой длины стека, если арность операций не более двух.

Машина с допустимой адресностью 3 может быть оправдана лишь при условии, что в числе выполняемых ею операций имеются операции, арность которых больше двух. Такие операции могут представлять собой, например, комбинацию арифметического действия с отсылкой результата в главную память или комбинацию двух арифметических действий.

В техническом отношении машины с различной адресностью различаются прежде всего длиной регистра команды. В нуль-адресной машине этот регистр достаточно иметь состоящим из одной секции, в которой единовременно может находиться либо операнд, либо операция.

В машине с допустимой адресностью 1 регистр команд двухсекционный, и при последовательном продвижении программы через эти две секции возможны четыре сочетания их содержимого: 1) операнд, операция, 2) операнд, операнд, 3) операция, операнд, 4) операция, операция.

В случае 1) машина выполняет операцию, используя имеющийся перед ней операнд. Если операция одноместная, то она производится над этим операндом и результат засылается в стек. Если операция двухместная, то второй операнд извлекается из стека.

В случае 2) содержимое левой секции интерпретируется как команда засылки в стек, после выполнения которой в левую секцию передается содержимое правой, а в правую принимается очередной элемент программы.

В случаях 3) и 4) операция, указанная в левой секции, выполняется над операндами, извлекаемыми из стека, как в нуль-адресной машине. Затем в левую секцию передается содержимое правой, а в правую принимается очередной элемент программы.

Логика функционирования машин с большей адресностью, естественно, сложнее, но подчинена единому правилу: если в самой левой секции оказалась операция, то соответствующая команда выполняется как нуль-адресная; если одна или несколько секций, начиная с самой левой, содержат операнды и затем имеется секция с операцией, то данная операция выполняется с использованием находящихся перед ней операндов, получая недостающие операнды из стека; если же во всех секциях регистра команды находятся только операнды, то самый левый из них используется для засылки в стек, а затем остальные сдвигаются на одно место влево и в освободившуюся справа секцию принимается очередной элемент программы.

Преимущества машины с изменяемой адресностью очевидны. Можно сказать, что машина с допустимой адресностью n обладает возможностями нуль-, одно-, ... и n-адресной машин. При этом набор ее команд-операций остается столь

же компактным, как у нуль-адресной машины, а все разнообразие виртуальных команд порождается автоматически в процессе выполнения программы и к программисту или компилятору не имеет никакого отношения. Задача программиста (компилятора) — представить программу в виде совершенной ПОЛИЗ, а каждая машина проинтерпретирует встречающиеся в этой программе умолчания операндов и операций по предписанным ей правилам.

Может показаться, что эти преимущества получены за счет уменьшения быстродействия машины, так как выборка виртуальной команды, как правило, означает выборку нескольких команд-операндов и команд-операций. Но на самом деле скорость выборки программы определяется, как и в обычных машинах, быстродействием главной памяти: буфер, не длиннее регистра команд, обеспечивает возможность выборки виртуальными командами максимальной длины.

Нам остается показать, как может быть устранен недостаток стековой машины, заключающийся в том, что при использовании стека требуется много копирований и взаимообменов. Следует сказать, что в стековой машине с допустимой адресностью n копирований и взаимообменов не больше, а значительно меньше, чем в обычной машине. Речь может идти лишь о том, что по сравнению с машиной, снабженной собственной памятью процессора в виде набора быстрых регистров, многократное использование одного и того же значения в стековой машине затруднено, так как связано с повторной выборкой этого значения из главной памяти. Нет, однако, принципиальных возражений против того, чтобы в стековой машине воспользоваться тем же приемом, который применен в многорегистровой машине, т. е. засылкой многократно используемых значений в регистры с быстрым доступом. Ссылки на эти регистры в стековой машине естественно сделать безадресными, включив функцию выбора нужного регистра в содержание команды-операции. Это равносильно введению составных операций, комбинирующих действие с именем операнда. Например, при наличии в машине двух регистров, обозначенных p и q, возможны одноместные команды вида «прибавить p», «прибавить q», «вычесть p» и т. п., задающие двухместные операции над содержимым вершины стека и указанного в названии операции регистра.

ЛИТЕРАТУРА

1. Alt F. L. Electronic digital computers. New York, Academic Press, 1958.
2. Hager K. Die Bewertung einiger Rechnerkerntypen für das Verarbeiten von arithmetischen Ausdrücken. «Elektron. Rechenanlag», 13, № 6 241—249, 1971.

3. Bell G., Cady R., McFarland H., Delagi B., O'Laughlin J., Noonan R. A new architecture for mini-computers. — The DEC PDP-11. «AFIPS SJCC Proc.», 657—675, 1970.
4. Брусенцов Н. П., Жоголев Е. А. Структура и алгоритм функционирования малой вычислительной машины. В сб.: «Вычислительная техника и вопросы кибернетики», вып. 8. Изд-во ЛГУ, 1971.
5. Жоголев Е. А. Интерпретатор ПОЛИЗ-63. ЖВМ и МФ, **5**, № 1, 1965.
6. Amdahl G. M., Blaaum G. A., Brooks F. P. Architecture of the IBM System/360. «IBM Journ. Research and Development», **8**, № 2, 87—101, 1964.
7. Hamblin C. L. Translation to and from Polish notation. «Computer Journ.», **5**, № 3, 210—213, 1962.
8. Брусенцов Н. П. Усовершенствованная бесскобочная запись формул. ЖВМ и МФ, **12**, № 3, 1972.
9. Брусенцов Н. П., Руднева Т. Л. Алгоритм трансляции алгебраических выражений в почти совершенную бесскобочную запись. В сб.: «Вычислительная техника и вопросы кибернетики», вып. 9. Изд-во МГУ, 1972.
10. Брусенцов Н. П., Руднева Т. Л. Алгоритм трансляции алгебраических выражений в совершенную бесскобочную запись. В сб.: «Вычислительная техника и вопросы кибернетики», вып. 11. Изд-во МГУ, 1974.
11. Брусенцов Н. П. Устройство управления цифровой вычислительной машины с магазинной памятью. Авторское свидетельство № 391562, кл. G 06f 9/10. Опубликовано 25.7.73. Бюллетень № 31.