АКАДЕМИЯ НАУК ЭСТОНСКОЙ ССР Институт кибернетики

ПРОГРАММИРОВАНИЕ МИКРОПРОЦЕССОРНОЙ ТЕХНИКИ

(сборник трудов целевой подгруппы по технологии программирования микропроцессорной техники)

Таллин 1984

УДК 681.032

Ю. Матиясевич, А. Терехов 16-РАЗРЯДНАЯ ВИРТУАЛЬНАЯ ЭВМ, ОРИЕНТИРОВАННАЯ НА АЯВУ

В настоящее время получили большое распространение однобайтовые микро-ЭВМ с архитектурой и системой команд микропроцессора 6502, например, ЭВМ APPLE. Выпуск таких ЭВМ планируется и у нас в стране. Большая часть известного матобеспечения микро-ЭВМ опирается на использование гибких дисков. В состав матобеспечения для более простых комплектов микро-ЭВМ обычно входит транслятор с языка ассемблера и интерпретатор языка БЕЙСИК. Интерпретатор работает довольно медленно, да и сам язык БЕЙСИК имеет очень слабые "выразительные" возможности. Мы решили реализовать подмножество языка АЛГОЛ-68 в виде кросс-системы, действующей на ЕС ЭВМ, с последующим переносом на микро-ЭВМ методом раскрутки.

В выбранное подмножество АЛГОЛа-68 (будем называть его А68А) были включены следующие типы данных: int , bits, ref m, char, bool , структуры, процедуры, одномерные и двухмерные массивы. В качестве основной единицы хранения было принято слово из двух байтов. Прямая трансляция двухбайтовых операций в коды однобайтовой микро-ЭВМ потребовала бы очень длинных объектных программ, что невыгодно для ЭВМ с быстродействием 500 000 оп/сек и памятью всего 64 К. Поэтому было решено использовать традиционный для ЭВМ такого класса путь создания интерпретатора 16-разрядной виртуальной машины. Оказалось, что априорная ориентация на программирование на алгоритмическом языке высокого уровня (в данном случае - на А68А) существенно повлияла на архитектуру и систему команд виртуальной машины. В некотором смысле данная работа родственна многим работам по аппаратной реализации АЯВУ.

Начнем со схемы управления памятью. В А68А текст процедуры разрешается вкладывать только в собственно программу, но не в другие тексты процедур, поэтому возможны всего два уровня локализации - локальный (в текущей процедуре) и глобальный (в собственно программе). Процедуры могут быть рекурсивными. В А68А есть глобальные генераторы, которые отводят области памяти, действительные во всей программе. Соответственно, вся память, доступная для распределения, состоит из двух участков, растущих навстречу друг другу: по возрастанию адресов - стек, по убыванию адресов - куча. При вызове процедуры в стеке захватывается область памяти, достаточная для размещения формальных параметров, объектов, описанных в процедуре, различной служебной информации. Обычно эта область памяти называется статикой, т.к. ее размер известен еще во время трансляции.

Статики вызванных процедур связаны в динамическую цепочку (нулевое слово каждой статики содержит адрес начала статики вызывающей процедуры). На статику собственно программы указывает служебный регистр виртуальной машины G Ø, на статику текущей процедуры - регистр L Ø, на левую границу кучи – HEAP Ø. Служебные регистры занимают по 2 байта в нулевой странице памяти микро-ЭВМ.

Длину каждой статики мы ограничиваем 256 байтами (чтобы смещение задавалось 1 байтом). Для элементов массивов это ограничение является слишком жестким, поэтому решено отводить память под элементы массивов в куче.

Значения представляются следующим образом: значения данных типа int, bits, ref m, char, bool и процедурные значения занимают по 1 слову, причем из-за особенностей архитектуры левый байт адреса является младшим, а правый - старшим. Например, единица кодируется как $Ø1ØØ, а десятичное 258 – как $Ø2Ø1. В значении вида bits разряды левого байта нумеруются с 9 по 16, а правого - с 1 по 8. Литера кодируется левым байтом слова, а логическое представляется целым числом 1 для true и 0 - для false . Процедурное значение представляется адресом входа. Такое значение можно передавать параметром, присваивать переменным, собирать в структуры и массивы и т.д.

Массив представляется паспортом (в статике) и элементами (в куче). Паспорт одномерного массива содержит C_0 , I , *u* , где C_0 - адрес "нулевого" элемента, I и u - границы. Паспорт двухмерного массива - C_0 , I_1, u_1, I_2, u_2. Здесь C_0 – адрес одномерного массива, содержащего "нулевые" элементы каждой строки матрицы (вектор Айлифа).

Структура представляется последовательностью полей.

Для сокращения объектного кода и облегчения трансляции с АЯВУ основные операции виртуальной машины выполняются над стеком, максимальная глубина которого фиксирована (скажем, 8 слов). Никаких команд по динамической откачке стека в память не предусмотрено: предполагается, что за переполнением стека следит транслятор. Стек расположен в нулевой странице микро ЭВМ (левая граница ST Ø). В процессе работы виртуальной машины на верхушку стека указывает регистр X микро-ЭВМ.

Большая часть команд состоит только из кода операции (1 байт), операнды берутся из стека, туда же помещается результат. Есть 3 команды загрузки в стек:

– загрузка из статики собственно программы (тип G);

– загрузка из статики текущей процедуры (тип L);

– загрузка непосредственная.

Загрузка непосредственная имеет двухбайтовый операнд - целое число, остальные загрузки имеют однобайтовый операнд - смещение от начала статики.

Аналогично устроены 2 команды выгрузки из стека (типа G и L).

К унарным операциям относятся ”-” и abs для int , not для bool и not для bits. Благодаря выбранному представлению значений не требуют специальных команд abs для char и bool, а также repr для получения литеры по целому коду.

Класс бинарных операций значительно шире:
- для int : +, -, *, % (деление нацело), mod;
- для bool и bits : and , or ;
- для bits : elem, shr, ohl (выделение элемента и сдвиги).

В виртуальной машине не предусмотрена автоматическая выработка различных признаков (нуль, минус, переполнение и т.д.) всеми командами. Чаще всего эти признаки не используются (они нужны только в условных переходах), а их выработка в процессе интерпретации стоит дорого. Более существенным аргументом является тот факт, что при наличии признаков резко усложняется программирование логических формул из-за различного представления логических значений в памяти и разрядах признаков. Для каждого типа отношений предусмотрена специальная команда, вырабатывающая на стеке I для true и Ø для false.
Итак, нужны команды =, /=, < , < = , > , > =. Для значений вида bits нужны особые команды <= и >= (теоретико-множественные включения).

В принципе, нужны всего две команды передачи управления – безусловная и по Ø в стеке. Для оптимизации программ типа if not выражение then … добавлена команда условной передачи управления по I в стеке.

В А68А существенную роль играют имена - адреса объектов. Имена идентификаторов, описанных в описании переменных, принято называть статическими, т.к. еще во время трансляции известен адрес именуемого значения, поэтому разыменование статического имени можно выполнить еще на этапе трансляции. Преобразование статического имени в нестатическое (например, при передаче параметром) состоит в сложении значения базового регистра (L Ø или G Ø) и смещения. Для нестатических имен, полученных, например, генераторами или вырезками из массивов, нужна специальная операция разыменования, которая берет с вершины стека адрес и помещает на его место значение слова памяти, расположенное по этому адресу. Присваивание нестатическому имени - однобайтовая команда, которая берет адрес и присваиваемое значение из стека. Присваивание статическим именам осуществляется обычными командами выгрузки из стека. Присваивание в А68А вырабатывает значение, поэтому возможны тексты типа a := b := c или (a := b)+c , и для команд присваивания определены парные команды, оставляющие значение в стеке.

Присваивание сложных значений (массивов и структур) невыгодно производить через стек, поэтому нужна команда копирования указанного числа слов. Каждый из операндов в присваивании a := b может иметь тип L или G или представляться адресом в стеке, в связи с чем возможны 9 вариантов копирования.

Многие конструкции А68А реализуются командами, специально предназначенными для этих конструкций:
- глобальный генератор (параметр - длина значения);
- оператор выбора (параметры: N и N + 1 - адрес передачи управления);
- цикл вида for i to n do (параметры i и n);
- цикл общего вида for i from A by B to c do (параметр - адрес области памяти, где хранятся значения i, B, C; там же хранится адрес перехода на начало цикла);
- статический вызов (параметр - адрес начала процедуры);
- нестатический вызов (параметров нет, т.к. адрес процедуры вычисляется в стеке).

Семантика этих команд полностью определяется техникой реализации соответствующих конструкций.

Заключение. Системы команд виртуальных машин принято определять самым общим образом (в целях расширения возможностей их применения). Нам кажется, что это ведет к неоправданным потерям с точки зрения эффективности и к трудностям реализации трансляторов. Использование алгоритмического языка высокого уровня заданного класса само по себе является достаточно общей платформой. Вряд ли целесообразны исследования часто повторяющихся действий и типичных процедур пользователя (зачастую именно так подбирают объекты для аппаратной реализации). Нужно эффективно реализовать основные конструкции алгоритмических языков, это действительно дает гарантию, что усилия и затраты не пропадут зря. Архитектура виртуальной ЭВМ при этом получается достаточно специализированной (например, фиксирован способ

распределения памяти, представление массивов, способы адресации), появляются многие необычные команды, а многие традиционные черты отпадают за ненадобностью (например, разряды кода условия, индексные регистры; аппаратная защита может быть заменена защитой при трансляции и т.д.).